TSCH Networks for Health IoT: Design, Evaluation and Trials in the Wild. ACM Transactions on Internet of Things

Elsts, Atis; Fafoutis, Xenofon; Oikonomou, George; Piechocki, Robert J.; Craddock, Ian

Published in:
ACM Transactions on Internet of Things

Link to article, DOI:
10.1145/3366617

Publication date:
2020

Document Version
Peer reviewed version

TSCH Networks for Health IoT: Design, Evaluation and Trials in the Wild

ATIS ELSTS, Institute of Electronics and Computer Science (EDI), Latvia
XENOFON FAFOUTIS, Technical University of Denmark, Denmark
GEORGE OIKONOMOU, University of Bristol, UK
ROBERT PIECHOCKI, University of Bristol, UK
IAN CRADDOCK, University of Bristol, UK

The emerging Internet of Things (IoT) has the potential to solve major societal challenges associated with healthcare provision. Low-power wireless protocols for residential Health IoT applications are characterized by high reliability requirements, the need for energy-efficient operation, and the need to operate robustly in diverse environments in presence of external interference. We enhance and experimentally evaluate the Time-Slotted Channel Hopping (TSCH) protocol from the IEEE 802.15.4 standard to address these challenges. Our contributions are a new schedule and an adaptive channel selection mechanism to increase the performance of TSCH in this domain. Evaluation in a test house shows that the enhanced system is suitable for our e-Health application and compares favorably with state-of-the-art options. The schedule provides higher reliability compared with the Minimal Scheduling Function from the IETF 6tisch Working Group, and has a better energy-efficiency/reliability tradeoff than the Orchestra scheduler. Results from 29 long-term residential deployments confirm the suitability for the application and show that the system is able to adapt and avoid channels used by WiFi. In these uncontrolled environments, the system achieves 99.96 % average reliability for networks that generate 7.5 packets per second on the average.

CCS Concepts: • **Networks** → **Link-layer protocols**; *Network experimentation*; • **Computer systems organization** → *Sensor networks*; Reliability.

Additional Key Words and Phrases: IEEE 802.15.4, Time Slotted Channel Hopping, Sensor Network Deployments, Internet of Things

ACM Reference Format:
Atis Elsts, Xenofon Fafoutis, George Oikonomou, Robert Piechocki, and Ian Craddock. 2019. TSCH Networks for Health IoT: Design, Evaluation and Trials in the Wild. *ACM Trans. Internet Things* 1, 1 (October 2019), 27 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Time Slotted Channel Hopping (TSCH) protocol for low-power wireless networks from the IEEE 802.15.4 standard is described as highly reliable, energy-efficient and predictable [13]. These properties make TSCH attractive for many Internet IoT applications, including industrial and automotive applications [5, 11], as well as e-Health applications in residential environments.

Authors' addresses: Atis Elsts, Institute of Electronics and Computer Science (EDI), Dzerbenes Street 14, Riga, LV-1016, Latvia, atis.elsts@edi.lv; Xenofon Fafoutis, Technical University of Denmark, Denmark, xefa@dtu.dk; George Oikonomou, University of Bristol, UK, g.oikonomou@bristol.ac.uk; Robert Piechocki, University of Bristol, UK, r.j.piechocki@bristol.ac.uk; Ian Craddock, University of Bristol, UK, ian.craddock@bristol.ac.uk.

2577-6207/2019/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

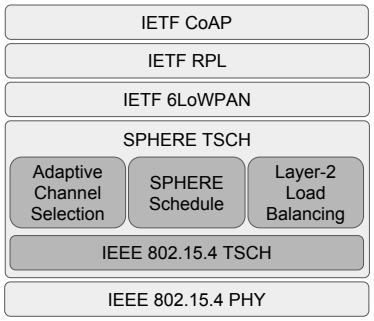


Fig. 1. Architectural overview of SPHERE TSCH.

However, few reports on real-world deployments of IEEE 802.15.4 TSCH are available in the research literature, especially on TSCH deployments at scale. More empirical results are required to demonstrate whether TSCH, as defined by the open standards, lives up to its promised properties, to identify weaknesses, to gather evidence for improvements, and to evaluate aspects that the IEEE 802.15.4 TSCH standard leaves to the implementers, such as scheduling.

This paper presents the design and evaluation of an IEEE 802.15.4 TSCH based network stack developed for the SPHERE (Sensor Platform for Healthcare in a Residential Environment) project [51]. As part of the SPHERE *100 Homes Study* it aims to install a healthcare monitoring system in homes of the general public. At the time of writing this, the system has been deployed in more than 30 homes. The SPHERE deployment context presents a unique opportunity to perform large-scale evaluation and experimentation with IEEE 802.15.4 TSCH in realistic, uncontrolled environments.

Compared with typical industrial monitoring [5, 11], environmental monitoring [48] and smart building [9] applications of TSCH, residential e-Health applications pose several new challenges:

- Health and behavior-related sensors – accelerometers, gyroscopes, electroencephalogram (EEG) and electrocardiogram (ECG) sensors – create data with a much higher rate compared with most environmental sensors: a higher-capacity network layer is required.
- Residential home networks have to deal with highly dynamic interference and fading. It is highly desirable to automatically adapt to changes in the radio propagation environment.

To address these challenges and to make the TSCH protocol more suitable for Health IoT applications, this paper contributes SPHERE TSCH (Fig. 1), a version of TSCH enhanced with:

- A schedule for the SPHERE application, including a load-balancing mechanism that increases the capacity and reliability of the TSCH network;
- An adaptive channel hopping mechanism to self-configure the network to avoid external interference.

SPHERE TSCH compares favorably with state-of-the-art techniques such as the Orchestra TSCH schedule [16], and the 6TiSCH (*IPv6 over TSCH*) Minimal Scheduling Function (MSF) [10] when used for the SPHERE application.

Moreover, we present a report of the performance of SPHERE TSCH in a large-scale real-world experimental evaluation. To the extent of authors' knowledge, such a large-scale evaluation of TSCH is not present in the literature yet.

This paper builds on our previous work in the area [19, 20]. Besides the report from the real-world trials, the main novelty of the present paper is the investigation of load balancing with multiple gateways in TSCH networks. Compared with the previous work, we also add complete descriptions

of the SPHERE scheduling algorithm and the adaptive channel selection algorithm, as well as describe a mechanism to reduce inter-network interference.

In terms of structure, Section 2 gives background information and motivates the selection of TSCH. Section 3 positions this paper among related work; Section 4 describes the design, implementation, and the evaluation of the SPHERE TSCH scheduler in a test house; Section 5 covers the adaptive channel selection. Section 6 reports the main results from the SPHERE *100 Homes Study*, Section 7 discusses the results and future work, and Section 8 provides concluding remarks.

2 BACKGROUND AND MOTIVATION

2.1 The SPHERE Application

The SPHERE project has developed an in-home sensing platform that aims to serve as a prototype for residential healthcare systems of the future [21, 51]. The main task of the project is to characterize long-term health-related behaviors in residential environments by using multiple sensing modalities and state-of-the-art machine learning techniques. In the context of low-power sensing and networking, the SPHERE platform has to provide this functionality [21]:

- Collect accelerometer data from wrist-worn wearable nodes, one per inhabitant.
- Provide an indoor localization service for each inhabitant, with better-than-room-level accuracy.
- Collect data about the home environment, such as temperature and light levels.
- Timestamp all sensor data with sub-second accuracy.

Unlike many other smart health platforms that are limited to laboratory environments, the SPHERE platform is deployed and evaluated in real-world houses of volunteering participants. It was first installed in the *SPHERE House* (Fig. 2): a residential property rented by the project for experimentation and testing. As part of the *100 Homes Study*, the SPHERE platform is now being deployed for up to 12 months in up to 100 volunteer homes in Bristol, UK.

While the functional requirements of the SPHERE application are fairly typical for wireless sensor networks, challenges arise from the facts that the project is public-facing, the system is deployed at scale, and the data collected from the system will be used for exploratory research:

- The data scientists in SPHERE require that the "raw" acceleration data are collected.
- The continuous indoor localization service requires that the whole home is constantly covered with wireless signal, not just selected rooms or areas.
- The networks must be able to cope with diverse residential environments in terms of the size of the deployment area, construction materials, and levels of external interference.
- The wireless medium in urban and sub-urban homes is unpredictable and dynamic.
- The number of deployments makes it impractical to calibrate the platform for each house.
- The target residential environments are private spaces of the participants, hence, frequent or lengthy maintenance visits are undesirable.
- Several thousand devices must be manufactured and deployed. Due to budget constraints, complex algorithms (e.g., cognitive radio) that require dedicated hardware are not an option.
- Last but not least, the wearable and environmental sensor nodes must be battery powered, hence low energy consumption is crucial.

The total packet rate of a SPHERE deployment is untypically high for low-power wireless networks. This is due to the fact that raw acceleration data must be collected, as specified the project's proposal, and requested by the project's data scientists – the future users of the data. Consequently, it is not possible to reduce the data size with lossy compression [29], in-network aggregation[27], or with other well-known techniques. There is no reason to believe that SPHERE is unique in this

aspect; major organizations such as the UK Medical Research Council (MRC) explicitly require [12] that the raw data is collected and stored in studies funded by them.

2.2 Low-Power Wireless Technologies

In this section we overview some of the standards considered for the SPHERE application. We prefer to use a standardized protocol instead of developing a custom one, aiming for reduced development time, better technology transfer potential, and integration of third-party components in the future.

2.2.1 IEEE 802.15.4. The IEEE 802.15.4 standard describes a collection of physical (PHY) layer and medium access control (MAC) layer specifications for low-power wireless networks [6]. The IEEE 802.15.4 standard was significantly extended in January, 2016, when a new version of the document was published incorporating multiple new MAC protocols, including TSCH (Section 2.5).

2.2.2 ZigBee. ZigBee [52] is a specification of high-level communication protocols on top of IEEE 802.15.4. It provides mesh network functionality for low-power wireless networks. ZigBee is not an open standard and requires membership in the ZigBee alliance; it also is not compatible with the IP network stack. ZigBee uses a single channel MAC layer, which is not sufficient to achieve high reliability in challenging environments [18].

