6TiSCH Wireless Industrial Networks: Determinism Meets IPv6

Maria Rita Palattella[1], Pascal Thubert[2], Xavier Vilajosana[3,4], Thomas Watteyne[4,5], Qin Wang[4,6], and Thomas Engel[1]

[1] University of Luxembourg, Luxembourg.
{maria-rita.palattella,thomas.engel}@uni.lu
[2] Cisco Systems, France.
pthubert@cisco.com
[3] Universitat Oberta de Catalunya, Catalonia, Spain.
xvilajosana@uoc.edu
[4] University of California at Berkeley, CA, USA.
{xvilajosana,watteyne}@eecs.berkeley.edu
[5] Linear Technology/Dust Networks Product Group, CA, USA.
twatteyne@linear.com
[6] University Science and Technology Beijing, China.
wangqin@ies.ustb.edu.cn

1 Introduction

Operational Technology (OT) refers to industrial networks that are typically used for monitoring systems and supporting control loops, as well as movement detection systems for use in Process Control (i.e., process manufacturing) and Factory Automation (i.e., discrete manufacturing).

For the last 40 years or more, industrial networks were developed in parallel to the Public Switched Telephone Network (PSTN), and to early data networks such as Bitnet, which was based on IBM's Systems Network Architecture (SNA). For the last 25 years, the Internet Protocol (IP) [1] has become the *de-facto* standard for the networking layer of the Information Technology (IT) and for the Internet at large.

For the last 15 years, Voice over IP (VoIP) and the emergence of Internet protocols such as the Media Gateway Control Protocol (MGCP) [2] and the Session Initiation Protocol (SIP) [3] enabled a progressive convergence of voice and data networking technologies, over IP. The Real-Time Transport Protocol (RTP) and the Real-Time Transport Control Protocol (RTCP) [4] enabled video streaming and conferencing, leading to the convergence of voice, data, and video over the IP infrastructure.

Information and Communications Technology (ICT) extends the term IT to refer to that convergence, including the software and hardware pieces that

enable the manipulation, access, transport and persistence of information over an IP network, be it for voice, data, or video applications.

The next convergence step is the integration of IT and OT technologies, enabling OT traffic to be transported over a shared IT, IP-based, infrastructure. This integration presents a number of new challenges. Due to its different goals, OT has evolved in a manner that is radically different from IT, focusing on highly secure, reliable and deterministic networks, with limited scalability over a bounded area.

Traffic flows such as control loops and motion detection systems are deterministic in that their communication patterns are known a priori. Routing paths and communication schedules can be computed in advance, in a fashion similar to a railway system, to avoid losses due to packet collisions, and to perform global optimizations across multiple flows. Based on that knowledge, a *deterministic network* allocates the required resources (buffers, processors, medium access) along the multi-hop routing path at the precise moment the resources are needed. The forwarding elements can handle data with an amount of jitter that can be negligible for a particular application.

This model is different from the IP Quality of Service (QoS) model, which relies of selective queuing and discarding of packets to achieve end-to-end flow control on statistically multiplexed traffic flows. With strict ingress shaping and resource over-provisioning, QoS can reduce the congestion loss and induced jitter, but never eliminate them completely.

OT is moving gradually towards Ethernet and IP in order to reuse Commercial Off-The-Shelf (COTS) products and reduce cost. In that case, IP technology is only used as yet another fieldbus, with still a clear physical separation between the IT and OT networks and no IP routing between the two domains. End-to-end communication between applications and field devices over IP are i) mostly a green field ii) a long term investment for a new factory and iii) expected to scale to large numbers of devices, possibly tens of thousands in a large plant. It results that IP version 6 (IPv6) [5] is the de-facto IP standard version for the IT/OT convergence. A legacy IPv4 domain can still be reached using Network Address Translation between IPv6 an IPv4 (NAT64) [6] techniques.

To achieve a real convergence, OT needs not only to adopt the formats of IPv6 packets, but also to share the use of the Internet Protocol suite over the common fabric. The required protocol suite for OT includes at a minimum the Internet Control Message Protocol (ICMPv6) [7], the IPv6 Neighbor Discovery Protocol (NDP) [8], the Routing Protocol for Low-Power and Lossy Networks (RPL) [9], and possibly protocols such as the Dynamic Host Configuration Protocol (DHCPv6) [10]and the Multicast Listener Discovery (MLD) protocols [11], as well upper layer protocols such as the User Datagram Protocol (UDP) [12, 5] and the Constrained Application Protocol (CoAP) [13].

Taken separately, those protocols offer many options. The task of selecting and including them all in an overall Machine-to-Machine (M2M) standard can be cumbersome for an industrial Standards Developing Organization (SDO)

such as the HART Communication Foundation (HCF), the International Society of Automation (ISA), or the International Electrotechnical Commission (IEC).

It makes sense for the Internet Engineering Task Force (IETF), the SDO that defines the IP protocol suite, to participate in this integration effort, and to propose a simplified bundled suite for M2M that is less cumbersome to include.

The bundled suite needs to provide evolved IETF protocols that match OT requirements and constraints, exploiting the latest technologies from the IEEE for deterministic Media Access Control (MAC), and best practices for (i) network virtualization to achieve strict flow isolation, (ii) high availability to ensure continuous operation, and (iii) security to enable trusted local and remote access [14].

To enable IT and OT technologies to converge over a shared network fabric, there is also a need to adapt the Internet Protocol suite to match the constraints and requirements of automation loops, but also provide components to enable IPv6 operations over medium access technologies such as deterministic Ethernet and IEEE802.15.4e TSCH. Moreover, there is a need to provide an architecture that ties this adapted protocol suite together in order to simplify the adoption of the bundle.

A new Working Group called 6TiSCH [15, 16] is being created at the IETF to enable IPv6 over the TSCH mode of the IEEE802.15.4e standard [17]. The WG considers an architecture in which low-power wireless devices (often called "motes") form a multi-hop Low-power Lossy Network (LLN). This LLN connects to the traditional Internet through one or more LLN Border Routers (LBRs) [18].

At the heart of IEEE802.15.4e TSCH is the notion that the nodes in the LLN communicate by following a schedule. A Time Division Multiple Access (TDMA) structure instructs each mote what to do in each slot: in other words, if it has to be active and transmit, or receive; or be inactive and thus, sleep. The way this schedule is built determines the amount of traffic that the LLN can produce, the latency of a packet and the redundancy of the network. It also determines the amount of energy each node consumes, hence the lifetime of the network. The goal of 6TiSCH is to develop a standard approach to manage this schedule and match it against the traffic needs in the network [19].

Three different modes are considered for building and maintaining this schedule:

- In the *minimal* case, the schedule is static, either preconfigured, or learnt by a node when joining the network. While it does not exploit the full benefits of IEEE802.15.4e TSCH, it can be used as a "fall-back" mode.
- In the *centralized* case, a specific schedule entity called Path Computation Element (PCE) is located onto the Internet, and continuously gathers network state information and traffic requirements from the motes in the network, adjusting the TSCH schedule accordingly.

- In the *distributed* case, motes in the LLN agree on a schedule by using distributed multi-hop scheduling protocols and neighbor-to-neighbor scheduling negotiation.

The remainder of this chapter is organized as follows. Section 2 introduces the newly published IEEE802.15.4e MAC standard, in particular its "Timeslotted Channel Hopping" (TSCH) mode. After presenting the context in which this standard was developed, Section 2 introduces the fundamental concepts of slotframes, time synchronization, channel hopping and schedule. Section 3 presents the architecture envisioned by 6TiSCH. After identifying the scope, Section 3 presents the protocol stack, and in particular the 6top sublayer responsible for managing the TSCH schedule. It also covers the different scheduling modes and forwarding mechanisms. Section 4 presents the range of applications enabled by 6TiSCH. Finally, Section 5 concludes this chapter by presenting the on-going work and future milestones in the 6TiSCH group.

2 IEEE802.15.4e Timeslotted Channel Hopping

The IEEE802.15 Task Group 4e (TG4e) was created in 2008 to *redesign* the existing IEEE802.15.4-2006 Medium Access Control (MAC) standard to obtain a low-power multi-hop MAC better suitable to emerging needs of embedded industrial applications. The final goal of the TG4e was to overcome the two main drawbacks of the IEEE802.15.4 MAC, consisting in (i) the high energy consumption of router nodes which require their radio to be always on, regardless of their actual traffic; and (ii) the high level of interference and fading, due to the fact that all communication happens on a single frequency.

The IEEE802.15.4e standard [17] has been published in 2012 as an amendment of the IEEE802.15.4-2011 MAC protocol [20]. Three different MACs have been proposed by the 15.4e TG:

- Low Latency Deterministic Network (LLDN)
- Timeslotted Channel Hopping (TSCH)
- Deterministic and Synchronous Multi-channel Extension (DSME)

The Timeslotted Channel Hopping (TSCH) mode facilitates multi-hop operation and deals well with fading and interference. The TSCH mode of the IEEE802.15.4e standard has been the focus of the activities of the 6TiSCH group, and the object of this chapter.

At the core of the TSCH is a medium access technique which uses time synchronization to achieve ultra low-power operation and channel hopping to enable high reliability. This is very different from the "legacy" IEEE802.15.4 MAC protocol.

IEEE802.15.4e does not amend the physical layer. That is, it can operate on any IEEE802.15.4-compliant hardware. The IEEE802.15.4 PHY is arguably the standard with the longest-standing impact in low-power wireless

mesh technology, and has been widely used by low-power battery-powered devices to build Low Power and Lossy Networks (LLN).

The need to interconnect IEEE802.15.4-based low power networks to the Internet has triggered the birth of various working groups (WGs) within the IETF, including 6LoWPAN [21], ROLL (the group behind the RPL routing protocol [9]), and CORE (behind the CoAP web transfer protocol [13]) that have defined how to fit an IPv6 protocol stack on top of IEEE802.15.4. Given the appealing features of the IEEE802.15.4e for enabling ultra-low power and reliable LLNs, the 6TiSCH WG aims at building IPv6-enabled LLNs, rooted in the IEEE802.15.4e TSCH MAC layer.

