Wireless Personal Communications
https://doi.org/10.1007/s11277-023-10213-w

Multi-Modal Industrial IoT Networks: Recent Advances and Future Challenges

Robbe Elsas[1] · **Dries Van Leemput**[1] · **Jeroen Hoebeke**[1] · **Eli De Poorter**[1]

Accepted: 15 February 2023

Abstract
While the ongoing fourth industrial revolution continues to be a major driver behind wireless communication technologies, some environments are so prohibitive that even state-of-the-art solutions can barely achieve ubiquitous wireless connectivity (if at all). For example, in industrial sites with large metal constructions (such as petrochemical plants), highly localized and time-varying changes in wireless link quality are quite common. Oddly enough, much of the capabilities needed to deal with such effects are already present at the physical layer (PHY), but remain largely unexploited by higher protocol layers. In fact, little Industrial Internet of Things (IoT) (IIoT) research has considered harnessing the full multi-modal capabilities of modern multi-PHY/multi-band IoT hardware in general. As such, in this vision paper, we: (1) analyze recent advances towards enabling multi-modal IIoT through link- and routing layer operations; and (2) describe challenges and opportunities for future IIoT deployments, based on the design choices that emerged from said analysis. In summary, we identify a combination of a modified/extended Time-Slotted Channel Hopping (TSCH) link layer, using either fixed or variable duration timeslots, together with a Parent-Oriented (PO) Routing Protocol for Low-Power and Lossy Networks (LLNs) (RPL) approach to be the most promising way forward.

Keywords 6TiSCH · IEEE 802.15.4 · IIoT · Multi-modal · RPL · TSCH · WSN

✉ Robbe Elsas
robbe.elsas@ugent.be

Dries Van Leemput
dries.vanleemput@ugent.be

Jeroen Hoebeke
jeroen.hoebeke@ugent.be

Eli De Poorter
eli.depoorter@ugent.be

[1] IDLab, Department of Information Technology, Ghent University–imec, Technologiepark–Zwijnaarde, Ghent 9052, Belgium

Published online: 22 March 2023

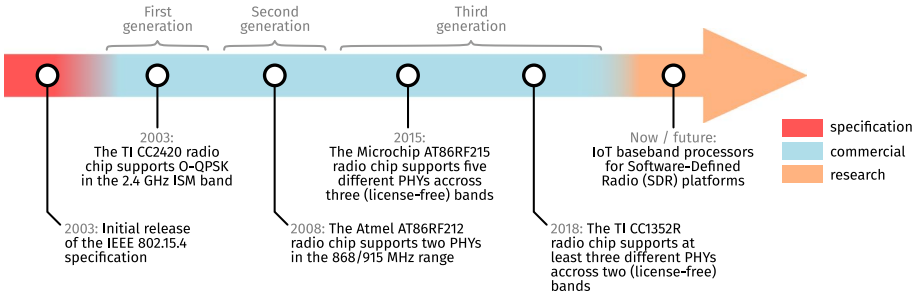

Fig. 1 The trend towards flexibility in Industrial Internet of Things (IoT) (IIoT): IEEE 802.15.4 [1] radios support an increasing number of Modulation and Coding Schemes (MCSs) and frequency ranges. This trend is expected to continue, culminating in fully flexible Software Defined Radio (SDR)-based IoT platforms that are no longer specific to IEEE 802.15.4. The commercial radios pictured in this timeline are merely popular examples

1 Introduction

The hardware used in Wireless Sensor Networks (WSNs) for Industrial Internet of Things (IIoT) applications has become increasingly configurable in recent years. Consider IEEE 802.15.4 [1] radios, for example. Figure 1 constitutes a timeline of important occurrences in terms of IEEE 802.15.4 radio flexibility. The first generation of these radios supported only one frequency band and Modulation and Coding Scheme (MCS). Next, in the second generation, multi-band radio chips became commercially available. More recently, chip manufacturers have introduced multi-band radios with support for multiple MCSs, allowing for even more flexibility whilst lowering cost/footprint. Literature suggests this trend towards increased radio flexibility will continue in the foreseeable future, as we are now witnessing the first scientific publications presenting Software-Defined Radio (SDR) platforms implementing low-power IoT-specific baseband processors [2–4].

Despite often having multiple communication modes (i.e., multiple interfaces, multiple MCSs, or a combination thereof) available to them, most IoT devices are limited to using one mode for their entire operational lifetime. Rather than choosing a fixed communication mode per use case with fixed range/energy consumption/throughput characteristics, individual links could be configured according to local operating conditions. This allows for network optimization in all areas of a single deployment (e.g., by switching to less robust modulations or higher frequencies to achieve required data rates), or to provide coverage in challenging isolated industrial sites [5] with large metallic obstacles through multi-hop mesh networks using the most appropriate MCS for each hop (or by switching between radios).

When we say that a link can be "configured" we mean that a device may opt (or otherwise be instructed) to switch to a specific communication mode when sending/exchanging traffic to/with a directly-connected device in its vicinity. Switching to a different communication mode may mean switching to a different MCS on a given interface, switching to a different interface altogether, and so on. By extension, we say an IoT network is multi-modal if it at least allows devices to switch between communication modes at run-time and on a per-link basis. In order to achieve truly adaptive/dynamically configured links in IIoT networks, the network must be multi-modal.

In this vision paper, we go over the technical considerations, state-of-the-art, and future research directives related to exploiting the multi-modal capabilities of modern IoT devices, that is at the protocol layers directly above the physical (PHY) layer. The remainder of this paper is structured as follows. First, in Sect. 2, we briefly discuss the history of IEEE 802.15.4 [1] as the basis of IIoT-specific protocol stacks. Next, we discuss solutions towards and challenges of enabling multi-modal communication at the Medium Access Control (MAC) layer (see Sect. 3) and the routing layer (see Sect. 4). Finally, we conclude this paper with a summary of our vision for the future of multi-modal IIoT (see Sect. 5).

2 IEEE 802.15.4: A Multi-Layer Standard for Industrial IoT

Many would argue that IEEE 802.15.4 [1] is *the* most widespread PHY and MAC protocol layer standard for industrial IoT. Besides it being a multi-layer standard, IEEE 802.15.4 now describes 20 PHY specifications (some of which are multi-rate) and at least four MAC modes. Whilst this enables lots of fine-tuning, it is not helpful for interoperability (nor consumer understanding), since most commercially available transceivers support only a subset of those PHYs and embedded Operating Systems (OSs) for WSNs/IoT typically support two of its MAC modes at best.

In the past (i.e., pre- 2011/2012 [6, 7]), when IoT researchers talked about IEEE 802.15.4, they typically meant a combination of the Offset Quadrature Phase-Shift Keying (O-QPSK) [8, Section 7.3.2] or Binary Phase-Shift Keying (BPSK) [8, Section 7.2.3] PHY [1, pp. 467–477] and the Beacon-Enabled (BE) or Non-Beacon-Enabled (NBE) MAC mode respectively. From now on we shall group these together as "legacy" implementations. The fact that well-known protocol stacks such as Zigbee [9] and Thread [10] are based on legacy IEEE 802.15.4 is a major source of confusion. In particular, due to Zigbee's (past) popularity, non-experts started incorrectly using the wording IEEE 802.15.4 and Zigbee interchangeably.

Although they had previously defined a way of sending Internet Protocol (IP) version 6 (IPv6) traffic over IEEE 802.15.4 links through the IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) specification [11], the Internet Engineering Task-Force (IETF) really got involved with IEEE 802.15.4-based networks in 2011/2012, with an improved 6LoWPAN header compression format [12] and (more importantly) the Routing Protocol for Low-Power and Lossy Networks (LLNs) (RPL) [13], which first[1] enabled true end-to-end IP connectivity and allowed for integration of IEEE 802.15.4-based networks into (existing) IP infrastructure without the need for proprietary high-level "translation units" at the network edge. From then on, any combination of an IEEE 802.15.4 PHY and MAC with 6LoWPAN and RPL (and more traditional IP-related technologies defined by IETF) became known to some as the IETF protocol stack (Fig. 2).

Later on, the IETF stack diverged even farther from Zigbee with the IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) MAC mode. Since its inception in 2012 [7] and adoption in 2016 [14], TSCH has gained traction in various IoT "verticals". Originally, it was designed to answer concerns that were (at the time) specific to industrial use cases by adopting (medium-access) principles from WirelessHART [15] and ISA 100.11a [16]. Nonetheless, as the (unlicensed) frequency spectrum became more crowded, TSCH also offered an improvement (over legacy MAC modes) for non-industrial use-cases, mainly in

[1] Thread also does this but the Thread Group alliance was only formed in 2014.

