IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2018, accepted June 12, 2018. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2018.2849090

Hybrid Schedule Management in 6TiSCH Networks: The Coexistence of Determinism and Flexibility

ABDULKADIR KARAAGAC, INGRID MOERMAN, AND JEROEN HOEBEKE
IDLab, Department of Information Technology, Ghent University—imec, B-9052 Ghent, Belgium
Corresponding author: Abdulkadir Karaagac (abdulkadir.karaagac@ugent.be)

ABSTRACT With the emergence of the Internet of Things (IoT), Industry 4.0, and cyber-physical system concepts, there is a tremendous change ongoing in industrial applications, which is imposing increasingly diverse and demanding network dynamics and requirements with a wider and more fine-grained scale. The purpose of this paper is to investigate how a hybrid schedule management in 6TiSCH architecture can be used to achieve the coexistence of applications with heavily diverse networking requirements. We study the fundamental functionalities and also describe network scenarios, where such a hybrid scheduling approach can be used. In addition, we present the details about the design and implementation of the first 6TiSCH centralized scheduling framework based on CoAP management interface (CoMI). We also provide theoretical and experimental analysis, where we study the cost of schedule management operations and illustrate the operation of the CoMI-based 6TiSCH schedule management.

INDEX TERMS 6TiSCH, IEEE 820.15.4e TSCH, centralized scheduling, distributed scheduling, CoMI, industrial IoT.

I. INTRODUCTION

As the ideas and technologies behind the Wireless Sensor Networks (WSNs) and the Internet of Things (IoT) take root, a vast array of new possibilities and applications is emerging with a significantly increased number of services and devices connected to the Internet. With respect to Industry, this paradigm of intelligent connectivity of everything is further imposing industrial applications with increasingly diverse and demanding network dynamics and requirements [1].

In time-critical applications (e.g. automation and control systems) with strict real-time constraints, even a fractional violation of designated constraints can lead to performance drops or even system outages which can incur significant cost for the industrial companies [2]. On the other side, large-scale and dynamic industrial monitoring applications bring more concerns about the network flexibility and scalability. Therefore, as the diversity in the ecosystem of industrial applications increases, there is a growing need for more flexible and reconfigurable networking technologies where diverse applications can coexist while all system requirements are being met simultaneously.

Over the past few years, we have seen the emergence of several wireless standards based on Time-Slotted Channel-Hopping (TSCH) scheme to meet the stringent requirements of industrial applications, such as WirelessHart [3] and ISA100.11a [4], and finally 802.15.4e TSCH [5] and 6TiSCH [6]. Compared to its TSCH-based predecessors, 6TiSCH has opted for an open and standardized communication stack as well as support for different scheduling schemes, turning them into a more open and flexible, but also equally reliable and deterministic wireless communication solution that can be used in heterogeneous industrial applications.

In 6TiSCH, as all TSCH-based networks, the communication is orchestrated by a schedule that indicates to each device what to do (transmit, receive, sleep) in each time slot [7]. These schedules can be either static (predefined) or managed in a centralized or a distributed manner. On the one hand, centralized scheduling techniques can theoretically obtain better performance, so they are the preferred choice if the focus is on realizing best performance or latency bounded connectivity. However, due to the need of maintaining complete topology information at a central unit, centrally managed networks generally do not scale to large configurations and they are generally advised for fairly static networks. On the other hand, distributed scheduling techniques are relatively more

suitable for dynamic or large networks, while they are not able to provide determinism in any sense [8], [9].

Therefore, in order to meet the highly dynamic and strong expectations of todays industrial settings, the schedule management needs to be performed coordinately by both local and centralized logic where each can target different network performances. In this sense, 6TiSCH enables the combination of SDN-type centralized routing and scheduling, for time-sensitive flows, with distributed routing and scheduling based on RPL, for ancillary traffic [10]. The fundamental question of this paper is how this increased wireless networking flexibility and improved access to lower-level wireless functionality in 6TiSCH can be optimally exploited. For that purpose, we investigate light-weight and highly automated schedule management mechanisms for 6TiSCH networks based on the ongoing standardization efforts in IETF [6], [11], [12].

This paper provides an overview of how the 6TiSCH architecture can be used to achieve the coexistence of deterministic and latency-bounded applications with scalable and dynamic applications in the same network and also investigates the potential of 6TiSCH Networks to become one of the de-facto wireless communication technologies for low data rate industrial applications with heterogeneous constraints.

In summary, the contributions of this paper are as follows:

i. Study of the fundamental functionalities for the joint coordination and interaction of distributed and centralized scheduling mechanisms.
ii. Description of network scenarios and approaches where such hybrid schedule management can be used.
iii. Details about the design, architecture and implementation of the first 6TiSCH Centralized Scheduling Framework via a standardized interface: CoMI [11].
iv. Theoretical and experimental analysis for the cost and functionality of 6TiSCH schedule management.

The remainder of this paper is organized as follows. Section II provides detailed background on the target technologies and protocols. Section III introduces the Hybrid Scheduling approaches in 6TiSCH Networks and the fundamental functionalities for the coordination and interaction of distributed and centralized scheduling mechanisms. In Section IV, the CoMI-based 6TiSCH network management mechanism is described with details about the architecture, data models and other fundamental functionalities, including a real implementation and cost analysis. Section V illustrates an experimental study, following by conclusions in Section VI.

II. TECHNICAL BACKGROUND

A. 6TiSCH: IPv6 OVER IEEE 802.15.4e TSCH

IEEE 802.15.4e is a recent MAC amendment of the IEEE 802.15.4 standard, specially designed for harsh industrial environments with a reliable and deterministic communication scheme based on Time-Slotted Channel Hopping [5]. In a TSCH network, time is sliced up into time slots and all motes across a network are synchronized in order of tens of microseconds. The overall communication is orchestrated by a schedule which defines the action (transmit, receive, sleep) of each node in each time slot [5]. In this TSCH schedule, a single element, named *cell*, is identified by a pair of *slotOffset* and *channelOffset*, which is used to define the time and frequency that will be used by communicating motes. The proper functioning of a TSCH network depends on this schedule which can be typically created in various ways, but should be computed according to the specific requirements of the applications, such as latency, reliability and energy.

Recently, a new IETF Working Group (WG), named 6TiSCH, has been formed to investigate IPv6 connectivity over the TSCH mode of IEEE 802.15.4e protocol [6]. Currently, the 6TiSCH WG is still active with two finalized RFCs and several active Internet Drafts. The WG has defined an operation sub-layer, called 6top, in order to bind the prior IPv6-enabled standards (IETF 6LoWPAN, RPL and CoAP) with IEEE 802.15.4e TSCH and it targets to create a standardized approach to build and maintain a schedule, perform TSCH configuration and control procedures. Also, a minimal 6TiSCH configuration is defined in order to achieve basic interoperability among all implementations. The group is also working on a set of protocols for setting up a TSCH schedule in distributed approach with various scheduling functions [13].