The basic concept of TSCH (i.e. the combination of time synchronization and channel hopping) is not new. It was introduced by Dust Networks in 2006 in its proprietary Time Synchronized Mesh Protocol (TSMP) [22]. The core ideas of TSMP then made it into standards such as WirelessHART (2007) [23] and ISA100.11a (2009) [24, 25]. These standards have targeted the industrial market, which requires ultra-high reliability and ultra-low power. WirelessHART is for instance the wireless extension of HART, a long standing protocol suite for networking industrial equipment.

IEEE802.15.4e TSCH inherits directly from these industrial standards, which are already deployed as commercial products in ten of thousands of networks in operation today. TSCH is thus a proven technology. One important difference with existing industrial standards is that IEEE802.15.4e TSCH focuses exclusively on the MAC layer. This clean layering allows for TSCH to fit under an IPv6-enabled "upper" protocol stack.

2.1 Slotframe and Time Synchronization

All motes in a TSCH multi-hop network are synchronized. Time is sliced up into timeslots. A timeslot is long enough for a MAC frame of maximum size to be sent from mote `A` to mote `B`, and for mote `B` to reply with an acknowledgement (ACK) frame indicating successful reception. The duration of a timeslot is not imposed by the IEEE802.15.4e standard, and can be tuned to fit the application's needs.With IEEE802.15.4-compliant radios operating in the 2.4GHz frequency band, a maximum-length frame of 127 bytes takes about $4ms$ to transmit; a shorter ACK takes about $1ms$. With a $10ms$ slot (a typical duration), this leaves $5ms$ to radio turnaround, packet processing and security operations.

TSCH defines a timeslot counter called Absolute Slot Number (ASN). When a new network is created, the ASN is initialized to 0; from then on, it increments by 1 at each timeslot. A mote learns the current ASN when it joins the network. Since motes are synchronized, they all know the current value of the ASN, and any time. The ASN is encoded as a 5-byte number: this allows it to increment for hundreds of years (the exact value depends on the duration of a timeslot) without wrapping. The ASN is used to calculate the

frequency to communicate on, jointly with the `channelOffset`, as described in Section 2.3.

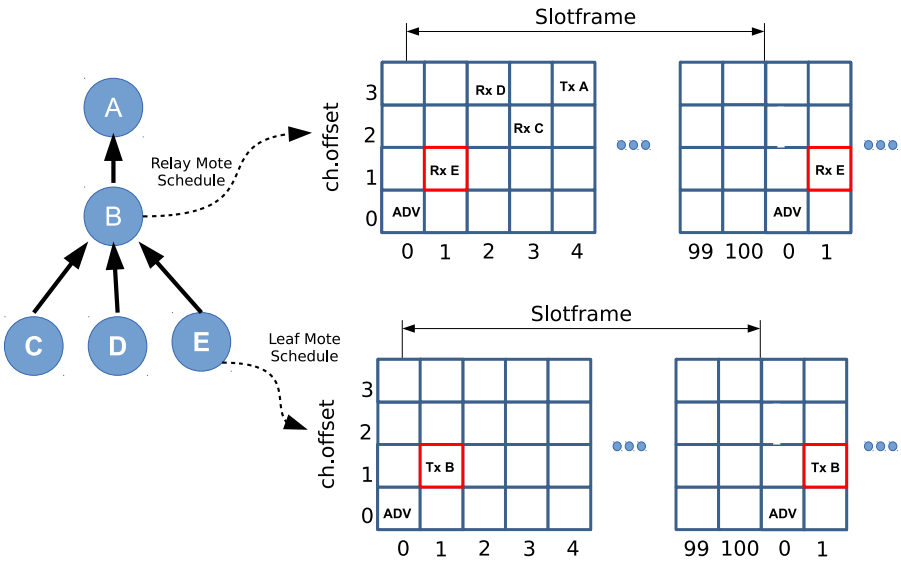


Fig. 1. Slotframe representation for a set of motes. The schedule of mote `B` is configured so it can relay information of the leaf nodes. Leaf node E is configured to be able to send a packet once every slotframe. Empty slots are "sleep" slots. The slotframe is composed of 101 slots of 15 ms each.

As shown in Fig. 1, timeslots are grouped into one or more slotframes. A slotframe continuously repeats over time. The IEEE802.15.4e TSCH standard does not impose any slotframe size. Depending on the application needs, these can range from 10s to 1000s of timeslots. The shorter the slotframe, the more often a timeslot repeats, resulting in more available bandwidth and lower latency, but also in higher power consumption.

Because of the slotted nature of communication in a TSCH network, motes have to maintain tight synchronization. All motes are equipped with clocks to keep track of time. Because clocks in different motes drift with respect to one another, neighbor motes need to periodically re-synchronize. Each mote periodically synchronizes its network clock to at least one other mote, and it also provides its network time to its neighbors. IEEE802.15.4e does not define how a mote chooses its "time source neighbor"; this is left to the upper layer, which needs to ensure that the synchronization structure is loop-free.

Nodes already in the network periodically send Enhanced Beacons (EBs) to announce the presence of the network. When a new mote boots, it listens for those to synchronize to the TSCH network. EB frames contain information about the timeslot length, the slotframes and timeslots the beaconing mote

is listening on, and a 1-byte join priority (i.e., number of hops separating the node sending the EB, and the PAN coordinator).

IEEE802.15.4e TSCH implicitly adds timing information to all packets that are exchanged. Motes are required to transmit data packets at precise time in the time slot, and timestamp the instant of arrival of received data frames. A node (re-)synchronizes to it time source neighbor using this timing information. IEEE802.15.4e defines two methods for a device to synchronize to its time source neighbor: "acknowledgement-based" and "frame-based" synchronization.

In both cases, the receiver calculates the difference in time between the expected and actual time of frame arrival. In acknowledgement-based synchronization, the receiver writes that measured offset in a field in the acknowledgement frame it sends to the sender. This allows the sender mote to synchronize to the clock of the receiver. In frame-based synchronization, the receiver uses the measured time offset to adjust its own clock. In this case, it is the receiver mote which synchronizes to the clock of the sender.

Regardless the synchronization method adopted, data traffic is used to implictly re-synchronize a node with its time source neighbor. In the absence of data traffic, motes are still required to periodically synchronize to their time source neighbor(s) to account for clock drift. If they have not been communicating for some time, motes can exchange an empty data frame, often referred to as a *Keep-alive* message, to re-synchronize. The frequency at which such messages need to be transmitted depends on the stability of the clock source, and the "guard time" duration (i.e., how "early" each mote starts listening for data). With a $10ppm$ clock source and a $1ms$ guard time, this period can be up to $100s$.

Different synchronization policies are possible in a TSCH network. Motes can keep synchronization exclusively by (i) exchanging EBs, or (ii) periodically sending valid frames to time source neighbors, or (iii) by using a combination of both. Which solution to use depends on network requirements and operational conditions. For instance, temperature variations might introduce important variations on clock alignment and might require adaptive techniques to maintain synchronization without impacting energy consumption [26]. In any case, the cost of synchronization in a TSCH network is negligible in most applications.

2.2 TSCH schedule

Timeslotted Channel Hopping is different from traditional low-power MAC protocols because of its scheduled nature. In a TSCH network, each mote follows a **schedule**. This schedule looks like a matrix of width equal to the slotframe size, and of height equal to the number of available frequencies (as shown in Fig. 1).

A single element in the TSCH schedule, identified by a `slotOffset`, and a `channelOffset`, is called a **cell** [27]. A cell can be considered as an atomic

unit to be allocated by a scheduling algorithm. Because of the channel hopping nature of TSCH (see Section 2.3), the scheduling algorithm does not need to worry about the actual frequency communication happens on, since it changes at each slotframe iteration.

A TSCH schedule instructs each mote what to do in each timeslot: transmit, receive or sleep. A cell can therefore be scheduled or unscheduled. During an unscheduled cell, the node does not communicate. When a cell is scheduled, it is assigned a MAC-layer slotframe identifier, a neighbor MAC address (which can be the broadcast addressand one or more of the following flags: `TX`, `RX`, `shared`, `timeskeeping`, `hard` (see Section 3.1 for details). Note that a broadcast cell is an alias for "a scheduled cell with neighbor address the broadcast address".

The TSCH schedule contains all the scheduled cells from all the slotframes and is sufficient to qualify the communication in the TSCH network. Once a mote obtains its schedule, it executes it:

- For each `TX` cell, the mote checks whether there is a packet in the outgoing queue which matches the neighbor written in the schedule information for that timeslot. If there is none, the mote keeps its radio off for the duration of the timeslot. If there is one, the mote can ask for the neighbor to acknowledge it, in which case it has to listen for the acknowledgement after transmitting.
- For each `RX` cell, the mote listens for possible incoming packets. If none is received after some listening period, it shuts down its radio. If a packet is received, addressed to the mote, and passes security checks, the mote can send back an acknowledgement, if requested.

Assuming the schedule is well built, if mote `E` is scheduled to transmit to mote `B` at `slotOffset` 1 and `channelOffset` 1, mote `B` will be scheduled to receive from mote `E` at the same `slotOffset` and `channelOffset`, as depicted in Fig. 1.

If there is a lot of data flowing from mote `E` to mote `B`, the schedule might contain multiple cells from `E` to `B` in the same slotframe. Multiple cells scheduled to the same neighbor are typically equivalent, i.e., the MAC layer sends the packet on whichever of these cells happens to show up first after the packet was put in the MAC queue. The union of all these cells identified by different `[slotOffset, channelOffset]`, which are scheduled for a same purpose, with the same neighbor (e.g., between two neighbors, `E` and `B`), with the same flags, and the same slotframe, is called a **bundle** [27]. Since the slotframe repeats over time, each cell gives a "quantum" of bandwidth to a given neighbor. Modifying the number of cells in a bundle modifies the amount of resources allocated between two neighbors. An example of bundle is shown in Fig. 2.

By default, each scheduled `TX` cell within the TSCH schedule is dedicated, i.e., reserved only for mote `E` to transmit to mote `B`. IEEE802.15.4e allows also to mark a cell as shared. In a shared cell, multiple motes can transmit at

the same time, on the same frequency. To avoid contention, TSCH defines a back-off algorithm for shared cells.

A scheduled cell can be marked as both transmit (`TX`) and receive (`RX`). In this case, a mote transmits if it has an appropriate packet in its output buffer, listens otherwise. Marking a cell as `[TX,shared,RX]` results in slotted-Aloha behavior.