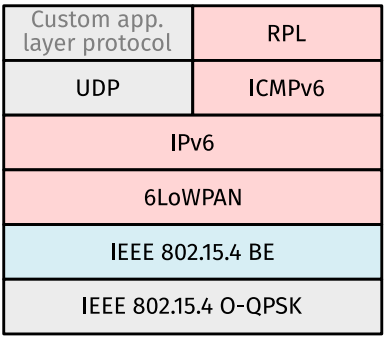

Fig. 2 Example of an early implementation of the Internet Engineering Task Force (IETF) protocol stack. Note that network-layer functionality is indicated in red, whilst link-layer functionality is shown in blue

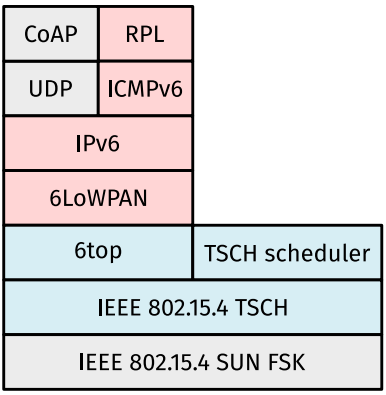

Fig. 3 Example of an IETF IPv6 over the TSCH mode of IEEE 802.15.4 (6TiSCH) protocol stack implementation. Note that network-layer functionality is indicated in red, whilst link-layer functionality is shown in blue. (Color figure online)

terms of interference robustness. The IETF noticed this rise in popularity and started the IPv6 over the TSCH mode of IEEE 802.15.4 (6TiSCH) Work Group (WG) with the added goal of consolidating the IETF stack, which has relatively recently resulted in several 6TiSCH specification documents [17, 18]. Figure 3 gives an example of what a 6TiSCH protocol stack implementation typically looks like.

3 The Link Layer: Multi-Modal Solutions and Challenges

In this section, we focus on TSCH, i.e., the default MAC protocol for most new IIoT deployments that implement the IETF protocol stack. With TSCH, nodes synchronize to a periodic slotframe consisting of timeslots [19]. Each timeslot allows a pair of devices to exchange a data frame and corresponding acknowledgement. The number of timeslots passed since the start of the network is known as the Absolute Slot Number (ASN). Two big advantages of time-slotted access are: (1) it generally increases throughput by eliminating collisions amongst nodes competing for medium access; and (2) it enables deterministic latency. Moreover, TSCH employs multi-channel communication through channel hopping. More specifically, initially, several predefined frequency channels are available. Of those, the eligible channels are contained in the "hopping sequence" array. Given a TSCH link is defined as "the pairwise assignment of a directed communication between devices

... in a given timeslot on a given [channel offset]" [1, p. 70], the channel on which to communicate can be calculated with (1).

$$CH = HS[(ASN + CO)\%|HS|] \tag{1}$$

where CH is the frequency channel to communicate on; HS is a finite ordered list ($\neq$ set; duplicates are allowed) of frequency channels to communicate on, i.e., the "hopping sequence"; ASN is the absolute slot number of the current timeslot; CO is the current channel offset, as determined by the TSCH scheduler being used (e.g., Orchestra [20]); and $|HS|$ is the length of the hopping sequence.

Distinct node pairs can thus *communicate in the same timeslot, if they use a different channel offset*. Such multi-channel communication increases network capacity and (partially) mitigates, e.g., interference[2] and (potentially) multi-path fading, that is if channels are spaced apart sufficiently.

A major part of TSCH is the "link schedule", i.e., the assignment of TSCH links to node-pairs. Determining an optimal schedule depends heavily on the given use-case. Hence, *IEEE 802.15.4 deliberately leaves the scheduling algorithm at the discretion of the implementer*.

There are two ways of looking at a schedule. That is, it can be viewed as a one-dimensional slotframe made up of timeslots ("slots", for short). Each slot is uniquely identified by its slot offset in the slotframe, and channel offsets are applied after the fact. After all, as it is generally assumed a node can't simultaneously transmit and/or receive on a different channel, most schedulers don't allow a given node to schedule multiple timeslots with the same slot offset anyway, regardless of channel offset. Alternatively, a schedule can be viewed as a two-dimensional slotframe made up of cells, each uniquely identified by a slot *and* channel offset in the slotframe. Referring to a slot instead of a cell when channel offset is of importance is bad practice. In addition, we can't overstate the difference between an actual schedule and its result in time-frequency space. Most importantly, the same channel offset maps to a different channel for consecutive slot offsets. This distinction is less critical for the time component, as within a slotframe, the slot offset and ASN only differ by a constant, i.e., the ASN in the first slot. Figure 4 illustrates the difference between a one- and two-dimensional slotframe, as well as its result in time-frequency space, using a very simple TSCH network as an example.

Many TSCH schedulers have been proposed in literature. Yet, to our knowledge, *most TSCH scheduling algorithms assume that all nodes in a given network use one statically configured communication mode throughout the network, thereby ignoring the capabilities of current/future IoT devices.*

Ensuring neighboring devices know when/how to communicate is the responsibility of the MAC protocol. As such, supporting multi-modal communication implies the MAC protocol enables link endpoints[3] to know the proper configuration for a given link, be it passively (by simultaneously listening to all communication modes, which is not always possible) or actively (by either explicitly negotiating link configurations or implicitly requiring a mode-switch at predetermined points in time).

For the remainder of this section, we shall limit the discussion to active link configuration, as passive approaches are rather impractical when they're not coupled with a multi-modal routing solution, which is a discussion reserved for Sect. 4.

[2] That is, interference originating from devices within the network itself.

[3] That is, the transmitting and receiving device in a direct exchange over the wireless medium.

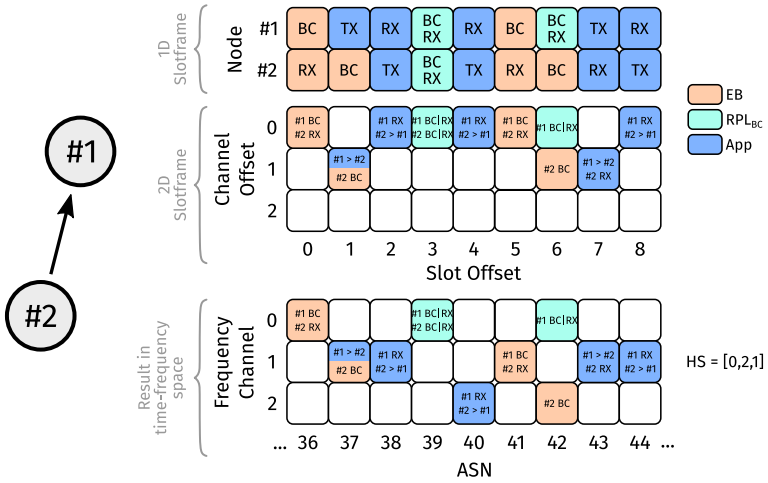

Fig. 4 Example of a very simple Time-Slotted Channel Hopping (TSCH) network, including a visual representation of a possible schedule in the form of both a one- and two-dimensional slotframe, as well as the result of schedule in time-frequency space. Note that this schedule was created following (a configuration of) the rules of Orchestra [20]

3.1 Recent Advances in Multi-Modal TSCH

Most TSCH-related multi-modal research focuses on using multiple PHYs[4] in one slotframe, which immediately raises the question: should a fixed timeslot duration be used, or should it be varied according to the PHY? Variable duration leads to more efficient schedules and thus higher throughput and/or lower end-to-end latency compared to fixed slot duration. However, channel hopping (1) requires that: (1) the ASN is defined with respect to the same time boundaries across the entire TSCH network ($\pm$ drift), a requirement inherently satisfied by equal duration timeslots; and (2) a transmission (+ acknowledgement, if applicable) does not exceed those boundaries. With variable duration timeslots, the only way (we know of) to satisfy the first requirement is to choose the slot duration such that consecutive timeslots may be grouped together into a "virtual timeslot" if you will, that is if needed to accommodate for slower PHYs. Unfortunately, this breaks the second requirement, which could presumably be circumvented if the scheduler is aware of timeslot durations. However, this would prevent one from using most (if not all) existing schedulers. In contrast, with fixed timeslot duration you simply set the slot duration according to the slowest PHY you wish to support, no custom scheduler needed.[5] Figure 5 uses an abstracted representation of a one-dimensional slotframe to illustrate what it means to use either fixed or variable duration timeslots.

The choice between fixed or variable duration timeslots was also recognized by Brachmann et al. [21], who saw multi-PHY TSCH slotframes as a way to, e.g., deal with the often dissimilar performance requirements imposed by different applications, improve network reliability in the presence of interference, and so on. To this extent, the authors proposed two distinct design approaches using either fixed or variable slot duration respectively. For

[4] It would technically be more accurate to say multiple MCSs instead of multiple PHYs.

