Collision Prevention in Distributed 6TiSCH Networks

Ali J. Fahs*‡§, Rodolphe Bertolini*‡§, Olivier Alphand†‡, Franck Rousseau†‡, Karine Altisen†§, Stéphane Devismes*§
Université de Grenoble Alpes, CNRS
**UGA, †Grenoble INP*
‡LIG, §VERIMAG
F-38000 Grenoble, France
firstname.lastname@imag.fr

***Abstract*—The IEEE802.15.4e standard for low power wireless sensor networks defines a new mode called Time Slotted Channel Hopping (TSCH) as Medium Access Control (MAC). TSCH allows highly efficient deterministic time-frequency schedules that are built and maintained by the 6TiSCH operation sublayer (6top). In this paper, we propose a solution to limit the allocation of identical cells to co-located pair of nodes by the distributed TSCH scheduling algorithms. It consists of making nodes able to overhear past cell negotiations exchanged in shared cells by their neighbors and prevent the nodes from reusing already assigned cells in future allocations. Our mechanism has been tested through simulations that show significant improvements to random scheduling algorithms supplemented by our approach.**

***Keywords*-IOT; IEEE802.15.4e; TSCH; 6TiSCH; 6top; scheduling function;**

I. Introduction

Network Technologies have been shifted from a limited number of interconnected expensive computers with high performance processing units to networks made of a huge number of cheap entities with limited processing capabilities, called *things*, leading to the idea of *Internet of Things (IoT)*.

Among the many standards dedicated to IoT, IEEE802.15.4 [1] aims at offering low power short range communications at a low cost. However its MAC layer includes several limitations (no channel diversity, unbounded delay and poor multi-hop topology support). To overcome those limitations, IEEE802.15.4e [2] extended IEEE802.15.4 in 2012 by proposing a new mode called *TSCH (Time-Slotted Channel Hopping)*. TSCH defines a periodical slotframe consisting of fixed-size *cells*, each indexed by a specific timeslot offset and a channel offset. Each cell has the same duration and its length should be large enough to accommodate a maximum frame length and its acknowledgment, while ensuring at the same time synchronization between communicating nodes. TSCH also provides channel diversity, enabling parallel communications on different channels within the same timeslot, as well as channel hopping in the same cell from one slotframe from another. The channel of a cell is deduced from the channel offset and the absolute slot number. There are two types of TSCH cells: *shared cells* where multiple nodes may either listen or transmit; and *dedicated cells* for contention-free link-local communication between two neighboring nodes.

Since we are dealing with millions of nodes, sensors, and things that should be connected to the Internet, large scaling capabilities are mandatory. With this goal in mind, the IETF has standardized an IoT stack based on 802.15.4. The integration of TSCH in this stack is currently being carried out by the 6TiSCH IETF Working Group through the 6TiSCH operation sublayer, 6top. It includes the definition of algorithms to schedule dedicated cells and a 6top protocol (6P) to operate cell negotiations between nodes [3].

In this paper we focus on improving those distributed scheduling algorithms, which are mainly designed to address the problem of convergecast traffic towards a unique sink in a multi-hop network. *RPL (Routing Protocol for Low-Power and Lossy Networks)* [4] organizes the network as a *DODAG (Destination-Oriented Directed Acyclic Graph)* topology along which the data generated by each node are relayed up to its root. Scheduling algorithms thus attempt to create efficient TSCH schedules along RPL routes.

Most of existing scheduling functions select cells randomly among the cells that are not located in timeslots already used by both negotiating nodes with their respective DODAG parents and children. However the cell selection process should also preclude dedicated cells already reserved by other co-located neighboring nodes that may interfere leading to collisions and consequently to packet dropping. Our solution makes distributed scheduling algorithms aware of dedicated cells already assigned in the neighborhood by overhearing past cell negotiations exchanged with 6P in shared cells.

To evaluate our solution, we compare the performances of the *On-the-Fly* [5] scheduling algorithm with and without our approach. Results show significant improvements, in particular a decrease of colliding cells[1] and colliding packets during the bootstrap phase.