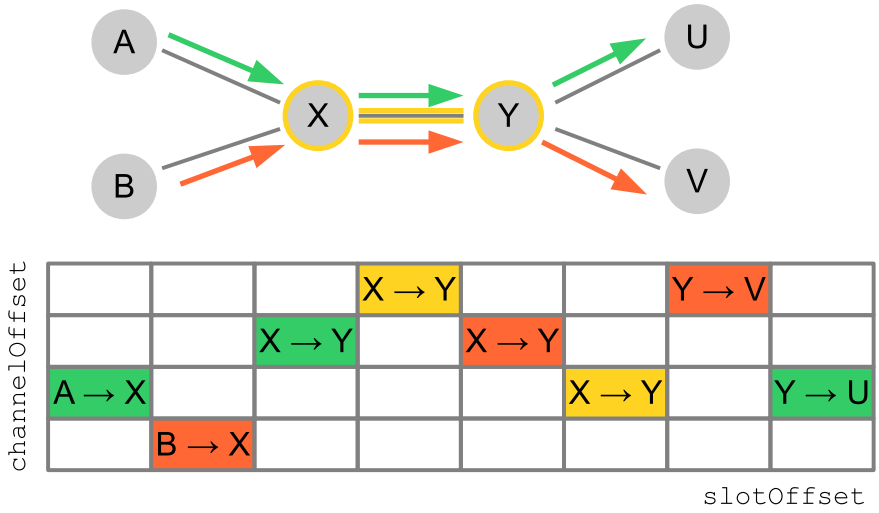


Fig. 2. Example of TSCH schedule including a bundle (in yellow), and two distinct tracks (in green, and orange, respectively) for a simple network scenario.

A sequence of cells scheduled along a multi-hop path is called a **track** [27] (see Fig. 2). It is the result of a reservation, and it belongs to the node that initializes the process for establishing the track itself.

The schedule defines entirely the synchronization and communication between motes. By adding/removing cells between neighbors, one can adapt a schedule to the needs of the specific application. It is possible to:

- Make the schedule "sparse" for applications where motes need to consume as little energy as possible, at the price of reduced bandwidth.
- Make the schedule "dense" for applications where motes generate a lot of data, at the price of increased power consumption.
- Add more cells along a multi-hop route over which many packets flow.

TSCH defines the mechanisms to execute a communication schedule. Yet, how the schedule is built, updated and maintained, and and by which entity, is outside of the scope of the IEEE802.15.4e standard. Several scheduling approaches have been investigated by the research community.

A centralized Traffic Aware Scheduling Algorithm (TASA) has been proposed recently [28]. It exploits matching and coloring procedures to schedule cells and tracks to all the motes across the entire network topology graph. In detail, it allows to set up the TSCH schedule, based on the network topology and the traffic load. Thus, it uses the information related to the paths, coming from the routing protocol (e.g., RPL), and those related to the traffic (e.g., average traffic load generated by each node) in order to schedule cells and tracks, and provide the required level of QoS (duty cycle, throughput, etc.) to each flow.

Decentralized approaches have been studied in the context of the OpenWSN project [29]. uRes [30] proposes a bargaining based approach where nodes negotiate with their neighbors to allocate cells. uRes allocates cells minimizing the number of collisions as nodes have a certain knowledge of their neighbors schedule. Collisions can still occur; they are resolved by re-allocating the colliding cells [31].

A different decentralized approach has been published recently by Morell *et al.* [32]. The article introduces the concept of label switching in TSCH networks and proposes the use of reservation to establish and manage tracks between nodes in the network. This approach computes the schedule of the network by collecting information along the track and installing it during the downstream reservation message, in a way similar to the RSVP standard [31].

The aforementioned protocols [28, 30, 32] can be seen as candidate scheduling solutions within the 6TiSCH WG.

2.3 Channel Hopping

The TSCH mode of IEEE802.15.4e combines time synchronization and channel hopping. It is thereby able to provide ultra-high reliability and ultra-low power. For each scheduled cell, the schedule defines a `slotOffset` and a `channelOffset`. The latter is translated by both communicating motes into a frequency using (1).

$$frequency = F(channelOffset + ASN)\%nFreq \tag{1}$$

The function `F` consists of a look-up table containing the set of available channels. The size of this look-up table is equal to the number of available frequencies, $nFreq$. There are as many `channelOffset` values as there are frequencies available (e.g., 16 when using IEEE802.15.4-compliant radios at 2.4GHz, when all channels are used). Since both motes have the same `channelOffset` written in their schedule for that scheduled cell, and the same ASN counter since they are synchronized, they compute the same frequency. At the next iteration (cycle) of the slotframe, however, the `channelOffset` is the same, but the ASN will have changed, resulting in the computation of a different frequency.

Fig. 3 illustrates this behavior. It highlights a single cell in a 31-slot long slotframe. This cell is at `slotOffset` 14, at `channelOffset` 11 and at ASN

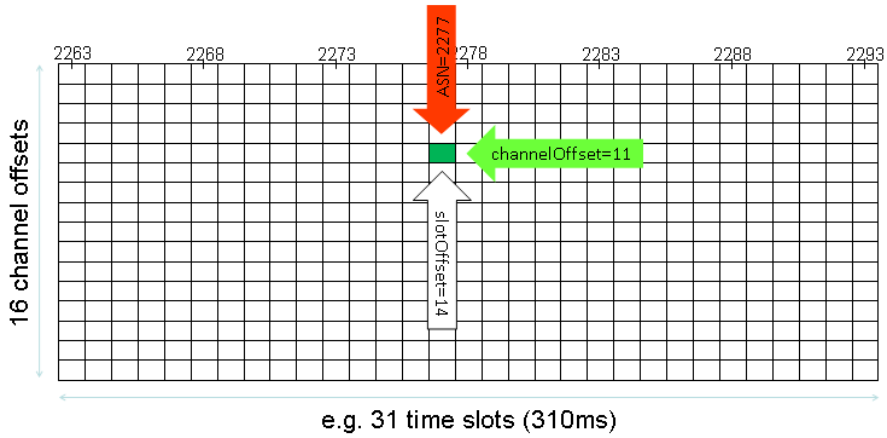


Fig. 3. Channel hopping happens even when the schedule does not change.

2277 (at this iteration of the slotframe). When communicating, the neighbor nodes using this cell will apply (1) to obtain a certain frequency. At the next iteration of the slotframe, even is the cell is still at the same `slotOffset` and `channelOffset`, the ASN will have changed and applying (1) will result in a different frequency calculation.

The channel hopping nature of TSCH causes links to be very stable. Wireless phenomena such as multi-path fading and external interference impact a wireless link between two motes differently on each frequency. If a transmission between two mote fails, retransmitting on a different frequency has a higher likelihood of succeeding that retransmitting on the same frequency. As a result, even when some frequencies are "misbehaving", channel hopping averages the contribution of each frequency, resulting in more stable links, and therefore a more stable topology.

2.4 TSCH Deterministic Networks

The IEEE802.15.4e TSCH, due to the combination of Time Division Multiplexing (TDM), time synchronization and the time formatted into slotframes, results in a *deterministic* wireless MAC standard, suitable for deterministic traffic, i.e., traffic flows with an emission rate and routing path patterns that are well-known in advance. For such traffic, a deterministic network allocates the required resources (buffers, processors, medium access) along the multihop routing path at the precise moment the resources are needed. In a TSCH network, the bandwidth is pre-formatted in a TDM fashion. Thus, unlike the traditional CSMA/CA-based networks, there is no contention for gaining access to the channel. In fact, a time slot becomes a unit of throughput that can be allocated to a given deterministic flow. Deterministic networks are also

adapted by *Traffic Engineering*, i.e., to support several isolated flows. Different optimized paths and tracks can be built for all the isolated flows, based on their specific requirements.

A deterministic network guarantees timely transmission. A good example of a deterministic network is a railway system, where trains are scheduled periodically to leave a railway station at a certain time, to traverse the stations along a predetermined track at precise times so that, in the end, a given train arrives at its final station at the expected time, with virtually no jitter from a human perspective. Moreover, collisions are eliminated: there is never another train blocking the rail and delaying this train.

The IEEE802.15.4e TSCH, with its deterministic behavior, opens up a set of new applications which require high reliability and low-power. It enables control loops in wireless process control/automation networks, where wired solutions have been used so far. Supporting Traffic Engineering and isolated traffic flows, it enables *umbrella* networks, transporting data from different independent clients. Each flow is isolated and shaped, according to the requirements specified by the client. With its low-power nature and predictable power consumption, TSCH enables networks of energy harvesting motes. Finally, it supports widespread monitoring (like corrosion monitoring or pipe leak detection), which requires slow periodic reporting rates from a large number of sensors and open loop operation. More details about potential 6TiSCH applications are provided in Sec. 4.

These new applications will make a lot more sense if they are able to reuse the same building blocks and thus share a same converged network for IT/OT, based on an IPv6-architecture.

3 The 6TiSCH Architecture

In a large extensive factory floor, hundreds to as many as tens of thousands of constrained field devices might be deployed as a single LLN and share a same physical environment. The logical structure of the radio meshes vary as the conditions change, even in the absence of physical movement.

The memory and processing resources of a constrained device such as a battery-operated sensors or actuators are drastically limited, so all states, including those related to addressing and routing, must be kept to a minimum.

For an IPv6-enabled device, it is commonplace to save an extra identifier and use a (the) IPv6 address of the device as a unique identifier. This implies that the IPv6 address is permanent – the device cannot be renumbered – as long as it keeps playing the role that corresponds to that device identifier.

It results that the device retains its prefix information quasi-permanently, and thus can only be reachable over IP within the range of the network where the IPv6 subnet associated to that prefix is defined. In order to allow mobility and unhindered reorganization of the routing topology into multiple small mesh networks, it is necessary that the subnet spans the whole factory floor.

For a large factory, this means that potentially thousands of field devices will form a single subnet, sharing an IPv6 prefix from which all their global (or unique-local) IPv6 addresses are derived.

A possible model to scale a Low-power Lossy Network (LLN) to the thousands as a single IPv6 subnet consists in laying out a high-speed backbone that spans the area and provides a fast transit facility to federate the network. A mesh protocol partitions the LLN in a collection of logical graphs such as Destination Oriented Directed Acyclic Graphs (DODAGs) oriented towards the backbone. IPv6 routing or ND proxy operation over the backbone enable the overall connectivity effectively forming a so-called multilink subnet.