[5] You still need a mechanism to signal the PHY to be used though.

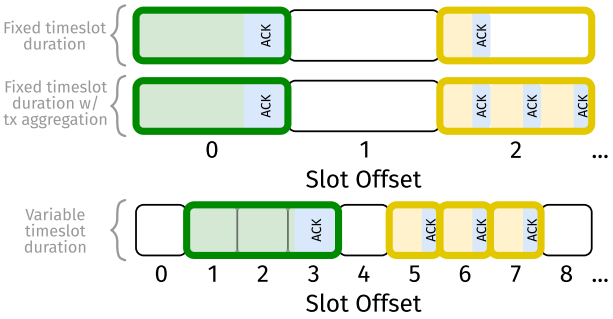

Fig. 5 Abstracted representation of one-dimensional multi-PHY TSCH slotframes using either fixed or variable slot duration. Each continuous coloured border represents a single (virtual) timeslot and each PHY (which has a certain data-rate) is depicted in a dedicated colour. Note that there is more than one way to aggregate multiple transmissions into a fixed duration timeslot

the sake of clarity, we shall hereafter call variable duration timeslots "superslots" (in a one-dimensional slotframe) and designate their constituent (unit duration) slots as "subslots". In a two-dimensional slotframe this would translate to "supercells" and "subcells" respectively.

Note that the authors' implementation is not strictly multi-modal. That is, in their variable slot duration proof-of-concept (fixed duration was only used to characterize the medium), only TSCH beacons were sent using a low-rate PHY, whilst all data traffic used the same high-rate PHY. Put differently, their implementation *mandates* devices to switch between communication modes (= PHYs, in this case) at run-time on a *per-traffic* basis, as opposed to *allowing* said devices to switch PHYs on a *per-link* basis. This is seemingly a minor semantic difference. However, as we understand it from [21], only switching PHYs for TSCH beacons required zero negotiation because all beacons occurred in a known continuous range of (super)slots, followed by a known continuous range of (super) slots reserved for data traffic. This then makes channel hopping significantly easier because within those known ranges, all superslots remain of equal duration. As such, in a way, the authors' proof-of-concept was more of a hybrid between fixed and truly variable slot duration, but it was an important milestone nonetheless.

Using either one or multiple PHYs (or more generally, communication modes) for TSCH beacons is a second important design choice, since Enhanced Beacons (EBs) [1, Figure 7-6] allow a newly-booted node to discover neighboring[6] devices and ultimately synchronize to (and thereby join) a TSCH network. For example, with single-mode discovery, provided favourable conditions and using the most robust communication mode, it may take as much or less time and energy (on average) compared to multi-mode discovery for a new node to synchronize, that is depending on how nodes scan for beacons when using multi-mode discovery.[7] In addition, with single-mode discovery, already synchronized nodes need not allocate as much cells to broadcast beacons, which means less energy is consumed doing so, and more slots/cells remain available for other traffic. Similarly, it takes considerably less resources to receive beacons once synchronized. However, the

[6] While the concept of neighbors often aligns for TSCH, IP, and RPL, this is not always the case.

[7] For example, nodes may scan using a fixed mode of their own choice, they might cycle through different scanning modes in a certain predetermined order, and so on.

chosen (supposedly most robust) communication mode may become unreliable to the point where it takes more time and energy to scan for beacons and (stay) synchronize(d) than it would for a multi-mode alternative (again, depending on how devices scan for beacons). Moreover, the link layer must likely track a metric (see Sect. 4.1) for every communication mode towards each connected device anyway. For some metrics, such as Link Quality Level (LQL), receiving a packet provides sufficient information. As such, multi-mode discovery inherently facilitates metric tracking through receiving TSCH beacons alone. That said, "receive-only" metrics are often less accurate and they certainly don't verify two-way reachability. As such, their popularity is waning. Note that these considerations concerning discovery apply mainly to homogeneous networks, i.e., networks wherein all devices posses the same set of communication modes. Heterogeneous networks require some more thought.

The choice between single- or multi-mode discovery also fits within a larger discussion, that is whether or not to use multiple communication modes for all management traffic (which covers more than just beacons). Most (if not all) existing solutions use a single mode for at least part of management traffic. Said communication mode is typically picked based on extensive in-situ experiments or simulations. However, when a network's existence/continuity depends on a single communication mode, the benefits of a multi-modal IIoT network are effectively limited to improving the performance of networks deployed in environments where ubiquitous connectivity is already achievable with single-mode technologies. While this has its merits, it foregoes one of the main value propositions of multi-modal IIoT, namely the much improved chance of providing coverage in all areas of a single deployment without prior knowledge and with a high likelihood of (highly localized) time-varying conditions.

Van Leemput et al. [22] showed that a fixed slot duration is not as limited as one would think, especially in terms of throughput. The authors saw the value in being able to use most existing schedulers as-is. Hence, they aggregated multiple transmissions into a single (fixed duration) timeslot if the data rate of a given PHY allowed it. More specifically, the authors proposed two ways of performing such aggregation for unicast links, i.e., by either: (1) explicitly acknowledging each unicast packet individually; or (2) by sending a burst of unicast packets and responding with a single acknowledgement.

Both approaches were implemented using the Orchestra [20] autonomous TSCH scheduler. With Orchestra, all traffic is divided into three "planes". Conceptually speaking, a dedicated TSCH slotframe is associated with each plane, resulting in one slotframe for TSCH beacons, one for (RPL) broadcast traffic, and one for the remaining unicast traffic. In addition, there are four "slotframe types", i.e., rule-sets for building a conceptual slotframe. To be precise, each of the three slotframes is used for a certain type of traffic whilst also being of a given slotframe type. Which rules to use for which of the three slotframes, as well as their respective length,[8] is common knowledge[9] to all nodes (not yet) in the network, allowing each node to locally compute the three conceptual slotframes and merge them (continuously). Depending on the slotframe type, nodes use some deterministic function to compute a slot and channel offset for every slot it must include in a slotframe of said type. If two slots from a different slotframe land on the same slot offset, the slot from the plane with the highest priority takes precedence during merging. It is presumably

[8] The lengths of the three conceptual slotframes must be mutually prime.

[9] Typically, these rules are hardcoded onto every node.

impossible for a node to transmit and/or receive in the same timeslot, even if on a different channel, and so channel offset is not considered during merging.

In both cases, Van Leemput et al. [22] used the same low-rate PHY for all slots in the beacon and (RPL) broadcast slotframes respectively, while allowing slots in the unicast slotframe to use either that same low-rate PHY or an alternative high-rate PHY. The timeslot duration was set to accommodate a single low-rate transmission, meaning transmission aggregation applied exclusively to the unicast slotframe, unicast traffic being the type of traffic for which it would be most useful anyway (from a throughput point of view).

Again, in both cases, (unicast) links are explicitly reconfigured to use a given PHY in exactly the same manner. More specifically, starting from the low(est)-rate PHY, the receiving endpoint of a link gauges the link quality upon receiving a packet and decides (through a clever rate-limiting mechanism) whether or not to switch to a high(er)-rate PHY starting from the next scheduled timeslot for that same link. The receiver feeds this decision back to the transmitting endpoint by means of some re-purposed bits in the acknowledgement(s) it sends back in response. To prevent a scenario wherein the transmitter and receiver can no longer communicate because an acknowledgement got lost, the transmitter will automatically switch to the next high(er)-rate PHY for a given unicast link if it did not receive an acknowledgement for four consecutive transmissions, that is starting from the next scheduled timeslot for said link. Great care must be taken to properly configure the reconfiguration rate-limiting mechanism, as it is extremely expensive to cycle through PHYs. This simple explicit reconfiguration mechanism is ultra-lightweight and (deceivingly) easy to implement. However, it does beg the question whether it is a good idea to signal a PHY reconfiguration using a PHY you explicitly don't want to use anymore, especially because missing a reconfiguration instruction is so expensive.

Rady et al. [23] proposed a variable slot duration approach based on the 6TiSCH Minimal Scheduling Function (MSF) [24], a decentralized scheduling algorithm which uses the 6TiSCH Operation Sublayer (6top) Protocol (6P) [25] to (amongst other things) negotiate cell allocation. Much like Orchestra [20], MSF divides traffic into three planes, associates a TSCH slotframe with each plane, and merges these slotframes. However, the way in which traffic is divided is different. One slotframe is used for TSCH EBs *and* RPL broadcast traffic, wherein a single cell (typically with slot and channel offset 0) is reserved for a shared link. This cell is called the minimal cell. In a second slotframe, each node autonomously (i.e., without negotiation) installs cells of which the slot *and* channel offset are a hash of the link-layer address of either the source or destination of a link (depending on the cell type). The cells in this slotframe are called autonomous receive and transmit cells respectively. Autonomous cells are used exclusively for 6top transactions, such as the two-way 6P `ADD` request and response exchanges [25, Section 3.3.1] used to negotiate cells in the third and final slotframe. This third slotframe is used for all remaining traffic and its cells are called negotiated cells. When (and which type of) cells are negotiated is subject to many rules (see RFC 9033 [24]).