B. SCHEDULE MANAGEMENT in 6TiSCH NETWORKS

The 6TiSCH protocol defines four approaches to manage the TSCH schedule [6]. Initially, there can be static schedules which are predefined or learned during the joining process, such as the shared slots that are known and used by every node in the network. Secondly, a Neighbor-to-Neighbor scheme is defined where the motes agree on a schedule by using distributed scheduling protocols and neighbor-to-neighbor negotiation (6P) [14]. Moreover, 6TiSCH also enables remote schedule management where a central entity, called Path Computation Element (PCE), is continuously adjusting the TSCH schedule according to the network state and traffic requirements. Lastly, a hop-by-hop scheduling is defined to achieve cell reservations along paths for particular data flows. An overview of the 6TiSCH Network Architecture and Scheduling schemes are presented in Figure 1.

As it is presented in Figure 1, a cell scheduled via centralized scheduling is called ''Hard Cell'' and the 6top sublayer or any local entity cannot dynamically alter such a cell, whereas, a ''Soft Cell'' is scheduled by a distributed scheduling entity.

After the formation of the 6TiSCH WG, the focus on TSCH networks and scheduling algorithms has increased immensely with several research efforts. Besides the scheduling functions defined by the 6TiSCH WG, a large portion of these efforts mainly focus on the distributed scheduling algorithms and mechanisms [8], [9], [15], [16]. These efforts target resource reservation algorithms, as an alternative to centralized approach, in order to build better scalable, but non-deterministic, networks which can cope with disturbances and network changes. Similarly, [17] introduces Orchestra,

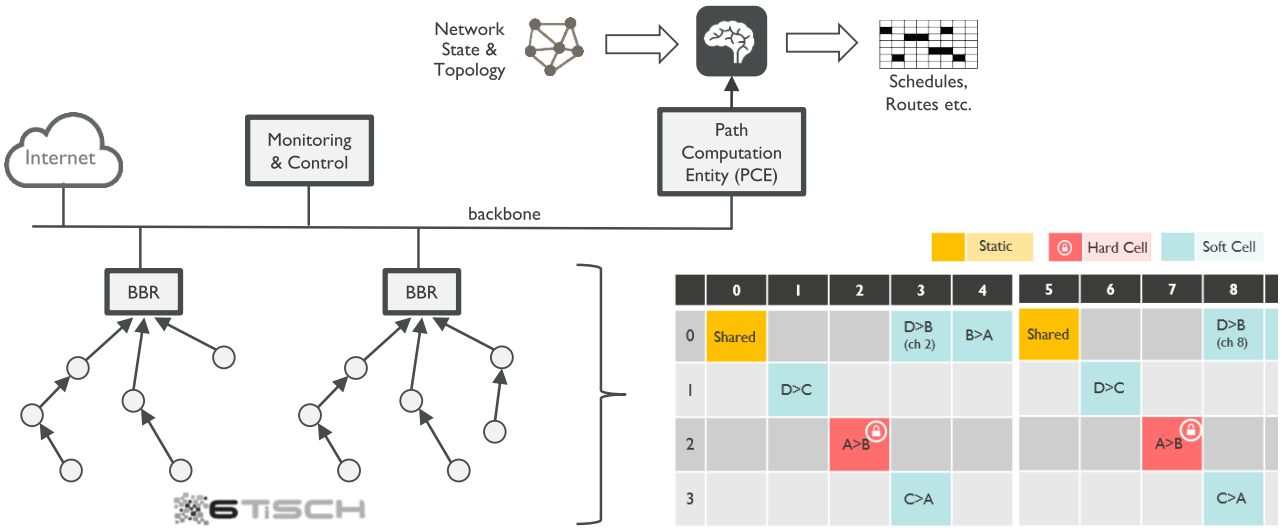

FIGURE 1. Architecture and scheduling in 6TiSCH networks.

which allows nodes to autonomously compute schedules without any central or distributed scheduler and without signaling overhead. On the other side, we also see efforts targeting optimal or heuristic scheduling algorithms for centralized scheduling [18]–[21]. However, they mainly target theoretical analysis of scheduling algorithms based on network topology and/or traffic load, but no real implementation or practical experiment.

C. CoAP MANAGEMENT INTERFACE (CoMI)

Recently, the IETF CoRE working group has started to work on a network management interface, called CoAP Management Interface (CoMI) [11], for the management of constrained devices and networks. CoMI uses the CoAP protocol and defines efficient Client-Server-based interaction models in order to access structured data specified in YANG. Table 1 positions the CoMi-based approach against other network management standards defined by the IETF.

TABLE 1. IETF standards for network management.

	SNMP	NETCONF	RESTCONF	CoMI
Resources	OIDs	Paths	URIs	URIs
Data Models	MIBs	Yang Models (IETF)	Yang Models (IETF)	Yang Models (IETF)
Data Modeling Language	SMI	YANG	YANG	YANG
Management Operations	SNMP	NETCONF	HTTP Operations	CoAP
Encoding	BER	XML	XML, JSON..	CBOR
Transport Protocol	UDP	TCP	TCP	UDP
Target Use	Modem, Router, Switch	Modem, Router, Switch	Modem, Router, Switch via Web Apps.	Constrained Devices

By using standardized data sets specified in a standardized language (YANG), CoMI promotes the interoperability between devices and applications from different ecosystems and/or manufacturers. As client and server share an off-line schema, like a contract, defined in YANG, CoMI does not waste bandwidth for the distribution of lots of meta-data describing supported data models and resources.

In CoMI, CBOR [12], a binary encoding scheme, is used to better fit M2M and IoT requirements on constrained devices. CoMI uses the YANG to CBOR mapping and converts YANG identifier strings to numeric identifiers for payload size reduction as defined in [22]. Therefore, in CoAP messages, the YANG objects are carried as maps containing (identifier, value) pairs. CoMI end points that implement the CoMI management protocol support at least one discoverable management resource of ''resource type'' (rt): core.c, with a recommended path root '/c'. Every data node of the YANG modules loaded in the CoMI server represents a resource of the data store container (e.g. $/c/sid_{base64}$). These YANG Schema Item Identifiers (SID) are globally unique 64-bit numeric identifiers used to identify all items used in YANG and sid_{base64} represents the sid encoded in base64 using the URL and Filename safe alphabet as defined by [23]. Reference [24] defines a file format used to persist and publish assigned SIDs.

As CoMI uses the CoAP protocol, it allows data store contents to be read, created, modified and deleted via CoAP methods (Get, Fetch, Post, Put, iPatch, Delete). Moreover, it also offers a query parameter ('k') that can be used to address a specific set of contents. However, in addition to the CoMI agent that performs the management functionalities, the device would also need a separate CoAP based agent to control the actuation and sensing features. The architecture and design of CoMI is summarized in Figure 2.

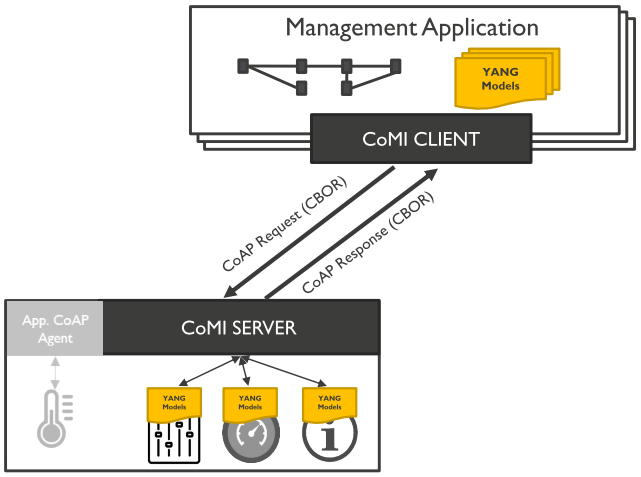

FIGURE 2. CoAP management interface.

