Éviter les collisions dans les réseaux 6TiSCH

Olivier Alphand[4], Karine Altisen[1], Rodolphe Bertolini[4], Stéphane Devismes[2], Ali J. Fahs[3] et Franck Rousseau[4]

[1]*Grenoble INP, VERIMAG ;* [2]*UGA, VERIMAG ;* [3]*Université de Rennes, IRISA ;* [4]*Grenoble INP, LIG*

Les réseaux multi-sauts 802.15.4e TSCH s'appuient sur des échéanciers de communication efficaces. Notre solution complète les algorithmes de construction de tels échéanciers en limitant la réutilisation de cellules temps-fréquence déjà utilisées par des nœuds voisins. Nos simulations montrent un gain significatif par rapport à l'existant.

Mots-clefs : IoT, IEEE802.15.4e, TSCH, 6TiSCH, 6top, Scheduling Functions

1 Introduction

Among many standards dedicated to the *Internet of Things (IoT)*, the initial version of IEEE802.15.4 aimed at low power short range communications. However its MAC layer presented several limitations (no channel diversity, unbounded delay, and poor multi-hop topology support). To overcome these limitations, IEEE802.15.4e extended the standard by proposing a new mode: *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 to another. The channel of a cell is derived from the channel offset and the absolute slot number. There are two types of TSCH cells: *shared cells* where multiple nodes may either receive or transmit, and *dedicated cells* for contention-free link-local communications between two neighbors. IETF (Internet Engineering Task Force) 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 called *6top*. It includes the definition of algorithms to schedule dedicated cells and a 6top protocol (*6P*) to negotiate cells between nodes. We focus here on improving the distributed scheduling algorithms that are mainly designed to address the problem of convergecast traffic towards a unique sink in multi-hop networks. *RPL (Routing Protocol for Low-Power and Lossy Networks)* organizes the network as a *DODAG (Destination-Oriented Directed Acyclic Graph)* 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 randomly select cells among those that are not already used by either negotiating nodes with their respective DODAG parents or children. However, the cell selection should also preclude dedicated cells already reserved by other neighbors 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 performance of the *On-the-Fly* (OTF) [4] scheduling algorithm with and without our approach. Results show a significant improvement.

Roadmap. Section 2 gives an overview of 6TiSCH and related works. Section 3 describes our solution and Section 4 details some results obtained by simulations. We conclude in Section 5.

2 Background

6TiSCH operation sublayer (6top). To integrate TSCH within the IoT stack, the IETF 6TiSCH Work Group defines the 6TiSCH operation sublayer (6top). 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 transmitting node (Tx) and a receiving node (Rx) that communicate (*i.e.*, Rx is a parent of Tx in the DODAG), 6P supports three types of operations, encapsulated into a transaction, that can be triggered by *Scheduling Functions* (SF). If the *Scheduling Algorithm* determines that Tx needs for more (resp. less) cells to communicate with Rx, it will issue *a 6top transaction to reserve* more cells (resp. *to release* some TSCH cells) in the TSCH schedule. If the *Relocation Algorithm* detects that

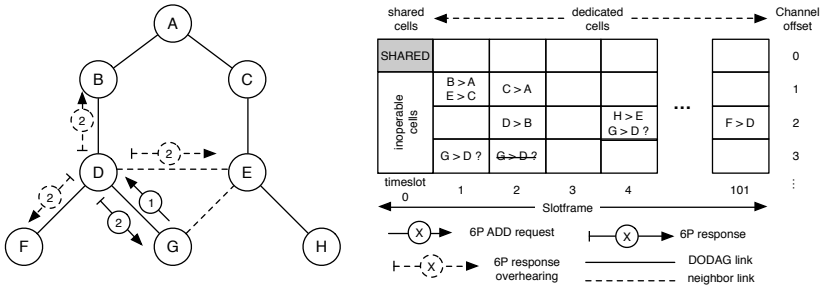

Figure 1: 6P transactions and TSCH schedule update

a dedicated cell is facing collisions, this defected cell is *replaced through 6top transactions*. A 6top transaction consists of a negotiation between Tx and Rx that updates the TSCH table and takes place in two or three steps. The SF decides whether to use a 2-step or a 3-step transaction, or a mix of them. Moreover, the SF has two options to communicate the list of cells: either *black listing* where the node sends all cells in use, or *white listing* where the node sends a number of cells available for communication. We detail in Figure 1 the sequence of a 2-step transaction using white listing to allocate two cells for link $G \rightarrow D$:

1. The SF of node G detects that two additional cells are needed and selects three candidate available cells $\{(1,3),(2,3),(4,2)\}$[†] before triggering a 6P ADD request towards node D in the shared slot.
2. The SF of D precludes the cell (2,3), selects 2 cells $\{(1,3),(4,2)\}$, and issues a 6P response to G.

