A distributed density optimized scheduling function for IEEE 802.15.4e TSCH networks

TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2017; **00**:1–15
DOI: 10.1002/ett

RESEARCH ARTICLE

A Distributed Density Optimized Scheduling Function for IEEE 802.15.4e TSCH Networks

Esteban Municio*, Kathleen Spaey, Steven Latré

University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science

ABSTRACT

The Industrial Internet of Things (IIoT) is a challenge for Wireless Sensor Networks (WSN), where ultra-reliability, guaranteed performance and ultra-low-power consumption are mandatory requirements to enable critical IoT applications. The Time-Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4e standard is one of the most promising technologies to accomplish these requirements by yielding guaranteed performance and, simultaneously, efficiently combating external interference and multi-path fading. One of the challenges in TSCH networks is to build an efficient schedule for managing the access of the nodes to the timeslots and channels. Several scheduling algorithms have been proposed. Currently, the Scheduling Function Zero *SFx* is one of the proposed scheduling algorithms for 6TiSCH, the ongoing standard for an IPv6-enabled stack working over TSCH. *SFx* is based on random resource allocation according to the traffic demand, which makes it inadequate for large-scale and dense deployments due to internal collisions. This paper has two main goals. First, we extensively investigate the performance of *SFx* for large-scale and dense scenarios, analyse its scalability and identify its scheduling limitations. Second, we present a new TSCH-based scheduling function, the Distributed Broadcast-based Scheduling algorithm (DeBraS), a scheduling solution designed for dense deployments based on sharing scheduling information between nodes in order to reduce collisions pro-actively. We show, through extensive and large-scale simulations, that DeBraS supports much larger densities than state-of-the-art scheduling functions, outperforming the current distributed 6TiSCH algorithm *SFx* up to 1.61 times in terms of throughput for large network sizes, at the expense of an increase in power consumption.

***Correspondence**
Esteban Municio, University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science
Email: esteban.municio@uantwerpen.be

1. INTRODUCTION

Nowadays we are witnessing the potential of the Internet of Things (IoT) paradigm. A large number of IoT devices (low complexity and resource-constrained) are already running numberless applications in various areas such as security, energy management and entertainment. However in industrial applications such as critical process monitoring and automation, IoT applications are still in their infancy. This is mainly because their requirements of ultra reliability, guaranteed performance and ultra low-power consumption sometimes seem to be antagonistic.

Industrial IoT applications are designed to run over Low-power Lossy Networks (LLNs) where the wireless medium is highly unreliable. Additionally, since nodes are highly resource constrained embedded devices, the Time-Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4e seem to be a promising solution for delivering these high-demanding IIoT requirements. The channel hopping technique addresses unreliability issues caused by factors such as multipath fading and narrow-band external interference. The Time Division Multiple Access (TDMA) nature of TSCH provides the guaranteed performance. Finally, the PHY layer plus a schedule based on a combination of timeslots and channels allow an energy-efficient radio management that results in ultra-low power consumption at the resource constrained devices [1, 2].

The TSCH mode is only defined as a MAC layer and therefore needs upper layers to address the issues related with scheduling, network addressing, routing and end-to-end communication. In order to build a whole IoT solution, *IPv6 over the TSCH mode of IEEE 802.15.4e* (6TiSCH) [3] is being defined in the 6TiSCH IETF Working Group as an attempt to create a standard based on the IEEE 802.15.4e link-layer TSCH mode together with an IPv6-enabled IoT upper stack. The performance of 6TiSCH is starting to accomplish the IIoT requirements (Figure 1) [4], however as we will show, the scalability of current 6TiSCH

Prepared using ettauth.cls [Version: 2012/06/19 v2.10]

implementations is low and their performance drops drastically for large scale and dense network scenarios. This is mainly due to the increase of the number of packet collisions caused by internal interference when the network size increases. Internal interference occurs because of the non-optimality of the scheduling algorithm and is usually present in any distributed wireless network.

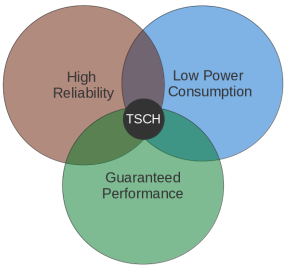


Figure 1. The three IIoT requirements

The 6TiSCH WG is currently investigating different scheduling algorithms. However, to the best of our knowledge the *DeBraS* algorithm is the best approach for coping with packet collisions in a proactive and distributed manner [5] in large-scale and dense environments. *DeBraS* fosters higher throughput per node and lower delays than current scheduling algorithms by allowing nodes to share scheduling information. However, the cost for these improvements is a higher energy consumption. This paper is a continuation of our previous work and the contribution is two folded. First, to offer a formal and profound analysis of the 6TiSCH performance when the network size increases and second, to evaluate in depth the performance of the proposed *DeBraS* algorithm, assessing its role as enabler for improving the scalability of large scale and dense 6TiSCH networks.

The remainder of this paper is organized as follows. Section 2 introduces the 6TiSCH architecture and provides an overview of the different scheduling approaches available in the current literature for TSCH. Section 3 formally formulates the scheduling problem as a throughput maximization problem, and Section 4 describes the simulator environment and studies SFx scalability through extensive simulations. In Section 5, the *DeBraS* algorithm is described and explained in detail. Finally, Section 6 compares the result of the formal model solved as an Integer Linear Program (ILP), with both the SFx performance and the *DeBraS* performance.

2. 6TISCH AND RELATED WORK

2.1. 6TiSCH

Figure 2 presents the 6TiSCH protocol stack. The 6TiSCH architecture is composed of the IEEE 802.15.4 PHY layer, the IEEE 802.15.4e TSCH link layer, the 6Top sublayer [6], the 6LoWPAN Header Compression sublayer, the IPv6 layer [7], the transport layer and several specific applications that lay on top of the stack. By integrating all the different layers, 6TiSCH is able to form a complete functional synchronized low-power mesh network.

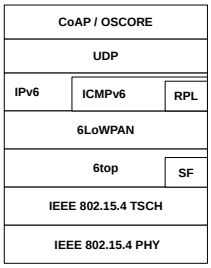


Figure 2. The IETF 6TiSCH Protocol Stack

The IEEE 802.15.4e TSCH layer sets the basis of 6TiSCH. The core mechanism is based on dividing the time in slots, of typically 10 ms duration, which are long enough for a node to send a packet of maximum size 127 bytes and receive an ACK. Timeslots are grouped in slotframes, whose duration depends on the target network. 6TiSCH defines by default a slotframe of 101 timeslots. Each timeslot is identified by an Absolute Slot Number $ASN = (k \cdot S) + t$ where k is the slotframe cycle, S the slotframe size and t is the timeslot offset. At each timeslot CH PHY channels are available. As a result a Channel Distribution Usage (CDU) matrix is formed, of which each cell is a combination of a timeslot and a channel offset. In order to perform the channel hopping, the channel offsets are mapped to frequencies by Eq. 1:

$$f = F\{(ASN + channelOffset) \ \% \ CH\} \qquad (1)$$

where F is a lookup table formed by the output of a 9-bit Linear Feedback Shift Register (LSFR) with polynomial $x^9 + x^5 + 1$ and starting seed 255. According to the standard, the slotframe size S and the number of available channels CH have to be relatively prime, such that he sequence of frequencies used in a particular cell of the CDU matrix repeats ever CH slotframes. This way, the channel hopping is represented as a static schedule. One or more cells in the schedule represent one or more links in the network, forming a Directed Acyclic Graph (DAG). Figure 3 illustrates the correspondence between the network and the CDU, for two nodes of the network.

There are 4 cell types, TX, RX, SHARED and OFF where nodes can transmit, receive or sleep:

- TX and RX cells are dedicated cells used for respectively transmitting and receiving a data packet from the upper layer and acknowledging it through a smaller ACK packet. Data packets can originate from both user or control plane.
- SHARED cells are usually used for those cases in which several nodes transmit or receive packets following a contention-based CSMA with exponential back-offs. SHARED cells can also be

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

used for sending data packets, originating from both user or control plane. In this paper, we consider SHARED cells only for signalling and assume that user data packets always use dedicated contention-free cells (TX/RX), in order to obtain guaranteed performance.

- OFF cells are non scheduled cells in which nodes turn off their radios in order to save power.

