This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

IETF 6TiSCH: A Tutorial

Xavier Vilajosana, *Senior Member, IEEE*, Thomas Watteyne, *Senior Member, IEEE*, Tengfei Chang, Mališa Vučinić, Simon Duquennoy, Pascal Thubert

***Abstract*—The IETF IPv6 over the TSCH mode of IEEE802.15.4e (6TiSCH) working group has standardized a set of protocols to enable low power industrial-grade IPv6 networks. 6TiSCH proposes a protocol stack rooted in the Time Slotted Channel Hopping (TSCH) mode of the IEEE802.15.4-2015 standard, supports multi-hop topologies with the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) routing protocol, and is IPv6-ready through 6LoWPAN. 6TiSCH has defined the missing control plane protocols to match link-layer resources to the routing topology and application communication needs. 6TiSCH has also defined a secure light-weight join processes combining link-layer security features (through Counter with CBC-MAC (CCM*)) with a secure joining procedure using the Constrained Application Protocol (CoAP). This tutorial provides a comprehensive overview of the 6TiSCH architecture and protocol suite, including the 6TiSCH Operation Sublayer (6top), the 6top Protocol (6P), and how it uses 6LoWPAN, IP-in-IP encapsulation, and RPL. This document is meant to be used both as a primer, and as a reference. It is tailored to the advanced researcher and engineer implementing and building upon IETF 6TiSCH specifications.**

I. Introduction

In the last decade, the Internet Engineering Task Force (IETF) has standardized a set of protocols to respond to the increasing demand for IP-enabled constrained devices. Several working groups have been created to design and develop standard specifications for devices to empower the IoT. These groups have targeted the integration of different link layer technologies to the Internet Protocol (IP) ecosystem, dealing with major limitations imposed by the underlying technologies in terms of payload size, memory footprint, computing capacity and non-trivial topologies, while ensuring IP-compliance.

The IETF IPv6 over the TSCH mode of IEEE802.15.4e (6TiSCH) Working Group (WG) was created to work on the standardization of the control plane and the IP layer adaptation on top of the IEEE802.15.4 TSCH link layer [1]. 6TiSCH has been one of the main efforts to bring IPv6 to industrial low-power wireless, bridging Time Slotted Channel Hopping (TSCH) networks with 6LoWPAN networks. This pioneering work has identified and addressed the many remaining challenges when building IPv6 on low capacity networks. It has become a glue for the different layers, and has triggered improvements in header compression, IP-in-IP encapsulation, and 6LoWPAN Neighbor Discovery, providing more capable routing schemes and security management consistently aiming at more efficiency and simplicity.

X. Vilajosana is with the Open University of Catalunya (UOC-IN3) and Worldsensing S.L, Spain.

T. Watteyne, T. Chang and M. Vučinić are with Inria, France.

S. Duquennoy is with the Swedish Institute for Computer Sciences (SICS), Sweden.

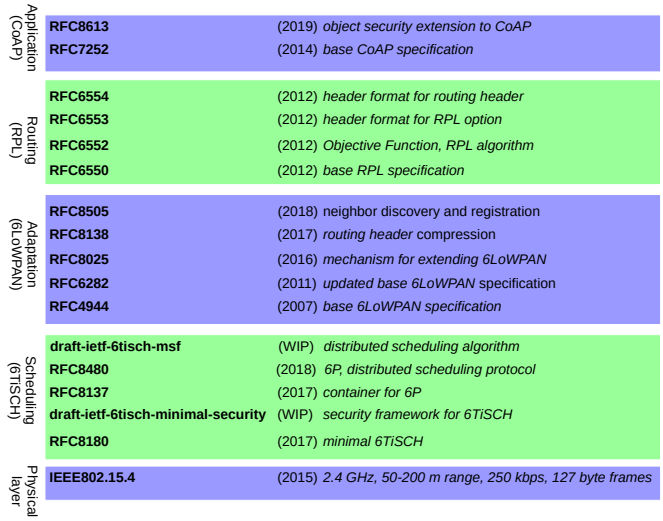

Fig. 1. The different standards forming the 6TiSCH Stack. Labels starting with *RFC* and *draft-* refer to published and work-in-progress IETF standards, respectively.

The IETF 6TiSCH protocol suite [2], [3] is rooted in the IEEE802.15.4 TSCH link layer, which it complements with a set of protocols, as described in the 6TiSCH Architecture [4]. Fig. 1 summarizes the main building blocks that form the core 6TiSCH stack. These protocols include a minimal profile for the bootstrapping of the network [5], a lean security mechanism to support secure join procedures [6], a network schedule management suite composed of a distributed cell management protocol [7], and a set of scheduling policies [8]–[11]. At the network layer, the 6LoWPAN framework provides stateless header compression (RFC4944 [12]) and contextual header compression (RFC6282 [13], RFC8025 [14] and RFC8138 [15]) and fragmentation through RFC4944 and upcoming updates from the IETF. Neighbour discovery is optionally provided by RFC8505 [16]. The Routing Protocol for Low Power Lossy Networks (RPL) [17] organizes the network in a multi-hop topology following an objective function (RFC6552) [18]. The suite also defined protocol extensions to transport routing information (RFC6553) [19] and enable downward source routes (RFC6554) [20]. 6TiSCH provides a mechanism by which the link layer topology is matched with the routing topology, enabling the nodes to maintain synchronization with their best-connected parent [5]. CoAP [21] and OSCORE [22] provide the tools to support a secure join process [6] as well as complement the stack, enabling low-overhead secure RESTful interaction.

IETF 6TiSCH combines the industrial performance of IEEE802.15.4 TSCH with an IPv6-ready upper stack, to form

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

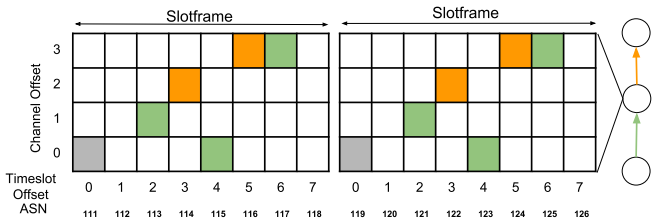

Fig. 2. A 6TiSCH network is synchronized; time is cut into timeslots. Timeslots are organized into slotframes which repeat over time.

a well-tuned and secure complete networking solution. Fig. 1 depicts the resulting protocol stack, and lists the different standards (some are still work-in-progress) operating at the different layers. We will refer to that figure throughout this tutorial.

Table I positions the technology with respect to other low power wireless industrial initiatives. 6TiSCH is rooted at the TSCH mac layer analogously to the WirelessHART and ISA100.11a standards and in opposition to other connectivity solutions such as Thread and Zigbee. As a main differences is the fact that schedule management is decentralized and coordinated as layer 2.5 mechanism in opposition to the application layer schedule management of similar technologies.

Several publications and implementations cover different parts of the 6TiSCH architecture, vision and stack. For example, our previous work [3], presents the 6TiSCH vision and early architecture when the working group was formed. Several ideas have been developed and materialized into standards. Others are still open and drive future direction within the 6TiSCH WG or novel groups at the IETF such as Reliable and Available Wireless (RAW). IEEE802.15.4 TSCH is supported in all the major open-source low-power wireless implementations, including OpenWSN, and Contiki(-NG), and details of its architecture can be found in different survey and evaluation papers [23], [24]. The OpenWSN project [25], [26] includes a step-by-step guide on how to implement TSCH and achieve tight time synchronization and ultra-low power operation. The base 6LoWPAN protocol is well described by Shelby and Bormann [27]. Several elements have recently been added to 6LoWPAN, including the interaction between 6LoWPAN and RPL [28], the 6LoWPAN Routing Header (6LoRH) [14], [15], and fragment forwarding [29], [30] and recovery [31]. CoAP is well documented [32] and several reference open-source implementations are available [33]. OSCORE, an object-security extension to CoAP, has recently been added to CoAP. Open source reference implementations of 6TiSCH such as OpenWSN [26] and Contiki(-NG) [34] are seen as valuable tools for the 6TiSCH community and are widely adopted by academia and industry to bootstrap their adoption of 6TiSCH technologies. 6TiSCH has been, as well, proposed as an educational tool since it integrates a complete, yet simple protocol stack, covering medium access, management plane, networking and transport features. The simple yet complete approach is claimed to be an appropriate tool to train students in networking courses [35].

The goal of this article is to serve as the reference document of the 6TiSCH stack, putting together the information contained in the fragmented set of documents listed above, and adding the standards that have been developed since their publication. This tutorial is tailored towards the practicing engineer and the advanced researcher, and contains both a comprehensive description of the standards, as well as key advice for implementing them. This article is meant as both a primer on 6TiSCH, and as a reference.

This tutorial takes a bottom-up approach, both describing the components, building blocks or protocols, but also giving insights from an implementation perspective. We start from the IEEE802.15.4 TSCH link layer in **Section II**, defining the operation of transmission and reception slots, and detailing of mechanisms such as network synchronization and link-layer security. We then describe in **Section III** how 6TiSCH manages the TSCH schedule, including how resources are allocated. **Section IV** is entirely dedicated to security, and covers both the secure join protocol, and how the network stays secure after nodes have joined. 6TiSCH is complemented by the use of several standards specifications part of the 6LoWPAN framework (IP header compression, fragmentation) and the Routing Protocol for Low Power and Lossy Networks (RPL). These are presented in **Sections V and VI**, respectively. After the technical review of the 6TiSCH standards, **Section VII** leverages our implementation experience and details the hardware requirements that need to be taken into account when implementing the stack. We provide hints on how to achieve tight synchronization, and recommend the use of hardware acceleration to perform costly encryption tasks. We also list the 6TiSCH implementations (both open-source and commercial) we are aware of. Section VIII presents the future directions of the 6TiSCH specification. Finally, **Section IX** concludes this tutorial.

II. IEEE802.15.4 TSCH

The IEEE802.15.4 Task Group has been driving the development of low-power low-cost radio technology. The Timeslotted Channel Hopping mode, added to the 2015 revision of the IEEE802.15.4 standard, is targeted at the embedded and industrial world, where reliability, energy consumption and cost drive the application space.

The IEEE802.15.4 physical layer has been designed to support demanding low-power scenarios targeting the use of unlicensed bands, both the 2.4 GHz and sub GHz Industrial, Scientific and Medical (ISM) bands. This has imposed requirements in terms of frame size, data rate and bandwidth to achieve reduced collision probability, reduced packet error rate, and acceptable range with limited transmission power. The PHY layer supports frames of up to 127 bytes. The Medium Access Control (MAC) sublayer overhead is in the order of 10-20 bytes, leaving about 100 bytes to the upper layers. IEEE802.15.4 uses spread spectrum modulation such as the Direct Sequence Spread Spectrum (DSSS). A 2012 amendment of the physical layer, known as IEEE802.15.4g has introduced new modulations including OFDM, supporting a large number of data rates and larger packet sizes up to 2047 bytes. This specification has been adopted by the Wi-SUN alliance [36] to address metering use cases.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

TABLE I
COMPARISON OF 6TISCH TO OTHER INDUSTRIAL STANDARDS.

	IETF 6TiSCH	**ANSI/ISA100.11a**	**WirelessHart IEC62591**	**Thread**	**Zigbee**
IP-Compliant	Yes	Yes	No	Yes	Yes
Access	TSCH	TSCH	TSCH	CSMA/CA	CSMA/CA
Scheduling	Descentralized	Centralized	Centralized	None	None
Routing	RPL	Mesh Under	Mesh Under	RIP	RPL
Control Plane	Cross-Layer	Application	Application	None	None
Security	One Touch	One Touch	One Touch	One Touch	One Touch
Transport	Connectionless (UDP)	Connectionless (UDP)	Connectionless Service	Connectionless (UDP)	Connectionless (UDP)
Application Layer	Restful (CoAP)	Software Objects	HART Universal Commands	Restful (CoAP)	Zigbee Device Objects and profiles

A. Time-Slotted Channel Hopping

As the core technique in IEEE802.15.4, TSCH splits time in multiple time slots that repeat over time. The structure is referred as a Slotframe. For each timeslot, a set of available frequencies can be used, resulting in a matrix-like schedule (see Fig. 2). This schedule represents the possible communications of a node with its neighbors, and is managed by a Scheduling Function (described in Section III-C). Each cell in the schedule is identified by its slotoffset and channeloffset coordinates. A cell's *slotoffset* indicates its position in time, relative to the beginning of the slotframe. A cell's *channeloffset* is an index which maps to a frequency at each iteration of the slotframe. Each packet exchanged between neighbors happens within one cell. An Absolute Slot Number (ASN) indicates the number of slots elapsed since the network started. It increments at every slot. This is a 5 byte counter that can support networks running for more than 300 years without wrapping (assuming a 10 ms timeslot). Channel hopping provides increased reliability to multi-path fading and external interference [37], [38]. It is handled by TSCH through a channel hopping sequence referred as *macHopSeq* in the IEEE802.15.4 specification. At each slot, the channel (CH) to be used by the radio is computed by Eq. (1), resulting in pseudo-random channel hopping. The channel hopping sequence, wraps every *macHopSeqLen* timeslots. The macHopSeqLen is defined in the IEEE802.15.4 configuration and corresponds to the number of elements in the *macHopSeq* list.

$$COUNTER = ASN \ + \ ch_{offset}$$
$$CH = macHopSeq[COUNTER \ mod \ macHopSeqLen] \quad (1)$$

Fig. 2 presents an example of an Slotframe repeating over time. The grey cell (0,0) is a control plane cell, defined by RFC8180 [5] and referred as the "minimal cell". The cell, as described in detail in Section III-A, is used to establish the initial schedule when the network boots. Green cells (2,1), (4,0) and (6,3) are used to receive packets from one of its neighbor. Yellow cells (3,2) and (5,3) are used for transmitting packet to another neighbor.

Timeslot timing. Fig. 3 presents the IEEE802.15.4 TSCH timeslot timing used by the transmitter and the receiver. Within a timeslot, there are a maximum of four timeouts used by the transmitter node. Assuming that the timeslot starts at time offset zero, timeout $T1$ is scheduled at the moment when the clear channel assessment (CCA) needs to be done, this is exactly at *TsCCAOffset* since the beginning of the timeslot. CCA is optional. The exact moment when the packet needs to be transmitted is *TsTxOffset*; timeout $T2$ must occur there so the radio can be enabled to transmit. Timeout $T3$ is scheduled to start listening for acknowledgment frames at a relative time offset after the end of frame event have been received. This relative offset is referred to as *TsRxAckDelay*. Timeout $T4$ is used to limit the time the radio keeps listening, i.e. for at most the duration of the Guard-time in the case an acknowledgment packet is not received. On the receiver side, the radio is turned on at exactly *TsRxOffset*, determined by timeout $R1$. Timeout $R2$ is used to determine how long the radio keeps listening in the case a packet is not received (idle-listening). After the reception of a packet, timeout $R3$ is scheduled to transmit the acknowledgment at an exact offset since the *endOfFrame* event has been received. This offset is referred to as *TsTxAckDelay*.

