On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks

Maria Rita Palattella, *Member, IEEE,* Thomas Watteyne, *Senior Member, IEEE,* Qin Wang, *Member, IEEE,* Kazushi Muraoka, *Member, IEEE,* Nicola Accettura, *Member, IEEE,* Diego Dujovne, *Member, IEEE,* Luigi Alfredo Grieco, *Senior Member, IEEE* and Thomas Engel *Member, IEEE*.

***Abstract*—In smart factory applications, sensors, actuators, field devices, and supervision systems often require a high degree of reliability and timeliness in information exchange. The Quality of Service provided by the underlying industrial communication network is a key requisite for Quality of Control. In this context, the WirelessHART, ISA100.11a, and IEEE802.15.4e Timeslotted Channel Hopping (TSCH) standards contribute novel protocols to support quasi-deterministic services, based on wireless short-range communication technologies. The recently created IETF 6TiSCH working group binds IPv6 to this market. Within the 6TiSCH architecture, the 6top sublayer manages the way communication resources are scheduled in time and frequency. The On-The-Fly (OTF) bandwidth reservation module plays a complementary role; it is a distributed approach for adapting the scheduled bandwidth to network requirements. This article first describes the OTF module and its interactions with the 6top sublayer. It then presents simulation results of OTF, drawn from a realistic 50-sensor mote multi-hop network that models a small industrial plant. Results show that OTF can attain an end-to-end latency of the order of a second, with over 99% end-to-end reliability. The first real-world OTF implementation in OpenWSN is presented to demonstrate it can easily be added within the 6TiSCH architecture.**

***Index Terms*—IEEE802.15.4e Sensor Networks, Timeslotted Channel Hopping, 6TiSCH Architecture, 6top, On-The-Fly (OTF) Bandwidth Reservation.**

I. Introduction

For a long time, industrial systems have relied on wired networks for robust and predictable communication. These networks connect sensing and control capabilities at different steps in the production process [1], [2]. The installation and maintenance costs of wires in an industrial setting have fueled the introduction of wireless technology. Yet, for wireless networks to be usable in this context, they need to fulfill two fundamental requirements: almost deterministic communication and robustness. These two constraints were first met by the Time Synchronized Mesh Protocol (TSMP) [3]. In TSMP, time-scheduling enables precise end-to-end delay calculation, while frequency hopping reduces packet loss due to interference and multi-path fading.

To guarantee the interoperability of wireless sensor motes, the core concepts of TSMP were introduced into WirelessHART (2008) [4], ISA100.11a (2011) [5] and IEEE802.15.4e (2012) [6]. While the first two standards define a complete protocol stack, the latter only defines the physical and three distinct Medium Access Control (MAC) layers: Low Latency Deterministic Network (LLDN), Timeslotted Channel Hopping (TSCH), and Deterministic and Synchronous Multi-channel Extension (DSME). Among these, the TSCH mode is the one facilitating multi-hop operation, and is able cope efficiently with external interference and multi-fading in distributed sensing and process control applications. The clean layering allows IEEE802.15.4e TSCH to fit under an IPv6-enabled protocol stack for low-power wireless networks. This architecture was recently proposed by the IETF 6TiSCH Working Group (WG)[1].

The IETF 6TiSCH WG, created in October 2013, is playing a key role introducing IPv6 [7] to industrial standards, and fostering the convergence between Operational Technology (i.e. industrial networks) and Information Technology (i.e. computer networks) [8]. The goal of the 6TiSCH architecture [9] is an IPv6 multi-link subnet, including several Low Power and Lossy Networks (LLNs) connected by a high-speed backbone, through a number of Backbone Routers (BBRs). 6TiSCH "glues" together the IEEE802.15.4e TSCH MAC layer with an IPv6-enabled upper stack (e.g. 6LoWPAN [10], and the Routing Protocol for Low-Power and Lossy Networks, RPL [11]). TSCH, like TSMP, combines time synchronization with channel hopping: all sensor motes in the network follow a common schedule that specifies for each sensor mote at which time slot and on which channel it can communicate with its neighbors. The IEEE802.15.4e standard does not define how the schedule is built and matched to network traffic requirements.

To bridge this gap, 6TiSCH has defined the 6TiSCH Operation Sublayer (6top) [12], which (de)allocates resources within

M. R. Palattella and T. Engel are with the SnT, University of Luxembourg, Luxembourg (e-mail: maria-rita.palattella, t.engel@uni.lu).

T. Watteyne is with Inria, EVA team, France (e-mail: thomas.watteyne@inria.fr).

Q. Wang is with University of Science and Technology Beijing, China, (e-mail: wangqin@ies.ustb.edu.cn).

K. Muraoka is with Swarm Lab, University of California Berkeley, USA and with Cloud System Research Laboratories, NEC Corporation, Japan (e-mail: k-muraoka@eecs.berkeley.edu).

N. Accettura is with BSAC, University of California Berkeley, USA (e-mail: accettura@eecs.berkeley.edu).

D. Dujovne is with Universidad Diego Portales, Chile (e-mail: diego.dujovne@mail.udp.cl).

L. A. Grieco is with the Dip. di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy (e-mail: a.grieco@poliba.it).

Manuscript received XXX, 2015; accepted YYY, ZZZ.

[1]IPv6 over the TSCH mode of IEEE802.15.4e (6TiSCH), `http://tools.ietf.org/wg/6tisch/charters`.

the schedule, while monitoring performance and collecting statistics. Different bandwidth reservation approaches are supported by the 6TiSCH architecture. A centralized approach is suitable for applications with static traffic patterns. This is the approach taken by industrial wireless standards such as WirelessHART (2008) [4] and ISA100.11a (2011) [5]. For applications with highly dynamic topologies and bursty data transmissions, distributed scheduling is an alternative.

For instance, when a Wireless Sensor Network (WSN) is deployed for continuous real-time monitoring of a given field, it is easier to estimate and tune the amount of traffic that each sensor mote has to deliver over the time and, hence, define a schedule using a centralized approach. Contrariwise, if the WSN is set up to detect an event, sensor motes will be almost silent over the time (except for unavoidable signaling messages) but they will start sending notifications upon the event they are looking for is detected. Under these circumstances, it is not possible to predict in advance the traffic patterns and a distributed approach would be more useful.

In safety-related industrial sensors applications, traffic can be bursty. For instance, gas leak detection [15], [16] must start transmitting at a higher sample rate to reduce the time to locate the leak, to calculate its magnitude, and to estimate the prospective evolution of this failure. Another example is monitoring grain silos to prevent fire or explosions [17]; sensor motes are programmed to change the data capture frequency and wake up secondary sensing devices when certain humidity, temperature, oxygen, or carbon monoxide levels are reached. During these bursts, a high packet loss cannot be tolerated.

On-The-Fly (OTF) bandwidth reservation [13] is a solution for distributed scheduling. Based on a bandwidth allocation algorithm, OTF dynamically matches the scheduled bandwidth between pairs of sensor motes to the actual traffic load. OTF decides when to add/delete resources, and the 6top sublayer then makes the appropriate changes to the TSCH schedule. The range of application domains where OTF can be applied spans from semi-centralized systems (in which a rapid reaction is needed when a surge in bandwidth is detected) and fully distributed systems (where neighbor motes negotiate bandwidth with one another without the intervention of a Path Computation Element, PCE).

