Optimized Scheduling for Time-Critical Industrial IoT

Keoma Brun-Laguna, Pascale Minet, Yasuyuki Tanaka

Optimized Scheduling for Time-Critical Industrial IoT

Keoma Brun-Laguna[1], Pascale Minet[2], Yasuyuki Tanaka[3]
Inria, Paris, France
Email: [1]keoma.brun@inria.fr, [2]pascale.minet@inria.fr, [3]yasuyuki.tanaka@inria.fr

Abstract

The main requirement of Industrial Internet of Things (IIoT) is reliability with a targeted reliability of above 99.999%. Then come latency and energy-efficiency requirements. The next step for the IIoT is to target time-critical applications. Even if IIoT technologies are now adopted worldwide, challenges remain and some of the limits of the technologies are still not fully understood. In this paper, we address TSCH-based (Time Slotted Channel Hopping) Wireless Sensor Networks and study their latency and lifetime limits under real-world conditions. We compute theoretical bounds on the end-to-end latency in a perfect radio environment and then in real deployments with unreliable links. We compare the performance of different scheduling algorithms and evaluate the impact of unreliable links on end-to-end latency and network lifetime, by means of simulations using traces collected from real deployed TSCH networks.

Index Terms

IIoT, Industrial IoT, TSCH, Latency, Reliability, Scheduling, Network Lifetime, IEEE802.15.4e, 6TiSCH.

I. Introduction

The Internet of Things (IoT) is a technology concept that is easy to deploy and brings sustainability and economic growth. It is a major component of the Industry 4.0 [1], where machines are capable of sensing their physical environment, taking decentralized decisions and collaborating with other devices or users. Periodic monitoring applications are the most common, where data are periodically collected from sensors. But current solutions do not provide guarantees on latency and are thus not suitable to address applications that require time-critical data delivery. We need to know how far a solution is from the optimal. Does-it worth trying to improve the network performance? What are the trade-offs? Those are the main issues addressed in this paper.

This paper is organized as follows. In Section II, we give a brief state of the art about transmission scheduling in a Time Slotted Channel Hopping (TSCH) network specified in the latest IEEE 802.15.4 standard [2]. In Section III, we define the two scheduling problems addressed and describe the approach proposed for comparative performance evaluation. Section IV provides theoretical bounds for latency and network lifetime, whereas results of the performance evaluation are reported in Section V both in a canonical case and in a more realistic case with unreliable links. Finally, Section VI concludes the paper.

II. Related Work

In this paper, the chosen MAC layer for the multi-hop wireless mesh network supporting time-critical applications is the IEEE802.15.4-2015 standard [2] with the TSCH mode. Using the TSCH mode allows both low duty cycle ($<1\%$) and high reliability ($>99.999\%$). In TSCH, the scheduling consists in allocating cells to nodes, where a cell is defined by a pair (slot offset, channel offset). The more cells a node has, the more opportunities it has to transmit or receive. Although scheduling has a strong impact on latency, reliability and energy consumption, it is not defined in IEEE802.15.4-2015.

A. Scheduling algorithms

In their survey [3], Teles Hermeto et al. classify the scheduling algorithms according to their paradigms (i.e. centralized or distributed) and optimization goal (e.g. low latency or reliability). They point out that centralized schedules are ideal for static topologies with periodic traffic, whereas distributed scheduling is more suited for mobile topologies and traffic that is not determined in advance.

The IETF 6TiSCH working group defined: *1) Neighbor-to-Neighbor Scheduling*: each node exchanges messages with its neighbors to allocate/deallocate cells between each other; *2) Per Flow Scheduling*: nodes allocate cells along a path from the flow source to its destination; *3) Id-based Scheduling*: nodes decide which cells to allocate according to an identifier. This identifier is either this from the sender or from the receiver. When two nodes want to communicate, they choose one of the two identifiers and translate it into cell coordinates (i.e. slot offset and channel offset). How to translate a unique identifier into cell coordinates is not defined. The only constraint is to make sure the mapping to a cell coordinate is unique. This approach presented in Orchestra [4] is now being merged in the 6TiSCH Minimal Scheduling Function (MSF). It does not require any wireless communication and negotiation, but may lead to scheduling collisions.

One pioneer centralized scheduling algorithm over TSCH is Traffic Aware Scheduling Algorithm (TASA) [5]. TASA is built for periodic data gathering. TASA starts to allocate cells to the most loaded node and uses matching and coloring heuristics. TASA guarantees a schedule of minimal length but does not provide any guarantees in terms of reliability, mainly because it assumes perfect links and no retransmissions.

Gaillard et al. extended TASA by adding extra cells for retransmissions. They also proposed Kausa, a KPI-aware scheduling function [6] that guarantees per flow QoS, even if several applications share the same network.