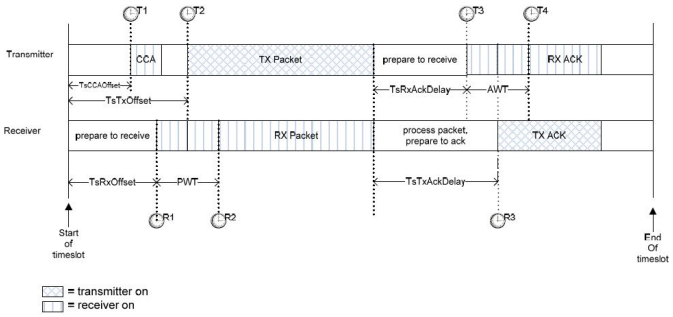

Fig. 3. An IEEE802.15.4 TSCH timeslot timing.

TX slot state machine. Fig. 4 shows the state machine of an transmit (TX) slot. As can be observed, different software components are involved in the slot activities. These include the `Schedule` database that indicates the type and target of the current slot, the `Queue` where packets to be sent are stored, the `Radio` as a radio driver enabling the data to be immediately sent, the `Timer` enabling the setting of precise alarms in the near future, the `Headers` providing helper functions to parse the IEEE802.15.4 headers, and the `ULMAC` encapsulating higher layer functionalities. When a timeslot starts, the node calls *getLinkDestination()* to get the associated link-layer destination from the `Schedule` database. Then it calls *getPacketToSend()* to check whether there is a packet

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

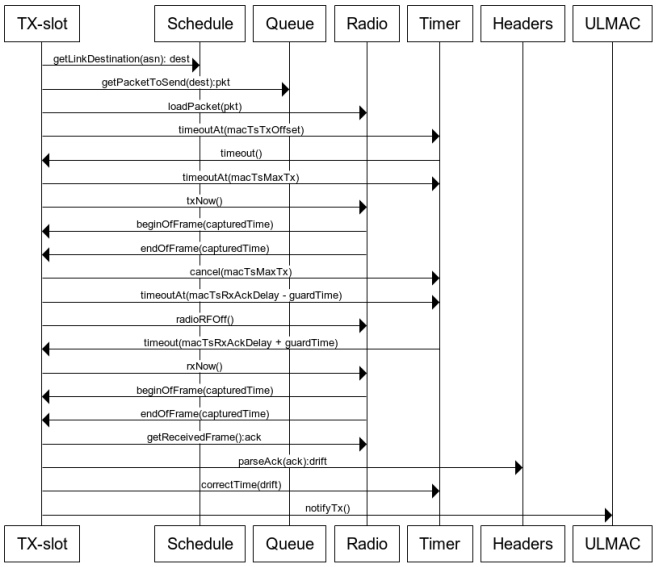

Fig. 4. TX slot state machine.

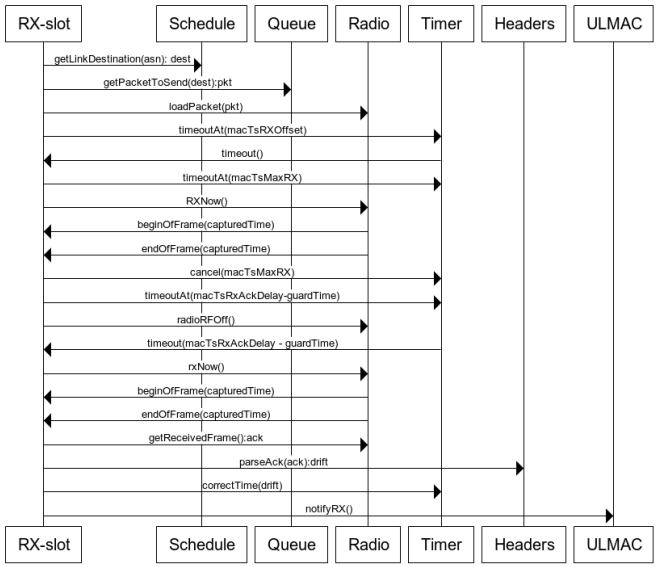

Fig. 5. RX slot state machine.

pending in the `Queue`. If so, the packet will be loaded into `Radio` by calling *loadPacket()*. An alarm is then scheduled by the `Timer` to send the packet at the exact offset within the slot, referred as *macTsTxOffset* by the standard. When the alarm expires, *txNow()* is called by the `Radio` to send the packet out. The *beginOfFrame* signal is captured by the `Radio` when the first bit of the packet leaves the radio. This event is used to capture the value of the time slot timer to timestamp precisely when this occurs within the slot. The *endOfFrame* signal occurs when the last bit of the packet leaves the radio. If the packet needs to be acknowledged, this is indicated in the IEEE802.15.4 header. The node schedules a timeout at *macTsRxAckDelay-guardTime* to switch its radio to reception mode. At this moment the radio is turned off waiting for the timeout to occur. When the timeout expires, the radio is turned on and configured in reception mode by calling *rxNow()*. A new timeout is scheduled at *macTsRxAckDelay+guardTime*, defining a listening period of two times the *guardTime*. If a packet is received before the timeout, the *beginOfFrame* signal first and *endOfFrame* signal later will be captured by the `Radio`. After the acknowledgement frame is received, its header is parsed and the time correction is obtained. The node applies the correction if the neighbor is its time source neighbor and notifies to the upper layers the status of the transmission by calling *notifyTx()*.

RX slot state machine. Fig.5 shows the state machine of an receive (RX) slot. The activities within a reception slot are similar to those in the transmission slot, but inverting the radio states: first receive a frame, then transmit an ACK. At the beginning of an RX Slot, the node calls *getLinkSource()* to know the expected source of the packet to be received. After, a timeout is scheduled to start listening. A node must listen at *macTsRxOffset* so as to receive the frame (Fig. 3. After receiving the frame, the `Radio` is configured to transmit an acknowledgment, if requested.

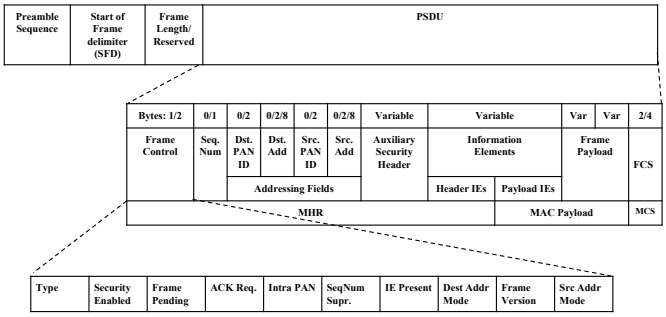

Fig. 6. Physical and MAC headers of an IEEE802.15.4 TSCH frame.

B. Packet Format

The IEEE802.15.4 header has been described in the literature and in the IEEE standard specification [1], [2]. We focus however in those fields that are relevant for a 6TiSCH implementation, detailing their use and configuration. Fig. 6 presents the IEEE802.15.4 MAC Header. The Frame Control field (FCF) is used to configure the frame attributes. The IE Present field is extensively used within the 6TiSCH control plane frames. When this bit is set, Information Elements (IEs) are transported by the frame, either as part of the header (Header IEs) or of the payload (Payload IEs). IEs as defined by IEEE802.15.4 are information placeholders that should be used to distribute information between communicating nodes. Header IEs are not encrypted, but may be authenticated. Payload IEs are both encrypted and authenticated.

The IEEE802.15.4 specification defines the list of available IEs. This tutorial focuses on IEs used by 6TiSCH, listed in Table II. A complete list of IEs can be found in the IEEE802.15.4 specification [1].

The IEEE802.15.4 header contains an auxiliary security header field, see Fig. 7. This security header is used to indicate the security configuration used for the frame. IEEE802.15.4 security is highly configurable supporting different levels of

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

TABLE II
IEEE802.15.4 INFORMATION ELEMENTS (IES) USED BY 6TISCH.

Name	Number	Type	RFC 8180	Description
Time Correction IE	0x1e	Header IE	No	Used by ACK to indicate time correction
Header Termination 1	0x7e	Header IE	Yes	Delimits Header IEs and Payload IEs
Header Termination 2	0x7f	Header IE	No	Delimits Header IEs and MAC Payload
TSCH Synchronization IE	0x1a	Payload IE Short	Yes	Contains ASN and Join Metric. According to RFC8180, the Join Metric value is set to DAGRank(rank)-1 when RPL is used.
TSCH Slotframe and Link IE	0x1b	Payload IE Short	Yes	Indicates the size and number of slotframes and announces the number of active cells in the advertised schedule. The location (timeslot offset, channel offset) is indicated for each active cell. RFC8180 recommends a single active cell.
TSCH Timeslot IE	0x1c	Payload IE Short	Yes	Timing template for a timeslot. According to RFC8180 use the default timeslot template. Only the timeslot template identifier is present with value 0 (macTimeslotTemplateId = 0).
Channel hopping IE	0x9	Payload IE Long	Yes	Channel hopping sequence description. According to RFC8180 use the default channel hopping template. Only the channel hopping template identifier is present with value 0 (macHoppingSequenceID = 0).
IETF IE	0x5	Payload IE Long	No	Reserved IE to transport IETF information (RFC8137)

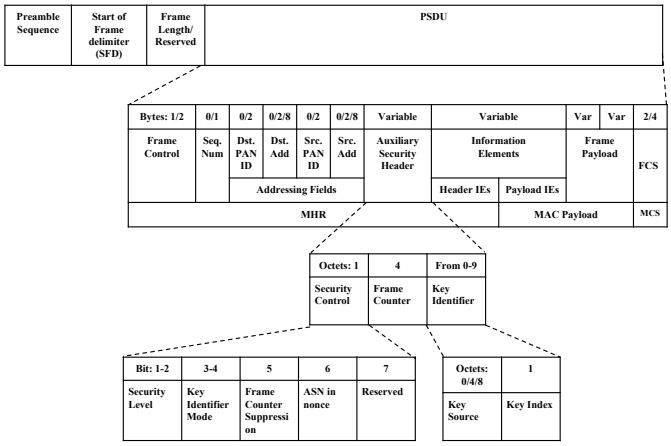

Fig. 7. Detail of the Auxiliary Security header of an IEEE802.15.4 TSCH frame.

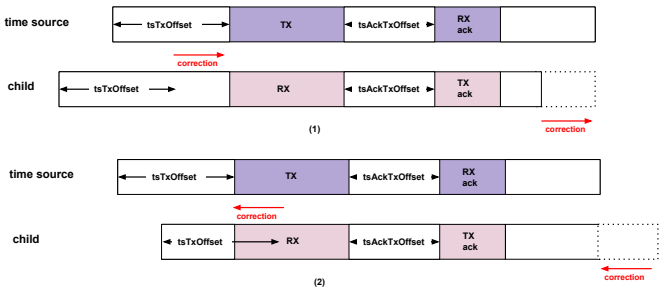

Fig. 8. Packet based synchronization diagram.

security. Satry et.al., [39] analyze those configurations and identify main security considerations. A Security Control Field is used to specify such a level, providing different levels of data authenticity (e.g., MIC-32) and optionally confidentiality (e.g. ENC-MIC-32). Other sub-fields enable the suppression of the Frame Counter in the Auxiliary Security Header. The suppressed Frame Counter signals the use of the implicit Absolute Slot Number (ASN) in the Nonce. The KeySource field is used to identify the originator of that key and hence delimit the scope of protection of the key. The Key index is the index of the used key among those stored from that originator.

The Nonce in IEEE802.15.4 TSCH is computed using the EUI64 (8 Bytes) and the ASN (5 Bytes). When the short address is used, the Nonce is composed of a composition of 8 B formed by the 3-byte long Company Identifier (CID), 1 padding byte (0x00), the network PANID, and the short address. This is appended with the ASN to form a 13 Bytes Nonce. The ASN in the Nonce is formatted using *big endian* byte ordering while the rest of the MAC header uses *little endian*.

IEEE802.15.4 uses the Advanced Encryption Standard (AES) in Counter with CBC-MAC (CCM*) mode to encrypt the frame and provide data authenticity. The inputs to the CCM* transformations are the secret key, the Nonce, the byte string representing the part of the frame that is encrypted (*m data*), and the byte string representing the part of the frame that is authenticated (*a data*). The IEEE802.15.4 standard details how these byte strings are formed. In the case of encryption, the output of the CCM* transformation is the ciphertext and the Message Integrity Code (MIC) that are appended before the Cyclic Redundancy Check (CRC). In the case of decryption, the output of the CCM* transformation is the plaintext and a signal on whether the MIC of the received frame matches the one generated locally.

C. Synchronization

As defined by IEEE802.15.4 TSCH, the synchronization process between a node and its time source neighbor is achieved through pairwise communication. A node synchronizes both when receiving a packet ("packet-based synchronization") or an acknowledgment (ACK) ("acknowledgment-based synchronization") from its time source neighbor, see Fig. 8.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

When using packet-based synchronization, the time source node sends a packet to its child at a precise offset from the beginning of the slot (`TsTxOffset`). When the child receives the packet, it timestamps the reception of the first bit and computes the time difference between the actual reception time and the ideal reception time (`TsTxOffset`). The child node then increases (Fig. 8(1)) or decreases ((2)) the duration of the current time slot to compensate for the synchronization error.

Acknowledgment-based synchronization follows a similar principle, only the synchronization error is carried explicitly in a field in the acknowledgment frame. A child node transmits a frame to its time source neighbor, which timestamps the time of arrival of that frame. The time source neighbor then indicates the synchronization error in an Information Element (Sync IE) in the acknowledgment it sends back. The child node compensates for that error by changing the duration of the current timeslot.

Previously published work describes this process in detail, including adaptive techniques to minimize the drift between nodes in the network [40]–[43].

III. 6TiSCH Management Plane

6TiSCH defines the *6TiSCH Operational sublayer* (6top), a thin 2.5 layer (above MAC) that provides management and control primitives to bind the asynchronous IP layer to the synchronous IEEE802.15.4 TSCH MAC. This section describes the 6top sublayer and how it manages the TSCH schedule.

A. Minimal Profile

The RFC8180 [5] defines a minimal 6TiSCH profile. This profile schedules the minimal bandwidth for network advertisement and join traffic, so the network can be formed. RFC8180 can also be used as a fallback mode of operation, i.e. if the dynamic scheduling fails, all the nodes can rely on the minimal configuration providing the minimal bandwidth to recover the network operation.