2.2.3 6LoWPAN and TSCH. 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) [4] is a set of open standards aims to solve some of the above-mentioned shortcomings of ZigBee. 6LoWPAN provides an energy-efficient way to transfer IP packets in low-power networks. It does that through IP header compression. 6LoWPAN also includes specifications for routing, thus making it possible to extend the range of the network by joining the devices in a mesh.

The TSCH protocol (see also Section 2.5) is a Time Division Multiple Access (TDMA) MAC protocol for low-power wireless networks. It is defined in the IEEE 802.15.4-2015 standard [6], and is compatible with IEEE 802.15.4 physical layer, as well as with 6LoWPAN network and routing layers. The TSCH protocol has roots in industrial monitoring applications: first known as TSMP (Time-Synchronized Mesh Protocol) [39], it was included in the WirelessHart [11] and ISA.100.11a [5] (International Society of Automation) standards aimed for industrial networks. It was one of the first protocols that reported wire-like reliability in low-power and lossy multihop networks [39]. The IETF (Internet Engineering Task Force) 6TiSCH Working Group tackles the problem of integration of the 6LoWPAN standard with the IEEE 802.15.4-2015 TSCH, and has produced several RFC and internet drafts [10, 46, 47].

2.2.4 Bluetooth Low Energy and Bluetooth Mesh. Bluetooth Low Energy (BLE) is a wireless personal area technology aimed to provide similar range and functionality as Bluetooth, but with reduced power consumption [8]. The PHY layer of BLE is more efficient [40] than the PHY layer of IEEE 802.15.4 and has higher data rate. However, the communication range of BLE is shorter, and

Table 1. Representative examples of low-power wireless hardware

Protocol	**6LoWPAN**	**BLE**	**WiFi** Low Energy
Example chip	CC2650[1]		CC3200[2]
Data rate	0.25 Mbps	1 Mbps	1–54 Mbps
Tx current @ 0 dBm	9.1 mA		≈ 250 mA
Rx current	5.9 mA		≈ 50 mA
Rx sensitivity	−100 dBm	−97 dBm	−95 to −74 dBm

Table 2. Comparison of low-power wireless options

Protocol	**ZigBee**	**TSCH / 6LoWPAN**	**BLE**	**WiFi**
Very low-power hardware	+	+	+	−
Duty cycling in standard	+	+	+	−
High reliability	−	+	−	+
TCP/IP	−	+	−	+
High datarates	−	−	−	+
Low Tx power	+	+	+	−
Meshing support	+	+	+/−	−/+
Selected for env. sensors		+		
Selected for wearables			+	
Selected for backbone		+		

the protocol is more vulnerable to bit errors, as it has less redundancy in the physical layer than IEEE 802.15.4. Same as ZigBee, BLE is not an open standard, and is not out-of-the-box compatible with IP.

BLE 4.1 was the first version of BLE specification that allowed a device to be slave and master in the same. It was released in 2013, and this feature made the idea of a BLE mesh mode possible. In 2017, when SPHERE's architecture was already finalized, the BLE standards finally added support for mesh networks via Bluetooth Mesh [7]. The Bluetooth Mesh 1.0 standard uses connectionless mode and distributes information on top of BLE advertisements. This does not match the SPHERE's reliability requirements, as BLE advertisements are not collision-free; reliability is only supported via packet repetition (flooding). In addition, BLE advertisements are not acknowledged and have no retransmissions. Moreover, BLE advertisements do not channel hop; instead they use repetition in the three advertisement channels. Meshing on top of the connection-oriented BLE mode is also possible, although it is not standardized. However, Murillo *et al.* [36] compare the two modes and conclude that they have similar performance in terms of reliability.

2.2.5 WiFi. WiFi is a technology for wireless local area networking based on the IEEE 802.11 standards. These standards are primarily optimized towards high data rates, not energy-efficient operation. However, even though duty-cycling is not part of the original IEEE 802.11 (WiFi) standards, there are customized solutions that offer some form of saving energy, for example the Texas Instruments CC3200 [2] chip (Table 1). Nevertheless, it is challenging to do that while remaining compliant with the IEEE 802.11 standard requirements for transmit power, beacon interval, and data rates [3]. There are options for meshing WiFi devices; however, they are not widely used.

2.3 Design Choices

While the focus of this paper is on SPHERE's TSCH network, for the sake of completeness, this section discusses some of the design choices (Table 2) behind the SPHERE architecture.

(1) To collect data from the Wearable Sensor nodes, SPHERE uses BLE as the protocol with the least required amount of energy for a data byte transmission. On the average, multiple mains-powered infrastructure devices are expected to pick up each transmission, thus minimizing the problems from the reduced range and bit errors.

(2) To collect data from the Environmental Sensors, we use IEEE 802.15.4. Each Environmental Sensor transmits data to a single infrastructure device, dynamically selected through the 6LoWPAN routing protocol. We leverage the high reliability of TSCH to achieve good packet delivery reliability in this setup.
(3) To transfer data from the infrastructure devices to a central gateway, we also use TSCH on top of IEEE 802.15.4, and connect the devices in a mesh.

For the first two uses, WiFi would be too energy inefficient as compared with BLE and IEEE 802.15.4 (Table 1). For the third point above, WiFi is a reasonable option, however, we opted to use IEEE 802.15.4 due the following reasons: first, we wanted to reduce the total number of protocols used. Second, the transmission power levels of WiFi (14–17 dBm typical) are much higher than those of IEEE 802.15.4 (0–5 dBm). Using WiFi for data transmission could interfere the residents' own wireless networks, and, for some individuals, may lead to perceived health concerns. Taking this into account, we selected 6LoWPAN, even though this meant operating close to its data rate limits.

When compared with BLE, 6LoWPAN has the advantages of being an open standard, and being more flexible in its operating modes. When TSCH is used in conjunction with 6LoWPAN, it adds high-accuracy global time synchronization to the network, thus solving the problem of getting sub-second accuracy timestamps for all sensor data.

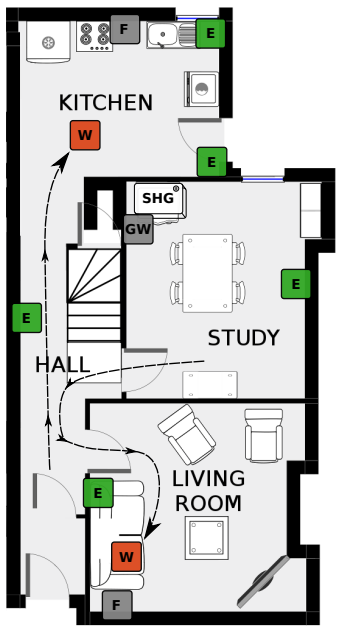


Fig. 2. **SPHERE device layout in the first floor of the SPHERE House.** Mains-powered forwarding nodes (*F*) form a backbone TSCH network and collect data from the battery-powered Environmental Sensors (*E*). In addition, *F* nodes are equipped with BLE radio for collecting the data broadcast by mobile Wearable Sensors (*W*). The central SPHERE IoT Gateway node (*GW*) collects the data from the network and sends it to the SPHERE Home Gateway (*SHG*) computer over USB.

2.4 The SPHERE Devices

Three bespoke IoT platforms are designed for the SPHERE system [23]. The *SPHERE Environmental Sensors* are responsible for capturing the environmental properties of a house at room-level granularity. More specifically, the SPHERE Environmental Sensors incorporate temperature, humidity, light, and air pressure sensors, as well as a Passive Infrared (PIR) sensor for capturing motion [24]. In addition, a custom water flow sensor [45] is integrated with the environmental sensor of the kitchen in order to detect water flow in kitchen taps. The SPHERE Environmental Sensor is based on the CC2650 System-on-Chip (SoC) for processing and wireless communication at 2.4 GHz.

The *SPHERE Wearable Sensors* are wrist-worn mobile sensor nodes. They are equipped with a low-power accelerometer, and capture the activity patterns of the residents of the house [26]. These devices continuously measure 3-axis acceleration at 12.5 or 25 Hz; a sampling frequency of at least 20 Hz is required for accurate activity recognition [44]. The SPHERE Wearable Sensors are also based on the CC2650 SoC, and communicate the sensor data to the SPHERE home infrastructure over undirected non-connectable BLE advertisements.

The third bespoke platform of SPHERE is named *SPHERE Gateway*. The data from the Wearable and Environmental Sensors are collected over a backbone IEEE 802.15.4 TSCH network of SPHERE Gateways, acting as forwarders. These mains-powered devices have two CC2650 SoCs, each of which has a separate operating system. One of them is running BLE, the other 6LoWPAN over IEEE 802.15.4 TSCH. Moreover, these devices incorporate temperature, humidity and air pressure sensors. The BLE side of a SPHERE Gateway is responsible for receiving BLE advertisements from the Wearable Sensors and passing them to the 6LoWPAN side over SPI. The 6LoWPAN side is responsible for forwarding that data over the TSCH network. The SPHERE Environmental Sensors are directly connected to the TSCH network as leaf nodes, i.e., they do not forward traffic on behalf of other nodes. The root SPHERE Gateway is connected to the SPHERE Home Gateway – a Linux server that collects and stores the data.

2.5 Introduction to Time Slotted Channel Hopping

Time in TSCH networks is divided in *slots*. All nodes in a TSCH network synchronize their time to the network coordinator node to ensure that the timeslot with the same number starts and ends at the same time on each node. The actions of a node in a specific slot are determined by the *schedule* of that node. The schedule defines slots for transmission (for data and control packets), slots for reception, and idle slots, which allow the nodes to save energy. The schedule also determines which *channel offset* to use for communication. A channel offset is translated into an IEEE 802.15.4 channel using the TSCH *channel hopping sequence*; all nodes in a TSCH network share the same hopping sequence. Given a specific timeslot with a fixed Absolute Sequence Number (ASN), the translation is done as follows:

$$channel = HS\big[(ASN + \text{channelOffset}) \textbf{ mod } ||HS||\big],$$

where HS is the hopping sequence and $||HS||$ is its length.

The pseudorandom channel hopping allows TSCH to be resilient against external interference and frequency-selective multipath fading. If a transmission of a packet fails, with high probability it is retransmitted on a different channel. In this way, the network remains highly reliable even if several bad-quality channels are present.

In TSCH networks, *enhanced beacon* (EB) messages are periodically sent out for network maintenance purposes. These EB messages allow new nodes to join an existing network, and already joined nodes to update their configuration parameters. Network configuration is distributed in *information elements* (IE) within EB messages: for example, there is an IE for distributing the hopping sequence. In this way, TSCH networks are able to adapt to changes.

