Experimental Validation of a Distributed Self-Configured 6TiSCH with Traffic Isolation in Low Power Lossy Networks

Fabrice Theoleyre
CNRS, ICube lab
University of Strasbourg
Strasbourg, France
theoleyre@unistra.fr

Georgios Z. Papadopoulos
IRISA, Télécom Bretagne
Institut Mines Télécom
Rennes, France
georgios.papadopoulos@telecom-bretagne.eu

ABSTRACT

Time Slotted Channel Hopping (TSCH) is among the proposed Medium Access Control (MAC) layer protocols of the IEEE 802.15.4-2015 standard for low-power wireless communications in Internet of Things (IoT). TSCH aims to guarantee high network reliability by exploiting channel hopping and keeping the nodes time-synchronized at the MAC layer. In this paper, we focus on the traffic isolation issue, where several clients and applications may cohabit under the same wireless infrastructure without impacting each other. To this end, we present an autonomous version of 6TiSCH where each device uses only local information to select their timeslots. Moreover, we exploit 6TiSCH tracks to guarantee flow isolation, defining the concept of shared (best-effort) and dedicated (isolated) tracks. Our thorough experimental performance evaluation campaign, conducted over the open and large scale FIT IoT-LAB testbed (by employing the OpenWSN), highlight the interest of this solution to provide reliability and low delay while not relying on any centralized component.

Keywords

IoT; 6TiSCH; IEEE802.15.4-2015-TSCH; distributed reservation; self-configuration; traffic adaptive; experiments;

1. INTRODUCTION

During the last years we have experienced the emergence of a new paradigm called Internet of Things (IoT) in which smart and connected objects cooperatively construct a (wireless) network of things [1].

In an IoT deployment, energy-efficiency is one of the most important parameters, since smart objects have to save energy to meet the lifetime requirements of typical applications. Since MAC layer is responsible for controlling the main source of energy consumption [2], therefore, a strong focus has been put on the medium access: the devices should turn OFF their radio for most of the time.

MSWiM '16, November 13–17, 2016, Malta, Malta.

ISBN 978-1-4503-4502-6/16/11…$15.00

DOI: http://dx.doi.org/10.1145/2988287.2989133

Besides, many modern applications cannot accommodate the best effort approach. For instance, losing one packet for a smart meter application means the measures are incorrect for this subscriber. More importantly, several applications or even clients should cohabit under the same wireless infrastructure to reduce the cost of deployment [3]. These requirements lead researchers to design deterministic algorithms for medium access. Thus, the wireless infrastructure is able to provide *stable* and *predictable* performance.

IEEE 802.15.4-2015 standard was published in 2016 [4], to offer a certain quality of service for deterministic industrial-type applications. Among the MAC schemes defined in this standard, Time-Slotted Channel Hopping (TSCH) is for lower-power and reliable networking solutions. Indeed, by adopting a channel hopping approach, the standard makes the transmissions more reliable [5]. Besides, TSCH relies on scheduling where the contention is solved deterministically, and each radio link receives a given amount of bandwidth. Thus, TSCH is particularly efficient to save energy: a node has to stay awake only when it transmits or receives a frame.

6TiSCH IETF Working Group (WG) is currently defining a set of protocols to fill the gap between IPv6 and TSCH [6]. It aims at executing IPv6 over a reservation based MAC by reserving TSCH slots. Moreover, 6TiSCH introduces the concept of tracks, where certain timeslots are mapped to a given flow, guaranteeing thus traffic isolation. Different flows from different applications are mapped to different tracks. We consider this feature as a key enabler for reliability in IoT. Recently, Orchestra proposed an innovative way to exploit the 6TiSCH architecture, highlighting the relevance of a distributed version [7]. However, it does not guarantee traffic isolation since it does not implement any track management solution.

Our 6TiSCH version is resilient: we propose to adapt the schedule locally, based on the radio link quality and the actual amount of traffic to forward. We do not rely on a centralized component which would increase the delay to react to changes. In this paper, we present a distributed version of 6TiSCH to guarantee traffic isolation and deterministic performance.

The contributions presented in this paper are threefold:

1. We first propose a distributed implementation of 6TiSCH with traffic isolation using tracks, where each pair of nodes negotiates the required bandwidth.

2. We then describe mechanisms so that each node decides locally which timeslot to use, and for which track;

3. We perform a thorough experimental evaluation over the FIT IoT-LAB testbed, to highlight the feasibility of an high reliability and low delay solution.

2. BACKGROUND AND RELATED WORK

Hereafter, we first provide the necessary background about TSCH and 6TiSCH. We then review the existing contributions that provide high reliability with deterministic performance for Low-Power Lossy Networks (LLNs).

2.1 TSCH Overview

As previously mentioned, TSCH is among the MAC protocols defined in the IEEE 802.15.4-2015 standard [4] that provide high-level reliability and low-power operation in LLNs.

In TSCH networks, a schedule is computed and distributed among the nodes, therefore nodes must remain time synchronized throughout the network deployment. Moreover, under TSCH operation, time is divided into timeslots of equal length, large enough to transmit a frame and to receive an acknowledgement. A set of timeslots constructs a slotframe. At each timeslot, a node may transmit or receive a frame, or it may turn its radio OFF for saving energy. Finally, each timeslot is labelled with Absolute Sequence Number (ASN), a variable which counts the number of timeslots since the network was established. Thus, based on ASN and its scheduler, each node decides when to transmit or receive a frame.

Furthermore, to defeat noise and interference, and consequently to enable high reliability, TSCH proposes to implement a channel hopping scheme. To each transmission opportunity is attached a *channel offset*. In TSCH, a *cell* is a transmission opportunity described by a pair of timeslot and channel offset. Thus, at the beginning of each timeslot, the selection of the actual channel is derived from the channel offset and the ASN.