We call a "pledge" a node that wants to join a 6TiSCH network[1]. All the nodes already in the network regularly send Enhanced Beacons (EBs). When switched on, the pledge listens for EBs, thereby discovering the nodes around it. RFC8180 recommends to wait for NUM_NEIGHBOURS_TO_WAIT (with value = 2) EBs from different nodes, then select the node that will play the role of the Join Proxy (JP) during the secure join process. Which node is chosen as JP depends on connectivity metrics such as link quality and the value of the Join Metric present in the Synchronization IE within the EB. The policy recommends to select the neighbor with lowest Join Metric. The EB contains a set of payload Information Elements that enable the pledge to learn the minimal schedule and MAC layer configuration to use (see Table II indicating what IEs are recommended).

[1]The term "pledge" is used in IETF to designate a node that attempts to join a given network, but has not yet completed the join process and is therefore not trusted by the network. The term relates to a prospective member of a fraternity/sorority undergoing the "pledging" process.'

Through parsing these IEs, the joining node learns the configuration to use to proceed with the network and security bootstrap. RFC8180 recommends the use of a single shared cell for joining traffic (i.e. slotted aloha), leaving the size of the slotframe open to implementers, who can trade energy consumption for shorter join times. The recommended minimal cell is located at slot offset 0 and channel offset 0, the linkType is defined as ADVERTISING so EBs can be sent through it. The linkOptions is set with the flags transmission, reception, shared and timekeeping set. This converts this cell in a slotted Aloha link where any type of packet can be sent, i.e. Keep Alive (KA) packets for time synchronization, EBs, and data packets.

During the joining process the Pledge node synchronizes using EBs. While when joined uses KA packets.

If RPL is used, the non-storing mode is mandated (storing mode being optional). RFC8180 matches the link-layer topology and the routing topology using the Join Metric field, part of the Synchronization IE and transmitted in the Enhanced Beacon. The value used as Join Metric is a representation of the RPL rank as defined in RFC6550 obtained from Eq. (2). This ensures that a pledge picks a JP which is close to the root of the network, and which has a good chance of also being its RPL preferred parent.

$$JoinMetric = DagRank(Rank) - 1 \tag{2}$$

When using RPL, RFC8180 mandates the use of the Expected Transmission Count (ETX) [44] and the normalization guidance by the Objective Function Zero (OF0) [18]. The latter builds a network topology that minimizes the end-to-end (re)transmission count.

On top of the minimal profile, 6TiSCH supports dynamic scheduling, in which link-layer resources (cells in the TSCH schedule) are dynamically added/deleted to match the applications' communication requirements. Dynamic scheduling is divided in two conceptual parts. The 6top protocol (6P) [7] is a protocol allowing neighbor nodes to negotiate which cells to add/remove to/from their schedule. A Scheduling Function (SF) is the algorithm that decides when to add/delete cells, and trigger 6P negotiations. 6P and SFs are detailed in Sections III-B and III-C, respectively.

B. 6top Protocol (6P)

The 6P Protocol (6P) as defined in the RFC8480 [7], provides a pairwise negotiation mechanism to the control plane operation. The protocol supports agreement on a schedule between neighbors, enabling distributed scheduling. Table III lists the commands 6P supports.

Pairwise negotiation can be handled using 2-step and 3-step transactions. A 2-step transaction enables the requester node to select the cells to ADD, DELETE or RELOCATE. A 3-step transaction lets the receiver node to pick cells, allowing for example a parent to split a block of cells evenly among its children. The type of cells to be installed is defined by the CellOptions field. Fig. 9 depicts an example of a 3-step transaction to schedule 2 cells between two nodes. Node A issues a request to add 2 TX and Shared cells, without

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

TABLE III
THE COMMANDS SUPPORTED BY THE 6TOP PROTOCOL (6P).

Command	Code	Description
ADD	1	Add cell(s) between the two neighbors. The CellOptions bitmap indicates the type of cell(s) to add.
DELETE	2	Delete cell(s) from the schedule.
RELOCATE	3	Relocate cell(s) in the schedule. Used to handle schedule collisions.
COUNT	4	Count the cells with a particular CellOptions.
LIST	5	List the cells with a particular CellOptions.
SIGNAL	6	Placeholder for SF-specific commands.
CLEAR	7	Clears all cells between the two neighbors. Possibly used to handle schedule inconsistencies.

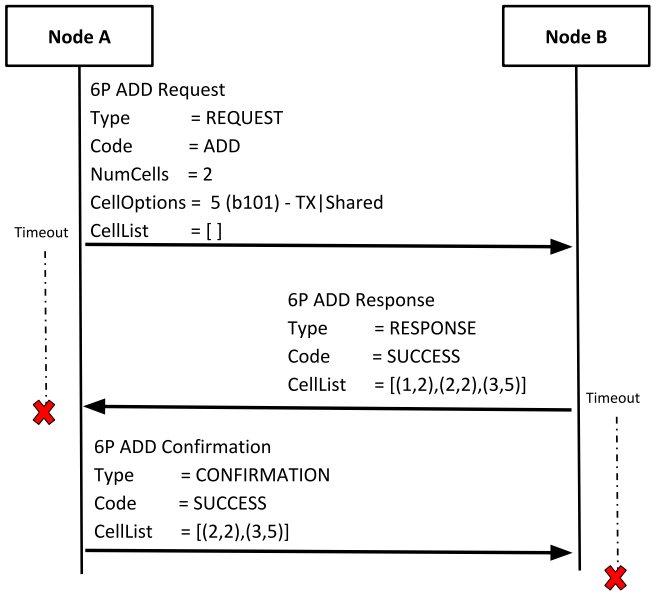

Fig. 9. A 3-step 6P transaction which results in 2 additional cells from node A to neighbor node B.

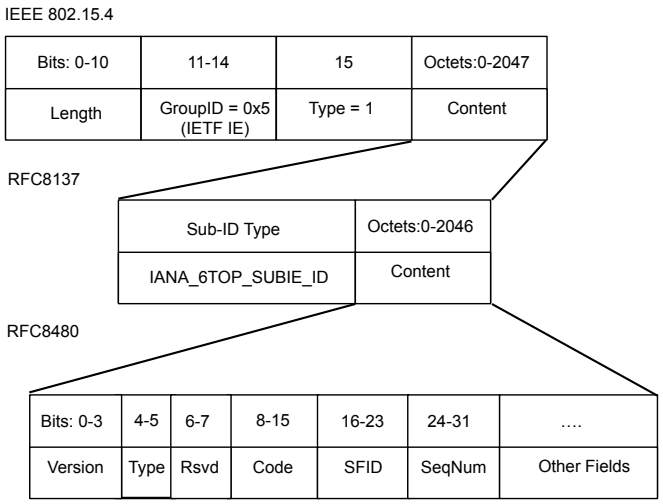

Fig. 10. 6P Information Element.

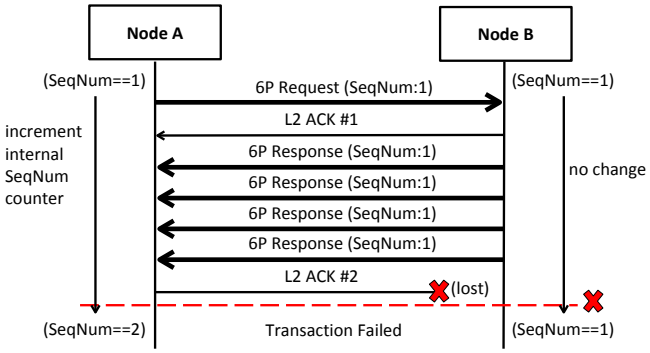

Fig. 11. 6P transaction failure.

specifying which ones; node B responds with 3 candidate cells; node A picks two of those. After the transaction, both nodes A and B have installed the agreed-upon cells in their schedule. In detail Node A installed 2 Cells (2,2) and (3,5) as TX Shared cells in its schedule. Node B in its turn have installed the same cells but with CellOptions RX and Shared.

6P commands are transported in Information Elements (IEs); they only travel a single hop. The Institute of Electrical and Electronics Engineers (IEEE) has reserved a Group ID for the IETF, referred as IETF IE (value `0x05`), and defined in RFC8137 [45]. 6P uses a sub-type on that Group IE (SUBID_6TOP) requested to IANA. Fig. 10 shows the encapsulation of the 6P IEs within the 6P packet. It shows the Payload IE header followed by the IETF IE SubHeader and the 6P protocol header. All 6P messages utilize the general header fields; additional fields are present depending on the command type.

A 6P transaction succeeds when its 2 or 3 steps execute correctly on both neighbor nodes. 6P uses a timeout to cancel a long-lasting transaction. Schedule inconsistencies are detected thanks to the use of a Sequence Number (SeqNum), and the detection of message loss when the last ACK is not received.

Fig. 11 illustrates a possible inconsistency situation when node A starts a transaction, but after the last Response, the link-layer ACK is lost. The figure also illustrates how the SeqNum is managed. In particular,the SeqNum is incremented when requester (node A) considers the transaction has succeeded. The SeqNum however is not incremented at the other side (node B), as the transaction failed due to the lost ACK. Upon detecting the lost ACK, node B applies the rule defined by the Scheduling Function (SF). This can be issuing a CLEAR command to reset all cells between A and B, or trying to correct the possible inconsistency through a rollback action. In the latter case, node B, may issue a LIST command after the timeout to identify the schedule differences and correct them through subsequent ADD, DELETE or RELOCATE operations. The SeqNum is also used to detect duplicate messages and node resets. The SeqNum must be implemented as a 1-octet lollipop counter wrapping from `0xff` to `0x01`, and increments by exactly 1 at each request. A value of `0x00` is only possible after a node has reset or after a CLEAR command.

Because nodes in the network store the last successful SeqNum seen from a neighbor schedule, resets can be detected. This happens when a node receives a SeqNum of 0 in a 6P request while it stores a SeqNum for that neighbor with a different value. This situation needs to be handled by the

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

receiver node by sending an RC_SEQNUM response code (refer to Table IV for a description of the return codes). Any node receiving such a response may start a procedure to recover or reset the schedule as described above.

C. Scheduling Functions

6P goes hand-in-hand with a Scheduling Function (SF), the policy that decides how to maintain cells and trigger 6P transactions. The Minimal Scheduling Function (MSF) [11] is the default SF defined by the 6TiSCH WG; other standardized SFs can be defined in the future. MSF extends the minimal schedule configuration, and is used to add child-parent links according to the traffic load. The policy defines two types of cells referred as *Autonomous Cells* and *Negotiated Cells*. Autonomous Cells provide connectivity to any neighbor without requiring any signaling. Negotiated cells in its turn, are handled by the 6P protocol and installed or removed from the schedule following a reactive traffic-based policy.

A node running MSF maintains two types of autonomous cells: *autonomous RX cell* and *autonomous TX cell*. The *autonomous RX cell* is a cell configured with a Link Options field where the receive (RX) bits is set. It is identified by a pair (slotoffset, channeloffset) computed as a hash of the node's *own EUI64*.

The *autonomous TX cell* is a cell configured with a Link Options field where the transmit (TX) bits are set. They are identified by a pair (slotoffset, channeloffset) computed as a hash of the layer 2 destination EUI64 address of a frame to be transmitted.

*autonomous RX cell*is permanently installed in schedule while *autonomous TX cell*is installed on demand. When there is a unicast frame to be transmit and there is no negotiated TX cell in schedule, the *autonomous TX cell*is installed. The cell is removed right away when the frame is sent out.

MSF uses the SAX [46] hash function and computes the coordinates (slotOffset,channelOffset) from an EUI64 address. The method ensures a homogeneous distribution of the cells along timeslot-offsets and channel-offsets. The first timeslot-offset is avoided to not collide with the minimal cell (as per RFC8180). Eq 3 details how the coordinates are computed given an EUI64 address.

$$slotOffset = 1 + hash(EUI64, (len(Slotframe) - 1))$$
$$channelOffset = hash(EUI64, 16) \quad (3)$$

Because of hash collisions, there are cases where the autonomous TX and RX cells are scheduled at the same time offset and/or channel offset. Hash collisions among a set of cells at a given time offset is resolved at run-time following a set of rules: *1) The autonomous TX cell with the most packets in the outgoing queue takes precedence. 2) If all autonomous TX cells have empty outgoing queues, the autonomous RX celltakes precedence.* Other rules imposed by MSF ensure that the autonomous cells are maintained during the network lifetime. The *autonomous TX cell*and *autonomous RX cell*must be kept in the schedule, even after a 6P RESET or CLEAR transaction. *autonomous TX cell*is only removed when there is no uni-cast frame to transmit or there is negotiated TX cell for frames to transmit.

MSF relies on the 6TiSCH Minimal Security specification, detailed in Section IV, for a node to securely join a 6TiSCH network. With the keying material obtained from the join process, the node is able to decrypt the RPL DIOs and to acquire its Rank and select its preferred routing parent. *autonomous RX cell* is installed when mote boots up. During the join process and 6P transaction, *autonomous TX cell* is installed according to the layer 2 destination of join request/response and 6P request/response on both the pledge and JP sides. The explanation of the join procedure including the flows between Join request and response is detailed in Section IV.

After having acquired a RPL rank, the node starts synchronizing only to its preferred parent, by sending TSCH keep-alive frames (recall that while waiting for the join procedure to finish it synchronizes through listening EBs from all neighbors). Then the node generates a 6P ADD request to ask installing the first dedicated negotiated TX Cell to its parent. In case the 6P transaction failed, the node repeats to send 6P ADD request until one negotiated TX cell installed to its parent. After that, the node starts sending EBs and RPL DIOs, allowing new pledge nodes to join through it. The joined node at that point relies on MSF and 6P to dynamically support application traffic.

MSF is responsible to add/delete a Negotiated cell based on the rate at which the node exchanges application packets with its preferred parent. MSF keeps track of the cell usage of the Negotiated cell, the portion of cells used with its preferred parent: it increments **numCellElapsed** at each elapsed cell, and increments **numCellUsed** each time that cell was used to send or receive a frame from that neighbor. Every 16 slotframes, MSF computes the cell usage as the ratio between **numCellUsed** and **numCellElapsed**. If the cell usage is greater than 75%, MSF issues an 6P ADD request to add one (Negotiated) dedicated cell to that neighbor. If the cell usage is smaller than 25%, MSF issues a 6P DELETE request to delete one (Negotiated) dedicated cell to that neighbor. By default, at least one Negotiated cell to a parent is kept in the schedule.

