Localized Scheduling for End-to-End Delay Constrained Low Power Lossy Networks with 6TiSCH

Inès Hosni, Fabrice Theoleyre, Noureddine Hamdi

Localized Scheduling for End-to-End Delay Constrained Low Power Lossy Networks with 6TiSCH

Inès Hosni
Laboratory Systems Communications
University of Tunis El Manar
National Engineering School of Tunis
email: ines.hosni@hotmail.fr

Fabrice Théoleyre
CNRS, ICube
University of Strasbourg, France
email: theoleyre@unistra.fr

Noureddine Hamdi
Laboratory Systems Communications
University of Tunis El Manar
National Engineering School of Tunis
email: noureddine.hamdi@ept.rnu.tn

***Abstract*—The IoT expects to exploit IEEE802.15.4e-TSCH, designed for wireless industrial sensor networks. This standard relies on techniques such as channel hopping and bandwidth reservation to ensure both energy savings and reliable transmissions. The 6TiSCH working group currently proposes to exploit the RPL routing protocol on top of the IEEE802.15.4-2012-TSCH layer. Since many applications may require low end-to-end delay (e.g. alarms), we propose here a distributed algorithm to schedule the transmissions while upper bounding the end-to-end delay. Our strategy is based on stratums to reserve time-bands for each depth in the routing structure constructed by RPL. By allocating a sufficient number of timeslots for the possible retransmissions, we guarantee that any packet is delivered during one single slotframe, wherever the source is located. Experiments on a large scale testbed prove the relevance of this approach to reduce the end-to-end delay while minimizing the number of collisions, prejudicial to the reliability in multihop networks.**

***Index Terms*—distributed scheduling; end-to-end delay; IEEE 802.15.4e-TSCH; 6TiSCH;**

I. Introduction

The IEEE802.15 working group has proposed the IEEE802.15.4e amendment [1]. In particular, the TSCH mode aims at improving the reliability for industrial sensor networks in noisy environments. A common schedule aims at reserving a certain amount of bandwidth for each flow, and channel hopping aims at defeating narrow band noise. This standard is particularly accurate for the Internet of Things, where devices transmit periodically their measures to the Internet, through a border router [2].

Nodes maintain a slotframe structure, i.e. a sequence of timeslots which repeats over time. Then, the schedule specifies the action of each node during each timeslot. At the beginning of a slot, a node either sleeps or turns its radio on to receive or transmit a frame. By appropriately selecting the set of active nodes during a timeslot, we can avoid the collisions.

However, the current stack of protocols for the Internet was not designed initially to cope with a MAC layer based on reservations. Thus, the 6TiSCH [3] working group aims at defining a set of protocols to fill the gap between the Link and Network layers. While RPL [4] constructs the end to end routes and maintains the control plane, 6top [5] reserves the transmission opportunities between a pair of neighbors.

To assign the resource for each packet is currently a very challenging objective: 6top has to define which *cell* has to be used for each flow, along each hop from the source to the border router. Scheduling in multihop wireless networks has already been widely studied in the literature [6], [7], [8]. Some solutions also proposed distributed algorithms adapted for channel hopping [9] or TSCH networks [10].

In this paper, we present a distributed scheduling solution to reduce the end-to-end delay. Intuitively, we should select consecutive timeslots along the route to reduce the end-to-end delay. However, we must also take into account the link reliability: retransmissions may impact negatively the delay if they are not properly scheduled.

We propose here a distributed scheduling based on *stratums*: all the nodes for a given depth own to the same *stratum*, and all their transmissions are scheduled in the same time-frequency block (a *band*). These bands should be carefully dimensioned to avoid the funneling effect [11]. Besides, we propose to reuse the On The Fly scheduling (OTF) mechanism [12]. When a node has too much traffic to forward, it negotiates with its parent additional timeslots, picked in the accurate *band*. Scheduling all the retransmissions in the same band allows the end-to-end delay to be contained.