The main contributions of our paper are:

- Overhearing of 6P transactions to collect dedicated

[1]A colliding cell is a dedicated cell reserved by multiple co-located pairs of nodes where packets might collide in case of simultaneous transmissions.

cells reserved by neighbors without introducing any signaling overhead.

- Addition to 6P transactions of a short-term memory of past reserved cells to make the overhearing more robust.
- Modification of the cell selection algorithm in 6top to avoid the cells used by neighboring nodes.
- Comparative evaluation of a random scheduling algorithm in a simulator with and without our approach.

The outline of this article is organized as follows: Section II gives an overview of 6TiSCH background and the existing distributed scheduling algorithms. Section III describes our solution with its different features. Section IV first details the changes done in the simulator as well as the simulations parameterization, then the results of simulations are provided and discussed. We make concluding remarks in Section V.

II. Background

A. 6TiSCH Operation Sublayer (6top)

The IETF puts a lot of efforts in the standardization of an IoT stack compliant with the IEEE802.15.4 PHY/MAC layer. However the original 802.15.4 MAC layer lacks some important features. Those gaps were filled by the TSCH mode of IEEE802.15.4e. To properly integrate TSCH within the IoT stack, the IETF 6TiSCH Work Group defines the 6TiSCH operation sublayer (6top). Figure 1 illustrates how 6top integrates in the IoT stack.

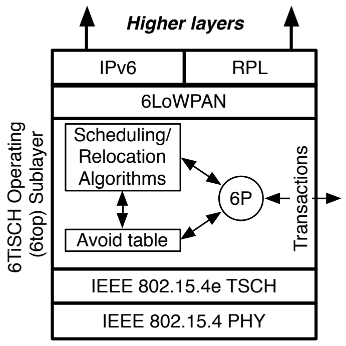

Figure 1. TSCH integration in the IETF IoT stack

6top includes the definition of policies to manage TSCH schedules and the 6top protocol (6P) to control the communication for cell reservation and deletion between nodes.

If we consider a Tx node (child) and Rx node (parent) that are communicating in the 6TiSCH network, 6P supports 3 types of operations that can be triggered by *Scheduling Functions* (SF):

1) If the *Scheduling Algorithm* in Tx determines that it needs more cells to communicate with Rx, it will issue a 6top transaction to reserve more cells into TSCH schedule.
2) If the *Scheduling Algorithm* in Tx determines that it needs less cells to communicate with Rx, it will issue a 6top transaction to release some TSCH cells.
3) The *Relocation Algorithm* in Tx and/or Rx detects whether a dedicated cell is facing collisions or not. If so, Tx or Rx replaces the defected cell through 6top transactions.

As explained before, each operation is encapsulated into a transaction. A 6top transaction consists of a negotiation between Tx and Rx, that updates the TSCH table. This transaction takes place in 2 or 3 steps. In a 6top transaction, the scheduling function decides whether to use a 2-step transaction, a 3-step transaction, or a mix between them. Furthermore, the scheduling function has two alternatives to communicate the list of cells:

1) Black listing: where the node sends all cells in use.
2) White listing: where the node sends a number of cells available for communication.