Rady et al. [23] made several adaptations to MSF. First, in the minimal slotframe, each device has a minimal supercell for each of its communication modes[10]. Normally, MSF recommends using the TSCH Slotframe and Link Informational Element (IE) [1, Figure 7-52] carried in TSCH EBs to reserve the cell with slot and channel offset 0 as the minimal cell for every node. However, with multiple minimal supercells of different "duration", nodes now use this IE to reserve groupings of subcells with consecutive slot offsets, each group

[10] Note that the author's approach is both multi-interface *and* multi-PHY.

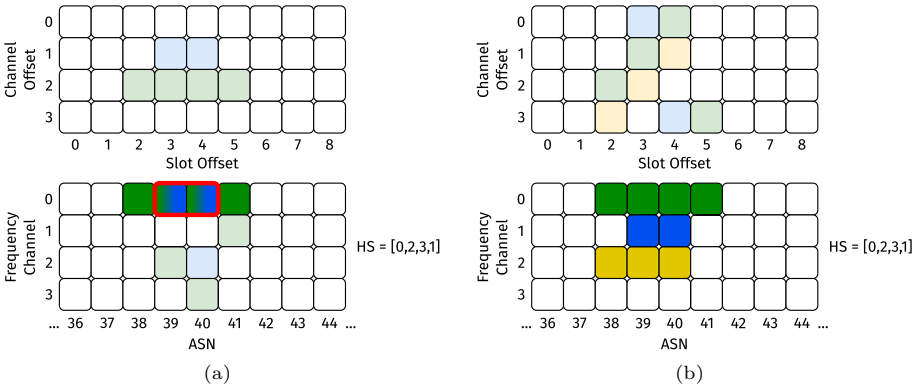

Fig. 6 The pitfall of improper consecutive cell reservations for variable duration TSCH superslots. **a** Reserving cells with consecutive slot offsets but fixed channel offset means the reserved frequency space no longer aligns with the frequency space actually being used from the second subslot onwards. If another node reserves a superslot, and the Absolute Slot Number (ASN) (i.e., the slot offset + a constant value) and channel offset of its first subslot point to the same channel as an already active superslot (according to (1)), a collision occurs, as indicated in red. **b** Reserving cells with consecutive slot offsets and with channel offsets which (in combination) point to the same channel for every subslot avoids such collisions

of subcells together forming the minimal supercell for a given mode. To this extent, the authors modified the IE to also indicate the communication mode to be used in a subcell.

What about channel offsets? From their implementation [26], it is clear that for every subcell of a supercell (be it minimal, autonomous, or negotiated), the authors (erroneously) reserved consecutive slot offsets but the same channel offset. Presumably, the channel to be used in a superslot doesn't change mid-transmission, meaning the channel calculated for its first subslot (with (1)) is used for the entire superslot. However, as the ASN increments for consecutive subslots, the actual channel ($\neq$ channel offset) reserved for those subslots changes according to (1). Thus, if another node reserves a superslot with either of the endpoints of, and some slot offset after the start of, another active superslot, a collision could easily occur as shown in Fig. 6a.

Fortunately, for a given node, MSF (on which [23] is built) does not allow cell reservations (be it for incoming or outgoing links) that overlap in time, as it assumes a single node can't simultaneously transmit and/or receive on separate channels. Hence, the outcome is the same, and no collisions can occur this way. However, it seems only right to properly reserve the time- *and* frequency space you are actually using. To this extent, it suffices to reserve cells with consecutive slot and channel offsets according to (2) and (3). Figure 6b contains an example of proper cell reservations with true consecutive offsets in time and frequency.

$$T_{a,b} = \{n \in \mathbb{N} : 0 \leq a \leq n \leq b < |SO|\} \tag{2}$$

$$CO_{SO} = \begin{cases} \begin{cases} hash(MAC_{dest}) & \text{for Auto-TX cells} \\ hash(MAC_{src}) & \text{for Auto-RX cells} \\ rand(0, |CO| - 1) & \text{for Dedicated cells} \end{cases} & \text{when } SO = a \\ \big(|CO| - 1 + CO_{SO-1}\big)\%|CO| & \forall\, SO \in T_{a+1,b} \end{cases} \tag{3}$$

Table 1 Overview of Time-Slotted Channel Hopping (TSCH)-based active link configuration solutions/approaches for multi-modal operation

References	Fixed or variable slot duration	Single- or multi-mode discovery	Explicit or implicit configuration	6TiSCH compliant[a]
[21]	Fixed	–	–	✓
	Variable	Single	Implicit	✗
[22]	Fixed[b]	Single	Explicit	±[c]
[23]	Variable	Multi[d]	Explicit	✓

Note that a ✓indicates full support, an ✗indicates no support, and a ± sign signifies partial support

[a]We consider a solution to be compliant with IPv6 over the TSCH mode of IEEE 802.15.4 (6TiSCH) if it can be integrated into a 6TiSCH stack as-is

[b]Multiple packets may be aggregated into a single timeslot

[c]6TiSCH compliance depends on the transmission acknowledgement scheme

[d]While discovery is multi-mode, negotiation traffic isn't

where $T_{a,b}$ is the set of consecutive slot offsets in the superslot starting at slot offset a and ending after slot offset b; $|SO|$ is the number of slot offsets in a slotframe (starting from 0), i.e., the"slotframe length"; SO is a slot offset expressing a discrete time position relative to the start of the slotframe; CO_{SO} is the channel offset at a given slot offset in the slotframe; and $|CO|$ is the number of channel offsets in a slotframe (starting from 0).

Second, the 6P slotframe uses only the slowest (and presumably most robust) communication mode. Hence, if said communication mode can't be used for some reason, you would not be able to negotiate subcells in the third slotframe. However, installing autonomous supercells for multiple communication modes in a single slotframe is non-trivial. For example, you couldn't use the same hashing function to calculate the slot and channel offset of (the first subcell of) autonomous supercells for different communication modes, because they would likely collide.[11] Even if you were to solve this problem, autonomous (super)cells take up valuable time-frequency space. That is, some of them remain in the slotframe indefinitely and autonomous (super)cells always take precedence over negotiated (super)cells when merging slotframes (which only looks at slot offsets). It is easy to see why this may be problematic, especially in dense networks.

Finally, the authors modified 6P `ADD` requests [25, Section 3.3.1] to indicate the communication mode to be used in requested cells. Thus, if a node wants to negotiate a supercell for a link towards a given neighbor, it sends out and `ADD` request (on an autonomous cell towards said neighbor) containing proposed groupings of consecutive cells. Naturally, each grouping must at least contain enough cells to support a single transmission/exchange using the indicated communication mode.

For the sake of clarity, Table 1 provides an overview of the discussed TSCH-based solutions/approaches towards multi-modal IIoT, and how they differ from one another in the most basic sense possible.

[11] Unless, of course, they belong to distinct interfaces operating in different frequency bands.

3.2 The Future of Multi-Modal TSCH

In the previous section we discussed multi-modal TSCH solutions and identified key design choices. Two main approaches emerged, i.e., to use either fixed or variable duration timeslots. Which approach is "best" depends on your specific use-case. Nonetheless, we see opportunities for improving on the discussed solutions, irrespective of slot duration. Firstly, they all assumed one "well-chosen" (i.e., most robust) communication mode would ensure adequate delivery of management traffic critical to TSCH network operation, either in part [23], or in whole [22]. This is a missed opportunity to finally take departure from the (borderline) guesswork inherent to designing single-mode (I)IoT networks. After all, choosing one communication mode either requires substantial prior knowledge, or otherwise leads to less-than-ideal implementations based on simplistic best practices. Moreover, the environments in which IIoT networks are deployed are rarely static. As such, even if you did initially make an informed decision, it may not remain the best choice down the line, e.g., because a sudden change in the electromagnetic properties of an environment disproportionately affects the chosen communication mode through fading. Similarly, it leaves the network vulnerable to interference/jamming. As such, we strongly recommend that future research at least considers using two or more communication modes for all management traffic.