Finally, in TSCH, a cell may be either shared or dedicated, see Fig. 1. In the former mode, several interfering nodes are authorized to transmit: they must execute a slotted CSMA-CA mechanism to avoid collisions. In the latter case, a collection of non interfering transmitters are the owners of the cell: they transmit in contention-free mode.

2.2 6TiSCH Overview

6TiSCH IETF WG aims at defining protocols to bind IPv6 (i.e., 6LoWPAN) and reservation based MAC layer (i.e., TSCH). In 6TiSCH minimal [8], one shared cell is reserved at the beginning of the slotframe to exchange control packets (cf. Fig. 1). For instance, Enhanced Beacons (EBs) are transmitted during the shared timeslot so that the neighbors may associate with the existing network, while the rest of the slotframe comprises dedicated cells.

Furthermore, 6TiSCH makes a distinction between the protocol which defines how to negotiate the cells (i.e. 6P [9]) and the algorithm deciding how much cells to allocate in the schedule (the Scheduling Function such as SF0 [10]). The solution is very flexible since any scheduling algorithm may be practically implemented: a new Scheduling Function has just to be defined and interfaced with 6P. Thus, 6P may work either in a centralized manner (e.g. a node asks a Path Computation Element for new cells to use) or in a distributed manner (e.g. SF0 decides how many cells to allocate based on the local measures).

6TiSCH introduces the concept of *track* [11]. A track corresponds to dedicated radio resource, along with a multihop path. More precisely, a set of cells (a *bundle*) is reserved for each hop. By selecting different cells for different data flow, 6TiSCH may provide traffic isolation. A track forwarding scheme is in this case applied: when 6P receives a frame to forward, it automatically finds the outgoing bundle associated with the incoming cells.

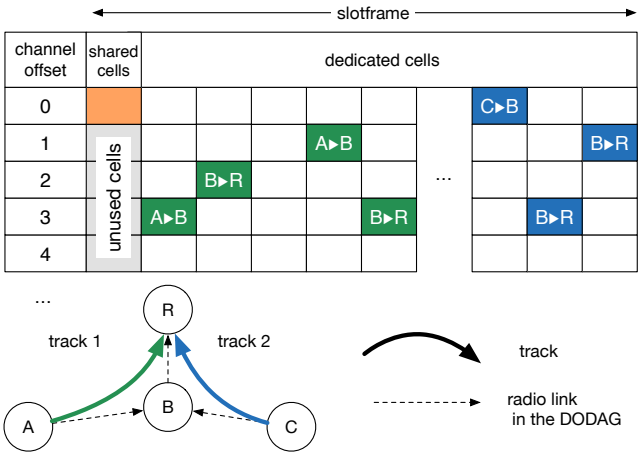


Figure 1: Schedule in a 6TiSCH network, using two different tracks for traffic isolation.

Furthermore, 6TiSCH introduces the concept of *chunk*: each node is able to separate the scheduling matrix in non overlapping chunks[12]. Then, the churns are allocated (in a centralized or distributed manner) to different nodes, so that each of them can pick in its churn when it has to allocate new cells.

In Fig. 1, the flow from A to the border router R, via node B, will be assigned to track 1. Besides, the same node (e.g., B) may forward an additional flow, for instance from C, using a different track (i.e., 2).

2.3 Scheduling

6TiSCH may rely either on an centralized scheduler (i.e., the Path Computation Element (PCE)) or on a distributed algorithm.

2.3.1 *Centralized Scheduling*

Several scheduling algorithms can be used by 6P. In [13], the tradeoff between a centralized and a distributed scheduling is studied. By adopting a queue theory approach, the authors showed that a centralized one is more efficient.

Ghosh *et al.* [14] proposed to minimize the schedule length in a multichannel TDMA environment. Similarly, TASA proposed to construct a centralized scheduling for a multihop TSCH network achieving the same objective [15].

Yigit *et al.* [16] studied the impact of routing on scheduling. The nodes have to use a larger number of timeslots when the radio links are unreliable. Schedex [17] proposed to schedule additional timeslots until an end-to-end delay constraint is not anymore respected.

All these centralized approaches assume the interference can be estimated accurately. They are particularly well suited for industrial networks in controlled environments.

2.3.2 *Distributed Solutions*

To not rely on a centralized controller, certain distributed

solutions have been proposed. DeTAS presented a decentralized version of TASA [18], assembling micro-schedules. Phung *et al.* [19] proposed to use a Reinforcement Learning based scheduling algorithm to cope with a variable traffic. However, the authors do not propose dedicated cells: the nodes have always to execute a CSMA-CA phase before transmitting their packets. In the same way, Z-MAC [20] proposes a mix between CSMA and TDMA, where timeslots are assigned distributively, solving iteratively the collisions.

Orchestra was proposed recently [7]. As exposed previously, Orchestra uses dedicated cells selected according to a pseudo-random sequence. However, it does not implement tracks (i.e., traffic isolation) and, thus, may suffer from the funneling effect [21], since each nodes reserves the same number of timeslots with its parent.

SF0 [10] proposed a Scheduling Function to compute the number of cells to allocate/remove. Palattella proposed also to use an hysteresis function to decide when new cells have to be added or removed [22]. While it presents a promising approach, they did not present experimental evaluation, and moreover, they didn't describe the integration of tracks to provide traffic isolation (the bandwidth estimation algorithm is let to a future work).

3. A FULLY DISTRIBUTED VERSION OF 6TISCH

Hereafter, we will present our distributed 6TiSCH architecture where all decisions are taken locally, and guaranteeing traffic isolation: different applications may be assigned to different tracks, and use consequently different cells.

- the shared cells are only used to transmit 6P packets (unicast) and Enhanced Beacons and DIO (broadcast);
- the other unicast packets such as DAO or data packets use dedicated cells, reserved for a given track.

In other words, data transmissions use the deterministic part of TSCH, while random access is only reserved for 6P negotiations and beacons.