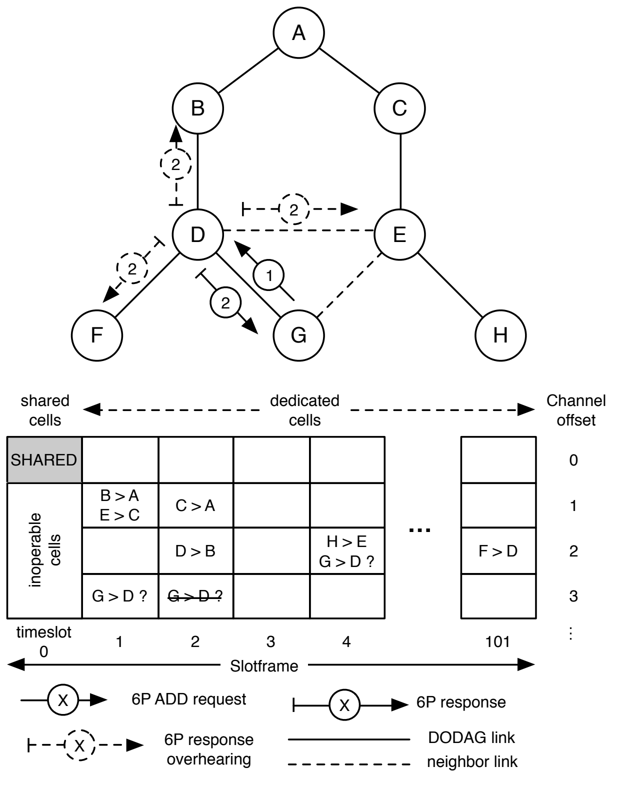

Figure 2. 6P Transactions and TSCH schedule update

For a better understanding, we detail in Figure 2 the sequence of a 2-step transaction using white-listing to allocate 2 cells for the link $G \rightarrow D$:

(1) The Scheduling Function (SF) of node G detects that two additional cells are needed and selects 3 candidate

available cells $\{(1,3),(2,3),(4,2)\}$[2] before triggering a 6P ADD request towards node D in the shared slot.

② The SF running on node D precludes the cell (2,2), selects 2 cells $\{(1,3),(4,2)\}$ and issues a 6P response to node G.

As we can see in this example, the cell selection is not done randomly among all the cells of the TSCH slotframe. The 6top negotiation natively precludes all the cells in timeslots where both negotiating nodes are communicating with their DODAG parents and children. The selection of candidate cell is then randomly done in the set of remaining cells. In the previous negotiation, D will not consider the timeslot 2 since it already uses it to communicate with its parent B. G has no dedicated cells so it does not remove any cells *a priori*.

In next section, we will describe TSCH scheduling distributed algorithms and, in particular, how they tackle the problem of reducing scheduled colliding cells.

B. Related Work

First we consider networks that are dynamic in terms of traffic and/or topology. So, centralized scheduling approaches should be given up, since gathering all information to a single node and redistributing the computed schedule to every node imply a huge overhead in terms of signaling. In such a context, distributed algorithms are much more interesting as cell scheduling is addressed autonomously by each node on the basis of local information exchanged with neighbors. Thus, the signaling traffic is reduced, but the probability of collisions is increased since nodes have only a local view of the network.

A lot of distributed TSCH scheduling algorithms only relies on the basic 6top random cell selection. The inherent colliding cells are reduced a posteriori by a reactive relocation algorithm.

On-the-Fly (OTF) [5] implements a bandwidth reservation mechanism which compares the traffic sent by a node to each of its neighbors and the corresponding scheduled cells. Above/under a certain threshold, it triggers the negotiation to add/delete cells. The obtained cell schedule is not conflict-free but a mechanism called *housekeeping* relocates under-performing cells [6]. [7] provides two policies: the random strategy and the random contiguous one. Both strategies reduce colliding cells thanks to a relocation algorithm. It is the same for [8]. The problem of such approaches is that the relocation of colliding cells takes time, implies extra 6top traffic, and could have been avoided earlier by not selecting those colliding cells initially in the scheduling algorithm.

Several proactive algorithms follow this latter rule. Among them, DeTAS [9] and Wave [10] propose to build collision-free schedule. Both algorithms are not entirely distributed, as each child needs some input from its parent to locally execute the algorithm.

[2]The tuple (1,3) identifies the cell at timeslot 1 with channel offset 3.

The problem with Wave is that it requires the knowledge of the conflict graph, while no instruction is given to build this graph.

DeTAS avoids interference between devices by separating in time and frequency each branch of the DODAG and re-uses channels 3 hops away. However such a solution limits the spatial reuse of timeslot and channel in the whole DODAG. Compared to fully distributed algorithms, this approach allows to obtain collision-free schedules but incurs in return increased latency and signaling overhead. Instead, our approach is to improve fully distributed algorithms with lighter signaling. The next section details how our solution proactively reduces colliding cells in such random scheduling without adding any overhead.