To the best of our knowledge, we propose the first algorithm to assign reactively the cells in 6TiSCH, with uniquely localized information, and upper-bounding the end-to-end delay.

The contribution of this paper is threefold:

1) we present a distributed scheduling algorithm which reduces the end-to-end delay by organizing the network in stratums. A packet should be delivered before the end of the slotframe;
2) we explain how to implement this scheduling algorithm in the 6TiSCH stack, reserving new cells when a low link quality is measured locally. Our solution is entirely decentralized and reactive;
3) we provide experimental results conducted on the FIT-IoT Lab platform to validate our approach.

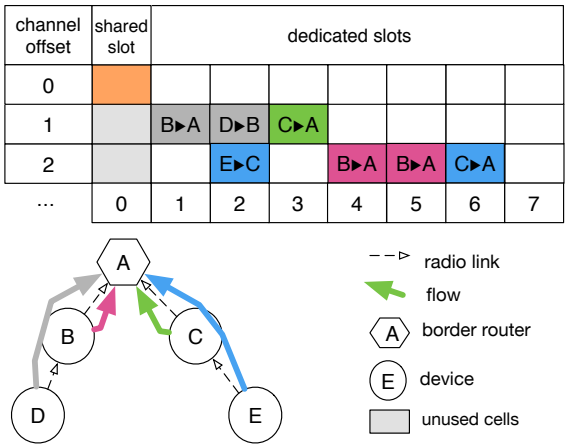

Fig. 1. Schedule in a IEEE802.15.4-TSCH network – illustration of a slotframe with 8 timeslots

II. Related Work

A. IEEE802.15.4-TSCH

IEEE802.15.4e has proposed the TSCH mode for industrial wireless sensor networks. To improve the reliability while maximizing energy savings, a schedule is defined and repeated periodically.

At the beginning of each timeslot, a device examines the schedule to know if it has to wake-up to transmit or receive a packet. A timeslot can be either dedicated (without contention) or shared (with a slotted CSMA-CA mechanism to solve the conflicts between the contenders). During an *incoming cell*, a node waits for a reception, while it is in transmission mode during the *outgoing cell*. When a node is neither receiver nor transmitter, it turns its radio off.

To improve the reliability, TSCH proposes to implement channel hopping. In the TSCH jargon, a *cell* is a pair of timeslot and *channel offset*. The channel offset is translated into a frequency to actually use during a given timeslot:

$$freq = (ASN + ch\_offset)\%16 \quad (1)$$

where ASN (absolute sequence number) counts the number of timeslots since the beginning, and ch_offset is the channel offset assigned to this cell in the schedule.

Let's consider the schedule illustrated in figure 1. We have 1 shared cell at the beginning of the slotframe, using the channel offset 0. The shared cell is typically used for control traffic (i.e. new reservations, routing control packets, etc.). The other cells are dedicated: only the owners of a cell can transmit a packet, without contention. Each DODAG link has one or several cells to forward the packets to its parent. The link $A \rightarrow$B (pink) reserved for instance two dedicated cells either because it has many packets to forward, or because retransmissions are expected.

B. 6TiSCH

The 6TiSCH IETF working group designs the protocols to operate IPv6 (6LoWPAN) over a reservation based MAC layer (IEEE802.15.4-TSCH). 6TiSCH introduces the concept of track to reserve an amount of dedicated cells for a particular flow [13]. Hop by hop, each intermediary node inserts in its schedule some cells for each non best effort flow (i.e. track instance $\neq 0$). Label switching is implicit: a node knows the track associated to an incoming cell, extracted directly from the schedule. Thus, it just has to forward this packet in an outgoing cell with the same track id.

6top defines how a node may negotiate a *cell* with one neighbor: the enquirer specifies a list of available $< timeslot, channel\_offset >$ and the number of cells it asks for [5]. The neighbor will accept the request if these cells are available also in its schedule: it sends an Information Element to notify the enquirer. However, 6top does not define *which* cells should be selected.