Application examples of OTF as a distributed bandwidth reservation module include temporary monitoring deployments for specific magnitude measurement, such as vibration and strain in bridges [18]–[20] where sensor motes reserve bandwidth dynamically according to the number of parameters, sampling rate and aggregate requirements. OTF can be used in industrial processes monitoring applications to take advantage of flexible bandwidth allocation. For example, transport belts in the mining industry [21], [22] require several sensing points such as misalignment switches, speed monitors, belt wear monitors and metal detectors. When the dynamics of the belt changes, it triggers a spike in data generation until a new equilibrium point is reached.

This article describes the 6top sublayer and the OTF bandwidth reservation module, and presents their performance through simulation. This article is not an attempt at answering which approach to use (centralized, distributed, hybrid), but rather shows the performance of OTF, and the first proof of concept, proving OTF can be implemented in low power sensor motes, in real scenarios.

The remainder of this article is organized as follows. Section II provides an overview of the IEEE802.15.4.e TSCH MAC and the 6top sublayer, on which OTF is built. Section III details the OTF mechanism, including the bandwidth allocation algorithm at its core. Section IV describes the 6TiSCH simulator used in this contribution and reports key performance indicators for OTF as well as its preliminary experimental evaluation. Section V concludes this article.

II. 6TiSCH Architecture

The OTF Bandwidth Reservation module is designed for 6TiSCH networks. The 6TiSCH architecture implements the IEEE802.15.4e TSCH link-layer protocol, the 6top sublayer, and the RPL routing protocol. The TSCH MAC and the 6top sublayer are described in more details in this section.

A. IEEE802.15.4e TSCH

IEEE802.15.4e [6] is an amendment to the MAC protocol of IEEE802.15.4-2011 [14]. The Time Synchronized Channel Hopping (TSCH) mode of IEEE802.15.4e yields ultra-high reliability and lower-power operation. All sensor motes in a IEEE802.15.4e network are synchronized. Time is split into time slots, each typically $10ms$ long. Time slots are grouped into a slotframe, which continuously repeats over time. All communication is orchestrated by a "schedule" which instructs each sensor mote what to do in each slot of the slotframe: sleep, transmit (TX) or receive (RX). Channel diversity is achieved by specifying, for each TX or RX slot, a `channel offset`. In consecutive slotframe cycles, the same channel offset translates into a different communication frequency, resulting in "channel hopping". Channel hopping reduces the impact of external interference and multi-path fading, thereby increasing the reliability of the network [23].

The TSCH schedule can be represented by 2-D matrix of width the number of slots in a slotframe, and height the number of available frequencies. Each element in the matrix is called a "cell". A cell is "scheduled" when used either to transmit or receive. Each scheduled cell is an opportunity for a sensor mote to communicate with its neighbor. The schedule is managed through a centralized or distributed scheduling entity; OTF – presented in this article – is a example of the latter. The role of the scheduling entity is to ensure that there are enough communication opportunities to satisfy the communication needs of the applications (amount of traffic load offered to the network, reliability, latency).

The IEEE802.15.4e standard specifies the scheduling mechanism (i.e. how to execute a schedule), but does *not* specify the scheduling policy (i.e. how the schedule is built and maintained). Constructing a schedule is policy-specific. Example policies include a centralized scheduling based on graph coloring techniques [25], [26], and distributed scheduling using gossiping at two-hop neighbors [27]. The scheduling entity requires a method to distribute and install the schedule at each sensor mote.

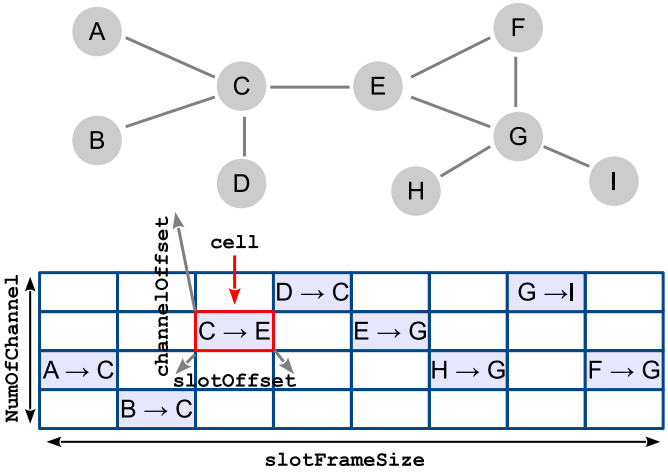

Fig. 1. A simple TSCH schedule.

B. 6TiSCH Operation Sublayer (6top)

The 6TiSCH working group is defining the 6top sublayer as the next upper layer of the TSCH MAC (see Fig. 2). Layers on top of 6top can use it to manage the TSCH schedule, configure monitoring, and collect statistics [12].

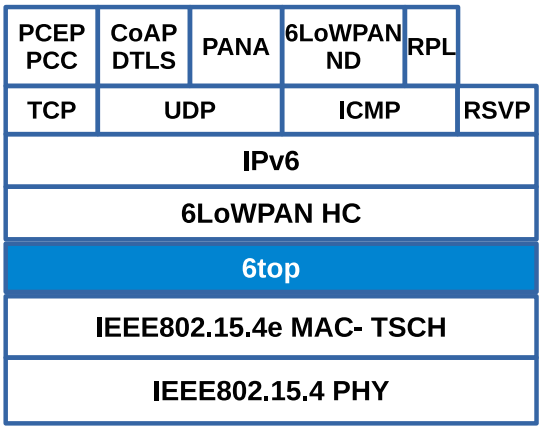

Fig. 2. The 6TiSCH protocol stack.

6top has been designed to be used with several scheduling approaches. With a centralized one, a PCE collects topology and traffic requirements that are used to build a communication schedule, which is then sent to the sensor motes in the network. In a distributed approach, sensor motes compute their own schedule according to local information or by using a distributed resource reservation protocol similar to the Resource Reservation Protocol (RSVP) [24], [27] or Next Steps In Signaling (NSIS) [28].

6top labels each scheduled cell by its `cellType`, which can be either "hard" or "soft". A *hard cell* is a cell that cannot be dynamically relocated[2] by 6top. The cell is installed by 6top at a particular `slotOffset` and `channelOffset` in the schedule. This type of cell is typically scheduled by a PCE. Once installed, only the PCE can move/delete it inside the TSCH schedule. A *soft cell* is a cell that can be relocated by 6top dynamically. This type of cell is typically scheduled by a distributed scheduling entity, e.g. OTF proposed in this paper. Instead of specifying the exact `slotOffset` and `channelOffset`, the scheduling entity indicates how many cells to schedule to a given neighbor, i.e. the bandwidth requirement. With the bandwidth requirement, 6top will choose specific `slotOffset`(s) and `channelOffset`(s) through the *soft cell negotiation procedure* described in [12]. The monitoring process of 6top keeps track of the performance of each of the cells to the same neighbor. If a cell performs significantly worse than the other cells scheduled to the same neighbor, 6top relocates this cell at a different `slotOffset` and `channelOffset` inside the TSCH schedule. The concept of soft cells allows the 6top sublayer deal with the interference efficiently.