III. Proposed Mechanism

As mentioned in Section I, our approach is complementary to scheduling algorithms since we aim at preventing the scheduling function of nodes from considering cells already allocated in their neighborhood.

In the example detailed in Section II-A, the negotiation results in the allocation of cell (4,2) to the link $G \rightarrow D$ which is already allocated to the link $H \rightarrow E$. If both links are active in this cell, a collision occurs. Now, this could have been prevented if G was aware of the previous allocation negotiated by H with E which is within the 1-hop neighborhood of G. More generally, this kind of collision occurs at the reception node Rx, and the neighboring nodes of Rx are the only nodes that can induce collisions. This occurs when Rx has two or more neighbors transmitting using the same scheduled cell, one of these must be a child, and the rest are neighbors transmitting to their parents.

Our solution is based on the idea of local mutual exclusion (LME). Local means the neighborhood of the reception node, and mutual exclusion is applied to prevent nodes from accessing a common resource at the same time. When a pair of Rx and Tx nodes reserves a new communication cell, Rx must inform its neighbors of the set of reserved cells.

A. Overhearing for achieving LME

The idea behind our approach is to overhear 6top transactions from our neighbors. 6top transactions are sent as unicast in the shared slot and received by all the neighbors at the PHY layer. Then, all of them except the destination reject the packet due to the MAC filtering. If we apply a slight modification at the MAC sublayer we can prevent filtering those transactions, and consequently, collect the cells reserved by the neighbors. Moreover it is achieved without introducing any overhead.

This mechanism is highly efficient at the network bootstrap since no dedicated cell has been scheduled. To allocate their first cells, nodes will send their 6P messages in the shared cells defined by the 6TiSCH minimal TSCH schedule [11]. Their location is either preconfigured or learnt by nodes at the bootstrap. One example of this schedule can be seen in Figure 2 where one shared cell is located at the first timeslot 0 of the slotframe with the channel offset 0. The access to shared cells follows slotted ALOHA rules. So, the higher the number of 6P transactions is, the higher the collisions and the retransmission in shared cells are. Once dedicated cells are allocated between a pair of nodes, further 6P transactions should use dedicated cells. In our approach, we always consider 6P transactions that use shared cells to benefit from the overhearing advantage.

In case the scheduling function uses 2-step transactions, we can collect the cells reserved by Rx from the ADD response, using white listing. Similarly this approach can be implemented with 3-step transactions by using blacklisting.

B. The Avoid Table

The collected cells will be saved in a table called *Avoid table*. The structure of this table is similar to that of TSCH where each cell is represented by a timeslot and a channel offset. Each cell in the Avoid table can be marked as available or reserved. The cell is available if it is not used by any of the neighbor nodes. If it is reserved, it should be avoided because of possible collisions. The last step of the transaction is to force scheduling functions to avoid cells found in the Avoid table.

C. Cell buffer

Due to the unreliable nature of low power wireless links, some transactions might not be received by all neighbors leading to inconsistencies among neighbors. Therefore those neighbors could reserve already allocated cells leading to collisions. To mitigate the effect of lost transactions, we added a cell buffer that stores the last k cells reserved by node with its children, where k is the length of this buffer. Thus, whenever a cell is reserved, instead of sending only the recently reserved cells, the cell buffer will be sent. As a result all the cells in the received buffer will be added to the avoid table. Technically, it means that each cell will be transmitted k times to the neighborhood. It effectively increases the number of successful reception of the cell reserved by neighbors, and consequently reduces the number of collision.