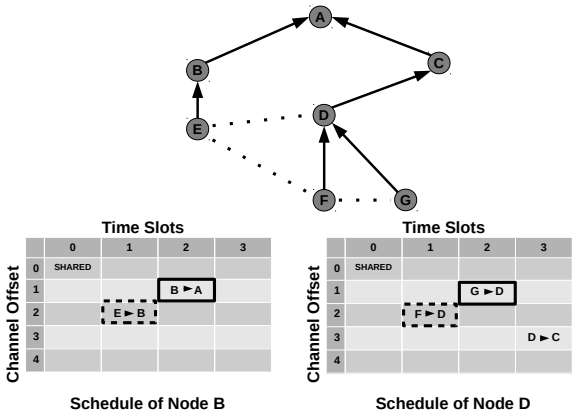


Figure 3. Example of TSCH network and CDUs at nodes B and D. At cell t=1, ch=2 collisions could occur between the links E-B and F-D since E and D are in range. However at cell t=2, ch=1 no collisions are expected between the links B-A and G-D since neither B is in the range of D nor D is in the range of A.

How to assign the cell types is the scheduler's task and usually done through the 6Top Protocol (6P), which provides the scheduler with primitives such as ADD, DELETE, LIST, etc., which can be sent from node to node through both TX and SHARED cells. An example of a 6P ADD transaction and its corresponding states is shown in Figure 4. The scheduler can be configured according to many parameters, such as topology, traffic demand and link quality. The 6P does not predefine how to build the schedule and perform these operations, but rather defines a set of Scheduling Functions (SF) which are selected by the target network and referred to with a SFID, (i.e., SF0, SF1, etc.). In the following section different scheduling approaches will be reviewed.

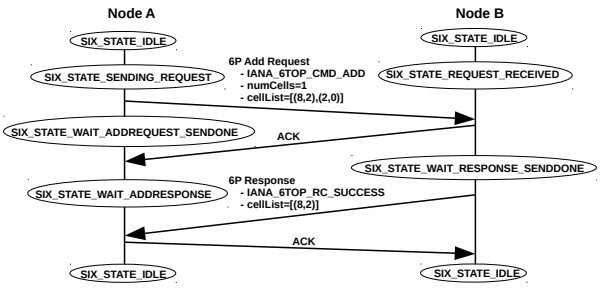


Figure 4. Two-Step 6P ADD transaction

Above the 6P, the 6LoWPAN HC layer enables transmissions of IPv6 packets over the lower IEEE 802.15.4 layers through different mechanisms of compression and address translation. As a result, 6TiSCH can operate with the versatility of a IPv6 network. On top of IPv6, together with its corresponding transport layer, the Routing Protocol for Low power IPv6 networks (RPL) is defined as the default routing protocol for 6TiSCH. Its is based on a gradient approach where a Destination-Oriented DAG (DODAG) is created according to link costs, node attributes and an objective function defined by the optimization criteria of the target network. Different objective functions can be used through different RPL instances. In order to create the topology, one or more LLN Border Routers (LBR) are designed as the DAGRoot of the tree and each node is assigned a *rank*, which decreases monotonically and specifies the cost of the link according to different metrics (e.g., distance or Expected Transmission Count (ETX)).

Upstream routes are created by DAG Information Object (DIO) messages, which are broadcasted periodically by each node and contain information of the rank. This way nodes can set a preferred parent for sending uplink traffic. Downstream routes are established by Destination Advertisement Object (DAO) Messages, which are sent as unicast traffic to back-propagate routing information from leaf nodes to the root. In order to avoid unnecessary overhead, a *trickle* algorithm is used, adapting the flooding frequency to the dynamics of the network.

2.2. Related work on TSCH-based scheduling algorithms

Configuring the schedule of every node by assigning one of the previously defined cell types to each cell, can be performed using either centralized or non-decentralized algorithms. Scheduling in WSNs is a well-studied topic in literature and we discuss those related with TSCH. We pay special attention to SFx, one of the most state of the art scheduling functions in 6TiSCH.

Centralized Algorithms

There are a number of centralized TSCH scheduling algorithms in which a central element builds the schedule for all nodes. This element has complete or partial knowledge of the network and calculates how resources in the network have to be shared. The first centralized solutions were the TSCH traditional precursors: WirelessHART and ISA100.11a [8], [9], which are application-specific, impose high-demands on hardware and of which current implementations are proprietary. More recently, other algorithms closer to 6TiSCH have been proposed. Palattella et al. present a traffic pattern and topology aware scheduling algorithm [10]. A similar approach is made in Farías et al. [11]. However, while these centralized scheduling techniques can theoretically obtain better performance, their real implementations trigger the exchange of a large amount of signalling overhead, making these networks rather poorly scalable. Therefore, centralized schedulers

are advised only for static networks, which are actually only a fraction of the real capabilities of WSNs. We refer to Section 4 for a quantitative analysis of this overhead.

Non-centralized Algorithms

On the other hand, a number of decentralized and distributed algorithms have been proposed as well, although not completely 6TiSCH compatible yet. The first non-centralized proposal was done by Tinka et al., but was never functionally implemented for general use [12]. However based on this work, several other decentralized schedulers have been introduced, such as the traffic aware scheduler from Accettura et al. [13]. In this work an interesting hierarchical approach is used to coordinate nodes sequentially in an *even-odd* fashion to void collisions. However the work of Accettura et al. it may not be suitable for irregular traffic patterns and traffic information from children nodes has to be collected recursively. This limits the scope of this approach. Other distributed algorithms have followed an RSVP-like approach such as Zand et al. [14] and Morell et al. [15], in which nodes trigger a chain of messages from source to destination to reserve cells along the path. RSVP-based approaches are robust and reliable, but can lose scheduling consistency when simultaneous operations occur. Moreover, it is prone to create a huge protocol overhead when the number of nodes increases or when some dynamism exists in the network. This may make these approaches not very scalable.

A more flexible approach, but still not fully reliant with the 6TiSCH architecture, can be found in the work of Duquennoy et al. [16], where the TSCH+RPL based *Orchestra* framework is presented. Orchestra provides a first step towards an autonomous contention-free TSCH network, which builds its schedules, locally and without signalling overhead, using only RPL information. This implies that there is no need for 6P. However it is an approach in which a sparse traffic model is assumed and reliability is reached by over-provisioning the different data planes. This means that the different schedules are built initially by a parametric trade-off static configuration designed specifically by the target network (i.e. nodes with different traffic patterns).

SFx: SF0 Experimental

Finally, a totally dynamic, adaptable and distributed 6TiSCH-reliant algorithm is proposed in Palattella et al. [17]. The *On-the-Fly* bandwidth reservation algorithm is the base for Scheduling Function Zero (SF0), SFx in its experimental form [18], which currently is still one of the most popular 6TiSCH scheduling algorithms proposed by the 6TiSCH WG that uses 6P. Recently the WG recommends the Minimal Scheduling Function (MSF) as the default SF for a given 6TiSCH network that implements the Minimal Configuration (RFC8180) [19]. However both SFs have few differences *. MSF, as SFx, translates the traffic demand to scheduling primitives to increase or reduce the number of cells used per node as needed, so that the nodes check periodically the local TX queue size in each node and add or delete cells using a threshold-based mechanism. Figure 5 shows the basis of this mechanism. Although this algorithm increases the signalling overhead compared to the Orchestra approach [16], the advantages are that the schedules are dynamically configured regardless of the traffic patterns, the scheduling is now independent of the protocol used in the routing layer and the scheduler is compatible with 6P. Different improvements have been proposed for SFx, such as a PID controller in order to more accurately define the number of cells needed per link [20], and low latency scheduling algorithms such as LLSF [21] and ReSF [22] which try to sort the scheduled cells in a row (daisy-chain) to allow packets to perform more than one hop at every slotframe to reduce end-to-end delay. It is also worth to mention the work of Phan et al. [23], where the scheduling is improved by adding the concept of density to different CDU portions. This way, in the 6P negotiation, the parent node can choose the portion with lower collision probability given by the portion with lower cell density.

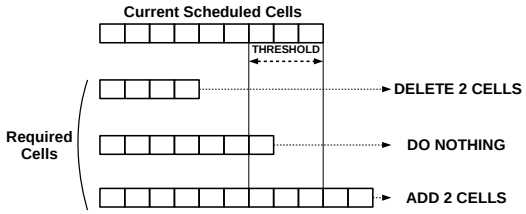


Figure 5. SFx Allocation policy

However, in all these SFx-based approaches, the nodes's local schedules are still uncoupled and collisions occur between them when the network size increases. This is because the actual cell selection is performed randomly at each node. To mitigate this effect, SFx tries to reactively relocate the collided cells relying on a collision detection procedure [24]. This procedure consists in a set of simple 6Top housekeeping rules based on the ETX metric that are periodically performed to reduce the effects of the internal interference. A more elaborated version of this mechanism has been studied and tested in Chang et al. [25]. In our work 6Top housekeeping functions for cell relocation are also used, however, our aim eventually is to pro-actively avoid cell overlapping and reduce internal interference.