3 RELATED WORK

3.1 TSCH in Deployments

The behaviour and challenges faced by TSCH networks that are deployed "in the wild" have received relatively little attention from the research community. Most of the few existing research papers report on SmartMesh: a commercial, closed-source implementation of a TSCH-based network stack from Linear Technology [9, 48] or its precursor from Dust Networks [39]. There are no reports on the open protocols defined in IEEE 802.15.4 and IETF 6TiSCH standards, with one exception: the eCare@Home system [43]. This system uses the open-source Contiki TSCH stack, similarly to SPHERE. However, the eCare@Home project limits itself to a controlled environment in a single house. The published SmartMesh studies similarly report results from a single, low-datarate network.

3.2 Scheduling for TSCH

The IEEE 802.15.4 TSCH standard [6] does not impose a specific schedule, offloading the responsibility to select the schedule to the implementer. As a result, there have been both closed commercial implementations as well as a lot of academic research on scheduling for TSCH. The IETF 6TiSCH Working Group is taking a more comprehensive approach and has published RFC on the minimal 6TiSCH schedule [46], as well as the 6top mechanism for distributed scheduling [47].

The existing TSCH schedulers are surveyed by Hermeto *et al.* [30] and by Kharb *et al.* [32]. However, most of the existing research (see e.g., [37, 38, 42] for some highlights) are not immediately suitable for SPHERE. Some typical problems are: (1) being at too low technology readiness level and lacking details necessary for an implementation; (2) having too high run-time complexity for implementation on embedded devices; (3) insufficient support for requirements critical to SPHERE, such as high schedule capacity and freedom from collisions; (4) focus on requirements not relevant to SPHERE, such as low latency.

The commercial SmartMesh implementation of TSCH by Linear Technology [49] is a highly reliable centralized scheduler. Furthermore, its performance in real-world deployments is published [9, 48]. The main conceptual drawback of the SmartMesh network stack and scheduler is its closed nature. We aim to perform academic research that builds on top of existing open standards (such as IETF RPL and IEEE TSCH) instead of a completely closed routing protocol and network scheduling algorithm.

As the two state-of-the-art options for the experimental evaluation in this paper (Section 4.6), we selected Orchestra [16], and versions 2 and 4 of the 6TiSCH Minimal Scheduling Function (MSF) [10]. Orchestra is an autonomous scheduling algorithm that achieves high reliability in harsh environments, and requires no additional traffic to set up the schedule. However, Orchestra is not suitable for heterogeneous networks where the devices have different data rates. Moreover, it is not collision free and not a good fit for high-rate applications, as it allocates a single slot for each neighbor, puts all unicast cells at the same channel offset, and cannot dynamically resolve slot conflicts. The 6top mechanism and the 6TiSCH MSF allows to dynamically allocate cells based on traffic requirements. However, while the 6top mechanism itself is standardized in the RFC 8480 [47], the MSF is still in a draft stage, and by its very nature, the runtime behavior of 6top scheduling is less predictable compared with the static schedule used in SPHERE.

3.3 Adaptive Channel Selection

Our goal in this paper is not to introduce a novel adaptive channel selection (ACS) method, but rather to make an informed choice when designing our solution for the deployments, and report

back the real-world evaluation results. Hence, we briefly overview existing work in the area, focusing on practicality more than on optimality.

One of the first published ACS methods for TSCH is ATSCH [14] (Adaptive TSCH). It is a local method that uses noise RSSI for channel quality assessment and utilizes upstream-driven hopping sequence replacement together with negotiated blacklisting. The channels that are blacklisted on either the upstream node or the downstream node are not used for communication. However, ATSCH has several drawbacks: it reserves several slots in each slotframe for channel sampling and one slot for control traffic, which significantly reduces the network capacity and incurs large costs for idle listening.

ETSCH [41] (Enhanced TSCH) is an improvement upon ATSCH; it is a global method that provides more accurate channel quality assessment thanks to more frequent RSSI sampling and avoids losing schedule capacity. It achieves this by sampling the RSSI in the idle period of a timeslot, i.e., after the start of the slot, but before the start of packet transmission. Our work utilizes this technique from ETSCH.

Li *et al.* [34] take a different approach and contribute a local, downstream driven method. It uses the packet reception ratio (PRR) metric and has promising results. However, their work is missing details necessary for a full implementation of the method, as well as makes assumptions that are unrealistic in SPHERE's context: for example, that an upstream node knows the packet generation rate for each downstream node.

Our previous work on the topic [19] shows that the downstream-based PRR method performs best; however, the upstream-driven RSSI-based one is a close second. In this work we selected the latter due to its lower resource requirements: the limited RAM size of the SPHERE hardware does not allow to keep detailed per-neighbor per-channel statistics, and its implementation simplicity.

For the same reasons, Kotsiou *et al.*'s [33] local, downstream driven method is not practical in our setup. It also shows too aggressive blacklisting behaviour, leaving less than 4 channels usable on average (for more than 2 m long links) even without artificial interference [33]. In contrast, SPHERE requires that at least 7 channels are always kept active to achieve the full capacity of the schedule (Section 4).

The SmartMesh includes a mechanism for global blacklisting. However, unlike the other methods, it is not automated and requires external intervention, therefore is not as well suited for the SPHERE system, where the goal is to minimize the effort for deployment and maintenance. Furthermore, changing the blacklist requires a reset of the network. Also, background noise levels are not monitored in the blacklisted channels, thus making it difficult to know when to remove a channel from this list.

Adaptive channel selection is not a TSCH-specific technique; it is also used in the Bluetooth standard in the form of Adaptive Frequency Hopping (AFH) [8], and in the ZigBee standard, where a single active channel is adaptively selected. However, selecting one out of n channels is a different and easier problem than selecting k channels out of n.

4 THE SPHERE TSCH SCHEDULE

4.1 The Challenge

The time in TSCH is usually split in 10 ms long timeslots. This implies a hard throughput limit for a single node: 100 packets per second (*pps*). According to IEEE 802.15.4, the maximal PHY-layer payload is 127 bytes long; when IEEE 802.15.4 MAC, 6LoWPAN and higher layer header overheads are accounted for, the actual SPHERE application payload limit is 64 bytes per packet, leading to 6.4 kBps application-layer data rate. Since the SPHERE Wearable Sensors generate data at high frequency, the network gateway (sink) operates close to this capacity limit. In particular,

the SPHERE application requires $S_{\texttt{TOTAL}}$ active slots per second on the sink:

$$S_{\texttt{TOTAL}} = S_{\texttt{DOWNLINK}} + S_{\texttt{ENVIRONMENTAL}} + S_{\texttt{FORWARDER}} \tag{1}$$

where $S_{\texttt{DOWNLINK}}$ slots are allocated for downlink traffic, $S_{\texttt{ENVIRONMENTAL}}$ for data collection from the environmental sensors nodes, and $S_{\texttt{FORWARDER}}$ from the forwarders. The latter is in turn given by:

$$S_{\texttt{FORWARDER}} = \sum_{i=1}^{N_f} \frac{N_w \frac{f_s}{s}}{p_i} \tag{2}$$

where N_f is the total number of forwarder nodes and N_w is the total number of the Wearable Sensors in the deployment. Each Wearable Sensor generates acceleration data at f_s sampling frequency, and thus is responsible for a packet generation rate of $\frac{f_s}{s}$ TSCH packets per second, where s is the number of three-axis acceleration samples contained in a TSCH packet. Assuming the worst-case scenario of a forwarder nodes picking up all advertisements from all of the Wearable Sensors, that forwarding node is going to generate $N_w \frac{f_s}{s}$ packets per second. Moreover, slot overallocation is needed to account for link-layer retransmissions, which depend on the quality of the channel from the forwarder to the sink. Each packet is expected to be transmitted $1/p_i$ times on average, where p_i is the link-layer packet acknowledgement ratio.

In practice, the SPHERE application is required to support $N_w = 5$ Wearable Sensors at $f_s = 25$ Hz. Each TSCH packet has sufficient space for $s = 12$ readings of the three-axis accelerometer together with monitoring data and packet headers. Moreover, it needs to support links of $p_i = 0.5$. Lastly, SPHERE requires at least $N_f = 5$ forwarder nodes for good in-house wireless coverage, to ensure that multiple copies of the advertisement packets are received for indoor localization. As a result, off-the-shelf TSCH does not offer the capacity that SPHERE needs ($S_{\texttt{TOTAL}} > S_{\texttt{FORWARDER}} > 100$).

We address the capacity challenge with two techniques (Section 4): first, multiple gateways (sinks); second, slot sharing (as described in [20]). The multiple gateway technique reduces the schedule capacity required at the egress points of the network, which are the most congested points in data collection networks. Through balancing the upstream traffic between two SPHERE Gateway nodes and allowing concurrent transmissions in different IEEE 802.15.4 channels, we almost double the data collection capacity of the network.

4.2 Overview of the SPHERE Schedule

The SPHERE schedule is collision-free for unicast communication and supports communication between three node categories: gateways, environmental sensor nodes, and forwarding nodes. The latter forward data from the SPHERE Environmental Sensors, as well as from the Wearable Sensors, which communicate exclusively over BLE (Section 2) and thus are not included in the TSCH schedule.

The SPHERE schedule is created statically, before compilation of the code, and is built in the firmware deployed on the sensor nodes. Essentially, all possible topologies of networks with k forwarding nodes are accommodated in this static schedule. During runtime, the nodes activate the cells corresponding to the specific topology that they are seeing in the network. This allows for fast, dynamic adaptations and leads to a schedule that is always consistent with routing. This approach relies on the number of forwarding nodes being small; to ensure collision-free unicast communication, k must be smaller than the number of channels used for communication.

The schedule consists of a single, repeating TSCH slotframe (Fig. 3), which in turn consists of five parts:

(1) A single cell for broadcast communication between all nodes (*Bcast*).

(2) A repeating block of forwarder-to-gateway and forwarder-to-forwarder cells ($F* \rightarrow G*$, $F* \rightarrow F*$).
(3) Two cells for gateway-to-forwarder communication ($GW \rightarrow F*$).
(4) A block of cells for transmissions *to* environmental sensors ($GW \rightarrow E*$ and $F* \rightarrow E*$): one slot per each sensor.
(5) A block of cells for transmissions *from* environmental sensors ($E* \rightarrow *$): one slot per each sensor.