Let's consider the figure 1 which illustrates a possible schedule with 4 different tracks. We see that the link $B \rightarrow A$ supports two tracks with respectively the source B (in pink) and the source D (in gray). Since each track has dedicated resource, we guarantee traffic isolation: the packets of D do not impact the traffic of B. For the flow $(D \rightarrow A)$, the incoming cell in B is located *after* the outgoing cell. Thus, B has to buffer the corresponding packet during the whole slotframe, increasing drastically the end-to-end delay.

In this paper, we use the 6TiSCH stack to reserve reactively the cells for each track: each flow reserves hop-by-bop its own dedicated cells, while taking care of limiting the end-to-end delay.

C. Traffic Aware Scheduling Algorithm

To assign the resource for each packet is currently a very challenging objective: 6top has to define which *cell* should be used for each flow, along each hop of the route to the border router. While 6TiSCH defines how the cells are negotiated (with the 6top protocol), any scheduling algorithm may be actually implemented. We propose here a distributed scheduling algorithm adapted to 6top.

Ghosh *et al.* [14] proposed to minimize the schedule length in a multichannel TDMA environment. Tsitsiklis *et al.* [7] studied the tradeoff between a centralized and a distributed scheduling. By adopting a queue theory based approach, they demonstrated a centralized algorithm is more efficient. TASA proposed to construct a centralized scheduling for a multihop IEEE802.15.4-TSCH network [8]. Yigit *et al.* [15] studied the impact of routing on the scheduling: using unreliable links increases the number of timeslots required to achieve a minimum reliability. However, these centralized approaches assume the radio topology is known a priori: the protocol must know precisely the group of interfering links. These approaches are well suited for industrial networks with strict requirements on reliability and delay.

DeTAS proposed a decentralized version of TASA [10]: the children of the border routers collect the information and

$ETX(A,B)$	ETX value from A to B
$nbCellsOut(A,B)$	Number of outgoing cells (transmissions) from A to B
SF_{length}	Number of cells (i.e. the slotframe length)
T_{slot}	Timeslot duration (by default 15ms)
$p = \{S..D\}$	path from S to D
d_{max}	Maximum hop distance for block re-utilization (i.e. frequency reuse)

TABLE I
NOTATIONS

compute the schedule of their subtree (called micro-schedules). Finally, the micro-schedules are re-arranged into a globally acceptable schedule. Thus, the scheduling is still concentrated in a few nodes, which are aware of the radio interference. Phung *et al.* [16] proposed to use a Reinforcement Learning based scheduling algorithm to cope with a variable traffic. However, the authors do not propose to use dedicated cells: the nodes have always to execute a CSMA-CA phase before transmitting their packets. We propose rather an approach similar to Z-MAC [17] where a pair of nodes negotiate locally the timeslots they will use further. However, Z-MAC does not exploit a fully synchronized FTDMA network.

III. Problem Statement

We consider a model designed for Low Power Lossy networks (LLN) organized by RPL in a single DODAG (Destination Oriented Directed Acyclic Graph), anchored in a border router. Each node maintains its *rank* denoting its virtual distance from the border router. Typically, the rank can be the average cumulative number of transmissions (ETX) along the path to the border router [4].

We consider here the standard version of RPL, where a node uses only its preferred parent to route its packets. Thus, a node has to negotiate a set of cells with its parent, next hop to the border router. The traffic may be constant or variable, and we use tracks to reserve dedicated cells for each flow. When the traffic is sporadic, a dedicated cell is used infrequently (e.g. one every 10 slotframes).

TASA [10] is particularly efficient to schedule in a centralized manner the cells while minimizing the end-to-end delay. However, centralized scheduling deals inefficiently with variability: when a new node or flow is inserted, the schedule must be updated. In the same way, bad radio links require to over-provision cells for retransmissions. To react quickly to changes, we propose here a localized scheduling algorithm, robust to any change (traffic, reliability, topology).