Although it will likely always require more energy to use multiple communication modes for all management traffic, the strain it puts on scheduling can (partially) be solved, especially for unicast management traffic such as the autonomous 6top transactions in [23]. For example, if you allow scheduling (sub)slots with the same slot offset for a given node-pair, provided they use "orthogonal" communication modes, you could at least use those modes for unicast management traffic to provide some resilience against sudden fading and interference/jamming (that is, if other management traffic is also multi-mode). We define two communication modes to be orthogonal for a given node-pair if, and only if, both nodes can simultaneously transmit and/or receive using said modes without them interfering. In most cases, this means they belong to separate interfaces, the need for which may be cost and/or footprint prohibitive. As always, trade-offs remain inevitable.

Second, with many focusing on the time domain, the frequency domain implications of multi-PHY slotframes are easily overlooked. As was mentioned by Van Leemput et al. [22], a major consideration is whether to allow different PHYs to occupy the same frequency space (assuming they normally operate in the same band). That is, if you divide a frequency band into channels only to be used by a given PHY, you reduce the number of channels available to each PHY. Hence, you also increase the chance of collisions between transmissions of disjoint node-pairs that use the same PHY. Alternatively, if overlap is allowed and each PHY has access to the same channels, the chance of such collisions does not increase. However, you have now introduced a chance of collisions between transmissions of disjoint node-pairs that use different PHYs. Conceivably, the overall chance of collisions between disjoint node-pairs is larger in one of these scenarios. Which scenario that is depends on the likelihood of one PHY being used over the other, how many channels are available to each PHY in either scenario, how wide the channels of each PHY are in case they are allowed to overlap, and so on. Note that collisions between disjoint node-pairs apply mainly to autonomous and decentralized schedulers, as centralized schedulers would technically have all the information required to prevent such collisions from happening. However, centralized schedulers typically require much more time-frequency resources for management traffic and this problem tends to become worse as the network grows denser.

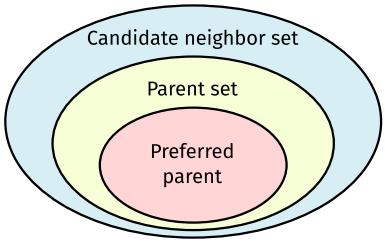

Fig. 7 Relationship between three conceptual Routing Protocol for Low-power and Lossy Networks (LLNs) (RPL) sets tracked by each network node. A RPL Objective Function (OF) dictates how these sets are formed. Reprinted from [27] with permission from the authors

Finally, the advances made/required to enable multi-modal IIoT (and for TSCH in particular) have different (related) applications as well. TSCH currently focuses primarily on connectivity. However, there is a clear trend towards reusing time-frequency resources for other purposes such as, e.g., localisation. Novel multi-modal TSCH solutions could focus on supporting slots with durations and/or structures suitable for such purposes. For example, one could define a new TSCH slot structure for ranging to/with multiple nodes at once and combine it with the idea of variable slot duration by allowing the slot to change in duration depending on: (1) how many nodes you want to range to/with; and (2) which communication mode you're using to perform ranging with. After all, ranging to individual nodes at different slot offsets (or even worse, in consecutive slotframes) and combining those data points to get a location is often not accurate (enough).

4 The Routing Layer: Multi-Modal Solutions and Challenges

RPL is the default routing protocol for IETF-based IIoT. It is a distance-vector routing protocol which forms a Destination Oriented Directed Acyclic Graph (DODAG), i.e., a tree-like routing topology converging at a sink or "root" node. For this purpose, each RPL node (conceptually) keeps track of three sets of neighbors (i.e., directly connected nodes; see Fig. 7): (1) the candidate neighbor set; (2) the parent set; and 3) the preferred parent (= the default next hop up the DODAG, i.e., towards the sink/root).

Managing these sets requires information spread through the DODAG via control messages called DODAG Information Objects (DIOs) and the DODAG Configuration option typically attached to them. Most importantly, RPL nodes use DIOs to advertise their "rank" in (a given version of) the DODAG. The concept of node rank constitutes RPL's distance-vector component, i.e., it is "a scalar representation of the location or radius of a node within a DODAG Version" [13, p. 20]. Node ranks must decrease following an end-to-end path to the DODAG root, and they must do so monotonically. This property allows for routing loop avoidance [28, Section 4.1]. When a routing loop does occur, it must be detected so the DODAG can be repaired. For this purpose, an abstract collection of information called the RPL Packet Information (RPI) is attached to most data packets. The RPI, together with a node's rank, enables loop detection [13, Section 11.2].

The traffic flow in a RPL DODAG is mostly Multi-Point to Point (MP2P), i.e., traffic flowing up the DODAG and converging at the root. Hence, the formation/maintenance of an "upward" routing topology is arguably the most important part of RPL. Say a newly-booted RPL node wants to join a version of a DODAG (for which it received a DIO). Then, in order for it to become a parent itself and start advertising, it must attach to the DODAG and obtain a rank. The node is considered "attached" as soon as its parent set is no longer empty. At some point during or after forming a parent set (from the candidate neighbor set)

and selecting a preferred parent (from the parent set), the RPL node computes the rank it will thereafter advertise (in DIOs) towards nodes that may then become its children.

Upward route discovery and topology formation/maintenance is governed by an Objective Function (OF). Firstly, as stated by RFC 6550 [13, p. 67], "the candidate neighbor set is a subset of the nodes that can be reached via link-local multicast. The selection of this set is implementation and OF dependent." Next, the OF partially prescribes how an RPL node should form a parent set from the candidate neighbor set. We say "partially" because universal (i.e., OF-agnostic) rules do apply. Most importantly, "a node's advertised rank must be greater than the rank advertised by any of its parent set members" [27, p. 4].

The OF's primary objectives are preferred parent selection and rank computation. Both processes typically require a routing metric [29]. For the sake of clarity,"it is useful to distinguish between: (1) a metric that a node can infer from the link towards a neighbor (a link metric) or based on a property it possesses (a node metric); and (2) a metric reported by a neighbor, based on the upward path it provides" [27, p. 4].

Depending on the OF, rank computation may or may not be part of forming the parent set and selecting a preferred parent. For example, with Objective Function Zero (OF0) [30], for every candidate neighbor, a node first computes (see [27, Equation 1]) what its rank would be if it chose said neighbor as its preferred parent. It then keeps (in its parent set) the two nodes for which its rank would be lowest (provided they comply with universal rules). The parent for which the computed rank is lowest becomes preferred and the node shall advertise said rank in future DIOs.

Alternatively, rank computation might follow preferred parent selection. For example, with the Minimum Rank with Hysteresis Objective Function (MRHOF) [31], a node calculates the "path cost" through every candidate neighbor, keeps (in its parent set) a fixed number of candidate neighbors through which the path cost is lowest (provided they comply with universal/MRHOF-specific rules), and sets the lowest-cost parent as preferred parent, that is if the improvement (in path cost) over the current preferred parent exceeds a hysteresis threshold. Only then would a node perform rank computation for each parent and determine the rank to advertise in its future DIOs, as described in [31].

Since OF rules have a big impact on network performance in many different ways and under many different circumstances, *RPL does not mandate a specific OF*. As such, *numerous OFs* (and RPL optimizations in general) *have been proposed in scientific literature* in the past decade. However, *almost none of them have looked at incorporating the multi-modal capabilities of modern IoT devices*, not as a means to an end (i.e., to achieve an optimization-goal) nor as a goal in itself (i.e., from a purely technical RPL-management standpoint).

4.1 Recent Advances in Multi-Modal RPL

Little RPL-related research has focused on multi-modal operation. The few RPL-based solutions that do can generally be subdivided into three categories, as first identified by [32] and generalized by us for the purpose of this paper:

1. Each communication mode a device possesses is tied to a separate DODAG (in a separate RPL instance; see [13, Section 5]), meaning devices would switch data packets to a different DODAG if they wanted to forward them over a more appropriate communica-

tion mode. These solutions can broadly be classified as *Multiple RPL Instances (MI)* approaches.
2. Each communication mode a device possesses is seen as a separate RPL neighbor within a single DODAG. These solutions can broadly be classified as *Mode-Oriented (MO)* approaches. The extent to which the distinction between RPL neighbors translates to addressable entities on the link- and routing layer varies between proposed MO solutions (see Sect. 4.1).
3. Physical devices, and not communication modes, are seen as separate RPL neighbors within a single DODAG. Presumably, this also means that devices, and not communication modes, are the addressable entities on both link- and routing layer. These solutions can broadly be classified as *Parent-Oriented (PO)* approaches.

The remainder of this subsection discusses recent advances in multi-modal RPL in light of these three broad categories.