Khoufi et al. [7] propose a debt-based scheduler where the node with the highest amount of traffic to transmit is scheduled first. We reuse this concept in Section III-A, taking retransmissions into account.

B. The Scheduling Function in the 6TiSCH stack

The IETF 6TiSCH working group is standardizing Minimal Scheduling Function (MSF) [8], which operates with the 6TiSCH minimal configuration defined by RFC 8180 [9]. MSF uses a combination of Neighbor-to-neighbor Scheduling and Id-base scheduling that may lead to scheduling collisions. MSF defines how a node joining the network behaves and how the the distributed schedule adapts to application and routing changes as well as scheduling collisions. More generally, any Scheduling Function (SF) in 6TiSCH decides which cells to allocate/deallocate locally, and then triggers a 6top Protocol (6P) Transaction, which is a series of messages a node and its neighbor exchange to negotiate schedule modification. If both nodes agree, they modify their schedule.

C. Evaluation of end-to-end latency and network lifetime

The energy consumption is directly linked to the number of cells used, because the main source of consumption is the radio. The latency depends on how the schedule is built. As the schedule is typically done as a function of the amount of data to transfer and link quality (how many retransmissions needed), if those parameters are fixed and known in advance, we can estimate latency (as done in Khoufi et al. [7]) and energy consumption (as done in Vilajosana et al. [10]).

Several studies report latency results for multi-hop TSCH networks. However, there is still a lack of understanding about how TSCH behaves in different environments and what are the trade-offs between reliability, latency and network lifetime.

III. Scheduling problem and approach proposed

We first define the two addressed scheduling problems, and then show how to compare scheduling algorithms.

A. Scheduling problem statement

1) IIoT Application:
We focus on data gathering applications where data generated at a sampling rate s_i by sensor nodes, denoted as N_i with $i = 1 \cdots n$, are transmitted, through several hops if needed, to a sink denoted as N_0. The set of sensor nodes is denoted as $Nodes$. There is no data aggregation on sensor nodes. There is no fragmentation of data samples: each data sample is transmitted in a single frame. Applications express their requirements with regard to *1) the end-to-end latency required*, denoted as L, which is the maximum acceptable time elapsed between data generation by any sensor node and its delivery to the sink; *2) the end-to-end reliability required*, denoted as R, which is the minimum delivery probability to the sink of data generated by any sensor node; *3) the required network lifetime*, denoted as E, which is the time up to the first failure of a sensor node due to battery exhaustion. The goal is to minimize the energy consumption of each sensor node, while providing the services requested by the application.

2) Radio environment:
Sensor nodes are deployed in a given environment characterized by the deployment area, sensor node density (i.e. average number of neighbor nodes per sensor node) and radio propagation features (multipath fading, external interference, link quality and their dynamics) [11]. Each wireless link l, has an intrinsic reliability evaluated by the probability P_l of a successful transmission over this link. Each sensor node has a single radio interface.

The routing protocol, RPL, is in charge of maintaining a routing tree rooted at node N_0 (i.e. the sink) and covering each sensor node N_i, with $1 = 1 \cdots n$.

3) Optimal scheduling problems:
In the following, **we only consider schedules that are periodic, traffic-aware, valid and conflict-free, do not provide spatial reuse and schedule the maximum number of transmissions per message over a given link and for a given flow**. We define a *flow* as a set of successive messages having the same source address, destination address and QoS (Quality of Service) parameters (e.g. end-to-end reliability and latency).

A Traffic-aware schedule assigns a number of cells to each sensor node which is sufficient to transmit successfully all its data to its parent in the routing tree. *A Valid schedule* ensures that no node is involved as transmitter or receiver in more than one transmission in the same slotOffset. *A Conflict-free schedule* ensures that no node involved as transmitter or receiver in a transmission T_1, is also one-hop away from a transmitting or receiving node in a transmission $T_2 \neq T_1$ scheduled in the same slotOffset and channelOffset. There is *No spatial reuse*: in any cell there is at most one transmission scheduled. Such schedules are more robust with regard to dynamic radio environment. In addition, the schedule systematically schedules the *Maximum number of transmissions per message, per link and per flow* in order to reach the targeted end-to-end reliability.

Property 1. *All the scheduling algorithms considered in this paper avoid conflict between simultaneous transmissions in the same slotOffset by assigning them different channelOffsets. If the number of simultaneous transmissions in the same slotOffset exceeds the number of available channels, the scheduler tries the next slotOffset.*

We define two optimal scheduling problems, which are the two extremes of the trade-off between end-to-end latency and network lifetime.
- The fastest schedule: aims at minimizing the worst case end-to-end latency L, while meeting the requested end-to-end reliability R.
- The longest lifetime schedule: aims at minimizing the energy consumed, while meeting the requested end-to-end latency L and the requested end-to-end reliability R.