A. Large end-to-end Delay with a Random Distributed Scheduling

We first compute here the average end-to-end delay we may obtain with a random scheduling algorithm. More precisely, when a pair of nodes has to negotiate a cell, the transmitter selects randomly a free timeslot and channel offset. Then, it sends a request to the receiver to verify if this cell is also free for the other side. The process reiterates until a common free cell is selected.

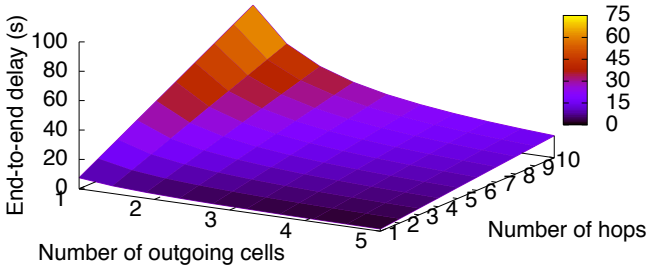

Fig. 2. End-to-end delay with a slotframe of 1001 timeslots (=15s)

Without loss of generality, let's assume the first hop has selected the timeslot 0. A packet has to be enqueued until the next outgoing cell, and these cells are uniformly distributed in the slotframe. Besides, a packet may be received uniformly in the slotframe. Thus, it needs to wait on average:

$$OutCell\_delay = \frac{SF_{length} * T_{slot}}{2 * nbCellsOut(A,B)} \quad (2)$$

with SF_{length} being the length of the slotframe, T_{slot} the duration of a timeslot and $nbCellsOut(A,B)$ the number of outgoing cells.

Besides, a packet needs possibly several retransmissions if the link is unreliable. Precisely, it needs on average ETX transmissions [18]: the average number of transmissions from A to B before receiving an acknowledgement.

Finally, the packet is enqueued during:

$$hop\_delay(A \rightarrow B) = \frac{SF_{length} * T_{slot}}{2 * nbCellsOut(A,B)} * ETX(A,B) \quad (3)$$

with $ETX(A,B)$ the ETX from A to B.

Since we assume a packet may be generated at anytime, the end-to-end delay for a packet along the route p is finally:

$$E2E\_Delay(p) = SF_{length} * T_{slot} \sum_{i=1}^{|p|-1} \frac{ETX(N_i, N_{i+1})}{2 * nbCellsOut(N_i, N_{i+1})} \quad (4)$$

Figure 2 illustrates the end-to-end delay obtained with a typical slotframe of 1001 slots. This slotframe models typically an application where a flow has to send one packet every 15 seconds.

In conclusion, we may reduce the end-to-end delay by allocating more cells. However, this over-provisioning increases both the number of collisions and the energy consumption.

Besides, the end-to-end delay increases linearly with the network diameter. While we implemented this solution as a comparison purpose, we have to propose a specific smarter solution to reduce the end-to-end delay.

IV. Distributed Stratum Scheduling

We propose here a localized scheduling to be robust to any change (topology, traffic, routing paths). We are convinced a distributed schedule, reactively computed, may cover the needs of various applications (e.g. smart homes).

We propose to guarantee a maximum end-to-end delay equal to the slotframe duration: any packet which is generated at the beginning of the slotframe should be delivered before the end of the slotframe. By fixing the slotframe length, the network administrator is able to also fix the maximum delay.

We divide the network in *stratums*: all the nodes which are k hops far from the border router constitute the stratum k. We denote by *depth* the hop distance from the border router. Thus, each node includes in its DIO (DODAG Information Object, a RPL control packet) a field denoting its depth.

We aim now at assigning a time-frequency block (a *band*) to each stratum. All the nodes in the stratum k must pick their timeslots in the block k. We construct a global schedule in which the blocks from contiguous stratums are consecutive. In this way, we guarantee a packet is delivered before the end of the slotframe. Besides, our algorithm is localized since the stratum is directly determined by the depth.