The slotframe is parametrized by the total number of nodes in the network. Multiple variations of this slotframe are deployed in the *SPHERE 100 Homes Study*, which have networks ranging from one to two gateways, 8 – 10 environmental sensor nodes, and 3 – 5 forwarding nodes that collect data from 1 – 5 SPHERE Wearable Sensors. The total length of the slotframe is 99 slots, i.e., approximately one second. Out of those, 76 to 80 slots are used for forwarder-to-gateway and forwarder-to-forwarder communication.

Cell overallocation is used to accommodate retransmissions: at least two times many cells are reserved for transmissions as the expected maximal packet generation rate (Section 4.1). This allows SPHERE to support close-to-100 % end-to-end packet delivery rate on paths with as low as 50 % link-layer packet acknowledgement ratio.

The deployed SPHERE networks are collision-free for unicast traffic, since k is kept less than the number of channels. As a result, no packets are lost due to internal interference. This increases the total packet delivery rate in SPHERE networks, as well as makes the SPHERE network performance data much more useful for studying other packet loss factors, such as signal fading and external interference. With our approach, internal factors such as packet transmission rate do not bias the performance statistics, while this is often a problem in CSMA networks.

The schedule is tailored for networks where the forwarders generate data at high rate (in our case, mostly due to forwarding the SPHERE Wearable Sensor data) and environmental sensors generate low-rate data. However, the same approach could be applied to support different rates. By reducing the number of copies of the "repeating block" and/or adding copies of the $E \rightarrow *$ slots, the balance could be changed towards more data from the environmental sensors. By adding more slots for specific nodes, heterogeneous datarates can be supported.

4.3 Energy Efficiency

The SPHERE Environmental Sensors are battery-powered, therefore their schedule is designed to be energy-efficient. These nodes operate in RPL leaf mode: they never forward any packets. A slotframe for an Environmental Sensor has one active Tx (transmission) slot (to all forwarder and gateway nodes), one Rx (reception) slot for broadcast packets, and one Rx slot for the routing parent.

One limitation of this schedule is that an Environmental Sensor can only receive unicast data from their direct routing parent: for example, $E2$ in Fig. 3 only from $F3$. It is sufficient for the SPHERE's traffic patterns: configuration and data query messages to the Environmental Sensors originate outside of the TSCH network and strictly follow the routing tree when inside it. This limitation caused by the desire to keep the size of the slotframe small and to reduce the energy consumption. The radio duty cycle and energy consumption of a node are mostly determined by the number of its Rx slots. Even when no packets are received in an Rx slot, the node spends substantial energy for idle listening. This is partially due to the TSCH guard time [35], which is set to 1 ms in the deployed code (Table 3), partially due to the fact the CC2650 requires approximately 1 ms to be turned on and tuned on the right frequency, during which is consumes almost as much as during normal listening. Consequently, each idle Rx slot adds approximately 0.25 % to the radio

duty cycle (i.e., 2.5 milliseconds per second). If an Rx slot per each RPL neighbor was added to the node's schedule, its energy consumption would increase substantially; e.g. to $0.25 \times 8 = 2\,\%$ idle radio duty cycle if slots to 8 additional neighbors were added. In contrast, a Tx slot is only used if there is a packet to transmit. Most of the Tx slots are left idle, since a node only produces one sensor data message per 20 seconds on the average, as well as some amount of monitoring and network maintenance traffic.

4.4 Multiple Gateways

The first publicly deployed version of the SPHERE system supports just one gateway (Table 3). The channel capacity at the gateway is the data-rate bottleneck in these networks. Additionally, this lone gateway is a single point-of-failure in the system. The performance of the whole network highly depends on it; at the same time, it is prone to localized signal fading or obstruction to the same degree as all other nodes. It is clear that multiple, simultaneously operational gateways can increase both the capacity and the reliability of the network. Moreover, after the first 31 deployments of the SPHERE system the Wearable Sensor sampling rate was increased from 12.5 Hz to 25 Hz, effectively doubling the data rate requirements for the TSCH network and further increasing the need to have a second gateway.

Adding multiple gateways is a natural way to increase the performance of the network. However, this idea has not been explored in TSCH research papers previously, because doing this at the TSCH level is challenging without external time synchronization mechanisms. The network's time synchronization graph must retain an acyclic structure, which does not happen naturally in a network with multiple equally-ranked root nodes.

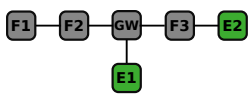
We experimented with two approaches for supporting the multiple-gateway functionality:

(1) Layer-3 load balancing. In this approach, a forwarder node keeps all reachable gateways in its routing and neighbor tables, and randomly routes each data packet to one of these gateways; meaning that it puts the packet in the gateway's packet queue and transmits it in cells scheduled for that gateway.
(2) Layer-2 load balancing. In this approach, a single "main" gateway is defined in the software. The forwarder node puts all data packets in the packet queue of the main gateway (i.e., $G1$ in Fig. 3d). However, it uses slotframe cells scheduled to *any* gateway (Fig. 3d) to transmit that packet. Upon reception this packet, any gateway sends an ACK to the forwarder node on behalf of $G1$; i.e., the link-layer destination address in the packet is that of $G1$.

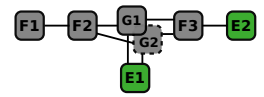
The essential difference between the approaches is that with Layer-3 load balancing, an unacknowledged packet is always retransmitted to the same neighbor. In contrast, Layer-2 load balancing effectively randomizes the neighbor selection for each retransmission, adding spatial diversity to the protocol. As shown in the evaluation (Section 4.8), the latter has major benefits from reliability perspective. As a result, we use the Layer-2 approach in the SPHERE deployments, even though it departs from the IEEE 802.15.4 requirement to not acknowledge packets addressed to other nodes.

For the implementation, we exploit the fact that the SPHERE SPG-2 nodes have two radios and two microcontrollers. As a result, we achieve dual-gateway load-balancing without deploying extra devices: instead, one "side" of the root SPG-2 node is programmed as $G1$, while the other is programmed as $G2$. Both are connected to the SPHERE Home Gateway using two separate USB cables.

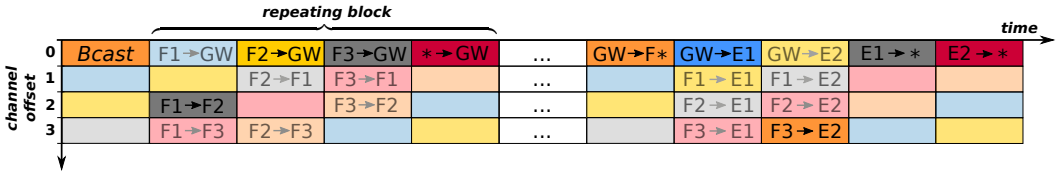
There are no fundamental reasons that limit the system to just two gateways: more could be added in the future if desired. However, the cost of more gateways is an additional operational and maintenance complexity.



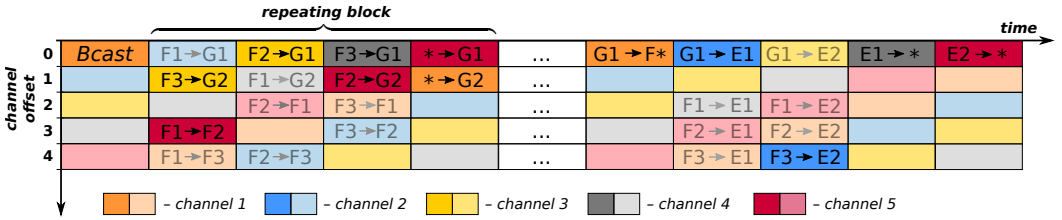
(a) Example network with a single IoT gateway



(b) Example network with two co-located IoT gateways



(c) Schedule for the single-gateway network



(d) Schedule for the double-gateway network

Fig. 3. **A simplified example of SPHERE IoT networks and their TSCH schedules.** The different colors correspond to different IEEE 802.15.4 MAC-layer channels; inactive slots have lighter colors. GW ($G1$, $G2$) denotes gateway nodes, $F1, F2, F3$ denote forwarder nodes, $E1, E2$ environmental sensors. $BCast$ marks a common timeslot reserved for broadcast. $* \rightarrow GW$ denotes a shared slot [20], where all nodes directly connected to the gateway are allowed to transmit. $GW \rightarrow F*$ denotes communication from the gateway to the forwarders. Comparison between schedules (c) and (d) shows that the "repeating block" part in (c) has 4 active cells, while in (d) it has 7 active cells: 75 % increase in capacity. The "repeating block" part is not repeated in identical copies; in subsequent repetitions the order of its timeslots is randomized to improve robustness of this schedule.

4.5 Runtime Behavior

To summarize our design, at the start of each active TSCH slot, a node wakes up and dynamically selects which cell to use depending on its RPL (Routing Protocol for Low power and Lossy Networks) routing state. Figures 3c and 3d shows this through displaying active cells in brighter colors (cells active specifically for the topologies in Figures 3a and 3b). In this way, topology changes are handled in a highly dynamic way, without needing to modify the schedule itself, unlike in other TSCH scheduling solutions.

Algorithm for the forwarding nodes:

(1) On a broadcast slot:
  (a) if any broadcast packets are queued, send one of them using channel offset (CO) zero;
  (b) otherwise, listen for packets on CO zero.
(2) On a Tx slot to a specific leaf node: if any packet to that node, send out a packet to the node, using the CO corresponding to the forwarder itself.
(3) On an Rx slot from any leaf node: listen to packets on CO zero.

(4) On a shared slot that has been successfully reserved for this node during the dedicated communication period: transmit a packet to the negotiated gateway. Otherwise, enter a low-power mode.
(5) On a dedicated slot:
  (a) If a direct child of this forwarder is marked as a sender in that slot: listen for packets using the CO of itself.
  (b) Otherwise, if there is any neighbor with a non-empty packet queue, select a packet for that neighbor and transmit it using the CO from the cell scheduled to that neighbor. If the receiver is the gateway, the use of the next shared slot may be negotiated.
  (c) Otherwise, enter a low-power mode.

The algorithm is similar on the gateway node, except that the *GW* node always listens during shared slots.