3.1 Track Management

6TiSCH track consists in one track owner (64 bits) and one track ID (16 bits). We propose here a way to exploit these tracks to enable the following features:

Isolated track: a source node has to reserve one end-to-end track for its flow. To guarantee traffic isolation, the track must be only used by one data flow. Thus, the track owner is its own MAC address (64 bits) and the track ID (16 bits) corresponds to the application in execution on the node. The track ID may be assigned uniquely before the deployment, following the same approach as the well known ports. Thus, a network may support at most 65,535 different applications.

Convergent track (shared track, e.g., alarms): because data transmissions are in this case unfrequent, reserving one cell during each slotframe may waste bandwidth. However, we must guarantee small end-to-end delays for alarms, and the maximum time between two cells should be minimized. In particular, if an event is detected by several nodes, the delay of the first exemplary received by the border router is the most important one.

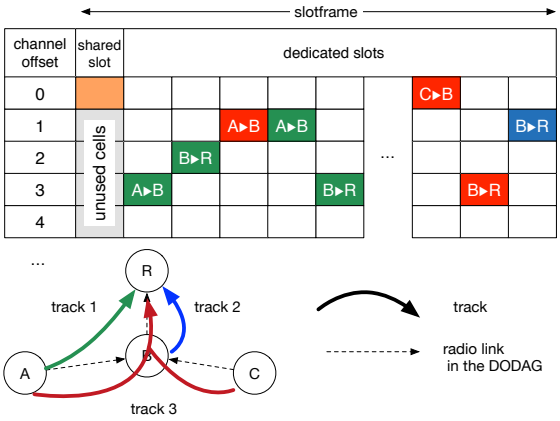


Figure 2: Mixing heterogeneous applications in the same 6TiSCH instance.

> We propose for these applications to reserve a common convergecast track, shared among all the sources of the same application. Thus, the track owner is the address of the destination, and the track ID is specific to the application. All the other nodes which generate data packets to this destination will, by definition, use the same track. It is important to note that the packets from different sources do not collide: they use dedicated cells. However, the packets after the first one have to be buffered, increasing the delay, and the probability of a buffer overflow if a large amount of traffic is generated at the same time.
>
> When packets are routed via the border router, the track owner is extracted from the RPL DAG Information Object (DIO).

Furthermore, we may mix different applications in the same network. In Fig. 2, the smart metering application uses isolated tracks (blue and green for two applications). Bandwidth reserved for the the blue track cannot be used by the *green* flow. On the contrary, the application on A and C use a convergent track (in red). Let's imagine that both A and C generate one packet during the same slotframe. B has only one outgoing cell to forward the two packets: the second one has to be forwarded in the next slotframe.

The convergent track strategy represents well the behavior of classical wireless sensor networks: each source may generate an indefinite number of packets, and the relay nodes expects to forward them in a best effort manner. However, different applications use different tracks, and therefore orthogonal radio resources. In other words, we multiplex only the transmissions from the same application (and different source nodes).

Note that we do not implement a GMPLS-like approach to manage the tracks, as advocated by [11]. We insert rather information in the 6P transaction to specify which track is associated with the cells to insert in the schedule. For this purpose, our Scheduling Function may use the `metadata` field of the 6P `Add Request` message. Later, when a node receives a data frame, it has to extract the track associated with the incoming cell, and tag the frame accordingly in the queue. We also implemented a FIFO discipline: the node picks up the first packet from the queue with the trackID corresponding to the outgoing dedicated cell.

3.2 Bandwidth Requirement Estimation with SF

We propose a simple strategy to allocate the bandwidth on-demand. The Scheduling Function SF0 [10] collects the bandwidth requirements from neighbor nodes. Note that it does not detail how per track requirements are computed. We here propose a Scheduling Function SF_{loc} which takes decisions considering only local information. Besides, SF_{loc} allocates different cells for different tracks. In this way, we are able to compute a schedule in a distributed manner while maintaining traffic isolation: two applications using different tracks will be managed independently by SF_{loc}. Besides, we do not require each application defines a priori its requirements: SF_{loc} is self-adaptive and learns from the forwarded packets.

In our explanations, we focus on the convergecast traffic pattern. A node has consequently to negotiate with its RPL preferred parent a list of cells to use for each of its tracks. A node, which has to decide how many cells to reserve, considers only its own buffer and its schedule utilization.

3.2.1 Cell insertion

We have to decide when a new cell has to be negotiated with a neighbor. We follow a similar approach to SF0, while only using the amount of forwarded traffic to compute the bandwidth requirements:

1. when a packet is inserted in the queue, SF_{loc} is triggered;
2. if the node already has an on-going 6P transaction, SF_{loc} stops: only one track may be modified at the same time;
3. for each track tr, the node computes:
   (a) the list of outgoing cells $\mathcal{C}_{out}$ toward the parent and the average number of transmissions required before receiving an `acknowledgment` ($\forall k \in \mathcal{C}_{out}, ETX(k)$);
   (b) the number of packets present in the queue for this track $N_{pk}(tr)$
   (c) if the following condition does not hold:

$$\sum_{k \in \mathcal{C}_{out}} \frac{1}{ETX(k)} \geq N_{pk}(tr) \quad (1)$$

   SF_{loc} asks 6P to reserve the following number of cells:

$$N_{pk}(t) - \sum_{k \in \mathcal{C}_{out}} \frac{1}{ETX(k)} \quad (2)$$

   (upper bounded by 3 since an 6P Information Element may contain at most 3 timeslots).

Our approach is robust to interference: a colliding cell will present a larger Expected Transmission Count (ETX), and will be detected quickly, using the same mechanism as [23] (*tx-housekeeping* feature). A bootstrapping node will reserve quickly new cells for the given track: this over-provisioning efficiently deals with lossy links. The colliding cells will be detected and removed later.