Ultimately, the only 6TiSCH-based study that addresses the collision issue in large-scale and dense networks is in our previous work [5], in which we presented a first overview of the DeBraS algorithm. The current paper is a major extension of that version and differs in several ways:

*The main difference is that MSF assumes bidirectional dedicated cells meanwhile SFx assumes them unidirectional

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

First, we present a significantly modified and improved algorithm, making it more realistic in terms of its real integration with 6TiSCH and adding two different modes: Aloha-mode and TDMA-mode. Second, in order to provide a benchmark for the DeBraS algorithm, we present an optimal, but not computationally feasible in real-time, algorithm which calculates the optimal schedule for maximal throughput based on an ILP formulation. Third, a new realistic 6TiSCH compatible DeBraS protocol definition has been included. Fourth, we perform a more extensive analysis both of the scalability study and of the performance evaluation of the DeBraS algorithm. Finally, we have extended the existing open source 6TiSCH simulator with new features and open-sourced them.

3. FORMAL PROBLEM FORMULATION

In this section, the problem of assigning links to the nodes in a TSCH network is formulated as a throughput maximization problem. Links are pairwise assignments of directed communication between two nodes in a specific cell of the TSCH schedule (i.e., a timeslot and channel offset combination). This problem is a special case of the NP-hard Graph Labeling problem: we need to assign labels (i.e., slot transmissions opportunities) between two nodes in the network. The goal is to maximize the number of labels we attach, without causing interference between two conflicting transmissions. Similar problems are defined in the literature, such as [26] and [27], however they are usually formulated for different scenarios and technologies, and most of the assumptions and objective functions are not comparable. Our approach is focused on IEEE 802.15.e TSCH networks and uses a queue-based model that maximizes the throughput at the root node by selecting the optimum scheduling configuration at every node by examining the in and outgoing traffic at each queue. This makes the model robust against varying traffic patterns compared to the state of the art scheduling models. Notation used in the model is described in Table I

Table I. Problem Notation

i	node where $i \in \mathbf{N}$
e	edge where $e \in \mathbf{E}$
s(e)	node source in edge e
d(e)	node destination in edge e
p(i)	node parent of node i
$A_{t,i}$	generated packets in node i at time t
$S_{t,e}$	arrival packets from edge e at time t
$x_{t,c,e}$	tx packet in edge e in channel c at time t
$Q_{t,i}$	queue size in node i at time t

3.1. System Model and Notations

Consider a network with $N+1$ nodes, which are organized as a tree. The set of all nodes is denoted by $\mathbf{N} = \{0, 1, \ldots, N\}$, in which 0 is the root node. At every non-root node, packets are generated according to some arrival process, where $A_{t,i}$ denotes the number of packets generated in timeslot t at node i. These packets are to be sent towards the root node over the path $i \to p(i) \to p(p(i)) \to \cdots \to 0$, where $p(i)$ denotes the parent node of a non-root node i. The set of all edges is denoted by $\mathbf{E} = \{1, \ldots, N\}$. For every edge $e \in \mathbf{E}$, $s(e)$ denotes the source node and $d(e)$ the destination node of the edge. At every non-root node a queue will be maintained in which packets wait until they can be sent towards their parent node according to the TSCH transmission schedule. The number of packets in the queue of node i at time t (i.e., at the beginning of timeslot t) is denoted by $Q_{t,i}$. The system will be considered over T timeslots, where T corresponds to the number of slots in one TSCH slotframe. The number of channels available is C. $\mathbf{T} = \{0, 1, \ldots, T-1\}$ and $\mathbf{C} = \{0, 1, \ldots, C-1\}$ denote respectively the sets of considered timeslots and available channels. Every node is assumed to have R radios, implying that a node can only transmit or receive packets on at most R channels in the same timeslot.

The aim of the optimization problem is to assign pairs of timeslots and channels to the links, in such a way that with the corresponding transmission schedule, the number of packets that arrive at the root node of the tree after T timeslots is maximized. For this the following decision variables are defined:

$$x_{t,c,e} = \begin{cases} 1 & \text{if cell } (t,c) \text{ is assigned for sending} \\ & \text{on edge } e \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $t \in \mathbf{T}$, $c \in \mathbf{C}$ and $e \in \mathbf{E}$. In other words, this decision variable $x_{t,c,e}$ determines whether there is a scheduled transmission on edge *e* in channel *c* at timeslot *t*. Obviously, different nodes can send on the same timeslot and channel as long as they are not interfering. To model the impact of interference, we define a Received Signal Strength Indicator (RSSI) value $R_{i,j}$, corresponding with a certain Packet Delivery Ratio (PDR)[†].

If there is a transmission in cell (t,c) on an edge e, then all other nodes k, except the original sender ($k \neq s(e)$), that transmit in the same cell might interfere with this transmission. The interference value is given by the accumulated RSSI from all interfering nodes:

$$I_{t,c,e} = \sum_{f \in \mathbf{E} \setminus \{e\}} R_{s(f),d(e)} x_{t,c,f} \tag{3}$$

where $t \in \mathbf{T}$, $c \in \mathbf{C}$ and $e \in \mathbf{E}$. The Signal to Interference plus Noise Ratio (SINR) at the receiving node of edge e in cell (t,c) is calculated by

$$\text{SINR}_{t,c,e} = \frac{R_{s(e),d(e)}}{I_{t,c,e} + \mathcal{N}} \tag{4}$$

[†] This translation is made by using a table from a real WSN database available at http://wsn.eecs.berkeley.edu/connectivity/?dataset=dust from [17].

where $\mathcal{N}$ denotes the noise floor power[‡]. Also the SINR values are mapped to PDR values.

3.2. Constraints and Optimization Criterion

There are a couple of constraints that need to be taken into account when optimizing the link assignment.

Constraint 1: A node that is assigned a cell for receiving, is not allowed to also transmit in that cell:

$$\forall t \in \mathbf{T}, \forall c \in \mathbf{C}, \forall e \in \mathbf{E} \text{ with } d(e) \neq 0 \\ \forall f \in \mathbf{E} \text{ with } s(f) = d(e) : x_{t,c,e} + x_{t,c,f} \leq 1 \quad (5)$$

Constraint 2: The number of packets in the queue of node i at time $t+1$ should equal the number of packets in the queue at time t, minus the number of packets that left the queue during timeslot t, plus the number of packets that entered the queue during timeslot t:

$$\forall t \in \mathbf{T}, \forall i \in \mathbf{N} : Q_{t+1,i} = Q_{t,i} \\ - S_{t,(i,p(i))} + A_{t,i} + \sum_{\substack{e \in \mathbf{E} \\ d(e)=i}} S_{t,e} \quad (6)$$

where $S_{t,e}$ denotes the number of packets successfully transmitted over edge e in timeslot t.

Constraint 3: The number of packets that leave a queue during a timeslot cannot be larger than the number of packets present in that queue at the beginning of the timeslot, or the queue would underflow:

$$\forall t \in \mathbf{T}, \forall i \in \mathbf{N} : S_{t,(i,p(i))} \leq Q_{t,i} \quad (7)$$

Constraint 4: In a cell, a node can receive a packet from only one child:

$$\forall t \in \mathbf{T}, \forall c \in \mathbf{C}, \forall i \in \mathbf{N} : \sum_{\substack{e \in \mathbf{E} \\ d(e)=i}} x_{t,c,e} \leq 1 \quad (8)$$

Constraint 5: In a timeslot, a node can transmit or receive packets on at most R channels simultaneously:

$$\forall t \in \mathbf{T}, \forall i \in \mathbf{N} : \\ \sum_{c \in \mathbf{C}} \Big(\sum_{\substack{e \in \mathbf{E} \\ s(e)=i}} x_{t,c,e} + \sum_{\substack{e \in \mathbf{E} \\ d(e)=i}} x_{t,c,e} \Big) \leq R \quad (9)$$

Constraint 6: For a packet to be transmitted successfully over an edge, a cell needs to be assigned to that edge, and both the RSSI value and the SINR value corresponding to the transmission in that cell should allow successful packet reception:

$$\forall t \in \mathbf{T}, \forall e \in \mathbf{E} : S_{t,e} = \sum_{c \in \mathbf{C}} x_{t,c,e} \rho^{\text{PHY}}_{t,c,e} \rho^{\text{COL}}_{t,c,e} \quad (10)$$

where $\rho^{\text{PHY}}_{t,c,e}$ and $\rho^{\text{COL}}_{t,c,e}$ are input variables having a 0 or 1 as value. These 0 or 1 values are generated according to the PDRs corresponding to respectively $R_{s(e),d(e)}$ for $\rho^{\text{PHY}}_{t,c,e}$ and $\text{SINR}_{t,c,e}$ for $\rho^{\text{COL}}_{t,c,e}$.