The value of k is calculated using a probabilistic model for the 6top transactions. We assume that the probability that a neighbor successfully receives the transaction is p and that the number of successfully received transmissions (Y) follows a binomial distribution law $\mathcal{B}(k,p)$ (each cell in the buffer will be transmitted k times). We denote by P_o the probability of at least one successful reception.

$$Y \sim \mathcal{B}(k,p) \qquad P_o = P(Y \geq 1) = 1-(1-p)^k$$

Solving this equality give us:

$$k = \left\lceil \frac{log(1-P_0)}{log(1-p)} \right\rceil$$

By simulating the topologies, we are running on, we have calculated the average value of the *Packet Delivery Ratio (PDR)* for the neighborhood of each node. To ensure our proposition, we have taken a low value of PDR ($p = 0.3$). According to the equations we will have the results in Table I:

k	8	9	10	11	12
P_o	94.23%	95.96%	97.17%	98.02%	98.61%

Table I
P_o *vs.* CELL BUFFER LENGTH k

Finally, with a buffer length of 10 we can ensure with $97\,\%$ confidence that the cell reserved will be received by all the neighbors.

IV. Results

A. 6TiSCH Simulator

In our tests, we have used the 6TiSCH simulator[3]. This simulator is an implementation of IEEE802.15.4e that uses RPL for Routing, 6top for the management of TSCH, and OTF (On-The-Fly) for scheduling [5].

Parameter	Value
Number of Motes	100
Number of cycles per run	500
Number of runs per simulation	1000
Timeslot duration	$10ms$
Slotframe length	101
Number of channels	16
Area	$1Km \times 1Km$
Topology constraint	$\geq$ 3 neighbors with PDR $50\,\%$
Radio sensitivity	-97 dBm
Radio range	100m
Traffic	1 packet/cycle for each node

Table II
SIMULATION PARAMETERS

We first had to overcome a limitation of the simulator which bypassed the exchange of 6P messages in the existing implementation. Cell negotiations between nodes were "instantaneous" not resulting in interferences, collisions and retransmissions. We fix it and now cell negotiations trigger the exchange of 6P packets in shared cells. Secondly, we have implemented the mechanism of collision prevention in the 6top layer, by adding a structure to collect cells in each

[3]https://bitbucket.org/6tisch/simulator/src

node, and by adding the modifications explained previously in the 6top transactions.

The simulations were done over a wide range of topologies and each approach was run on the same topologies to ensure fairness.

Table II summarizes the simulation parameters.

B. *Comparison without housekeeping*

First we have compared the performance of OTF with and without our approach. Housekeeping is disabled. As explained before, housekeeping is based on a periodic function that will check the underperforming scheduled cells by tracking their PDR. Under a certain threshold, a cell will be relocated.