In existing deployments, the high-speed backbone may effectively be an Ethernet Switched fabric, or a wireless mesh network based on IEEE802.11 technology. A Mesh Access Point connects a specific Gateway that terminates IP flows towards the backbone. The Gateway connects to the field devices over a LLN mesh that is a based on a variation of the TSCH technology that guarantees deterministic transmissions, and may either use IPv6 as a Network Layer or completely bypass that layer.

The natural evolution of that model is to implement the end-to-end principle that sustains the Internet, based in the Internet Protocol. An application agnostic (Layer 3) Router physically replaces the Gateway, so that the end-to-end flows from a field device are no more constrained to terminate at that point, but may flow over a converged IT/OT infrastructure to terminate in servers that are located in the carpeted floor.

6TiSCH will document that evolved model as an open standards-based architecture, highlight best practices, and standardize the missing components to achieve industrial-grade performance in terms of jitter, latency, scalability, reliability and low-power operation for IPv6 over IEEE802.15.4e TSCH [17]. Although not addressed directly by 6TiSCH, it is envisioned that the resulting techniques will be applicable to technologies other than 2.4GHz IEEE802.15.4 [20].

As illustrated in Fig. 4, the scope of the architecture is a potentially large IPv6 multilink subnet that may span thousands of LLN devices, federated over a backbone, as discussed above.

The architecture will address how multiple BBRs are supported for a higher degree of scalability and reliability, and how nodes maintain synchronization in the presence of multiple BBR [33]. This work implies new IPv6 ND proxy operations as illustrated in Section 3.6.

When possible, the architecture will reuse existing protocols such as IPv6 Neighbor Discovery (ND) [8], IPv6 Low power Wireless Personal Area Networks (6LoWPAN) [21], and the Routing Protocol for Low Power and Lossy Networks (RPL) [9], with the minimum adaptation required to meet criteria for reliability and determinism within the mesh, and scalability over the backbone.

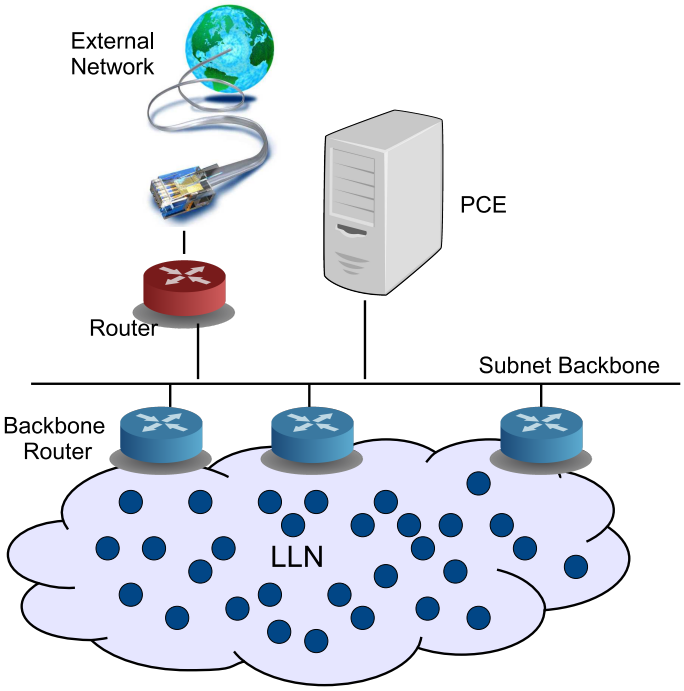


Fig. 4. 6TiSCH reference architecture.

3.1 The 6TiSCH stack

The 6TiSCH working group aims at filling the gaps between IEEE lower layers of the protocol stack and the IETF higher layers, to enable an open standards-based protocol stack for deterministic wireless mesh networks. Fig. 5 presents a general overview of the integrated protocol stack rooted at the IEEE802.15.4 physical layer and operated by the IEEE802.15.4e TSCH MAC amendment that provides determinism while ensures very low power operation. As described earlier, missing gaps need to be filled by 6TiSCH so that IETF 6LoWPAN Header Compression and RPL, which enables IPv6 packetization and routing, can optimally operate on top of the TSCH MAC layer.

The deterministic nature of IEEE802.15.4e TSCH and its strict requirements in terms of schedule management make it necessary to have a sublayer which enables management entities (MEs) to operate the network. Therefore, one of the main goals of the 6TiSCH WG is to define the 6top sublayer (see Section 3.1).

The 6TiSCH architecture [34] specifies how packets that belong to a deterministic IPv6 flow are marked and routed or forwarded over the mesh within jitter and latency budgets. The schedule management translates into routes and track management.

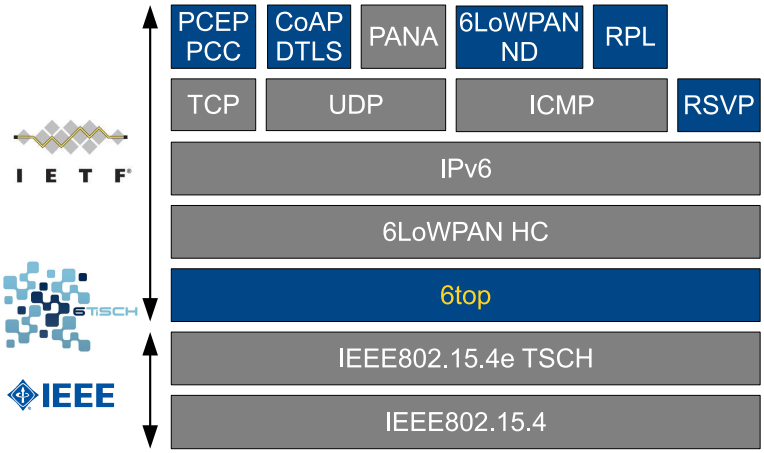


Fig. 5. 6TiSCH IPv6-enabled protocol stack for LLNs.

Route computation can be achieved either in a centralized fashion by a Path Computation Entity (PCE), which is located either on the backbone or farther in the IPv6 network over a backhaul, or in a distributed fashion using RPL and a multi-path resource reservation protocol.

Track allocation can be globally optimized and then pushed on the network from the PCE that computes the routes, and/or managed by a distributed scheduling protocol along routes that are computed by RPL.

In detail, as shown in Fig. 5, in centralized approaches, schedule management is handled by a PCE and its control messages are transported by protocols such as PCEP/PCC [35] or COAP/DTLS [13] on top of UDP. In decentralized approaches, the management entities benefits from reservation protocols such as RSVP [31] or NSIS [36] where QoS requirements are transported and installed along a path, and then implemented as cell reservation in the schedule of each node on the path.

The 6TiSCH architecture also covers security. The security requirements will be identified, and then the group will work on a secure and scalable key management framework (and requirements of related protocols) for 6TiSCH networks. The Protocol for carrying Authentication for Network Access (PANA) [37] is a potential candidate as Key Management Protocol for the bootstrapping phase of the 6TiSCH networks. It is a promising protocol since it supports mutual authentication, stateless authentication relay function [38] and encrypted distribution of attributes [39].

The 6top sublayer

In the 6TiSCH architecture [34], the 6top sublayer [40] is built on top of IEEE802.15.4e TSCH MAC layer, and allows a management entity to drive the TSCH schedule. 6top also includes statistics collection functionality, which

an upper layer (including the RPL routing protocol) can use to gather connectivity information. The 6top sublayer offers management commands to upper layers in order to operate, configure and enforce QoS at the MAC layer. Monitoring processes are used to determine that particular cells do not perform as well as expected enabling its rescheduling so deterministic behavior can be maintained.

6top is designed to be used with several scheduling approaches. In a centralized approach, a central PCE collects topology and traffic requirements, used to build a communication schedule, which it then sends to the different nodes in the network. In a decentralized approach, nodes compute their own schedule according to local information or by using a decentralized resource reservation protocol. To enable both approaches, 6top adds a new IEEE802.15.4e `LinkOption` flag [17]; in addition to the `TX`, `RX`, `Shared` and `Timekeeping` flags, a cell is also qualified as either a *hard cell* or *soft cell*. This option is mandatory; all cells are either hard or soft.

- A *hard cell* is a cell that cannot be dynamically reallocated by 6top. This type of cell is typically scheduled by a PCE. Once installed, only the PCE can move it inside the TSCH schedule, or delete it. When installing a hard cell, the PCE indicates the exact `slotOffset` and `channelOffset` of the cell.
- A *soft cell* is a cell that can be reallocated by 6top dynamically. This type of cell is typically scheduled by a distributed scheduling entity in the upper layer of 6top. Instead of specifying the exact `slotOffset` and `channelOffset`, the scheduling entity indicates how many cells must be scheduled to a given neighbors. 6top is responsible for allocating specific `slotOffset` and `channelOffset` values corresponding to the request from the scheduling entity. Furthermore, the monitoring process of 6top keeps tracking the performance of each cell to the same neighbor. If a cell performs significantly worse than others scheduled to the same neighbor, 6top reallocates this cell at different `slotOffset` and `channelOffset` inside the TSCH schedule.

When using a centralized scheduler, the PCE needs a protocol to send schedule updates to the nodes in the network. Candidate protocols include PCEP [35], OpenFlow [41], and ForCES [42].

When using a distributed scheduler, a protocol is needed to reserve MAC-layer resources along the multi-hop path identified by RPL, to satisfy certain QoS constraints (e.g., bandwidth, latency). Candidate protocols include RSVP [31, 32] or NSIS [36]. NSIS provides the semantics for transport layer packets to visit each node along a multi-hop RPL path, and indicating Quality Of Service (QoS) requirements. Upon reception of a QoS request, the 6top layer configures the appropriate MAC layer resources.

6top maintains statistics about the performance of scheduled cells. When using a centralized scheduler, this information is periodically sent to the PCE, which continuously adapts the schedule and sends schedules updates

as needed. This information can also be used by the RPL protocol's objective function.