Due to its' open, flexible and efficient design, the CoMI protocol has already attracted significant interest from the research community and started to be adopted for remote management of devices and networks with limited resources, including 6TiSCH architecture. For that reason, we chose to develop our 6TiSCH Schedule Management platform, described in the following sections, based on the CoMI protocol.

III. HYBRID SCHEDULING IN 6TiSCH NETWORKS

With distributed scheduling, nodes locally build and maintain their schedule. Therefore it is easier to scale for large network sizes and it is relatively more suitable for dynamic and mobile networks. Whereas in centralized scheduling, the PCE responsible for building and maintaining the schedule. In case of any topological change, this information would have to be reported to the central scheduler, which would have to re-compute a schedule and inform the nodes about the change. Therefore, while these centralized scheduling techniques can theoretically obtain better performance, their real implementations trigger the exchange of a large amount of signaling overhead, making these networks rather scale poorly. Therefore, they are only advised for fairly static networks.

As we described in the previous sections, 6TiSCH offers the combination of SDN-type centralized routing and scheduling with distributed routing and scheduling. So far, there are several pieces of work targeting each of these schedule management schemes, however their combination in the same network is still a vague point which needs further attention from the research community. In this section, we investigate the integration and cooperation of centralized and distributed scheduling mechanisms which will allow the coexistence of time-sensitive and scalable Industrial IoT Applications.

Our architecture for supporting such Hybrid Scheduling of 6TiSCH networks is presented in Figure 3. As it can be seen

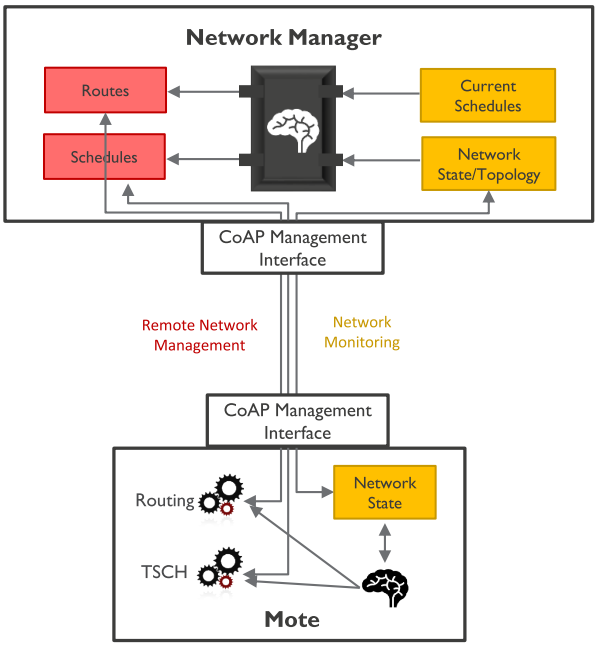

FIGURE 3. Hybrid scheduling architecture.

in this figure, there are two different logical entities which can define the schedules that the nodes will use. The local scheduling entity monitors the network state and decides on the schedules based on the local network view. A centralized entity collects the network state and topology, current schedules and calculate the routes and schedules based on the global view of the network. This entity can create deterministic behavior by eliminating congestion loss, guaranteeing a worst case latency and eliminating equipment failure losses via multi-path or path replication techniques [25]. This entity uses the CoMI interface for network monitoring and schedule installation.

A. MANAGEMENT OF THE SOFT VS HARD CELLS

The coexistence of two independent control logics which manage the schedules on a single device raises the question on how to coordinate those control entities and how to solve potential conflicts.

The distributed scheduling mechanisms can avoid creating any kind of contradiction, as they have all local knowledge about the hard and soft cells in the node. However, a centralized scheduling mechanism may not know the locally created schedules (or created by other centralized entities) or it may choose to change these schedules. Therefore, in each remote schedule installation, the node needs to validate schedules and take action to remove or update other schedules which might create a conflict. Figure 4 provides a flow chart for this validation and schedule installation process happening upon the reception of a schedule installation packet via the CoMI interface. As it is presented in this flowchart, during remote schedule installation, if a hard cell with the same slot offset exists in the node, the schedule installation fails and returns an error message. If the installed schedule contradicts with any

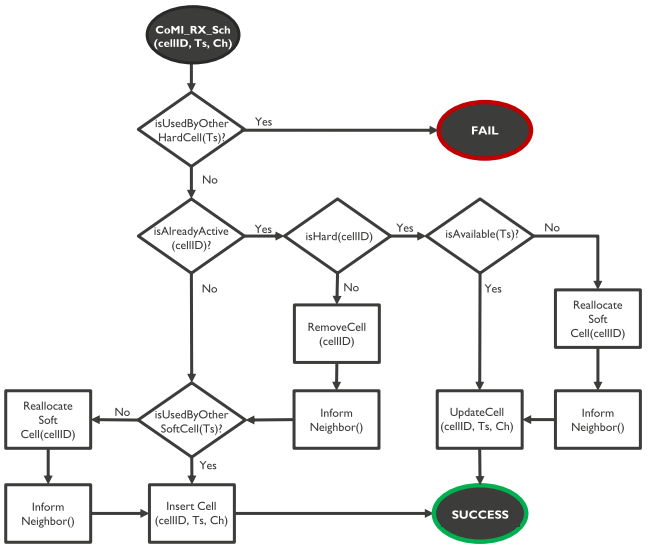

FIGURE 4. Schedule installation flowchart.

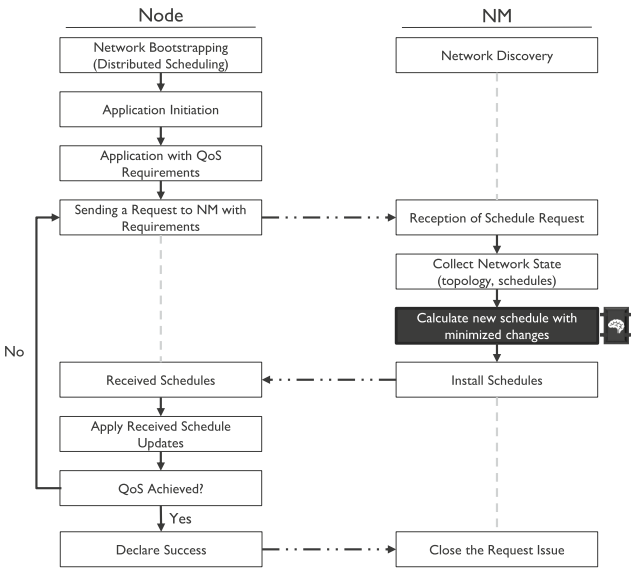

FIGURE 5. Simplified state machine for schedule management.

soft cell, then the node needs to first reallocate this soft cell by starting a negotiation process with its neighbor by means of the distributed scheduling mechanism, and finally add the newly installed hard cell. If the operation is updating an already existing hard cell, then the node only needs to change the relevant fields of the target hard cell. Once the schedule installation has been successfully finalized, the node needs to send a confirmation message to the centralized scheduler.