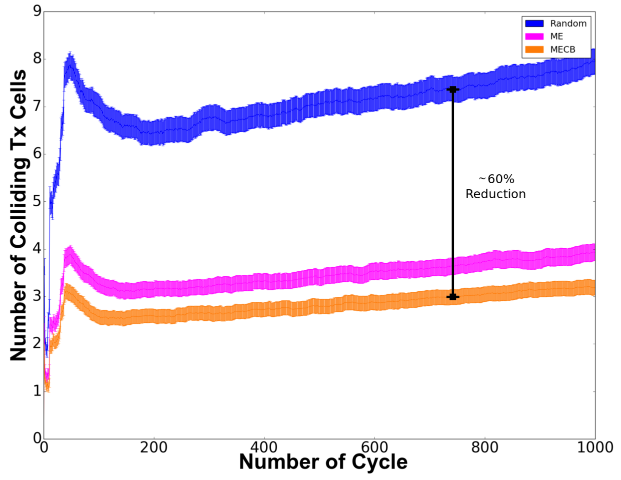

Figure 3. Number of colliding Tx cells as function of slotframe

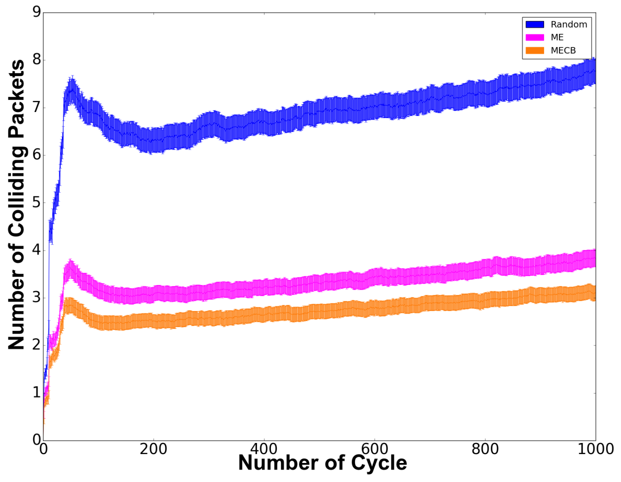

Figure 4. Number of colliding packets as function of slotframe

In Figure 3 and 4, *Random* stands for original OTF where nodes only avoid cells already reserved with parents and children. The curve *ME* is OTF supplemented with our local mutual exclusion mechanism but with no cell buffer while the curve *MECB* implements a cell buffer of a length equal to 10.

The simulation is done over 500 runs. Each run consists of 1000 cycles where the cycle refers to one slot-frame.

At the beginning of the simulation, the increase of collided cells corresponds to the bootstrap of the network where all nodes, during the creation of the DODAG by RPL, will start requesting dedicated cells by sending 6top messages in shared cells. Collisions in shared cells will be progressively resolved by CSMA/CA before being able to schedule first dedicated cells. As we can see in Figure 3, colliding dedicated cells increase and then stabilizes when all requested cells are scheduled.

The simulation shows a reduction of 62 % in the number of colliding Tx cells. The use of the cell buffer is responsible for a 12 % reduction between ME and MECB. We can conclude that the lost 6top transactions were causing 12% of the collisions.

In Figure 4 we can see the reduction induced by our approach in the number of colliding packets. The collided packets are directly related to the traffic of the network. In the simulation we have a constant traffic on the nodes (1 packet/cycle for each node). A colliding Tx cell does not necessarily mean that we will have a collision each cycle in this cell. For instance, one of the nodes will transmit packets while the other might have nothing to transmit.

Similarly our approach shows a reduction of 60 % in the number of colliding packets. The local mutual exclusion mechanism with cell buffer also shows an improvement over the local mutual exclusion without cell buffer.

Despite their reduction, collisions still remain for the following reasons:

- We still have a slight chance of losing 6P transactions.
- A special configuration where a node Tx2 belongs to the neighborhood of a node Rx1 but Rx2 (the parent of Tx2) has neither Tx1 or Rx1 in its neighborhood. If Rx2 first reserves a cell, Rx1 will not be aware of that transaction and will still be able to reserve the same cell.

The first issue should be reduced to a minimum with the mechanism of the cell buffer. The second issue can be solved by a reactive mechanism like housekeeping that is being evaluated in next section.

C. *Comparison with housekeeping*

As seen in Section II-B, housekeeping is another solution to solve the problem of colliding cells. However, it is a reactive solution which allows the collision to occur while our solution is proactive and prevents collisions to happen in the first place.

In Figures 5 and 6, we compare OTF with housekeeping alone to OTF with housekeeping and MECB.

Results show in particular a decrease of the peak of at least 60% of colliding cells and also a 60% of colliding packets in the bootstrap phase.

Housekeeping alone will take extra time to relocate the defected cells while MECB, with less colliding cells, will speed up the convergence of housekeeping.

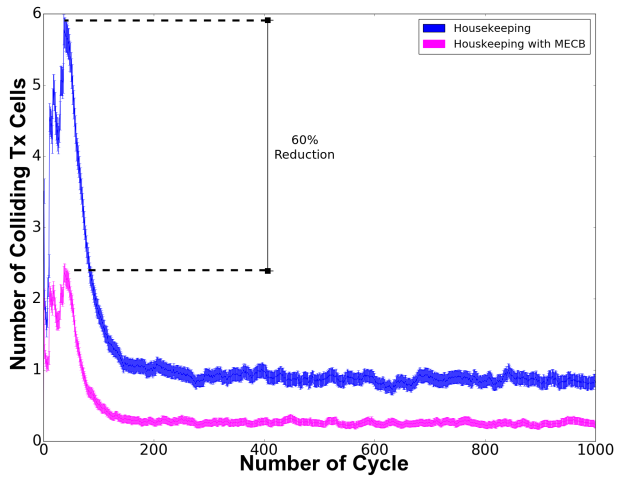

Figure 5. Number of colliding Tx cells as function of Cycle comparison while using housekeeping