6TiSCH network can transport different types of traffic, possibly for different administrative entities (e.g., lighting and HVAC data in a smart building), possibly with different QoS constraints. Thanks to the slotted nature of IEEE802.15.4e TSCH, 6top can mark different cells with identifiers of those different flows. This can result in perfect isolation. For example, the amount of HVAC traffic has no effect on the latency of the lighting traffic. This allows for true *umbrella* networks, managed by a network operator, and transporting data for different clients. An example is an urban network which is used to transport data from weather sensors, and actuation commands for the municipal sprinkler system.

When a packet enters the 6TiSCH network, the 6top layer at the ingress device identifies the service this packet belong to, and marks the packet, possibly by using DSCP field in the 6LoWPAN header. When traveling through the 6TiSCH network, each mote uses that marker to decide on which cell to transmit.

The 6top sublayer provides a management interface to a Management Entity (ME). The interface consists of bidirectional message flows, one is from ME to 6top sublayer (`commands`); and another from 6top sublayer to ME (`status` or values of `MIB attributes`). The messages are defined in Table 1.

The Management Entity (ME) exists in an upper layer, which is responsible for exchanging control messages via network, and mapping the control messages to/from the management interface of 6top. For the centralized scheduling, five different control flows have been defined. In detail, as shown in Fig. 6, there are: *action* flow and *query* flow from PCE to nodes; *report* flow, *event* flow, and *schedule update* flow from nodes to PCE. In order to help the understanding of these flows, and related messages, let us consider the action flow, for instance. The PCE sends a control message to node `A` to create hard cells. Then, the ME in node `A` translates the control message to `CREATE.hardcell command` of 6top, and 6top returns `status Success` or `Failure` to the ME. Finally, ME encodes the status into control message to PCE. Similarly, decentralized protocols such as RSVP or NSIS work with 6top through a Management Entity.

3.2 Centralized Scheduling

The 6TiSCH WG will initially focus on the definition of a static *minimal* schedule, which is pre-configured, or learnt by each mote when joining the network. The minimal 6TiSCH configuration draft [43] defines the basic setup for a TSCH network to inter-operate. Such setup includes the definition of a pre-configured schedule, the content of Enhanced Beacons, the TSCH MAC layer slot timing, the policy for selecting the time parent, and some insights for using the RPL Objective Function Zero [44] in a basic TSCH deployment.

ME to 6top	6top to ME
CREATE /DELETE/UPDATE .hardcell	Success/Failure
CREATE/DELETE/REALLOCATE .softcell	Success/Failure
CREATE/DELETE/UPDATE .slotframe	Success/Failure
CONFIGURE.monitoring	Success/Failure
CONFIGURE/RESET .statistics	Success/Failure
CONFIGURE.eb	Success/Failure
CONFIGURE.timesource	Success/Failure
CREATE/DELETE/UPDATE .neighbor	Success/Failure
CREATE/DELETE/UPDATE .queue	Success/Failure
CONFIGURE.security	Success/Failure
CONFIGURE.security.macKeyTable	Success/Failure
CONFIGURE.security.macSecurityLevelTable	Success/Failure
LabelSwitching.map	Success/Failure
LabelSwitching.unmap	Success/Failure
READ.cell	configuration of a specific cell
READ.slotframe	configuration of a specific slotframe
READ.monitoring.status	allocated/provision cells
READ.statistics	statistic MIB for given parameters
READ.eb	MIB of a specific Enhanced Beacon
READ.timesource	timesource information in MIB
READ.neighbor	specific neighbor's MIB
READ.all.neighbor	all neighbors in neighbor table
READ.queue.stats	queue configuration in MIB

Table 1. 6top sublayer management interface

Another approach is to use centralized scheduling. In the centralized approach, a key role is played by a Path Computation Element (PCE), which is a gateway connecting the network to the Internet (as shown in Fig. 4). It can be seen as the Management Entity (ME) responsible for building and maintaining the TSCH schedule. The PCE therefore can collect network state information and traffic requirements from the motes. Having a global view of the network, the PCE can build the schedules, making sure that the QoS requirements of all the network traffic flows are properly met.

Two different approaches can be used by the PCE for installing a track in the network. In detail, the PCE can

- talk to each node on the track individually, or
- talk only to the source node, which uses a separate protocol to install the resources along the track.

While the first approach seems to be easier to maintain, and also the one able to better ensure the correct installation of the track, the second one minimizes the amount of control traffic in the network between the PCE and the LLN.

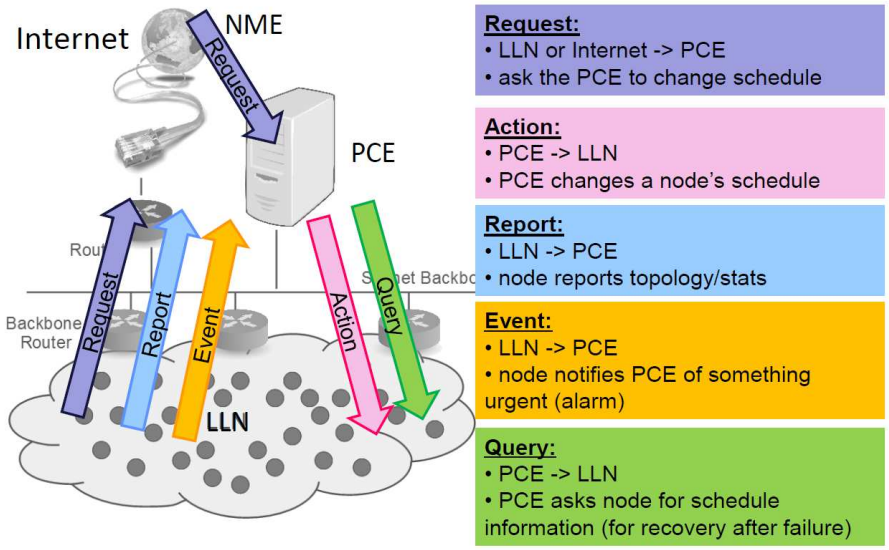


Fig. 6. Control flows defined between the Network Management Entity, NME (i.e, admin console, mobile handheld), or the ME (i.e.,PCE) and the nodes in the LLN, for the centralized scheduling.

6TiSCH will define the mechanism and format for control messages to be transported between the motes in the network. The 6TiSCH WG will first define the requirements for the protocol used by the PCE to communicate with a 6TSCH network, in order to identify existing protocols which may address (parts of) its needs. Transport protocols such as PCEP/PCC, and COAP/DTLS are possible candidate; they will be investigated and maybe adapted to the needs of the 6TiSCH architecture.

The Path Computation Element Communication Protocol (PCEP), specified in the RFC5440 [35], defines how to set up the communications between a Path Computation Client (PCC) and a PCE, or between two PCEs. In detail, the PCC can ask for path computation to the PCE, using a `PCReq` message that carries the specific requirements (e.g., bandwidth, metric-list, etc.) for the requested path. Upon reception of this request, the PCE replies with a `PCResp` message that specifies if the requirements, and thus, the path request, can be satisfied. Therefore, PCEP could be suitable for carrying the scheduling requirements of each traffic flow from the mote (playing the role of PCC) to the PCE. Security aspects such as data confidentiality, integrity and authentication will have to be taken into account while developing such centralized solution.

CoAP [13] appears as an interesting candidate as a transport mechanism between the central PCE or Management Entity (ME) and the 6top sublayer at the node. CoAP offers REST semantics with delivery guarantees while keeping low energy consumption profile. Operations on the 6top layer can be exposed as CoAP resources and accessed through well understood REST operations matching the operational flows between the ME and 6top as defined in the previous section. CoAP enables *Confirmable* and *Non-Confirmable* messages providing flexibility for those signals that need to be acknowledged and those events that can be transmitted unreliably. In addition, CoAP enables response piggybacking in message confirmation which reduces the cost of end-to-end acknowledgements plus responses. CoAP enables resources to be observed, analogously to an observer pattern, clients (e.g., the ME) can subscribe to resources on the server (e.g., a node in the network) and get notification upon changes. This feature enables the ME or PCE to subscribe to network information at nodes and be updated every time a node records new information.

The IETF DICE working group [45] is defining a subset of functionalities of DTLS to be supported in LLNs and together with CoAP provide a reliable, low footprint and secure framework to transport Management Entity requirements to 6top and vice-versa.

3.3 Distributed Scheduling

Distributed scheduling is analogous to building a set of paths (i.e., tracks) between nodes in the network. The direction of the tracks and the resources allocated for each of them are application-dependent and must be a parameter for the reservation mechanism. As in the case of transport networks (networks operating in the core of Internet), reservation of resources might be carried out by dedicated management protocols such as RSVP [31] or NSIS [36].

The operation of the protocol is driven by a path reservation request that travels along a existing track and ensures that there is connectivity between the two endpoints. At the end of the track, the reservation message is built and forwarded back, ensuring that the resources are available. The overlay used to transport the reservation requests is given by a distributed routing protocol such as RPL. In the 6TiSCH architecture, RPL is used to construct routing topologies using underlying L2 links.

In a distributed scheduling case, a LLN is formed by nodes receiving Enhanced Beacon (EB) messages from neighbors and selecting a best candidate to become its time source neighbor. Enhanced Beacons contain minimal information for nodes to establish a best effort schedule (i.e, shared cells) so they can start communicating [43]. Over these shared cells, RPL DIO and DAO control messages are used to discover and build an overlay topology used for routing. These best effort routes are used initially by nodes to reserve resources along a track (i.e., to a destination node – which usually is

the DAGRoot). Once a best effort route between two nodes exists, the distributed Management Entity (an application running on the node), can start requesting resources to build a track.

This process is analogous to the reservation of resources in a wide area transport network. A reservation protocol (or a label distribution protocol) is used to transport QoS requirements (e.g., requested bandwidth) along a track. The 6TiSCH working group is evaluating different approaches to tackle transport of QoS requirements in so constrained devices, including reduced versions of RSVP and NSIS. The idea that the transport protocol, hop by hop, should require 6top sublayer to implement the provision of the required QoS on that hop. Upon a request for bandwidth allocation, for example, 6top can start a pairwise negotiation with the next hop node in order to agree on a set of cells to be used (Fig. 7).