We plan in the future to implement a more sophisticated mechanism. Recently, Domingo-Prieto *et al.* [24] proposed to avoid oscillations by considering the dynamics of the bandwidth requirement. The solution was evaluated by simulations and seems promising.

We will detail in the next subsection how 6P selects the actual cells (timeslot and channel offset) to use.

3.2.2 Cell removal

The routing topology might changes since RPL is dynamic. In the same way, a node may be too aggressive and may have reserved too much bandwidth, when e.g., a new parent is selected and it has to empty its buffer. A node has to later remove the unused cells.

To select the cells to remove, we consider a given cell becomes *useless* if no `ack` is received during a long time. This cell is considered useless even if unicast packets are transmitted but not acknowledged. In this way, we remove both the colliding and the unused cells. Indeed, no contention is implemented for dedicated cells: the data packets will be deterministically lost if a pair of interfering radio links uses the same cell.

To this end, we propose a simple timeout based approach: when a cell is not used during at least $Timeout_{SF_{loc}-tx}$ seconds, the owner of the cell (the transmitter) asks the receiver to release this cell. More precisely, 6P sends a `link-removal` request, which specifies the timeslot and the channel offset to remove from the schedule.

Possibly, the radio link may not be anymore usable. Thus, the transmitter removes silently a cell if the 6P `link-removal` failed after all the possible retransmissions.

Inversely, a receiver removes silently a cell if it is unused during $Timeout_{SF_{loc}-rx}$. To avoid inconsistent decisions, the following condition must hold:

$$Timeout_{SF_{loc}-tx} << Timeout_{SF_{loc}-rx} \quad (3)$$

In particular, it must take into account the time to transmit the `link-removal` request and to receive the associated `ACK`.

3.3 Distributed Scheduling With 6P

To make 6TiSCH entirely distributed, a pair of nodes must be able to decide which cell should be used to exchange packets, without the help of a centralized PCE.

We present here two different approaches to either reduce the number of collisions (random) or to reduce the end-to-end delay (contiguous).

3.3.1 Random Scheduling

When the nodes select autonomously the cells to use with their parent, they do not have a vision of the cells already reserved by an interfering transmitter. To reduce the probability of collision, we propose to select randomly the cells to use: we minimize the probability to have colliding cells in the global schedule. As it is shown in performance evaluation Section, this simple and greedy approach is efficient and robust.

When a node has to reserve k new cells, 6P must ask its preferred parent to reserve some additional bandwidth for a given track, using a 2-step 6P transaction:

1. the transmitter N_t selects j timeslots ($j \geq k$), selected randomly among the available timeslots in its schedule. It selects one random channel offset for each of these timeslots;
2. N_t constructs a `link-request` including these cells and transmits the packet during a shared cell;

3. the receiver N_r selects the first k cells present in the `link-request` and available in its schedule. It enqueues a `link-reply`, which will be transmitted during one shared slot. If there is not enough available cells ($< k$), N_r refuses the transaction and sends an empty negative `link-reply`. Else, the reply includes the list of the selected cells.

4. N_t receives the `link-reply` and updates its schedule with the cells allocated by its parent if the reply is positive.

Selecting randomly the cells to use minimizes the probability of collision, but tends to increase the end-to-end delay. Intuitively, the outgoing and incoming cells being chosen randomly, the end-to-end delay equals on average:

$$delay = \frac{hops * SF_{length} * T_{slot}}{2} \quad (4)$$

With hops being the number of hops in the route, SF_{length} the slotframe length, and T_{slot} the timeslot duration (by default 15ms). Besides, a node, which forwards many flows, will have to buffer many packets since the outgoing and incoming cells may be far away. Thus, some packets may be dropped because of buffer overflows. However, as we highlighted in our experiments this effect is limited.

3.3.2 Contiguous Scheduling

In this second approach, we focus on minimizing the end-to-end delay. To this aim, we propose to minimize the buffering delay:

1. With SF_{loc}, the transmitter N_t computes the number of cells ($= k$) to reserve toward its parent for the track tr. It identifies in its schedule the last slot number $T_{rx\_last}(tr)$ which is used to receive a packet for the track tr.

   - If no incoming cell exists (i.e., N_t generated the packet), $T_{rx\_last}(tr)$ is chosen randomly;

2. N_t selects the first available cells after $T_{rx\_last}(tr)$ and chooses randomly one channel offset for each of them. It constructs a `link-request` to piggyback this list;

3. The receiver N_r selects the first k cells from the `link-request` also available in its schedule. The following cases may occur:

   (a) at least k cells are available: N_r selects the k first of them, and constructs the associated `link-reply` as usually;

   (b) not enough cells are available:

      i. N_r constructs a `link-reply` and piggybacks an Information Element which lists the *busy* cells.

      ii. N_t receives this `link-reply` and will insert in its own schedule the blacklisted cells, linked with the parent. When N_t will send another `link-request` to N_r, theses busy cells cannot be selected.

4. PERFORMANCE EVALUATION

4.1 Experimental Setup

Our experimental campaign was conducted over the FIT IoT-LAB platform in Grenoble [25]. This federated platform provides access to a large collection of motes. We employed the m3 nodes, based on a STM32 (ARM Cortex M3) microcontroller (ST2M32F103REY). It embeds an AT86RF231 radio chipset, providing thus, an IEEE 802.15.4 compliant PHY layer.

4.1.1 Topology

FIT IoT-LAB allows for several experiments to be executed concurrently. Thus, we evaluate the behavior of 6TiSCH under real-world conditions, with potential external interference. The nodes are placed in corridors, in false ceilings and floors, to mimic smart buildings scenarios. To construct multihop topologies, we select nodes sufficiently far from each other: we alternate between ceils and floors, and two consecutive nodes are separated by approximatively 1.5meters. We selected the nodes in the corridor located in the middle (cf. Fig. 3).