Optimization criterion: The objective of the optimization problem is to maximize the total number of packets that is successfully delivered at the root node of the network in a slotframe:

$$\max \sum_{t \in \mathbf{T}} \sum_{\substack{e \in \mathbf{E} \\ d(e)=0}} S_{t,e} \quad (11)$$

3.3. Problem Linearisation

The problem as stated above is non-linear due to the feedback introduced in the decision variables $x_{t,c,e}$ by the interference of other possible transmissions. In order to linearise the problem, the following relaxation has been introduced: we assume $\rho^{\text{COL}}_{t,c,e} = 1$ in Constraint 6 (Eq. 10) and define a new Constraint 7:

Constraint 7: A cell that is assigned to an edge e can not be assigned to other edges that are in the colliding link set J_e of that edge e.

$$\forall t \in \mathbf{T}, \forall c \in \mathbf{C}, \forall e \in \mathbf{E} : \\ x_{t,c,e} + \sum_{f \in \mathbf{J}_e} x_{t,c,f} \leq 1 \quad (12)$$

The colliding link set of edge e, $\mathbf{J}_e$, is defined as the set of all links f whose source node $s(f)$ has a PDR greater than 0 with the destination node $d(e)$ of edge e:

$$\forall e \in \mathbf{E} : \mathbf{J}_e = \{f \in \mathbf{E} | PDR(s(f), d(e)) > 0\} \quad (13)$$

Applying this modification, a collision-free scenario is assumed by not allowing transmissions $x_{t,c,e}$ and $x_{t,c,f}$ when both links have a probability greater than 0 for interfering one another. Also, by introducing Constraint 7, the model solves the hidden terminal problem, as considered by other works [10, 13].

4. SIMULATION ENVIRONMENT AND SFX SCALABILITY STUDY

In order to study the performance of SFx and its limitations in scalability, we have used the open-source event-driven 6TiSCH Simulator [17], created by the members of the IETF 6TiSCH WG. It implements the most important 6TiSCH layers, although some of them, specifically RPL and 6Top, are highly simplified. As a contribution, we have improved the simulator by implementing the following new features[§]:

[‡] Noise is obtained from the thermal noise equation $N = K \cdot T \cdot BW$

[§] These new features, as well as the implementation of the *DeBras* algorithm, are available at https://github.com/imec-idlab/6tisch-simulator-extended.

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

6Top and RPL layers

We have extended the simulator with more realistic 6Top and RPL layers which actually follow the packet exchange based on their protocol descriptions [6], [7]. In the original simulator, 6P and RPL messages are assumed to happen without any delay. This is a significant difference as, in real life, important protocol delays occur. In our simulator, 6P (ADD/DELETE requests and responses) and RPL messages (DIOs) are created, enqueued and sent in SHARED or TX cells in such a way that they are now performed with a realistic delay and susceptible to suffer from collisions and propagation drops as any other packet.

Extended PHY layer

Complementary to the original Pister-Hack model [28], a new PHY layer based on the Rayleigh propagation model has been implemented, which is more suitable for Non Line Of Sight (NLOS) and mobility scenarios [29]. Additionally, in order to obtain a more realistic behaviour in the PHY layer, a dynamic variation in the RSSI received at every node has been added so that the RSSI values are not constant during the whole simulation as it was before.

Mobility models

The Random Walk mobility model (RWM) [30] and the Reference Point Group Mobility model (RPGM) [31] have been added to the simulator. RWM consists of a scenario without obstacles in which all nodes but the root move following a Brownian motion. RPGM is implemented in a scenario with obstacles where all nodes move clustered towards a common destiny guided by a Global and a Relative component. The obstacles are two big rectangles (Figure 6), which are avoided by the nodes by means of the Global component which follows a Virtual Force Field (VFF) [32]. VFF builds a set of attraction vectors (black) and repulsion vectors (red) to generate a resulting moving vector (blue). The Relative component attaches the nodes in the cluster and follows a Brownian motion.

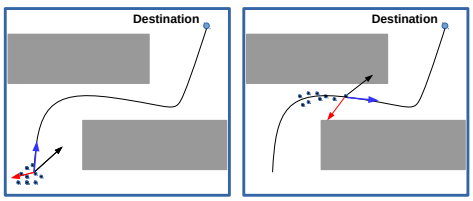


Figure 6. Example of the scenario for the RPGM simulation

Improved multichannel capabilities

In dense environments, nodes can often listen to multiple channels at once. Such multichannel capability has been added to the simulator. This feature allows to vary the number of simultaneous TX/RX per node between 1 and 16. This can be exemplified as a Software Defined Radio (SDR) node that can transmit or listen in different channels at the same time, or a node with several radios[¶].

Simulator Environment

With this tool we have evaluated the performance of the state of the art function SFx in terms of an increasing density. In order to compare the simulation results with the optimal solution in terms of throughput, we have implemented the problem formulation given in Section 3 as an Integer Linear Program (ILP), and solved it using the Gurobi library [33].

Table II. Simulation parameters

Area	Variable
Number of runs per sample	40
Simulation duration	Variable
Number of stable neighbours	1
Max number of channels	16
Sensitivity	-97 dBm
Stable link	-89 dBm (PDR $\sim$ 0.87)
Timeslot duration	10 ms
Slotframe size	101 cells
Max MAC Retries	5
Max Queue Size	10
6Top housekeeping period	5 s
App traffic model	Avg period 1.01 s
SFx threshold	0
SFx housekeeping period	5 s
RPL parents	1
RPL MinHopRankIncrease	265
RPL DIO period	5 s
Frequency band	2.4GHz
Propagation model	Friis + Pister Hack
Avg number of hops	2
Multichannel capabilities	16
Mobility model	Static

In the simulator we have evaluated the performance by simulating a range of different networks, varying parameters such as number of nodes, average number of hops or multichannel capabilities per node. The default simulation parameters used are shown in Table II. Some of the parameters are variable, such as Area and Simulation Duration, since they depend on the average number of hops used and the duration of the joining process respectively. The joining process has been implemented in such a way that only nodes that have received in a SHARED cell a DIO message (i.e., have a parent) and have successfully performed a 6P ADD Request transaction to get a dedicated TX cell to its parent, are considered joined nodes. 6P Requests are sent in SHARED cells only when dedicated TX cells are not available. Otherwise TX cells are used in order to reduce the overhead in the SHARED cells. Downwards 6P Responses are always sent in SHARED cells. For the simulations we have considered 5 SHARED cells in order to accelerate the joining process, which have been distributed in the CDU according to the optimal allocation introduced in Guglielmo et al. [34].

[¶]e.g., the devices Xilinx Zynq-7000 SoC family already support up to 16 simultaneous channels with a decent power consumption and a small form factor. Another example is Qorvo GP712 which support up to three simultaneous channels and the OpenMote B, which supports simultaneous dual radio operation.

By default, the traffic model used is periodic uplink UDP traffic with 127-byte packet size and with an average of 1 packet per second with a variance of 0.02 seconds. This configuration has been selected in order to obtain network saturation, and reach the worst case scenarios, e.g., a densely crowded network with high traffic demands. Variable traffic patterns are generated using a long-tailed Pareto distribution with a Hurst parameter H=0.6, which, according to the Q.3925 ITU-T recommendation [35], corresponds to peer-to-peer traffic. For bursty traffic, a combination of 1 packet per second periodic traffic and a burst of 10 packets generated at a random time in every node is considered.

Regarding the multichannel capabilities, by default it is considered that nodes can transmit or receive in up to 16 channels at every timeslot. This is possible since, in a standard 6TiSCH timeslot of 10 ms, there are about 1.96 ms between *ti9* and *ti1*, which for a average microprocessor is more than enough for enqueuing, compute CRC, prepend MAC headers, etc. simultaneously for different packets. Different assumptions can be made, from having multichannel capabilities at every node, to have them only in the root node. We have assumed in this paper to have multichannel capabilities in every node in order to obtain the best performance and dodge the bottlenecks. In spite of this assumption, there is no need to use 16 channels in all the nodes in a real network. Network nodes will never use more than 5 channels at a time, which is the integer part of 16 channels divided by 3 hops. For more that 3 hops, nodes can reuse channels without interfering one each other [13, 36, 37]. Only the root would theoretically need to have up to 16 channels. Network nodes would suffice with having 1 or just few simultaneous channels. Moreover, assuming 16 channels in a gateway is widely accepted in different IoT technologies. In order to simplify, in the following sections the multichannel capabilities will be referred to simply as $numChannels$.

Finally, networks are preconfigured by default with nodes at 2 hops from the root on average by varying the distance between them. This means that the actual number of hops at every node can also be lower or higher than two hops. The topologies used when solving the ILPs are the same as the ones used in the simulator.