Fig. 12 illustrates how MSF works, assuming that the packet delivery ratio (PDR) of each cell is 100%. At the beginning (leftmost column) the node needs to send 2 packets per slotframe to its preferred parent, but has only a single cell to that neighbor in its schedule. After 16 slotframes, the cell usage is 16/16=100%. MSF issues a 6P ADD request to add one TX cell to its preferred parent. After the cell is installed, and after another 16 slotframes, the cell usage is still 32/32=100%. Then MSF calls 6P again to add another TX cell. With the three cells scheduled to that neighbor, the cell usage drops to 32/48=67%, a stable region for MSF. Let's assume that, at that point, the node's traffic requirement drops to 1 frame every 3 slotframes. The cell usage hence drops to 6/48=12.5%. MSF issues a 6P DELETE request to delete one (Negotiated) TX cell to its preferred parent. After this action, cell usage increases to 6/32=18.75%, still below the 25% limit, triggering 6P to delete another cell. This causes the cell usage to increase to 6/16=37.5%, a stable region for MSF.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

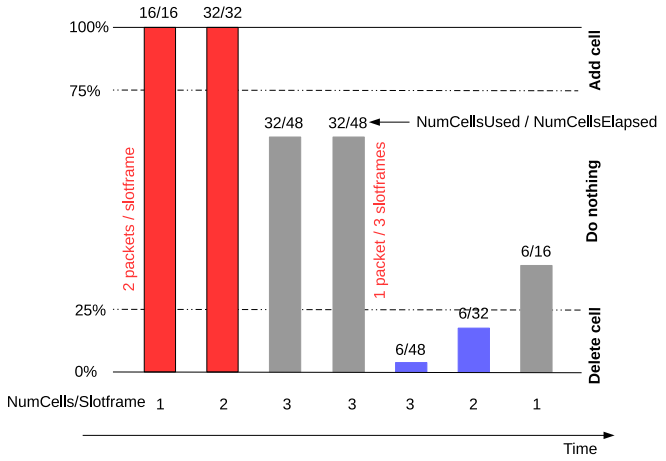

Fig. 12. MSF dynamically adapts the number of cells to a neighbor to the cell usage, i.e. the ratio of cells scheduled with that neighbor that are actually used.

When replacing the TX cell above by RX cell, MSF is capable to adapt to downstream traffic. At each RX cell including *autonomous RX cell*, **numCellElapsed** increments. If a valid frame with correct CRC is received on the RX cell, **numCellUsed** increments. Then the same strategy applies according to the cell usage to add/delete RX cells for downstream traffic adaptation.

In case a node switches its preferred parent, it schedules to the new parent the same number of cells it had to the previous parent, and then deletes all the cells to the previous parent.

In the Negotiated cells allocated by MSF, schedule collision may happen as well since cells are randomly chosen by neighbors, nearby pairs of neighbors may (accidentally) schedule the same cell(s). MSF detects schedule collisions and relocates the colliding Managed cells to other locations in the schedule. Every minute, MSF computes the Packet Delivery Ratio (PDR) of all the cells to its preferred parent, and identifies the cell with the highest PDR. It then compares that PDR to the PDR of all other cells to the same preferred parent. If the difference is larger than 50%, MSF considers that cell as colliding, and issues a 6P RELOCATE request. Muroaka *et al.* demonstrate that convergence happens even in very dense schedules [47].

When an error code is returned by the 6P protocol, MSF defines the behavior to handle it. Tab. IV presents the action that is taken upon the reception of a particular error code.

- nothing: Indicates that this Return Code is not an error. No error handling behavior is hence triggered.
- clear: Abort the 6P Transaction. Issue a 6P CLEAR command to that neighbor (this command may fail). Remove all cells scheduled with that neighbor from the local schedule. Keep that node in the neighbor and routing tables.
- quarantine: Same behavior as for "clear". In addition, remove the node from the neighbor and routing tables. Place the node's identifier in a quarantine list for 5 minutes. When in quarantine, drop all frames received from that node.
- wait-retry: Abort the 6P Transaction. Wait for a duration randomly and uniformly chosen in [30...60] s. Retry the same transaction.

TABLE IV
BEHAVIOR FOR EACH 6P ERROR CODE.

Code	Description	Recommended behavior
RC_SUCCESS	Operation succeeded	nothing
RC_EOL	Reached end of list when Listing Cells	nothing
RC_ERR	General error. Wrong format or invalid parameters	quarantine
RC_RESET	Critical error. Reset transaction	quarantine
RC_ERR_VERSION	Unsupported 6P version	quarantine
RC_ERR_SFID	Unsupported SFID	quarantine
RC_ERR_SEQNUM	Schedule inconsistency	clear
RC_ERR_CELLLIST	CellList error in a transaction	clear
RC_ERR_BUSY	Busy in a concurrent transaction	wait-retry
RC_ERR_LOCKED	Requested cells are locked by another transaction	wait-retry

IV. 6TiSCH Secure Join Process

Secure join [6] is the process by which a pledge gets admitted into the network[2]. The network is represented by a central entity called Join Registrar/Coordinator (JRC). A third entity in the process is called Join Proxy (JP), a radio neighbor of the pledge and an application-level relay, whose role is to facilitate communication between the pledge and the JRC, that may potentially run in the cloud. The JP is selected based on the Join Metric contained in the received EBs, as detailed in Section III-A.

The pledge and the JRC mutually authenticate, based on the knowledge of a secret key, called pre-shared key (PSK). The PSK must necessarily be pre-provisioned to the pledge. For each pledge authorized to join the network, the JRC stores its PSK and a unique identifier. The authentication based on certificates is supported as part of the ongoing zero-touch standardization work [48], but we do not detail it in this tutorial as it is still an early stage discussion.

The JRC manages dynamic link-layer keys used in the network, as well as the assignment of unique short link-layer addresses to the pledges. The key(s) and the assigned short address are provided to the pledge by the JRC upon successful authentication and authorization.

A. Link Layer Considerations

The join process requires at least one active cell in the pledge's schedule. This cell is provided by the RFC8180, as described in Section III-C. In a production network, and according to RFC8180, link-layer security must be enabled, EBs are authenticated using key K1, data frames are authenticated and their payload is encrypted using key K2. The two keys can have the same value.

[2] The pledge can also be admitted into the network out-of-band, which is not standardized in 6TiSCH, but the distributed parameters must necessarily be the same as in the in-band case. This is, however, not recommended as the parameters can change during the network lifetime, in which case the nodes need to execute the same mechanism [6] and therefore have the implementation available. To join a secure 6TiSCH network, the pledge must either go through the in-band or the out-of-band process.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

Because K1 and K2 are unknown to the pledge at join time, this necessitates a bypass mechanism called *secExempt* to be configured in IEEE802.15.4 security tables at both the pledge and the JP. The necessary link-layer security behavior for the join process to complete is:

- the pledge accepts EBs from potentially multiple nodes which it can not authenticate but needs to process in order to select a JP;
- the pledge accepts unprotected frames from JP; the pledge sends unprotected frames to JP;
- The JP accepts unprotected frames from the pledge; the JP sends unprotected frames to the pledge.

The *secExempt* flag provided by the IEEE802.15.4 MAC layer enables a node to locally override accepted security levels for particular neighbors, disabling the validation of the outcome of the security processing, and hence accepting insecure frames. It is important to stress that the configuration of the *secExempt* mechanism should be implemented with utmost care, as it may have severe security consequences for the network. While the *secExempt* configuration policy is implementation-specific, the IETF 6TiSCH working group is discussing a mechanism to propagate the "activation" signal through the network. The main idea is to use Information Elements within the EBs with a flag that indicates to JPs whether they can accept to forward traffic from pledges or not. Alternatively, this can be implemented locally at each node by, for example, pressing a button on the device or through a custom application-layer command.

In summary, frames exchanged between the pledge and the JP are not protected at the link-layer, but the end-to-end messages from the pledge to the JRC and vice versa, carried within those frames, are secured at the application level using keys derived from the PSK, as detailed in the following paragraphs.

B. Networking Layer Considerations

Moving on to the IP layer, we now detail the structure of IPv6 packets exchanged during join. Because the pledge at join time does not have the IPv6 prefix of the network, nor the JRC's IPv6 address, it cannot exchange packets directly with the JRC, which is potentially multiple IP hops away. Instead, the pledge communicates with the JP at the application layer, which then forwards the join messages towards the JRC[3]. Once the pledge selects a JP, it constructs the JP's link-local IPv6 address based on the link-layer source address of the received EB, as defined in Section 7 of RFC4944, and start sending IPv6 packets to it. When such packets (from the pledge to the JP and vice versa) undergo IP-level compression, they end up carrying stateless RFC6282 compression flags.

The JP processes the messages from the pledge at the application layer without having to understand all of its content, as they are partly encrypted with a key that JP does not posses.

[3] The JP has learned the IPv6 address of the JRC once has joined the network, either through *(1)* DODAGID field in RPL DIO messages, in case the JRC is co-located with the DODAG root or *(2)* explicitly in the join message received from JRC when JP acted as a pledge, in case JRC is in the cloud.

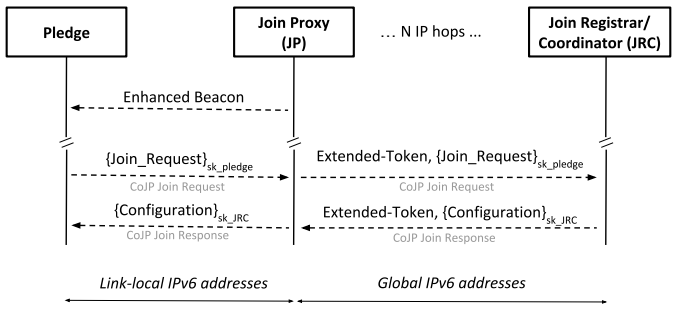

Fig. 13. 6TiSCH join process with pre-shared keys consists of network discovery through Enhanced Beacons and the execution of the Constrained Join Protocol. $\{...\}_K$ denotes authenticated encryption under key K, as defined by OSCORE. Join Request message carries a protected Join_Request data object. Join Response message carries the Configuration data object with the configuration parameters for the pledge. The response is cryptographically matched to the request, and both are protected against replay attacks. Both sk_pledge and sk_JRC keys are derived from the PSK: "sk" stands for sender key.

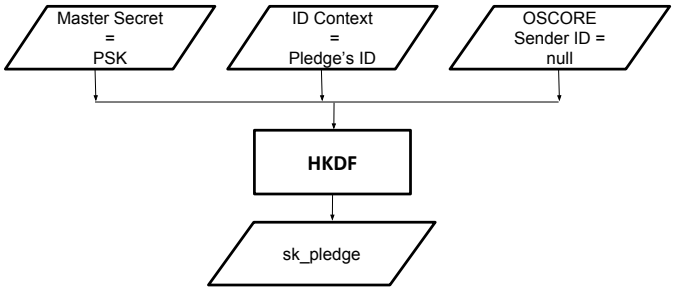

Fig. 14. Simplified OSCORE key derivation as used by Constrained Join Protocol. HKDF is a key derivation function defined in RFC5869 [49]. Key sk_JRC is derived by using OSCORE Sender ID equal to "JRC", all other inputs being equal.

Then, the JP forwards the message to the JRC in a new IPv6 packet, using global IPv6 addressing and stateful RFC6282 compression, as it knows the network prefix and JRC's IPv6 address.

C. Application Layer Considerations

A successful join exchange consists of two application-level messages: Join Request and Join Response. Messages are proxied in both directions by the JP, as depicted in Fig. 13. The pledge constructs the Join Request message as a CoAP [21] POST request, protected end-to-end by OSCORE [22]. Upon reception and correct verification of the Join Request, the JRC puts the link-layer key(s) and the assigned short address in the Join Response payload. Join Request and Join Response are encrypted under two different keys, both derived from the PSK that both the pledge and the JRC know. Fig. 14 details the inputs to the OSCORE key derivation process.

The actual OSCORE encryption algorithm, CCM* [1], is the same as the one used in the link-layer, making it feasible to reuse common libraries and hardware acceleration typically provided by IEEE802.15.4-compliant chips. The pledge and the JRC are implemented as CoAP applications: at join time, pledge acts as a client and once it joins it becomes a server;

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

JRC acts as a server during join and updates the parameters in the network on different nodes as a client. The JP acts as a generic CoAP proxy, thus no specific application-layer code is needed, but CoAP libraries need to be complemented with the processing of the CoAP Extended Token [50]. The pledge initiates the Join Request, adds the CoAP options defined in 6TiSCH [6], and uses OSCORE [22] to protect the request. The OSCORE security context used to protect the request is instantiated as defined in 6TiSCH [6]. OSCORE encrypts the request, leaving in the clear the options the proxy needs to process.

Before forwarding the request to the JRC, the JP wraps the state it needs to keep locally into the (extended) CoAP token, that is then echoed by the JRC in the Join Response. This allows the JP to remain stateless and avoid preserving per-pledge state in order to forward the response to a given pledge. The content and format of this extended token is relevant only to the JP and is therefore not standardized. Typically, it contains the link-layer unique identifier of the pledge, from which a link-local IPv6 address can be derived, an UDP port number if different than the default CoAP port, and potentially a local interface identifier in case the JP has more than one network interface. Before forwarding the response to the pledge, the JP removes this extended token.

V. 6LoWPAN and 6TiSCH integration

6LoWPAN refers to a set of specifications that enable transporting IPv6 datagrams on top of different constrained link layers. 6LoWPAN was initially specified to support IPv6 on IEEE802.15.4 [12], [13] but lately other constrained technologies such as Bluetooth Low-Energy, BACNET and DECT-ULE have been supported [51]–[53], and more are under development (NFC [54] and Power Line Communications [55]).

The 6LoWPAN framework is composed of different specifications that provide Header Compression (RFC4944 [12] and RFC6282 [13]), Neighbor Discovery (RFC6775 [56] and RFC8505 [16]), and Fragmentation (RFC4944), which is being updated at the time of this writing . It also contains features to improve header compression such as Paging Dispatch (RFC8025) [14] which enables the compression of RPL data-path artifacts (RFC8138) [15], and ESC Dispatch (RFC8066) [57] which enables additional extensions by protocols defined outside the IETF.

This section aims to clarify from an implementation point of view the major aspects of putting together such standards, with a special focus on 6TiSCH. We outline the latest compression extensions and the not so clear details when network boundaries are crossed. We refer Shelby and Bormann's book on 6LoWPAN [27] for extensive details on the protocol suite.

A. Stateless Header Compression

The original 6LoWPAN Header Compression [12] enables a degree of IPv6 and UDP header compression, but, being fully stateless, it cannot compress global IPv6 addresses efficiently. This function was superseded by RFC6282 [13], a partially stateful compression that relies on shared context to allow compression of arbitrary prefixes.