Once a TSCH schedule is established, 6top is responsible for feeding the data from the upper layer into TSCH. A 6TiSCH network can transport different types of traffic, possibly for different administrative entities (e.g. equipment or process status data like temperature, pressure, vibration and Heating, Ventilation, and Air Conditioning (HVAC) data in a smart building), with different Quality of Service (QoS) constraints. Since 6top is a sublayer between 6LoWPAN and IEEE802.15.4e TSCH (Fig. 2), the properties associated with a packet/fragment are translated by the upper layer into parameters such as next hop neighbor address, or priority associated with that packet. 6top comes with its own queuing model to satisfy the associated service differentiation.

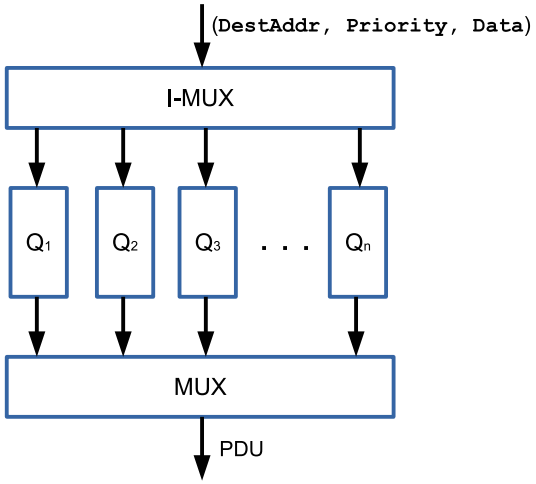

Fig. 3. The 6top queue model.

The 6top queue model is depicted in Fig. 3. Q_i represents a queue, which is either broadcast or unicast, and has an assigned priority. The number of queues is configurable. When 6top receives a packet to forward, the Inverse Multiplexer (I-MUX) module selects a queue where to insert it, according to the priority associated with the packet. The Multiplexer (MUX) module is invoked at each scheduled transmit cell by TSCH. When invoked, the MUX module goes through the queues using the neighbor address attached to the scheduled

[2]We use the term "relocate" to mean that the cell is moved from one `[ slotOffset,channelOffset]` coordinate to a different one in the TSCH schedule.

transmit cell, looking for the best matching frame to send. If it finds a frame, it hands it over to TSCH for transmission.

6top maintains statistics about the performance of scheduled cells. When using a centralized scheduler, this information is periodically sent to the PCE, which continuously adapts the schedule and sends schedule updates as needed. This information can also be used by the objective function of RPL or bandwidth reservation modules.

6top provides a set of commands to support decentralized, centralized and hybrid scheduling solutions. These form an interface an upper layer can use. For bandwidth reservation functions, `CREATE/DELETE.softcell` [12] adds or deletes soft cells. The monitoring process of 6top is responsible for the quality of those soft cells by relocating the cells with significantly low communication performance.

RPL can leverage the statistic information 6top gathers at each node to further optimize routes, subject to the QoS requirements of served applications. This way, soft real time services can be provided over the best effort track by dynamically updating paths, based on a quasi real-time monitoring of the different candidate next hop nodes along a path.

III. On-the-fly Bandwidth Reservation

Most industrial networks require a high level of determinism. Typically, data flows are known a priori, and it is therefore possible to compute the communication schedule in advance, taking into account the QoS level of each flow (amount of data, latency, reliability), while optimizing the use of cells scheduled to limit the energy spent. When flow requirements change (e.g. a sensor node generates a new data flow), the schedule needs to be adapted accordingly. Several approaches are possible, including centralized and distributed ones. The 6top sublayer follows a distributed approach, and allocates soft cells upon request from a higher level reservation protocol module, such as OTF, and makes sure that the requested QoS is constantly met in a hop-by-hop basis. The OTF module presented in this work deals with the best-effort track, which corresponds to the class of traffic with the lowest priority, prone to higher packet loss and variable delay. Consequently, OTF does not guarantee the same compliance with the delay and packet loss requirements offered by other tracks, with different QoS settings.

The OTF module is located on top of the 6top sublayer, and implements a distributed bandwidth allocation algorithm: this module monitors the amount of data being sent to each of the mote's neighbors; when this amount of data becomes too large (resp. too small) compared to the number of cells scheduled to that neighbor, OTF asks 6top to add (resp. delete) cells scheduled to that neighbor. Adding/deleting cells triggers a specific communication process ("negotiation") between the neighbor motes, which results in transmission overhead. Complimentary to the OTF bandwidth estimation algorithm, an allocation algorithm module is implemented before the requests are forwarded to 6top. By adding configurable over-provisioning through a threshold, this allocation algorithm module reduces bandwidth estimation algorithm instabilities such as cells continuously added/deleted. However, there is a need to find the optimal trade-off between over-provisioning too many cells (which implies motes would switch their radios on unnecessarily) or too few cells (which translates in an unnecessary increase of 6top's communication overhead).

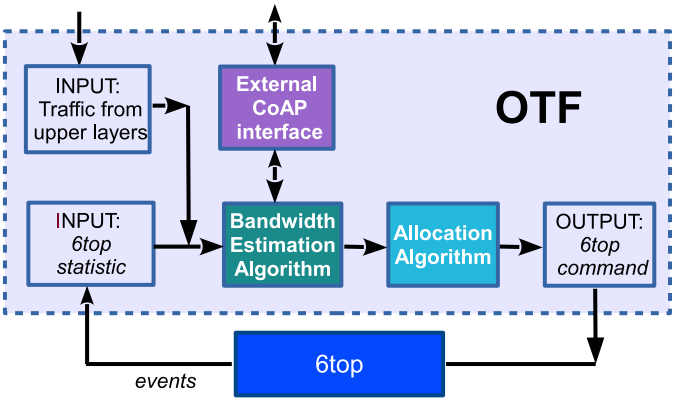

Fig. 4. Main components of the OTF module.

As shown in Fig. 4, a network administrator can fine-tune OTF parameters using the Constrained Application Protocol (CoAP), an application-level protocol [31] developed as a light version of HTTP for low power networks. The OTF module has the possibility to host several bandwidth estimation algorithms which can be selected from an external application by using CoAP. In this way, a single implementation can be configured to the needs of different applications. Furthermore, once the bandwidth estimation algorithm is selected, it can be adjusted to obtain the best performance by sending CoAP messages to modify internal parameters and configure the actual algorithm behavior.

