Interference Graphs to Monitor and Control Schedules in Low-Power WPAN

Tim van der Lee[a], Antonio Liotta[b], Georgios Exarchakos[a]

[a] *Eindhoven University of Technology, the Netherlands*
[b] *University of Derby, United Kingdom*

Abstract

With billions of connected devices in the near future, the major challenge is to develop networks to build an Industrial Internet of Things which is scalable, energy-efficient, reliable and affordable. To this end, low-power wireless personal area networks (LP-WPAN) provide a solution at minimum costs. However, to ensure continuous performance verification, LP-WPAN requires a centrally monitored and controlled service. This work proposes such an edge service, i.e. network monitoring and optimal reconfiguration of scheduled LP-WPANs. The approach is based on a transformation of the schedule into a new model, *interference graphs*. The interference graphs allow to design evaluation and rescheduling recommender methods to monitor and reconfigure the schedule. An experimental setup was developed to test and validate the approach. The results show that the model and methods provide an accurate representation of the behavior of the network, and that the new rescheduling recommender greatly improves the network's performance, compared to random rescheduling.

Keywords:
Internet of Things, Wireless sensor network, Interference graph, disconnection probability, Network management, Network scheduling

Email addresses: t.lee@tue.nl (Tim van der Lee), a.liotta@derby.ac.uk (Antonio Liotta), g.exarchakos@tue.nl (Georgios Exarchakos)

1. Introduction

Industry 4.0 is tightening the dependability requirements of low-power wireless personal area networks (LP-WPANs) to support reliable control loops. In parallel, Internet of Things (IoT) is promising very large networks of interconnected devices smaller, smarter and more energy efficient than ever. LP-WPANs offer unique solutions to the high installation and maintenance costs of copper or optical networks in industrial settings. With the target of optimizing and distributing production lines, big data coming from IoT could help on taking more informed decisions on controlling production. The more temporospatially fine-grained the collected information is, the more wireless communication is required resulting in more contended wireless medium, thus, increasing the unpredictability of control loops. For maximum predictability as well as continuous performance verification, smart factories need LP-WPAN status to be centrally monitored and controlled. This requirement could be fulfilled by a network-as-a-service (NaaS), a network resource allocation and optimization service for LP-WPANs.

While active research is on-going to define efficient scheduling algorithms for LP-WPANs in TSCH mode, very little attention is drawn on *a priori* evaluating the performance of these schedules and on the rescheduling problem. To the best of our knowledge, this work is the first to propose a rescheduling method to adapt to external constraints and possible topology changes. This work proposes an edge service to assess the performance of the LP-WPAN and trigger a targeted maximum-profit minimum-cost re-configuration. LP-WPANs, specified by IEEE802.15.4 [1], in Time Synchronized Channel Hopping (TSCH) mode allow for more deterministic resource allocation. Namely, transmitters, receivers, channels and timeslots used may be determined by a network control algorithm e.g. built for a LP-WPAN NaaS.

To reduce uncertainty introduced by the latency between end nodes and the LP-WPAN controller, the latter is assumed at the edge. As part of that controller, scheduling algorithms should avoid internal and external interference. Given the context of this work, network topology volatility, no scheduler in literature considers the complete topology and deploys minimum rescheduling of maximum performance gains, rescheduling problem. A performance manager and rescheduling driver, is required. This work proposes an edge-service to analyze and optimize scheduled networks, divided into three main contributions.

- A transformation of the scheduled network into interference graphs,

- A rating method to evaluate the performance of the scheduled network based on interference graph analysis,
- A recommender to indicate which connection is the most problematic also based on interference graph analysis.

While this work applies to the IEEE802.15.4-TSCH communication protocol, it can be easily extended to any scheduled network, *mutatis mutandis*. In order to evaluate the accuracy of both the assessment and rescheduling methods, experiments are made in a real-world LP-WPAN deployment.

In section 2, background and work related to the TSCH communication protocol is explained. The architecture of the different components presented in this work is detailed in the following section. Then, definitions are provided to describe a scheduled network and introduce the concept of interference graph. In section 5, our proposed approach presents an analysis of interference graphs to evaluate and pinpoint problematic connection in a scheduled network. A traffic aware scheduler is also presented. In section 6, experiments are run to confirm the validity of the rating method, and the rescheduling method is tested on the previously described traffic aware schedule. The paper concludes with a proposal for future work on this topic.

2. Background and related work

One of the main challenges in upscaled IoT is to provide reliable communication at the lowest possible energy cost. To this end, a wireless sensor networks (WSN) is preferred as it provides more flexibility and security than using the internet [2]. WSNs are already used in various application domain such as factory automation [3], distributed and process control [4, 5], smart grid [6] or in the healthcare domain [7, 8, 9, 10]. In the past few years, different technologies have attempted to address the reliability, energy efficiency, scalability and flexibility of WSNs. This includes new communication standard such as bluetooth [11], ZigBee [12], and the IEEE802.15.4 standard presented in the next paragraph. At the same time, new protocols are introduced by the Internet Engineering Task Force (IETF) such as 6LoWPAN[13], providing IPv6 for low-power wireless personal area networks.

2.1. The IEEE802.15.4-TSCH protocol

The IEEE802.15.4-2016 standard is designed for low-power, low-cost and low-rate network deployments. First published in 2006 [1], the standard

introduces the CSMA-CA algorithm, a communication protocol vulnerable to interference and multi-path fading [14, 15]. The unlimited delay possibilities of CSMA-CA makes it too unreliable for industrial deployment. The 2012 amendment introduces a new mode, Time Slotted Channel Hopping (TSCH), that has been proven to avoid interferences and more energy-efficient [16, 17].

TSCH enables time-slotted access to the communication layers of the device, and also allows to communicate with up to 16 different channels available at 2.4MHz. For each connection in the network, the time of the transmission has to be specified (*timeslot*), as well as the channel offset. The different devices of the scheduled network are kept synchronized through the emission of small packets, called enhanced beacons (EBs). The multichannel communication paradigm mitigates interference and congestion in the WSN [18]. The TSCH standard also introduces a channel hopping mechanism. When transmitting, the frequency f to be used for this connection is chosen as follows.

$$f = (ASN + channeloffset) \mod C$$

where ASN is the absolute slot number, that is the number of timeslots elapsed since the start of the network, and C is the total number of channels available. If communication occurs every x timeslots, x being prime with C, it is ensured that all available channels are used. Thanks to channel hopping, the scheduled network is less impacted by external perturbations [19, 20].

2.2. The scheduling problem

The schedule can be computed centrally, or in a distributed way [21]. TSCH minimal is the minimal TSCH schedule provided by the IETF [22]. It consists of a single shared cell used to both send and receive any type of packet. TASA, the *Traffic Aware Scheduling Algorithm* is a centralized scheduler based on edge coloring technique [23], reputated to be interference free, but limited when it comes to scalability. A decentralized version of TASA, DeTAS provides more scalability than TASA [24]. Orchestra is a decentralized algorithm implemented in contiki [25] providing high reliability in various test environments.

For all connections in a scheduled network, timeslot and channel offsets have to be specified. To avoid redundancy of information, they are specified for a certain amount of timeslots, composing a *slotframe*, sometimes referred to as *superframe*, which is repeated over time. During a timeslot, typically 10ms-15ms, a device can either transmit or listen over a certain channel offset,

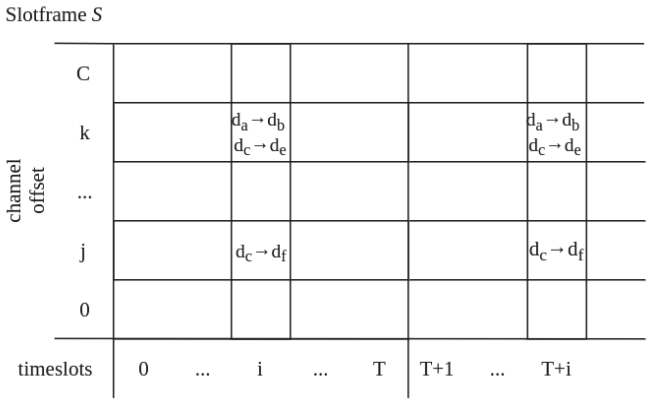


Figure 1: Example of slotframe containing a case of interference.

or remain idle. When transmitting an unicast packet, an acknowledgement is expected to be received. If absence of an acknowledgement, the device will attempt to retransmit at the next available possible timeslot. After the maximum number of retransmission is reached, the packet is dropped and considered as lost. In the following, a communication occurring at a certain timeslot and channel offset will be referred as connection.

The slotframe is potentially different for every devices. As an example, if at timeslot i and channel offset j device d_1 communicates with d_2, the cell (i, j) of the slotframe of d_1 will have to be a transmitting cell, and the cell (i, j) of device d_2 will have to be a listening cell. In this work, we assume an edge service to have full knowledge of the network. Therefore, we will consider only the slotframe of the scheduled network - cell (i, j) of the slotframe of the network will be $d_1 \rightarrow d_2$. The slotframe of the scheduled network is regarded as a matrix of T timeslots and C channel offsets. Filling this matrix with appropriate connections is called *scheduling* the TSCH network. The scheduling process is explained in [26]. Scheduling a TSCH network without interference is an NP-hard problem [27]. An example of a network slotframe is given by figure 1. In this example, a case of interference is present on cell (i, k). Two communications are occurring at the same time and channel, and assuming all devices are in interference range, they will interfere. There is also a case of interference at timeslot i, since d_c can not transmit to two different devices at the same time.

Using a different scheduler will affect the network in terms of latency, scal-

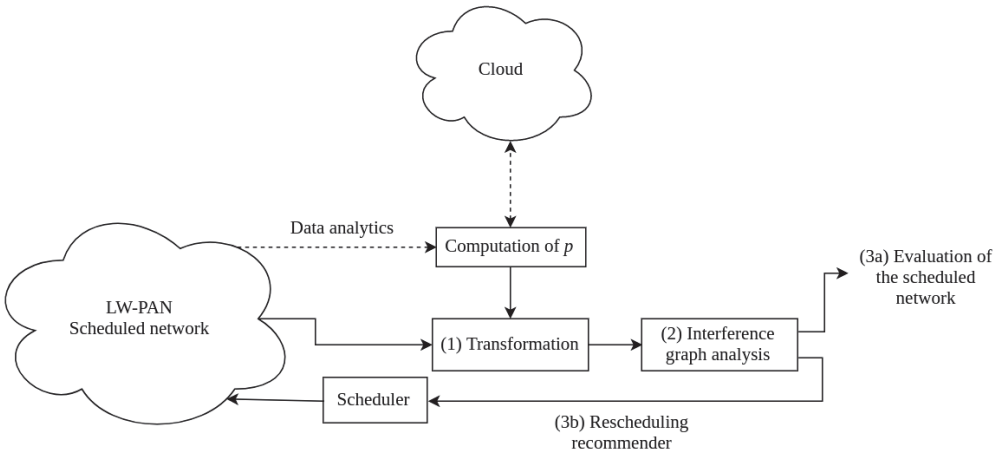


Figure 2: Schematic overview of the architecture of the proposed edge service.

ability, energy consumption and reliability. While most of these scheduling algorithms address these issues independently, there is not yet a scheduling algorithm able to handle all of them together, for any network. Most of the time, rescheduling is required to fine-tune the performance of the network according to the requirements of the use-case.

3. Architecture overview

Figure 2 summarizes the different components of the edge service proposed in this study, and how they interact. The scheduled LP-WPAN is first transformed into interference graphs (1), which rely on a key parameter p, the *disconnection probability*. p can be computed in different ways, e.g. machine learning. In this study, p will be evaluated through the analysis of experimental network-data. The second step is to analyze the interference graph (2), which leads to two outputs:

(3a): An evaluation of the scheduled network,

(3b): a recommender that indicates which connection to reschedule.

Evaluating a scheduled network can be used to predict and prevent failure in the scheduled network. It can also be used to compare two different schedules for the same network, and determining which one is best suited for the topology. The rescheduling recommender informs the scheduler of which

connection is the most problematic in the network. This information is used to directly change the schedule of the network and increases its reliability at runtime.

Deploying such an edge service in a real world deployment can be done in different ways. In our experimental setup, p is predicted based on the traffic pattern of the network, and does not require any interaction with the cloud, or the network. We perform step (1), (2) and the scheduling/re-scheduling process in the cloud. Schedules computed outside the LW-PAN network can be exchanged with the nodes with CoAP services such as plexi [28]. In our deployment, step (1) has to be performed outside the network because the devices used (JN5168) do not have enough RAM memory to analyze the entire network. However, depending on the capacity of the devices used, the different blocks presented in figure 2 can be ported to the device.

Using only standard graph analysis techniques, interference graph analysis is not computationally expensive. However, having one node per connection in the interference graph, step (1) and (2) may be expensive in terms of memory usage, if a lot of connections are scheduled. Therefore, we recommend to perform these tasks on non-constrained devices to address scalability issues.

4. Network model

This section provides the definitions and models needed to solve the rescheduling challenge of TSCH networks. First, a model of the TSCH scheduled network is given, followed by the definition of the interference graph, a representation of the TSCH scheduled network used in the presented approach. A table presenting the different acronyms and symbols used in this publication is given table 1.