Shelby and Bormann [27] extend the details presented by RFC4944, providing an exhaustive description of Stateless IPHC based on Header Compression 1 (HC1 – IP header compression) and Header Compression 2 (HC2 – UDP header compression) schemes. These compression schemes are very efficient when compressing link-local addresses, as they enable to fully or partially suppress the source and destination addresses, as prefixes in the same sub-network are identical and host addresses can be mapped to the EUI64 of the nodes. Nevertheless, stateless header compression does not perform as well for routable addresses, i.e. those that cross network boundaries or routers. Prefixes in those cases cannot be statelessly elided. RFC6282 therefore proposes a scheme to partially compress addresses based on contextual information pre-configured, for example during neighbor discovery.

In detail, a Context ID is used to identify parts of the source and destination addresses, for example the prefixes. An context ID field is added to the IPHC header.

6TiSCH relies on both header compression schemes. On the one hand, stateless address compression is used during the secure join procedure, when a pledge interacts with the JP. Communication between pledge and JP is performed using fully-compressed link local addresses, as the pledge does not have a global IP address yet. After the join procedure, when the pledge is part of the network, it uses global IP addressing to interact with the Internet, while still using stateless address compression and link local addresses for internal network signaling such as in control messages (e.g. RPL messages).

B. Crossing Network Boundaries

When a node is the source of an original packet, and the destination is known to be within the same RPL domain, the node should include the RPL Option directly within the original packet. Otherwise, nodes must use IPv6-in-IPv6 tunneling [58] and place the RPL Option in the tunnel header, as illustrated in Fig. 15. Using IPv6-in-IPv6 tunneling ensures that the delivered datagram remains unmodified and that ICMPv6 errors generated by a RPL Option are sent back to the node that generated the RPL Option.

This requirement was identified as problematic by the 6TiSCH WG since it adds extra overhead to the header. The requirement is more problematic when combined with the use of a source routing header [20], as in the case of the non-storing mode of RPL. To cope with that problem, 6lo, ROLL and 6TiSCH worked towards the compression of the RPL artifacts as described in the following section.

C. Extension Headers

6LoWPAN defines a one-octet Dispatch field at the very beginning of the frame that indicates what the content of the next bytes in the frame is made of. Multiple dispatch bytes can be used one after the other, for instance first fragmentation and then IP Header compression. But the initial documents (RFC4944, RFC6282) have mostly consumed the possible patterns and the possibility to extend the protocol became limited. In order to circumvent this, the IETF has produced two new mechanisms:

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

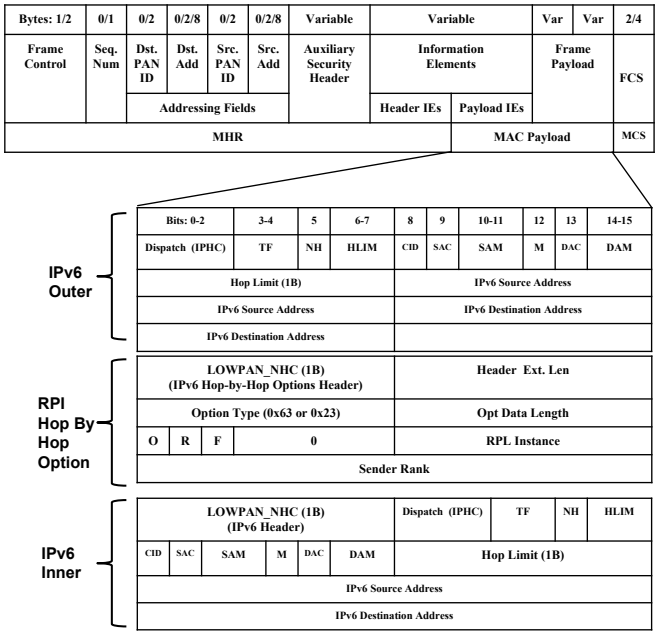

Fig. 15. Hop By Hop option header in the IPv6 outer header. IP-in-IP encapsulation as the packet is crossing a router. No 6LoRH nor Paging dispatch compression.

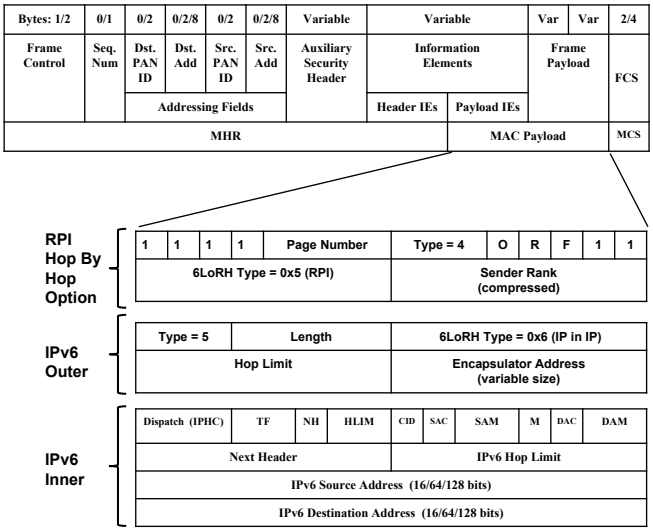

Fig. 16. Hop By Hop option header in the IPv6 outer header. IP-in-IP encapsulation as the packet is crossing a router.

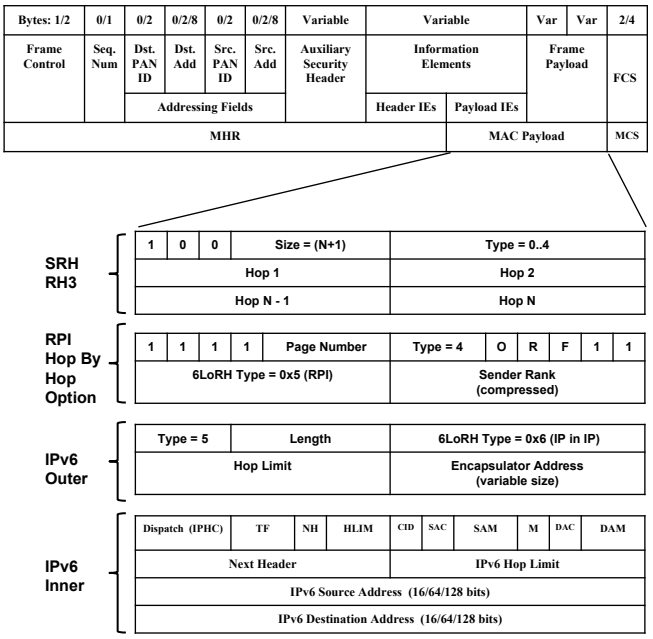

Fig. 17. SRH (RH3) for source routing. Hop By Hop option header in the IPv6 outer header. IP-in-IP encapsulation as the packet is crossing a router.

- the Paging Dispatch (RFC8025) [14] is a collection of 16 dispatch values each introducing a new dispatch space. The page dispatch appears as a switch, so that the next dispatch values after that switch are all interpreted within the page indicated by the switch, until another page switch appears as the frame is sequentially parsed. We refer to Fig. 16 and Fig. 17 as a reference of its use. The dispatch value `1111` followed by the page number is an indication of this IPHC extension. In the figures, we can see that it follows by a compressed RPL artifact as described below in this section.
- the ESC Dispatch (RFC8066) [57] increases the width of the Dispatch field, so new values can be allocated. The ESC Dispatch has been used so far only for non-IETF specifications, and hence it does not apply to 6TiSCH although it is described here for completeness. The drawback of this dispatch is that each escaped Dispatch value must be preceded by an ESC dispatch, each consuming 2 octets instead of one.

The first specification that leverages the Page Dispatch is "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header" (6LoRH) is RFC8138 [15]. RFC8138 compresses the RPL data-path artifacts that are added to enable routing within the RPL domain. Those artifacts are the RPL Packet Information (RPI) [19] and the RPL Source Routing Header (SRH) [20]. RFC8138 also enables an improved compression of the IP-in-IP encapsulation that is the necessary when manipulating IPv6 headers.

As presented in Fig. 16, the RPI header is compressed from its original form per RFC6553, consuming 2 bytes for the dispatch field, plus 6 bytes for the artifact fields. The RPI-6LoRH header consumes up to 4 bytes per RFC8138. The compression is achieved by reducing the space for the option identifier, removing the option length as it can be inferred, enabling the elimination of the RPL Instance ID if this is 0, and compressing the RPL Rank as defined by RFC6550 using Eq. (4).

$$DAGRank(rank) = floor(rank/MinHopRankIncrease) \quad (4)$$

The RPL source routing header is defined in RFC6554 to route packets downstream in a non-storing mode. The header appends in the packet a strict list of addresses a node should traverse to reach the destination. The RPL SRH consumes 8

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

TABLE V
ADDRESS COMPRESSION IN N.

6LoRH Type	Length of compressed IPv6 address (bytes)
0	1
1	2
2	4
3	8
4	16

bytes as header fields, plus a variable number of bytes that depends on the number of hops traversed. This header was limiting the number of hops that can be traversed in 6TiSCH networks, and hence required a compression mechanism. The SRH-6LoRH is the compression of the SRH proposed by RFC8138. The option is placed first in the compressed packet to simplify the hop-by-hop operation. With RFC8138, addresses that are consumed are removed from the header and not recoverable at arrival.

Fig. 17 shows an example use of the SRH-6LoRH header followed by the RPI. This is a compression of the source routing header defined in RFC6554 for the non-storing mode of RPL. The type field defines the length of the addresses carried in the header (see Table V). Addresses are compressed according to the common prefix part.

A final artifact that can be compressed by the 6LoRH specification is the IPv6-in-IPv6 header.

Fig. 15 presents a non-compressed IP-in-IP header while Fig. 16 presents its compressed form with 6LoRH. The order of the extension headers in the 6LoRH header is changed so the parsing can be done efficiently. The outer IPv6 header is compressed following the 6LoRH specification, by suppressing some fields that can be inferred from the inner IPv6 header. Further details and clarifications of the use of RPL artifacts with IPv6 headers are described in work in progress document edited by the ROLL WG [28]. The document as well introduces a new Option Type value 0x23 that enables to omit IP in IP encapsulation for upstream traffic crossing routers.

D. Fragment Forwarding

When an IPv6 packet is too long to be carried in a single IEEE802.15.4 frame, RFC4944 defines a mechanism to fragment it. In a naive implementation, this results in fragmentation and reassembly at every hop. There are at least 2 limits to doing per-hop fragmentation and reassembly. First, when reassembling, a node needs to wait for all the fragments to be received before being able to generate the IPv6 packet, and possibly forward it to the next hop. This may result in increased end-to-end latency compared to the case where each fragment would be forwarded without per-hop reassembly. Second, constrained nodes have limited memory, which can typically hold only 1-3 reassembly buffers. When a node is concurrently receiving fragmented packets, it may run out of reassembly buffers, and it will have no option but to drop one of the packets, lowering the end-to-end reliability [29].

An elegant solution is to implement 6LoWPAN fragmentation using a Virtual Reassembly Buffer [30]. That is, each node is still reassembling and fragmenting, only it forwards a fragment as soon as possible, without ever reassembling the entire packet. This results in fragment forwarding, and overcomes the limitations listed above. It is particularly elegant in that it is an implementation technique, rather than a new standard, but is sensitive to lossy links since a lost fragment will block the recomposition buffer for a long time.

In contrast, a new ongoing work at the IETF on reliable fragments [31] will enable reliable fragment transmissions with flow control and explicit congestion notification, but will require to upgrade all the devices in the network.

VI. ROUTING IN 6TISCH: RPL

RPL was designed by the IETF Roll WG. It is specified by RFC6550 [17] and a number of companion documents. The primary goal of RPL is to enable multi-hop routing. To address the low-power constraint, RPL forms a network anchored at a root, typically a border router to external networks. Communication between any pair of nodes is possible by first routing first upwards (towards the root), then downward (towards the destination). To address loss, RPL includes a number of mechanisms for agile route selection, loop detection, route repair, etc.

RPL is designed to cover a wide range of applications. Nodes can run multiple instances of RPL in the same physical network. Each instance specifies its own set of configuration parameters, such as the mode of operation, objective function and beaconing period, etc. Nodes within an instance can run multiple DODAGs, each anchored at a different root, adding fault tolerance and flexibility.

RPL offers two modes of operation: storing and non-storing. In the storing mode, nodes maintain routing tables locally in a distributed fashion. Downwards routing is done from any node to another node below in the DODAG, by following the routing tables from parent to child. We focus on non-storing mode, as this is the mode mandated by 6TiSCH. In the non-storing mode, nodes do not maintain any routing state locally. Instead, the DODAG root maintains a complete topology view, and performs source routing. All traffic is routed to the root first, where source routes are computed and appended to the datagram (as the IPv6 routing extension header). Then, the packet is sent downwards and forwarded as per the source routing header.

The RPL traffic control is anisotropic, meaning that it operates differently upwards and downwards. It leverages four dedicated ICMPv6 codes: DODAG Information Solicitation (DIS), DODAG Information Object (DIO), Destination Advertisement Object (DAO), and Destination Advertisement Object Acknowledgment (DAO-ACK). DIS and DIO messages are used to form the network topology as depicted in Fig. 18 and enable default routing upwards, while DAO and DAO-ACK populate the resulting DODAG with more specific downwards routes. More specifically: DIS messages are used for nodes to request routing information from a (set of) neighbor(s); DIO messages are used for nodes to build the DODAG starting at the root, and advertise default and possibly a more specific routing information to a (set of) neighbor(s); DAO messages

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

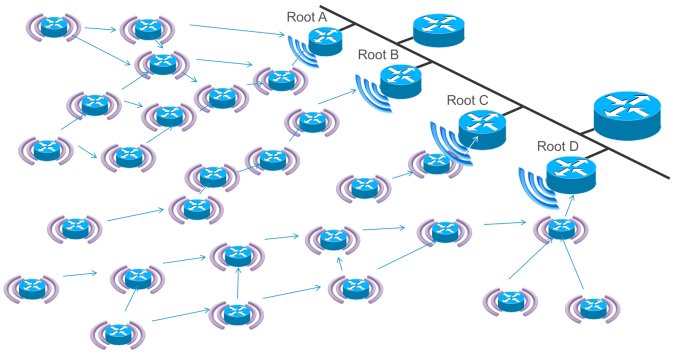

Fig. 18. RPL Upstream topology formed by downstream signaling messages firstly originated at the DAGRoot and subsequently for the nodes forming the network.

are used by nodes to obtain reachability back along the DODAG. Finally, DAO-ACK acknowledge DAO messages for reliable route (de)registration.