The choice between these two scheduling problems is application-dependent. Some applications prefer a short latency even at the cost of a shorter lifetime, whereas others prefer a longer lifetime due to the battery-replacement cost.

B. The approach proposed

1) A per-flow approach: Since the requirements expressed by the IIoT application may be specified per type of flows (e.g. event-triggered or periodic), even per flow, we adopt a per-flow approach: first when we compute the maximum number of retransmissions per link and per flow, and then when we schedule all the transmissions needed to deliver a message to the sink. Without loss of generality, we assume that each message is labeled with its flow tag.

2) Local selection of the message to transmit: Each sensor node selects for transmission the message with the highest priority. We distinguish two levels of prioritization:
- *inter-flow prioritization*: the message belonging to the flow having the highest priority is selected first.
- *intra-flow prioritization*: within a same flow, the oldest message is selected first. All messages are timestamped with the generation time of the oldest data they contain.

3) Canonical case study versus real deployment: Most scheduling algorithms are evaluated in a perfect radio environment where the reliability of each link is equal to 100% and the routing tree never changes. We call such a case the canonical case. This necessary step is not sufficient to assess the correctness of the tested algorithms and their performance in terms of latency, reliability and energy consumption. That is why, in this paper we first study the canonical case and then we consider unreliable links which may generate message losses. In both cases, we use connectivity traces collected from real network deployments as explained in Section V-A.

4) Estimating Retransmissions: A simple way to determine for any given flow, the maximum number of retransmissions per link taking into account the unreliability of each link, consists in requiring the same targeted reliability for each link, that is $R^{1/h}$, if the flow considered has to travel h hops to reach the sink. Let M_i^j denote the maximum number of transmissions for any message of the flow originated at N_i, on the link from node N_j to its parent:

$$1 - R^{1/h} \geq (1 - P_j)^{M_i^j} \tag{1}$$

$$M_i^j = \lceil \frac{Log(1 - R^{1/h})}{Log(1 - P_j)} \rceil \tag{2}$$

Let us consider the routing tree depicted in Figure 1, where the number written under any link $i \rightarrow parent(i)$ gives the success probability P_i of a transmission on this link. The numbers written above links give the maximum number of transmissions per flow for each visited link, needed to reach the targeted end-to-end reliability $R = 0.999$. These numbers are written in the color of the flow origin.

5) Scheduling algorithms evaluated: To schedule a flow originated at any sensor node N_i, any *cascading scheduler* assigns the number of cells needed per message successively on each node visited, from N_i to the sink. Cascading schedulers differ by the order in which they schedule flows. This order is given by the $Weight$ of nodes generating these flows. The node's $Weight$ can be its load, its depth, the number of transmissions to reach the sink, or its debt.

A Load-based scheduler schedules first the flows of the node with the highest load. The load of a node is computed as the sum of the number of cells needed by the node to send and receive. Figure 2 illustrates the schedule of the flows depicted in Figure 1. The flow is depicted with the color of its origin node. Since $Load(B) = 21$, $Load(C) = 22$ $Load(D) = 12$, $Load(E) = 22$ and $Load(F) = 9$, the flows are scheduled in the order C, E, B, D, F. Since Load-based uses 26 slots, which is the minimum number of slots according to Property 2, it is optimal for this configuration.

A generalized Depth-based scheduler schedules first the flows of the node requiring the highest number of transmissions to transmit a single frame originated at this node up to the sink. It can be considered as a generalization of the depth-based algorithm to an unreliable environment.

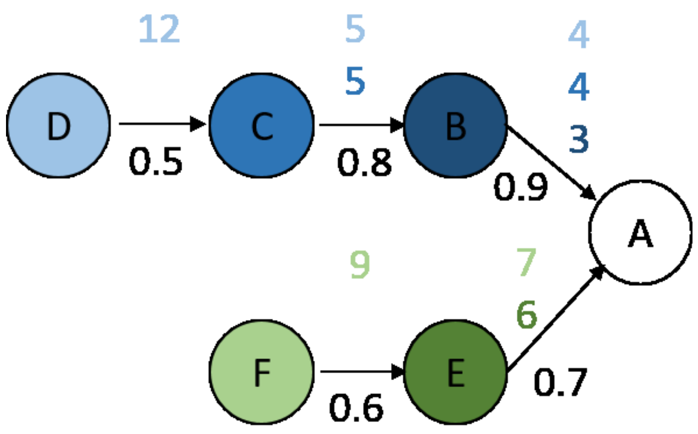

Fig. 1: Example of a routing tree with the PDR of each link.