4.1.2 Default parameters

Furthermore, we employed the OpenWSN*, that provides an open-source implementation of the 6TiSCH stack (i.e., IEEE802.15.4-2015 TSCH, 6P, SF, 6LoWPAN and RPL). We implemented in OpenWSN our proposal to handle tracks, schedule appropriately the cells, etc. Our modifications are freely available on GitHub†.

We use the default parameters value as depicted in Table 1. All nodes take local decisions, based on the information exchanged with their neighbors (EBs, 6P requests and replies). Each mote sends its packets to the border router according to a CBR traffic, see Table 1. To dissociate the impact of traffic and of the network size, we maintain the number of packets generated constant in the network, whatever the number of nodes. Thus, each node generates by default one data packet every $10 * nbNodes$ seconds.

4.1.3 Queue management

Because of tracks, the buffer does not follow a FIFO approach: data packets assigned to a track are buffered until the node enters in a dedicated cell for this particular track. Due to memory constraints, the data buffer is limited, and thus, it may be filled quickly. To guarantee sufficient space for the control packets in the buffer, we reserve by defaults 4 slots in a buffer of 10 frames (cf. Table 1). To avoid privileging old packets in the queues, we implemented a timeout based mechanism, where each data packet is tagged at reception and is dropped after 8 seconds (≈6 slotframes) if it is still present in the queue.

*openwsn - open-source implementations of protocol stacks based on Internet of Things standards, https://openwsn.atlassian.net/

†https://github.com/ftheoleyre/openwsn-fw/ and https://github.com/ftheoleyre/openwsn-sw/

†In this way, we have a constant number of packets generated in the whole network, even when increasing the number of nodes.

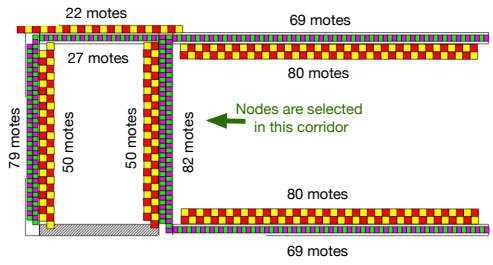
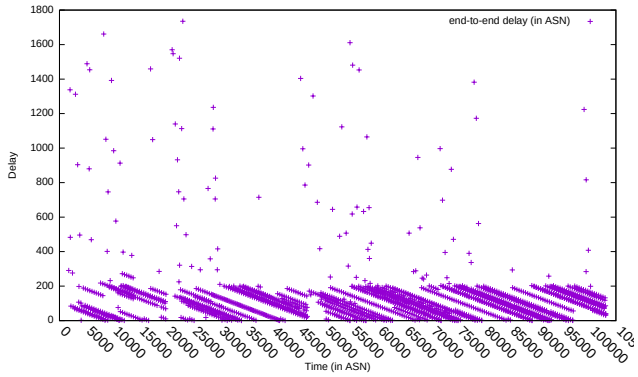


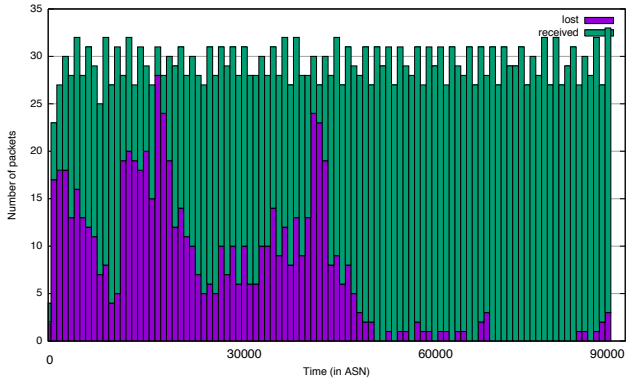
Figure 3: FIT IoT-LAB: topology in the Grenoble testbed.

Table 1: Default parameter values
(the inter-packet time depends on the number of nodes to maintain a constant load with a variable number of nodes)

Experiments	Duration	1.5 *hours*
	Topology	line (floors and ceils)
	Separation	$\approx$ 1.5 *meters*
TSCH	Slotframe length	101
	Nb. shared cells (N_{shared})	5
	Timeslot duration	15 *ms*
RPL	DAO period	50 *s*
	DIO period	8.5 *s*
CoAP	CBR	10 *s* * nbNodes
6TiSCH	$Timeout_{SF_{loc}-rx}$	25 *s*
	$Timeout_{SF_{loc}-tx}$	20 *s*
Queues	Timeouts	8 *s*
	Queue size	10 packets
	incl. data packets	at most 6 packets



(a) End-to-end delays for each received packets



(b) Packet losses versus received

Figure 4: 6TiSCH behavior during the convergence - topology with 12 nodes and one sink (1.5 hours).

4.2 Convergence in a simple topology

We first study the network behavior during the convergence, on a simple 2 hops topology, by selecting an appropriate set of nodes in the same corridor of the Grenoble's IoT Lab topology (cf. Fig. 3). The DODAG is constructed by RPL in a distributed manner.

We first measure the delay performance (see Fig. 4a). As it can be observed, most of the packets are delivered very quickly to the border router, even for nodes several hops far from the sink. A few packets need more time to be delivered: they correspond to retransmissions, since radio links may be unreliable. Finally, only a few packets are delivered in more than 101 slots (i.e., the slotframe length, $1,010$ *ms*).

Considering the packet losses distribution (see Fig. 4b), we can remark that most of the packets are lost at the beginning, during the convergence time. Indeed, each node has to request bandwidth to its parent, negotiating which timeslots and channel offsets should be used for the transmission. Meanwhile, the packets are buffered, and are dropped when they are stored in the buffer for a long duration (i.e., 8 seconds). However, after the convergence period, the reliability significantly increases, only a few packets are dropped, mainly due to the external interference (i.e., other experiments are executed simultaneously on the same testbed).