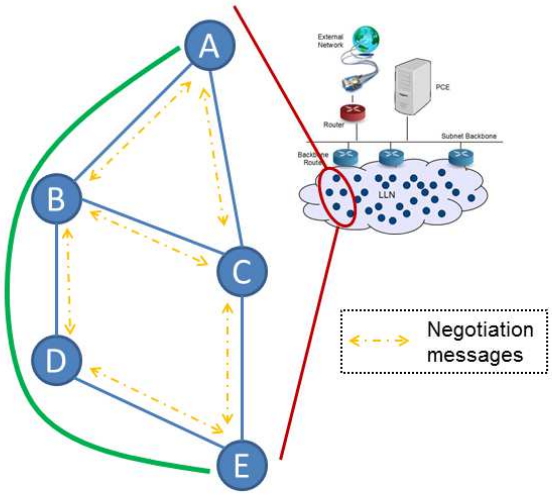


Fig. 7. Distributed scheduling applied to a set of nodes in the LLN. Node `A` sends a packet along the multi-hop path (track) `A-B-D-E`. At each hop, neighbor motes negotiate with one another to add cells into their TSCH schedule. A 6top monitoring process is needed to recover from topological changes and collisions.

How these cells are allocated is also in scope of 6top sublayer, and a set of recommendations will be defined by the working group, including an evaluation of a simple random cell allocation. Upon the request from a higher layer reservation protocol, the 6top sublayer will maintain the QoS levels as required. This task will also be transparent to the transport protocol, which will ensure that the QoS is enforced along the track. On its behalf, 6top con-

tinuously monitors the underlying MAC layer resources ensuring that QoS is met in a hop by hop basis by means of cell reallocation or over-provision.

3.4 Routing in the 6TiSCH Architecture

Even though the initial efforts will concentrate on distributed routing over a static schedule, the 6TiSCH architecture ultimately aims at enabling a mix of centralized and distributed routing over a dynamic schedule.

A distributed routing model such as offered by RPL can react rapidly and autonomously to changes in the network, but misses knowledge on the global capacity of the network vs. the load of individual links. It will typically direct all flows to a given destination over specific (shortest) paths, adding some excess load on some links when capacity exists elsewhere in the network, which limits its capability to serve optimally the end-to-end requirements of individual flows.

On the other hand, a centralized routing model such as offered by a PCE benefits from an overall perspective (also known as "God's view"), which enables the separate computation of multiple discrete paths that are globally optimized to meet their individual sets of constraints.

Either way, the path computation may be used for traditional hop-by-hop routing at the network layer, or for end-to-end switching at a lower layer along a predetermined track. Tracks can be installed on a per flow basis, so as to match a reservation with particular constraints along a path that is not necessarily shortest, whereas routes are typically shared by all flows as they converge towards a destination. It results that hop-by-hop operations are generally associated with distributed routing computation, whereas track operations are generally associated with centralized path computation, but that is not necessarily always the case.

3.5 Forwarding in the 6TiSCH Architecture

The 6TiSCH architecture supports traditional IPv6 routing and forwarding, but in order to cover deterministic flows, the architecture also supports lower layer switching operations along predetermined tracks.

IPv6 Forwarding refers to the traditional network layer operation whereby a router selects a next-hop adjacent router for a given packet, typically based on a destination address in the network layer header of the packet. This operation is performed at each hop along a path, using a routing table (also known as Routing Information Base, RIB) that can be programmed by the PCE, computed dynamically through the RPL protocol or a mix of the two. At least one bundle of cells is associated to the link to the next-hop router.

A typical IPv6 Forwarding operation includes the following steps:

- Selection of a next-hop router based on network layer information in the packet (typically the destination IPv6 address) using a routing table;

- Selection of a link (a bundle of cells) to reach that router (if there are multiple, then QoS information may come into play);
- Edition of the MAC layer header to indicate the next-hop;
- Queuing for transmission over the selected bundle based on QoS information;
- (later) Transmission over a cell in the bundle based on queue priority and ageing (this is the actual scheduling operation).

Track forwarding refers to a deterministic Generalized Multi-Protocol Label Switching (G-MPLS) [46] operation whereby a 6TiSCH node can switch a frame based on the cell of arrival as opposed to, classically, information in a header inside the frame. A track can be installed either by the PCE or by a reservation protocol, in which case the routes that are already present in the RIB will be used to select the path for the track. A Cell Switching Table uniquely binds a set of receive cells to a set of transmit cells, representing a forwarding state that can be used regardless of the upper layer protocol.

A typical Track Forwarding operation includes the following steps:

- Selection of the transmit cell or bundle based on the receive cell or bundle using the Cell Switching Table;
- Queuing for transmission over the bundle based on QoS information;
- (later) Transmission over a cell in the bundle based on queue priority and ageing (this is the actual scheduling operation).

Fragment Forwarding is an additional technique that is used to avoid the reassembly of 6LoWPAN fragments at each hop along a path in the LLN. The first fragment is routed using the IPv6 Forwarding method, and a state is installed for the subsequent fragments to follow the same route without a need to look up the routing table. The state is indexed by the *datagram tag* that is present in all fragments and identifies uniquely the packet. The *datagram tag* is locally significant; it is attributed at each hop upon the first fragment of a packet and switched in a classical MPLS fashion [47] upon the subsequent fragments [48].

Fig. 8 shows the three forwarding techniques supported by the 6TiSCH architecture applied to the track installed between the node `A` and node `U` in Fig. 2. In detail, red, green and yellow arrows correspond respectively to Track, Fragment and IPv6 Forwarding. It has to be noticed that at the relay nodes (i.e., node `X` and node `Y`) , the packet (or fragment) will reach a different layer of the protocol stack, according to the specific forwarding approach adopted in the network.

3.6 IPv6 Neighbor Discovery

Reachability within a subnet is classically obtained through the IPv6 Neighbor Discovery Protocol (NDP) [8]. But IPv6 ND relies heavily on multicast signalling messages on the local link, which is workable over the backbone

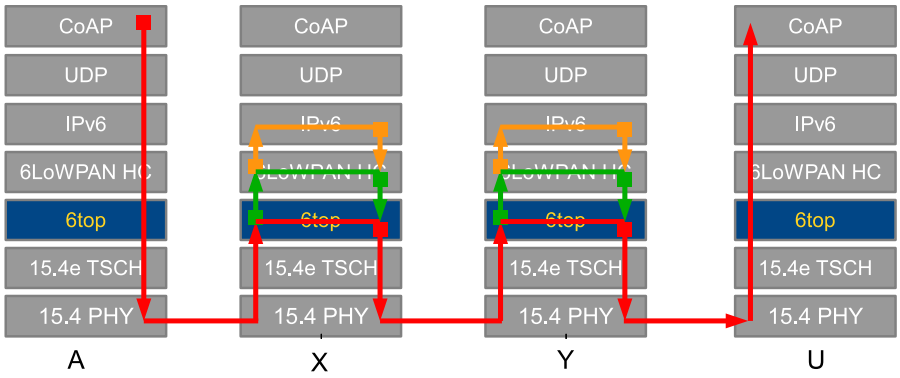


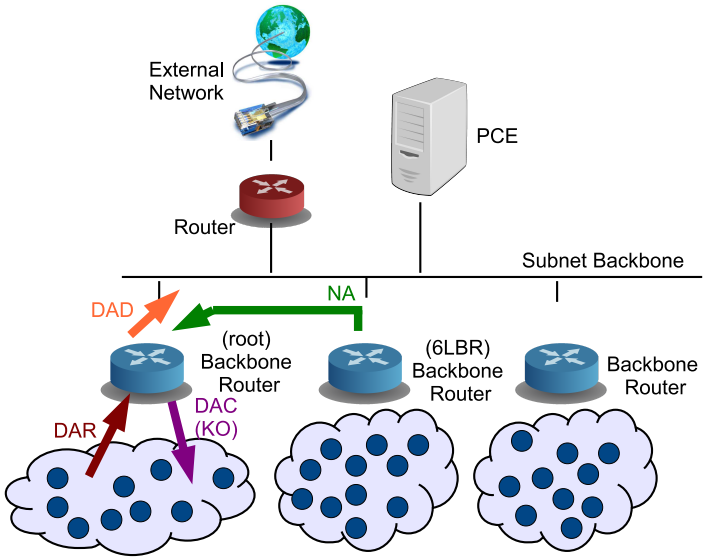
Fig. 8. 6TiSCH forwarding mechanisms.

but impractical for operations over a multilink subnet [49] [50]. Mobile IPv6 (MIPv6) [51] introduced a registration protocol to feed a Binding Table and enable reachability to a Mobile Node (MN) over an IP tunnel. In order to attract packets for a registered MN, a Home Agent (HA) performs IPv6 ND proxy operations over a Home Network where the subnet of the mobile node resides.

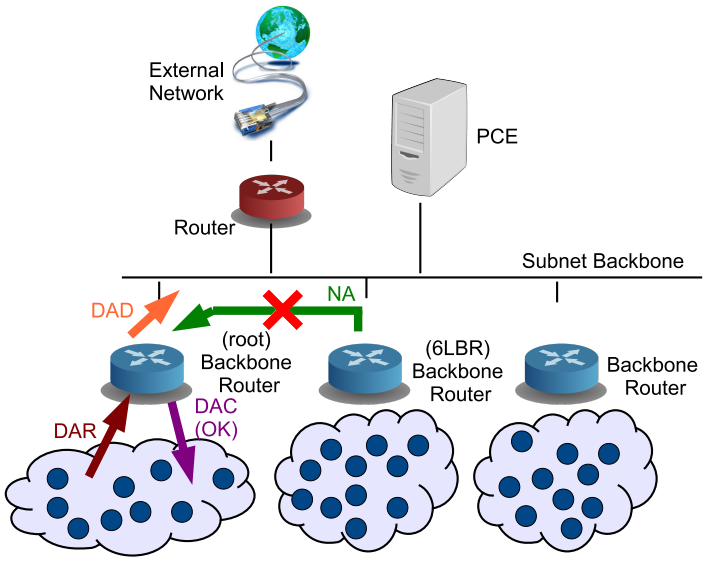
An IPv6 ND registration mechanism was standardized as Neighbor Discovery Optimization for Low-power and Lossy Networks [52], and extended by wireless ND [53]. The ND registration can also be used to feed a Binding Table for low-power devices that are attached directly to the Backbone Router. In detail, at the core of [53] there is the replacement of the multicast flooding, traditionally required for Duplicate Address Detection (DAD), with a unicast registration to a centralized binding table. The new message to create an entry is a `Duplicate Address Request (DAR)` that is answered by a `Duplicate Address Confirmation (DAC)` message. Finally, a new `Address Registration Option (ARO)` is introduced which contains a unique ID for the device, typically an EUI-64 address.