With On-The-Fly scheduling (OTF), a node will monitor the number of packets to forward. When its traffic exceeds its outgoing capacity, OTF asks 6top to reserve a new cell.

We will now to define which block will be assigned to each stratum.

A. Sizing the number of stratums and their width

We consider that the nodes are uniformly distributed in a given area, and all the nodes generate the same amount of traffic. We denote by *block* all the nodes which are equidistant (in hops) from the border router. Our algorithm consists in assigning a portion of the schedule to each *block*.

We will first explain how we assign a schedule to the nodes which are at most d_{max} hops far from the border router. We will explain in the next subsection how frequency re-use allows the other blocks to receive also a non interfering portion of the schedule.

The nodes closest to the border router have more traffic to forward (the so-called funneling effect). Thus, the block i should be larger than the block $i+1$. We propose consequently to use a recursive division: the block i is twice as large as the block $i+1$.

If the traffic is not uniformly distributed in the network, we may divide differently the global schedule into blocks. In the extreme case, if all the packets are generated by the nodes d_{max} hops far, all the blocks should have the same size.

B. Frequency re-use

Many interfering models consider interference may be neglected when two nodes are sufficiently far [19]. Let's denote by d_{max} the number of hops after which we may assume no interference arises.

Practically, we fix the number of blocks to be equal to d_{max}. A node in the stratum k will use the block $k \mod (d_{max})$, with $\mod()$ denoting the modulo operator. Thus, nodes which are more than d_{max} hops far from the border router re-use one already allocated cell, without creating interference since we consider they are sufficiently far.

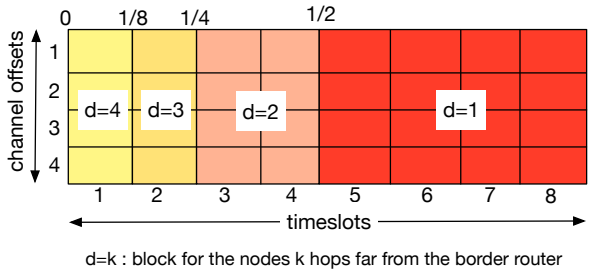

Fig. 3. Division of the schedule into disjoint blocks (4 channel offsets, 8 timeslots)

V. Traffic Isolation with 6TiSCH

We modified the openwsn (https://openwsn.atlassian.net) implementation of the 6TiSCH stack. It provides an open-source implementation of the whole 6TiSCH protocol's family (IEEE802.15.4-tsch, RPL, 6LoWPAN, CoAP) .

We modified the current openwsn implementation to cope with tracks. The best-effort track uses shared slots (with contention) for the RPL control packets (e.g. DIO, DAO) and 6top control packets (to negotiate the dedicated cells to use). Besides, each packet generated by an application is attached to a particular track (a track id of 16 bits, and a track owner of 64 bits): a flow reserves hop-by-hop its own dedicated resource, with dedicated slots without contention.

On-The-Fly scheduling is in charge of deciding how many cells to allocate for each radio link. We propose here a simple version of OTF: when the number of packets in the queue for a given track exceeds the number of outgoing cells for this track in the slotframe, OTF asks 6top to reserve some cells. More precisely, 6top has to reserve the difference between the number of packets and the number of outgoing cells.

6top then engages a bidirectional negotiation, as specified by 6TiSCH. A `Link-Request` is transmitted to the parent with the cells selected by the scheduling algorithm. Then, the parent replies with a `Link-reply` to accept or reject this allocation, depending of the availability of these cells in its schedule.

We propose here a reactive approach, in which OTF automatically adjusts the number of cells according to the reliability of the link. More retransmissions mean the queue becomes overfilled, requiring new outgoing cells.