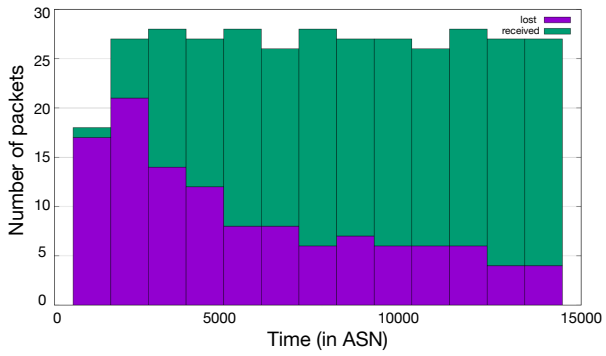
It is worth mentioning that medium access is deterministic after the convergence: the schedule does not change anymore, and the same cells are used for a given track (i.e., application).

4.3 Contiguous versus Uniformly Distributed Shared Cells

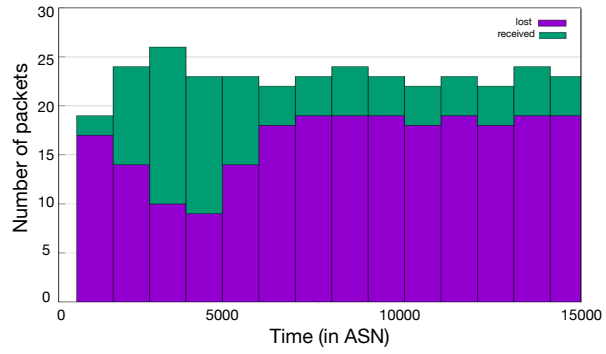
In 6TiSCH, a node may use a shared cell to transmit a frame under contention. Typically, these shared cells are used for EBs, broadcast packets (e.g., DIO), as well as to reserve bandwidth (`Link-Request`, and `Link-Replies`). By default, the N_{shared} first cells are shared in the slotframe (i.e., the *contiguous* scenario).

We here propose rather to distribute uniformly the shared cells. Indeed, IEEE802.15.4-2015-TSCH does not use a backoff at the beginning of a shared cell. It rather implements a CSMA-CA slotted approach *among* the shared cells: the backoff counts the number of shared cells to wait before transmitting a packet. Thus, collisions are very frequent during the shared cells.

We compare the *contiguous* and the *distributed* strategies during the convergence (Fig. 5). We measure the packet losses in a topology with 10 nodes and one sink. With contiguous cells, many collisions occur in the shared cells: `links requests` tend to be lost, retransmitted, and a node has to wait for a long time before receiving a dedicated cell to use. On the contrary, having uniformly distributed cells in the slotframe tends to reduce the collisions and to accelerate the convergence. Thus, we should either use very short slotframes with one single shared cell, or a long slotframe with uniformly distributed shared cells.



(a) Uniformly distributed cells



(b) Contiguous cells

Figure 5: Packet losses / received during an experiment of 30 minutes with 10 nodes and one sink.

4.4 RPL metric

RPL may use any objective function to compute the DAG rank of the nodes. However, minimizing the number of hops tends to perform poorly [26]. Thus, the ETX metric is often privileged: it consists in estimating the number of transmissions before receiving an acknowledgement.

To highlight the impact of the routing metric, we evaluate the behavior of the 6TiSCH stack with the following routing metrics:

Minhops: we minimize here the number of hops: the link metric is set to `minHopRankIncrease` for all the links;

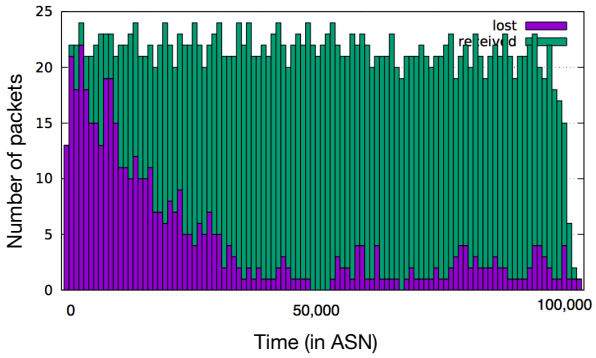
ETX: we use ETX, reported by the MAC layer: ratio of the number of acknowledged packets and the number of transmitted packets for a given neighbor;

RSSI: all links with a RSSI superior than a threshold value have the best link quality (=1). Then, the link metric is incremented for each dBm less.

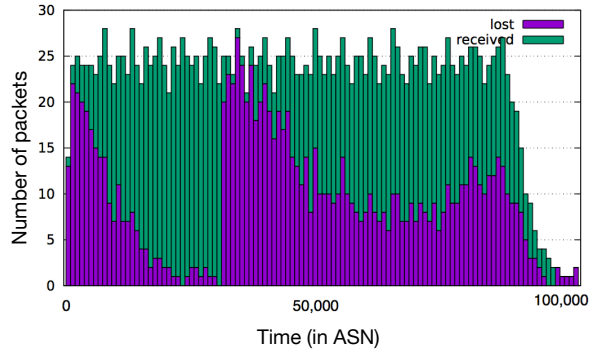
To compute the DAG rank, a node sums the rank of its parent, and the link cost (i.e. the OF0 objective function in RPL).

Besides, we maintain the default blacklisting method of OpenWSN: a radio link is marked as stable when its RSSI exceeds -80 dBm for 3 consecutive different packets. Inversely, links become unstable when its RSSI is inferior than -80 dBm for 3 consecutive packets.