4.1. TSCH scheduled network

A TSCH scheduled network is defined as a network of wireless devices using the TSCH communication protocol. We assume that the devices of this scheduled network are scheduled on 1 slotframe. If several sloframes are defined, it is always possible to find one slotframe, based on a combination of slotframes that will lead to the same behavior. Since slotframes are repeated over time, analyzing the behavior of the network during the entire duration of the slotframe is the same as analyzing the network at any time, assuming the environment does not change.

Table 1: Symbols and acronyms used

S, SF	Slotframe
C	Available channels for S
T	Available timeslots for S
$c_{i,j}, cell$	Cell of a slotframe
G	Connectivity graph model
F	Interference graph model
d	Device model in G
$e = (d_1, d_2)$	Connection between 2 devices in G
$l = (e_1, e_2)$	Link between 2 connections in F
$p(l)$	Disconnection probability (weight of l)
$\rho(X)$	Weighted density of a graph X
$deg(x)$	Node degree of a node x
PRR	Packet Reception Ratio

A TSCH scheduled network is composed of two main information elements, **topological** and **temporal**, and knowing both gives a full understanding of the scheduled network. The spatial information concerns the connectivity of the network, and informs us about all connections between the devices and the channel on which they occur. It also includes interfering devices, since the interference range may be greater than the direct connectivity range. Connections occur during a timeslot, hence the spatial information may change over time. The temporal information is there to keep track of these network changes in time.

Let us consider a TSCH scheduled network consisting of n devices communicating according to a slotframe S, with T timeslots and C channels. To represent the network we will use an approach similar to [23]. In this document, the TSCH network is represented as an directed graph $G = (V, E)$ where $V = \{d_0, d_1, ...d_{n-1}\}$ are the devices and $E \subset V \times V$ the connections of the network. As an example, a unicast connection from device d_a to d_b will be noted as $e = (d_a, d_b)$. Similarly, a broadcast connection initiated by device d_a will be a set of connections noted $\{(d_a, d_k), \forall k \in DN(d_a)\}$ where $DN(d)$ represents the set of devices that are in direct range of d. It is not always the case that direct range and interference range coincide.

This connectivity graph does not take into account the temporal information given by the slotframe. Therefore we introduce the slotted connectivity

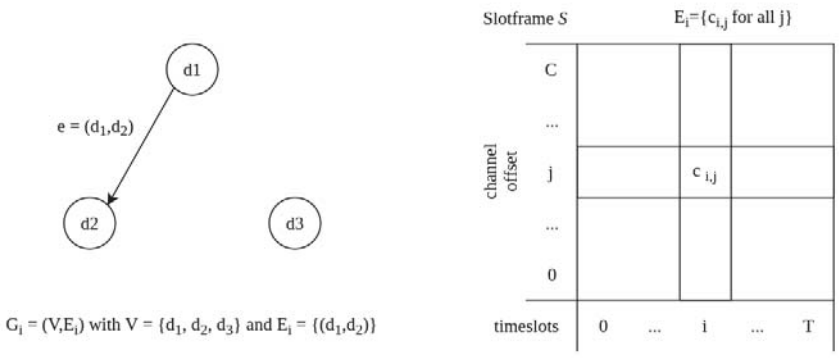


Figure 3: Construction method of the slotted connectivity graph.

graph G_i of a timeslot i. In the slotframe S of our scheduled network, each cell $c_{i,j}$ at timeslot i and channel offset j will indicate connections occurring at this timeslot. If no connections are occuring, c will be the empty set $\emptyset$. Thus, we can define the **slotted connectivity graph** of a scheduled network $G = (V, E)$ with a slotframe $S = \{c_{i,j}, \forall (i,j) \in T \times C\}$ at a timeslot i as

$$G_i = (V, E_i) \text{ with } E_i = \{e \in c_{i,j}, \forall j \in C\} \tag{1}$$

Figure 3 represents the construction method of the slotted connectivity graph. For a given slotframe S, all connections occuring at timeslot i are taken into account to build G_i. Following the definition of the slotted connectivity graph, the **complete connectivity graph** is defined as follows.

$$G^* = (V, E) \text{ with } E = \{e \in E_i, \forall i \in T\} \tag{2}$$

Similarly as G, G^* is a directed graph that does not take into account the temporal information given by the slotframe. However, G^* can be a multigraph, since the same connection may be present on multiple timeslots of the slotframe. Considering a set of devices $d \in V$ scheduled with the slotframe S, the information about the direct range of the devices is enough to construct G_i for all i and thus G^*. However, in order to study the impact of internal interference in a scheduled network and assess accurate deployments, we need to take into account devices in interference range. The following section describes a model based on G that takes into account interfering devices information.

4.2. Interference Graph

The interference graph is a representation of the scheduled network taking into consideration the possible interference and causes for packet loss, which can occur internally. First introduced in [29], its definition is refined in this work. Within a slotframe S, two connections will be considered as **conflicting** iff, on the same timeslot, a device has two different connections scheduled. The TSCH communication protocol will prioritize one of the connections, leading to a potential packet loss for the other one. Interference occurs when two different pairs of devices communicate on the same timeslot and channel. In this case, packets may be lost due to interference on the receiving side, assuming there is no clear channel assessment mechanism in place. In the remainder of this article, both conflicts and interference will be referred to as interference.

In the same way as the slotted and complete connectivity graphs were defined, we define a slotted, and a complete interference graph. Nodes of these graphs are connections occurring in the TSCH scheduled network, and these nodes are linked if the corresponding connections interfere. The slotted interference graph at timeslot i corresponding to a slotted connectivity graph G_i can be first defined as follows:

$$F_i = (E_i, L_i) \text{ with } L_i = \{(e_1, e_2) \in E_i^2 \text{ and } (e_1, e_2) \in L\}$$

F_i is a directed graph that illustrates the behavior of the network at timeslot i in terms of internal interference. However, not all interfering connections have the same impact on the performance of the network. As an example, if two unicast connections are interfering but have no traffic assigned, the impact on the performance of the scheduled network would be zero. Therefore, links of the interference graph are weighted with a weight factor $p : L \longrightarrow [0, 1]$ determining the impact of this interference on the global performance of the network, called *disconnection probability*. For simplicity, we do not consider interfering connections that have no impact on the behavior of the network. If $p(l) = 0$, the link l is removed from the interference graph. Note that for (e_1, e_2) and (e_2, e_1) in L we may have $p((e_1, e_2)) \neq p((e_2, e_1))$. The complete definition of the **slotted interference graph** at timeslot i has to include p.

$$F_i = (E_i, L_i, p) \tag{3}$$

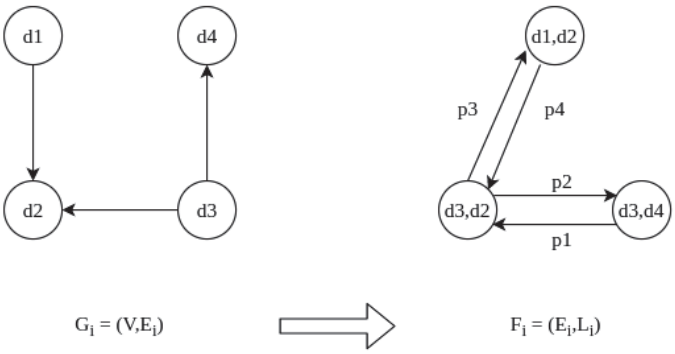


Figure 4: Construction method of the interference graph assuming all channels are different.

Figure 4 represents an example of transformation from the slotted connectivity graph G_i to the slotted interference graph F_i. Assuming all channels are different, the slotted connectivity graph G_i presents a conflict since device $d3$ cannot handle two transmitting operations at the same time. Thus, connection $d3, d2$ and $d3, d4$ are linked with different disconnection probabilities p. Similarly, device $d2$ can not listen to two connections at the same time, therefore linking the nodes $(d1, d2)$ and $(d3, d2)$.

The **complete interference graph** is defined in the same way as the complete connectivity graph. All connections are represented with a node in the graph, and all interfering connections are linked with a directed link of weight p.

$$F^* = (E, L, p) \tag{4}$$

where $L = \bigcup_{i \in T} L_i$. As for the complete connectivity graph, the complete interference graph may be a multigraph, since the same interference can occur several times in the slotframe.

5. Proposed approach

In this section, we first present a method to analyze the previously defined interference graph. This provides a rating method allowing us to evaluate the performance of a scheduled network. Then, this rating method is used to define a rescheduling technique applied on a traffic aware schedule.

5.1. Rating method

Our rating method aims to evaluate schedules in given topologies based on internal interference. Fortunately, the weighted links of the interference graphs precisely represent interference we want to detect. The weight p of these links also indicates the impact of the interference on the performance of the scheduled network. Consequently, we will rate schedules according to the weighted density of the interference graph. The density of a graph is the result of the number of links divided by the number of possible links. For a weighted graph such as $F^* = (E, L, p)$, we will take the density according to its definition in [30].

We define the **density** ρ of the complete interference graph or the slotted interference graph at timeslot i as follows:

$$\forall i \in T,\ \rho(F_i) = \frac{\sum_{l \in L_i} p(l)}{|E_i|(|E_i| - 1)} \tag{5}$$

$$\rho(F^*) = \frac{\sum_{l \in L} p(l)}{|E|(|E| - 1)} \tag{6}$$

ρ will be considered as our rating metric for schedules. An interference-free schedule will present no links in its interference graphs, and thus will obtain a density equal to 0. Since F^* may be a multigraph, its density is unbounded. The density, and thus the rating method, depends on the parameter p that can be described as the probability that the interference impacts the proper behavior of the scheduled network. Defining the function p as accurately as possible is required to rate different schedulers between each other and in different topologies. Furthermore, performing an analysis of the interference graph knowing beforehand the value of p, will allow to predict the performance of the schedule in a given topology. In this work, we focus on the evaluation of one scheduler in one topology. Our first approach to define p is explained in Section 6.1. A more precise definition of p will be provided in a subsequent paper.

5.2. Traffic-aware scheduler

In this section, we introduce a centralized scheduling algorithm. This generates a traffic-aware slotframe for the network, which is not interference free. A rescheduling method to make it interference free is presented in the next section. While the rescheduling method applies on any scheduler, we will test it on this particular traffic-aware schedule.

Let's define the number of frames to be sent as the traffic. The first step is to determine how many cells should be assigned to a connection to comply with the traffic load of the network. To establish a traffic-aware schedule, the amount of traffic generated by each node and the routing topology must be known. We note $q_{a,b}$ the traffic generated per slotframe by d_a for d_b and $route(d_a, d_b)$ the route from d_a to d_b.

Algorithm 1: Function for traffic aware cell attribution

```
Function get_cells(d_a,d_b)
    cells ← q_{a,b}
    for d_i,d_j in all devices do
        if (d_i, d_j) in route(d_a, d_b) then
            cells ← cells + q_{i,j}
        end
    end
    return cells
```

The function get_cells presented in Algorithm 1 computes the minimum number of cells required for the connection (d_a, d_b). One cell is theoretically enough to transport one frame per slotframe, and keep the queues empty. However, in a real deployment with external perturbations and interference, the number of cells must be increased. This function determines how many cells are required to reach a traffic-aware schedule, but the scheduler allocates these cells randomly. Therefore, the resulting slotframe is potentially subject to internal interference. This scheduler accepts request from external services to reschedule certain connections to empty cells.

5.3. On-the-fly rescheduling method

In this section, we present a rescheduling method that selects a problematic connection. The rescheduling method is based on the analysis of the slotted interference graph. In a scheduled network with internal interference, we will note F_i for $i \in T$ the slotted interference graphs, and the density of all slotted interference graphs $\rho(F_i)$ is calculated by equation 5. $\max_i(\rho(F_i))$ defines the most problematic timeslot in terms of internal interference, and must receive priority.

In the following, we note $F_m = (E_m, L_m, p)$ the slotted interference graph with the highest density. In this interference graph, the conflicting connections are linked with a weight p. In order to find the most problematic connection, we compute the weighted outdegree of each node in the graph

$deg^+(e)$, i.e. the number of edges going out of this node with their respective weight p. The connection with the highest degree is chosen to be rescheduled first, and is rescheduled to an unused cell for this device. If there are several candidates, the connection is chosen randomly. To summarize, the rescheduling process is split in three main parts:

- Find the most problematic m with $max_i(\rho(F_i))$,
- Find the most problematic connection with $deg^+(E_m)$,
- Indicate to the scheduler that this connection needs to be rescheduled.

The algorithm will converge to an interference-free schedule for the topology, if it exists. Because cells are re-allocated randomly, this process can become computational-intensive. For example, if a connection can only be scheduled on 1 specific cell, the probability to pick this cell is $\frac{1}{T \times C}$ where T is the number of timeslots and C the number of channels in the slotframe. Here we focus on rescheduling the most problematic connections first. Preliminary results for a more refined method, based on choosing timeslots, show that for a distributed scheduler, choosing the timeslot is more efficient with messaging the neighboring devices [31].

The proposed method uses the interference graphs F_i, hence depends on p. In some cases where p is constant and the network topology is fixed, the rescheduling process is almost immediate. However, if the topology is changing or if p takes a long time to compute, i.e. p based on packet reception ratio or p estimated with machine learning, the rescheduling process can become computational intensive.

6. Real-world experiments

To evaluate the rating method and the rescheduling method applied to the traffic-aware schedule, several experiments have been designed using a real environment. In a following section, the experimental setup is described. Then the rating method is analyzed and finally the performance of the rescheduling method compared with the random rescheduling method.