4.6 Implementation and Evaluation Metrics of the SPHERE Scheduler

We implement SPHERE TSCH on top of the Contiki TSCH and 6LoWPAN stack [17]. The SPHERE data collection application utilizes the CoAP protocol. Each individual sensor on the Environmental Sensor nodes is exported as an observable CoAP resource. These resources are observed by a CoAP client running on the SPHERE Home Gateway.

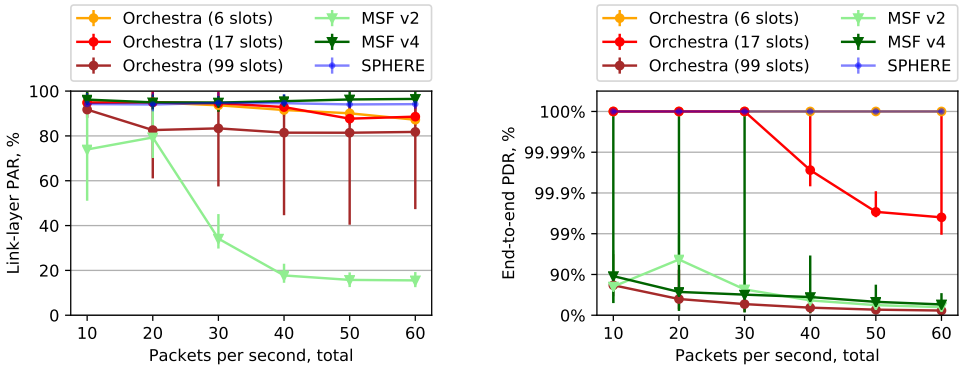
For the evaluation of the scheduler, we use the following metrics:

- **Packet Delivery Ratio (PDR)**: the number of CoAP packets received by the Home Gateway divided by the number of CoAP packets generated by the sensor nodes and destined to the Home Gateway. This is the metric of primary importance when measuring the reliability of the protocol. We account for application-level PDR; both packets that are dropped due to queue overflow and packets that are dropped due to the retransmission limit exceeded are counted as losses.
- **Packet Acknowledgement Ratio (PAR)**: the number of CoAP packets acknowledged by a node's neighbor divided by the number of CoAP packets sent to that neighbor. This is not as important metric on its own, but is useful as the main proxy for PDR. Low PAR means that many retransmissions are necessary to achieve high PDR, which requires that the schedule has a large overallocation factor.
- **Number of active reception slots per time unit.** This number is a proxy of how energy-efficient the software is. As shown in our previous work [25], the majority or large part of the total energy consumed by the SPHERE Environmental Sensors is consumed for radio listening and reception.
- **Charge consumption per time unit,** for the whole device or part of it (e.g., the radio). This metric is not directly measured in this work; instead, it is extrapolated from software-estimated activity profile of the device [15] or the number of active slots per second. This approach gives relatively accurate results ($\leq$10 % error [25]). We use charge consumption, instead of energy consumption, as the latter depends on hardware-specific details irrelevant to the protocol, such as battery voltage changing over time [25].

We keep these metrics thorough this paper, including Section 6.

4.7 Experimental Evaluation of Single Gateway Scheduling

For comparison, we also implement the SPHERE application on top of the Orchestra [16] and the 6top Minimal Scheduling Function (MSF) [10] schedulers. The MSF has significantly evolved during the writing of this paper; as a result, we compare both an early version (v2) of the MSF implemented



(a) Link-layer PAR comparison

(b) PDR comparison

Fig. 4. **Reliability comparison between SPHERE TSCH, Orchestra, and MSF schedules.** Experimental results in the SPHERE House with 5 forwarding nodes. The SPHERE schedule demonstrates 100 % reliability and close-to-constant link-layer packet acknowledgement ratio (PAR) under all loads. The v2 implementation of MSF fails to setup consistent schedules on the sender and receiver nodes, especially under higher traffic rates. The v4 implementation keeps the schedule consistent: high PAR in all experiments, but fails to allocate sufficient number of cells: low PDR due to queue losses. The reliability of Orchestra depends on its slotframe size, and it is not collision free, as shown by the PAR dropping under more heavily loads.

by us, and a latter version (v4) of the MSF implemented by INRIA[1]. For Orchestra, we vary the size of the slotframe (6, 17, and 99 slots); the other options all use 99 slot slotframes. We found that the packet burst option [6] has a large impact on the results. For Orchestra, the results are much better when this option is enabled. For MSF, the opposite is true. As a result, we keep this option enabled for Orchestra experiments, but disable for both MSF versions. Early versions of the MSF Internet Draft [10], including the version 2, asked for this option to be enabled, but since then they have changed their recommendations. For SPHERE TSCH the burst option is disabled.

We run a separate experiment for each protocol version, with a single gateway and 5 forwarder nodes. Initially, the data rate is set to 10 packets per second. After each 5 min increased by additional 10 *pps* until 60 *pps* are reached. The results are shown and summarized in Figs. 4 and 5. For simplicity, these experiments do not include the Environmental Sensors; the Orchestra schedule does not support heterogeneous networks, as discussed further, and the MSF results are already unsatisfactory for forwarder nodes on their own.

The main drawback of Orchestra in this application is that it forces a suboptimal trade-off between the throughput and energy efficiency. It does not support both high-rate nodes and low-rate energy-constrained nodes in the same network, as required by the SPHERE application. The 99-slot version of Orchestra would have similar energy consumption as SPHERE TSCH on the Environmental Sensors, but it shows much worse PAR and PDR in the experiments than the short-slotframe versions. The versions with high PDR would have several times higher energy consumption for radio Rx. The other drawback of Orchestra is that it is not collision free. The link-layer PAR decreases as the packet rate goes up (Fig. 4a) even for the 6-slot and 17-slot versions.

The 6top MSF schedulers are relatively slow to react on traffic changes; this is their main drawback for the application. The MSF v2 also suffers from an implementation problem: in Contiki code, 6top

[1] https://github.com/contiki-ng/contiki-ng/pull/1000

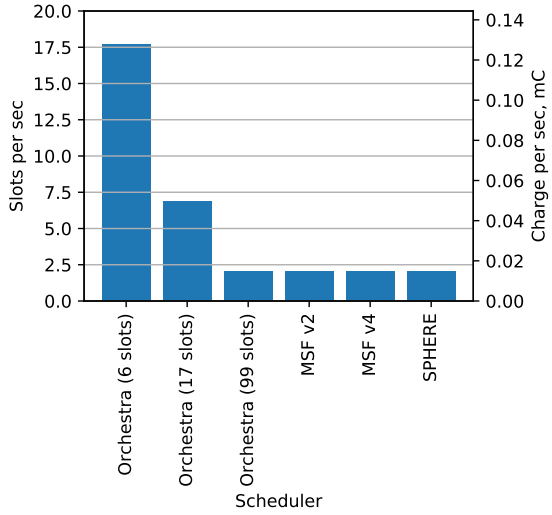


Fig. 5. **Active Rx slots per second on the Environmental Sensors and corresponding charge consumption for radio reception** (millicoulombs per second). Estimated results using settings from the SPHERE House experiment with 5 forwarding nodes. Orchestra imposes a tradeoff of short slotframe (more reliable) vs. long slotframe (more energy efficient). SPHERE avoids this tradeoff by using a different schedule on the Environmental Sensors.

packets are not prioritized over data packets; they get delayed in the packet queue or dropped altogether. The MSF Internet Drafts define a way to handle inconsistencies: the whole slotframe is cleared if one is detected [10]. We did not implement this in MSF v2 , as results showed that with higher traffic rates (50 *pps*) the 6top protocol is not able to set up the schedule after a complete reset due to the packet prioritization issue. The comparison with MSF v4 shows different results: the schedule consistency is preserved; however, the PDR is low as the scheduler fails to reserve negotiated cells. This is so even after we tweak the parameters to allocate the cells more agressively. The parameter values given in the Internet Draft require that new cells allocation starts when the schedule is at 75 % capacity. We report results with 25 % as the allocation threshold (300,% schedule overallocation) since the default settings gave worse performance. It remains to be seen in the future work whether the cause of the bad performance of MSF is in the implementation details and parameters.

4.8 Experimental Evaluation of Scheduling with Load Balancing

In terms of double-gateway scheduling, we compare Layer-2 and Layer-3 load balancing (Section 4.4); each point in the graph in the Figure 6 is obtained in a separate 10 min experiment. The results (Fig. 6) clearly show that the Layer-2 approach performs better: it is able to deliver 100 % of packets even at 112 *pps* datarate. The benefits from the Layer-2 approach come from its additional spatial diversity, as evidenced by the large difference in RSSI levels measured on the two gateways: the average difference in the average RSSI is 4 dBm, despite the two gateways being physically located on the same node, with antennas < 5 cm apart. For some of the forwarder nodes, this difference between the RSSI on the two gateways reaches 9 dB.

We do not compare the load balancing approach with other solutions, since the Orchestra and MSF schedulers do not support multiple TSCH gateways.

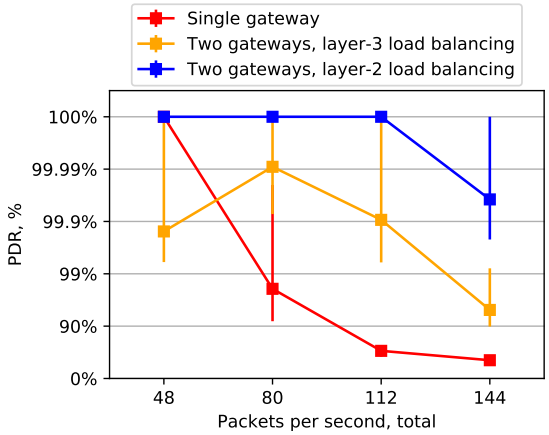


Fig. 6. **Reliability benefits from the load balancing approaches.** Experimental results in the SPHERE House with 8 forwarding nodes, using 80 % of slots for the data collection. Double-gateway approach with Layer-2 load balancing achieves 100 % reliability for up to 112 packets per second from the network. Layer-3 load balancing shows better results on the average than using single gateway, but loses some packets in the 48 *pps* experiment where the other approaches have 100 % PDR.