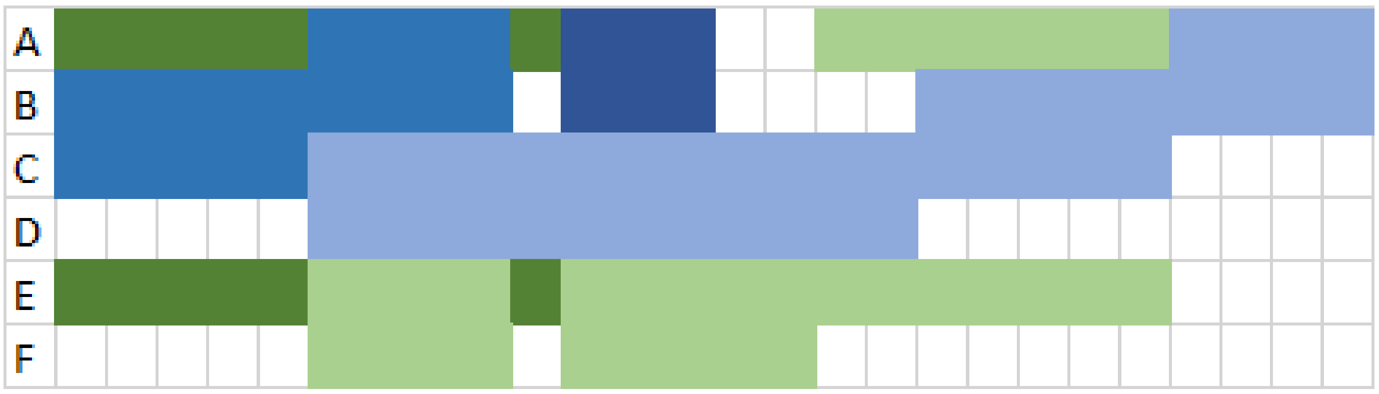

Fig. 2: Load-based schedule.

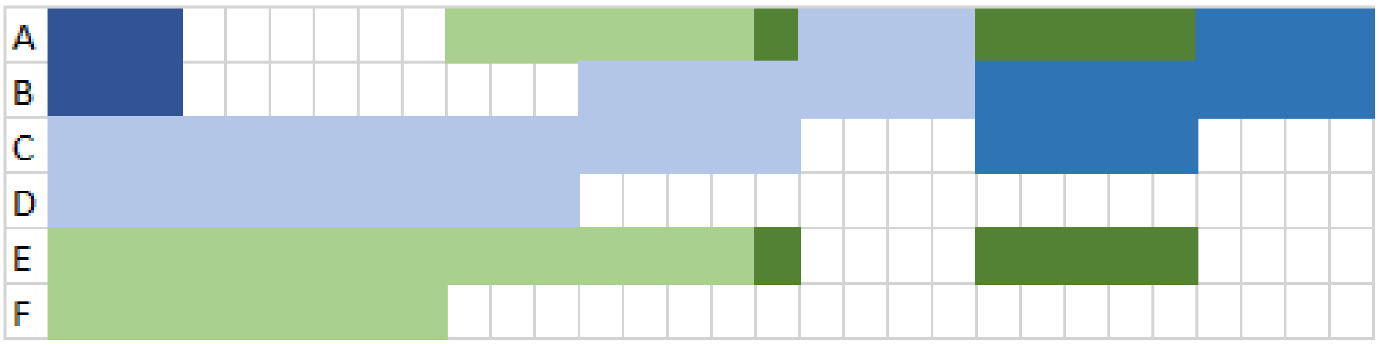

Fig. 3: Generalized Depth-based schedule.

As depicted in Figure 3, the generalized Depth-based schedules flows in the order D, F, C, E and B, leading to 30 slots, which is 4 slots more than Load-based. Hence, this scheduler is not optimal, this contradicts an intuitive idea. By scheduling first the flow requiring the highest total number of slots, the other flows tend to be assigned later slots on the loaded nodes, leading to a larger schedule size

A total Transmission-based scheduler schedules first the flows of the node with the highest total number of transmissions up to the sink, taking into account both local and forwarded traffic. This is a generalization of the previous scheduling. In the example considered, we get $TotalTrans$ = 11, 18, 21, 13 and 16, for B, C, D, E and F, respectively, leading to the same schedule as the generalized Depth-based scheduler. Hence, the total Transmission-based scheduler is not optimal.

A Debt-based scheduler schedules first the flows of the node with the highest debt. The debt of a node is equal to the maximum between the total number of transmissions needed to reach the sink and the load of the node. Intuitively, this schedule schedules first the node with the strongest constraint imposed either by the node load or by the total number of transmissions the network has to perform for the frames visiting this node. In the example considered, we get $Debt(B) = 21$,

$Debt(C) = 22$, $Debt(D) = 21$, $Debt(E) = 22$ and $Debt(F) = 16$, leading to the same schedule as the Load-based scheduler.