6.1. Experimental setup

In order to run the experiment, p has to be defined first. The goal is to evaluate and reschedule a slotframe in a given topology to reach an

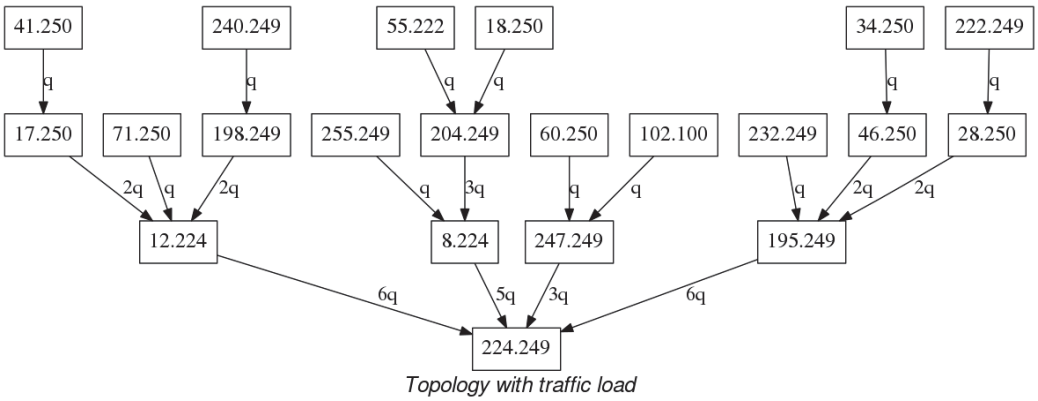


Figure 5: Topology of the devices with traffic load.

interference- and conflict-free slotframe. Taking $p = 1$ assumes that all interfering connections are equally affecting the good behavior of the network. However, depending on the traffic of these connections, the impact may be completely different. Therefore, we will take p based on the amount of packets in queue for both interfering connections. If we consider two interfering connections e_a, e_b in L, and the number of packets to be sent for a connection e as $q(e)$, p is defined as follows:

$$p((e_a, e_b)) = \frac{q(e_a) + q(e_b)}{\max_{(e_1,e_2)\in L}(q(e_1) + q(e_2))} \tag{7}$$

Then, it satisfies the requirement $0 < p \leq 1$. In this configuration, we also have $p((e_a, e_b)) = p((e_b, e_a))$. The experimental testbed is composed of 21 Jennic JN5168 devices in a tree topology. Devices use the latest version of Contiki with TSCH enabled. Figure 5 represents the topology on which the devices are tested. In today's IoT deployments, most low-power personal area networks (LW PAN) applications rely on a PAN coordinator to interface with devices outside the network. In our configuration, device 224.246 is the PAN coordinator, and all devices are transmitting packets towards it. All devices generate q packet per slotframes and forward all incoming traffic to their parent 224.246. Also, all devices are deployed in the same room and thus, are in interference range of each other.

The function presented in Algorithm 1 returns the theoretical minimum number of cells to be attributed per link. As we are in a real deployment, we allocate three times more cells than the number indicated. To simplify,

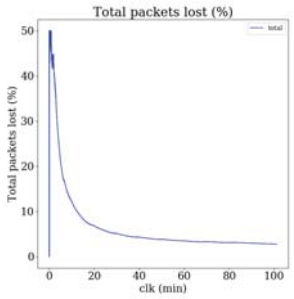
we divide the slotframe size by 3. Thus, $q = 0.3$ packets per slotframe. With timeslots of 15ms, the final slotframe size is 200 timeslots, and each node generates a packet every 9sec. In this way, the traffic load of each connection (see figure 5) corresponds to the number of cells to be attributed. All connections related to data are made on dedicated cells. The TSCH network is synchronized through EBs, additional messages that may occupy the queues and create interference. In order to focus the study on the data traffic only, EBs have a separate slotframe and channel. This way, results do not include data generated by EBs. The slotframe reserved for data is using the 15 other channels provided by the 802.15.4 communication protocol.

In the following sections, experiments are run to confirm the validity of both rating and rescheduling methods. In these experiments, the same topology and the same connections will be used but with different schedules. The gathered data does not vary after 1.5 hours, therefore experiments last between 1.7 to 2 hours. In all experiments, the number of retransmissions is set to 0, meaning that each packet has only one chance to be transmitted. If the transmission fails due to interference or external perturbations, the packet is dropped. This allows us to directly see the impact of perturbations in the network.

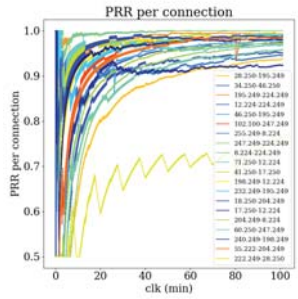
6.2. Rating method verification

Analyzing the density of the complete interference graph provides feedback on the amount of interference present in the network. For this feedback to be visualized, we generated four slotframes with different numbers of interfering connections.

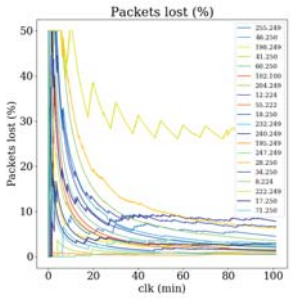
The first slotframe, $SF1$, contains no interference: it is an interference-free slotframe. For this topology, a random generated slotframe has a 16% chance of being interference-free. Experiments are run to analyze the behavior of the network. Metrics such as the packet reception ratio per link and the number of packet lost are displayed on figure 6. Since our experimental setup does not allow retransmissions, the packet reception ratio (PRR) per connection is not equal to 1, but tends to stabilize between 90% and 99% for all connections except one. It appears from figure 6c and 6b that device 198.249 exits and re-joins the network frequently. The network is subject to external interference, or perturbations in the hardware itself that causes transmission to fail. This explains why the perfect slotframe does not obtain a 100% PRR. However, the $SF1$ is very effective as its end-to-end PRR



(a) Packets lost for $SF1$ as a percentage of the total number of packet emitted by the network.



(b) Packet reception ratio per connection for $SF1$.



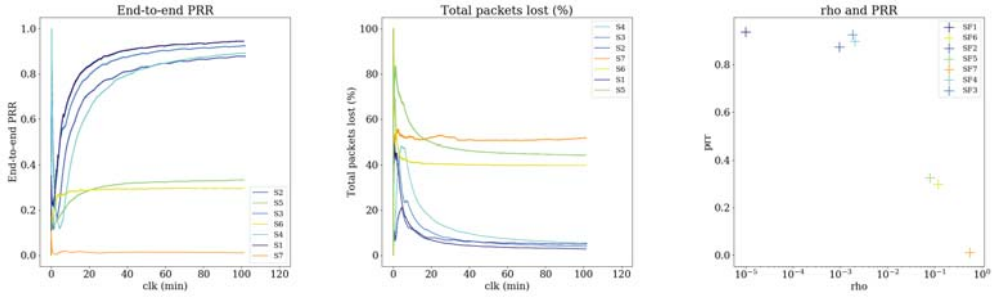
(c) Packets lost per connection for $SF1$ as a percentage of the number of packet sent through this connection.

Figure 6: Results of the performance analysis of $SF1$.

reaches 93.7%. Since the slotframe is interference-free, the densities of its interference graphs are 0.

The next tested slotframes are $SF2, SF3$ and $SF4$. $SF2$ presents one conflict for the device 8.224. $SF3$ and $SF4$ present both 2 problems: $SF3$ shows a conflict and a case of interference, and $SF4$ presents two conflicts. The probability of this occurring is approximatively 27%. When generating random schedule, this is the second most common situation. Generating a slotframe with 1 interference occurs with a probability of approx. 31%. For $SF2$ we obtain $\rho(F^*) = 0.00097$ and $PRR = 0.87$. We have for $SF3$ $\rho(F^*) = 0.00184$ and for $SF4$ $\rho(F^*) = 0.00203$. The difference between the impact of the interference detected by ρ is correctly observed when looking at the end-to-end PRR. For $SF3$, $PRR = 92.5\%$ and for $SF4$, $PRR = 89.8\%$. However, both $SF3$ and $SF4$ obtain a higher PRR than $SF1$ while having a higher density. This imprecision is due to the fact that p does not take into account if the conflicting connections are two receiving cells, two transmitting cells or a mix of both. Due to the implementation of TSCH, the case of two conflicting receiving cells present in $SF2$ is often subject to more packet loss than the other cases.

The measure of ρ seems to be correlated with the end-to-end PRR, three more slotframes were tested to confirm this hypothesis. $SF5$, $SF6$ and $SF7$ are unrealistic slotframes generated for the purpose of the experiment, as they contain 136, 190 and 861 cases of interference respectively. For $SF5$, we obtain $\rho(F^*) = 0.0789$ and an end-to-end PRR of 32.5%. For $SF6$,



(a) End-to-end packet reception ratio for all tested slotframes.

(b) Total number of packets lost for all tested slotframes.

(c) Relation between ρ and PRR for all tested slotframes.

Figure 7: Performance of all slotframes in relation with ρ, taken to 10^{-5} precision.

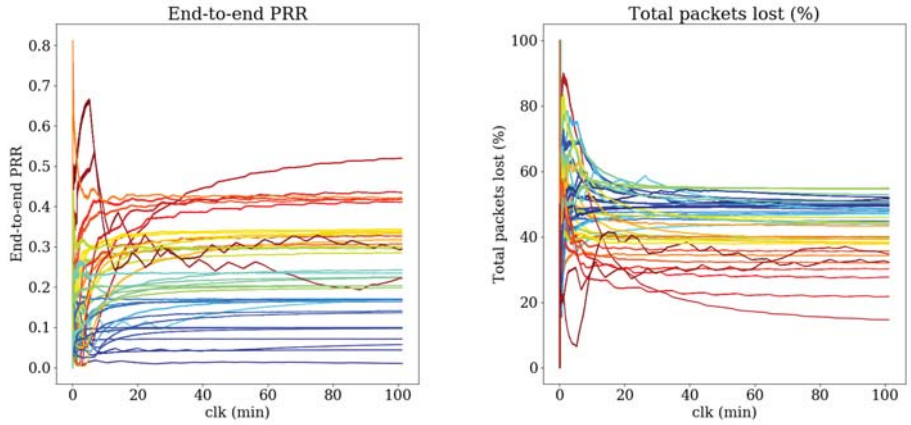
$\rho(F^*) = 0.1180$ and $PRR = 29.8\%$, and for $SF7$, $\rho(F^*) = 0.5528$ and $PRR = 1.02\%$.

Figure 7 indicates for all experiments, the end-to-end PRR, total packets lost and the relation between ρ and the PRR. Between all experiments, a correlation between $\rho(F^*)$ and the amount of packets lost seems to appear. Similarly, a high value of ρ corresponds to a low end to end PRR and thus, a poor global network performance. This relation is visible in figure 7c.

6.3. Rescheduling method verification

In order to visualize the performance of rescheduling, we will consider a scheduler, initializing the TSCH network with a traffic-aware schedule as presented in Section 5.2. The traffic-aware schedule positions cells in the worst possible way, with connections scheduled on timelot 0 and channel 0. However, this scheduler accepts recommendations from an external service to reschedule specific connections. As all connections interfere, all connections need to be rescheduled. We run two sets of experiments. In the first set, the connection to reschedule is chosen randomly among all interfering connections. In the second set, the connection to reschedule is chosen with interference graph analysis, as explained in Section 5.3. In the following, we refer to the first set as *random rescheduling* or rescheduling *without* interference graph analysis. The second set is referred to rescheduling *with* interference graph analysis. In both sets, the connection is rescheduled randomly to an available cell.

Figure 8 represents the performance of random rescheduling in the network, each figure representing 40 iterations. The colors are distributed ac-



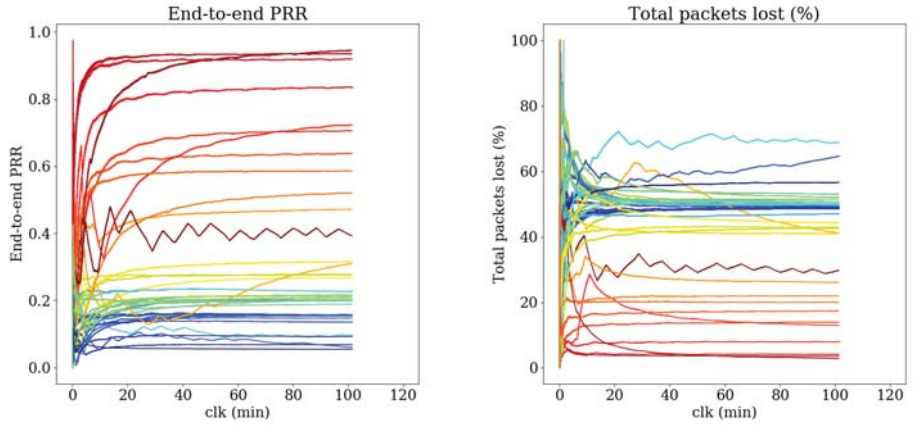
(a) End-to-end packet reception ratio of each iteration.

(b) Number of packets lost for all iterations in percentage of the total number of packet sent.