5 ADAPTIVE CHANNEL HOPPING

5.1 Background

Channel hopping adds reliability compared with single-channel solutions [50]. However, it is preferable that channels which suffer from heavy interference are completely excluded from the channel-hopping sequence.

The primary sources of interference in the 2.4 GHz frequency band are the WiFi networks of the house and of the neighboring houses. Fig. 7 shows a snapshot of the WiFi interference levels of the first nine deployments, taken during pre-deployment survey visits. The figure demonstrates that WiFi interference is distributed in all of the band, implying that it would be not possible to choose a static sequence of channels that would fit all deployments. Moreover, it is very likely that the active WiFi channels would change during a long-term deployment. Furthermore, additional sources such as microwave ovens are present in home environments; they can block large parts of the 2.4 GHz band when active [31]. It is therefore apparent that in such ever-changing and unpredictable conditions, static channel selection would operate sub-optimally. Instead, we tackle this challenge with an *adaptive channel hopping* mechanism.

5.2 Channel Quality Estimation

We use channel quality estimation based on background noise RSSI (received signal strength indicator), described in Elsts *et al.* [19], with the goal to detect and avoid external interference. Within a TSCH timeslot, there is a brief interval when all nodes in the network are guaranteed to be not transmitting: between the start of a TSCH timeslot and the Tx (transmission) offset, which is defined by the IEEE 802.15.4 standard to be 2120 μs later. The node that wishes to estimate channel quality samples RSSI during this interval. In SPHERE this estimation is done only on the main gateway, which is not energy-restricted. All 16 channels are sampled in a sequence, meaning that these RSSI statistics are available even for channels not actively used by TSCH.

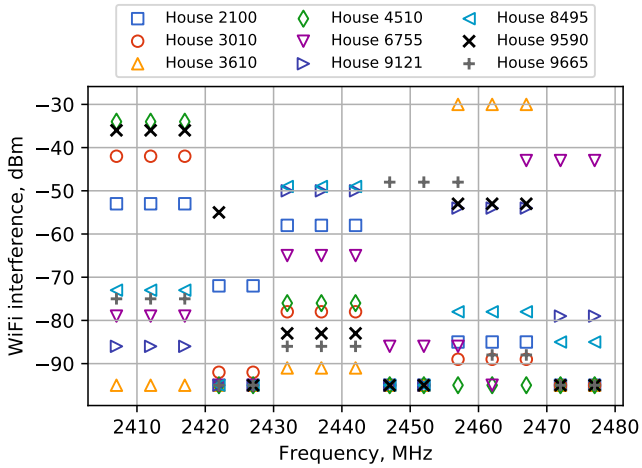


Fig. 7. **WiFi interference levels in SPHERE homes [21].** Data from 9 initial public deployments, taken on pre-deployment site-survey visits.

We select the Exponentially-Weighted Moving Average (EWMA) filter to incorporate history information in the RSSI metric. An EWMA channel quality estimator is defined as follows:

$$p_{n+1} = (1 - \alpha)p_n + \alpha Y_n \ , \tag{3}$$

where $Y_n \in \{0, 1\}$ depending on whether the n-th RSSI was below a certain threshold (−85 dBm in SPHERE). This filter is well-suited for embedded devices, as it requires only $O(1)$ memory, and its sensitivity can be tuned with its α parameter. The latter is set to 0.045 in SPHERE, selected using the algorithm in [19]. The EWMA-filtered value is computed for each channel separately. If this value is below another threshold (0.85 in SPHERE), the channel is considered to be busy; otherwise it is considered free.

5.3 The Channel Selection Algorithm

Whenever a channel changes its state (from free to busy or from busy to free), the following selection algorithm is run:

(1) Create an array with pairs of (*channel_quality*, *channel_number*) for all 16 channels.
(2) Set the quality of IEEE 802.15.4 channels 15 and 26 to zero. These channels are never selected in order to avoid interfering with SPHERE's BLE subsystem.
(3) Sort all channels by their quality.
(4) Mark the best N channels as free, irrespective of their quality ($N = 7$ in SPHERE). This is done to preserve the minimal number of free channels.
(5) Loop for all channels in the current hopping sequence that are still marked as busy, in order from the worst to the best. For each channel:
  (a) Mark that channel for replacement.
  (b) Set the quality threshold as the quality of the channel plus a hysteresis constant (0.1 in SPHERE).
  (c) Loop for all channels that are *not* in the current hopping sequence and that are marked as free, from the best to worst; break the loop when the quality of the next channel is lower than the quality threshold. Otherwise, if the new channel has been recently (5 min in

SPHERE) replaced, continue. Otherwise, replace the marked channel and break the inner `for` loop.

(d) If the marked channel has been replaced, break the outer `for` loop; otherwise, continue with the next channel.

Extra care is taken to avoid completely drifting away from the initial hopping sequence. This is a potential problem, since a node would not be able to rejoin the network in case the set of the channels it scans is completely distinct from the set of channels used by the network at the moment of scanning. For this reason, in step (5c) above, a channel is never selected as the replacement channel if it would replace the last channel in the current hopping sequence that is also present in the initial hopping sequence.

5.4 Hopping Sequence Redistribution

After a new hopping sequence has been selected by the main gateway, it must be made known to other nodes, so that channel mismatches between transmitter and receiver nodes are avoided. The IEEE 802.15.4 TSCH standard defines an Information Element (IE) for distributing hopping sequence in EB messages. The SPHERE system makes use of this, and after changing the hopping sequence the main gateway sends out a burst of EB packets.

Unfortunately, EB transmissions are not reliable, as they are done as broadcasts on shared slots, and there is only one EB slot in each slotframe. As a result, even bursts of EB propagate changes across the network in a relatively slow manner. To improve on this, SPHERE TSCH adds another IE to acknowledgement packets. This IE has 4-byte payload, which contains the new hopping sequence. In this way, the downstream node is able to update its hopping sequence as soon as it hears and acknowledgement from an upstream node.

Furthermore, in this way, the speed in which changes are propagated to a node is dependent on its data rate. There is only a small constant energy cost, as acknowledgement packets become 6 bytes longer.

5.5 Inter-Network Interference

Multiple SPHERE deployments may take place in neighboring houses or apartments. It is important to ensure that interference between these networks do not cause heavy packet loss. This can happen as two or more TDMA networks that share the same slotframe size are vulnerable to undesired temporary synchronization between their slotframes due to differences in clock drift on the networks' respective timesource nodes [28].

The solution applied by SPHERE is to randomize the hopping sequence of each network before deployment. This avoids heavy packet loss even when this undesired synchronization takes place: due to randomization, the same channel offset in the different networks translates to different frequencies. Even though the neighboring networks can, in theory, eventually arrive to the same hopping sequence through adaptive channel selection, it is very unlikely due to the large number of all possible permutations of N elements (where $N = 7$ in SPHERE).

6 THE *100 HOMES STUDY* TRIALS

6.1 Overview

Table 3 lists the configuration parameters used in the SPHERE *100 Homes Study* deployments. The first 31 public SPHERE deployments use version D; the subsequent ones all use SPHERE version E. Version E adds load balancing, increases queue sizes to better deal with bursty packet losses, and separates unicast packets into two classes: data packets and others. That allows to increase the number of retransmissions for data packets, while keeping the delivery of network monitoring packets as a lower priority. Version E was originally created because SPHERE needed to double the data rate of the Wearable Sensors (Section 4.4).

Table 3. Software configuration for SPHERE *100 Homes Study* deployments

Parameter	SPHERE version D	Version E
TSCH parameters		
Number of gateways	1	2
Queue size	8 packets	16 packets
Slotframe size	99 slots	
Shared slots	+	
Number of channels	7	
Channel adaptations	+	
Adaptive time sync	+	
Guard time	1000 μs	
Security	AES-128 CTR mode	
Other Contiki parameters		
Routing	RPL MRHOF	
Max transmissions for data	8 times	16 times
Max transmissions for other	8 times	3 times
Serial Line IP (SLIP) queue	4 packets	6 packets
Max transmissions over SLIP	4 times	
External parameters		
Wearable Sensor sampling	12.5 Hz	25 Hz

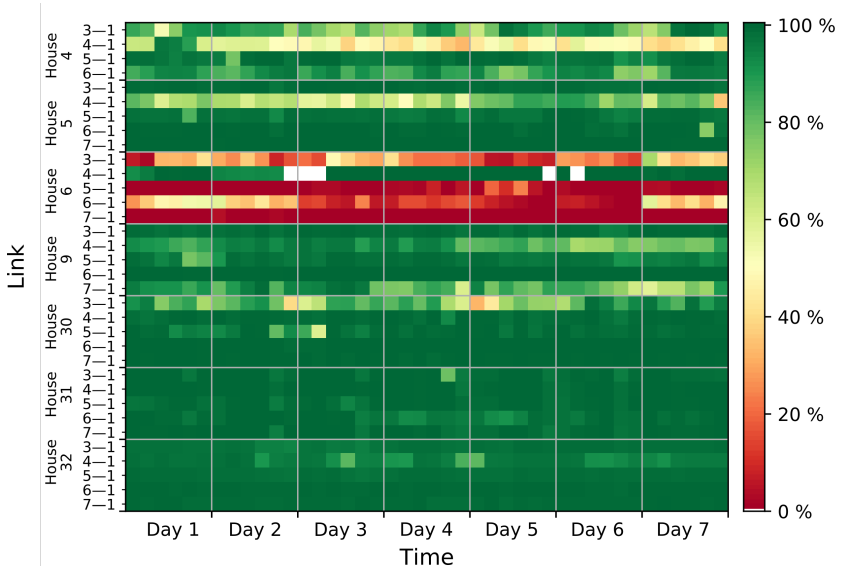


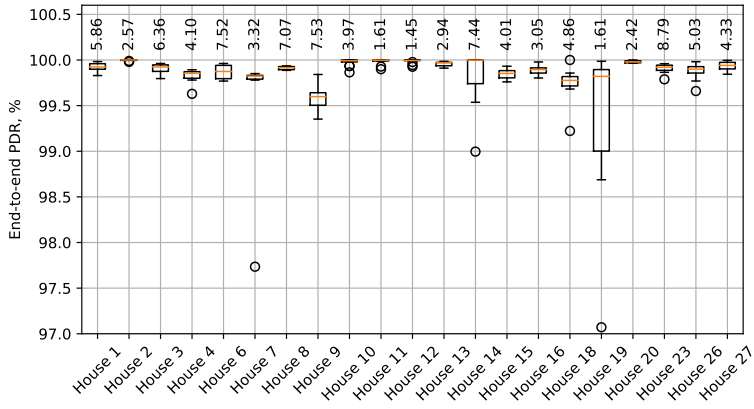
Fig. 8. **Link-Layer Packet Acknowledgement Rate (PAR).** Data from the first seven public deployments from which full data was collected. Both spatial and time diversity are present. For each house, the third week of its deployment is shown if the system was recording at that time, or the closest fully recorded week otherwise.