In a fashion similar to a MIPv6 HA, the BBR can proxy the ND protocol over the backbone on behalf of registered node. In particular, it may advertise its own MAC address in order (i) to avoid leaking additional MAC addresses in the backbone, and (ii) to make sure it receives and stores the packets for a sleeping device till the device is reachable to receive them. For the case of a multilink subnet based on RPL meshes that are interconnected by a backbone, the RIB installed by the RPL protocol can be used to feed the Binding Table that will keep track of which IPv6 address this BBR proxies for.

An issue arises when the same address is registered asynchronously on two different BBRs, as it is unclear whether that is a device that just moved, or if it is an address that is duplicated between two devices. Whether the BBR inter-working is done through a routing protocol or classical IPv6 ND, there is a need for an extension to assert this. RPL, and to that regard any traditional



(a) Duplication: different EUI-64 in ARO option.



(b) Mobility: same EUI-64 (newer TID wins).

Fig. 9. Determination of duplication vs. mobility.

routing protocol, will not consider that two advertisements can represent a duplication, but simply that there are probably two ways to get to the same device. On the other hand, the `ARO` in [52] can be used to find out when there is a duplication though a device unique ID as illustrated in Fig. 9(a), but cannot tell which is the current state that must be conserved from the stale one, that should be cleaned up. RPL uses a sequence counter (called `DAOSequence`) to detect stale advertisements, and there is probably a need to enhance the `ARO` to add a similar indication (a Transaction ID, TID) for use within the ND registration mechanism, as illustrated in Fig. 9(b).

Once the duplication problem is sorted out, there is still a need to discover and route over the backbone between a device that is attached to the backbone and a wireless device that is located inside a DODAG and reachable over the BBR. It is possible to extend RPL over the backbone and present the subnet as not-onlink in Router Advertisements, so as to always route, over the backbone and then along the DODAG. The alternate is a mixed mode over the backbone that consists in proxying ND operations. Upon a RPL route advertisement (called a `DAO`), the BBR that acts as root for the DODAG where a given device is located installs a host route towards the device over the LLN. Then, it advertises the device's address over the backbone using classical ND with extensions to check for duplication and movement. In this way, any legacy IPv6 device, using the classical IPv6 ND exchange of a `Neighbor Solicitation (NS)` and its `Neighbor Advertisement (NA)` response, resolves that the MAC address for the device is in fact that of the BBR. Then, it passes on the packet, which the BBR finally routes over the DODAG to the wireless destination. This procedure is illustrated in Fig. 10.

There are a number of questions to be answered in this proxy ND operation. In particular, how does this model work with multiple instances that are eventually rooted at different BBRs, and there are well-known possible answers such as the use of VLANs. Regardless of the answer, there is substantial work to be done to extend the simple model in [52] to operate over a backbone, and then enable routing from the backbone towards a LLN device.

4 Applications

As of today, several commercial low-power wireless networking providers are offering 99.999% reliable MAC layers, for instance [54], that provide radio duty cycles well below 1%, thereby reducing the mote power consumption and increasing the network lifetime [55]. This is facilitating the introduction of new monitoring and actuating devices as tools for operational technologies to improve the security, process automation, efficiency and productivity of the Industries [56, 57], and devices, a clear roadmap to the Industrial Internet paradigm [58].

The 6TiSCH working group is contributing to the consolidation of an open standards based protocol stack which will empower global adaptation of

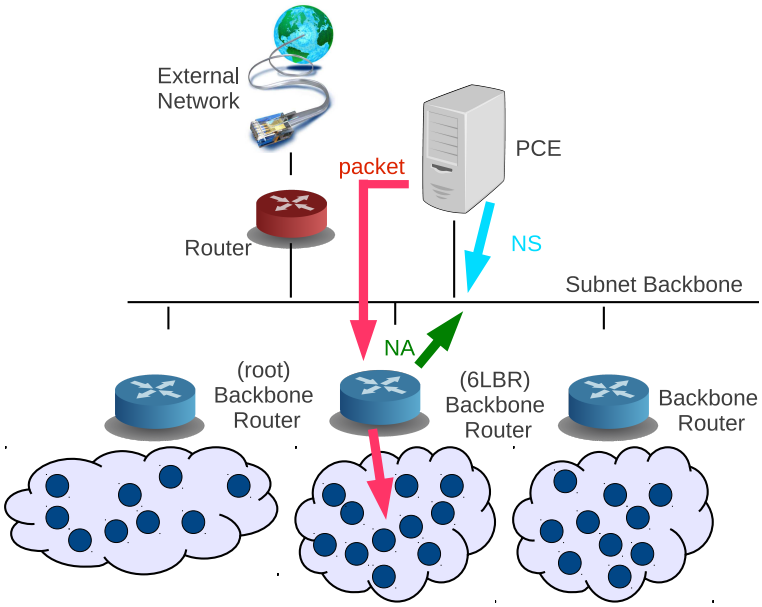


Fig. 10. ND Resolution.

low power wireless technologies and will align Informational Technologies to Operational Technologies enabling the real take off of the Industrial Internet era. Hereafter we present and describe some of the potential applications that will benefit from the 6TiSCH architecture introduced in this chapter. Note that this is a non-exhaustive list.

- **Internet of Factory** WirelessHART and ISA100.11a are two industrial wireless standards, widely used in industrial process automation. Since 6TiSCH architecture brings the deterministic feature of the two aforementioned standards into IPv6 context, it allows to employ those technologies and functionalities provided by Internet community such as *traffic engineering*. The 6TiSCH architecture allows more dynamics in industrial wireless networks while keeping the deterministic feature; in addition, it allows the industrial wireless networks to become part of the Internet automatically, enabling applications like *remote control*.
- **Smart cities** Smart city involves many aspects such as smart economy, smart people, smart mobility, smart environment, smart living and smart governance. But, all of them need a common foundation, i.e., IoT infrastructure, which provides the connectivity among physical world, cyber world and people. As part of the IoT infrastructure, wireless sensor/actuator networks play a critical role, e.g., sensing occupancy of parking spots, sensing air quality, sensing traffic condition, alarming accidence, and control street light. These applications can be built on top of a IoT

infrastructure based on open standards. By nature, the 6TiSCH architecture and protocol stack is a good solution for the IoT infrastructure in a smart city.

- **IoT Service Provider** Service Providers (SPs) have to deal with the different requests of their customers, and make sure that all of them are satisfied at the same time (whenever possible). The 6TiSCH architecture is very promising for service provider operating an *umbrella* network supporting different types of traffic flow, coming from several distinct customers. In fact, when a new customer asks to be granted access to the network, the network operator can predict with pretty good granularity the impact this new traffic will have on the remaining bandwidth of the network, and on the power consumption of individual motes. Thus, the operator will be able to grant/deny, and probably also charge differently each customer, according to the actual guaranteed service.
- **Internet of Buildings** The core functionality of building automation systems keeps the building climate within a specified range, provides lighting based on an occupancy schedule, monitors system performance and device failures, and provides malfunction alarms to building engineering/maintenance staff. The functionality reduces building energy and maintenance costs. By applying the 6TiSCH architecture and protocol stack, all of the sensors and actuators in the building automation system can be connected to Internet with very low energy consumption. Besides enabling more efficient and lower cost remote building controls, it will potentially let more information from the Internet input the building automation system, and make the buildings smarter.
- **Internet of Vehicles** Vehicular Automation is a system to assist vehicle operator. Manufacturers and researchers subsequently added a variety of automated functions to cars and other vehicles, and are now exploring how to take advantage of wireless networking technologies to integrate the vehicles into the Internet to improve driving safety, convenience and efficiency. There are two potential application fields, one is vehicle internal communication, another is vehicle-to-vehicle or vehicle-to-roadside communication. The functional requirement from the two fields are very different, but they require some features in common, e.g., both of them work in complex radio environment, while requiring highly reliable and deterministic network feature. Thus, the 6TiSCH architecture and protocol stack could be a candidate solution.
- **Internet of Home** Home automation is the residential extension of building automation. It is automation of the home, housework or household activity. Conventional home automation may include centralized control of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of doors, and other systems, to provide improved convenience, comfort, energy efficiency and security. In these home automation systems, different kinds of sensors and actuators play a critical role, and lower power operation is a welcome feature. Like other wireless networks,

adopting 6TiSCH can make home automation systems more flexible and easier to be installed. Besides, more importantly, while the sensors and actuators are equipped with the 6TiSCH protocol stack, the conventional home automation system can be extended to a remote monitor and control loop with very low additional energy consumption, which involves the human beings related with the home but being far away. In addition, since the 6TiSCH protocol stack can bridge the sensors in home and monitors in hospitals or some care centers, the home automation for the elderly and disabled can provide more efficient and more economical services from caregivers or institutional care.

5 Conclusion

In the last 40 years we have witnessed first, the emergence of the Operational Technology (OT) and the Information Technology (IT) in parallel, each of them established with its own scope and range of applications; then, the progressive convergence of IT over an IP infrastructure, imprinted by the birth of the Information and Communications Technology (ICT). Nowadays, we are witnessing the unstoppable evolution of the Internet of Things (IoT), and the upcoming integration of OT and IT. The technologies facilitating such OT/IT convergence only recently commenced to take shape.

In detail, existing industrial Wireless Sensor Network technologies have demonstrated that the IEEE802.15.4e *Timeslotted Channel Hopping* (TSCH) effectively enables industrial-grade deterministic properties for slow speed control loops with low latency, ultra-low jitter and a high reliability. It makes sense to extend this support to a distributed mode that can be cheaper, at the expense of the optimization that only a centralized approach can obtain.

To this aim, this chapter has hence introduced the work recently started at IETF by the 6TiSCH WG, and outlined the challenges that it will have to face when adopting cross IEEE and IETF standards within the IoT Industrial protocol stack for deterministic networks.

We have focused in particular on the 6TiSCH technically viable communication architecture, based on open standards, which will enable a new range of applications in automation (home, city, building), man-to-machine interfaces (cars, planes), and machine-to-machine communication.