Algorithm 1 Cascading Scheduler.

```
Require: the $Weight$ of each sensor node
  $M^{i}_{node}$ the max. number of transmissions from $i$ to its parent, for each msg originated from $node$
Ensure: Allocate cascading cells to flows,
  starting with the flow originated at the largest $Weight$ node
  $Nodes \leftarrow$ sensor nodes set ordered by decreasing $Weight$
  for $node \in Nodes$ do
    % Schedule each flow originated at this node
    for each $flow$ originated at $node$ do
      % Consider the flows by decreasing priority order
      % $NodeOffset =$ last slot used by flow on its origin
      $NodeOffset \leftarrow 0$
      for each $message$ of $flow$ do
        % Assign cascading cells for $message$
        $slotOffset \leftarrow NodeOffset$
        $child \leftarrow node$
        while $child! = sink$ do
          $M^{child}_{node}$ transmissions are required per $message$
          for each requiredTransmission do
            $parent \leftarrow parent(child)$
            AssignFirstAvailCell($child$,$parent$) $\geq$ slotOffset
            $slotOffset \leftarrow assignedslotOffset$
          end for
          if $child$ is the origin of the flow then
            %Next message assigned from this slotOffset
            $NodeOffset \leftarrow slotOffset$
          end if
          $child \leftarrow parent$
        end while
      end for
    end for
  end for
```

All these schedulers proceed as described in Algorithm 1. They first compute the $Weight$ of each sensor node. Then, they sort the nodes by decreasing $Weight$ order. Starting with the highest $Weight$ node, for each message generated by this node, these scheduling algorithms allocate the cells the node requires to transfer this message to the sink, taking into account the number of transmissions required for each message. They start with the first available cell on both the considered node and its parent, they proceed in a time (i.e. slot_offset) increasing manner. They repeat this for each message generated by the node considered. As a consequence, cells are allocated in a cascading way for each message originated from the node considered, as illustrated in Figure 2 and Figure 3. This processing is repeated for each sensor node, until all sensor nodes are scheduled.

Notice that if several nodes have the same weight, the farthest node from the sink is scheduled first. These flows are scheduled by decreasing priority order. For each flow considered, all the cascade of any message generated by this flow is scheduled before scheduling the next message of this flow.

Cascading schedulers have a complexity in $O(nfmh)$ where n denotes the number of nodes, f the number of flows generated per node, m the number of messages generated per flow and h is the maximum path length. Such schedulers present many advantages. *1) They are able to provide a QoS customized per flow. 2) They are traffic-aware*, since each sensor node is assigned a number of cells depending on its traffic (i.e. traffic generated + traffic forwarded). *3) They minimize the buffer size needed on any sensor node*, because a message received from a child is promptly transmitted to its parent. *4) They minimize the end-to-end latency*, since they minimize the number of time slots separating the time slot of the first message transmission on the origin node of the flow, from the time slot of message delivery to the sink. *5) They maximize the end-to-end reliability*, because they avoid conflicts by applying Property 1.

6) Energy consumption: We use the results from the radio chip (LTC5800-IPM) datasheet to determine the charge consumed by the mote in each timeslot. The values are listed in Table I, where $1\ C = 1\ As$. We assume that each node has a 2821.5 mAh capacity (a pair of Energizer L-91 AA batteries).

Type of slot	Charge
TxDataRxAck	54.5 μC
RxDataTxAck	32.6 μC
IdleListen	6.4 μC
Sleep	0 μC

TABLE I: Charge consumed by a mote per timeslot.

IV. Theoretical bounds

A. Lower Bound on the number of slots

We first introduce some additional scheduling notations: M_i^j is the number of transmissions made by any sensor node N_j to transmit a message originated at node N_i. $Gen(N_i)$ denotes the number of messages generated by sensor node N_i. $Desc(N_i)$ is the set of descendants of N_i in the routing tree, whereas $Desc^+(N_i) = Desc(N_i) \bigcup \{N_i\}$ is the set of descendants of N_i including itself.