Figure 8: Results of the first 40 iterations of the random rescheduling method.

cording to the matplotlib *jet* colormap [32]. Initial iterations are represented in dark blue, and latest iterations are in dark red. In the first iterations of random rescheduling, the packet reception ratio is close to 0, and more than half of all packets are lost. Most connections are indeed interfering since all connections are initially scheduled on timeslot 0 and channel 0. After 40 iterations, the number of packets lost drops to 17%. However, even if almost all connections are rescheduled, the packet reception ratio does not exceed 60%. Connections that are not rescheduled are critical connections in the network ($8.224 - 224.249$ and $247.249 - 224.249$), and therefore are responsible for important packet loss. While most of the experiments produce a stable end-to-end PRR and packet loss, some measures appear to be irregular. This is due to the fact that real-life experimentation are subject to unpredictable hardware failure or external perturbations.

Results of rescheduling using the interference graph analysis are represented by figure 9. Colors are also distributed similarly as the previous set of experiment. The first experiment is represented in blue, and presents significant interference: more than 50% of all packets are lost, and the end-to-end PRR does not exceed 10-15%. However, after several iterations, the PRR and amount of packets lost are significantly increased. The rescheduling method focuses on first rescheduling the most critical connections according to the interference graph. In this set, the most problematic timeslot is



(a) End-to-end packet reception ratio of the network for each iteration.

(b) Percentage of packets lost of the global network for each iteration.

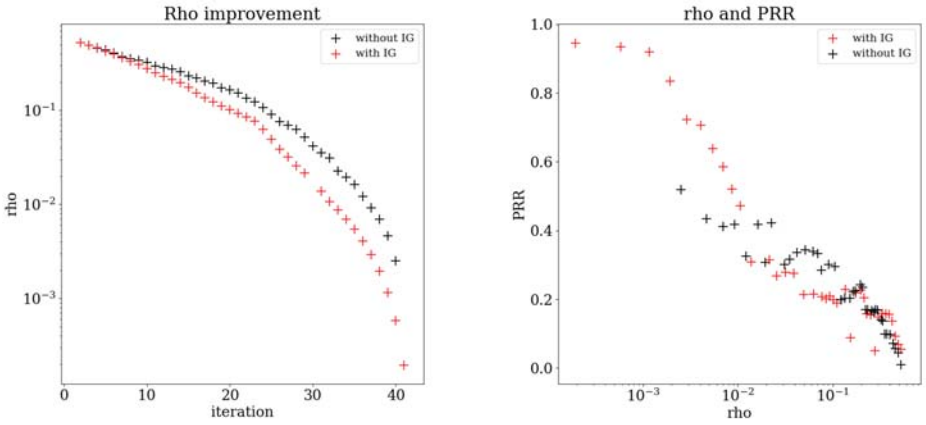
Figure 9: Results of the first 40 iterations of the rescheduling method based on interference graph analysis.

always timeslot 0 since all connections are scheduled on this timeslot initially. The interference graph analysis determines which connection has to be rescheduled based on the weighted out degree evaluation. With p based on the queues, connections close to PAN coordinator are chosen first, leading to a significant PRR and packet loss improvement after most of them are rescheduled.

Both methods are compared and represented in figure 10. Figure 10a represents the difference of improvement of the two methods for every iteration. We can see that rescheduling using the interference graph analysis improves the evaluation of the global network (density of the complete interference graph) faster than random rescheduling. Figure 10b represents the evolution of the PRR with the evaluation $\rho(F^*)$ of the schedule. Initially, both methods present a similar improvement. However, after 30 iterations, or when $\rho(F^*) < 10^{-2}$, the rescheduling method with interference graph analysis presents a better performance than the random rescheduling method.

6.4. Discussion

The first set of experiments compares random-generated slotframes. The density ρ applied on the complete interference graph F^* provides different values for these slotframes, as they present a different amount of internal



(a) Improvement of the rating metric ρ over 40 iterations.

(b) Packet reception ratio as function of the measure ρ for both experiment sets.

Figure 10: Side-by-side comparison the random rescheduling method (black) and the rescheduling method with interference graph analysis (red).

conflicts and interference. From the experiments, the packet reception ratio and the amount of packets lost appears related to the density calculated. In the second set of experiment, a slotframe fully conflicting is rescheduled. The density and the slotted interference graphs are used in order to determine which connections have to be rescheduled first. Compared with random rescheduling, the use of interference graphs improves significantly the rescheduling process.

In both sets of experiments, the performance of the network is measured with the PRR. In this regard, the density of the interference graph model provides a good prediction of the performance of the global network. However, in both sets of experiments, unpredicted failures occurred, resulting in an oscillating PRR that can be seen in some iterations of figure 8a or 9a. This creates small differences between the predicted performance and the actual performance of the global network. This difference is due to the fact that the chosen disconnection probability p is calculated based on the estimated traffic of each connection. It does not predict hardware failures, nor external perturbations, which can greatly impact the network since there are no retransmissions allowed in our setup.

7. Conclusion and future work

With an industrial IoT landscape growing exponentially, network volatility in scheduled networks and reliability in terms of interference need to be addressed. This work proposes an innovative edge service to address this challenge based on interference graph analyses, a method to pinpoint which connection requires to be rescheduled. Evaluating the weighted density of the complete interference graph allows to rate the scheduled network in terms of internal interference. The approach is validated with experimental results, showing that the density of the interference graph is accurately representing the amount of internal interference in the scheduled network. The accuracy of the rating method can be further improved with the *disconnection probability*, p. The rescheduling recommender experiments demonstrate that the new rescheduling method is superior to random rescheduling. It is also an improvement for the global performance of the network.

Potential for further improvement can be proposed. In the experiments, p is based on the queue size of both interfering connections and does not take into account which type of interference is occurring, or if cells are shared or dedicated. A better definition of p will therefore improve both evaluation and rescheduling methods. Since this service runs on the edge of the network, there are no resource constraints for the computation of p. Knowing preemptively p for a specific environment, and thus ρ, would allow to predict the performance of a schedule in this environment.

Different environments will have different impact on the performance of the scheduled network. In this work, experiments were run in the same room, with the same devices and with the same network topology. More experiments have to be conducted to refine the definition of p, and validate the evaluation method in any topology.

The edge service proposed in this study improves the performance and reduces the energy footprint of the scheduled network. Interference is responsible for retransmissions, an avoidable energy cost.

Acknowledgment

This research has been funded by the European Unions Horizon 2020 project INTER-IOT (grant number 687283), and was carried out at the Eindhoven University of Technology in the Netherlands.

[1] IEEE Standard for Low-Rate Wireless Networks, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) (2016) 1–709doi:10.1109/IEEESTD.2016.7460875.

[2] C. Alcaraz, P. Najera, J. Lopez, R. Roman, Wireless sensor networks and the internet of things: Do we need a complete integration?, in: 1st International Workshop on the Security of the Internet of Things (SecIoT10), 2010.

[3] A. Willig, Recent and emerging topics in wireless industrial communications: A selection, IEEE Transactions on industrial informatics 4 (2) (2008) 102–124.

[4] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, S. S. Sastry, Distributed control applications within sensor networks, Proceedings of the IEEE 91 (8) (2003) 1235–1246.

[5] V. C. Gungor, G. P. Hancke, Industrial wireless sensor networks: Challenges, design principles, and technical approaches, IEEE Transactions on industrial electronics 56 (10) (2009) 4258–4265.

[6] V. C. Gungor, B. Lu, G. P. Hancke, Opportunities and challenges of wireless sensor networks in smart grid, IEEE transactions on industrial electronics 57 (10) (2010) 3557–3564.

[7] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi, R. Jafari, G. Fortino, From modeling to implementation of virtual sensors in body sensor networks, IEEE Sensors Journal 12 (3) (2012) 583–593.

[8] A. Milenkovi, C. Otto, E. Jovanov, Wireless Sensor Networks for Personal Health Monitoring: Issues and an Implementation, Comput. Commun. 29 (13-14) (2006) 2521–2533. doi:10.1016/j.comcom.2006.02.011. URL `http://dx.doi.org/10.1016/j.comcom.2006.02.011`

[9] G. Fortino, M. Pathan, G. Di Fatta, Bodycloud: Integration of cloud computing and body sensor networks, in: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, IEEE, 2012, pp. 851–856.

[10] R. Gravina, P. Alinia, H. Ghasemzadeh, G. Fortino, Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges, Information Fusion 35 (2017) 68–80.

[11] S. Bluetooth, Bluetooth core specification version 4.0, Specification of the Bluetooth System.

[12] Z. Alliance, Ieee 802.15. 4, zigbee standard (2009).

[13] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, Rpl: Ipv6 routing protocol for low-power and lossy networks, RFC 6550, RFC Editor, http://www.rfc-editor.org/rfc/rfc6550.txt (March 2012).
URL http://www.rfc-editor.org/rfc/rfc6550.txt

[14] J. T. Adams, An introduction to IEEE STD 802.15.4, in: 2006 IEEE Aerospace Conference, 2006, pp. 8 pp.–. doi:10.1109/AERO.2006.1655947.

[15] D. D. Guglielmo, G. Anastasi, A. Seghetti, From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things, in: S. Gaglio, G. L. Re (Eds.), Advances onto the Internet of Things, no. 260 in Advances in Intelligent Systems and Computing, Springer International Publishing, 2014, pp. 135–152, dOI: 10.1007/978-3-319-03992-3_10.
URL http://link.springer.com/chapter/10.1007/978-3-319-03992-3_10

[16] T. Watteyne, J. Weiss, L. Doherty, J. Simon, Industrial IEEE802.15.4e networks: Performance and trade-offs, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 604–609. doi:10.1109/ICC.2015.7248388.

[17] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, K. S. J. Pister, A Realistic Energy Consumption Model for TSCH Networks, ResearchGate 14 (2) (2014) 482–489. doi:10.1109/JSEN.2013.2285411.

[18] R. Soua, P. Minet, Multichannel Assignment Protocols in Wireless Sensor Networks, Pervasive Mob. Comput. 16 (PA) (2015) 2–21. doi:10.1016/j.pmcj.2014.04.004.
URL http://dx.doi.org/10.1016/j.pmcj.2014.04.004

[19] T. Watteyne, S. Lanzisera, A. Mehta, K. S. J. Pister, Mitigating Multipath Fading through Channel Hopping in Wireless Sensor Networks, in: 2010 IEEE International Conference on Communications, 2010, pp. 1–5. doi:10.1109/ICC.2010.5502548.

[20] T. Watteyne, A. Mehta, K. Pister, Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense, in: Proceedings of the 6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN '09, ACM, New York, NY, USA, 2009, pp. 116–123. doi:10.1145/1641876.1641898.
URL http://doi.acm.org/10.1145/1641876.1641898

[21] G. Smart, N. Deligiannis, R. Surace, V. Loscri, G. Fortino, Y. Andreopoulos, Decentralized time-synchronized channel swapping for ad hoc wireless networks, IEEE Transactions on Vehicular Technology 65 (10) (2016) 8538–8553.

[22] X. Vilajosana, K. Pister, T. Watteyne, Minimal 6TiSCH Configuration, Internet-Draft draft-ietf-6tisch-minimal-19, Internet Engineering Task Force, work in Progress (Jan. 2017).
URL https://tools.ietf.org/html/draft-ietf-6tisch-minimal-19

[23] M. Palattella, N. Accettura, L. Grieco, G. Boggia, M. Dohler, T. Engel, On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH, IEEE Sensors Journal 13 (10) (2013) 3655–3666. doi:10.1109/JSEN.2013.2266417.

[24] N. Accettura, E. Vogli, M. Palattella, L. Grieco, G. Boggia, M. Dohler, Decentralized Traffic Aware Scheduling in 6tisch networks: design and experimental evaluation, IEEE Internet of Things Journal PP (99) (2015) 1–1. doi:10.1109/JIOT.2015.2476915.

[25] S. Duquennoy, B. Al Nahas, O. Landsiedel, T. Watteyne, Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys '15, ACM, New York, NY, USA, 2015, pp. 337–350. doi:10.1145/2809695.2809714.
URL http://doi.acm.org/10.1145/2809695.2809714

[26] Q. W. a. X. Vilajosana, 6top Protocol (6p) (Oct. 2016).
URL https://tools.ietf.org/html/draft-ietf-6tisch-6top-protocol-03

[27] R.-H. Hwang, C.-C. Wang, W.-B. Wang, A Distributed Scheduling Algorithm for IEEE 802.15.4e Wireless Sensor Networks, Computer Standards & Interfaces 52 (2017) 63–70. doi:10.1016/j.csi.2017.01.003. URL http://www.sciencedirect.com/science/article/pii/S0920548917300193

[28] G. Exarchakos, I. Oztelcan, D. Sarakiotis, A. Liotta, plexi: Adaptive re-scheduling web service of time synchronized low-power wireless networks, Journal of Network and Computer Applications.

[29] T. v. d. Lee, A. Liotta, G. Exarchakos, TSCH schedules assessment, in: 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), 2017, pp. 696–701. doi:10.1109/ICNSC.2017.8000175.

[30] G. Liu, L. Wong, H. N. Chua, Complex discovery from weighted ppi networks, Bioinformatics 25 (15) (2009) 1891–1897.

[31] T. v. d. Lee, G. Exarchakos, A. Liotta, Distributed TSCH scheduling: A comparative analysis, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 3517–3522. doi:10.1109/SMC.2017.8123176.

[32] Matplotlib colormap description, https://matplotlib.org/users/colormaps.html, accessed: 2017-12-25.