6.2 Link-Layer Performance

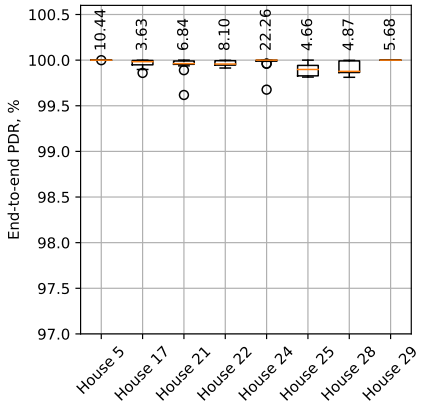
Figure 8 plots selected portions of the link-layer performance data from deployments, using data from the first fully-available datasets from seven houses. Both variations in time and variations among links are present. Since visualizations of the full datasets would be too difficult to include in this paper, only data from the third week of each deployment is plotted, where available.

6.3 Reliability

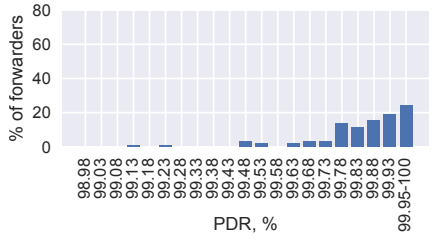
The PDR comparison (Fig. 9) shows that all nodes have $\geq 99\,\%$ average reliability, in all except two houses. The results from version E compares favorably with the results from version D, demonstrating the benefits from load balancing and parameter optimizations. The reliability is not dependent on the datarate, suggesting that packet losses are caused by environmental or hardware problems (e.g., by reboots of the gateway node). An ANOVA test confirms that the difference between the two versions is statistically significant ($p < 10^{-8}$).



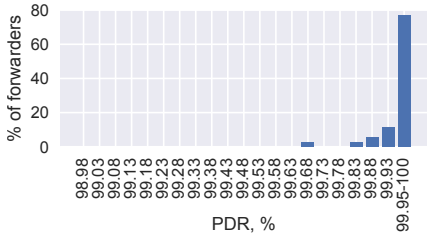
(a) SPHERE version D



(b) SPHERE version E



(c) Forwarders, version D



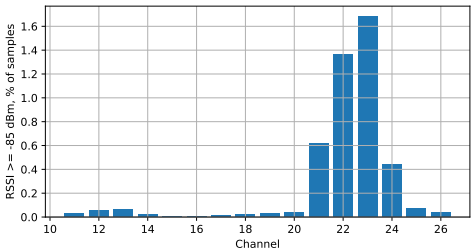
(d) Forwarders, version E

Fig. 9. **Reliability in the 100-homes deployment.** Data from the 29 deployments which were active on 21st–28th March, 2018, produced monitoring data, and did not have known installation issues (SHG problems). Boxes correspond to upper and lower quartiles; the orange line to the average; points (outliers) correspond to single nodes; *pps* rate is given on the top. **(a) SPHERE version D**: Average 2.7 *pps* per network, average 99.85 % PDR, all nodes show better than 97 % delivery rate. **(b) SPHERE version E**: Average 7.5 *pps* per network, average 99.96 % PDR, all nodes show better than 99.6 % delivery rate. **(c)** Distribution of the PDR on forwarder nodes, version D. **(d)** Distribution of the PDR on forwarder nodes, version E.

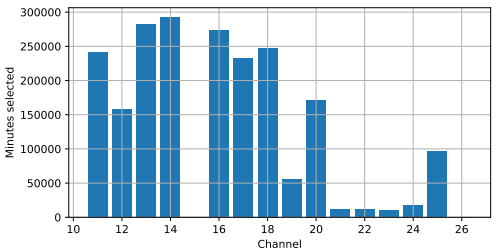
6.4 Channel Selection

As shown in Fig. 10, there is a high correlation between the IEEE 802.15.4 channels that are detected as "occupied" by WiFi access points in the pre-deployment site survey, and the channels that are detected as busy during the deployment. The SPHERE system is able to detect channels with high WiFi signal levels and select the other channels for the operation of the system (Fig. 10c).

(a) WiFi channel scan



(b) Percent of time channels detected as busy



(c) The channels selected by the SPHERE system

Fig. 10. **The TSCH network detects and avoids busy WiFi channels**; data from a 4 month-long representative deployment (House 32). (a) WiFi scan in a pre-deployment survey visit compared with (b) percent of time channels detected as busy by the TSCH network (on the IoT gateway node) and (c) the channels selected by the SPHERE system (the channels 15 and 26 are never selected to avoid interfering with SPHERE's BLE traffic.

6.5 Energy Efficiency

The Contiki Energest software [15] remains enabled during the deployment, allowing to extract the data afterwards and to plot the estimated daily charge consumption (Fig. 11). The results show stable performance, confirming that the energy consumption of the SPHERE TSCH network stack is highly predictable before deployments, as long as the energy consumption of the hardware platform is accurately known in its different operating states.

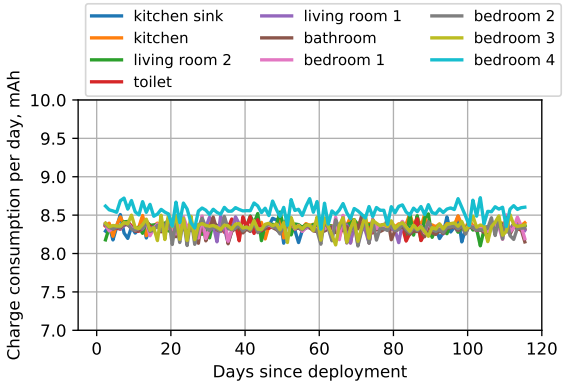


Fig. 11. **Charge consumption history in a representative deployment (House 32),** software-based estimation. All nodes show stable consumption over time with only minor random variations, allowing to accurately predict the total software-estimated energy consumption.

7 DISCUSSION

7.1 Suitability for the Application

Given the good overall performance in terms of PDR in uncontrolled environments (Fig. 9), the stability of the end-to-end PDR after long-term residential deployments (up to 1 year in Fig. 9a), the high datarate sustained (7.5 *pps* average in version E deployments, up to 40 *pps* in individual homes in some weeks), and the stable daily charge consumption (Fig. 11), we can conclude that SPHERE TSCH is highly suitable for the SPHERE data collection application.

We also note that the SPHERE results cannot be directly compared with the wire-like reliability promised by other TSCH implementations. This is partially due to the higher data rates in SPHERE compared with other TSCH evaluation studies [9, 16, 48], partially due to less-controlled environments, partially due to a terminology issue: not being able to send out a packet because of queue being full is considered a reliability (PDR) issue in SPHERE (Section 4.6), but an availability issue in other implementations (such as SmartMesh).

7.2 Impact of the SPHERE-Specific Aspects

- *Schedule overallocation.* SPHERE schedule is designed for links with $\geq 50\,\%$ average quality, as it has an overallocation factor of two. The link-layer performance data (Fig. 8) show that in most houses most of the time, this link quality assumption is justified. Furthermore, if that is not the case (as in the House 6), the 6LoWPAN routing protocol does avoid the low-quality links.
- *Multiple gateways.* Adding the second gateways almost doubles the schedule capacity (Fig. 3). Additionally, despite being co-located, the difference in RSSI between the two gateways is up to 9 dB for some links in the SPHERE House (Section 4.8); therefore the two gateways provide not just nominal capacity increase, but also spatial diversity.
- *Channel selection.* The results show that the SPHERE system can detect and avoid channels that are used by WiFi (Fig. 10).
- *Other parameters.* Increasing the queue size and reducing the number of retransmissions for monitoring data in version E decreased the probability that a data packet is lost because the queue is full.

The comparison between version D and version E (Fig. 9) clearly shows the effect from the second gateway, as well as the optimization of other parameters. Despite having to deal with two times higher data rate from the Wearable Sensors, version E drops 4 times fewer packets than version D: an order of magnitude cumulative improvement.

Finally, it is worth noting that the system achieves wire-like reliability (100 % PDR) in the semi-controlled environment of the SPHERE House, while it performs worse (99.96 % PDR on average) in the residential deployments "in the wild". This discrepancy highlights the need to evaluate network protocols in actual real-world conditions, not just in realistic testbeds.

7.3 Future Work

Looking forward to future applications of the SPHERE platform, the support for wire-like reliability is needed for safety-critical healthcare data (e.g., medical alarms). To achieve that, the network stack would have to, as a minimum, provide better support for multiple QoS classes. The SPHERE version E is already a step in this direction, as in this version different types of packets have different numbers of retransmissions. However, multiple packet queues would be needed as well. Multi-parent and opportunistic routing could increase the average reliability through increased spatial diversity. Care should be taken to vary the transmission destination between packet retransmissions; cf. the results in Fig. 6.

The second limitation is the lack of support for mobile nodes. Instead of TSCH, the mobile SPHERE Wearable Sensors currently use BLE in advertising mode. The advertising mode does not have retransmissions, consequently, it does not provide reliability. To support mobility on top of TSCH, the routing protocol would have to be changed or replaced, and anycast addressing or gateway handover mechanisms investigated. We present initial results TSCH for mobile wearable nodes using the *Instant* protocol in [22].

8 CONCLUSION

In this paper we presented SPHERE TSCH: an IEEE 802.15.4 TSCH based network stack for data collection in smart homes for healthcare. Performance evaluation in a test house shows that SPHERE TSCH achieves better reliability than the MSF distributed scheduling mechanism and better energy efficiency tradeoffs than the Orchestra scheduler. Trials in 29 public deployments, where the system is further enhanced with load balancing through multiple gateways and adaptive channel selection, show that SPHERE TSCH suits the application well in terms of reliability (99.96 % average PDR), schedule capacity (with up to 22 *pps* average packet rate), and energy efficiency (less than 9 mAh consumption per day on environmental sensors).