VI. Experimental Results

To evaluate thoroughly the behavior of our solution, we used the FIT/IOT-LAB testbed (https://www.iot-lab.info/, cortex-m3 nodes). We fixed the values for each parameter according to table VI, using openwsn modified as described in the previous section. We measured the following metrics:

Experiment duration	300 s
Traffic type, rate	CBR, inter packet time= $12s$
Data packet size	127 bytes
MAC layer	IEEE 802.15.4e-TSCH
Timeslot duration	15ms
Slotframe length	101 slots
Routing protocol	RPL
Routing metric	ETX
Hardware	AT86RF231 radio STM32F103REY
d_{max} **(frequency reuse range)**	6 hops

TABLE II
PARAMETERS FOR THE EXPERIMENTS

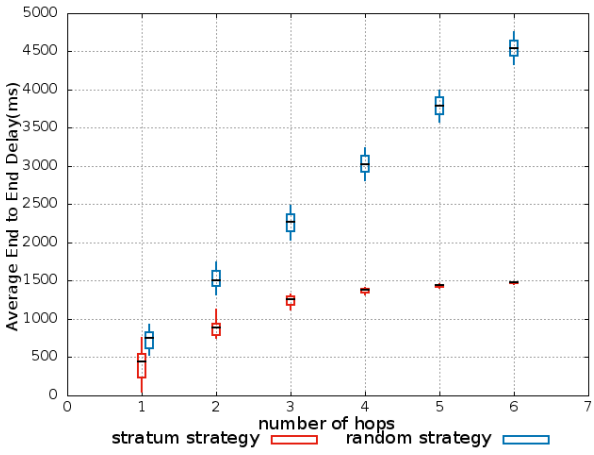

Fig. 4. End to end delay Vs. number of hops

- PDR (packet delivery ratio): ratio of packets actually delivered to the border router;
- E2E delay (end-to-end delay): the delay between the packet generation and its reception by the border router. Only delivered packets are considered;

We run 10 experiments and plot the boxplots for each dataset.

A. End-to-end Delay

We first measure the end to end delay of the random and stratum strategies (Figure 4). For the nodes which are far from the border router, the end to end delay increases: the packets have to be relayed by intermediary nodes, increasing mechanically the delay. For nodes which are 6 hops far from the border router, the delay of the random strategy may reach almost 4.5 seconds.

In particular, we can verify the delay of the random strategy is proportional to the hop length, validating the equation 4 in section III-A. In other words, the delay is on average $15ms * \frac{SF_{length}}{2} * depth$ if we reserve one outgoing cell for each hop ($= 1500ms$ for nodes which are two hops far from the border router).

On the contrary, the delay of the stratum strategy is not linear: it is rather logarithmic. Indeed, the delay for each hop is directly given by the block length. More precisely, the end-to-end delay is the cumulative sum of the block lengths from the source to the border route. According to our schedule division, a packet forwarded along a path of l hops has a delay equal to $\sum_{k=1}^{l}(\frac{1}{2})^k * SF_{length}$.

We can verify that all the packets are delivered during one slotframe (1,500ms), whatever the distance to the sink is. Let's focus on the nodes which are 2 hops far from the border router. They must pick their timeslot in the interval $[C_{min}, C_{max}] = [25, 50]$. Thus, the average delay is equal to 62 timeslots from the end of the slotframe. This finally gives approximatively $62 * 15 = 900\ ms$, as the figure illustrates.