Interference Graphs to Monitor and Control Schedules in Low-Power WPAN

Tim van der Lee[a], Antonio Liotta[b], Georgios Exarchakos[a]

[a] *Eindhoven University of Technology, the Netherlands*
[b] *University of Derby, United Kingdom*

Abstract

With billions of connected devices in the near future, the major challenge is to develop networks to build an Industrial Internet of Things which is scalable, energy-efficient, reliable and affordable. To this end, low-power wireless personal area networks (LP-WPAN) provide a solution at minimum costs. However, to ensure continuous performance verification, LP-WPAN requires a centrally monitored and controlled service. This work proposes such an edge service, i.e. network monitoring and optimal reconfiguration of scheduled LP-WPANs. The approach is based on a transformation of the schedule into a new model, *interference graphs*. The interference graphs allow to design evaluation and rescheduling recommender methods to monitor and reconfigure the schedule. An experimental setup was developed to test and validate the approach. The results show that the model and methods provide an accurate representation of the behavior of the network, and that the new rescheduling recommender greatly improves the network's performance, compared to random rescheduling.

Keywords:
Internet of Things, Wireless sensor network, Interference graph, disconnection probability, Network management, Network scheduling

Email addresses: t.lee@tue.nl (Tim van der Lee), a.liotta@derby.ac.uk (Antonio Liotta), g.exarchakos@tue.nl (Georgios Exarchakos)

1. Introduction

Industry 4.0 is tightening the dependability requirements of low-power wireless personal area networks (LP-WPANs) to support reliable control loops. In parallel, Internet of Things (IoT) is promising very large networks of interconnected devices smaller, smarter and more energy efficient than ever. LP-WPANs offer unique solutions to the high installation and maintenance costs of copper or optical networks in industrial settings. With the target of optimizing and distributing production lines, big data coming from IoT could help on taking more informed decisions on controlling production. The more temporospatially fine-grained the collected information is, the more wireless communication is required resulting in more contended wireless medium, thus, increasing the unpredictability of control loops. For maximum predictability as well as continuous performance verification, smart factories need LP-WPAN status to be centrally monitored and controlled. This requirement could be fulfilled by a network-as-a-service (NaaS), a network resource allocation and optimization service for LP-WPANs.

While active research is on-going to define efficient scheduling algorithms for LP-WPANs in TSCH mode, very little attention is drawn on *a priori* evaluating the performance of these schedules and on the rescheduling problem. To the best of our knowledge, this work is the first to propose a rescheduling method to adapt to external constraints and possible topology changes. This work proposes an edge service to assess the performance of the LP-WPAN and trigger a targeted maximum-profit minimum-cost re-configuration. LP-WPANs, specified by IEEE802.15.4 [1], in Time Synchronized Channel Hopping (TSCH) mode allow for more deterministic resource allocation. Namely, transmitters, receivers, channels and timeslots used may be determined by a network control algorithm e.g. built for a LP-WPAN NaaS.

To reduce uncertainty introduced by the latency between end nodes and the LP-WPAN controller, the latter is assumed at the edge. As part of that controller, scheduling algorithms should avoid internal and external interference. Given the context of this work, network topology volatility, no scheduler in literature considers the complete topology and deploys minimum rescheduling of maximum performance gains, rescheduling problem. A performance manager and rescheduling driver, is required. This work proposes an edge-service to analyze and optimize scheduled networks, divided into three main contributions.

- A transformation of the scheduled network into interference graphs,

- A rating method to evaluate the performance of the scheduled network based on interference graph analysis,
- A recommender to indicate which connection is the most problematic also based on interference graph analysis.

While this work applies to the IEEE802.15.4-TSCH communication protocol, it can be easily extended to any scheduled network, *mutatis mutandis*. In order to evaluate the accuracy of both the assessment and rescheduling methods, experiments are made in a real-world LP-WPAN deployment.

In section 2, background and work related to the TSCH communication protocol is explained. The architecture of the different components presented in this work is detailed in the following section. Then, definitions are provided to describe a scheduled network and introduce the concept of interference graph. In section 5, our proposed approach presents an analysis of interference graphs to evaluate and pinpoint problematic connection in a scheduled network. A traffic aware scheduler is also presented. In section 6, experiments are run to confirm the validity of the rating method, and the rescheduling method is tested on the previously described traffic aware schedule. The paper concludes with a proposal for future work on this topic.

2. Background and related work

One of the main challenges in upscaled IoT is to provide reliable communication at the lowest possible energy cost. To this end, a wireless sensor networks (WSN) is preferred as it provides more flexibility and security than using the internet [2]. WSNs are already used in various application domain such as factory automation [3], distributed and process control [4, 5], smart grid [6] or in the healthcare domain [7, 8, 9, 10]. In the past few years, different technologies have attempted to address the reliability, energy efficiency, scalability and flexibility of WSNs. This includes new communication standard such as bluetooth [11], ZigBee [12], and the IEEE802.15.4 standard presented in the next paragraph. At the same time, new protocols are introduced by the Internet Engineering Task Force (IETF) such as 6LoWPAN[13], providing IPv6 for low-power wireless personal area networks.

2.1. The IEEE802.15.4-TSCH protocol

The IEEE802.15.4-2016 standard is designed for low-power, low-cost and low-rate network deployments. First published in 2006 [1], the standard

introduces the CSMA-CA algorithm, a communication protocol vulnerable to interference and multi-path fading [14, 15]. The unlimited delay possibilities of CSMA-CA makes it too unreliable for industrial deployment. The 2012 amendment introduces a new mode, Time Slotted Channel Hopping (TSCH), that has been proven to avoid interferences and more energy-efficient [16, 17].

TSCH enables time-slotted access to the communication layers of the device, and also allows to communicate with up to 16 different channels available at 2.4MHz. For each connection in the network, the time of the transmission has to be specified (*timeslot*), as well as the channel offset. The different devices of the scheduled network are kept synchronized through the emission of small packets, called enhanced beacons (EBs). The multichannel communication paradigm mitigates interference and congestion in the WSN [18]. The TSCH standard also introduces a channel hopping mechanism. When transmitting, the frequency f to be used for this connection is chosen as follows.

$$f = (ASN + channeloffset) \mod C$$

where ASN is the absolute slot number, that is the number of timeslots elapsed since the start of the network, and C is the total number of channels available. If communication occurs every x timeslots, x being prime with C, it is ensured that all available channels are used. Thanks to channel hopping, the scheduled network is less impacted by external perturbations [19, 20].

2.2. The scheduling problem

The schedule can be computed centrally, or in a distributed way [21]. TSCH minimal is the minimal TSCH schedule provided by the IETF [22]. It consists of a single shared cell used to both send and receive any type of packet. TASA, the *Traffic Aware Scheduling Algorithm* is a centralized scheduler based on edge coloring technique [23], reputated to be interference free, but limited when it comes to scalability. A decentralized version of TASA, DeTAS provides more scalability than TASA [24]. Orchestra is a decentralized algorithm implemented in contiki [25] providing high reliability in various test environments.

For all connections in a scheduled network, timeslot and channel offsets have to be specified. To avoid redundancy of information, they are specified for a certain amount of timeslots, composing a *slotframe*, sometimes referred to as *superframe*, which is repeated over time. During a timeslot, typically 10ms-15ms, a device can either transmit or listen over a certain channel offset,

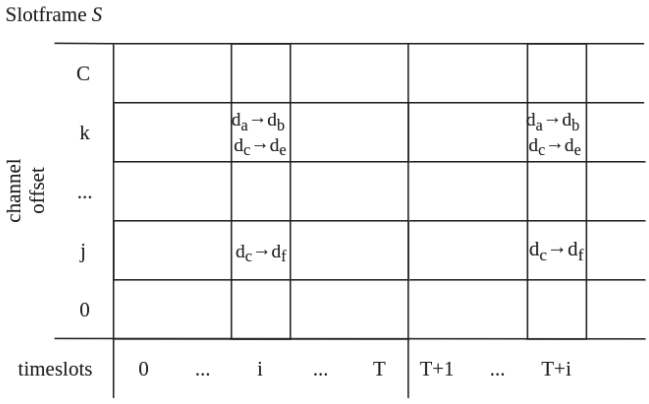


Figure 1: Example of slotframe containing a case of interference.

or remain idle. When transmitting an unicast packet, an acknowledgement is expected to be received. If absence of an acknowledgement, the device will attempt to retransmit at the next available possible timeslot. After the maximum number of retransmission is reached, the packet is dropped and considered as lost. In the following, a communication occurring at a certain timeslot and channel offset will be referred as connection.

The slotframe is potentially different for every devices. As an example, if at timeslot i and channel offset j device d_1 communicates with d_2, the cell (i, j) of the slotframe of d_1 will have to be a transmitting cell, and the cell (i, j) of device d_2 will have to be a listening cell. In this work, we assume an edge service to have full knowledge of the network. Therefore, we will consider only the slotframe of the scheduled network - cell (i, j) of the slotframe of the network will be $d_1 \rightarrow d_2$. The slotframe of the scheduled network is regarded as a matrix of T timeslots and C channel offsets. Filling this matrix with appropriate connections is called *scheduling* the TSCH network. The scheduling process is explained in [26]. Scheduling a TSCH network without interference is an NP-hard problem [27]. An example of a network slotframe is given by figure 1. In this example, a case of interference is present on cell (i, k). Two communications are occurring at the same time and channel, and assuming all devices are in interference range, they will interfere. There is also a case of interference at timeslot i, since d_c can not transmit to two different devices at the same time.

Using a different scheduler will affect the network in terms of latency, scal-

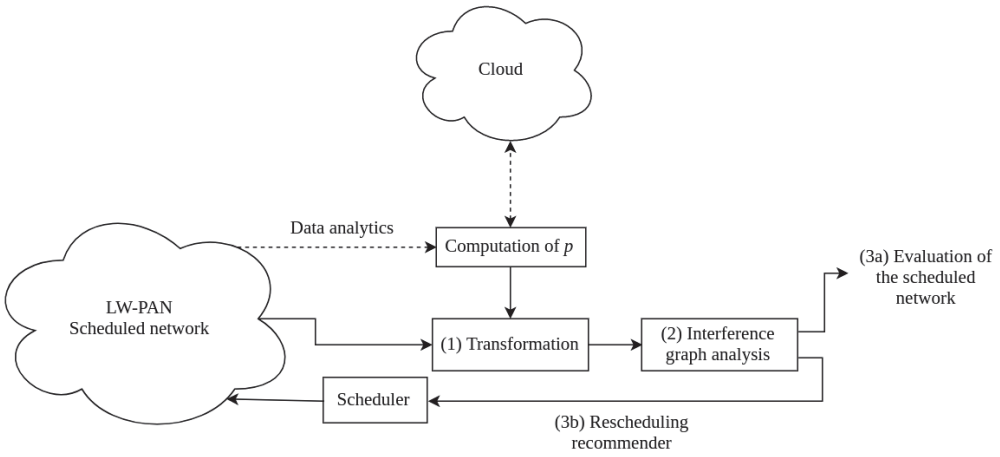


Figure 2: Schematic overview of the architecture of the proposed edge service.

ability, energy consumption and reliability. While most of these scheduling algorithms address these issues independently, there is not yet a scheduling algorithm able to handle all of them together, for any network. Most of the time, rescheduling is required to fine-tune the performance of the network according to the requirements of the use-case.

3. Architecture overview

Figure 2 summarizes the different components of the edge service proposed in this study, and how they interact. The scheduled LP-WPAN is first transformed into interference graphs (1), which rely on a key parameter p, the *disconnection probability*. p can be computed in different ways, e.g. machine learning. In this study, p will be evaluated through the analysis of experimental network-data. The second step is to analyze the interference graph (2), which leads to two outputs:

(3a): An evaluation of the scheduled network,

(3b): a recommender that indicates which connection to reschedule.

Evaluating a scheduled network can be used to predict and prevent failure in the scheduled network. It can also be used to compare two different schedules for the same network, and determining which one is best suited for the topology. The rescheduling recommender informs the scheduler of which

connection is the most problematic in the network. This information is used to directly change the schedule of the network and increases its reliability at runtime.

Deploying such an edge service in a real world deployment can be done in different ways. In our experimental setup, p is predicted based on the traffic pattern of the network, and does not require any interaction with the cloud, or the network. We perform step (1), (2) and the scheduling/re-scheduling process in the cloud. Schedules computed outside the LW-PAN network can be exchanged with the nodes with CoAP services such as plexi [28]. In our deployment, step (1) has to be performed outside the network because the devices used (JN5168) do not have enough RAM memory to analyze the entire network. However, depending on the capacity of the devices used, the different blocks presented in figure 2 can be ported to the device.

Using only standard graph analysis techniques, interference graph analysis is not computationally expensive. However, having one node per connection in the interference graph, step (1) and (2) may be expensive in terms of memory usage, if a lot of connections are scheduled. Therefore, we recommend to perform these tasks on non-constrained devices to address scalability issues.

4. Network model

This section provides the definitions and models needed to solve the rescheduling challenge of TSCH networks. First, a model of the TSCH scheduled network is given, followed by the definition of the interference graph, a representation of the TSCH scheduled network used in the presented approach. A table presenting the different acronyms and symbols used in this publication is given table 1.