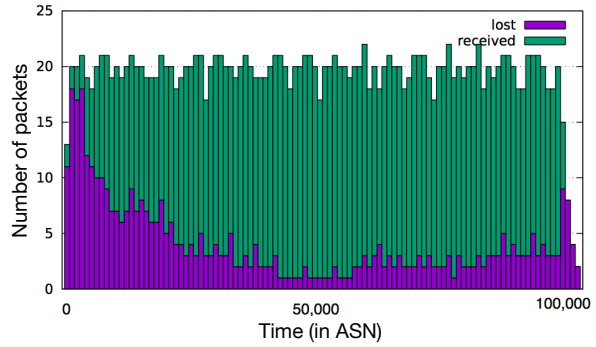
Figure 6 presents the impact of the routing metric on the reliability. ETX converges slowly: a significant part of the packets is lost during the beginning (first hour). On the contrary, the packet losses decrease more quickly with `minhop`.



(a) ETX



(b) MinHop



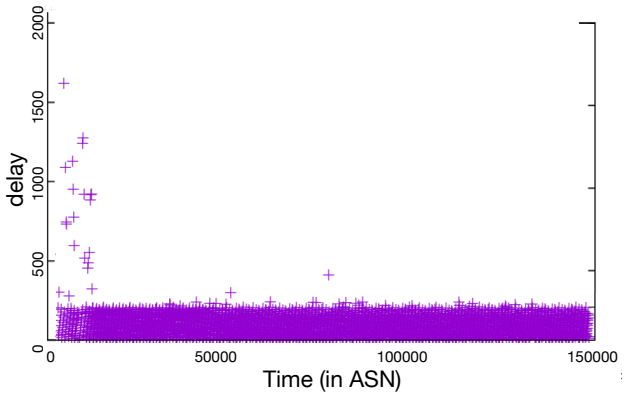
(c) RSSI

Figure 6: Packet losses / received during an experiment of 1.5 hours with 20 nodes and one sink.

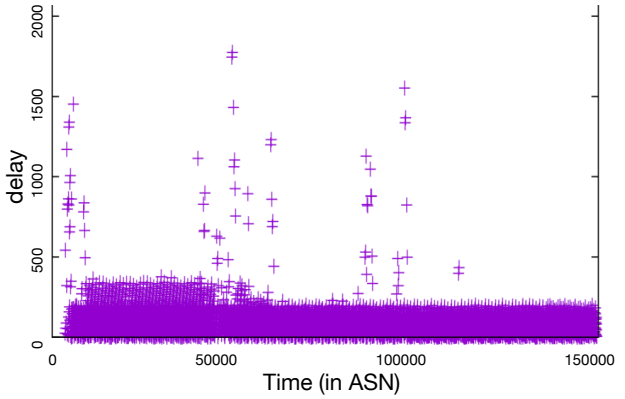
Finally, using the RSSI metric constitutes a tradeoff between minhop and ETX.

After 20 minutes, the packet losses increase drastically with `minhop`. A link performs badly and the RPL DAG structure is reconfigured: minHop tends to select unreliable long links. Thus, a large ratio of the nodes have to change their preferred parent, and a storm of requests use the shared cells. The network re-converges very slowly to a legal state. On the contrary, RSSI and ETX perform better: the performance is more stable after 1.5 hours.

In conclusion, ETX needs more time to converge because it represents an instantaneous metric and is subject to large variations, caused by the radio instability and the dependency on the traffic (i.e., more packets on a route impacts negatively the link quality) [27]. However, 6TiSCH achieves to exhibit stable performance after the network convergence.



(a) Contiguous scheduling



(b) Random scheduling

Figure 7: Packet delays during an experiment of 1.5hours with 5 nodes and one sink

4.5 Scheduling algorithm

We finally evaluated the performance of the scheduling algorithms (Fig. 7), comparing our *random* strategy and the *random contiguous* strategy.

The random contiguous strategies selects the outgoing cells to minimize the buffering delay. Mechanically, the *random contiguous* strategy achieves a smaller end to end delay than the *random* strategy.

Besides, we can remark a reconfiguration with the random strategy: after 30 minutes, the DODAG is reconfigured, leading to better performance. However, the end-to-end delay remains superior to that of the contiguous scheduling strategy.

Due to lack of space, we did not represent here the performance for larger topologies. While the random contiguous strategy keeps on reducing the buffering delay, the convergence time is longer: more `link-requests` have to be transmitted before finding an available cell. Besides, shared cells suffer from many collisions when we have more than 20 nodes. Thus, the packet delivery ratio is lower with the random contiguous strategy during the convergence time (30% versus 50% after 30 minutes). In conclusion, 6P has to be improved to enable more reliable transactions.

5. LESSONS

5.1 Guidelines

Based on our experimental results, we conclude the following guidelines should be applied in 6TiSCH:

Non contiguous shared cells: when shared cells are all placed at the beginning of the slotframe, all the requests are buffered during a whole slotframe before being transmitted in the first shared cells. The collisions are consequently very frequent.

On the contrary, we propose to distribute uniformly the shared cells in the slotframe, decreasing thus the pressure, and making 6TiSCH much more efficient.

Routing metric: a blacklisting approach is surprisingly efficient, while the minhop metric converges quickly to a legal schedule. However, it selects the *worst* radio links, which may dysfunction: these local faults impact globally the network, penalizing the convergence.

On the contrary, both the ETX and RSSI metrics perform better. However, some heuristics should be proposed to accelerate the convergence.

5.2 Open Challenges

While our distributed version of 6TiSCH works efficiently in many situations, we highlighted several challenges that still have to be addressed:

Radio link quality estimation: a node has currently to select a parent without exchanging packets (i.e., no dedicated cells has been reserved so far with this possible parent).

However, RSSI or the packet losses of EBs may be misleading to estimate the link quality [28]. Worse, the link might be asymmetrical: while EBs are received correctly, a `link-request` may never be received correctly by this parent. This process wastes bandwidth and energy.