OTF was designed with two main use cases in mind: as a completely distributed allocation module or as a complement to a PCE-based approach. In the first case, it may be used to allocate cells to build tracks between sensor nodes given routing and QoS requirements – for example for a temporal instrumentation setting – while in the second case, it may be used to increment available bandwidth among sensor nodes to adapt to surges in bandwidth requirements, for example to install and reconfigure firmware nodes, or for non-critical alarms from the production line. Given the wide range of network deployments and applications, the bandwidth estimation algorithm and the threshold value are implementation-specific.

Finally, the OTF allocation algorithm is triggered by a set of 6top events, such as the reception of a packet.

A. OTF Allocation Algorithm

The OTF allocation algorithm uses a set of three parameters for each of the mote's neighbors: (i) S, the number of currently scheduled cells; (ii) R, the number of cells required given the outgoing traffic to that neighbor; (iii) T, the neighbor-specific threshold. OTF uses the 6top statistics to obtain S. The Bandwidth Estimation Algorithm provides R to the allocation algorithm that, comparing R with S, establishes the number of cells to be added or deleted, according to the threshold T.

Algorithm 1 OTF Allocation Algorithm

procedure OTF ALLOCATION(S, R, T)
▷ S: Scheduled cells ▷ R: Required cells ▷ T: Threshold
  if $R < (S - T)$ **then** ▷ DELETE cells
    $S = R + \lfloor (T/2) \rfloor$
  else if $R > S$ **then** ▷ ADD cells
    $S = R + \lceil (T/2) \rceil$
  else ▷ $(S - T) \leq R \leq S$
    ▷ No cells to ADD or DELETE
  end if
end procedure

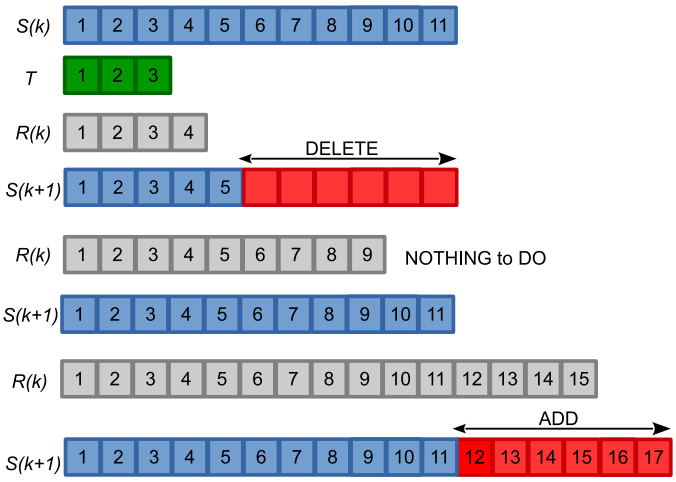

Fig. 5. Example of the OTF allocation algorithm, when $S = 11$, $T = 3$, for different values of R.

As described in Algorithm 1, for each neighbor, the allocation algorithm reserves the number of cells needed at that time (R), plus $T/2$ over-provisioned cells to allow for variations in the amount of bandwidth. In detail, the algorithm adds/deletes respectively the ceil/floor value of $T/2$, to avoid triggering a new 6top add/delete command when R even changes of 1 unit (as it should happen by considering T instead of $T/2$ in the formula). Fig. 5 shows the number of cells added/deleted by the OTF algorithm, for different values of R, when $S = 11$ and $T = 3$.

The allocation algorithm also increases the stability of the bandwidth estimation algorithm. For $T = 0$, the allocation algorithm has no effect, and no cells are over-provisioned. As a consequence, when a new packet increases the bandwidth requirement even in 1 byte/s on top of the currently allocated bandwidth, a new cell allocation request is sent to 6top; thus generating a negotiation process. After the packet is transmitted, another negotiation process to deallocate the cell is triggered, since the new allocated bandwidth will not be needed any more. This phenomenon will happen for every packet exceeding the allocated bandwidth, thus generating an oscillation on the cell allocation process. If $T \neq 0$, the allocation algorithm is enabled, and the former problem is avoided by over-provisioning: the required bandwidth is already allocated in excess when the packet arrives, and it will still be allocated when the packet has been transmitted. Furthermore, the two original negotiation processes do not take place, reducing both delay response and power consumption due to management traffic.

B. OTF-6top Interaction

OTF issues two 6top commands (`CREATE.softcell`, `DELETE.softcell` [12]) and retrieves statistics.

6top keeps statistics in the following structures [32]:

1) The `CellList` structure holds per-cell quality statistics (e.g. number of packets sent, number of packets acknowledged, signal strength of received packets on that cell).
2) The `MonitoringStatusList` structure holds per-neighbor scheduling statistics (e.g. number of scheduled cells to that neighbor).
3) The `NeighborList` structure holds per-neighbor connectivity statistics (e.g. signal strength of received packets from that neighbor).
4) The `QueueList` structure holds per-queue statistics (e.g. queue fill levels, amount of time a packet spends in a queue).

OTF uses the `MonitoringStatusList` to retrieve S, the number of scheduled cells for a specific neighbor. It determines R, the required number of cells, through `QueueList` and `NeighborList` statistics. OTF allocation algorithm then decides to add/delete cells.

OTF also configures 6top's monitoring function. This function is responsible for detecting schedule collisions (two pairs of sensor motes use the same soft cell in the schedule). When the link quality of a soft cell is lower than the average link quality of all soft cells to that neighbor, the monitoring function relocates the soft cell to a different (`slotOffset`,`channelOffset`) in the schedule.

When issuing `CREATE.softcell` and `DELETE.softcell` commands, OTF specifies the number of cells to be added/deleted. 6top is responsible for choosing the specific cells within the schedule. The mote and its neighbor go through a 6top *cell negotiation process*: the mote sends a candidate cell list to its neighbor, which picks a subset of those cells.

IV. Performance Evaluation

We evaluate the performance of OTF, both in simulation and experimentation. This section first details the simulation environment, presents the simulation results, and discusses lessons learned. It then describes the first OTF proof of concept presented at the IETF90 standardization meeting.

The goal of the performance evaluation is to answer the following questions: *What end-to-end latency can a network running OTF offer?*, *What about end-to-end reliability?* and *What trade-offs does the OTF threshold allow?*

A. The 6TiSCH Simulator

The 6TiSCH simulator[3] is used to measure the evolution of the number of scheduled cells, the number of OTF add/remove

[3]As an online addition to this article, the source code of the simulator used is available at `https://bitbucket.org/6tisch/simulator/`.