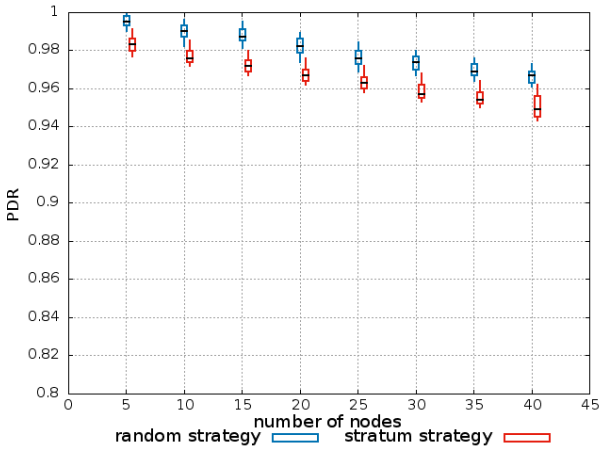

Fig. 5. Packet delivery ration Vs. number of nodes

B. Scalability

We also measured the packet delivery ratio when having an increasing number of nodes (Fig. 5). We selected topologies so that we maintain the density constant, increasing the network diameter to evaluate more the scalability than the robustness to high densities. We increased the number of nodes from 5 to 40 with a pitch of 5.

The larger the network is, the lower the packet delivery ratio is: more collisions may arise. The random strategy aims at minimizing the probability of collisions by selecting different timeslots: the PDR is consequently the largest. However, the stratum strategy only slightly impacts the reliability. While we increase slightly the number of collisions because we select a timeslot randomly inside a block, we still achieve a reliability of 95% even with 40 nodes. We consequently divide the maximum end to delay by 4 in the worst case, with a negligible impact on the end-to-end packet delivery ratio.

C. Traffic scalability

We finally increased the traffic pressure to quantify its impact (Figure 6). We represented the boxplot for the packet delivery ratio in function of the inter packet time for data packets generated toward the border router.

More traffic (lower inter packet time) means a lower PDR. We can notice that stratum strategy creates more collisions, explaining the decrease of PDR. The random strategy is more robust to large traffic conditions: the stratum strategy is rather designed for low traffic and low duty cycle ratio scenarios.

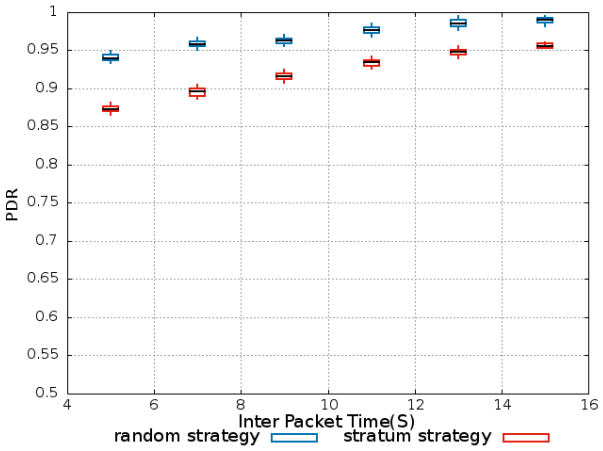

Fig. 6. Impact of the traffic intensity

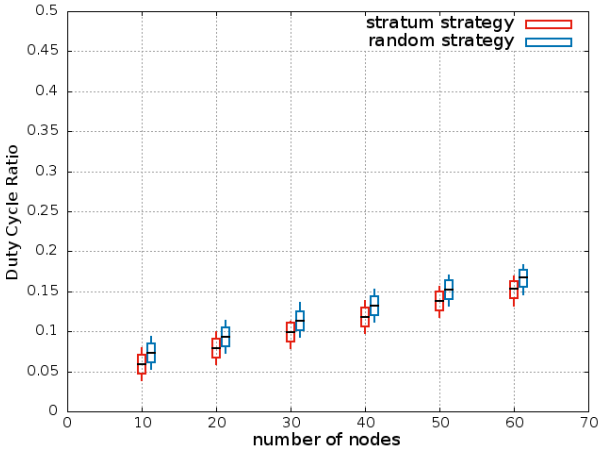

Fig. 7. Energy savings – Average Duty Cycle Ratio

D. Energy Savings

The stratum scheduling algorithm is able to provide a low network duty cycle ratio (DCR), computed as the percentage of timeslots during which a node is active (transmitting or receiving data). We can verify that both the random and the stratum strategies achieve a vey low duty cycle ratio (Fig. 7), always below 2% even with 60 nodes. Obviously, the ratio of active timeslots increases when the network has more traffic to forward.