We should propose a mechanism to estimate the link quality of a possible parent, while not being already attached to this node;

Dense topologies: the radio link quality estimation is even exacerbated with dense topologies: which neighbor should be selected as parent? While the path metric (its DAG rank) is important, the link quality is also of prime importance. We have to propose schemes, that discover an *acceptable* parent, with a tradeoff between discovery cost and optimality.

Centralized versus Distributed: we expect in the future work to compare quantitatively the performance of a centralized and a distributed scheduling solution. However, we must be able to report to the PCE the link quality of all the neighbors and to estimate the level of interference: this constitutes a challenging objective. Besides, the shared cells are subject to many collisions: how could we increase the end-to-end reliability of the communication to the PCE?

6. CONCLUSION AND PERSPECTIVES

In this paper, we presented a distributed solution of 6TiSCH: a node decides autonomously when and which cells to use with its parent. Moreover, we proposed to exploit *isolated tracks* to provide flow isolation, and to make the transmissions reliable and independent: each application has dedicated (i.e., reserved) bandwidth for its packets transmissions. Inversely, we are also able to mutualize the bandwidth reservation, using a *convergent track*: all the DAO in our experiments use the same convergent track toward the border router. This flexibility of the 6TiSCH architecture and the stability of the performance prove the relevance of a deterministic network for the low power lossy networks.

In a future work, we plan to explore the impact of very large topologies, and to study in depth the maximum distance and traffic TSCH may support efficiently. We also plan to study how we could accelerate the convergence time: we should first reduce the number of collisions among the 6P requests and replies, which increase the delay to negotiate a set of cells with its parents. Furthermore, we should explore which routing metric should be used to avoid re-changing the parent when the radio link quality is initially misestimated.

7. REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[2] Yi Zhi Zhao et al. A Survey and Projection on Medium Access Control Protocols for Wireless Sensor Networks. *ACM Computer Surveys*, 45(1):7:1–7:37, 2012.

[3] G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois. Service level agreements for wireless sensor networks: A wsn operator's point of view. In *Network Operations and Management Symposium (NOMS)*. IEEE, 2014.

[4] IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, April 2016.

[5] Thomas Watteyne, Ankur Mehta, and Kris Pister. Reliability through frequency diversity: Why channel hopping makes sense. In *PE-WASUN*. ACM, 2009.

[6] IPv6 over the TSCH mode of IEEE 802.15.4e. https://datatracker.ietf.org/wg/6tisch.

[7] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *SenSys*. ACM, 2015.

[8] X. Vilajosana, et al. Minimal 6TiSCH Configuration. draft-vilajosana-6tisch-minimal-16, June 2016.

[9] Q. Wang and X. Vilajosana. 6top protocol (6p). draft-ietf-6tisch-6top-protocol-02, July 2016.

[10] D. Dujovne, LA. Grieco, MR. Palattella, and N. Accettura. 6TiSCH 6top Scheduling Function Zero (SF0). draft-dujovne-6tisch-6top-sf0-00, March 2016.

[11] P. Thubert. An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4. draft-ietf-6tisch-architecture-10, June 2015.

[12] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert. 6tisch: deterministic ip-enabled industrial internet (of things). *IEEE Communications Magazine*, 52(12):36–41, December 2014.

[13] John N. Tsitsiklis and Kuang Xu. On the power of (even a little) centralization in distributed processing. In *ACM SIGMETRICS*, pages 161–172, 2011.

[14] A. Ghosh, O.D. Incel, V.S.A. Kumar, and B. Krishnamachari. Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks. In *IEEE MASS*, pages 363–372, 2009.

[15] M.R. Palattella, et al. On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH. *Sensors Journal, IEEE*, 13(10):3655–3666, 2013.

[16] Melike Yigit, Ozlem Durmaz Incel, and Vehbi Cagri Gungor. On the interdependency between multi-channel scheduling and tree-based routing for WSNs in smart grid environments. *Computer Networks*, 65(0):1 – 20, 2014.

[17] Felix Dobslaw, Tingting Zhang, and Mikael Gidlund. End-to-End Reliability-aware Scheduling for Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2014.

[18] N. Accettura, M.R. Palattella, G. Boggia, L.A. Grieco, and M. Dohler. Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things. In *WoWMoM*. IEEE, 2013.

[19] Kieu-Ha Phung et al. Schedule-based multi-channel communication in wireless sensor networks: A complete design and performance evaluation. *Ad Hoc Networks*, 26(0):88 – 102, 2015.

[20] Injong Rhee, A. Warrier, M. Aia, Jeongki Min, and M.L. Sichitiu. Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 16(3):511–524, 2008.

[21] Chieh-Yih Wan, Shane B. Eisenman, Andrew T. Campbell, and Jon Crowcroft. Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In *SenSys*. ACM, 2005.

[22] Maria Rita Palattella et al. On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks. *IEEE Sensors Journal*, 16(2):550–560, November 2015.

[23] K. Muraoka, T. Watteyne, N. Accettura, X. Vilajosana, and K. S. J. Pister. Simple distributed scheduling with collision detection in tsch networks. *IEEE Sensors Journal*, 16(15):5848–5849, Aug 2016.

[24] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne. Distributed pid-based scheduling for 6tisch networks. *IEEE Communications Letters*, 20(5):1006–1009, May 2016.

[25] Georgios Z. Papadopoulos, et al. Adding value to WSN simulation using the IoT-LAB experimental platform. In *WiMob*. IEEE, 2013.

[26] Lai Dhananjay, et al. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *GLOBECOM*. IEEE, 2003.

[27] O. Iova, F. Theoleyre, and T. Noel. Stability and efficiency of RPL under realistic conditions in Wireless Sensor Networks. In *PIMRC*. IEEE, 2013.

[28] Nouha Baccour et al. Radio Link Quality Estimation in Wireless Sensor Networks: A Survey. *ACM Transactions Sensor Networks*, 8(4):34:1–34:33, 2012.