TABLE I
SIMULATION PARAMETERS.

deployment	
number of sensor motes	50
deployment area	square, $2km \times 2km$
deployment constraint	3 neighbors with PDR>50%
application data generation	
period (same for each mote)	$1, 10, 60s$ $\pm 50\%$
RPL	
parent set size	3
`MinHopRankIncrease`	256
OTF	
OTF threshold	0, 2, 4, 6, 8, 10 cells
OTF housekeeping period	1s
IEEE802.15.4e TSCH	
timeslot duration	$10ms$
length of a slotframe	101 cells
max. MAC retries	5
radio and propagation	
radio sensitivity	$-101dBm$

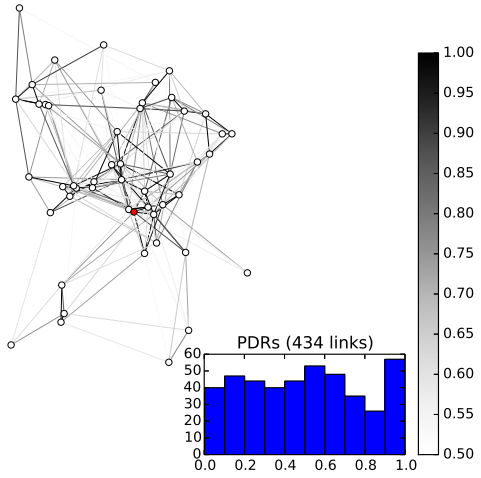

Fig. 6. Example simulated topology. The DAG root appears shaded in red (in the center of the topology). Links are shaded grey according to their PDR. The histogram shows the cumulative density function of the 434 links binned in different PDR levels.

operations, the end-to-end latency and end-to-end reliability, for a number of use cases and simulation parameters.

The 6TiSCH simulator is an open-source discrete-event simulator written in Python, developed by members of the 6TiSCH WG to quantify the performance of proposals early in the standardization process. It implements the standards and protocols defined in the 6TiSCH architecture document [9]: established standards (IEEE802.15.4e MAC [6], RPL [11]) and 6TiSCH draft standards (6top [12], OTF [13]). The latter two have been developed and added to the simulator as part of the work done and presented in this article. The set of parameters used in the simulations are summarized in Table I, and detailed in the remainder of this section.

The simulator is instrumented to measure and record the state of different parameters of interest (e.g. the timestamp at which a packet is transmitted). A single simulation run includes two stages: deploying a network with a given topology, and simulating the behavior of the network for a period of time. A new topology is used for each run. For each set of parameters, a large number of runs are performed. All results in Section IV-B are presented with a 95% confidence interval.

Each wireless link is associated a Packet Delivery Ratio (PDR), a number between 0.00 and 1.00. When a sensor mote transmits, the PDRs of the link connecting it to its neighbors is used to determine whether or not neighbors receive the packet. The PDRs are calculated on the basis of a Radio Signal Strength Indicator (RSSI) of each link, which is determined by a radio propagation model [33] according to the distance between two sensor motes. The PDR of a link is constant for the duration of a simulation run. In case different pairs of sensor motes transmit in the same timeslot and at the same frequency, packets collide. In this case, at each receiver, the RSSIs of each colliding packet are converted to Signal to Interference-plus-Noise Ratio (SINR), which is mapped to a PDR. The PDR of the link is lowered as a function of the interference by the colliding packet(s).

The simulation parameters (summarized in Table I) have been set according to RFC5673 [34], and are representative of industrial networks (e.g. deployed in a smart factory).

Deploying simulated sensor motes must result in enough wireless connectivity for the RPL routing protocol to create an efficient multi-hop topology. The deployment routine starts by placing the DAG root (the gateway mote, or sink) at the center of the deployment area. This placement is typical for industrial applications [34]. It then places other sensor motes one-by-one at random locations. At each candidate placement, the deployment routine ensures that the new sensor mote can communicate with at least 3 neighbors over wireless links with a PDR of at least 50%. It moves the new mote to a different random location until this condition is satisfied. Even though we ensure that each node is well connected to at least 3 neighbor, all links (including the ones with PDR$\leq$50%) are used in the simulation. Fig. 6 depicts an example of topology generated by the simulator. This yields topologies with an average (resp. maximum) depth of 1.963 (resp. 8 hops), when only taking into account links with PDR>50%.

For all simulation runs, the TSCH schedule contains 101 cells, each $10ms$ long (the default value in the IEEE802.15.4e standard [6]). When a mote transmits a packet to its neighbor, it expects an acknowledgment packet; if it has not received one after 5 tries, it drops the packet. The radio sensitivity (defined as the RSSI satisfying PDR of 1%) is $-101dBm$.

On top of TSCH, 6top is in charge of negotiating and deleting cells with neighbors. When adding cells, 6top takes as input by OTF the number x of cells to be ideally allocated with a neighbor. The 6top routine on a mote collects the timeslots available against its own schedule, and informs the neighbor about such availability and the desired number x of cells needed. The neighbor discards among these timeslots those that are already busy on its own schedule, i.e. timeslots already in use. From the remaining set of timeslots (which corresponds to the free timeslots common to both motes), the neighbor randomly picks x timeslots if that set cardinality is bigger than x. Otherwise, it picks exactly the remaining timeslots.

To each of the selected timeslots, the neighbors associate a randomly chosen channel offset among the 16 available at $2.4GHz$. When deleting cells, 6top sends to its neighbor the cells to be deleted on their link.

OTF decides whether to add or remove cells to neighbor motes and, in case, feeds 6top for performing such an operation. The OTF algorithm running on a mote is executed every second, and leads the bandwidth allocation strategy. It estimates the outgoing amount of traffic as the sum of: (i) the forwarded traffic, as the average number of packets received by "child" neighbors and to be relayed towards the DAGroot, and (ii) the self-generated traffic, as the number of packets generated by the mote itself for each slotframe. If the latter is very well known to an OTF-enabled mote, an estimation technique is required for evaluating the forwarded traffic. Such an estimation is performed through an auto-regressive filter, which averages with equal weights (i.e. 0.5) a previous estimation of the incoming traffic and the number of packets received in the time interval elapsed (expressed in number of slotframes) since the previous estimation took place. At steady state, i.e. when the network churn is null, this filter outputs the average traffic incoming to the mote.

On the basis of the estimated outgoing traffic, OTF computes the number R of required TX cells, and compares this value to the number of S TX scheduled cells. It takes the threshold T into account (Section III-A) to decide whether to trigger a 6top negotiation.

To assess the OTF over-provisioning performances, several OTF threshold values have been used (0, 2, 4, 6, 8 and 10 cells).

In the RPL implementation, each mote maintains a routing parent set of 3 motes. The `MinHopRankIncrease`, the minimum increase of the DAG rank between a mote and its parent [11], is set to 256, according to 6TiSCH-minimal-draft [35].

Each sensor mote generates data packets regularly, with a period equal to `packetPeriod` $\pm 50\%$. Three values of `packetPeriod` are used: $1s$, $10s$, $60s$. The `packetPeriod` is the same for all sensor motes in the network; each sensor mote uniformly selects a new random period within $[\frac{\texttt{packetPeriod}}{2} \ldots \frac{3 \cdot \texttt{packetPeriod}}{2}]$ after each transmitted packet.

B. Simulation Results

The simulation results presented in this section are depicted as average values with a 95% confidence interval. Each data point represents 100 simulation runs.

Fig. 7 shows the evolution of the number of scheduled cells over time, which is the sum of all cells scheduled by all the sensor motes in the network related to the activity (and energy consumption) in the network. The purpose of OTF is to match the number of communication cells scheduled in the network to the amount of data traffic. The lower the packet period, the more traffic, and hence the more cells are required. The OTF threshold is the number of cells that are over-provisioned: the larger the value of the threshold, the more cells are scheduled.