A node configured as root will advertise its instance and DODAG via multicast DIOs, and it will wait for nodes to join. Multicast DIOs are sent at a dynamic period, using a Trickle [59] timer. A node willing to join a network typically sends multicast DIS to solicit DIOs from neighbors (instead of passively waiting for the next multicast DIO). Upon receiving a DIO, a node may choose to join the instance and DODAG. In this case, it proceeds with parent selection, as dictated by the instance's Objective Function, discussed next. It will then advertise its parent by sending a DAO directly to the root (non-storing mode), which will then store the new child-parent relationship (described as target and transit information in the DAO). The root will then have the option to acknowledge the DAO with a DAO-ACK. When receiving the DAO-ACK, the newly joined node knows it is now reachable. It can then start advertising the DODAG with multicast DIOs, which include the Rank, a representation of the logical distance to the root. Other nodes hearing the DIO will update their neighbor table, select a parent accordingly, thus participating in multi-hop topology formation. During the network operation, the nodes keep monitoring link quality with their neighbors, and update their parent to keep the topology efficient.

RPL is designed as a generic Distance Vector (DV) protocol with a balance between route optimization and cost of control that operates as follows. On the one hand, routes from (Point to Multipoint, P2MP) and to (Multipoint to point, MP2P) the root of the RPL DODAG are optimized, which requires a certain amount of control messages. On the other hand, routes between devices in the network are stretched via a common parent. This allows to maximize the use of default routes, and minimize the cost of control message and routing state information for the device to device communication, which is often a secondary concern in LLN applications. Additionally, the rate of control messages can be kept very low, with a side effect that broken routes may not be repaired immediately. This is alleviated by a data-plane routing error detection based on a RPL Packet Information (RPI) that RPL places in data packets and that will enable to detect and remove stale routes and loops if and when they are effectively used. This has been discussed in Section V. Broken routes that are not effectively used may exist for a while without affecting the operation of the network, and there is no energy wasted in repairing them.

While RPL defines the generic DV operation in a fashion that optimized P2MP and MP2P traffic with the root, the desired result of the optimization depends on the problem that is being solved. Examples include application requirements such as latency and delivery ratio, metrics that the protocol can observe and make decision upon such as ETX and long term Link Quality indicators, and degree of redundancy that needs to be applied (multipath, FEC/coding). The understanding of those metrics really depends on the use case and is beyond RFC6550.

In order to adapt to the very diverse use cases it serves, RPL provides a concept that is somewhat analog to a plug-in in a web browser, whereby an objective function (OF) extends the generic DV operation and adapts it to the problem at hand. The underlying assumption is that other Standard Defining Organizations (SDOs) such as the IEEE, or even individual vendors, may provide their own optimizations for the networks they build, while benefiting from the loop avoidance and the generic DV operation that RPL offers. The selection of an OF determines the strategy for parent selection, redundancy, and the end-to-end properties such as reliability and timeliness, and plays a crucial role in how the multi-hop topology is built and maintained.

As examples of what an objective function can be, the IETF originally defined two of them: OF0 (RFC6552 [18]), a very simple OF that can serve use cases such as wired networks where the hop count is usable, or it can be adapted for more complex metrics that it will normalize, and MRHOF (RFC6719 [60]), which employ hysteresis for stable parent selection. Both objective functions can operate on a different set of metrics and constraints (defined in RFC6551 [44]). Metrics allow to optimize the topology for a particular performance criteria (e.g. trade reliability for latency or energy). Constraints allow to force routing through nodes with particular properties, e.g. to design an instance where routing takes place only over mains-powered nodes. RFC8180 in 6TiSCH uses OF0 with ETX as a metric [61].

RPL provides different security mechanisms; by default relying in the underlying security mechanism provided by the IEEE802.15.4 MAC and CCM* but also enabling the encryption of signaling messages using pre-established keys [17]. Raoof et.al. [62] provide a detailed security mechanism analysis and threat mitigation techniques for the RPL protocol.

VII. Implementation

This section details the (hardware) requirements for implementing 6TiSCH (Section VII-A), and lists existing open-source and commercial implementations (Section VII-B).

A. Requirements and Gotchas

Hardware dependencies of the 6TiSCH stack are inherited from the TSCH MAC layer. In this section, we discuss hardware choices that facilitate a well-performing software implementation of TSCH, on top of IEEE802.15.4-compliant

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

hardware. We discuss how hardware affects two high-level performance metrics: *(1)* the capacity of a 6TiSCH network, and *(2)* the minimum achievable duty cycle per node, that translates into device lifetime. Capacity directly depends to the duration of a TSCH timeslot that is by design common for all the nodes in the network (advertised in EBs). Idle intervals within a slot (no communication on the channel) account for imperfectly synchronized devices and processing delays. By minimizing these idle intervals and shortening the slot, we maximize the capacity. Both synchronization guard intervals and processing delays depend on hardware. The minimum achievable duty cycle is dominated by the need to keep synchronization by exchanging frames with the time-source neighbor.

Critical Processing Delays. To be compliant with 6TiSCH, a device ideally supports the default 10 ms slots. The time critical operation within a slot is the interval between a reception of a frame and the transmission of an acknowledgment. Since TSCH acknowledgments are critical for keeping synchronization, they are encrypted. The acknowledgment needs to be sent exactly TsTxAckDelay (see Fig. 4) after the last symbol of the frame has been received. Within TsTxAckDelay (1 ms in a 10 ms slot), the receiver needs to be able to:

1) retrieve the frame from the radio;
2) parse and decrypt the received frame;
3) construct and encrypt the acknowledgment;
4) load the acknowledgment in the radio.

Operations 1 and 4 are heavily influenced by the board-level architecture. Operations 2 and 3 depend on *(a)* the system clock's speed, which typically needs to be as low as possible to keep the power consumption low, and *(b)* the performance of hardware-accelerated encryption.

Board-level Architecture. The most common hardware implementation of IEEE802.15.4 is in the form of an independent radio chip (transceiver). In this case, the radio chip is controlled by the micro-controller unit (MCU) over a chip-to-chip communication interface, such as the Serial Peripheral Interface Bus (SPI). With this architecture, the overhead of chip-to-chip communication dominates the delays for loading and retrieving a frame from the radio. Some manufacturers provide System-on-Chip (SoC) implementations of IEEE802.15.4, where the micro-controller unit (MCU) and the radio transceiver share a common memory. In these cases, loading (resp. retrieving) a frame consists in simply writing (resp. reading) to a memory location, and is significantly faster. The SoC architecture is therefore preferred, but both have been demonstrated to be able to run 10 ms slots.

Hardware-accelerated Encryption. TSCH uses Advanced Encryption Standard (AES) in the CCM* mode to protect frames by authenticated encryption. CCM* encrypts and authenticates the frames by chaining counter (CTR) and cipher block chaining (CBC) modes. It uses AES in encrypt mode, both for CCM* encryption and decryption, making it optional to support AES decryption functionality.

Hardware-accelerated AES is provided by virtually all IEEE802.15.4-compliant chips, with varying levels of functionality which greatly affects the performance. Some chips only provide standalone AES encryption of a single 16-byte block, making it necessary for the MCU to load/retrieve 16 bytes at a time. The delay of a single 16-byte AES encryption is typically negligible and can be implemented in hardware in as low as 10 cycles. The problem is typically the delay to access the acceleration block and load/retrieve the data. When MCU communicates with such an acceleration block over the chip-to-chip communication interface such as SPI, the resulting delay to decrypt and authenticate a 127-byte frame can be well over a millisecond, making it infeasible to run a 10 ms slot. Other chips implement AES with different chaining modes, and some include full CCM* operation, making it feasible to load and encrypt the full 127-byte frame in one go. Some implementations within the radio chip allow the frame to be decrypted/encrypted while in the radio buffer, without having to be explicitly handled by MCU. These can be optimized to provide good performance, but are not well supported in open-source implementations of 6TiSCH, as the logical flow and interfaces are highly dependent on specific hardware.

The preferred case is the acceleration block within the MCU. If such a block supports full CCM* operation, the delay to encrypt/decrypt and authenticate a 127-byte frame at 32 MHz, can be below 60 μs. If only intermediary CTR and CBC modes are supported, and CCM*-specific vectors need to be constructed in software, the delay at 32 MHz can go up to 300 μs, but still making it possible to complete all the operations within 1 ms. Sciancalepore *et al.* [63] study in detail the achievable slot duration for different boards, depending on the encryption performance of hardware-acceleration blocks.

Synchronization. Because nodes in a network have independent clock sources, they inherently beat at slightly different frequencies. The differences come both from the imperfect manufacturing process and external factors such as temperature, supply voltage and aging, and cause clock drift. A node in a TSCH network keeps synchronized by exchanging frames and aligning its clock based on the feedback from its time-source neighbor (Section II-C). How often the node needs to synchronize to its time-source neighbor depends on the relative drift between them and the fixed guard interval used in the slot (2.2 ms with 10 ms slots).

Clock Sources. A typical TSCH board comes equipped with two external crystals:

1) A high-speed crystal – on the order of 20 MHz – attached to the radio, and used for IEEE802.15.4 PHY-compliant modulation. This crystal is powered only when the radio needs to communicate. Drift is typically below 40 ppm.
2) A low-speed crystal – typically 32 kHz – attached to the MCU. This crystal is constantly powered. Drift is typically below 30 ppm.

The critical design consideration is to have an MCU timer running clocked from the low-speed crystal, while the rest of the MCU (and the board) is powered off. At minimum, this timer needs one compare register in order to wake up the system by generating an interrupt at the correct instant. The power consumption of the board in sleep mode is then dominated by *(a)* consumption of the low-speed crystal, and *(b)* sleep mode consumption of the MCU. Sleep mode consumption is

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

a critical metric for any low-power system, as nodes typically spend more than 99% of their time sleeping.

Local Timekeeping. The precision of the clock source is only one factor that affects the synchronization accuracy. A TSCH implementation also needs to precisely timestamp different events and to schedule hardware actions based on the value of time. A very minimum is to know the instant at which the radio received the Start of Frame Delimiter (SFD). The timestamp can be read for example within an interrupt service routine, where the software polls hardware for the current time. Some hardware implementations also facilitate asynchronous event-based timestamping, where hardware automatically fills a given register with the time value, for example upon SFD arrival. The latter is preferred as it provides better timestamp accuracy which propagates into the synchronization accuracy of two nodes in the network. For outgoing frames, the hardware should at minimum be able to control the exact time at which the SFD leaves the radio. This could be achieved, for example, through an SPI command or by lifting a pin, in the case of an independent radio chip, or by writing to a specific memory location (register) in the case of a SoC.

B. Existing Implementations

The 6TiSCH stack is fully implemented in at least 4 open source projects: OpenWSN [26], Contiki(-NG) [34], RIOT [64] and TinyOS [65]. The implementations are mostly independent [23] and have demonstrated interoperability at different IETF plugtest events [66]. Table VII provides a list of most relevant evaluation tools developed by the 6TiSCH community.

The OpenWSN project is an open-source implementation of the 6TiSCH standards protocol stack for IEEE802.15.4 TSCH networks. The project originated at UC Berkeley, and has now an extended community of contributors around the world. OpenWSN implements the IEEE802.15.4 Time Synchronized Channel Hopping MAC layer, and couples it to the 6LoWPAN, RPL and CoAP through the 6TiSCH management layer. OpenWSN is an early adopter of the work conducted at the 6TiSCH working group, providing early implementations of the most recent standards.

OpenWSN is implemented in C and is ported to different hardware platforms including the popular TelosB, OpenMote [67] and the IoT-LAB [68] motes. It is also supported for a wide range of low power microcontrollers from major vendors. OpenWSN has been selected by ETSI as the reference implementation ("golden device") during interoperability events.

Contiki [69] and its fork Contiki-NG [70] is an open-source operating system for constrained devices, started in 2003, with contributors from both academia and industry. It features a fully certified IPv6 stack as well as many of the IETF low-power IPv6 protocols. Since 2015, it also supports 6TiSCH [71], with an implementation tested for interoperability in two IETF plugtests.

Contiki-NG [70], a fork of Contiki, was released in 2017 with a focus on dependable low-power IPv6 for modern IoT platforms. It brings a general cleanup, an updated 6TiSCH with 6top and 6P support, a streamlined RPL implementation, and many other features for low-power Internet devices. This implementation was successfully tested for interoperability in the July 2017 IETF plugtest, on three different hardware platforms (cc2538, cc2650, JN516x).

Analog Devices develops a line of products referred as a SmartMesh IP [81] that implement a pre-6TiSCH protocol stack. Technical details can be found in the literature [82].

Apart from O.S. for embedded devices providing the 6TiSCH stack we can find several commercial devices that are supported by those implementations. Table VI provides a lists of them together with the 6TiSCH stack they support.

TABLE VI
LIST OF AVAILABLE MOTES SUPPORTING 6TISCH IMPLEMENTATIONS

Mote	6TiSCH Stack	Reference
Analog Devices LTP5902	Proprietary	[72]
I3Mote	Contiki NG Proprietary	[73]
IoTeam Dusty	Proprietary	[74]
IoT Lab M3	Contiki NG OpenWSN RiOT	[68]
Jenic JN516x	Contiki NG	[75]
Nordic NRF51822	OpenWSN Contiki NG	[76]
OpenMote-CC2538	OpenWSN RiOT	[67]
OpenMote-B	Contiki NG OpenWSN RiOT	[77]
TelosB	Contiki NG OpenWSN RiOT	[78]
TI CC2650/1350	Contiki NG	[79]
Zolertia Re-Mote	Contiki NG	[80]

C. Multimedia documentation

6TiSCH is an effort that involved several open source projects and academic initiatives. These communities have generated documentation and multimedia material that may serve readers to extend their knowledge on the technology. In this section, we provide a summary of these sources. One of the major enablers of 6TiSCH is the OpenWSN project at UC Berkeley. The project generated a introductory material [25], hands on tutorials, seminar talks [91], [92], and complementary software tools such as packet sniffers, Wireshark dissectors available at the project github [83]. The OpenWSN team also developed a simulator for 6TiSCH enabling the simulation of large scale networks [84]. The Contiki community in its turn developed several tutorials to guide adopters. General Contiki tutorials can be found in the Contiki and Contiki-NG main web sites [93], [94]. Contiki-NG has a particular tutorial for 6TiSCH [95]. Contiki also features a simulator known as Cooja [85], in it 6TiSCH networks can be emulated.

VIII. FUTURE DIRECTIONS OF 6TISCH

The 6TiSCH standardization process is reaching maturity as fundamental specifications to securely bootstrap the network,

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

TABLE VII
SUMMARY OF AVAILABLE BENCHMARKING AND EVALUATION TOOLS FOR 6TISCH.