Pignolet et al. [33] (to our knowledge) first described how to extend RPL to enable multi-modal operation on the network layer. Later on, that same group of researchers published a similar proposal [34]. This approach falls in the MO category. Each physical device has two modes of communication in the form of two separate (statically configured) interfaces, and each communication mode through which an adjacent device or "node" may be reached is a separate RPL neighbor. Every communication mode is a separate addressable entity on the link and (to an extent) network layer. As we understand it, physical nodes do however keep just one routing table. Presumably this means routable IPv6 addresses belong to nodes and not their communication modes (as opposed to link-local IPv6 addresses).

Lemercier et al. [32, 35] came up with three different solutions, each aligning with an aforementioned category, i.e., the MI, PO, and Interface Oriented (IO) approach respectively. The authors assumed the communication modes to be separate (statically configured) interfaces. As such, IO solutions are just a subset of MO solutions. Anyhow, the authors' MI solution ties every communication mode to a separate DODAG. Its main advantage is the near-standard RPL implementation for every communication mode, each with its own (potentially) technology-specific OF. In contrast, packets are only allowed to switch to another communication mode once and in a given direction (i.e., between modes) only, as a means of loop prevention. Therefore, the proposed MI solution can handle but a single link ($\neq$ transmission) failure along a path at best.

Before we continue, we must introduce the concept of physical and virtual links, and how it applies (exclusively) to PO solutions. More specifically, a "virtual link towards a neighbor abstracts a nominal amount of physical links" [27, p. 9]. An inferred metric is associated with every physical link towards a neighbor, and from these inferred metrics the OF describes how to calculate a normalized metric associated with the virtual link towards that same neighbor. Metric normalization is typically abstracted from the rest of RPL operations. Physical links exist between corresponding communication modes on separate devices (i.e., neighbors), whilst virtual links "exist" between separate RPL neighbors. As such, if you where to use TSCH as the underlying MAC protocol, TSCH links would be redefined as the pairwise assignment of directed communication between devices in a given timeslot, on a given channel offset, and with a given communication mode.

With that in mind, Lemercier et al.'s [32, 35] PO solution is straightforward. The authors defined (what they called) a hybrid OF, which we termed the PO OF (POOF) [27]. POOF is based on OF0 [30], the main difference being the addition of a metric normalization

process. In perfect conditions, the normalized metric (which is used during rank computation) is simply the average inferred metric to a neighbor over all communication modes. However, when a communication mode is "unavailable",[12] its inferred metric is temporarily replaced by a large placeholder value when averaging. Averaging works for POOF because the authors assumed all devices possessed the same set of unique communication modes.

Contrary to OF0, POOF does not limit the parent set size and all candidate neighbors that satisfy universal rules [13, Sections 8.2.1 and 8.2.2.4] are considered parents. The parent for which the the computed rank is lowest becomes preferred. However, when a communication mode of the currently preferred parent has become unavailable since the last OF-call, its metric normalization is deferred until the next OF-call. Delaying metric normalization in this manner attempts to prevent DODAG instability.

Unlike MO solutions, PO solutions need a mechanism for selecting the default or "preferred" mode for (unicast) communication towards every candidate neighbor. This is actually an advantage for two reasons: (1) it means that switching communication modes does not automatically require a parent change; and (2) it allows for an additional degree of optimization in forming/maintaining the DODAG. The OF defines the criteria used for preferred mode selection (≠ preferred parent selection). For example, with POOF, the communication mode through which the inferred metric towards a neighbor is "best" becomes preferred.

Finally, Lemercier et al. [32, 35] defined an IO solution (which falls in the MO category) wherein each communication mode (= interface, in this case) is a separate RPL neighbor. However, it is not immediately clear to us whether communication modes or devices are the addressable entities on the link and/or routing layer. Like POOF, this solution is also based on OF0, but it does not rate-limit parent changes. The authors note that the absence of such mechanism, combined with the need for a parent change when a communication mode fails, leads to DODAG instability.

Rady et al. [23, 36] proposed a multi-modal 6TiSCH stack. Their solution falls in the MO category. The authors mention them looking at active TSCH-based link configuration solutions (e.g., [22]; see Sect. 3.1) and realizing there was much to be gained by making RPL aware of communication modes as well, rather than leaving all decision making up to the link-layer. To this end, the authors defined every communication mode (= a combination of PHY and interface, in this case) to be an independent RPL neighbor. However, contrary to [33, 34] (as far as we could tell), devices (≠ communication modes) are the addressable entities on both link- and routing layer. More specifically, the "combination of [a] neighbor IPv6 address and a PHY link is considered an independent [RPL] neighbor" [36, p. 84472].

To our knowledge, the work of Rady et al. [23, 36] is currently the only solution that works with 6TiSCH by design. The authors specifically re-purposed reserved bits in the `Link Options` field [1, Figure 7-55] typically present[13] in TSCH Enhanced Beacons EBs such that unsynchronized nodes may synchronize to the slotframe and learn about communication modes. In addition, they extended 6top `ADD` request and response

[12] When exactly a communication mode is considered unavailable depends on the metric used and/or implementation-specific rules.

[13] To be precise, the `Link Options` field is part of a `Link Information` field [1, Figure 7-54], which is itself part of a `Slotframe Descriptor` field [1, Figure 7-53], which in turn is contained within a TSCH Slotframe and Link Informational Element (IE) [1, Figure 7-52] that is typically attached to TSCH Enhanced Beacons (EBs).

exchanges [25, Section 3.3.1] such that synchronized nodes for which the RPL OF has determined a preferred parent (= IPv6 addr. + mode) may request the corresponding node to allocate dedicated cells for incoming data traffic using the communication mode determined by the OF. This seems confusing until you remember each node is a single addressable entity which owns several communication modes, each representing a separate RPL neighbor (which may in turn become a parent).

Most recently, the authors of [27] proposed the Dual-Radio interface RPL Objective Function (DRiPLOF). DRiPLOF is a PO solution which, on the surface, resembles POOF. That is, they both penalize neighbors (= physical nodes) for not being available through a communication mode. However, DRiPLOF differs majorly from POOF in: (1) how the normalized metric is calculated; and (2) how parent changes are rate-limited.

Both POOF and DRiPLOF administer metric penalties during metric normalization. Unlike POOF, the normalized metric of DRiPLOF equals the inferred metric of the preferred communication mode towards a given neighbor, that is provided all its communication modes are available. If not, the normalized metric is progressively biased towards a large constant "scale multiplier" for every unavailable communication mode. This translates to a two-step process of: (1) obtaining a metric normalization weight based on the amount of unavailable communication modes towards a given neighbor, according to (4); and (2) using said weight (and a constant scale multiplier) to derive a normalized metric from the inferred metric of the preferred communication mode towards the neighbor, according to (5).

$$W = min\big(C_{node} - VL, IL_{max}\big)/IL_{div}, \tag{4}$$

$$M_{norm} = (W \times S) + \big((1 - W) \times M_{pref}\big), \tag{5}$$

where W is the metric normalization weight; C_{node} is the number of unique communication modes per node; VL is the number of valid physical links towards a given neighbor; IL_{max} is the maximum number of invalid physical links towards any neighbor, must be ≥ 0 and $< C_{node}$; IL_{div} is a constant divider, must be $> IL_{max}$; M_{norm} is the normalized metric of the virtual link towards a given neighbor; S is a constant scale multiplier; and M_{pref} is the inferred metric of the preferred physical link towards a given neighbor.

Second, DRiPLOF and POOF rate-limit parent changes differently. With Lemercier et al.'s [32, 35] PO solution, a custom stability mechanism was required. This mechanism is difficult to enforce consistently, that is (in specific circumstances) POOF defers metric normalization for the currently preferred parent by one OF-call. However, when exactly the OF is called tends to vary between different OSs. DRiPLOF avoids such inconsistencies by being based on MRHOF[14] [31] instead of OF0.

Figure 8 illustrates how DRiPLOF forms/maintains a DODAG assuming the use of: (1) Expected number of Transmissions (ETX) [29, Section 4.3.2] as inferred metric; and (2) freshness probes [27, Appendix B] to keep inferred metrics updated. The root starts a DODAG by broadcasting DIOs across all interfaces. Upon receiving such a DIO on its green interface (which defaults to ETX = 3), node #1 sets it as preferred interface towards the root. However, the normalized metric to the root is penalized ($0.25 \times 8 + 0.75 \times 3 = 4.25$), as its blue interface is not known yet (that is, to #1). After selecting a preferred interface to the root, #1 joins the DODAG and sets the root as

[14] Remember MRHOF has a built-in mechanism to prevent excessive parent changes.