Fig. 8 shows the number of scheduled cells at the end of the simulation depicted in Fig. 7, i.e. after 100 slotframe cycles

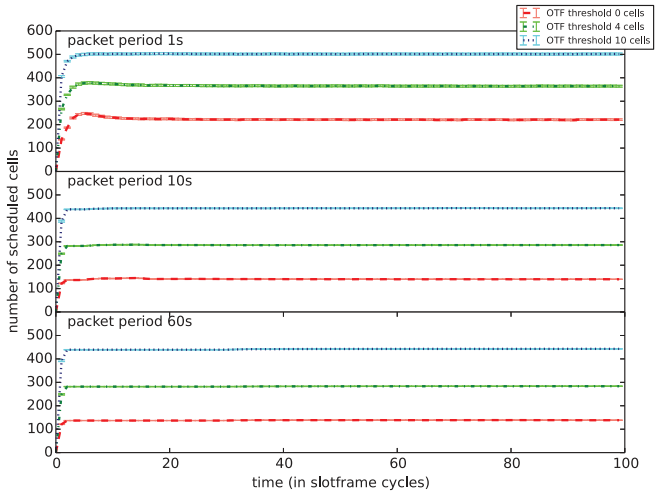

Fig. 7. Sum of the scheduled cells at each of the 50 motes in the network, as a function of time. One slotframe cycle is 101 slots long, or 1.010 s.

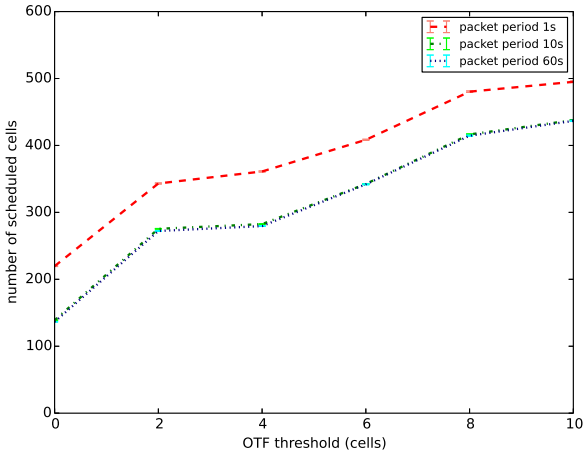

Fig. 8. Sum of the scheduled cells at each of the 50 sensor motes in the network, at the end of the simulation presented in Fig. 7, as a function of the OTF threshold. Note that results for packet period 10 s and 60 s are so close that they appear on top of one another.

have elapsed. This number of scheduled cells is plotted as a function of the OTF threshold. As expected, the larger the threshold, the more cells are scheduled.

These results are obtained for a slotframe duration of 101 slots. With 16 frequencies, there are hence 1616 cells in the schedule. Even with each of the 50 sensor motes generating one packet per second, and an OTF threshold of 10 slot, only approx. 500 cells are scheduled, thus, around 30%. Readers interested in the scalability limits of such scheduling are referred to the work of Samuel Zats [36].

Figs. 7 and 8 depict the averall number of cells schedule for all 50 motes. The evolution of the total number of cell (for example over time) is representative of the evolution of the number of scheduled cells at an average mote.

OTF triggers the 6top sublayer to add or delete cells to neighbors. This is called an "OTF operation". Fig. 9 depicts the average number of OTF operations per cycle (one cycle

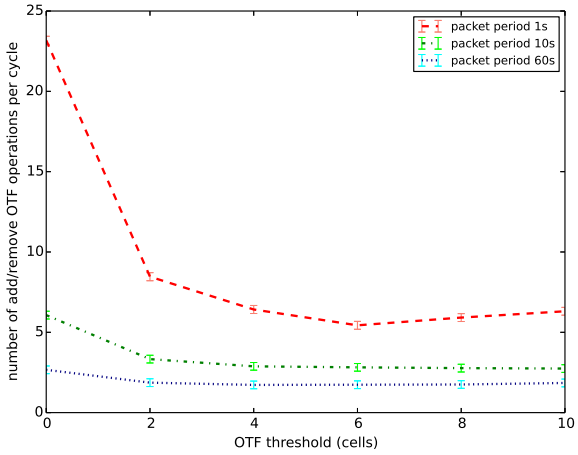

Fig. 9. The average number of times OTF asks the 6top sublayer to add/delete some cells to a neighbor per cycle, over the whole 50-sensor mote network.

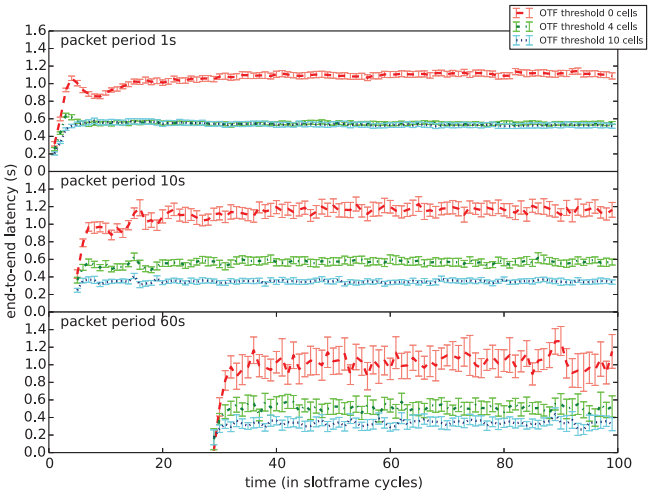

Fig. 11. The evolution of end-to-end latency over time.

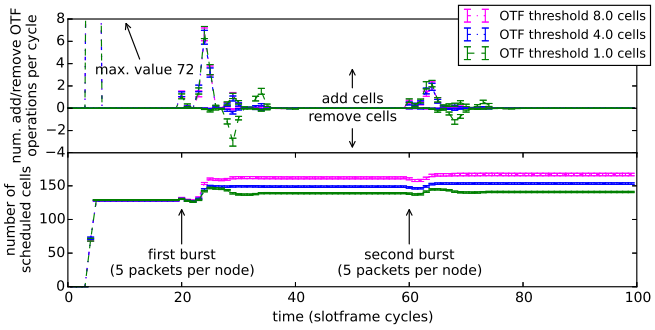

Fig. 10. Impulse response of OTF: 5 packets are generated by each mote at times 20 s and 60 s. The top plot shows the OTF activity (above the line quantifies the number of cells added, below the line the numbers of cells removed). The bottom portion shows the resulting number of scheduled cells.

is 101 cells long, or $1.01s$) for the whole network, over the first 100 cycles. The more dynamic OTF, the more operations it performs. Each operation causes the 6top sublayer of two neighbor motes to negotiate to add/delete a cell, which induces communication overhead. Lower activity is therefore preferred. The larger the OTF threshold, the more cells are over-provisioned in the network; this hysteresis will cause OTF to add/delete cells less often.