4.1. Network limits

Once the simulator scenarios and the ILP are described we begin stressing the network to test its scalability, reaching the network limits. Since the maximum theoretical performance is expected for 1 hop networks, a star topology is tested for different network sizes in order to evaluate the maximal theoretical average throughput per node achievable in the root node. This is illustrated in Figure 7, where we show results for two different versions: the outcome of the ILP (without any simulation) and a simulation of SFx. As expected the limitations are the $numChannels$ in the root node. For every $numChannels$, it is possible to see that the average throughput per node curve starts to decay when $101 \cdot numChannels$ is reached. For example, with $numChannels = 1$ per node, the optimum throughput is only achievable up to 100 sending nodes, which is evident since the root node can only have up to $101 \cdot numChannels$ RX cells per frame. When averaging over more nodes than 100, the throughput received in the root does not grow as fast as the overall traffic input introduced for all nodes.

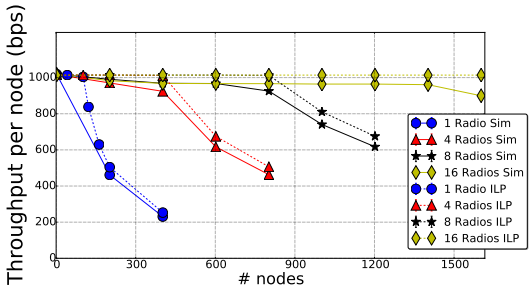


Figure 7. Average throughput per node for different number of radios per node

The difference in Figure 7 between the throughput obtained by the simulation and by the ILP is due to the more realistic behaviour of the simulator. First, in the simulator signalling traffic from 6P and RPL is present, which partially increases the load in the network. Second, SFx uses over-provisioning to avoid dropping packets when the queues are full. This triggers a larger number of cell reservations which accelerates the depletion of available cells in the the root node and reduces the overall throughput in the network.

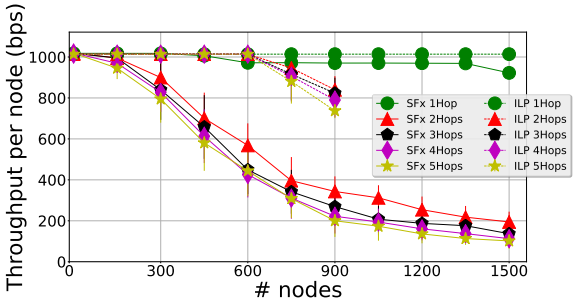
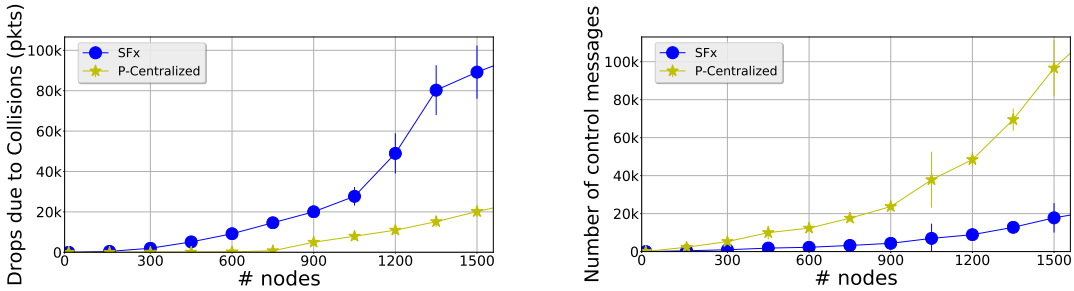


Figure 8. Average throughput per node for different average number of hops

While the 1-hop scenario renders the highest throughput, for multi-hop networks the performance in throughput becomes notably lower in terms of the average number of hops. The higher the average number of hops of the network topology, the more cells are required, and therefore, saturation is reached much faster. Since the cell usage is notably higher, the number of collision increases. Drops caused by packet collisions are one of the major issues in non-star topologies due to the fact that SFx randomly

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls



(a) Number of Collisions **(b)** Scheduling overhead

Figure 10. Comparison between the SFx and *P-centralized* algorithms

selects cells without considering possible interference with nodes other than the child node and parent node. Drops caused by both propagation and collisions are partially mitigated by retransmissions when the number of drops is low. However, retransmissions reinforce and delay the problem to upper layers, converting MAC drops to queue drops. As mentioned, SFx tries to reduce the queue drops by over-allocating cells, but a cell surplus in each link means using more cells than are actually needed which increases inefficiency and aggravates the problem. As shown in Figure 8 all these factors cause an important reduction in the performance for networks with more than 100 nodes and with topologies with more than 1 hop on average. For the ILP in non-star topologies, only values up to 900 nodes are shown due to computational limitations, however evidence that there is a big margin of improvement.

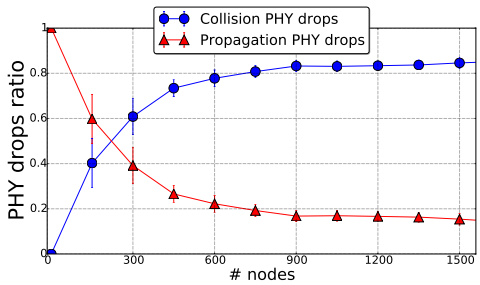


Figure 9. Ratio of packet drops due to propagation (PHY) and collisions (COL)

Comparing the difference in the performance between the collision-free analytical model results with the simulation results, reveals a notable poor network scalability and results evident that a mechanism for avoiding collisions is needed. This is confirmed in Figure 9, which shows that when the network size increases, the drops due to collisions become predominant while the drops due to propagation tend to have less significance regarding the absolute number of PHY drops.

4.2. Avoiding Internal Interference

As presented in the previous subsection, avoiding collisions is crucial for high-reliable and scalable networks. 6TiSCH provides some basic mechanisms to reduce collisions. However when scaling up the network, these mechanisms are counter-productive. Drops due to collisions result in higher ETXs per link, which will trigger additional signalling traffic due to RPL parent changes, 6Top relocations (if a relocation algorithm is present [24]), and SFx over-provisioning transactions. This additional traffic turns the network unmanageable.

Since collisions occur when different nodes in the same wireless range are transmitting in the same timeslot using the same channel, the ideal solution is to synchronize nodes to avoid this overlapping, as it is done in the analytical problem. The only scheduling approach that allows to implement a solution for the analytical problem is the centralized approach. However ILPs are non realistic since don't have all the protocol overhead and it is not feasible to solve them in real time. In order to asses the effect of collisions in simulation time on the performance of more realistic solutions, we have implemented a pseudo-optimal scheduling algorithm called *P-centralized*, which do not render the optimal throughput but however avoids internal interference by having global real time scheduling information in all nodes.

By having total knowledge of the network, nodes can allocate accordingly cells avoiding physical overlapping, meaning that two different links can use the same cell at the same time if their respective nodes are not in range. However, when nodes try to allocate cells and none is available, the *P-centralized* algorithm still chooses a random cell, even knowing that collision will occur. This is because there is always a probability, to get the packet correctly received even when interference exist.

Figure 10 a) shows the evolution of collisions for the SFx and for the *P-centralized* algorithm. For the *P-centralized* algorithm no collisions are observed until the network size is larger than 800 nodes. This value is defined in each network by the number of hops and its degree of sparsity (directly related with TX power and sensitivity). Still, also with less than 800 nodes some

collisions are noticeable in *P-centralized*. This is because 6P introduces delay and concurrent allocations occur. Although a centralized approach could seem the best option, real-time information is in practice not available in the root and would also involve a huge amount of overhead, making it non-viable in large-scale WSNs. Figure 10 b) shows how the signalling traffic becomes unmanageable in the centralized approach. Eventually, even neglecting these issues, the scheduling problem is a NP-problem as stated previously in Section 3, which makes a totally centralized approach infeasible.

5. DEBRAS ALGORITHM

Results presented in the previous chapter evidence the need of reducing cell overlapping in a distributed manner. The DeBraS algorithm is an heuristic approach with low computational load that reduces the internal interference notably without increasing the network load heavily, and without needing total real-time information from all nodes.

The basis of this approach is to locally broadcast scheduling information encapsulated in DeBraS messages using specific SHARED cells called DeBraS cells, which are distributed as shown in Figure 11. This way physical neighbours that do not share the same parent, can avoid interfering each other, by eavesdropping in their physical neighbourhood and being aware of which is the actual CDU usage of their neighbours. The algorithm is composed of two parts, a mechanism for initial network configuration and a scheduling information spreading procedure via DeBraS messages.