4.1. TSCH scheduled network

A TSCH scheduled network is defined as a network of wireless devices using the TSCH communication protocol. We assume that the devices of this scheduled network are scheduled on 1 slotframe. If several sloframes are defined, it is always possible to find one slotframe, based on a combination of slotframes that will lead to the same behavior. Since slotframes are repeated over time, analyzing the behavior of the network during the entire duration of the slotframe is the same as analyzing the network at any time, assuming the environment does not change.

Table 1: Symbols and acronyms used

S, SF	Slotframe
C	Available channels for S
T	Available timeslots for S
$c_{i,j}, cell$	Cell of a slotframe
G	Connectivity graph model
F	Interference graph model
d	Device model in G
$e = (d_1, d_2)$	Connection between 2 devices in G
$l = (e_1, e_2)$	Link between 2 connections in F
$p(l)$	Disconnection probability (weight of l)
$\rho(X)$	Weighted density of a graph X
$deg(x)$	Node degree of a node x
PRR	Packet Reception Ratio

A TSCH scheduled network is composed of two main information elements, **topological** and **temporal**, and knowing both gives a full understanding of the scheduled network. The spatial information concerns the connectivity of the network, and informs us about all connections between the devices and the channel on which they occur. It also includes interfering devices, since the interference range may be greater than the direct connectivity range. Connections occur during a timeslot, hence the spatial information may change over time. The temporal information is there to keep track of these network changes in time.

Let us consider a TSCH scheduled network consisting of n devices communicating according to a slotframe S, with T timeslots and C channels. To represent the network we will use an approach similar to [23]. In this document, the TSCH network is represented as an directed graph $G = (V, E)$ where $V = \{d_0, d_1, ...d_{n-1}\}$ are the devices and $E \subset V \times V$ the connections of the network. As an example, a unicast connection from device d_a to d_b will be noted as $e = (d_a, d_b)$. Similarly, a broadcast connection initiated by device d_a will be a set of connections noted $\{(d_a, d_k), \forall k \in DN(d_a)\}$ where $DN(d)$ represents the set of devices that are in direct range of d. It is not always the case that direct range and interference range coincide.

This connectivity graph does not take into account the temporal information given by the slotframe. Therefore we introduce the slotted connectivity

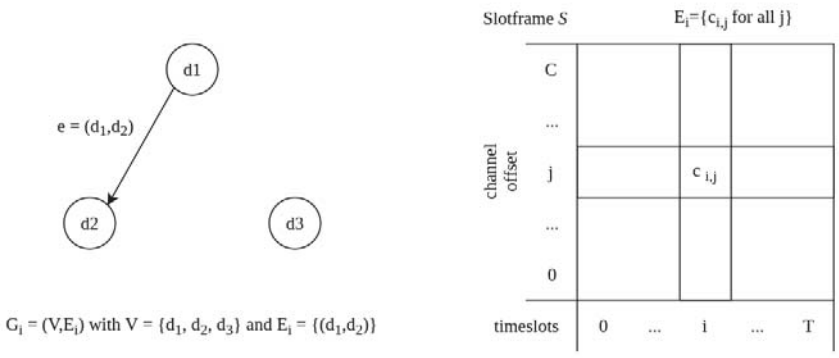


Figure 3: Construction method of the slotted connectivity graph.

graph G_i of a timeslot i. In the slotframe S of our scheduled network, each cell $c_{i,j}$ at timeslot i and channel offset j will indicate connections occurring at this timeslot. If no connections are occuring, c will be the empty set $\emptyset$. Thus, we can define the **slotted connectivity graph** of a scheduled network $G = (V, E)$ with a slotframe $S = \{c_{i,j}, \forall (i,j) \in T \times C\}$ at a timeslot i as

$$G_i = (V, E_i) \text{ with } E_i = \{e \in c_{i,j}, \forall j \in C\} \tag{1}$$

Figure 3 represents the construction method of the slotted connectivity graph. For a given slotframe S, all connections occuring at timeslot i are taken into account to build G_i. Following the definition of the slotted connectivity graph, the **complete connectivity graph** is defined as follows.

$$G^* = (V, E) \text{ with } E = \{e \in E_i, \forall i \in T\} \tag{2}$$

Similarly as G, G^* is a directed graph that does not take into account the temporal information given by the slotframe. However, G^* can be a multigraph, since the same connection may be present on multiple timeslots of the slotframe. Considering a set of devices $d \in V$ scheduled with the slotframe S, the information about the direct range of the devices is enough to construct G_i for all i and thus G^*. However, in order to study the impact of internal interference in a scheduled network and assess accurate deployments, we need to take into account devices in interference range. The following section describes a model based on G that takes into account interfering devices information.

4.2. Interference Graph

The interference graph is a representation of the scheduled network taking into consideration the possible interference and causes for packet loss, which can occur internally. First introduced in [29], its definition is refined in this work. Within a slotframe S, two connections will be considered as **conflicting** iff, on the same timeslot, a device has two different connections scheduled. The TSCH communication protocol will prioritize one of the connections, leading to a potential packet loss for the other one. Interference occurs when two different pairs of devices communicate on the same timeslot and channel. In this case, packets may be lost due to interference on the receiving side, assuming there is no clear channel assessment mechanism in place. In the remainder of this article, both conflicts and interference will be referred to as interference.

In the same way as the slotted and complete connectivity graphs were defined, we define a slotted, and a complete interference graph. Nodes of these graphs are connections occurring in the TSCH scheduled network, and these nodes are linked if the corresponding connections interfere. The slotted interference graph at timeslot i corresponding to a slotted connectivity graph G_i can be first defined as follows:

$$F_i = (E_i, L_i) \text{ with } L_i = \{(e_1, e_2) \in E_i^2 \text{ and } (e_1, e_2) \in L\}$$

F_i is a directed graph that illustrates the behavior of the network at timeslot i in terms of internal interference. However, not all interfering connections have the same impact on the performance of the network. As an example, if two unicast connections are interfering but have no traffic assigned, the impact on the performance of the scheduled network would be zero. Therefore, links of the interference graph are weighted with a weight factor $p : L \longrightarrow [0, 1]$ determining the impact of this interference on the global performance of the network, called *disconnection probability*. For simplicity, we do not consider interfering connections that have no impact on the behavior of the network. If $p(l) = 0$, the link l is removed from the interference graph. Note that for (e_1, e_2) and (e_2, e_1) in L we may have $p((e_1, e_2)) \neq p((e_2, e_1))$. The complete definition of the **slotted interference graph** at timeslot i has to include p.

$$F_i = (E_i, L_i, p) \tag{3}$$

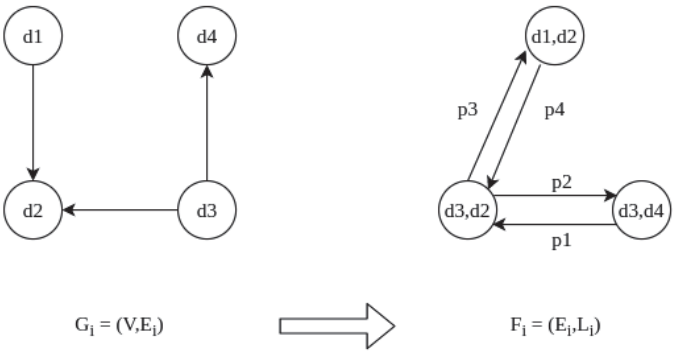


Figure 4: Construction method of the interference graph assuming all channels are different.

Figure 4 represents an example of transformation from the slotted connectivity graph G_i to the slotted interference graph F_i. Assuming all channels are different, the slotted connectivity graph G_i presents a conflict since device $d3$ cannot handle two transmitting operations at the same time. Thus, connection $d3, d2$ and $d3, d4$ are linked with different disconnection probabilities p. Similarly, device $d2$ can not listen to two connections at the same time, therefore linking the nodes $(d1, d2)$ and $(d3, d2)$.

The **complete interference graph** is defined in the same way as the complete connectivity graph. All connections are represented with a node in the graph, and all interfering connections are linked with a directed link of weight p.

$$F^* = (E, L, p) \tag{4}$$

where $L = \bigcup_{i \in T} L_i$. As for the complete connectivity graph, the complete interference graph may be a multigraph, since the same interference can occur several times in the slotframe.

5. Proposed approach

In this section, we first present a method to analyze the previously defined interference graph. This provides a rating method allowing us to evaluate the performance of a scheduled network. Then, this rating method is used to define a rescheduling technique applied on a traffic aware schedule.

5.1. Rating method

Our rating method aims to evaluate schedules in given topologies based on internal interference. Fortunately, the weighted links of the interference graphs precisely represent interference we want to detect. The weight p of these links also indicates the impact of the interference on the performance of the scheduled network. Consequently, we will rate schedules according to the weighted density of the interference graph. The density of a graph is the result of the number of links divided by the number of possible links. For a weighted graph such as $F^* = (E, L, p)$, we will take the density according to its definition in [30].

We define the **density** ρ of the complete interference graph or the slotted interference graph at timeslot i as follows:

$$\forall i \in T,\ \rho(F_i) = \frac{\sum_{l \in L_i} p(l)}{|E_i|(|E_i| - 1)} \tag{5}$$

$$\rho(F^*) = \frac{\sum_{l \in L} p(l)}{|E|(|E| - 1)} \tag{6}$$

ρ will be considered as our rating metric for schedules. An interference-free schedule will present no links in its interference graphs, and thus will obtain a density equal to 0. Since F^* may be a multigraph, its density is unbounded. The density, and thus the rating method, depends on the parameter p that can be described as the probability that the interference impacts the proper behavior of the scheduled network. Defining the function p as accurately as possible is required to rate different schedulers between each other and in different topologies. Furthermore, performing an analysis of the interference graph knowing beforehand the value of p, will allow to predict the performance of the schedule in a given topology. In this work, we focus on the evaluation of one scheduler in one topology. Our first approach to define p is explained in Section 6.1. A more precise definition of p will be provided in a subsequent paper.

5.2. Traffic-aware scheduler

In this section, we introduce a centralized scheduling algorithm. This generates a traffic-aware slotframe for the network, which is not interference free. A rescheduling method to make it interference free is presented in the next section. While the rescheduling method applies on any scheduler, we will test it on this particular traffic-aware schedule.

Let's define the number of frames to be sent as the traffic. The first step is to determine how many cells should be assigned to a connection to comply with the traffic load of the network. To establish a traffic-aware schedule, the amount of traffic generated by each node and the routing topology must be known. We note $q_{a,b}$ the traffic generated per slotframe by d_a for d_b and $route(d_a, d_b)$ the route from d_a to d_b.

Algorithm 1: Function for traffic aware cell attribution

```
Function get_cells(d_a,d_b)
    cells ← q_{a,b}
    for d_i,d_j in all devices do
        if (d_i, d_j) in route(d_a, d_b) then
            cells ← cells + q_{i,j}
        end
    end
    return cells
```

The function get_cells presented in Algorithm 1 computes the minimum number of cells required for the connection (d_a, d_b). One cell is theoretically enough to transport one frame per slotframe, and keep the queues empty. However, in a real deployment with external perturbations and interference, the number of cells must be increased. This function determines how many cells are required to reach a traffic-aware schedule, but the scheduler allocates these cells randomly. Therefore, the resulting slotframe is potentially subject to internal interference. This scheduler accepts request from external services to reschedule certain connections to empty cells.

5.3. On-the-fly rescheduling method

In this section, we present a rescheduling method that selects a problematic connection. The rescheduling method is based on the analysis of the slotted interference graph. In a scheduled network with internal interference, we will note F_i for $i \in T$ the slotted interference graphs, and the density of all slotted interference graphs $\rho(F_i)$ is calculated by equation 5. $\max_i(\rho(F_i))$ defines the most problematic timeslot in terms of internal interference, and must receive priority.

In the following, we note $F_m = (E_m, L_m, p)$ the slotted interference graph with the highest density. In this interference graph, the conflicting connections are linked with a weight p. In order to find the most problematic connection, we compute the weighted outdegree of each node in the graph

$deg^+(e)$, i.e. the number of edges going out of this node with their respective weight p. The connection with the highest degree is chosen to be rescheduled first, and is rescheduled to an unused cell for this device. If there are several candidates, the connection is chosen randomly. To summarize, the rescheduling process is split in three main parts:

- Find the most problematic m with $max_i(\rho(F_i))$,
- Find the most problematic connection with $deg^+(E_m)$,
- Indicate to the scheduler that this connection needs to be rescheduled.

The algorithm will converge to an interference-free schedule for the topology, if it exists. Because cells are re-allocated randomly, this process can become computational-intensive. For example, if a connection can only be scheduled on 1 specific cell, the probability to pick this cell is $\frac{1}{T \times C}$ where T is the number of timeslots and C the number of channels in the slotframe. Here we focus on rescheduling the most problematic connections first. Preliminary results for a more refined method, based on choosing timeslots, show that for a distributed scheduler, choosing the timeslot is more efficient with messaging the neighboring devices [31].

The proposed method uses the interference graphs F_i, hence depends on p. In some cases where p is constant and the network topology is fixed, the rescheduling process is almost immediate. However, if the topology is changing or if p takes a long time to compute, i.e. p based on packet reception ratio or p estimated with machine learning, the rescheduling process can become computational intensive.

6. Real-world experiments

To evaluate the rating method and the rescheduling method applied to the traffic-aware schedule, several experiments have been designed using a real environment. In a following section, the experimental setup is described. Then the rating method is analyzed and finally the performance of the rescheduling method compared with the random rescheduling method.