VII. Conclusion

The emerging industrial IoT is based on a standard communications stack. IEEE ratified the TSCH-based IEEE802.15.4e MAC to support real-time traffic, with deterministic medium access.

In this paper, we studied in depth the end-to-end delay provided by a TSCH MAC layer. We propose a distributed scheduling based on stratums: the end-to-end delay is upper bounded by the slotframe size, whatever the length of the path to the border router is. This strategy is much more efficient than a random distributed scheduling to upper bound the delay, independent of the hop length, and to deal efficiently with retransmissions caused by unreliable links.

In the future, we plan to propose mechanisms for OTF to predict accurately the number of cells it requires for both sporadic and periodic traffic. We also plan to adapt centralized algorithms to enable local reconfigurations of the schedules when the traffic / topology changes.

Acknowledgment

This research was supported by the ICube (University of Strasbourg) project SemSeN .

References

[1] IEEE Std 802.15.4e-2012: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer (2012)

[2] Palattella, M., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L., Boggia, G., Dohler, M.: Standardized protocol stack for the internet of (important) things. IEEE Communications Surveys Tutorials **15** (2013) 1389–1406

[3] 6tisch: Ipv6 over the tsch mode of ieee 802.15.4e wg. charter-ietf-6tisch-01-00 (2013)

[4] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.P., Alexander, R.: Rpl: Ipv6 routing protocol for low-power and lossy networks. rfc 6550, IETF (2012)

[5] Wang, Q., Vilajosana, X.: 6tisch operation sublayer (6top) interface. draft, IETF (2015)

[6] Pister, K., Doherty, L.: Tsmp: Time synchronized mesh protocol. In: Parallel and Distributed Computing and Systems. (2008)

[7] Tsitsiklis, J.N., Xu, K.: On the power of (even a little) centralization in distributed processing. In: ACM SIGMETRICS. (2011) 161–172

[8] M.R. Palattella, et al.: On optimal scheduling in duty-cycled industrial iot applications using IEEE802.15.4e TSCH. Sensors Journal, IEEE **13** (2013) 3655–3666

[9] Tinka, A., Watteyne, T., Pister, K.: A decentralized scheduling algorithm for time synchronized channel hopping. In: ICST Transactions on Mobile Communications and Applications. (2010)

[10] Accettura, N., Palattella, M., Boggia, G., Grieco, L., Dohler, M.: Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things. In: WoWMoM. (2013)

[11] Wan, C.Y., Eisenman, S.B., Campbell, A.T., Crowcroft, J.: Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In: ACM SenSys. (2005) 116–129

[12] Dujovne, D., Grieco, L.A., Palattella, M.R., Accettura, N.: 6tisch on-the-fly scheduling. draft, IETF (2015)

[13] Chen, Z., Wang, C.: Use cases and requirements for using track in 6tisch networks. draft, IETF (2015)

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

[15] Yigit, M., Incel, O.D., Gungor, V.C.: On the interdependency between multi-channel scheduling and tree-based routing for WSNs in smart grid environments. Computer Networks **65** (2014) 1 – 20

[16] Phung, K.H., Lemmens, B., Goossens, M., Nowe, A., Tran, L., Steenhaut, K.: Schedule-based multi-channel communication in wireless sensor networks: A complete design and performance evaluation. Ad Hoc Networks **26** (2015) 88 – 102

[17] Rhee, I., Warrier, A., Aia, M., Min, J., Sichitiu, M.: Z-mac: A hybrid mac for wireless sensor networks. IEEE/ACM Transactions on Networking **16** (2008) 511–524

[18] De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. Wireless Networks **11** (2005) 419–434

[19] Shi, Y., Hou, Y.T., Liu, J., Kompella, S.: How to correctly use the protocol interference model for multi-hop wireless networks. In: ACM MobiHoc, New Orleans, LA, USA (2009) 239–248