Tool	Type	Description	Reference
OpenWSN	Implementation	Full stack implementation	[26], [25]
Contiki-NG	Implementation	Full stack implementation	[70], [71]
RiOT	Implementation	Full stack implementation	[64]
OpenWSN simulator	Simulator	A simulator based on the OpenWSN implementation	[26], [83]
6TiSCH Simulator	Simulator	A python-based standalone simulator	[84]
Cooja	Simulator	The Contiki simulator	[85]
Argus	Accessory	Cloud based wireless sniffer tools	[86]
Wireshark Dissector	Accessory	6TiSCH Dissector for Wireshark	[87]
F-interop	Testing Platform	6TiSCH online conformance testing platform	[88]
OpenBenchmark	Testing/Benchmarking Platform	6TiSCH KPI and benchmarking tools and datasets	[89], [90]

manage the schedule and orchestrate the communication resources according to the traffic demands are materialized in standards. There are different activities towards evolving the technology at different layers and components of the protocol stack.

A. Physical layer

Relevant efforts are carried out by the IEEE802.15.4 working group to amend and improve the IEEE802.15.4 standard. For example, thanks to the 2015 revision, OFDM modulation was included as part of the integration of the IEEE802.15.4g amendment. This modulation provides a good trade-off considering spectrum-efficiency, reliability and energy consumption [96]. Future directions indicate that 6TiSCH will go beyond the current 2.4Ghz IEEE802.15.4 physical layer and will be used on subGhz bands [97]. Different trade-offs exists such as overcoming the duty-cycle limitation of those bands and supporting slower data rates. Thanks to the flexibility of IEEE802.15.4 TSCH and the management support of 6TiSCH, those seem to be achievable objectives [98]. As one of the prior works in this direction, the draft introducing the problem statement [99] describes an overall view on the aspects needs to be considered at each layer of 6TiSCH stack to apply on multiple PHYs network.

B. Security

A relevant battlefield in the 6TiSCH activities is the provisioning of security to the network. This activity has been naturally split in two important objectives. As indicated in this tutorial and as a first objective the development of a one touch approach provisioning scheme. It relies on a shared secret in order to derive the security credentials of joining nodes. The way this shared secret is provisioned is not addressed in the current standardization specifications (minimal security and CoJP [6]). As a second and more ambitious objectives, the zero-touch approach, aims to enable a secure mechanism to ensure that joining nodes can acquire secure keys in a safe manner, without relying on any out-of-band mechanism [100]. The work proposing the use of Ephemeral Diffie-Hellman Over COSE (EDHOC) at the CORE working group seems to drive this possibility [101], but still there are numerous barriers to address, including a differentiation to other approaches such as the heavy DTLS.

C. MAC layer and management

There are still ongoing activities towards permitting nodes to enable and disable join procedures and prioritize join proxy selection [102]. Future research directions may investigate the management and coexistence of dual-band/dual-radio technologies and how the management plane should address such capabilities. Few efforts have been put in maximizing network capacity in TSCH, for example by exploiting higher data rate physical layers and shorter time slots. This approach seems needed in the context of industry digitization [103] and other use cases such as geo-technical applications [104].

D. Network scheduling

Other future directions are the exploration of novel scheduling functions to ensure end-to-end paths with network-wide objective functions. Architectures such as those defined by RSVP and MPLS need to be explored, extending pioneering art [105] proving the suitability of the approach. The IETF is now taking the next step of enabling Deterministic (DetNet) flows over radios that can be scheduled. Such radios certainly include IEEE Std. 802.15.4 TSCH, but also IEEE Std. 802.11ac with upcoming Extreme High Throughput (EHT) and 3GPP Ultra Reliable and Low Latency Communications (URLLC) [106]. The new Reliable and Available Wireless WG to be (RAW) will enable the use of hard cells in a 6TiSCH network, and provide the full realization of the 6TiSCH architecture, a mix of best effort IPv6 and deterministic flows over a shared radio network. RAW will define mechanisms that can guarantee quality of service along a 6TiSCH Track, including redundancy mechanism at the DetNet service layer such as Packet ARQ, Replication, Elimination and Ordering (PAREO) functions [107].

E. IP and routing

Side effects due to the work at 6TiSCH have triggered a revisit to the fragmentation approaches in 6lo, virtual buffers for example are now described as a tool to avoid assembly and disassembly of entire packets at every hop [29]. The RPL routing protocol is undergoing major additions that 6TiSCH would inherit. To cite the least, a model based on BIER [108] whereby routing is done based on a bitmap where each destination is indicated as a particular bit may enable storing mode in constrained devices. The route projection [109] work enables a mix of storing mode and non-storing mode under

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

the centralized control of a PCE, providing shorter routing headers and traversal routes. Finally, the updated 6LoWPAN ND [16] allows RPL-unaware leaves [110] to participate to the RPL network as mobile end points.

IX. Conclusion

This tutorial provides deep technical insights into the IETF 6TiSCH standard suite. The suite is composed of several IETF standards, which build on top of the IEEE802.15.4 TSCH MAC and PHY layer. The tutorial provided a detailed description of the involved specifications and standards, giving details on how they are put together and what are the fundamental configurations and steps that are required to bootstrap a 6TiSCH network, secure it and make sure it performs well. This tutorial collects the efforts carried out within different standardization groups at the IETF and IEEE, and derives from the different specifications those details that make implementations possible.

Acknowledgements

This research is partially funded by the SPOTS project (RTI2018-095438-A-I00) funded by the Spanish Ministry of Science, Innovation and Universities.

References

[1] IEEE, *802.15.4-2015: 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., October 2015.

[2] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1389–1406, 12 December 2013.

[3] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: Deterministic IP-enabled Industrial Internet (of Things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, December 2014.

[4] P. Thubert, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-architecture-20 [work-in-progress], Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-architecture-20

[5] X. Vilajosana, K. Pister, and T. Watteyne, *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*, Internet Engineering Task Force Std. RFC8180, May 2017.

[6] M. Vučinić, J. Simon, K. Pister, and M. Richardson, *Minimal Security Framework for 6TiSCH*, Internet Engineering Task Force Std. draft-ietf-6tisch-minimal-security-09 [work-in-progress], November 2018.

[7] Q. Wang, X. Vilajosana, and T. Watteyne, *6top Protocol (6P)*, Internet Engineering Task Force Std. RFC8480, August 2018.

[8] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Conference on Embedded Networked Sensor Systems (SenSys)*. Seoul, South Korea: ACM, 2015, pp. 337–350.

[9] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed PID-based Scheduling for 6TiSCH Networks," *IEEE Communications Letters*, March 2016.

[10] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low Latency Scheduling Function for 6TiSCH Networks," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, May 2016, pp. 93–95.

[11] T. Chang, M. Vucinic, S. Duquennoy, D. Dujovne, and X. Vilajosana, *6TiSCH Minimal Scheduling Function (MSF)*, Internet Engineering Task Force Std. draft-ietf-6tisch-msf-06, August 2019.

[12] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, Internet Engineering Task Force Std. RFC4944, September 2007.

[13] J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, Internet Engineering Task Force Std. RFC6282, September 2011.

[14] P. Thubert and R. Cragie, *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch*, Internet Engineering Task Force Std. RFC8025, November 2016.

[15] P. Thubert, C. Bormann, L. Toutain, and R. Cragie, *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header*, Internet Engineering Task Force Std. RFC8138, April 2017.

[16] P. Thubert, E. Nordmark, S. Chakrabarti, and C. E. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery," RFC 8505, Nov. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8505.txt

[17] 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*, Internet Engineering Task Force Std. RFC6550, March 2012.

[18] P. Thubert, *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*, Internet Engineering Task Force Std. RFC6552, March 2012.

[19] J. Hui and J. Vasseur, *The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams*, Internet Engineering Task Force Std. RFC6553, March 2012.

[20] J. Hui, J. Vasseur, D. Culler, and V. Manral, *An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)*, Internet Engineering Task Force Std. RFC6554, March 2012.

[21] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, Internet Engineering Task Force Std. RFC7252, June 2014.

[22] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)," RFC 8613, Jul. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8613.txt

[23] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial wireless ip-based cyber-physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1025–1038, May 2016.

[24] D. D. Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey-," *Computer Communications*, vol. 88, pp. 1–24, 2016.

[25] "OpenWSN.org. Web Site." 2018. [Online]. Available: {http://openwsn.org/}

[26] 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*, vol. 23, no. 5, pp. 480–493, 2012.

[27] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010.

[28] I. Robles, M. Richardson, and P. Thubert, "Using RPL Option Type, Routing Header for Source Routes and IPv6-in- IPv6 encapsulation in the RPL Data Plane," Internet Engineering Task Force, Internet-Draft draft-ietf-roll-useofrplinfo-24, Jan. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-roll-useofrplinfo-24

[29] T. Watteyne, C. Bormann, and P. Thubert, "LLN Minimal Fragment Forwarding," Internet Engineering Task Force, Internet-Draft draft-ietf-6lo-minimal-fragment [work-in-progress], Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6lo-minimal-fragment-01

[30] C. Bormann and T. Watteyne, "Virtual reassembly buffers in 6LoWPAN," Internet Engineering Task Force, Internet-Draft draft-ietf-lwig-6lowpan-virtual-reassembly [work-in-progress], Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-lwig-6lowpan-virtual-reassembly-01

[31] P. Thubert, "6LoWPAN Selective Fragment Recovery," Internet Engineering Task Force, Internet-Draft draft-ietf-6lo-fragment-recovery [work-in-progress], Jan. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6lo-fragment-recovery-02

[32] "CoAP Main WebSite," 2018. [Online]. Available: {http://coap.technology}

[33] "CoAP Available implementations," 2018. [Online]. Available: {http://coap.technology/impls.html}

[34] S. Duquennoy, A. Elsts, B. Al Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Ottawa, Canada: IEEE, June 2017.

[35] T. Watteyne, P. Tuset-Peiro, X. Vilajosana, S. Pollin, and B. Krishnamachari, "Teaching communication technologies and standards for the

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

industrial iot? use 6tisch!" *IEEE Communications Magazine*, vol. 55, no. 5, pp. 132–137, May 2017.

[36] "WI-SUN Field Area Networks," 2018. [Online]. Available: {https://www.wi-sun.org/}

[37] T. Watteyne, A. Mehta, and K. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," in *Proceedings of the 6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. ACM, 2009, pp. 116–123.

[38] T. Watteyne, C. Adjih, and X. Vilajosana, "Lessons learned from large-scale dense IEEE802.15.4 connectivity traces," in *CASE*. IEEE, 2015, pp. 145–150.

[39] N. Sastry and D. Wagner, "Security considerations for ieee 802.15.4 networks," in *Proceedings of the 3rd ACM Workshop on Wireless Security*, ser. WiSe '04. New York, NY, USA: ACM, 2004, pp. 32–42. [Online]. Available: http://doi.acm.org/10.1145/1023646.1023654

[40] X. Vilajosana, P. Tuset-Peiro, F. Vazquez-Gallego, J. Alonso-Zarate, and L. Alonso, "Standardized Low-Power Wireless Communication Technologies for Distributed Sensing Applications," *Sensors*, vol. 14, no. 2, pp. 2663–2682, 2014.

[41] D. Stanislowski, X. Vilajosana, Q. Wang, T. Watteyne, and K. Pister, "Adaptive Synchronization in IEEE802.15.4e Networks," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 795–802, February 2014.

[42] B. Martinez, X. Vilajosana, and D. Dujovne, "Accurate Clock Discipline For Long-Term Synchronization Intervals," *IEEE Sensors Journal*, vol. 17, no. 7, pp. 2249–2258, April 2017.

[43] T. Chang, T. Watteyne, K. Pister, and Q. Wang, "Adaptive Synchronization in Multi-Hop TSCH Networks," *Computer Networks*, vol. 76, no. C, pp. 165–176, January 2015.

[44] D. Barthel, J. Vasseur, K. Pister, M. Kim, and N. Dejean, *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*, Internet Engineering Task Force Std. RFC6551, March 2012.

[45] T. Kivinen and P. Kinney, *IEEE 802.15.4 Information Element for the IETF*, Internet Engineering Task Force Std. RFC8137, May 2017.

[46] M. Ramakrishna and J. Zobel, "Performance in Practice of String Hashing Functions," in *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*. IEEE, April 1997, pp. 01–04.

[47] K. Muraoka, T. Watteyne, N. Accettura, X. Vilajosana, and K. Pister, "Simple Distributed Scheduling With Collision Detection in TSCH Networks," *IEEE Sensors Journal*, vol. 16, no. 15, pp. 5848–5849, August 2016.

[48] M. Richardson and B. Damm, *6tisch Zero-Touch Secure Join protocol*, Internet Engineering Task Force Std. draft-ietf-6tisch-dtsecurity-zerotouch-join-01 [work-in-progress], October 2017.

[49] D. H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, May 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5869.txt

[50] K. Hartke, *Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)*, Internet Engineering Task Force Std. draft-ietf-core-stateless-01 [work-in-progress], March 2019.

[51] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez, *IPv6 over BLUETOOTH(R) Low Energy*, Internet Engineering Task Force Std. RFC7668, October 2015.

[52] P. B. Mariager, J. T. Petersen, Z. Shelby, M. van de Logt, and D. Barthel, *Transmission of IPv6 Packets over Digital Enhanced Cordless Telecommunications (DECT) Ultra Low Energy (ULE)*, Internet Engineering Task Force Std. RFC8105, May 2017.

[53] K. Lynn, J. Martocci, C. Neilson, and S. Donaldson, *Transmission of IPv6 over Master-Slave/Token-Passing (MS/TP) Networks*, Internet Engineering Task Force Std. RFC8163, May 2017.

[54] Y. Choi, Y.-G. Hong, J.-S. Youn, D.-K. Kim, and J.-H. Choi, *Transmission of IPv6 Packets over Near Field Communication*, Internet Engineering Task Force Std. draft-ietf-6lo-nfc-09 [work-in-progress], January 2018.

[55] J. Hou, Y.-G. Hong, and X. Tang, *Transmission of IPv6 Packets over PLC Networks*, Internet Engineering Task Force Std. draft-hou-6lo-plc-03 [work-in-progress], December 2017.

[56] C. Bormann, Z. Shelby, S. Chakrabarti, and E. Nordmark, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, Internet Engineering Task Force Std. RFC 6775, November 2012.

[57] S. Chakrabarti, G. Montenegro, R. Droms, and J. Woodyatt, *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) ESC Dispatch Code Points and Guidelines*, Internet Engineering Task Force Std. RFC8066, February 2017.

[58] S. E. Deering and A. Conta, *Generic Packet Tunneling in IPv6 Specification*, Internet Engineering Task Force Std. RFC2473, December 1998.

[59] P. Levis, T. H. Clausen, O. Gnawali, J. Hui, and J. Ko, *The Trickle Algorithm*, Internet Engineering Task Force Std. RFC6206, March 2011.