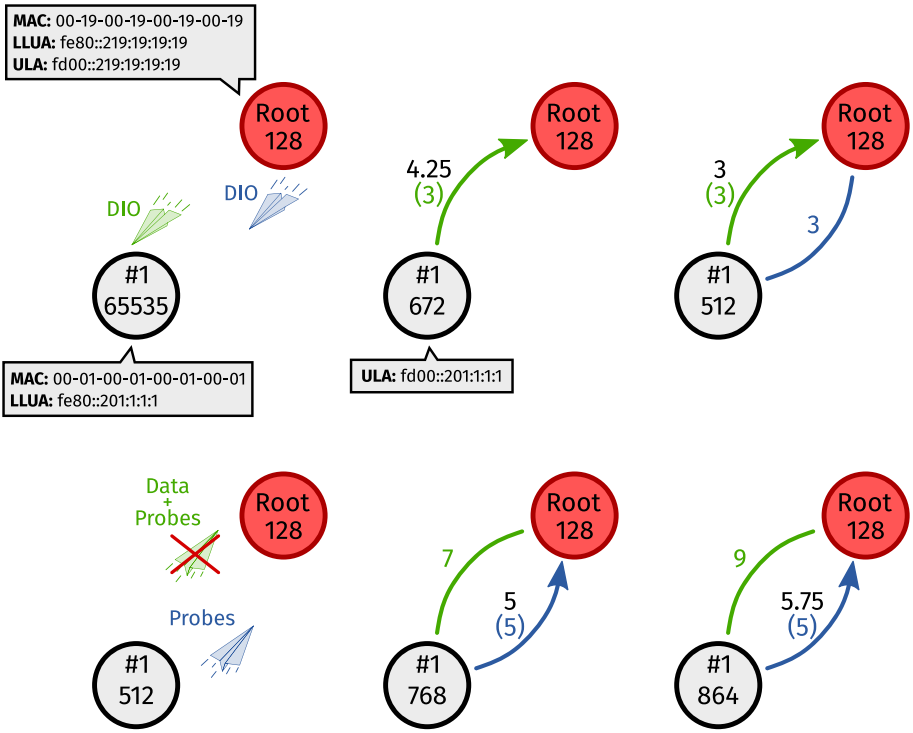

Fig. 8 Example of how the Dual-Radio Interface Routing Protocol For LLNs Objective Function (DRi-PLOF) [27] builds an upward routing topology. This figure is to be viewed left to right, top to bottom. All nodes have a blue and green interface. Inferred metrics and their corresponding interface are drawn in the same color; normalized metrics are in black; arrows indicate which interface is preferred. Note that $I_{node} = 2$, $IL_{max} = 1$, $IL_{div} = 4$, and $S = 8$. Reprinted from [27] with permission from the authors

preferred parent. When #1 finally receives a DIO on its blue interface (which also defaults to ETX = 3), the normalized metric penalty is lifted. Through the data sent to the root over its green (i.e., preferred) interface, and freshness probes [27, Appendix B] sent over both interfaces, #1 keeps all inferred metrics up to date. Then, let's say at a certain point, #1 can no longer reach the root over its green interface. Initially, this will only cause a preferred interface switch. However, eventually, it leads to #1's green physical link (to the root) exceeding the metric threshold, causing the normalized metric of the virtual link (to the root) to be penalized once again.

For the sake of clarity, Table 2 provides an overview of the discussed RPL-based solutions towards multi-modal IIoT, and how they differ from one another in the most basic sense possible.

4.2 The Future of Multi-Modal RPL

To us, PO approaches are most promising. Firstly, they are more scalable than MI or MO alternatives. For example, MI solutions run several DODAGs at once, which presumably operate 100% independent of one another (after all, that was part of MI's supposed strength). Hence, the traffic of a DODAG could interfere with that of another. This

Table 2 Overview of IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) solutions that support multi-modal operation

References	Solution category	Stability mechanism	6TiSCH compliant[a]	Interfaces
[32, 35]	MI	✗	✓	1 × 802.15.4g 1 × PLC
	PO	±	✗	
	MO	✗	✗	
[33, 34]	MO	✓	✗	1 × 802.15.4g 1 × PLC
[23, 36]	MO	✓	✓	2 × 802.15.4g
[27]	PO	✓	✗	2 × 802.15.4 (any)[b]

Note that a ✓indicates full support, an ✗indicates no support, a ± sign signifies partial support, and a **?** means we didn't have enough information to determine the kind of support (if any). Adapted from [27] with permission from the authors

[a] We consider a solution to be compliant with 6TiSCH if it can be integrated into a 6TiSCH stack as-is

[b] Although simulations were conducted with two interfaces per node, this solution technically supports more (interfaces per node, that is)

becomes exponentially worse as more communication modes are added. Indeed, this is "solved" if the communication modes used are orthogonal. However, this then limits the ability to fully harness the capabilities of modern IoT devices, which is our ultimate goal. Moreover, the memory required to run multiple concurrent DODAGs may by itself be prohibitive to many IoT devices. Similarly, MO approaches can be quite memory-heavy, as every communication mode is (at least) a separate RPL neighbor, which in turn leads to larger parent sets, and so on.

Another concern of MO approaches is that when the parent set is limited in size (e.g., because of memory constraints), you risk selecting parents clustered together at a limited amount of (nearby) physical devices. This is especially dangerous with battery-powered devices. Moreover, co-located communication modes may be impacted by the same local wireless distortions, thus posing a security risk. More specifically, this would make it even easier for a jammer to leave the network in a continuous state of convergence by periodically causing a node with a large sub-DODAG to empty its parent set and trigger a local DODAG repair, which is already a problem in non-multi-modal RPL networks, as demonstrated by Tabaja and Cohen [37].

Lastly, PO solutions avoid yet another concern of MO solutions, namely that changing to another communication mode requires a parent change, a (potentially extremely[15]) costly operation for a distance-vector routing protocol. In fact, Lemercier et al. [32, 35] have demonstrated that their PO proposal generally outperformed their own MO proposal, precisely because of the overhead caused by an increased number of parent changes. That said, they did not rate-limit parent changes in their MO approach, whereas Rady et al. [36] did. However, this simply shifts the problem and in turn highlights the inevitable trade-off with an MO solution: rate-limiting parent changes also limits the ability of the network to swiftly act on local wireless distortions, a virtually non-existent issue with (well-designed) PO approaches.

That said, PO solutions are not perfect. For example, both POOF [32, 35] and DRiPLOF [27] are based on a "ground truth". More specifically, both solutions require all

[15] That is, if said parent change leads to a local repair being triggered.

nodes to have a shared notion of the nominal amount of communication modes per device (denoted C_{node} in Fig. 8) in order for metric normalization to work reliably.[16] It is unreasonable to assume that all device vendors would provide the same set of communication modes. Indeed, this is easily circumvented by statically configuring each node to use the same set of communication modes or with knowledge about the communication modes it should possess. However, if you then wanted to modify this ground truth, the entire network would have to be re-programmed, a simply unacceptable operation for many IIoT networks that require minimal downtime. As such, the PO solutions from literature would benefit from a mechanism to spread this ground truth throughout the network, e.g., by repurposing the 8-bit Reserved field of DODAG Configuration options [13, Section 6.7.6], (potentially) combined with some way for non-root nodes to request a modification of the ground truth propagated by the root in those DODAG Configurations options (which can't be modified by non-root nodes).

Moreover, none of the discussed PO approaches can be integrated in a 6TiSCH stack without additional specifications/rules for link-layer operation.

Finally, the availability of multiple communication modes can likely be put to good use for other RPL-related research efforts such as, e.g., enabling application-aware RPL routing. However, forwarding to a different neighbor based on the type of traffic (presumably using a different communication mode) presents its own set of unique challenges, which may or may not conflict with key aspects of different types of multi-modal RPL routing solutions.

5 Conclusion

The increasingly powerful communication capabilities of modern (I)IoT devices are poorly reflected at the link and routing layer. More specifically, in IETF-based IIoT deployments, devices are typically limited to one communication mode for their entire operational lifetime, often based on network-wide worst-case operating conditions. Instead, individual links could (dynamically) be configured to use a given communication mode in response to local operating conditions (or to accomplish some other optimization goal). To achieve this, devices must be able to switch between modes at run-time, i.e., the network must become multi-modal. Enabling multi-modal operation can be done in many ways, both at the link and routing layer (or, most likely, a combination thereof). As such, in this paper, we explored the limited research available on multi-modal (IETF-based) IIoT, in an effort provide a clear path forward for the (presumably 6TiSCH-compliant) IIoT deployments of tomorrow.