Alternatively, the centralized scheduler can also try to avoid any conflict by monitoring the soft and hard cells on each node. This will save nodes from cell reallocation messages for soft cells, at the expense of further message overhead due to schedule monitoring.

B. COLLISION-FREE SCHEDULES IN HYBRID SCHEDULING

Normally in centralized scheduling, the PCE can easily avoid re-usage of the same cells across the whole network, resulting in a collision-free schedule. However, in hybrid scheduling, if no precautions are taken, distributed scheduling can possibly use the same cell and create collisions with centrally installed schedules at neighboring nodes and undermine the deterministic behavior.

In order to prevent any future collision for critical data which can occur due to such distributed cell assignment, the centralized scheduling can install *"hard sleep"* cells to other nodes (or at least neighboring nodes). This will guarantee the collision-free transmission of critical data by preventing all nodes in the network or only neighboring nodes to use the same cell.

C. HYBRID SCHEDULING SCENARIOS/APPROACHES

Hybrid Scheduling brings flexibility in 6TiSCH network management with a wide range of networking capabilities. In this section, we define four different scenarios that illustrate how one can use hybrid scheduling in order to support applications with heterogeneous nature.

1) THE COEXISTENCE OF DETERMINISM AND FLEXIBILITY

In the first approach, while the network nodes locally create soft cells targeting non-critical traffic, they need to send a request to the centralized scheduler in order to reserve necessary resources to meet QoS requirements of critical applications. In Figure 5, a simplified state machine for such a network schedule management process is provided.

As it can be seen in this figure, we allow the distributed scheduling mechanism to build the network during the bootstrapping phase. In this phase, the nodes are getting connected to the network and also they learn about their neighbors and link qualities. After this phase, the nodes with QoS requirements start sending requests to the Network Manager about their application requirements. When the Network Manager receives a scheduling request, it first collects the Network State information (topology, existing schedules etc.) via CoMI if this is necessary. Next, it calculates the necessary changes in schedule for the particular request, at the same time minimizing the overall number of changes in the schedules. After that, it installs the schedules to the particular nodes. Then, the node checks if the QoS is satisfied by the provided schedules. If so, then it declares success and the Network Manager closes the request issue. If the QoS is not satisfied, the node keeps sending requests to the Network Manager until the requirements are met.

By means of collision free schedules and deterministic flows (tracks), the central scheduler can create deterministic behavior for time-critical applications, while distributed scheduling mechanisms can maintain connectivity for scalable and dynamic monitoring applications.

2) LOCAL SUPPORT FOR CENTRALIZED SCHEDULING

In the second approach, centralized scheduling is responsible for managing the whole application traffic, while distributed

scheduling is only used for bootstrapping the network and maintaining connectivity during unsteady network behavior. For this purpose, we limit the distributed scheduling algorithm such that it can only create a single outgoing cell and update it in case of any change in the network. This cell will allow nodes to be more responsive against network dynamics and temporary fluctuations via local decisions, and keep nodes connected while the centralized scheduling is adapting the schedules according to the latest changes.

3) INFRASTRUCTURE-BASED 6TiSCH NETWORKS

In this scenario, the solution targets continuous and guaranteed connectivity for more dynamic networks with possibly mobile nodes. By means of pre-deployed fixed nodes, a wireless infrastructure is created as presented in Figure 6. The communication among infrastructure nodes is managed by a centralized scheduling entity, thus latency-bounded and more controlled. On the other hand, the communication between mobile nodes and infrastructure nodes is managed in a distributed manner. Thanks to the static and deterministic behavior in infrastructure communication, each node can predict how long it will take for their message to get delivered.

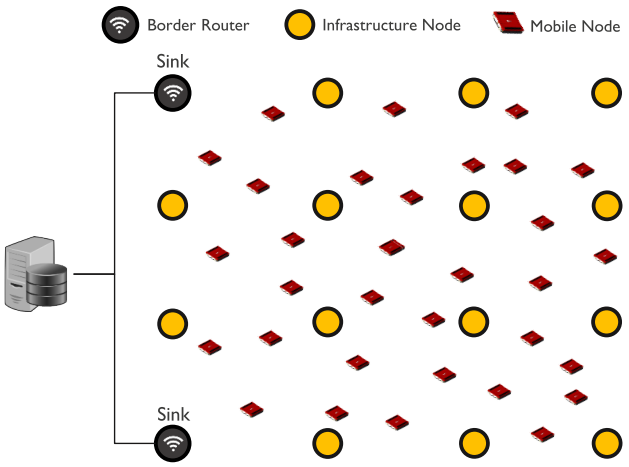

FIGURE 6. Infrastructure-based 6TiSCH networks.

4) CENTRALLY ASSISTED DISTRIBUTED SCHEDULING

In the last approach, distributed scheduling mechanism will be the main control unit which locally decides which slot-offset and channel-offset to use in order to communicate with parent and child nodes. However, there will be a centralized scheduling entity which maintains a comprehensive view of the network status and performance and assists the distributed scheduler on each node based accordingly. Moreover, one can also bring the credit and quota concept in the 6TiSCH network, where the number of incoming and outgoing schedules (locally decided) of each node can be controlled by the central unit. This way, the central unit can perform traffic shaping and distribute traffic load to different nodes in order to eliminate congestion loss or improve battery performance and life-span of the network nodes. However, it will not offer any deterministic network behavior.

IV. CoMI-BASED HYBRID SCHEDULING: IMPLEMENTATION AND PRACTICAL ANALYSIS

A. 6TiSCH CoMI INTERFACE

As CoMI follows a client-server architecture, it necessitates that each 6TiSCH node holds a CoMI Server which interacts with the Client residing at any centralized management or scheduling entity (called PCE). This will allow management entities to collect data via exposed resources and install new schedules via interfaces defined by standardized data sets specified in YANG.

For this purpose, we defined a YANG model for 6TiSCH Interface that contains all fundamental resources which we consider as required for basic schedule management. The overview of all implemented resources is provided in Table 2, together with their type, schema item identifier (SIDs), URI, allowed CoAP methods and short description. As it is presented in this table, the designed 6TiSCH CoMI interface provides access to TSCH schedules, neighbor and routing tables and enables the configuration of distributed schedulers. The resources for the schedule list and routing table are exposed with read and write access which allows management entities to install schedules and routes, while neighbor table resources are read-only.

For each resource, we used a custom SID which is within the range of SIDs reserved for RFCs (1000-59000) but not assigned yet [24]. Another reason to choose these SIDs is that their base64 encodings result in a URI with length of 2 characters (4001:"-h") rather than 3.

Regarding TSCH schedules, the CellList resource holds the list of the TSCH cells which can be queried via CellID. Each cell exposes resources providing information about the slot offset, channel offset, link options (transmit/receive, shared, hard/soft etc.), node address and cell performance statistics. The PCE can retrieve the hard/soft cells by sending a Get request to the CellList resource "/c/-h" or it can install a TSCH schedule (hard cell) by sending a Put request to a cell with a certain CellID (/c/-h?k = 2), together with all cell data in payload encoded in CBOR.