Algorithm 1 Initial Cell Configuration:

```
1: procedure PREPARE BROADCAST CELLS
2:     brID = 0
3:     for ch_b = 0 : numChannels-1 do
4:         ts_b = 1
5:         for n =0 : numBrCells do
6:             if brID < numNodes then
7:                 addCellToSchedule(ts_b,ch_b,SHARED,brID)
8:                 ts_b+=⌈slotFrameLength/numBrCells⌉
9:                 brID += 1
10:            end if
11:        end for
12:    end for
13:    if DeBraS-ALOHA-Mode == True then
14:        maxWin= 4 · ⌈ numNodes/numBrCells ⌉
15:    end if
16:    if DeBraS-TDMA-Mode == True then
17:        maxWin= ⌈ numNodes/brID ⌉
18:        numWaitings= ⌈ node.id / brID ⌉
19:        for cell in schedule.cells() do
20:            if cell.id == ( node.id % brID ) then
21:                brTs=cell.ts
22:                brCh=cell.ch
23:            end if
24:        end for
25:    end if
26: end procedure
```

The Initial Network Configuration, described in Algorithm 1, can be performed in two ways: by following a Aloha-based approach (DeBraS-Aloha) and by following a TDMA-based approach (DeBraS-TDMA). DeBraS-Aloha is totally distributed and allows nodes to share the DeBraS cells stochastically following a simple contention-based medium access based on a congestion window. This Aloha-based approach allows to increase the DeBraS cells usage and leverage from spatial multiplexing, at the expense however of increasing DeBraS message collisions. The DeBraS-TDMA approach predefines how a node uses the DeBraS cells every time joins the network. DeBraS-TDMA allocates each DeBraS cell in the CDU to a different node in a distributed manner, but based on knowledge of the maximum number of nodes in the network and the node ID. The DeBraS-TDMA approach will enable a complete collision-free channel for nodes send to DeBraS messages. Every node can transmit a DeBraS message in its assigned DeBraS cell during its assigned slotframe (*numWaitings*). During the remaining DeBraS cells, nodes listen to other nodes in their range to receive DeBraS messages. How often the nodes send their DeBraS messages depends on the number of nodes in the network and hence, the number of DeBraS cells. Notice in Algorithm 1 that ts_b is initialized to 1 in order to allocate the DeBraS cells always 1 timeslot further than the SHARED cells used for signalling.

Algorithm 2 Schedule Spreading: DeBraS-Aloha

```
1: procedure SHARED ACTIVE CELL (CELL)
2:     if cell.type == DeBraS then
3:         if numWaitings == 0 then
4:             prepareSendBrMessage(schedule)
5:             numWaitings=random(1, maxWin)
6:         else
7:             numWaitings-=1
8:             prepareReceiveMessage()
9:         end if
10:    end if
11: end procedure
```

Algorithm 3 Schedule Spreading: DeBraS-TDMA

```
1: procedure SHARED ACTIVE CELL (CELL)
2:     if cell.type == DeBraS then
3:         if cell.Ts == brTs && cell.Ch == brCh then
4:             if numWaitings == 0 then
5:                 prepareSendBrMessage(schedule)
6:                 numWaitings=maxWin-1
7:             else
8:                 numWaitings-=1
9:                 prepareReceiveMessage()
10:            end if
11:        else
12:            prepareReceiveMessage()
13:        end if
14:    end if
15: end procedure
```


Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

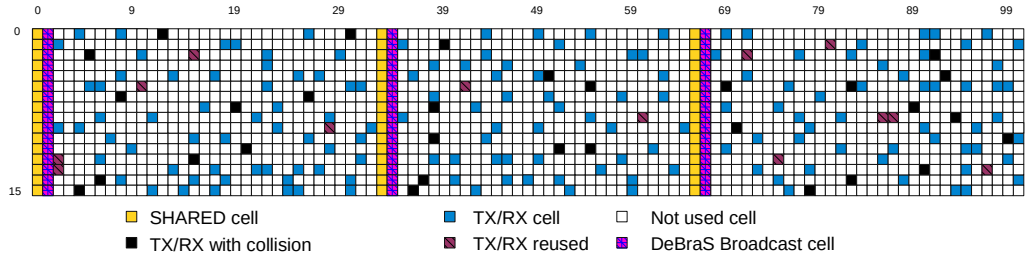


Figure 11. Example of a CDU with 15 channels and 101 timeslots and how cells are distributed with the *DeBraS* Algorithm

The Schedule Update procedure, defined in Algorithm 2 and 3, is executed for every DeBraS cell and consists simply of sequentially sending or listening to DeBraS messages in the DeBraS cells previously defined (with its respective timeslot $brTs$ and channel offset $brCh$). Whenever a DeBraS message is successfully received, nodes will update their neighbourhood schedule, adding the used cells and increasing the *Freshness* of the cell, a metric used to allow the nodes discern potentially colliding cells from unlikely colliding cells. The neighbourhood schedule is a parallel schedule built at every node that allows for every 6P transaction, to exclude the cell set that will most likely collide with the candidate cells given by SFx. This way, whenever a new cell is requested, end-point nodes in each link not only will avoid their used cells but also the ones used in their wireless neighbourhood. In case a node when receiving a 6P ADD Request does not have cells available after applying the neighbourhood cell set exclusion, the node will choose randomly from the cells with lower *Freshness*. A cell with low *Freshness* usually means that after a number of slot-frames, the cell has become available again and is no longer used by any known neighbour.

In order to share the scheduling information according to the 6TiSCH stack, a new DeBraS packet has been defined, which is sent in the SHARED DeBraS cells and does not compete with other signalling traffic such as RPL or 6P. DeBraS packets are carried as payload of a 802.15.4 Payload Information Element (IE) [38] and travel encrypted over a single hop. Since each packet has a maximum size of 127 bytes, from which 16 bytes are used for the MAC header, the maximum allowed DeBraS payload is 111 bytes. Figure 12 and Table III show the packet structure.

The scheduling information is in the Info Cell field. This 3-byte field contains the Cell ID, which ranges from 0 to 1615, since maximum 101 $Slots \cdot 16$ $numChannels$ cells are considered, the Cell Type and the Neighbour ID. Similarly, DeBraS assumes a maximum of 1616 nodes, since is the maximum theoretical number of dedicated cells that could be scheduled in the root for the star topology and complete multi channel capabilities (best case scenario). Due to the limitations in the packet size, a maximum of 36 Cell Info fields are expected. Eq. 14 defines how many DeBraS messages are needed for a given schedule.

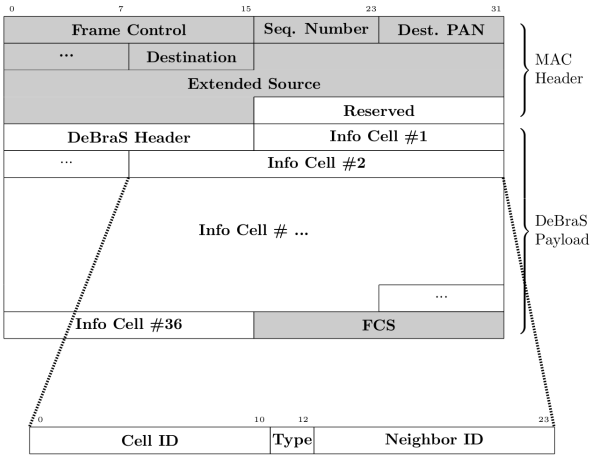


Figure 12. DeBraS packet structure

$$nMessages = \frac{(numTXcells + numRXcells)}{36} \quad (14)$$

1	00000000000
...	...
1616	11001010000

(a) Cell IDs

TX	01
RX	10
SHARED	11

(b) Cell Types

1	00000000000
...	...
1616	11001010000

(c) Neighbor IDs

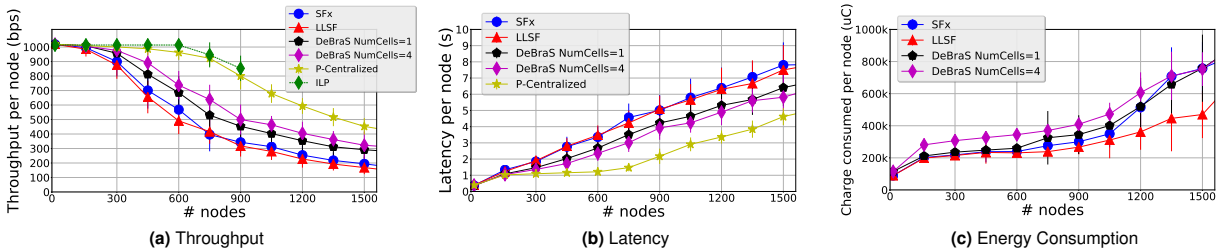
Table III. Info Cell Field structure

In the worst case scenario, a node will need a number of DeBraS messages lower than 40 to successfully broadcast its complete schedule, since less than 40 messages are needed for a complete CDU excluding SHARED cells.