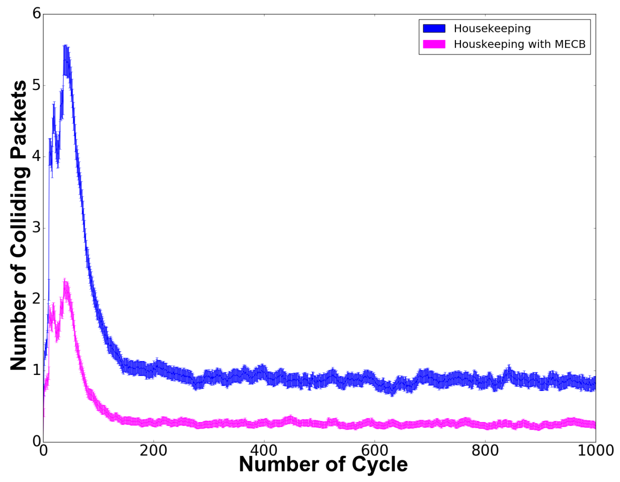

Figure 6. Number of colliding packets as function of Cycle comparison while using housekeeping

V. Conclusion

In this paper, we present the problem of conflicting TSCH schedules computed by current distributed scheduling algorithms. We explained why a distributed scheduling may lead to collisions, yet it still has a lot of benefits in the means of time required to reserve a cell and of the complexity of the reservation process. We propose a solution to reduce conflicting allocations in TSCH schedules set up by 6TiSCH distributed algorithms. The idea is to achieve local mutual exclusion by making nodes able to overhear 6P transactions exchanged by their neighbors. This mechanism is done without any overhead, provided the transactions are sent in shared slots. The 6P transactions have been also augmented with a short-term memory of past allocated cells. This mechanism makes the overhearing more robust for neighbors that may miss some transactions because of punctual variations in the quality of their wireless link.

The performance of our approach has been evaluated through simulations. The results have shown significant reductions in the number of colliding packets.

For the future work, we are currently interested to study the effect of traffic on different scheduling schemes. After the emerging of the traffic aware scheduling on IEEE802.15.4 the traffic effects on this network is a hot topic that needs more investigation.

Acknowledgment

The authors would like to thank ANR Persyval project IOIT and ANR Persyval DACRAW for funding this project.

References

[1] *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003): IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications f.* IEEE, 2006. [Online]. Available: https://books.google.fr/books?id=S-13nQAACAAJ

[2] "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)*, pp. 1–225, Apr. 2012.

[3] Q. Wang, X. Vilajosana, and T. Watteyne, "6top Protocol (6P)," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-6top-protocol-05, May 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-6top-protocol-05

[4] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6550

[5] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, A. Grieco, and T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," *IEEE Sensors Journal*, vol. 16, no. 2, Jan. 2016.

[6] K. Muraoka, T. Watteyne, N. Accettura, X. Vilajosana, and K. S. Pister, "Simple Distributed Scheduling With Collision Detection in TSCH Networks," *IEEE Sensors Journal*, vol. 16, no. 15, pp. 5848–5849, May 2016.

[7] F. Theoleyre and G. Z. Papadopoulos, "Experimental validation of a distributed self-configured 6tisch with traffic isolation in low power lossy networks," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16, New York, NY, USA, 2016. [Online]. Available: http://doi.acm.org/10.1145/2988287.2989133

[8] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed pid-based scheduling for 6tisch networks," *IEEE Communications Letters*, vol. 20, no. 5, pp. 1006–1009, 2016.

[9] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," *IEEE Internet of Things Journal*, vol. 2, no. 6, Dec. 2015.

[10] R. Soua, P. Minet, and E. Livolant, "Wave: a distributed scheduling algorithm for convergecast in ieee 802.15. 4e tsch networks," *Transactions on Emerging Telecommunications Technologies*, 2015.

[11] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8180.txt