An example CoMI interaction for reading the Cell with id (CellID) 2 is provided below:

```
REQ: GET coap://<ip>:5683/c/-h?k=2
RES: 2.05 Content(format:application/
     YANG-Patch+cbor)
{
    +1 : 2,      // SID 4002
    +2 : 0,      // SID 4003
    +3 : 2,      // SID 4004
    +4 : 3,      // SID 4005
    +5 : 0x80    // SID 4006
    +6 : 0xae    // SID 4007
    +7 : 87      // SID 4008
    +8 : 123     // SID 4009
}
```

TABLE 2. 6TiSCH CoMI interface resources.

Resource		URI Query	Type	SID	URI	CoAP Methods	Description
CellList		CellID	cell[]	4001	"c/-h"	Get/Fetch/Post/ Put/iPatch/Delete	The list of the cells on the node
	CellID		uint8	4002	"c/-i"	Get/Put	The ID of the cell which can be used to query cells
	SlotframeID		uint8	4003	"c/-j"	Get/Put	The ID of the Slotframe which the cell belong to
	SlotOffset		uint8	4004	"c/-k"	Get/Put	The slot-offset of the cell
	ChannelOffset		uint8	4005	"c/-l"	Get/Put	The channel-offset of the cell (0..15)
	LinkOptions		bits[8]	4006	"c/-m"	Get/Put	0: transmit, 1: receive, 2: shared, 3: timekeeping, 4: priority, 5: soft/hard, 6-7: reserved
	NodeAddress		addresstype	4007	"c/-n"	Get/Put	The last byte of the targeted node's address
	StatisticsValue		uint16	4008	"c/-o"	Get	Communication statistics for the cell
	diffASN		uint16	4009	"c/-p"	Get	ASN passed since last packet exchange in the cell
NeighborList		NeigborID	neighbor[]	4011	"c/-r"	Get/Fetch/Delete	The list of neighbors
	NeigborID		uint8	4012	"c/-s"	Get	The Neighbor Id which is used to query neighbors
	Node Address		addresstype	4013	"c/-t"	Get	The last byte of the neighbor address
	RSSI		uint8	4014	"c/-u"	Get	Received Signal Strength stats for the neighbor
	LinkQuality		uint8	4015	"c/-v"	Get	Link Quality stats for the neighbor
	diffASN		uint16	4016	"c/-w"	Get	ASN passed till last packet exchange with neighbor
RouteList		RouteID	route[]	4021	"c/-1"	Get/Fetch/Post/ Put/iPatch/Delete	The list of Routes on the node
	RouteID		uint8	4022	"c/-2"	Get	The Route Id which is used to query route entries
	NodeAddress		addresstype	4023	"c/-3"	Get/Put	The last byte of the parent address
	DagRank		uint8	4024	"c/-4"	Get/Put	The DAG Rank of the node
	RankIncrease		uint8	4025	"c/-5"	Get/Put	The rank increase relative to parent node's rank
	RouteType		bits[8]	4026	"c/-6"	Get/Put	0: soft/hard, 1-7: Reserved
Scheduler List		SchedulerID	scheduler[]	4027	"c/-7"	Get	The list of Scheduling Functions
	SchedulerID		uint8	4028	"c/-8"	Get	The id of the scheduling function
	Status		bits[8]	4029	"c/-9"	Put/Get	0: Active/Inactive
	Quota (Incoming)		uint8	4030	"c/–"	Put/Get	Number of incoming schedules
	Quota (Outgoing)		uint8	4031	"c/-_"	Put/Get	Number of outgoing schedules

```
The CBOR encoding of the Response
Message (21 bytes):
A8              # map(8)
   01           # unsigned(1)
   02           # unsigned(2)
   02           # unsigned(2)
   00           # unsigned(0)
   03           # unsigned(3)
   02           # unsigned(2)
   04           # unsigned(4)
   03           # unsigned(3)
   05           # unsigned(5)
   41 80        # bytes(1)   ''\x80''
   06           # unsigned(6)
   41 AE        # bytes(1)   ''\xAE''
   07           # unsigned(7)
   18 57        # unsigned(87)
   08           # unsigned(8)
   18 7B        # unsigned(123)
```

On the other side, the NeighborList resources expose the neighbor table together with link quality metrics for each neighbor and RouteList represents the routing table which can be used to retrieve the node's parent in RPL and also install centrally calculated routes. By sending a CoAP Observe request on the RouteList (/c/−1) resource, the central management entity can monitor any topology changes in the network. By means of these interfaces, any application with a CoMI Client can monitor the network (neighbor tables, link qualities etc.), while creating any kind of schedules by using CoAP methods.

In addition, our 6TiSCH CoMI interface also allows remote management of the distributed schedulers on the nodes by means of SchedulerList (/c/−7) resources. By setting/unsetting the first bit of the *Status* resource, a scheduling function can be activated or deactivated. Moreover, it is also possible to set/unset limitations, named quota, for incoming and outgoing cells for certain schedulers. By means of this functionality, we obtain Centrally Assisted Distributed Scheduling in 6TiSCH networks, where the PCE can perform traffic shaping and distribute traffic load to different nodes in order to eliminate congestion loss or improve battery performance and lifespan of the network nodes. Moreover, this functionality is also used in the scenario "Local Support for Centralized Scheduling" by limiting the number of outgoing soft cells to 1.

B. IMPLEMENTATION DETAILS

In this section, the implementation details of the hybrid scheduling mechanisms and the CoMI 6TiSCH Interfaces are presented. For the implementation, we used the OpenWSN OS [26] at the node side, which is an open-source reference implementation for the 6TiSCH networking stack. Currently, this platform includes TSCH, some 6TiSCH related components (6top, SF0), and relatively simple and incomplete implementations of the higher layers of IETF stack (RPL, 6LoWPAN and CoAP). However, it does not implement any interface for remote network and schedule management and it does not support deterministic flow operations as defined in the 6TiSCH architecture. Our CoMI 6TiSCH interface implementation consists of 4 main components, namely CoAP Server, CBOR Handler (decoder and encoder), Base64 Converter and Six-TiSCH Interfacing (mapping between CoMI resources and their counterpart in 6TiSCH stack). For the CoAP Server, we leveraged on the simple CoAP implementation in OpenWSN and extended it with simple CoAP Observe and Block-Wise Transfer functionalities, as these are crucial for our remote schedule management platform.

In addition to the implementation for the OpenWSN nodes, we also need to implement another CoMI interface in the OpenVisualizer, which is used for plugging OpenWSN networks into the Internet, in order to monitor and manage schedules of the connected Border Routers.

Concerning the centralized network manager, we implemented a Centralized Schedule Management Platform which can monitor network topology and states, manage 6TiSCH nodes and install schedules by means of CoMI interfaces. This platform can be easily extended by any network scheduler algorithm. The details about the implementation of the CoMI-based schedule management are presented in Figure 7.

As the nodes become connected by means of distributed scheduling (SF0 implementation in OpenWSN), they first register their resources to a CoAP Resource Directory via a registration interface. This way, the centralized network manager discovers each node with schedule related CoMI resources of resource type (rt): 'core.c.datanode', with path: '/c'. After the registration and discovery process, the centralized network manager starts managing 6TiSCH resources as defined in the Table 2. Currently, our CoMI interface implementation is only supporting CoAP Get, Post, Put and Delete methods, but Fetch and Patch, which might offer more efficient operations in partial read and update of a resource, are not supported yet.