As shown in the 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 in which either negotiating nodes communicate with their DODAG parents or children. The selection of the candidate cell is then randomly done in the set of remaining cells. In the previous negotiation, D will not consider timeslot 2 since it already uses it to communicate with its parent B. G has no dedicated cells so it does not remove any cell *a priori*.

Related work. 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 by a single node and redistributing the computed schedule to every node would imply a huge overhead in terms of signaling. In distributed solutions, cell scheduling is addressed autonomously by each node on the basis of local information. Thus, the signaling traffic is reduced, but the probability of collisions increases since nodes only have a partial view of the network. Many distributed TSCH scheduling algorithms only rely on the basic 6top random cell selection. The inherent colliding cells are reduced *a posteriori* by a reactive relocation algorithm. *On-the-Fly (OTF)* [4] implements a bandwidth reservation mechanism that 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 resulting cell schedule is not conflict-free but a mechanism called *housekeeping* [3] relocates underperforming cells. [6] and [2] provide random policies that reduce the collisions and a relocation algorithm; but relocation takes extra time and implies extra 6top traffic which could be avoided *a priori*. Several proactive algorithms, such as DeTAS [1] and Wave [5] build a collision-free schedule, but both of them are not entirely distributed. Compared to fully distributed algorithms, they allow to obtain collision-free schedules but increase latency and signaling overhead. Instead, our approach is to improve fully distributed algorithms by using light extra signaling.

3 Proposed Mechanism

Our approach is complementary to scheduling algorithms as we aim at preventing the scheduling function of nodes from considering cells already allocated in their neighborhood. In Figure 1, the negotiation results in allocating (4,2) to link $G \rightarrow D$, which is already allocated to link $H \rightarrow E$. If both links are active in this cell, a collision occurs. This could have been prevented if G was aware of the previous allocation negotiated by H with E (*n.b.*, E is in the 1-hop neighborhood of G). More generally, this kind of collision occurs at some reception node when two or more of its neighbors are transmitting in the same scheduled cell.

Overhearing. Our approach first consists in overhearing 6top transactions from neighbors. 6top transactions are sent as unicast in a shared slot and received by all neighbors at the PHY layer. Then, all of them except the destination reject the packet due to the MAC filtering. With a slight modification at the MAC sublayer, we disable the filtering of those transactions and collect cells reserved by the neighbors, without any overhead. This mechanism is highly efficient at the bootstrap since no dedicated cell has been scheduled. To allocate their first cells, nodes send their 6P messages in the shared cells defined by the

[†] The tuple (i,j) identifies the cell at timeslot i with channel offset j.

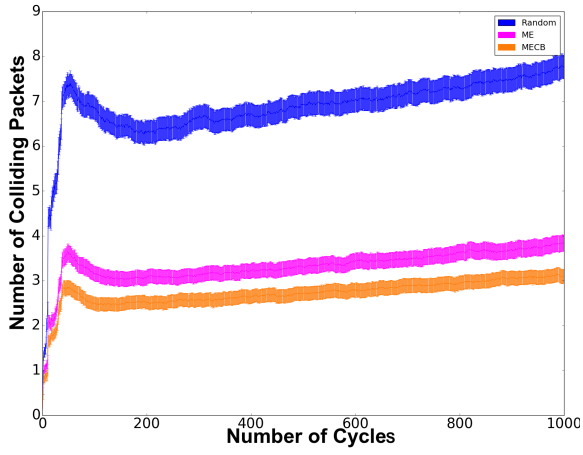

Figure 2: Number of colliding packets per cycle without housekeeping

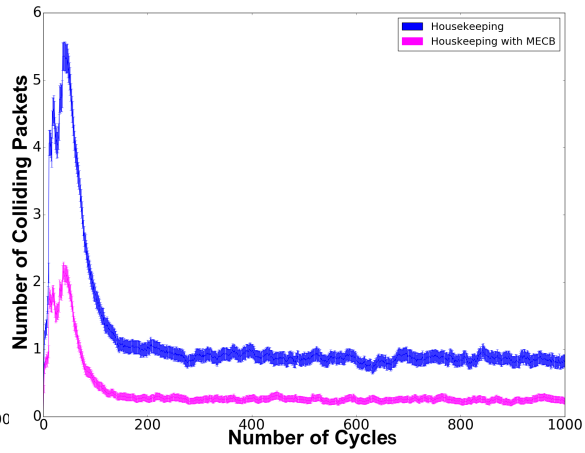

Figure 3: Number of colliding packets per cycle with housekeeping

6TiSCH minimal TSCH schedule. Their location is either preconfigured or learnt by nodes at bootstrap. One example of this schedule can be seen in Figure 1 where one shared cell is located at timeslot 0 of the slotframe with channel offset 0. The access to shared cells follows slotted ALOHA rules. So, the higher the number of 6P transactions, the more collisions and retransmissions in shared cells. 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 overhearing. 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 black listing.