6. PERFORMANCE EVALUATION

6.1. DeBraS-Aloha vs DeBraS-TDMA

Before comparing DeBraS with SFx, it is necessary to select the best approach among DeBraS-Aloha and DeBraS-TDMA. Preliminary results show that DeBraS-TDMA performs slightly better than DeBraS-Aloha



(a) Throughput **(b)** Latency **(c)** Energy Consumption

Figure 15. *DeBraS* Performance for different network sizes in terms of Throughput, Latency and Energy Consumption.

for the different hops (Figure 13). This difference is because, although the DeBraS-TDMA approach allows to sequentially send DeBraS messages without collisions, only one node in the network is transmitting at a time in a specific DeBraS cell, which means that DeBraS cells are underutilized. Although DeBraS-Aloha has more collisions when sending DeBraS messages, it leverages from the spatial multiplexion and fosters almost equivalent throughput than DeBraS-TDMA. Since we are focusing on distributed scheduling, we favor an ALOHA-based approach. Therefore in the remainder of this article we will always refer to the DeBraS-Aloha approach.

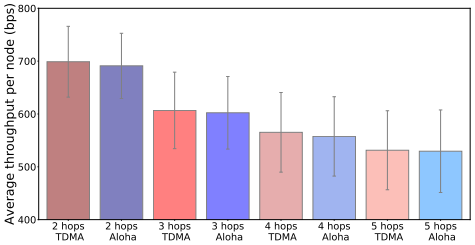


Figure 13. Throughput comparison between DeBraS-Aloha and DeBraS-TDMA for different topologies

6.2. Scheduler Performance

By using DeBraS, less collisions are expected, which is translated into higher throughput at the root. This performance will vary mainly regarding the number of DeBraS cells used in the CDU. Figure 14 shows different averaged throughput values per node for different number of DeBraS cells. Values have been averaged for networks sizes from 2 to 1616 nodes and the environment for the multichannel configuration from Section 4 is used. Although all DeBraS configurations foster up to 17% higher throughput than SFx, the maximum throughput value is achieved for 4 DeBraS cells per channel, where the improvement is 24.8% higher than SFx on average for all network sizes. However for more than 4 DeBraS cells, the performance decreases. This is because when using DeBraS, more collisions are avoided since nodes can transmit and receive scheduling information more frequently. However, using too many DeBraS cells implies that less cells are available for transmitting user data.

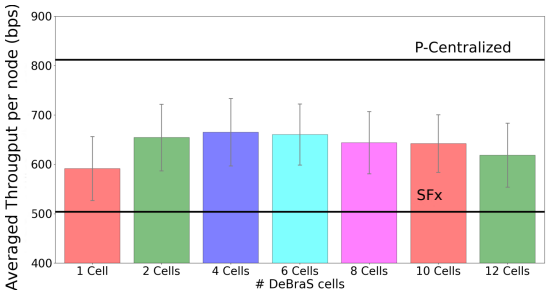


Figure 14. Averaged Throughput for different number of DeBraS cells per channel

Also, results of DeBraS performance are shown in Figure 15 regarding the network size. The performance of SFx, LLSF, the *P-Centralized* algorithm and the solution of the ILP (up to 900 nodes) are shown as reference values. Figure 15 a) shows that throughput improvements regarding SFx are significant for all network sizes, being up to 33.8% and 61.1% higher for 1 and 4 DeBraS cells per channel respectively for networks with 750 nodes. As can be seen, LLSF performance in terms of throughput is similar to SFx since, although the allocation is done in order to minimize delay, cell overlapping still occurs in the same manner as in SFx. On the other hand, as expected, DeBraS performance is always lower when comparing with the ILP and the P-Centralized algorithm. This is because with DeBraS, scheduling information is still not perfectly shared in real time between nodes. There are three consequences of not having real-time scheduling information available. First, as the network size increases, nodes can not send their DeBraS messages in every frame and hence their neighbour nodes do not have fully updated information at every frame. Second, nodes that are not able to send a DeBraS message to a node in the edge of its range properly, can still interfere it and provoke collisions. This is because the transmission and interference ranges of a node are usually not equal. Finally, the packet size limitation does not allow to send the whole schedule when more than 36 cells are allocated. This implies that in order to successfully transmit the schedule, several DeBraS messages have to be transmitted.

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

Equivalent results for latency are shown in Figure 15 b), where for the case of a network size of 750 nodes, the reduction is 23% and 34% for 1 and 4 DeBraS cells respectively. Reductions in latency for all the network sizes are because the reduction in collisions is directly translated in less retransmissions and hence, lower end-to-end delay. Latency in LLSF is lower than in SFx as expected, however not significantly. This is because when running LLSF in a distributed manner, "daisy-chains" end-to-end reservations are not possible since nodes only try to allocate TX cells after RX cells, but without considering bad quality cells and their global position in the tree.

However, these improvements in performance come at the cost of an increase in energy consumption. Instinctively, since DeBraS cells are always used for either transmitting or receiving DeBraS messages, the consumption will be higher, although comparable with the consumption in SFx and LLSF. As can be observed in Figure 15 c), for low node density scenarios, the consumption using DeBraS is significantly higher than SFx. However for larger network sizes, the consumption per node tends to be equivalent since retransmissions and 6P overhead increases the consumption in SFx. LLSF have lower consumption for high network sizes, however at the expense of the performance, since less cells are scheduled due to the impossibility of scheduling RX cells before TX cells when the CDU becomes full.

6.3. DeBraS Robustness against Mobility and Variable Traffic

In this section, we investigate DeBraS's performance under more challenging conditions such as with a high level of mobility and variable traffic. Results shown in Figure 16 illustrate that the performance of DeBraS with 4 DeBraS cells is always higher than that of SFx for the three different traffic types: constant, variable Pareto and bursty. The improvement in performance is shown in terms of number collisions. This difference in collisions lays in the amount of 6P operations performed. Variable and bursty traffic significantly increases the number of 6P operations, increasing as well the probability of allocating colliding cells. DeBraS however, can perform 6P operations reducing the number of scheduling collisions, which renders a lower number of packet drops.

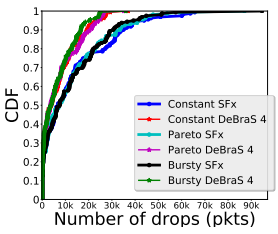


Figure 16. Number of drops due to collisions for different traffic patterns

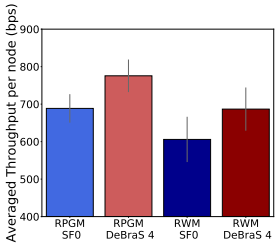


Figure 17. Evaluation for the different mobility models

Similarly, we have tested the RWM and RPGM mobility models for both SFx and DeBraS and compared the different performances in Figure 17. As illustrated, DeBraS still offers better performance than SFx in these scenarios when averaging the throughput per node for the different network sizes. This is due to RWM, RPGM have higher parent change rate than a static scenario, and when relocating cells, DeBraS offers lower collision probabilities than SFx. Furthermore for RPGM, when simulating scenarios, nodes tend to group when moving for avoiding the obstacles, which increases significantly the network density. More density means more collisions, and when using DeBraS, collisions are lower.

6.4. DeBraS performance for different multichannel capabilities

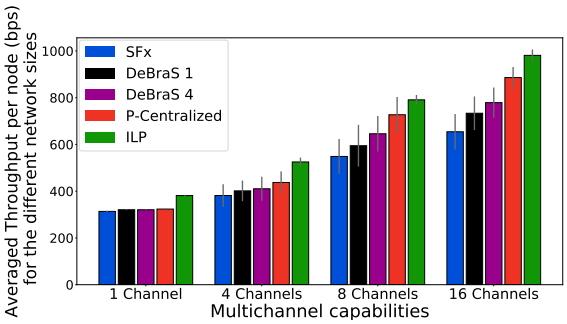


Figure 18. Throughput per node for different number of simultaneous channels when averaging network sizes from 2 to 900 nodes

Different number of $numChans$ in the devices offer different overall network performance, specially in the root node. In order to illustrate this, Figure 18 compares how the different multichannel capabilities influence the performance when averaging for all network sizes, from 2 to 900 nodes in terms of average throughput per node. The performance evidences, that the gain of DeBraS is the highest when nodes could use 16 simultaneous channels. For values lower than 16 channels, there is already a significant performance improvement from 4 channels onwards. Since the values are averaged for network sizes from 2 to 900 nodes, the bottleneck created in the root due to the lack of channels as shown in Figure 7, masks the effect of collisions when only using 1 simultaneous channel in the root. For more simultaneous channels, the differences become evident. Although we are assuming that nodes have multichannel capabilities, as stated in Section 4, in real network nodes would only need 1 or just few channels, being only the root node the only node that would ever theoretically need to use these 16 channels simultaneously. Only when the number of channels in the root node is low, the bottleneck appears. By using only single-channel devices, performance problems arise quickly whenever the network grows (even below 100 nodes). If network scalability needs to be improved,

the multichannel approach becomes highly relevant and our DeBraS algorithm a possible alternative to improve the performance. In order to illustrate that multichannel capabilities are fundamental for network scalability, Figure 19 shows the zoom-in performance of SFx around 100 nodes for three different assumptions: single channel nodes, single channel nodes with a multichannel root node and multichannel capabilities in all nodes. It proves that the scalability in a single channel network is notably limited due to the mentioned bottleneck, which appears in network sizes higher than 80 nodes. Figure 19 also shows that, by using 16 channels only in the root node, the performance can be equivalent to have all nodes with 16 simultaneous channels, which makes makes the proposed approach more practically applicable.