C. COST ANALYSIS

Centrally managing a 6TiSCH Network means monitoring the network topology and state (neighbor table and statistics, routing table, optionally schedule list), calculating the communication schedules and routes, and finally installing the schedule and routes on the nodes. So each operation will consume network resources. Therefore, in this section,

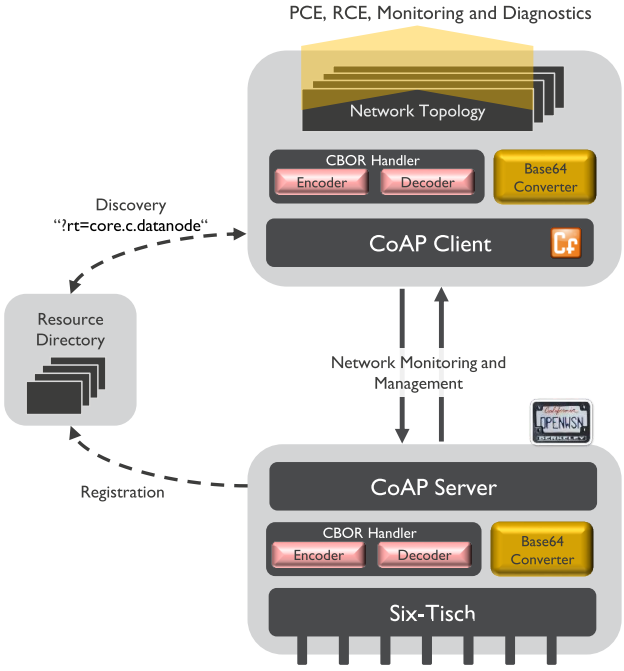

FIGURE 7. The implementation of CoMI based scheduling.

we examine the cost of the management operations of our CoMI-based Centralized Management System.

As the total management cost of a network will heavily depend on the number of nodes, the network topology and the utilized scheduling algorithm, we will only calculate the amount of data and number of packets (means channel resources) that need to be exchanged per operation. As such, the goal of this analysis is not to evaluate the overall performance of centralized scheduling, but instead to try to understand the cost of different remote management operations, which should be taken into account in the design phase of Hybrid Scheduling mechanisms. So, a management entity can consider the gain and cost of each operation and decide to perform it or not. Actually, Livolant *et al.* [27] presents a similar analysis only for schedule installation and discusses the efficiency of installing and updating a communication schedule in a 6TiSCH network with a central scheduler. However, they were considering an optimized scheduling mechanism, so a single change in the topology results in changes in the schedules of all of the nodes, so that creates an extensive number of scheduling packets. With a heuristic approach, the scheduling overhead would be a lot less. Secondly, they were considering an older version of CoMI which was less efficient than the current version that we use. Finally, our hybrid approach considers different approaches for centralized scheduling, which requires different and lower amount of data to be exchanged.

For these cost analysis, we analyzed possible CoAP packet formats (with payload attached) and how much payload can be carried by a single packet in different CoAP configurations. Table 3 provides the packet formats, including details of the overhead for each network layer (the header

TABLE 3. CoMI packet format for different operations.

	IEEE802.15.4e TSCH	IP (6LowPAN)	UDP (6LowPAN)	CoAP Header	CoAP Option			Payload Marker	Payload
					URI	Block	Uri Query		
Monitoring (Resp.)	29	19	2	4	0	0	0	1	72
Monitoring (Resp.&Block)	29	19	2	4	0	2-3	0	1	64
Installation (Req.&Query)	29	19	2	4	5	0	3-5	1	62-64
Installation (Req.&Block)	29	19	2	4	5	2-3	0	1	64
Installation (Req.&Query&Block)	29	19	2	4	5	2-3	3-5	1	32

*Numbers indicate the number of bytes in each field.

sizes used are the same as in [27]) of the 6TiSCH protocol stack. Due to the Maximum Transmission Unit (MTU) defined by IEEE802.15.4, a transmitted frame can consist of at most 127 bytes and 29 bytes of this is occupied by the IEEE802.15.4e TSCH MAC Header. After applying 6LoWPAN compression, the compressed IP and UDP headers constitute 19 and 2 bytes of each frame as in [27]. After these headers, there is only 77 bytes left for the application layer, i.e. the CoAP part. For the CoAP part, in addition to 4 bytes of the basic CoAP header, we consider different CoAP options (URI, URI Query and Block option) which are required in different configurations and the 1 byte of payload marker (0xFF). In every CoMI request, two URI options (e.g /c and /-h) are needed in order to define the target CoMI resource which results in 5 bytes of overhead. This option is not carried in the response messages. Secondly, if a request targets a queried resource, then it will bring at least 3 bytes of overhead (e.g. ?k = 8) which depends on the query size. And finally, in case of CoAP Block-Wise transfer, there will be another 2 or 3 bytes overhead for each block transfered in different frames. Considering all of these header overheads, the remaining payload size which can be carried in a single frame is also provided for each configuration.

These calculations show that a response message (e.g. topology monitoring) can carry at most 72 bytes, but if the payload is larger than that, then a 64B block size can be used to transfer these packets. In installation packets (requests), a single frame can carry at most 64 bytes of payload and a maximum of 32 byte block size can be used to transfer larger payload sizes.

Based on these packet formats, we calculated how many packets need to be carried for each operation as presented in Table 4. The first column represents the CBOR bytes to be delivered in order to perform each action. The second column shows the number of blocks or packets required in order to transfer the given amount of data and finally the last columns shows the number of messages which also represents the corresponding request and confirmation packets. One important remark here is that this number of messages only includes the number of messages exchanged between the CoMI client and the resource. However, the actual number of messages will be equal to the given numbers multiplied by the depth of the 6TiSCH node in the network.

TABLE 4. Number of messages required for network monitoring and management operations.

	CBOR (bytes)	Number of Blocks	Number of Messages
Reading Single Neighbor	12-16	1	2
Reading Neighbor Table (20)	241-321	4-6	8-12
Reading Single Route	13-16	1	2
Reading Route List (20)	261-321	5-6	10-12
Reading Single Cell	21-25	1	2
Reading Cell List (20)	421-501	7-8	14-16
Installing Single Cell	13-14	1	2
Installing Cell List (10)	131-141	3	6
Installing Cell List (20)	261-281	5	10
Installing Cell List (50)	651-701	11	22
Installing Single Route	11-13	1	2
Installing Route List (20)	221-261	4-5	8-10
Set/Read SF Configuration	10-13	1	2

V. EXPERIMENTAL EVALUATION

In order to illustrate the operation of the CoMI-based 6TiSCH Schedule Management, the coexistence of heterogeneous 6TiSCH networks and the possible contribution of the central schedule management into the deterministic and latency-bounded behavior, we performed an experimental study. However, the detailed network behavior and performance in case of various topologies, larger and more dynamic settings is not in the scope of this experimental evaluation and left for the future studies. In our experiments, we used pre-calculated and optimized schedules for the centralized scheduling and installed these schedules on the relevant nodes by means of our 6TiSCH CoMI Interface.