6.1. Experimental setup

In order to run the experiment, p has to be defined first. The goal is to evaluate and reschedule a slotframe in a given topology to reach an

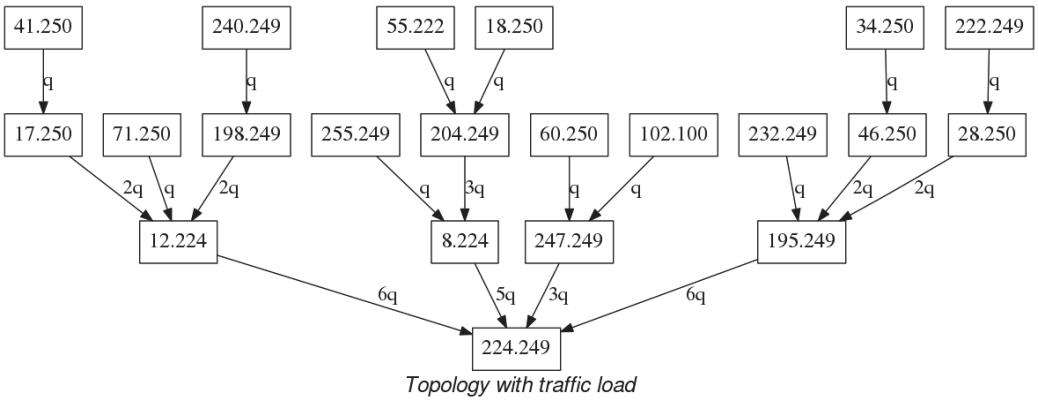


Figure 5: Topology of the devices with traffic load.

interference- and conflict-free slotframe. Taking $p = 1$ assumes that all interfering connections are equally affecting the good behavior of the network. However, depending on the traffic of these connections, the impact may be completely different. Therefore, we will take p based on the amount of packets in queue for both interfering connections. If we consider two interfering connections e_a, e_b in L, and the number of packets to be sent for a connection e as $q(e)$, p is defined as follows:

$$p((e_a, e_b)) = \frac{q(e_a) + q(e_b)}{\max_{(e_1,e_2)\in L}(q(e_1) + q(e_2))} \tag{7}$$

Then, it satisfies the requirement $0 < p \leq 1$. In this configuration, we also have $p((e_a, e_b)) = p((e_b, e_a))$. The experimental testbed is composed of 21 Jennic JN5168 devices in a tree topology. Devices use the latest version of Contiki with TSCH enabled. Figure 5 represents the topology on which the devices are tested. In today's IoT deployments, most low-power personal area networks (LW PAN) applications rely on a PAN coordinator to interface with devices outside the network. In our configuration, device 224.246 is the PAN coordinator, and all devices are transmitting packets towards it. All devices generate q packet per slotframes and forward all incoming traffic to their parent 224.246. Also, all devices are deployed in the same room and thus, are in interference range of each other.

The function presented in Algorithm 1 returns the theoretical minimum number of cells to be attributed per link. As we are in a real deployment, we allocate three times more cells than the number indicated. To simplify,

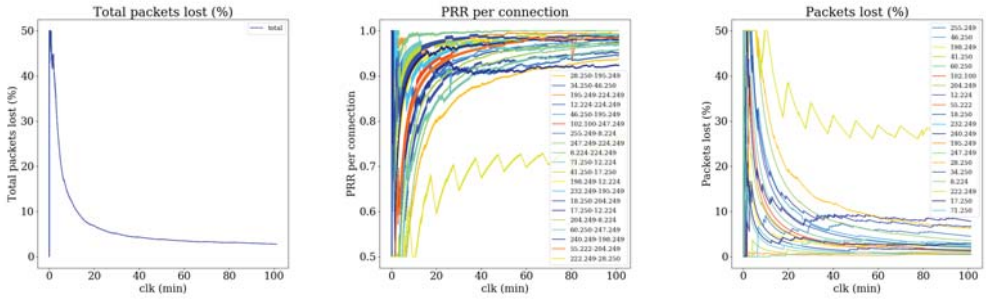
we divide the slotframe size by 3. Thus, $q = 0.3$ packets per slotframe. With timeslots of 15ms, the final slotframe size is 200 timeslots, and each node generates a packet every 9sec. In this way, the traffic load of each connection (see figure 5) corresponds to the number of cells to be attributed. All connections related to data are made on dedicated cells. The TSCH network is synchronized through EBs, additional messages that may occupy the queues and create interference. In order to focus the study on the data traffic only, EBs have a separate slotframe and channel. This way, results do not include data generated by EBs. The slotframe reserved for data is using the 15 other channels provided by the 802.15.4 communication protocol.

In the following sections, experiments are run to confirm the validity of both rating and rescheduling methods. In these experiments, the same topology and the same connections will be used but with different schedules. The gathered data does not vary after 1.5 hours, therefore experiments last between 1.7 to 2 hours. In all experiments, the number of retransmissions is set to 0, meaning that each packet has only one chance to be transmitted. If the transmission fails due to interference or external perturbations, the packet is dropped. This allows us to directly see the impact of perturbations in the network.

6.2. Rating method verification

Analyzing the density of the complete interference graph provides feedback on the amount of interference present in the network. For this feedback to be visualized, we generated four slotframes with different numbers of interfering connections.

The first slotframe, $SF1$, contains no interference: it is an interference-free slotframe. For this topology, a random generated slotframe has a 16% chance of being interference-free. Experiments are run to analyze the behavior of the network. Metrics such as the packet reception ratio per link and the number of packet lost are displayed on figure 6. Since our experimental setup does not allow retransmissions, the packet reception ratio (PRR) per connection is not equal to 1, but tends to stabilize between 90% and 99% for all connections except one. It appears from figure 6c and 6b that device 198.249 exits and re-joins the network frequently. The network is subject to external interference, or perturbations in the hardware itself that causes transmission to fail. This explains why the perfect slotframe does not obtain a 100% PRR. However, the $SF1$ is very effective as its end-to-end PRR



(a) Packets lost for $SF1$ as a percentage of the total number of packet emitted by the network.

(b) Packet reception ratio per connection for $SF1$.

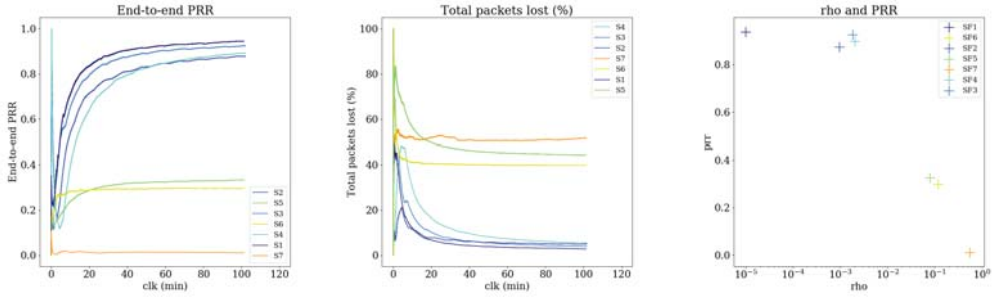
(c) Packets lost per connection for $SF1$ as a percentage of the number of packet sent through this connection.

Figure 6: Results of the performance analysis of $SF1$.

reaches 93.7%. Since the slotframe is interference-free, the densities of its interference graphs are 0.

The next tested slotframes are $SF2, SF3$ and $SF4$. $SF2$ presents one conflict for the device 8.224. $SF3$ and $SF4$ present both 2 problems: $SF3$ shows a conflict and a case of interference, and $SF4$ presents two conflicts. The probability of this occurring is approximatively 27%. When generating random schedule, this is the second most common situation. Generating a slotframe with 1 interference occurs with a probability of approx. 31%. For $SF2$ we obtain $\rho(F^*) = 0.00097$ and $PRR = 0.87$. We have for $SF3$ $\rho(F^*) = 0.00184$ and for $SF4$ $\rho(F^*) = 0.00203$. The difference between the impact of the interference detected by ρ is correctly observed when looking at the end-to-end PRR. For $SF3$, $PRR = 92.5\%$ and for $SF4$, $PRR = 89.8\%$. However, both $SF3$ and $SF4$ obtain a higher PRR than $SF1$ while having a higher density. This imprecision is due to the fact that p does not take into account if the conflicting connections are two receiving cells, two transmitting cells or a mix of both. Due to the implementation of TSCH, the case of two conflicting receiving cells present in $SF2$ is often subject to more packet loss than the other cases.

The measure of ρ seems to be correlated with the end-to-end PRR, three more slotframes were tested to confirm this hypothesis. $SF5$, $SF6$ and $SF7$ are unrealistic slotframes generated for the purpose of the experiment, as they contain 136, 190 and 861 cases of interference respectively. For $SF5$, we obtain $\rho(F^*) = 0.0789$ and an end-to-end PRR of 32.5%. For $SF6$,



(a) End-to-end packet reception ratio for all tested slotframes.

(b) Total number of packets lost for all tested slotframes.

(c) Relation between ρ and PRR for all tested slotframes.

Figure 7: Performance of all slotframes in relation with ρ, taken to 10^{-5} precision.

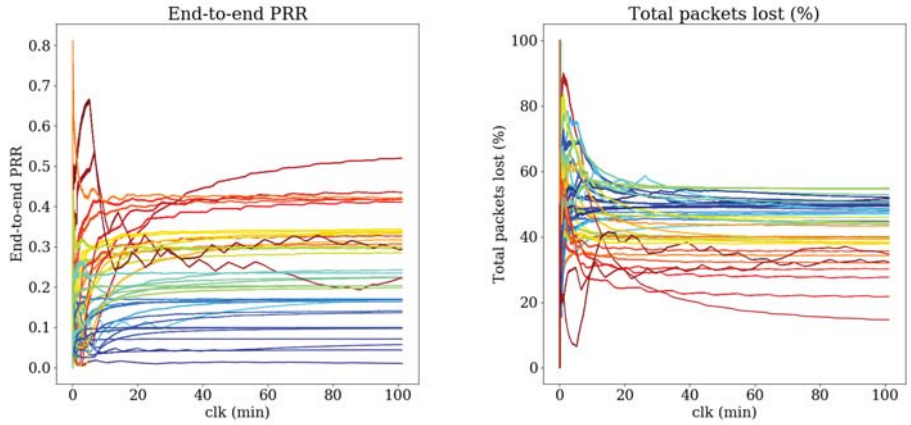
$\rho(F^*) = 0.1180$ and $PRR = 29.8\%$, and for $SF7$, $\rho(F^*) = 0.5528$ and $PRR = 1.02\%$.

Figure 7 indicates for all experiments, the end-to-end PRR, total packets lost and the relation between ρ and the PRR. Between all experiments, a correlation between $\rho(F^*)$ and the amount of packets lost seems to appear. Similarly, a high value of ρ corresponds to a low end to end PRR and thus, a poor global network performance. This relation is visible in figure 7c.

6.3. Rescheduling method verification

In order to visualize the performance of rescheduling, we will consider a scheduler, initializing the TSCH network with a traffic-aware schedule as presented in Section 5.2. The traffic-aware schedule positions cells in the worst possible way, with connections scheduled on timelot 0 and channel 0. However, this scheduler accepts recommendations from an external service to reschedule specific connections. As all connections interfere, all connections need to be rescheduled. We run two sets of experiments. In the first set, the connection to reschedule is chosen randomly among all interfering connections. In the second set, the connection to reschedule is chosen with interference graph analysis, as explained in Section 5.3. In the following, we refer to the first set as *random rescheduling* or rescheduling *without* interference graph analysis. The second set is referred to rescheduling *with* interference graph analysis. In both sets, the connection is rescheduled randomly to an available cell.

Figure 8 represents the performance of random rescheduling in the network, each figure representing 40 iterations. The colors are distributed ac-



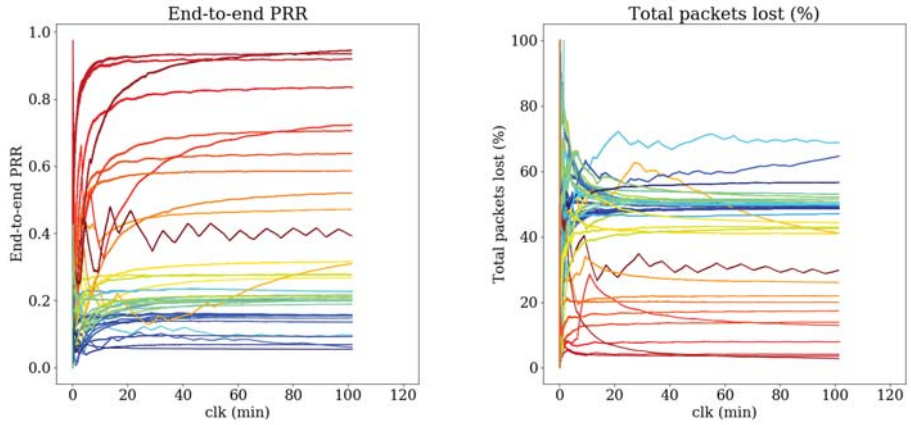
(a) End-to-end packet reception ratio of each iteration.

(b) Number of packets lost for all iterations in percentage of the total number of packet sent.