Fig. 10 shows the "impulse response" of OTF. Instead of generating data at a constant rate, he application on each sensor mote generates 5 packets at the time, at t=20 s and t=60 s. Fig. 10 shows how this results in OTF adding a number of cells, then removing those as the burst of traffic has ended.

End-to-end latency is calculated as the amount of time between the moment a packet is generated at a sensor mote, and the moment it is received at the DAGroot, possibly after traveling multiple hops. Fig. 11 shows this latency as a function of time. Motes only start generating data at the earliest $\frac{\texttt{packetPeriod}}{2}$ seconds after the beginning of the simulation. The packets generated at neighbor motes of the DAG root can reach it the fastest, leading to a "ramp up" effect of the latency. The steady-state portion of Fig. 11 shows how the latency is smaller for larger values of the OTF threshold. With a larger threshold, more cells are scheduled, giving a packet more frequent opportunities to progress one hop, which results in lower latency.

The portion of packets generated in the network that reaches the DAG root is called end-to-end reliability. A packet can be lost for three reasons: (1) the maximum number of MAC retries is exceeded, possibly because of a too low link PDR or collisions, (2) a sensor mote has no RPL routing parent to forward data to, or (3) a sensor mote has no outbound cells to communicate with that RPL parent. The simulator logs all three events. Except for the first 2 slotframe cycles during which the very first cells are installed, all packet loss is due to reason (1).

Fig. 12 shows the end-to-end reliability as a function of OTF threshold and packet period. For a packet period of $10s$ and $60s$, end-to-end reliability is above 99%. It is well below for a $1s$ packet period because of collisions. With a $1s$ period, between 200 and 500 combined cells are scheduled in the network (see Figs. 7 and 8). The probability that two pairs of neighbor motes randomly pick the same cell becomes significant, resulting in a higher packet collision rate and lower reliability. This is exacerbated at higher OTF threshold values, since sensor motes are more cell-hungry. When trying to greedily over-provision their bandwidth, some sensor motes will not be able to allocate a sufficient number of cells. Hence, an equilibrium of load-balanced resources among sensor motes will not be reached. This is also confirmed by observing that, with the OTF threshold equal to 6, the reliability starts decreasing while the OTF activity starts increasing (Fig. 9).

C. Lesson learned from Simulation Results

The obtained simulation results show that OTF is a valid approach for managing a TSCH schedule in a distributed manner. They highlight the importance of choosing a value of the OTF threshold, as it determines the trade-off between energy consumption, network stability, latency and reliability.

A higher threshold leads to a higher number of scheduled cells (Fig. 8) – hence energy consumption–, but a more stable

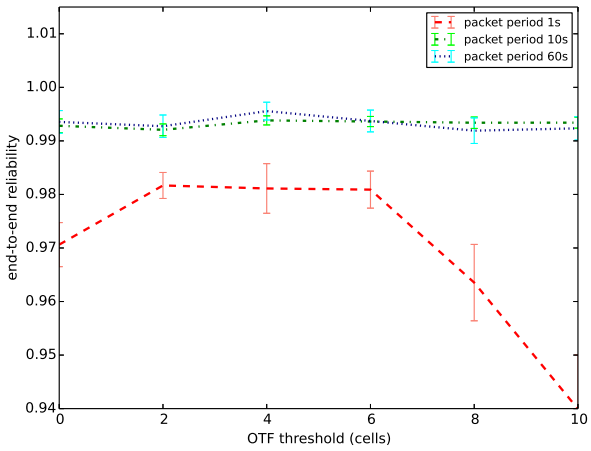

Fig. 12. End-to-end reliability as a function of OTF threshold.

network with less OTF operations (Fig. 9), a lower latency (Fig. 11) and, in general, a higher reliability (Fig. 12).

Yet, the cells OTF schedules are shared by nature. This "slotted Aloha" behavior means that packet collisions are possible, and can lead to network collapse if too many cells are scheduled. Fig. 12 shows the end-to-end reliability plummeting for a $1s$ packet period and an OTF threshold of 10 cells. Under these parameters (see Fig. 7) only about a third of the 1616 cells (101 cells $\times$ 16 frequency channels) in the network are scheduled.

To answer the questions posed at the beginning of Section IV, these results show how, in a 50-sensor mote multi-hop network typical for an industrial application, a purely distributed approach such as OTF can achieve end-to-end latencies in the order of a second, with over 99% end-to-end reliability.

D. First OTF Proof of concept

During the IETF90 standardization meeting (in Toronto, Canada), a plugfest[4] was organized to demonstrate different applications of the 6TiSCH, 6lo[5] and ROLL[6] Working Groups.

At the plugfest, we presented the first OTF proof of concept, implemented in the context of the OpenWSN project [37]. OpenWSN[7] is an open-source implementation of a protocol stack based on Internet of Things standards, using a variety of hardware and software platforms. Its default protocol stack includes IEEE802.15.4e, as well as 6LoWPAN and other upper-layer IETF standards, including CoAP, RPL and several recent drafts from the 6TiSCH WG.

Among them, the OTF draft was implemented and run on a heterogeneous 10-sensor mote wireless mesh network. In detail, as the first proof that OTF can run in a real network, on low power motes, we implemented a simplified version of OTF, based on a chain-reaction. Each sensor mote, along the multi-hop path from a leaf mote towards the DAG root, following the request of more bandwidth (soft cells) from its child, asked for the same number of cells to its parent sensor mote. By doing so, the new request of bandwidth was fulfilled, in a distributed manner in the whole network, along each multi-hop path, without the intervention of a PCE.

This chained reservation (cascade of cells allocations) from the leaf until the DAG root did not use any centralized scheduling entity, but relied on the OTF module to establish a higher bandwidth path among all sensor motes on the multi-hop path. This powerful behavior demonstrates the potential of OTF for providing distributed soft cell allocation.

The OTF proof of concept presented at IETF90 plugfest covered its basic functionality. As next steps, we plan to develop a full OTF implementation, including bandwidth allocation algorithms, threshold management, triggering events, and CoAP interface.

[4]Plugfest, https://openwsn.atlassian.net/wiki/display/OW/IETF90+plugfest
[5]6lo, http://datatracker.ietf.org/wg/6lo/charter/
[6]ROLL, http://datatracker.ietf.org/wg/roll/charter/
[7]OpenWSN, http://www.openwsn.org

V. Conclusion

This article presents the OTF bandwidth reservation module for 6TiSCH networks, based on the 6top sublayer and IEEE802.15.4e TSCH. OTF is a distributed scheduling algorithm that dynamically matches link-layer resources to the communication needs of network applications. OTF monitors 6top statistics and, if required, triggers 6top to add/delete cells. 6top executes an internal negotiation protocol to make the necessary changes to the TSCH schedule. A threshold mechanism in OTF provides hysteresis, thereby preventing ringing effects in consecutive cell allocations. This threshold balances communication overhead (due to frequent 6top negotiations) and energy consumption (due to unused over-provisioned cells).

Performance of OTF was assessed through simulations run over a 50-sensor mote multi-hop network representative of a smart factory. The results have shown that OTF can pursue end-to-end latencies of the order of a second, with over 99% end-to-end reliability. The first OTF proof of concept implementation in OpenWSN was demonstrated and also presented in this article in order to pragmatically highlight the relevance of OTF to the 6TiSCH architecture.