The 6TiSCH WG is currently putting effort on developing distributed routing operation over a static TSCH schedule. At the same time, a *Minimal 6TiSCH Configuration* defining how to build a 6TiSCH networks using the Routing Protocol for LLNs (RPL) and a static TSCH schedule, is being produced. Finally, the 6TiSCH WG will also produce an *Information Model* containing the management requirements of a 6TiSCH node. Such model will include a description of the mechanisms to be used by an entity to (i) manage the TSCH schedule in a 6TiSCH node, and (ii) query timeslot information

from that node. A *Data Model* mapping for an existing protocol, such as Concise Binary Object Representation (CBOR) over the Constrained Application Protocol (CoAP), will be also provided.

Acknowledgment

This publication was supported by the FP7 projects IoT6-288445, CALIPSO-288879, RELYONIT-317826 and SWAP-251557, which are partially funded by the European Community. Xavier Vilajosana is funded by the Spanish Ministry of Education under Fullbright-BE grant (INF-2010-0319).

References

1. Postel J (1981) Internet Protocol, RFC 791, Internet Engineering Task Force
2. Andreasen F, Foster B (2003), Media Gateway Control Protocol (MGCP) Version 1.0, RFC3435, Internet Engineering Task Force
3. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M, Schooler E (2002), SIP: Session Initiation Protocol, RFC 3261, Internet Engineering Task Force
4. Schulzrinne H, Casner S, Frederick R, Jacobson V (2003) RTP: A Transport Protocol for Real-Time Applications, RFC3550, Internet Engineering Task Force
5. Deering S, Hinden R (1998) Internet Protocol, Version 6 (IPv6) Specification, RFC2460, Internet Engineering Task Force
6. Baker F, Li X, Bao C, Yin K (2011) Framework for IPv4/IPv6 Translation, RFC6144, Internet Engineering Task Force
7. Conta A, Deering S, Gupta M (2006) Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version6 (IPv6) Specification, RFC4443, Internet Engineering Task Force
8. Narten T, Nordmark E, Simpson W, Soliman H (2007) Neighbor Discovery for IP version 6 (IPv6), RFC4861, Internet Engineering Task Force
9. Winter T, Thubert P, Brandt A, Hui J, Kelsey R, Levis P, Pister K, Struik R, Vasseur J P, Alexander R (2012) RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550, Internet Engineering Task Force
10. Droms R, Bound J, Volz B, Lemon T, Perkins C, Carney M (2003), Dynamic Host Configuration Protocol for IPv6 (DHCPv6), RFC3315, Internet Engineering Task Force
11. Vida R, Costa L (2004) Multicast Listener Discovery Version 2 (MLDv2) for IPv6, RFC3810, Internet Engineering Task Force
12. Postel J (1980) User Datagram Protocol, RFC768, Internet Engineering Task Force
13. Shelby Z, Hartke K, Bormann C, Frank B (2011) Constrained Application Protocol (CoAP), IETF CoRE Working Group
14. Palattella M R, Accettura N, Vilajosana X, Watteyne T, Grieco L A, Boggia G, and Dohler M (2012) Standardized Protocol Stack For The Internet Of (Important) Things, IEEE Communications Surveys and Tutorials, vol.15, no.3, pp. 1389 - 1406

15. 6TiSCH Mailing list available at: https://www.ietf.org/mailman/listinfo/6tsch
16. 6TiSCH homepage available at: https://bitbucket.org/6tsch/
17. IEEE802.15.4e (2012) IEEE Standard for Local and Metropolitan Area Networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer, Institute of Electrical and Electronics Engineers Std., April
18. Thubert P, Watteyne T, Palattella M R, Vilajosana X, Wang Q (2013) IETF 6TSCH: Combining IPv6 Connectivity with Industrial Performance, in Proc. of Int. Workshop on Extending Seamlessly to the Internet of Things (esIoT), Taiwan, July
19. Watteyne T, Palattella M R, Grieco L A (2013) Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals. draft-watteyne-6tsch-tsch-lln-context-01 (work in progress), February
20. IEEE802.15.4 (2011) IEEE Standard for Local and metropolitan area networks – Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs), Standard for Information Technology Std., September
21. Kushalnagar N, Montenegro G, Schumacher C (2007) IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, RFC 4919, Internet Engineering Task Force
22. Pister K and Doherty L (2008) TSMP: Time Synchronized Mesh Prootocol, Proc. of Int. Symp. Distributed Sensor Networks (DSN), Florida, USA
23. Highway Addressable Remote Transducer, a group of specifications for industrial process and control devices administered by the HART Foundation. Available at www.hartcomm.org.
24. ISA, ISA100, Wireless Systems for Automation, May 2008. Available at http://www.isa.org/Community/ SP100WirelessSystemsforAutomation.
25. M. Nixon, A Comparison of WirelessHART and ISA100.11a (2012), July, white paper.
26. Stanislowski D, Vilajosana X, Wang Q, Watteyne T, Pister K (2013) Adaptive Synchronization in IEEE802.15.4e Networks, Industrial Informatics, IEEE Transactions on , vol.PP, no.99, pp.1
27. Palattella M R, Thubert P, Watteyne T, Wang Q (2013) Terminology in IPv6 over Time Slotted Channel Hopping. draft-palattella-6tsch-terminology-00 (work in progress), March
28. Palattella M R, Accettura N, Grieco L A, Boggia G, Dohler M, Engel T (2013) On Optimal Scheduling in Duty-Cycled IoT Industrial Applications using IEEE 802.15.4e TSCH, IEEE Sensors Journal, vol.13, no.10, pp.3655 - 3666
29. Watteyne T,Vilajosana X, Kerkez B, Chraim F, Weekly K, Wang Q, Glaser S, Pister K (2012) OpenWSN: A Standards-Based Low-Power Wireless Development Environment, Transactions on Emerging Telecommunications Technologies, vol.23, no.5, pp. 480 493
30. uRes, available at https://openwsn.atlassian.net/wiki/display/OW/uRES
31. Braden R, Zhang L, Berson S, Herzog S, Jamin S (1997) Resource ReSerVation Protocol (RSVP) : Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force
32. Morell A, Vilajosana X, Vicario J L, Watteyne T (2013) Label Switching over IEEE802.15.4e Networks. Transactions on Emerging Telecommunications Technologies, vol.24, no.5, pp.458 475

33. Thubert P (2013), 6LoWPAN Backbone Router. draft-thubert-6lowpan-backbone-router-03 (work in progress), February
34. Thubert P, Assimiti R A, Watteyne T (2013) An Architecture for IPv6 over Time Synchronized Channel Hopping. draft-thubert-6tsch-architecture-00 (work in progress), March
35. Vasseur J P, Le Roux J L (2009) Path Computation Element (PCE) Communication Protocol (PCEP) RFC 5440, Internet Engineering Task Force
36. Hancock R, Karagiannis G, Loughney J, Van den Bosch S (2005) Next Steps in Signaling (NSIS): Framework. RFC 4080, Internet Engineering Task Force
37. Forsberg D, Ohba Y, Patil B, Tschofenig H, Yegin A (2008) Protocol for Carrying Authentication for Network Access (PANA), RFC 5191, Internet Engineering Task Force
38. Duffy P, Chakrabarti S, Cragie R, Ohba Y, Yegin A (2011) Protocol for Carrying Authentication for Network Access (PANA) Relay Element, RFC 6345, Internet Engineering Task Force
39. Yegin A, Cragie R (2012) Encrypting the Protocol for Carrying Authentication for Network Access (PANA) Attribute-Value Pairs, RFC 6786, Internet Engineering Task Force
40. Wang Q, Vilajosana X, Watteyne T (2013) 6tus Adaptation Layer Specification. draft-wang-6tsch-6tus-00 (work in progress), March
41. The OpenFlow Switch Specification. Available at http://OpenFlowSwitch.org.
42. Doria A, Hadi Salim J, Haas R, Khosravi H, Wang W, Dong L, Gopal R, Halpern J (2010) Forwarding and Control Element Separation (ForCES) Protocol Specification RFC 5810, Internet Engineering Task Force
43. Vilajosana X, Pister K (2013) Minimal 6TSCH Configuration, draft-vilajosana-6tsch-basic-01 (work in progress), July
44. Thubert P (2012) Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL), RFC 6552, Internet Engineering Task Force
45. IETF Working Group. DTLS In Constrained Environments (DICE), - charter at http://datatracker.ietf.org/wg/dice/charter/
46. Mannie E (2004) Generalized Multi-Protocol Label Switching (GMPLS) Architecture, RFC3945, Internet Engineering Task Force
47. Rosen E, Viswanathan A, Callon R (2001) Multiprotocol Label Switching Architecture, RFC3031, Internet Engineering Task Force
48. Thubert P, Hui J W (2013) LLN Fragment Forwarding and Recovery, draft-thubert-roll-forwarding-frags-02 (work in progress), September
49. Thaler D, Huitema C (2002) Multi-link Subnet Support in IPv6, draft-ietf-ipv6-multilink-subnets-00.txt, Internet Draft, Internet Engineering Task Force
50. Thaler D (2007) Multi-Link Subnet Issues, RFC4903, Internet Engineering Task Force
51. Perkins C, Johnson D, Arkko J (2011) Mobility Support in IPv6, RFC 6275, Internet Engineering Task Force
52. Shelby Z, Chakrabarti S, Nordmark E, Bormann C (2012) Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), RFC 6775, Internet Engineering Task Force
53. Chakrabarti S, Nordmark E, Wasserman M (2012) Efficiency aware IPv6 Neighbor Discovery Optimization draft-chakrabarti-nordmark-6man-efficient-nd-01 (work in progress), November
54. Doherty L, Lindsay W, Simon J (2007) Channel-specific wireless sensor network path data, In Proc. of IEEE ICCN 2007 Conf., pages 89 – 94

55. Dust Networks Linear Technology (2013) Smart mesh ip
56. Vilajosana I, Llosa J, Martinez M, Pacho J C (2012) Wireless sensors helps monitoring one of world most advanced load and unload harbor terminals, White Paper available at www.loadsensing.com
57. Emerson (2013), Emerson wireless technology helps RWE maximize gas storage capacity and improve efficiency and safety, White Paper available at www.emersonpress.com.
58. Evans P C and Annunziata M (2012) Industrial internet: Pushing the boundaries of minds and machines, White Paper available at www.ge.com