Figure 8: Results of the first 40 iterations of the random rescheduling method.

cording to the matplotlib *jet* colormap [32]. Initial iterations are represented in dark blue, and latest iterations are in dark red. In the first iterations of random rescheduling, the packet reception ratio is close to 0, and more than half of all packets are lost. Most connections are indeed interfering since all connections are initially scheduled on timeslot 0 and channel 0. After 40 iterations, the number of packets lost drops to 17%. However, even if almost all connections are rescheduled, the packet reception ratio does not exceed 60%. Connections that are not rescheduled are critical connections in the network ($8.224 - 224.249$ and $247.249 - 224.249$), and therefore are responsible for important packet loss. While most of the experiments produce a stable end-to-end PRR and packet loss, some measures appear to be irregular. This is due to the fact that real-life experimentation are subject to unpredictable hardware failure or external perturbations.

Results of rescheduling using the interference graph analysis are represented by figure 9. Colors are also distributed similarly as the previous set of experiment. The first experiment is represented in blue, and presents significant interference: more than 50% of all packets are lost, and the end-to-end PRR does not exceed 10-15%. However, after several iterations, the PRR and amount of packets lost are significantly increased. The rescheduling method focuses on first rescheduling the most critical connections according to the interference graph. In this set, the most problematic timeslot is



(a) End-to-end packet reception ratio of the network for each iteration.

(b) Percentage of packets lost of the global network for each iteration.

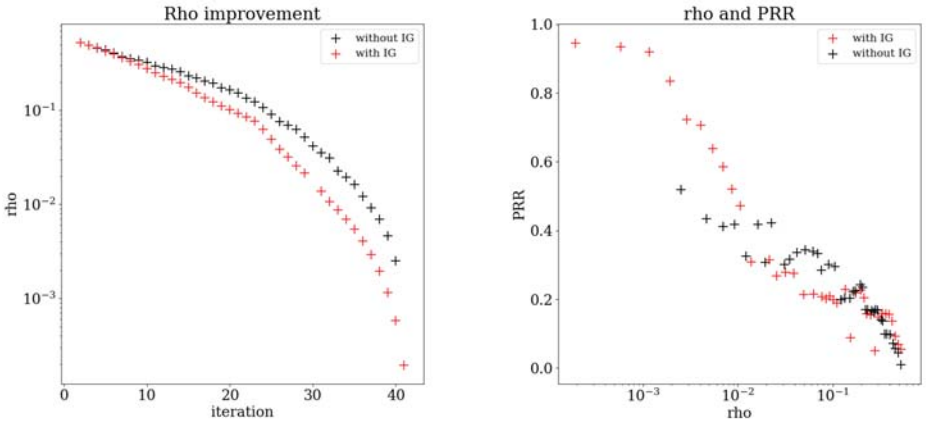
Figure 9: Results of the first 40 iterations of the rescheduling method based on interference graph analysis.

always timeslot 0 since all connections are scheduled on this timeslot initially. The interference graph analysis determines which connection has to be rescheduled based on the weighted out degree evaluation. With p based on the queues, connections close to PAN coordinator are chosen first, leading to a significant PRR and packet loss improvement after most of them are rescheduled.

Both methods are compared and represented in figure 10. Figure 10a represents the difference of improvement of the two methods for every iteration. We can see that rescheduling using the interference graph analysis improves the evaluation of the global network (density of the complete interference graph) faster than random rescheduling. Figure 10b represents the evolution of the PRR with the evaluation $\rho(F^*)$ of the schedule. Initially, both methods present a similar improvement. However, after 30 iterations, or when $\rho(F^*) < 10^{-2}$, the rescheduling method with interference graph analysis presents a better performance than the random rescheduling method.

6.4. Discussion

The first set of experiments compares random-generated slotframes. The density ρ applied on the complete interference graph F^* provides different values for these slotframes, as they present a different amount of internal



(a) Improvement of the rating metric ρ over 40 iterations.

(b) Packet reception ratio as function of the measure ρ for both experiment sets.

Figure 10: Side-by-side comparison the random rescheduling method (black) and the rescheduling method with interference graph analysis (red).

conflicts and interference. From the experiments, the packet reception ratio and the amount of packets lost appears related to the density calculated. In the second set of experiment, a slotframe fully conflicting is rescheduled. The density and the slotted interference graphs are used in order to determine which connections have to be rescheduled first. Compared with random rescheduling, the use of interference graphs improves significantly the rescheduling process.

In both sets of experiments, the performance of the network is measured with the PRR. In this regard, the density of the interference graph model provides a good prediction of the performance of the global network. However, in both sets of experiments, unpredicted failures occurred, resulting in an oscillating PRR that can be seen in some iterations of figure 8a or 9a. This creates small differences between the predicted performance and the actual performance of the global network. This difference is due to the fact that the chosen disconnection probability p is calculated based on the estimated traffic of each connection. It does not predict hardware failures, nor external perturbations, which can greatly impact the network since there are no retransmissions allowed in our setup.

7. Conclusion and future work

With an industrial IoT landscape growing exponentially, network volatility in scheduled networks and reliability in terms of interference need to be addressed. This work proposes an innovative edge service to address this challenge based on interference graph analyses, a method to pinpoint which connection requires to be rescheduled. Evaluating the weighted density of the complete interference graph allows to rate the scheduled network in terms of internal interference. The approach is validated with experimental results, showing that the density of the interference graph is accurately representing the amount of internal interference in the scheduled network. The accuracy of the rating method can be further improved with the *disconnection probability*, p. The rescheduling recommender experiments demonstrate that the new rescheduling method is superior to random rescheduling. It is also an improvement for the global performance of the network.

Potential for further improvement can be proposed. In the experiments, p is based on the queue size of both interfering connections and does not take into account which type of interference is occurring, or if cells are shared or dedicated. A better definition of p will therefore improve both evaluation and rescheduling methods. Since this service runs on the edge of the network, there are no resource constraints for the computation of p. Knowing preemptively p for a specific environment, and thus ρ, would allow to predict the performance of a schedule in this environment.

Different environments will have different impact on the performance of the scheduled network. In this work, experiments were run in the same room, with the same devices and with the same network topology. More experiments have to be conducted to refine the definition of p, and validate the evaluation method in any topology.

The edge service proposed in this study improves the performance and reduces the energy footprint of the scheduled network. Interference is responsible for retransmissions, an avoidable energy cost.

Acknowledgment

This research has been funded by the European Unions Horizon 2020 project INTER-IOT (grant number 687283), and was carried out at the Eindhoven University of Technology in the Netherlands.

[1] IEEE Standard for Low-Rate Wireless Networks, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) (2016) 1–709doi:10.1109/IEEESTD.2016.7460875.

[2] C. Alcaraz, P. Najera, J. Lopez, R. Roman, Wireless sensor networks and the internet of things: Do we need a complete integration?, in: 1st International Workshop on the Security of the Internet of Things (SecIoT10), 2010.

[3] A. Willig, Recent and emerging topics in wireless industrial communications: A selection, IEEE Transactions on industrial informatics 4 (2) (2008) 102–124.

[4] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, S. S. Sastry, Distributed control applications within sensor networks, Proceedings of the IEEE 91 (8) (2003) 1235–1246.

[5] V. C. Gungor, G. P. Hancke, Industrial wireless sensor networks: Challenges, design principles, and technical approaches, IEEE Transactions on industrial electronics 56 (10) (2009) 4258–4265.

[6] V. C. Gungor, B. Lu, G. P. Hancke, Opportunities and challenges of wireless sensor networks in smart grid, IEEE transactions on industrial electronics 57 (10) (2010) 3557–3564.

[7] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi, R. Jafari, G. Fortino, From modeling to implementation of virtual sensors in body sensor networks, IEEE Sensors Journal 12 (3) (2012) 583–593.

[8] A. Milenkovi, C. Otto, E. Jovanov, Wireless Sensor Networks for Personal Health Monitoring: Issues and an Implementation, Comput. Commun. 29 (13-14) (2006) 2521–2533. doi:10.1016/j.comcom.2006.02.011. URL `http://dx.doi.org/10.1016/j.comcom.2006.02.011`

[9] G. Fortino, M. Pathan, G. Di Fatta, Bodycloud: Integration of cloud computing and body sensor networks, in: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, IEEE, 2012, pp. 851–856.

[10] R. Gravina, P. Alinia, H. Ghasemzadeh, G. Fortino, Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges, Information Fusion 35 (2017) 68–80.

[11] S. Bluetooth, Bluetooth core specification version 4.0, Specification of the Bluetooth System.

[12] Z. Alliance, Ieee 802.15. 4, zigbee standard (2009).

[13] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, Rpl: Ipv6 routing protocol for low-power and lossy networks, RFC 6550, RFC Editor, http://www.rfc-editor.org/rfc/rfc6550.txt (March 2012).
URL http://www.rfc-editor.org/rfc/rfc6550.txt

[14] J. T. Adams, An introduction to IEEE STD 802.15.4, in: 2006 IEEE Aerospace Conference, 2006, pp. 8 pp.–. doi:10.1109/AERO.2006.1655947.

[15] D. D. Guglielmo, G. Anastasi, A. Seghetti, From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things, in: S. Gaglio, G. L. Re (Eds.), Advances onto the Internet of Things, no. 260 in Advances in Intelligent Systems and Computing, Springer International Publishing, 2014, pp. 135–152, dOI: 10.1007/978-3-319-03992-3_10.
URL http://link.springer.com/chapter/10.1007/978-3-319-03992-3_10

[16] T. Watteyne, J. Weiss, L. Doherty, J. Simon, Industrial IEEE802.15.4e networks: Performance and trade-offs, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 604–609. doi:10.1109/ICC.2015.7248388.

[17] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, K. S. J. Pister, A Realistic Energy Consumption Model for TSCH Networks, ResearchGate 14 (2) (2014) 482–489. doi:10.1109/JSEN.2013.2285411.

[18] R. Soua, P. Minet, Multichannel Assignment Protocols in Wireless Sensor Networks, Pervasive Mob. Comput. 16 (PA) (2015) 2–21. doi:10.1016/j.pmcj.2014.04.004.
URL http://dx.doi.org/10.1016/j.pmcj.2014.04.004

[19] T. Watteyne, S. Lanzisera, A. Mehta, K. S. J. Pister, Mitigating Multipath Fading through Channel Hopping in Wireless Sensor Networks, in: 2010 IEEE International Conference on Communications, 2010, pp. 1–5. doi:10.1109/ICC.2010.5502548.

[20] T. Watteyne, A. Mehta, K. Pister, Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense, in: Proceedings of the 6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN '09, ACM, New York, NY, USA, 2009, pp. 116–123. doi:10.1145/1641876.1641898.
URL http://doi.acm.org/10.1145/1641876.1641898

[21] G. Smart, N. Deligiannis, R. Surace, V. Loscri, G. Fortino, Y. Andreopoulos, Decentralized time-synchronized channel swapping for ad hoc wireless networks, IEEE Transactions on Vehicular Technology 65 (10) (2016) 8538–8553.

[22] X. Vilajosana, K. Pister, T. Watteyne, Minimal 6TiSCH Configuration, Internet-Draft draft-ietf-6tisch-minimal-19, Internet Engineering Task Force, work in Progress (Jan. 2017).
URL https://tools.ietf.org/html/draft-ietf-6tisch-minimal-19

[23] M. Palattella, N. Accettura, L. Grieco, G. Boggia, M. Dohler, T. Engel, On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH, IEEE Sensors Journal 13 (10) (2013) 3655–3666. doi:10.1109/JSEN.2013.2266417.

[24] N. Accettura, E. Vogli, M. Palattella, L. Grieco, G. Boggia, M. Dohler, Decentralized Traffic Aware Scheduling in 6tisch networks: design and experimental evaluation, IEEE Internet of Things Journal PP (99) (2015) 1–1. doi:10.1109/JIOT.2015.2476915.

[25] S. Duquennoy, B. Al Nahas, O. Landsiedel, T. Watteyne, Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys '15, ACM, New York, NY, USA, 2015, pp. 337–350. doi:10.1145/2809695.2809714.
URL http://doi.acm.org/10.1145/2809695.2809714

[26] Q. W. a. X. Vilajosana, 6top Protocol (6p) (Oct. 2016).
URL https://tools.ietf.org/html/draft-ietf-6tisch-6top-protocol-03

[27] R.-H. Hwang, C.-C. Wang, W.-B. Wang, A Distributed Scheduling Algorithm for IEEE 802.15.4e Wireless Sensor Networks, Computer Standards & Interfaces 52 (2017) 63–70. doi:10.1016/j.csi.2017.01.003. URL http://www.sciencedirect.com/science/article/pii/S0920548917300193

[28] G. Exarchakos, I. Oztelcan, D. Sarakiotis, A. Liotta, plexi: Adaptive re-scheduling web service of time synchronized low-power wireless networks, Journal of Network and Computer Applications.

[29] T. v. d. Lee, A. Liotta, G. Exarchakos, TSCH schedules assessment, in: 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), 2017, pp. 696–701. doi:10.1109/ICNSC.2017.8000175.

[30] G. Liu, L. Wong, H. N. Chua, Complex discovery from weighted ppi networks, Bioinformatics 25 (15) (2009) 1891–1897.

[31] T. v. d. Lee, G. Exarchakos, A. Liotta, Distributed TSCH scheduling: A comparative analysis, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 3517–3522. doi:10.1109/SMC.2017.8123176.

[32] Matplotlib colormap description, https://matplotlib.org/users/colormaps.html, accessed: 2017-12-25.