[60] O. Gnawali and P. Levis, *The Minimum Rank with Hysteresis Objective Function*, Internet Engineering Task Force Std. RFC6719, September 2012.

[61] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 134–146. [Online]. Available: http://doi.acm.org/10.1145/938985.939000

[62] A. Raoof, A. Matrawy, and C. Lung, "Routing attacks and mitigation methods for rpl-based internet of things," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[63] S. Sciancalepore, M. Vučinić, G. Piro, G. Boggia, and T. Watteyne, "Link-layer security in tsch networks: effect on slot duration," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 1, 2017.

[64] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "Riot os: Towards an os for the internet of things," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2013, pp. 79–80.

[65] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, *TinyOS: An Operating System for Sensor Networks*. Springer, 2005, pp. 115–148.

[66] M. R. Palattella, X. Vilajosana, T. Chang, M. A. Reina Ortega, and T. Watteyne, "Lessons Learned from the 6TiSCH Plugtests," in *Conference on Interoperability in IoT (InterIoT), part of the iOT360 Summit*. Rome, Italy: EAI, October 2015.

[67] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: Open-Source Prototyping Platform for the Industrial IoT," in *International Conference on Ad Hoc Networks (AdHocNets)*. San Remo, Italy: EAI, September 2015.

[68] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "Fit iot-lab: A large scale open experimental iot testbed," in *2015 IEEE 2nd World Forum on Internet of Things (W-iF-IoT)*, Dec 2015, pp. 459–464.

[69] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[70] "Contiki-NG: The OS for Next Generation IoT Devices," 2018. [Online]. Available: {http://contiki-ng.org/}

[71] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomo, "Tsch and 6tisch for contiki: Challenges, design and evaluation," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2017, pp. 11–18.

[72] Analog Devices, "Smartmesh ip wireless IEEE802.15.4e pcba module with antenna connector," 2018. [Online]. Available: {http://www.analog.com/en/products/ltp5902-ipm.html}

[73] B. Martinez, X. Vilajosana, I. H. Kim, J. Zhou, P. Tuset-Peir, A. Xhafa, D. Poissonnier, and X. Lu, "I3mote: An open development platform for the intelligent industrial internet," *Sensors*, vol. 17, no. 5, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/5/986

[74] "IoTTeam Dusty Breakout," 2018. [Online]. Available: {http://ioteam.strikingly.com/}

[75] NXP, "Jenic jn516x mote," 2018. [Online]. Available: {https://www.nxp.com/products/wireless-connectivity/proprietary-ieee-802.15.4-based/support-resources-for-jn516x-mcus:SUPPORT-RESOURCES-JN516X-MCUS}

[76] Nordic, "Nordic nrf52832 advanced performance bluetooth5/ant/2.4ghz proprietary soc," 2018. [Online]. Available: {https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832}

[77] OpenMote Technologies, "Openmote-b dual radio open source mote," 2018. [Online]. Available: {http://www.openmote.com/}

[78] "TelosB data sheet," http://www.sentilla.com/files/pdf/eol/tmote-skydatasheet. pdf, Sentilla, 2010.

[79] Texas Instruments, "CC2650 SimpleLink multi-standard 2.4 GHz ultra-low power wireless MCU," 2018. [Online]. Available: {http://www.ti.com/product/CC2650}

[80] Zolertia, "Re-mote," 2018. [Online]. Available: {https://zolertia.io/}

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

[81] "Analog Devices SmartMesh IP," 2018. [Online]. Available: {https://www.analog.com/en/products/rf-microwave/wireless-sensor-networks/smartmesh-ip.html}

[82] T. Watteyne, L. Doherty, J. Simon, and K. Pister, "Technical overview of smartmesh ip," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2013, pp. 547–551.

[83] "OpenWSN.org. Software Repository at GitHub," 2018. [Online]. Available: {https://github.com/openwsn-berkeley}

[84] E. Municio, G. Daneels, M. Vuini, S. Latr, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6tisch networks," *Transactions on Emerging Telecommunications Technologies*, vol. 0, no. 0, p. e3494, 2018, e3494 ett.3494. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3494

[85] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "Cooja/mspsim: Interoperability testing for wireless sensor networks," in *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 27:1–27:7. [Online]. Available: https://doi.org/10.4108/ICST.SIMUTOOLS2009.5637

[86] "Argus, Cloud interface for a Wireless Sniffer," 2018. [Online]. Available: {https://github.com/openwsn-berkeley/argus}

[87] "Wireshark 6TiSCH Dissector," 2018. [Online]. Available: {https://github.com/openwsn-berkeley/dissectors}

[88] M. R. Palattella, F. Sismondi, T. Chang, L. Baron, M. Vučinić, P. Modernell, X. Vilajosana, and T. Watteyne, "F-interop platform and tools: Validating iot implementations faster," in *Ad-hoc, Mobile, and Wireless Networks*, N. Montavont and G. Z. Papadopoulos, Eds. Cham: Springer International Publishing, 2018, pp. 332–343.

[89] M. Vučinić, B. Škrbić, E. Kočan, M. Pejanović-Djurišić, and T. Watteyne, "OpenBenchmark: Reproducible and Repeatable Internet of Things Experimentation on Testbeds," in *INFOCOM CNERT*, 2019.

[90] M. Vučinić, M. Pejanović-Djurišić, and T. Watteyne, "SODA: 6TiSCH Open Data Action," in *Workshop on Benchmarking Cyber-Physical Networks and Systems, Porto, Portugal*, 2018, pp. 42–46. [Online]. Available: https://doi.org/10.1109/CPSBench.2018.00014

[91] "OpenWSN.org. Tutorials section," 2018. [Online]. Available: {https://openwsn.atlassian.net/wiki/spaces/OW/pages/688195/Tutorials}

[92] "OpenWSN.org. Video Recordings and Talks," 2018. [Online]. Available: {https://openwsn.atlassian.net/wiki/spaces/OW/pages/28835862/Overview+Talks}

[93] "Contiki-OS getting started guide," 2014. [Online]. Available: {http://www.contiki-os.org/start.html}

[94] "Contiki-NG Wiki," 2018. [Online]. Available: {https://github.com/contiki-ng/contiki-ng/wiki}

[95] "Contiki-NG 6TiSCH tutorial," 2018. [Online]. Available: {https://github.com/contiki-ng/contiki-ng/wiki/Tutorial:-TSCH-and-6TiSCH}

[96] P. Tuset-Peiro, F. Vázquez-Gallego, J. M. noz, T. Watteyne, J. Alonso-Zarate, and X. Vilajosana, "experimental interference robustness evaluation of ieee 802.15.4-2015 oqpsk-dsss and sun-ofdm physical layers."

[97] J. Munoz, T. Chang, X. Vilajosana, and T. Watteyne, "Evaluation of ieee802.15.4g for environmental observations," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/10/3468

[98] J. Munoz, P. Muhlethaler, X. Vilajosana, and T. Watteyne, "Why Channel Hopping Makes Sense, even with IEEE802.15.4 OFDM at 2.4 GHz," in *Global IoT Summit (GIoTS)*, Bilbao, Spain, Jun. 2018. [Online]. Available: https://hal.inria.fr/hal-01756523

[99] J. Munoz, X. Vilajosana, and T. Chang, *Problem Statement for Generalizing 6TiSCH to Multiple PHYs*, Internet Engineering Task Force Std. draft-munoz-6tisch-multi-phy-nodes-00 [work-in-progress], July 2018.

[100] M. Richardson and B. Damm, "6tisch Zero-Touch Secure Join protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-dtsecurity-zerotouch-join-02, Apr. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-dtsecurity-zerotouch-join-02

[101] G. Selander, J. Mattsson, and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)," Internet Engineering Task Force, Internet-Draft draft-selander-ace-cose-ecdhe-13, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-selander-ace-cose-ecdhe-13

[102] D. Dujovne and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join and Enrollment Information," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-enrollment-enhanced-beacon-00, Jul. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-enrollment-enhanced-beacon-00

[103] B. Martinez, C. Cano, and X. Vilajosana, "A square peg in a round hole: The complex path for wireless in the manufacturing industry," *IEEE Communications Magazine*, vol. 57, no. 4, pp. 109–115, April 2019.

[104] X. Vilajosana, B. Martinez, I. Vilajosana, and T. Watteyne, "On the suitability of 6tisch for wireless seismic data streaming," *Internet Technology Letters*, vol. 1, no. 2, p. e20, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.20

[105] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label switching over ieee802. 15.4 e networks," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 458–475, 2013.

[106] P. Thubert, *Reliable and Available Wireless Technologies*, Internet Engineering Task Force Std. draft-thubert-raw-technologies-00 [work-in-progress], May 2019.

[107] J. De Armas, P. Tuset, T. Chang, F. Adelantado, T. Watteyne, and X. Vilajosana, "Determinism through path diversity: Why packet replication makes sense," in *Intelligent Networking and Collaborative Systems (INCoS), 2016 International Conference on*. IEEE, 2016, pp. 150–154.

[108] P. Thubert, "RPL-BIER," Internet Engineering Task Force, Internet-Draft draft-thubert-roll-bier-02, Jul. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-thubert-roll-bier-02

[109] P. Thubert, R. A. JADHAV, and J. Pylakutty, "Root initiated routing state in RPL," Internet Engineering Task Force, Internet-Draft draft-ietf-roll-dao-projection-04, Jun. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-roll-dao-projection-04

[110] P. Thubert, "Routing for RPL Leaves," Internet Engineering Task Force, Internet-Draft draft-thubert-roll-unaware-leaves-05, May 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-thubert-roll-unaware-leaves-05

Xavier Vilajosana He is currently Professor and Principal Investigator at the Wireless Networks Research Lab at the Universitat Oberta de Catalunya. He is also co-founder of Worldsensing. From January 2012 to January 2014, Xavier was visiting Professor at the University of California Berkeley holding a prestigious Fulbright fellowship. In 2008, he was visiting researcher of France Telecom R&D Labs, Paris. Xavier has been one of the main promoters of low power wireless technologies, co-leading the OpenWSN.org initiative at UC Berkeley, and promoting the use of low power wireless standards for the emerging Industrial Internet paradigm. He also contributed to the industrialization and introduction of Low Power Wide Area Networks to urban scenarios through Worldsensing. Xavier is author of different Internet Drafts, as part of his standardization activities for low power industrial networks. Xavier is contributing actively at the IETF 6TiSCH WG. He holds an MsC degree on Computer Science from the Universitat Politècnica de Catalunya (UPC) and a PhD on Computer Science from the Universitat Oberta de Catalunya. At the moment, Xavier is author of 30 patents, more than 50 high impact journal publications and more than 50 International conference contributions. Technically, Xavier has extensive experience in Distributed Systems, Wireless Networks and Industrial Networks.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2939407, IEEE Communications Surveys & Tutorials

Thomas Watteyne is an insatiable enthusiast of low-power wireless mesh technologies. He holds an advanced research position at Inria in Paris, in the EVA research team, where he designs, models and builds networking solutions based on a variety of Internet-of-Things (IoT) standards. He is a Senior Networking Design Engineer at Analog Devices, in the Dust Networks product group. Since 2013, he co-chairs the IETF 6TiSCH working group, which standardizes how to use IEEE802.15.4e TSCH in IPv6-enabled mesh networks, and is member of the IETF Internet-of-Things Directorate. Prior to that, Thomas was a postdoctoral research lead in Prof. Kristofer Pister's team at the University of California, Berkeley. He founded and co-leads Berkeley's OpenWSN project, an open-source initiative to promote the use of fully standards-based protocol stacks for the IoT. Between 2005 and 2008, he was a research engineer at France Telecom, Orange Labs. He holds a PhD in Computer Science (2008), an MSc in Networking (2005) and an MEng in Telecommunications (2005) from INSA Lyon, France. He is Senior member of IEEE.

Tengfei Chang is a postdoctoral Research Engineer at Inria-EVA, Paris, and is the project lead of the OpenWSN project, an open-source project founded by UC Berkeley. In 2014, he was visiting scholar at the University of California, Berkeley. He is one of the main implementors of the IETF 6TiSCH protocol stack. He has huge interest in wireless sensor and actuator Network, swarm robotics and embedded system design. He holds is a Computer Science and Technology PhD from University of Science and Technology, Beijing (2017).

Mališa Vučinić holds a starting Research Scientist position at Inria in Paris in the EVA research team. He obtained his PhD degree from Grenoble Alps University, Grenoble, in 2015 and did his postdocs at Inria-EVA in Paris and University of Montenegro in Podgorica. His research interests are at the intersection of (IoT) security and performance analysis, theory and practice. He is active in the IETF and co-authors multiple drafts in different working groups, including 6TiSCH. Main concepts from his PhD dissertation have found way towards the Internet standards tackling object security and its use in the IoT. He has extensive practical experience with IoT networks, low-power devices and RESTful protocols. He is a core developer of the OpenWSN project, implementing the 6TiSCH stack, and of the 6TiSCH discrete-event simulator. He has significant international experience spanning the United States (UC Berkeley), France (University Grenoble Alps, Inria), Italy (Politecnico di Torino) and Montenegro (University of Montenegro).

Simon Duquennoy is a senior researcher at the Swedish Institute of Computer Science (SICS Swedish ICT). He obtained his PhD from Université de Lille 1 (Inria, CNRS, France) in 2010, and did his postdoc at SICS as the holder of an ERCIM Alain Bensoussan fellowship. His research interests are at the intersection between operating systems and networking for constrained embedded devices, with a focus on IP-based sensor networks and the Internet of Things. He is a core developer and maintainer of the Contiki Operating System. He has been TPC member of internationally recognized conferences such as ICDCS, MASS, DCOSS, LCN and EWSN, and publishes regularly in the sensor networks community flagship conferences ACM/IEEE IPSN and ACM SenSys.

Pascal Thubert has been actively involved in research, development and standards efforts on Internet mobility and wireless technologies since joining Cisco in Y2K. He currently works at Cisco's Chief Technology and Architecture office, where he focuses on products and standards in the general context of the Internet of Things. Pascal specializes in IPv6 as applied to mobility and wireless devices and developed microcode for routers and switches with Cisco's core IPv6 product development group. In parallel with his R&D missions, he has authored multiple IETF RFCs and draft standards dealing with IPv6, mobility and the Internet of Things, a number of papers, and more than a hundred patents in related areas. In particular, Pascal participated to the ISA100.11a specification as well as the NEMO, 6LoWPAN, RPL and now 6lo and DetNet IETF standards, and co-chairs the 6TiSCH and the LPWAN WGs, all in an effort to bring together the broad features of the IT/OT convergence for an Industrial Internet.

1553-877X (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.