The $Load$ of any sensor node N is equal to the number of cells needed by N to transmit and receive messages.

$$\begin{aligned} Load(N) = & \sum_{D \in Desc^+(N)} M_D^N \times Gen(D) \\ + & \sum_{C=child(N)} \sum_{D \in Desc^+(C)} M_D^C \times Gen(D) \end{aligned} \tag{3}$$

$$Load(sink) = \sum_{C=child(sink)} \sum_{D \in Desc^+(C)} M_D^C \times Gen(D) \tag{4}$$

The total number of transmissions needed to deliver the data generated by the sensor node N to the sink is:

$$Ttrans(N) = Gen(N) \times \sum_{X \in Path(N)} M_N^X \tag{5}$$

We also define $Ttrans$ as the total number of transmissions needed to deliver the data from all sensor nodes.

$$Ttrans = \sum_{N \in Nodes} Ttrans(N) \tag{6}$$

The schedule of minimum length (i.e. minimum number of slots used) should meet the following constraints:
- *Constraint C1*: it should allow the sink to receive all messages sent to it. The number of slots should be higher than or equal to $Load(sink)$.
- *Constraint C2*: it should schedule $Ttrans$, all the transmissions needed to collect data generated on each sensor node. According to Property 1 and our assumptions, the number of transmissions that can be scheduled is equal to the number of cells available in the schedule. The number of cells is equal to the number of slots multiplied by $NumChannels$, the number of channels used by the TSCH network. Hence, the number of slots should be higher than or equal to $\lceil \frac{Ttrans}{NumChannels} \rceil$.
- *Constraint C3*: it should allow the last message transmitted by any sensor node to reach the sink. Since the last slot used by any sensor node N is a Tx slot. This slot is higher than or equal to $Load(N)$. Since, we do not know a priori to which flow this message belongs, it may be any flow visiting N, we have to take into account the weakest condition generated by these flows. Hence, to allow this last message transmitted by N to reach the sink, the number of slots in the schedule should be higher than:

$$NLoad(N) = Load(N) + \min_{D \in Desc^+(N)} \sum_{X \in Path(parent(N))} M_D^X \tag{7}$$

Notice that in a real deployed TSCH network, unlike the perfect environment with no retransmission, the most loaded node is not necessarily the sink or one of the first-hop nodes. It can be a node with many retransmissions, and not necessarily with many descendants.

Property 2. *A bound on the minimum slotframe size is:*

$$MinLength \geq Max\{Load(sink), \lceil \frac{Ttrans}{NumChannels} \rceil, \max_{N \in Nodes} NLoad(N)\} \tag{8}$$

Property 2 tells that no schedule can obtain a lower number of slots than (8), it does not mean that such a schedule exists.

Property 3. *With our assumptions, the Load-based scheduler is optimal for any configuration meeting* $Load(N) < Load(parent(N))$, $\forall N$ *at least 2-hop away from the sink, and the most loaded node is 1-hop away from the sink.*

Proofs are omitted for space reasons.

B. Latency Upper Bound

With our assumptions given in Section III) and assuming that all flows have the same priority and there is no parent change in the routing tree, the worst data acquisition time is right after the last slot assigned to this flow on its origin node starts. The first opportunity to transmit these data arrives $SlotframeSize - 1$ slots later. Then, in the worst case, these data need the maximum number of retransmissions per hop to reach the destination. Knowing $MinLength$ the minimum number of slots to schedule the application flows, the smallest maximum end-to-end latency provided to the IIoT application is:

$$Latency \leq (SlotframeSize - 1 + MinLength) \times SlotDuration \tag{9}$$

C. Lifetime Bounds

We can evaluate the energy consumption of a node per slotframe by means of energy consumption parameters per slot type given in Table I. Network lifetime can be expressed as:

$$\underset{N \in Nodes}{Min} \frac{Initial\_Energy(N) \times SFDuration}{Average\_Energy\_Consumption(N)} \tag{10}$$

We can increase the lifetime of a mote by adding Idle slots at the end of the slotframe, provided that the number of messages generated during the new slotframe duration remains equal to that obtained with the current slotframe. This reduces the average energy consumption but also increases latency, which highlights the trade-off latency versus network lifetime. The maximum average consumption of a node is given by:

$$\frac{MinLength \times ActiveAvgCharge + SlotIncrease \times IdleCharge}{MinLength + SlotIncrease} \tag{11}$$

Where $ActiveAvgCharge$ is the average charge spent during the minimal slotframe of size $MinLength$, and $IdleCharge$ is the charge spent during one Idle slotframe.

V. Performance results

A. Simulation using traces of deployed networks

1) Trace-based simulation:
Simulation enables rapid performance evaluation of networking protocols. Too simplified radio propagation models often lead to results that are not representative of the real world, whereas complex models require large processing times. Trace-based simulation offers a good trade-off between complexity and real world representativeness. It uses connectivity traces collected from real deployments operating in environments close to this of the application considered.

Connectivity traces consist of the Packet Delivery Ratios (PDRs) measured in an IEEE802.15.4-based low-power wireless network. The PDR of each link is evaluated as the ratio of acknowledged frames over the number of frames sent. PDR is preferred to RSSI, because it is hardware-independent.

Topology changes observed in the real deployment are replayed in the simulator. When the simulation reaches the time of a change in the trace file, the simulation topology is updated (including the neighbor tables and preferred parents). The scheduling task is run every time there is a change in the trace file. The network schedule built is then pushed to the nodes.