Avoid table. The collected cells are saved in a table called an *avoid table*. The structure of this table is similar to that of TSCH in which 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.

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 add a cell buffer that stores the last k cells reserved by a node with its children, where k is the size of the buffer. Thus, whenever a cell is reserved, instead of sending only the recently reserved cells, we send the cell buffer. As a result, all the cells in the received buffer are added to the avoid table. This means that each cell is transmitted k times to the neighborhood. This effectively increases the number of successful receptions of the cell reserved by neighbors, and consequently reduces the number of collisions. The value of k is calculated as follows. 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 $\mathcal{B}(k,p)$ (each cell in the buffer will be transmitted k times). The probability P of at least one successful reception is then $P = P(Y \geq 1) = 1-(1-p)^k$. We have evaluated, by simulation, the average value of the *Packet Delivery Ratio (PDR)* for the neighborhood of each node. As a pessimistic and conservative approximation of those results, we fixed $p = 0.3$. The equation then ensures that, with a buffer size of $k = 10$, we have $P = 97\,\%$ confidence that the cell reserved will be received by all the neighbors.

4 Results

In our tests, we have used a 6TiSCH simulator[‡], which 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. We first had to overcome a limitation of the simulator that bypassed the exchange of 6P messages in the existing implementation. Cell negotiations between nodes were “instantaneous”, neither resulting in interference, collisions

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

nor retransmissions. We have fixed this issue so that cell negotiations trigger the exchange of 6P packets in shared cells. Second, we have implemented our proposal. The simulations were done over a wide range of topologies and each tested protocol was run on the same topologies to ensure fairness. Below, we summarize the simulation parameters:

Number of motes	100	Number of runs per simulation	500	Number of cycles per run	1000
Area	1 km × 1 km	Radio range	100 m	Radio sensitivity	−97 dBm
Timeslot duration	10 ms	Number of channels	16	Slotframe length	101
Traffic for each node	1 packet/cycle	Topology constraint	≥ 3	neighbors	with PDR 50 %

Comparisons without Housekeeping. First, we have compared the performance of OTF with and without our approach when housekeeping is disabled. Housekeeping is based on a periodic function that checks the underperforming scheduled cells by tracking their PDR. Under a certain threshold, a cell will be relocated. In Figure 2, *Random* stands for the original OTF protocol in which nodes only avoid cells already reserved with parents and children. The *ME* curve corresponds to OTF with our proposed mechanism but with no cell buffer while the curve *MECB* represents the behavior of the protocol with a cell buffer of size $k = 10$. At the beginning of the simulation, the increase of collided cells corresponds to the bootstrap of the network: during the creation of the DODAG by RPL, all nodes 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 the first dedicated cells. Figure 2, shows a reduction of 60 % in the number of colliding data packets resulting from our approach. The cell buffer mechanism also improves results over the scenario without cell buffer. Despite the reduction of their number, there still remain collisions since:

1. Nodes still have a slight chance of losing 6P transactions.
2. In a specific configuration in which 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 by intensively using the mechanism of cell buffer. The second issue can be solved using a reactive mechanism like housekeeping (see below).

Comparisons with Housekeeping. 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 Figure 3, we compare OTF with housekeeping alone to OTF with housekeeping and *MECB*. The results show in particular a decrease of the peak of at least 60% of colliding packets in the bootstrap phase. Indeed, housekeeping alone takes some extra time to relocate the defected cells while *MECB*, with less colliding cells, speeds up the convergence of housekeeping.

5 Conclusion

We have proposed a solution to reduce conflicting allocations in TSCH schedules created with 6TiSCH distributed algorithms. The idea is to make nodes able to overhear 6P transactions exchanged by their neighbors to prevent collisions. This mechanism implies a low overhead, provided the transactions are sent in shared slots. The 6P transactions have also been augmented with a short-term memory of past allocated cells, which makes overhearing more robust for the neighbors that may miss some transactions because of intermittent variations in the quality of their wireless links. The performance of our approach has been evaluated through simulations. The results show a significant reduction of collisions.

References

[1] 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*, 2(6), 2015.

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

[3] K. Muraoka, T. Watteyne, N. Accettura, X. Vilajosana, and K. S. Pister. Simple Distributed Scheduling With Collision Detection in TSCH Networks. *IEEE Sensors Journal*, 16(15):5848–5849, 2016.

[4] 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*, 16(2), 2016.

[5] 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.

[6] F. Theoleyre and G. Z. Papadopoulos. Experimental validation of a distributed self-configured 6tisch with traffic isolation in low power lossy networks. MSWiM, 2016.