In these experiments, the testbeds (OpenWSN) are deployed on the OpenMote-cc2538 motes in an indoor environment with network topology provided in Figure 8. This topology consists of 1 backbone router and 4 monitoring nodes which can generate periodical non-critical data and a source (alarm) and destination (vital machine) mote for critical application data. Such heterogeneous network structures can be seen commonly in industrial setups. We used 10ms time slot duration with a slotframe length of 21 including 2 shared and 4 serial slots.

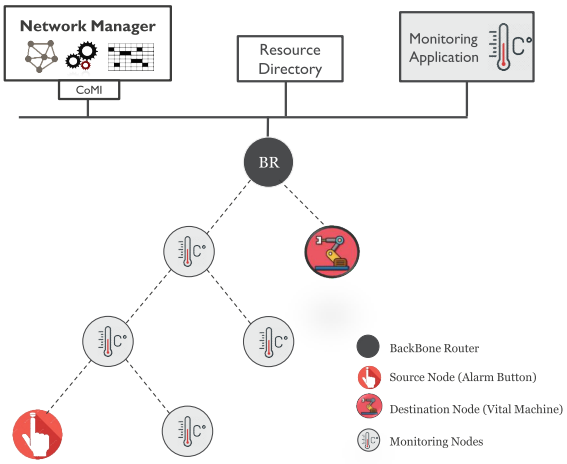

FIGURE 8. The network topology for the experimental setup.

For the experiments, we applied a fixed scenario on three different communication schemes. In each scheme, we first executed the network with only time-critical application traffic (1 packet/sec) between source and destination and collected end-to-end latency values. Afterwards, we triggered the periodic generation of monitoring data (1 packet/sec) by dedicated monitoring nodes, destined to a monitoring application in the backbone network. In the first scheme, the motes were only using 6top protocol (with SF0) in order to calculate their schedules in a distributed manner based on routes from RPL protocol. Secondly, the combination of distributed and centralized scheduling is used. After the network initiation, the pre-calculated optimized schedules are installed on the motes where time-critical data traffic would occur. This operation created a number of Hard Cells which can be used for any type of application traffic. Finally, the deterministic flows are used with the combination of distributed scheduling, where we limited the use of hard cells to certain traffic flow in order to create a deterministic flow for time-critical data. Therefore, any monitoring data or control message can not use these cells.

For each networking scheme, the end-to-end latency values for 500 time-critical packets are presented in Figure 9 and the yellow line represents the initiation of monitoring data after the packet 250.

As it is demonstrated in the performance trends in Figure 9, the network with only distributed scheduling was performing relatively worse in terms of latency, determinism and reliability. Since the schedules are calculated locally without the knowledge of overall network status and health, the end-to-end latency values for any kind of data is relatively higher than other schemes. Especially after the insertion of the monitoring data, we encounter a notable deviation in the latency performance and even a significant amount of packet loss for time-critical data.

In the second scheme, where we installed schedules with optimized path delays, the minimum and average end-to-end latencies have already improved remarkably compared to the first scheme. As it is presented in Figure 9b, the network was

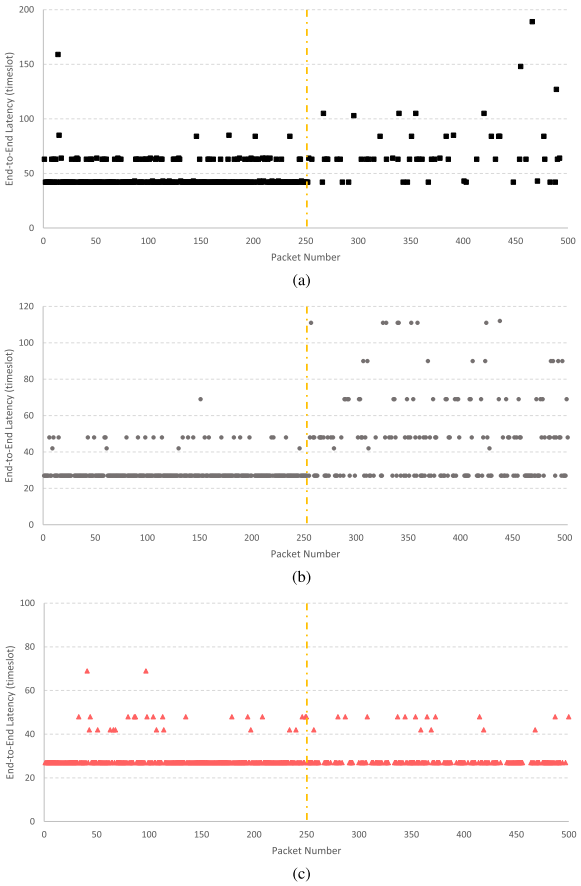

FIGURE 9. End-to-end latency values for time-critical data. (a) With distributed scheduling. (b) With hybrid scheduling. (c) With hybrid scheduling (Deterministic flows).

able to perform very well in terms of time-bounded packet delivery until the insertion of monitoring data. Nevertheless, the time-critical traffic faced a performance drop and consequently the latency and jitter of the packets has escalated with the initiation of monitoring data.

Concerning last approach, where we created dedicated and deterministic flows for time-critical data, the network was able to perform very well in the course of the whole experiment. As Figure 9c demonstrates, the QoS for the time-critical data traffic was not influenced by the non-critical data communication. Therefore, the network was still able to provide latency-bounded and reliable communication for time-critical traffic even with other inserted traffic across the network.

These performance trends show that the centralized scheduling can provide better-performance schedules and improve the latency and reliability in the 6TiSCH networks, compared to the distributed scheduling. However, centralized scheduling alone cannot ensure any performance guarantee or time boundary as usually desired for time-critical applications. Therefore, deterministic flows with dedicated paths and reserved resources are still crucial in order to create networks with deterministic behavior without any influence of other network traffic.

VI. CONCLUSION

In this article, we investigate the joint coordination and interaction of distributed and centralized scheduling mechanisms for 6TiSCH networks which will allow the coexistence of time-sensitive and scalable Industrial IoT Applications. We study the fundamental functionalities for a Hybrid Schedule Management and also describe network scenarios and approaches where such an approach can be used.

In addition, we present the details about the design and implementation of first 6TiSCH Centralized Scheduling Framework based on CoMI, which allows management entities to collect data and install new schedules via interfaces defined by uniform data models. We also provide cost analysis where we study the amount of data and channel resources that is needed to be exchanged per operation. Finally, we provide an experimental study which illustrate the operation of the CoMI-based 6TiSCH Schedule Management, the coexistence of heterogeneous 6TiSCH networks and the contribution of the central schedule management into the deterministic and latency-bounded behavior. This evaluation showed that, in a hybrid scheduling approach, distributed schedulers can manage the schedules for non-critical communication, while the centralized scheduling can improve the network performance via schedule optimization or create deterministic paths when QoS is a concern.

As a future work, we will further investigate the integration of the already existing centralized scheduling algorithms in our hybrid management platform and analyze their performances in various network topologies with larger and more dynamic settings. Secondly, we will study automatic application requirement determination (mainly focusing on 6TiSCH) in industrial applications and application-aware scheduling for 6TiSCH Networks.

REFERENCES

[1] *Industry 4.0: Challenges and Solutions for the Digital Transformation And Use of Exponential Technologies*, Deloitte AG, Zürich, Switzerland, 2015.