2) Simulation parameters:
We selected the 6TiSCH simulator [12], which is a discrete-event simulator written in Python designed for fast prototyping of 6TiSCH. The simulator uses the parameters given in Table II. Each node generates a single flow.

	Parameter	**Value**
Config.	Number of nodes	50
	Maximum distance to the sink	6 hops
	Application data rate	27 bytes every 30 s
	Slot duration	7.25 ms
Required	Minimum end-to-end reliability	99.9%
	Maximum end-to-end latency	5 s
	Minimum end-to-end lifetime	none, then ≥ 1 year
Simulation	Number of simulation runs	100
	Number of slotframes per run	20,000

TABLE II: Simulation parameters.

B. *Canonical Case Study*

To capture the topology of deployed networks, collected traces are injected in the 6TiSCH simulator. In this subsection only, the PDR of radio links is set to 100%. Figure 4 depicts the performance results of the 4 schedulings for 99.9% of the data, and the number in the legend is the maximum value got by each scheduling. For this topology, all scheduling algorithms give a schedule of 49 slots, the minimum size (Property 2). As per Eq. 9, the smallest end-to-end latency is 0.70325s, which is achieved by the Load-based scheduling. Depth-based needs one additional slotframe, because it does not use timestamps for message prioritization (Section III-B2).

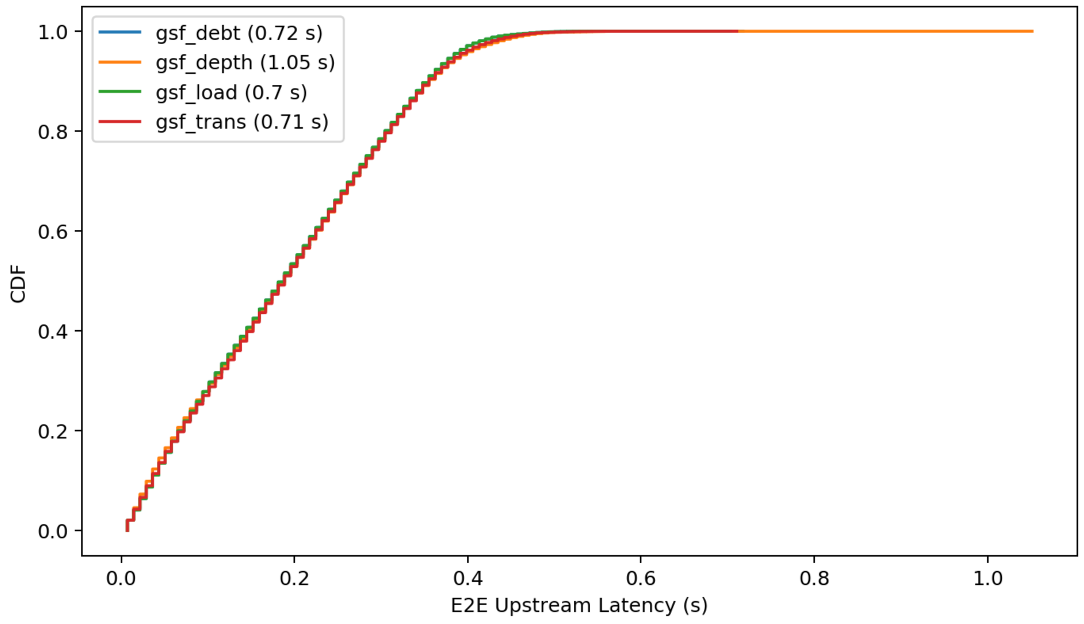

Fig. 4: End-to-end latency in the canonical case.