ACKNOWLEDGMENTS

This work was funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1, as well as the ERDF Activity 1.1.1.2 "Post-doctoral Research Aid" (No. 1.1.1.2/VIAA/2/18/282). A full list of SPHERE collaborators can be found at http://www.irc-sphere.ac.uk. The authors acknowledge the valued contribution of the 50+ researchers and support personnel of the project.

REFERENCES

[1] [n. d.]. CC2650 SimpleLink Multistandard Wireless MCU. http://www.ti.com/lit/ds/symlink/cc2650.pdf.
[2] [n. d.]. CC3200 SimpleLink Wi-Fi and Internet-of-Things Solution, a Single-Chip Wireless MCU. http://www.ti.com/lit/ds/symlink/cc3200.pdf.
[3] [n. d.]. SimpleLink CC3100/CC3200 Wi-Fi Internet-on-a-chip Networking Sub-system Power Management. http://www.ti.com/lit/an/swra462/swra462.pd.

[4] 2007. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals.* RFC 4919. IETF.

[5] 2007. ISA100: Wireless Systems for Industrial Automation-Developing a Reliable, Universal Family of Wireless Standards. ISA.

[6] 2015. IEEE Standard for Local and metropolitan area networks—Part 15.4. IEEE Std 802.15.4âĂŘ2015.

[7] SIG Bluetooth. 2017. Mesh Profile Specification: 1.0. *Bluetooth Special Interest Group: Kirkland, WA, USA* (2017).

[8] Bluetooth SIG. 2010. Specification of the Bluetooth System - Covered Core Package version: 4.0.

[9] Keoma Brun-Laguna, Ana Laura Diedrichs, et al. 2018. Using SmartMesh IP in Smart Agriculture and Smart Building applications. *Computer Communications* (2018).

[10] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne. 2017. *6TiSCH Minimal Scheduling Function (MSF).* Internet Draft. IETF.

[11] Deji Chen, Mark Nixon, and Aloysius Mok. 2010. *WirelessHART(TM): Real-Time Mesh Network for Industrial Automation.* Springer. ISBN: 1441960465.

[12] Medical Research Council. 2017. Data and Tissues Tool Kit: Archiving. http://www.dt-toolkit.ac.uk/researchscenarios/archiving.cfm.

[13] Domenico De Guglielmo, Beshr Al Nahas, Simon Duquennoy, Thiemo Voigt, and Giuseppe Anastasi. 2017. Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA algorithm. *IEEE Trans. on Vehicular Technology* 66, 2 (2017), 1573–1588.

[14] Peng Du and George Roussos. 2012. Adaptive time slotted channel hopping for wireless sensor networks. In *4th Computer Science and Electronic Engineering Conference (CEEC).* IEEE, 29–34.

[15] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. 2007. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors.* ACM, 28–32.

[16] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *ACM SenSys.* 337–350.

[17] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomou. 2017. TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation. In *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS).* 11–18. https://doi.org/10.1109/DCOSS.2017.29

[18] Atis Elsts. 2016. Source Node Selection to Increase the Reliability of Sensor Networks for Building Automation. In *EWSN.* 125–136.

[19] Atis Elsts, Xenofon Fafoutis, Robert Piechocki, and Ian Craddock. 2017. Adaptive channel selection in IEEE 802.15.4 TSCH networks. In *Global Internet of Things Summit (GIoTS), 2017.* IEEE, 1–6.

[20] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock. 2017. Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks. In *DCOSS.* IEEE, 3–10. https://doi.org/10.1109/DCOSS.2017.20

[21] Atis Elsts, Xenofon Fafoutis, Przemyslaw Woznowski, Emma Tonkin, George Oikonomou, Robert Piechocki, and Ian Craddock. [n. d.]. Enabling Healthcare in Smart Homes: The SPHERE IoT Network Infrastructure. *Accepted for publication in IEEE Communications Magazine* ([n. d.]).

[22] Atis Elsts, James Pope, Xenofon Fafoutis, Robert Piechocki, and George Oikonomou. 2019. Instant: A TSCH Schedule for Data Collection from Mobile Nodes. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN).*

[23] X. Fafoutis, A. Elsts, R. Piechocki, and I. Craddock. 2018. Experiences and Lessons Learned from Making IoT Sensing Platforms for Large-Scale Deployments. *IEEE Access* 6 (2018). https://doi.org/10.1109/ACCESS.2017.2787418

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

[25] Xenofon Fafoutis, Atis Elsts, Antonis Vafeas, George Oikonomou, and Robert J Piechocki. 2018. On Predicting the Battery Lifetime of IoT Devices: Experiences from the SPHERE Deployments. In *RealWSN@ SenSys.* 7–12.

[26] Xenofon Fafoutis, Antonis Vafeas, et al. 2017. Designing Wearable Sensing Platforms for Healthcare in a Residential Environment. *EAI Endorsed Trans. Pervasive Health and Technology* 17, 12 (Sept. 2017). https://doi.org/10.4108/eai.7-9-2017.153063

[27] Elena Fasolo, Michele Rossi, Jorg Widmer, and Michele Zorzi. 2007. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications* 14, 2 (2007).

[28] Laura Marie Feeney, Michael Frey, Viktoria Fodor, and Mesut Gunes. 2015. Modes of inter-network interaction in beacon-enabled IEEE 802.15.4 networks. In *Ad Hoc Networking Workshop (MED-HOC-NET), 2015 14th Annual Mediterranean.* IEEE, 1–8.

[29] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd annual Hawaii international conference on system sciences.* IEEE.

[30] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. 2017. Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey. *Computer Communications* 114 (2017), 84–105.

[31] A. Kamerman and N. Erkocevic. 1997. Microwave oven interference on wireless LANs operating in the 2.4 GHz ISM band. In *Personal, Indoor and Mobile Radio Communications, 1997. Waves of the Year 2000. PIMRC '97., The 8th IEEE International Symposium on*, Vol. 3. 1221–1227 vol.3. https://doi.org/10.1109/PIMRC.1997.627080

[32] Seema Kharb and Anita Singhrova. 2018. A survey on network formation and scheduling algorithms for time slotted channel hopping in industrial networks. *Journal of Network and Computer Applications* (2018).

[33] Vasileios Kotsiou, Georgios Z Papadopoulos, Periklis Chatzimisios, and Fabrice Theoleyre. 2017. LABeL: Link-based Adaptive BLacklisting Technique for 6TiSCH Wireless Industrial Networks. In *ACM MSWiM*. 25–33.

[34] Peishuo Li, Tom Vermeulen, Hong Liy, and Sofie Pollin. 2015. An adaptive channel selection scheme for reliable TSCH-based communication. In *IEEE ISWCS*. 511–515.

[35] Alexandros Mavromatis, Georgios Z Papadopoulos, et al. 2016. Impact of Guard Time Length on IEEE 802.15.4e TSCH Energy Consumption. In *IEEE SECON*. 1–3.

[36] Yuri Murillo, Brecht Reynders, Alessandro Chiumento, Salman Malik, Pieter Crombez, and Sofie Pollin. 2017. Bluetooth now or low energy: Should BLE mesh become a flooding or connection oriented network?. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 1–6.

[37] Maria Rita Palattella, Nicola Accettura, Mischa Dohler, Luigi Alfredo Grieco, and Gennaro Boggia. 2012. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15.4e networks. In *IEEE PIMRC*. 327–332.

[38] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. 2013. On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH. *IEEE Sensors Journal* 13, 10 (2013), 3655–3666.

[39] K Pister and Lance Doherty. 2008. TSMP: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks* (2008), 391–398.

[40] Matti Siekkinen, Markus Hiienkari, et al. 2012. How low energy is Bluetooth Low Energy? Comparative measurements with ZigBee/802.15.4. In *Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 232–237.

[41] Rasool Tavakoli, Majid Nabi, Twan Basten, and Kees Goossens. 2015. Enhanced Time-Slotted Channel Hopping in WSNs using Non-Intrusive Channel-Quality Estimation. In *IEEE MASS*. 217–225.

[42] Andrew Tinka, Thomas Watteyne, and Kris Pister. 2010. A decentralized scheduling algorithm for time synchronized channel hopping. In *International Conference on Ad Hoc Networks*. Springer, 201–216.

[43] Nicolas Tsiftes, Simon Duquennoy, et al. 2016. The E-Care@Home Infrastructure for IoT-Enabled Healthcare. In *International Conference on IoT Technologies for HealthCare*. Springer, 138–140.

[44] Niall Twomey, Stephen Faul, and William P Marnane. 2010. Comparison of accelerometer-based energy expenditure estimation algorithms. In *4th Int. Conf. Pervasive Computing Technologies for Healthcare*. IEEE, 1–8.

[45] A. Vafeas, A. Elsts, J. Pope, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock. 2018. Energy-Efficient, Noninvasive Water Flow Sensor. In *Proc. 4th IEEE Int. Conf. Smart Computing (SMARTCOMP)*.

[46] X. Vilajosana, K. Pister, and T. Watteyne. 2017. *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*. RFC 8180. IETF.

[47] Q. Wang, X. Vilajosana, and T. Watteyne. 2018. *6TiSCH Operation Sublayer (6top) Protocol (6P)*. RFC 8480. IETF.

[48] Thomas Watteyne, Ana Laura Diedrichs, et al. 2016. PEACH: Predicting Frost Events in Peach Orchards Using IoT Technology. *EAI Endorsed Trans. on the Internet of Things* (2016).

[49] Thomas Watteyne, Lance Doherty, Jonathan Simon, and Kris Pister. 2013. Technical overview of SmartMesh IP. In *Innovative mobile and internet services in ubiquitous computing (imis), 2013 seventh international conference on*. IEEE, 547–551.

[50] Thomas Watteyne, Ankur Mehta, and Kris Pister. 2009. Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense. In *Proc. 6th ACM Symp. on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. ACM, 116–123. https://doi.org/10.1145/1641876.1641898

[51] Przemyslaw Woznowski, Alison Burrows, Tom Diethe, et al. 2017. 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, 315–333. https://doi.org/10.1007/978-3-319-44924-1_14

[52] ZigBee Alliance. 2006. ZigBee-2006 Specification. http://www.zigbee.org/. (2006).