The future work includes evaluating the performance of OTF using different bandwidth estimation algorithms, and 6top negotiation procedures. Both mechanisms have been left open in the 6top and OTF standardization work to allow practitioners to tailor 6TiSCH to their own vertical requirements. Finally, large-scale experimental studies are envisaged in the near future using OpenWSN.

Acknowledgment

This publication was supported by the INTER/POLLUX/12/4434480 CoSDN project, funded by the Fonds National de la Recherche, Luxembourg, the Anillo Project ACT-53, Fondecyt No. 11121475 and the CIRIC (INRIA-Chile) Project "Network Design". The Authors would like to thank Pascal Thubert, Xavier Vilajosana and Kris Pister for the valuable discussions and suggestions.

REFERENCES

[1] J.-D. Decotignie and P. Pleinevaux, "A Survey on Industrial Communication Networks," *Annales Des Télécommunications*, vol. 48, no. 9-10, pp. 435–448, 1993.

[2] P. Gaj, J. Jasperneite, and M. Felser, "Computer Communication Within Industrial Distributed Environment: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 182–189, 2013.

[3] K. Pister and L. Doherty, "TSMP: Time Synchronized Mesh Protocol," *International Symposium on Distributed Sensor Networks (DSN)*, pp. 391–398, 2008.

[4] *WirelessHART Specification 75: TDMA Data-Link Layer*, HART Communication Foundation Std., Rev. 1.1, 2008, hCF_SPEC-75.

[5] *ISA-100.11a-2011: Wireless Systems for Industrial Automation: Process Control and Related Applications*, International Society of Automation (ISA) Std., May 2011.

[6] *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*, IEEE Std., 16 April 2012.

[7] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6)*, IETF Std., 1998.

[8] M. R. Palattella, P. Thubert, X. Vilajosana, T. Watteyne, Q. Wang, and T. Engel, *Internet of Things: Challenges and Opportunities*. Springer, 2014, ch. 6TiSCH Wireless Industrial Networks: Determinism Meets IPv6, pp. 111–141.

[9] P. Thubert, *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e*, IETF Std. draft-ietf-6tisch-architecture-08 [work-in-progress], 12 May 2015.

[10] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, *Neighbor Discovery Optimization for IPv6 over Low-power Wireless Personal Area Networks (6LoWPANs)*, IETF Std., 2012.

[11] 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*, IETF Std. RFC6550, March 2012.

[12] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top)*, IETF Std. draft-wang-6tisch-6top-sublayer-01 [work-in-progress], 4 July 2014.

[13] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura, *6TiSCH On-the-Fly Scheduling*, IETF Std. draft-dujovne-6tisch-on-the-fly-06 [work-in-progress], 4 July 2015.

[14] *IEEE std. 802.15.4. Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless PersonalArea Networks (LR-WPANs).*, IEEE Std., 2011.

[15] Chraim, F.; Erol, Y.; Pister, K., "Wireless Gas Leak Detection and Localization," Industrial Informatics, IEEE Transactions on , vol.PP, no.99, pp.1,1

[16] Murvay, Pal-Stefan, and Ioan Silea. "A survey on gas leak detection and localization techniques" Journal of Loss Prevention in the Process Industries 25, no. 6 (2012): 966-973.

[17] Ogle, Russell A., Scott E. Dillon, and Mark Fecke. "Explosion from a smoldering silo fire." Process Safety Progress 33, no. 1 (2014): 94-103.

[18] Hu, Xiaoya, Bingwen Wang, and Han Ji. "A Wireless Sensor Network-Based Structural Health Monitoring System for Highway Bridges." Computer-Aided Civil and Infrastructure Engineering 28, no. 3 (2013): 193-209.

[19] Chae, M. J., H. S. Yoo, J. Y. Kim, and M. Y. Cho. "Development of a wireless sensor network system for suspension bridge health monitoring." Automation in Construction 21 (2012): 237-252.

[20] Bocca, Maurizio, Lasse M. Eriksson, Aamir Mahmood, Riku Jntti, and Jyrki Kullaa. "A synchronized wireless sensor network for experimental modal analysis in structural health monitoring" Computer-Aided Civil and Infrastructure Engineering 26, no. 7 (2011): 483-499.

[21] Jia Luo, Weijian Huang, Shirong Zhang, "Energy cost optimal operation of belt conveyors using model predictive control methodology", Journal of Cleaner Production, Available online 2 October 2014,

[22] Wang, Li, and Li Zhigang. "Research on Control System of Belt Conveyor in Coal Mine" In Electrical, Information Engineering and Mechatronics 2011, pp. 885-891. Springer London, 2012.

[23] T. Watteyne, A. Mehta, and K. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," in *Symposium on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*. ACM, 2009, pp. 116–123.

[24] R. Braden, L. Zhang, S. Berson, S. Herzog, and J. S., *Resource ReSerVation Protocol (RSVP)*, IETF Std. RFC2205, 1997.

[25] 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 *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on* (pp. 327-332). IEEE, September 2012.

[26] M.R. Palattella, N. Accettura, L.A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802. 15.4e TSCH. *Sensors Journal, IEEE*, 13(10), 3655-3666, 2013.

[27] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label Switching over IEEE802.15.4e Networks," *Transactions on Emerging Telecommunications Technologies (ETT)*, vol. 24, no. 5, pp. 458–475, 2013.

[28] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch, *Next Steps in Signaling (NSIS): Framework*, IETF Std., 2005.

[29] P. Thubert, *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*, RFC6552, March 2012.

[30] O. Gnawali, and P. Levis, *The Minimum Rank with Hysteresis Objective Function*, RFC 6719, September 2012.

[31] R. Sudhaakar and P. Zand, *6TiSCH Resource Management and Interaction using CoAP*, IETF Std. draft-ietf-6tisch-coap-02 [work-in-progress], 4 December 2014.

[32] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top) Interface*, IETF Std. draft-ietf-6tisch-6top-interface-04 [work-in-progress], 6 July 2015.

[33] S. Zats, "Wireless Sensor Networks Scaling and Deployment in Industrial Automation," Master's thesis, University of California, Berkeley, 2010.

[34] K. Pister, P. Thubert, S. Dwars, and T. Phinney, *Industrial Routing Requirements in Low-Power and Lossy Networks*, IETF Std. RFC5673, October 2009.

[35] X. Vilajosana and K. Pister, *Minimal 6TiSCH Configuration*, IETF Std. draft-ietf-6tisch-minimal-11 [work-in-progress], 6 July 2015.

[36] S. Zats, R. Su, T. Watteyne and K. Pister, "Scalability of Time Synchronized Wireless Sensor Networking," *IEEE Industrial Electronics Society Conference (IECON)*, IEEE, Melbourne, Australia, 7-10 November 2011.

[37] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: A Standards-Based Low-Power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies (ETT)*, vol. 23, no. 5, pp. 480–493, August 2012.