C. *Unreliable links*

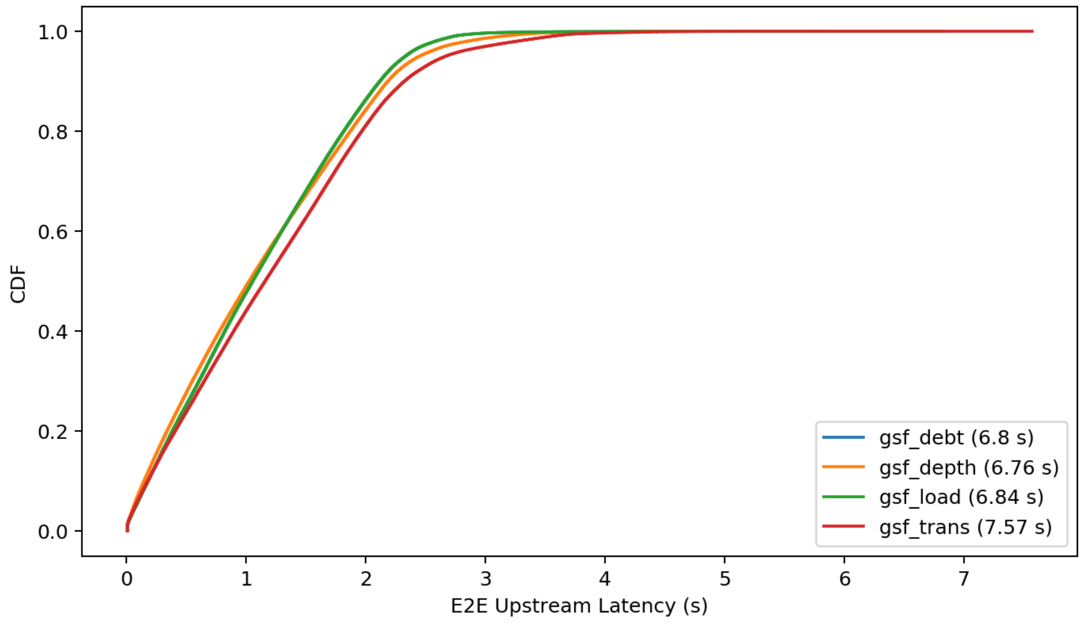

Fig. 5: End-to-end latency with unreliable links.

Links are now unreliable. The 6TiSCH simulator uses the PDR of each link, computed from traces collected from real deployed networks. Only links with a $PDR \geq 0.5$ are used. Figure 5 shows that the unreliability of links degrades network performances. The schedule size ranges from 288 to 371 slots, which is up to 7.57 times higher than in the canonical case. The Load-based scheduling gives the largest probability to have an end-to-end latency ≤ 5 s. If timestamps are not used to prioritize messages, one additional slotframe is needed wrt Eq. 9, leading to a maximum latency of 6.8585 s, representing an increase of 50%.

VI. Conclusion

Time-critical industrial applications have to meet three Key Performance Indicators (KPIs): a maximum acceptable E2E latency, a minimum E2E reliability and a minimum network lifetime. In this paper, we showed how a TSCH network meets such requirements, by means of simulations based on connectivity traces collected from real deployed TSCH networks. We highlighted the best trade-off between E2E latency and network lifetime. We proved that the *Load-based* scheduling is optimal for many configurations. Simulation results showed that it always gives latencies very close to the optimal. They also highlighted the strong influence of message prioritization on end-to-end latencies at high loads. Link unreliability has a large impact on KPIs: for instance, a schedule size up to 6.57 times higher and end-to-end latencies up to 9.66 higher. We will study how to minimize the total number of transmissions per message to reach the sink with the targeted E2E reliability.

References

[1] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *Proceedings of the 2016 49th Hawaii Int. Conf. on System Sciences (HICSS)*, ser. HICSS '16. Washington, DC, USA: IEEE Computer Society, 2016, pp. 3928–3937.

[2] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[3] R. Teles Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," *Computer Communications*, vol. 114, pp. 84–105, Dec. 2017.

[4] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *13th ACM Conf. on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 337–350.

[5] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic Aware Scheduling Algorithm for Reliable Low-Power Multi-Hop IEEE 802.15.4e Networks," in *2012 IEEE 23rd Int. Symp. on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, Sep. 2012, pp. 327–332.

[6] G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois, "Kausa: KPI-aware Scheduling Algorithm for Multi-flow in Multi-hop IoT Networks," in *Ad-hoc, Mobile, and Wireless Networks*, N. Mitton, V. Loscri, and A. Mouradian, Eds. Lille, France: Springer International Publishing, 2016, pp. 47–61.

[7] I. Khoufi, P. Minet, and B. Rmili, "Scheduling transmissions with latency constraints in an IEEE 802.15.4e TSCH network," in *VTC 2017 - IEEE 86th Vehicular Technology Conf.*, Toronto, Canada, Sep. 2017.

[8] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," IETF, Internet-Draft draft-ietf-6tisch-msf-00 [work-in-progress], Aug. 2018.

[9] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," IETF, Tech. Rep. RFC8180, May 2017.

[10] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. Pister, "A Realistic Energy Consumption Model for TSCH Networks," *Sensors Journal, IEEE*, vol. 14, pp. 482–489, 02 2014.

[11] K. Brun-Laguna, P. Henrique Gomes, P. Minet, and T. Watteyne, "Moving Beyond Testbeds? Lessons (We) Learned about Connectivity," *IEEE Pervasive Computing, Special Issue on Beyond Testbeds: Real-World IoT Deployments*, 2018.

[12] E. Municio, G. Daneels, M. Vicinic, S. Latre, J. Famaey, Y. Tanaka, K. Brun-Laguna, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies*, 2018.