[2] T. Sauter, "Accessing factory floor data," in *Industrial Communication Technology Handbook*, vol. 8, 2nd ed. Boca Raton, FL, USA: CRC Press, Jun. 2014, pp. 9–10.

[3] *Industrial Communication Networks—Wireless Communication Network and Communication Profiles-Wirelesshart*, document IEC-62591.

[4] *Industrial Networks—Wireless Communication Network and Communication Profiles—ISA 100.11a*, document IEC-62734.

[5] *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*, IEEE Standard 802.15.4e, 2012.

[6] P. Thubert, *An Architecture for IPv6 Over the TSCH Mode of IEEE 802.15.4*, IETF Standard draft-ietf-6tisch-architecture-14, Apr. 2018.

[7] R. Yu and T. Watteyne, *White Paper: Reliable, Low Power Wireless Sensor Networks for the Internet of Things: Making Wireless Sensors as Accessible as Web Servers*, Dust Networks Product Group, Linear Technology, Milpitas, CA, USA, 2013.

[8] P. Zand, S. Chatterjea, J. Ketema, and P. Havinga, "A distributed scheduling algorithm for real-time (D-SAR) industrial wireless sensor and actuator networks," in *Proc. ETFA*, Sep. 2012, pp. 1–4.

[9] A. Tinka, T. Watteyne, and K. Pister, "A decentralized scheduling algorithm for time synchronized channel hopping," in *Proc. Int. Conf. Ad Hoc Netw.*, J. Zheng, D. Simplot-Ryl, and V. Leung, Eds., 2010, pp. 201–216.

[10] P. Thubert, M. R. Palattella, and T. Engel, "6TiSCH centralized scheduling: When SDN meet IoT," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2015, pp. 42–47.

[11] M. Veillette, P. Van der Stok, A. Pelov, and A. Bierman, "CoAP Management Interface," document draft-ietf-core-comi-02, IETF Internet-Draft, Jul. 2017.

[12] C. Bormann and P. E. Hoffman, *Concise Binary Object Representation (CBOR)*, document RFC 7049, IETF, Oct. 2013.

[13] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura, *6TiSCH 6top Scheduling Function Zero/Experimental (SFX)*, IETF Standarddraft-ietf-6tisch-6top-sfx-00 , Sep. 2017.

[14] Q. Wang, X. Vilajosana, and T. Watteyne, *6top Protocol (6P)*, document draft-ietf-6tisch-6top-protocol-09, IETF Internet-Draft, Oct. 2017.

[15] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 455–470, Dec. 2015.

[16] M. R. Palattella *et al.*, "On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, Jan. 2016.

[17] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. 13th ACM Conf. Embedded Networked Sensor Syst.*, New York, NY, USA, 2015, pp. 337–350

[18] 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 *Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2012, pp. 327–332.

[19] 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," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3655–3666, Oct. 2013.

[20] R. Soua, P. Minet, and E. Livolant, "MODESA: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks," in *Proc. IEEE 31st Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2012, pp. 91–100.

[21] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[22] M. Veillette, A. Pelov, A. Somaraju, R. Turner, and A. Minaburo. (Feb. 2018). *CBOR Encoding of Data Modeled With YANG*. [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-yang-cbor-06

[23] S. Josefsson, *The Base16, Base32, and Base64 Data Encodings,* , document RFC 4648, IETF, OCt. 2006.

[24] M. Veillette and A. Pelov, *YANG Schema Item iDentifier (SID)*, IETF Standard draft-ietf-core-sid-04,Jun. 2018.

[25] A. Karaagac, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Time-critical communication in 6TiSCH networks," in *Proc. 1st Int. Workshop Time-Critical Cyber Phys. Syst. (TC-CPS)*, Apr. 2018, pp. 161–166.

[26] T. Watteyne *et al.*, "OpenWSN: A standards-based low-power wireless development environment," *Trans. Emerg. Telecommun. Technol.*, vol. 23, no. 5, pp. 480–493, 2012.

[27] E. Livolant, P. Minet, and T. Watteyne, "The cost of installing a 6TiSCH schedule," in *Proc. Int. Conf. Ad-Hoc Netw. Wireless*, 2016, pp. 17–31.

ABDULKADIR KARAAGAC received the master's degree in communication systems from EPFL, Lausanne, Switzerland, in 2013, with specialization in networking and mobility. During his studies, he had opportunity to do internships in well-known technological companies, Logitech Europe SA and ABB Corporate Research. During these placements, he was working as a part of the R&D teams that are responsible for technological innovation of future products. From 2013 to 2016, he was working in building and HVAC automation industry, where, besides being a co-owner of the company, he was responsible for all of the processes regarding building automation applications.

He is currently pursuing the Ph.D. degree with the Internet and Data Laboratory Research Group, Department of Information Technology, Ghent University, Ghent, Belgium. His current research interests include the Internet of Things, semantic interoperability, the Web of Things, wireless sensor networks, and localization technologies.

INGRID MOERMAN received the Degree in electrical engineering and the Ph.D. degree from Ghent University, Ghent, Belgium, in 1987 and 1992, respectively.

She became a part-time Professor with Ghent University, in 2000. She is currently a Staff Member with the Internet and Data Laboratory (IDLab), a core research group of imec with research activities embedded at Ghent University and the University of Antwerp, Antwerp, Belgium. She is coordinating the research activities on mobile and wireless networking and leading a research team of about 30 members with the IDLab, Ghent University. She has longstanding experience in running and coordinating national and EU research funded projects. At the European level, she is in particular very active in the future networks research area, where she has coordinated and is coordinating several FP7/H2020 projects, such as CREW, WiSHFUL, eWINE, and ORCA, and participating in other projects, such as Fed4FIRE, FORGE, FLEX, and Flex5Gware. She has authored or co-authored over 700 publications in international journals or conference proceedings. Her current research interests include Internet of Things, low-power wide area networks, high-density wireless access networks, collaborative and cooperative networks, intelligent cognitive radio networks, real-time software-defined radio, flexible hardware/software architectures for radio/network control and management, and experimentally supported research.

JEROEN HOEBEKE received the master's degree in engineering (computer science) from Ghent University, Ghent, Belgium, in 2002, and the Ph.D. degree in adaptive *ad hoc* routing and virtual private ad hoc networks in 2007.

Since 2002, he has been with the Internet and Data Laboratory (IDLab) Research Group, Department of Information Technology, Ghent University—imec, where he has been an Assistant Professor since 2014, leading research on mobile and wireless networks, IoT communication solutions, and embedded communication stacks. His expertise has been applied in a variety of IoT domains, such as logistics, Industry 4.0, building automation, healthcare, and animal monitoring. He has authored or co-authored over 90 publications in international journals or conference proceedings. His current research interests include solutions for realizing the Internet of Things covering wireless connectivity (802.11, 802.15.4, BLE, LoRa, and 802.11ah), standard-based solutions (IETF CoAP, IPv6, IPSO, and OMA LWM2M), distributed intelligence, robust wireless communication, deployment and self-organization of smart objects, application enablers, wireless diagnosis, realizing the Internet of Things covering wireless connectivity, standard-based solutions, distributed intelligence, robust wireless communication, deployment and self-organization of smart objects, application enablers, and wireless diagnosis.