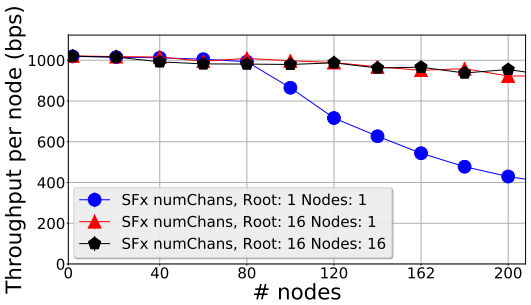


Figure 19. Zoom-in around 100 nodes in the throughput per node comparison for the three different multichannel assumptions

7. CONCLUSION

In this work we have studied the performance of 6TiSCH in large-scale and dense networks, emphasizing on the scalability of this technology. The results showed that state of the art 6TiSCH distributed scheduling algorithm SFx heavily suffer from cell overlapping when the number of nodes increases due to the lack of coordination between them. To address this, the DeBraS scheduling algorithm has been proposed to mitigate collisions and increase network scalability, significantly outperforming current 6TiSCH distributed scheduling algorithms in up to 1.61 times in terms of throughput for large network size scenarios. One example scenario that would leverage from DeBraS is a network formed by hundreds of people (wearables) densely placed where maximum resource usage is crucial. As a drawback, DeBraS involves an increase of energy consumption due to a higher overhead in the network. In this work we have extended the study of DeBraS by analysing different DeBraS configurations and network scenarios. The results have shown that DeBraS still outperforms the current scheduling algorithms, with significant improvements in challenging environments such as mobile networks or for networks with complex traffic patterns. As such, DeBraS represents a suitable approach for current and future highly demanding high-dense and large scale IoT scenarios.

8. ACKNOWLEDGEMENTS

Part of this research was funded by the ICON project CONAMO, co-funded by imec and VLAIO.

REFERENCES

1. Watteyne T, Mehta A, Pister K. Reliability through frequency diversity: why channel hopping makes sense. *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ACM, 2009; 116–123.
2. Pister K, Doherty L. TSMP: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks* ; .
3. Dujovne D, Watteyne T, Vilajosana X, Thubert P. 6TiSCH: deterministic IP-enabled industrial internet. *IEEE Communications Magazine* December 2014; .
4. Paventhan A, Krishna H, Pahuja N, Khan MF, Jain A, *et al.*. Experimental evaluation of IETF 6TiSCH in the context of smart grid. *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, IEEE, 2015.
5. Municio E, Latré S. Decentralized broadcast-based scheduling for dense multi-hop TSCH networks. *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, ACM, 2016; 19–24.
6. Wang Q, Vilajosana X, Watteyne T. 6top Protocol (6P). *Internet-Draft draft-ietf-6tisch-6top-protocol-09*, Internet Engineering Task Force Oct 2017.
7. Winter T. RPL: IPv6 routing protocol for low-power and lossy networks 2012; .
8. Specification W. 75: TDMA Data-Link Layer. *HART Communication Foundation Std., Rev* 2008; **1**(1).
9. *Standard ISA-100.11a-2011: Wireless Systems for Industrial Automation : Process Control and Related Applications : 2011*. ISA, 2011.
10. Palattella MR, Accettura N, Dohler M, Grieco LA, Boggia G. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, IEEE, 2012; 327–332.
11. Farías ÁA, Dujovne D. A queue-based scheduling algorithm for PCE-enabled Industrial Internet of Things networks. *Embedded Systems (CASE), 2015 Sixth Argentine Conference on*, IEEE, 2015; 31–36.
12. Tinka A, Watteyne T, Pister K. A decentralized scheduling algorithm for time synchronized channel hopping. *International Conference on Ad Hoc Networks*, Springer, 2010; 201–216.
13. Accettura N, Vogli E, Palattella MR, Grieco LA, Boggia G, Dohler M. Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation. *IEEE Internet of Things Journal* 2015; **2**(6):455–470.
14. Zand P, Dilo A, Havinga P. D-MSR: A distributed network management scheme for real-time monitoring and process control applications in wireless industrial automation. *Sensors* 2013; **13**(7).

15. Morell A, Vilajosana X, Vicario JL, Watteyne T. Label switching over IEEE802.15.4e networks. *Transactions on Emerging Telecommunications Technologies* 2013; **24**(5):458–475.
16. Duquennoy S, Al Nahas B, Landsiedel O, Watteyne T. Orchestra: Robust mesh networks through autonomously scheduled tsch. *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ACM, 2015; 337–350.
17. Palattella MR, Watteyne T, Wang Q, Muraoka K, Accettura N, Dujovne D, Grieco LA, Engel T. On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks. *IEEE Sensors Journal* 2016; .
18. Dujovne D, Grieco LA, Palattella MR, Accettura N. 6TiSCH 6top Scheduling Function Zero / Experimental (SFX). *Internet-Draft draft-ietf-6tisch-6top-sfx-00*, Internet Engineering Task Force Sep 2017. Work in Progress.
19. Vilajosana X, Pister K, Watteyne T. Minimal IPv6 over the TSCH Mode of IEEE 802.15. 4e (6tisch) Configuration. *Technical Report* 2017.
20. Domingo-Prieto M, Chang T, Vilajosana X, Watteyne T. Distributed pid-based scheduling for 6tisch networks. *IEEE Communications Letters* 2016; .
21. Chang T, Watteyne T, Wang Q, Vilajosana X. LLSF: Low Latency Scheduling Function for 6TiSCH Networks. *Distributed Computing in Sensor Systems (DCOSS)*, IEEE, 2016.
22. Daneels G, Spinnewyn B, Latré S, Famaey J. ReSF: Recurrent low-latency scheduling in ieee 802.15. 4e tsch networks. *Ad Hoc Networks* 2018; **69**:100–114.
23. Duy TP, Dinh T, Kim Y. Distributed cell selection for scheduling function in 6TiSCH networks. *Computer Standards & Interfaces* 2017; **53**:80–88.
24. Muraoka K, Watteyne T, Accettura N, Vilajosana X, Pister KS. Simple distributed scheduling with collision detection in tsch networks. *IEEE Sensors Journal* 2016; **16**(15):5848–5849.
25. Chang T, Watteyne T, Vilajosana X, Wang Q. Ccr: Cost-aware cell relocation in 6tisch networks. *Transactions on Emerging Telecommunications Technologies* 2017; .
26. Kodialam M, Nandagopal T. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. *Proceedings of the 11th annual international conference on Mobile computing and networking*, ACM, 2005; 73–87.
27. Sasaki M, Furuta T, Takamori U, Ishizaki F. TDMA scheduling problem avoiding interference in multi-hop wireless sensor networks. *Journal of Advanced Mechanical Design* 2016; .
28. Le HP, John M, Pister K. Energy-Aware Routing in Wireless Sensor Networks with Adaptive Energy-Slope Control. *EE290Q-2 Spring* 2009; .
29. Stüber GL. *Principles of mobile communication*, vol. 2. Springer, 2001.
30. Sichitiu ML. Mobility models for ad hoc networks. *Guide to Wireless Ad Hoc Networks*. Springer, 2009.
31. Hong X, Gerla M, Pei G, Chiang CC. A group mobility model for ad hoc wireless networks. *International workshop on Modeling, analysis and simulation of wireless*, ACM, 1999.
32. Borenstein J, Koren Y. Real-time obstacle avoidance for fast mobile robots in cluttered environments. *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, 1990.
33. Optimization G, *et al.*. Gurobi optimizer reference manual. *URL: http://www. gurobi. com* 2012; **2**:1–3.
34. De Guglielmo D, Brienza S, Anastasi G. A Model-based Beacon Scheduling algorithm for IEEE 802.15. 4e TSCH networks. *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A*, IEEE, 2016; 1–9.
35. Recommendation ITU-T Q.3925, Traffic flow types for testing quality of service parameters on model networks. *Technical Report* 2012.
36. Namboothiri PG, Krishna MS. Capacity analysis of multi-hop wireless sensor networks using multiple transmission channels: A case study using ieee 802.15. 4 based networks. *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, IEEE, 2010; 168–171.
37. Slimane JB, Songi YQ, Koubaa A. Control and data channels allocation for large-scale uwb-based wsns. *Communications and Networking, 2009. ComNet 2009. First International Conference on*, IEEE, 2009; 1–8.
38. IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)* April 2012; :1–225.