In summary, it is our opinion that multi-modal 6TiSCH-compliant solutions using an active/explicit TSCH-based link layer configuration combined with PO RPL-based routing hold the most promise. This requires several innovations compared to the current state-of-the-art. Most importantly, none of the PO RPL solutions described in literature were designed with a synchronized MAC protocol (let alone TSCH) in mind, meaning they were limited to using orthogonal communication modes only, which, in practice, translated to multi-interfaced operation. Since TSCH comes with its own management traffic (for synchronization purposes), the information obtainable from (part of) this traffic likely removes the need for (what would then become) excessive RPL-management traffic currently (i.e., with a non-synchronized MAC protocol) fulfilling the same purpose. As such,

[16] Otherwise, the severity of a metric penalty would be inconsistent.

these management traffic semantics must be revisited. In addition, it must be investigated whether the ability of a PO solution to change preferred communication mode (towards its preferred parent) without automatically needing a parent change can be used to facilitate some basic form of application awareness on the link layer. Furthermore, combining application awareness on the routing layer ($\neq$ link layer) with a multi-modal RPL approach could be incredibly powerful. Finally, current PO solutions rely on all RPL nodes being aware of a common ground truth (i.e., the nominal amount of communication modes). However, for now, the only way to facilitate this is to hard-code it onto every device, which is less-than-ideal. As such, it must be investigated whether this ground truth might instead be propagated through the DODAG by the root. The latter fits within a broader discussion that needs to be had about how the design choices we identified apply to heterogeneous networks specifically.

Less clear-cut is the debate of fixed versus variable TSCH slot duration. With fixed duration timeslots, the ability to use most existing schedulers is presumably the biggest advantage. However, if you remember that IEEE 802.15.4 (i.e., the standard that defines TSCH) deliberately leaves the scheduling algorithm at the discretion of the implementer, precisely because the most optimal schedule so heavily depends on the use-case, one does wonder whether this ability holds any value other than the (relative) ease of adoption. That said, ease of adoption alone may well be a strong enough argument for fixed duration timeslots, especially with a technology stack plagued by complexity (like, e.g., 6TiSCH). Variable slot duration, on the other hand, does imply the need for a custom scheduler. Nonetheless, variable duration timeslots are likely to result in a more efficient schedule, even compared to fixed duration slots with transmission aggregation (although that would need more testing). While the maximum achievable throughput is presumably very similar (i.e., for fixed duration slots with transmission aggregation), the actual achieved end-to-end throughput (i.e., from a leaf node the root) is likely higher with variable slot duration, largely because of the freedom the scheduler would have in terms of (virtual) timeslot boundaries. Besides, transmission aggregation likely does not help with end-to-end latency anyway, as that is mainly determined by slotframe instead of timeslot duration, and slotframes could conceivably be shorter for a more efficient schedule. All this to say that your mileage may vary when deciding on either fixed or variable slot duration for your own (future) multi-modal 6TiSCH stack implementation.

Author Contributions All authors contributed to the study conception and design. The first draft of the manuscript was written by Robbe Elsas and all authors commented on previous versions of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding This research was funded by the Research Foundation - Flanders (FWO) through strategic basic research grant 95027: *SDR-Based Industrial IoT: Towards Ubiquitous Wireless Connectivity*.

Data Availability Not applicable.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. IEEE Standard for Low-Rate Wireless Networks, IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015), July (2020).
2. Chen, Y., Lu, S., Kim, H.-S., Blaauw, D., Dreslinski, R. G., & Mudge, T. (2016). A low power software-defined-radio baseband processor for the internet of things. In *Proceedings of the 2016 IEEE international symposium on high performance computer architecture (HPCA'16)* (pp. 40–51).
3. Amor, H. B., & Bernier, C. (2019). Software-hardware co-design of multi-standard digital baseband processor for IoT. In *Proceedings of the 2019 design, automation & test in europe conference & exhibition (DATE'19)* (pp. 646–649).
4. Xhonneux, M., Louveaux, J., & Bol, D. (2021). Implementing a LoRa software-defined radio on a general-purpose ULP microcontroller. In *Proceedings of the 2021 IEEE workshop on signal processing systems (SiPS'21)* (pp. 105–110).
5. Aalsalem, M. Y., Khan, W. Z., Gharibi, W., Khan, M. K., & Arshad, Q. (2018). Wireless sensor networks in oil and gas industry: Recent advances, taxonomy, requirements, and open challenges. *Journal of Network and Computer Applications, 113*, 87–97.
6. IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), September (2011).
7. 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 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011), April (2012).
8. Grami, A. (2016). *Introduction to digital communications*. Academic Press.
9. Zigbee Specification, Zigbee Alliance Document 05-3474-22, Rev. R22 1.0, March (2017).
10. Herrero, R. (2022). Thread architecture. In *Fundamentals of IoT communication technologies* (pp. 213–225). Springer.
11. Montenegro, G., Hui, J., Culler, D., & Kushalnagar, N. (2007). Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4944.txt
12. Thubert, P., & Hui, J. (2011). Compression Format for IPv6 datagrams over IEEE 802.15.4-based networks.' RFC 6282. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6282.txt
13. Alexander, R., Brandt, A., Vasseur, J., Hui, J., Pister, K., Thubert, P., Levis, P., Struik, R., Kelsey, R., & Winter, T. (2012). RPL: IPv6 Routing protocol for low-power and lossy networks. RFC 6550. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt
14. IEEE Standard for Low-Rate Wireless Networks, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), (2016).
15. TDMA Data Link Layer Specification, FieldComm Group Document HCF_SPEC-075 FCG TS20075, Rev. 1.2, (2016). [Online]. Available: https://library.fieldcommgroup.org/20075/TS20075/1.2/
16. Nixon, M. (2012). A comparison of WirelessHART and ISA100.11a. Emerson Process Management, Round Rock, TX, USA, Tech. Rep., [Online]. Available: https://www.controlglobal.com/assets/15WPpdf/150423-Emerson-WirelessHART-ISA10011a.pdf
17. Vilajosana, X., Pister, K., & Watteyne, T. (2017). Minimal IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) configuration. RFC 8180, [Online]. Available: https://www.rfc-editor.org/rfc/rfc8180.txt
18. Thubert, P. (2021). An architecture for IPv6 over the time-slotted channel hopping mode of IEEE 802.15.4 (6TiSCH). RFC 9030. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9030.txt
19. Gaglio, S., & Re, G. L. (2014). *Advances onto the Internet of Things* (Vol. 349). Springer.
20. Duquennoy, S., Al Nahas, B., Landsiedel, O., & Watteyne, T. (2015). Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proceedings of the 13th ACM conference on embedded networked sensor systems* (pp. 337–350).
21. Brachmann, M., Duquennoy, S., Tsiftes, N., & Voigt, T. (2019). IEEE 802.15.4 TSCH in Sub-GHz: Design considerations and multi-band support. In *Proceedings of the 2019 IEEE 44th Conference on local computer networks (LCN'19)* (pp. 42–50). IEEE.
22. Van Leemput, D., Bauwens, J., Elsas, R., Hoebeke, J., Joseph, W., & De Poorter, E. (2021). Adaptive multi-PHY IEEE802.15.4 TSCH in sub-GHz industrial wireless networks. *Ad Hoc Networks* (Vol. 111).

23. Rady, M., Lampin, Q., Barthel, D., & Watteyne, T. (2021). 6DYN: 6TiSCH with Heterogeneous slot durations. *Sensors, 21*(5).
24. Chang, T., Vučinić, M., Vilajosana, X., Duquennoy, S., & Dujovne, D. R. (2021) 6TiSCH minimal scheduling function (MSF). RFC 9033. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9033.txt
25. Wang, Q., Vilajosana, X., & Watteyne, T. (2018). 6TiSCH operation sublayer (6top) protocol (6P). RFC 8480. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8480.txt
26. Rady, M. (2020). minarady1/openwsn-fw at develop_FW-891. [Online]. Available: https://github.com/minarady1/openwsn-fw/tree/develop_FW-891
27. Elsas, R., De Poorter, E., & Hoebeke, J. (2022). DRiPLOF: An RPL extension for multi-interface wireless sensor networks in interference-prone environments. *Sensors, 22*(10).
28. Tsvetkov, T. (2011). RPL: IPv6 routing protocol for low power and lossy networks. In *Proceedings of the seminar"sensor nodes: Operation, network and application" (SN'11)* (pp. 59–66).
29. Barthel, D., Vasseur, J., Pister, K., Kim, M., & Dejean, N. (2012). Routing metrics used for path calculation in low-power and lossy networks. RFC 6551. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6551.txt
30. Thubert, P. (2012). Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6552.txt
31. Gnawali, O., & Levis, P. (2012). The minimum rank with hysteresis objective function. RFC 6719. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6719.txt
32. Lemercier, F., Montavont, N., Toutain, L., Vijayasankar, K., Vedantham, R., & Chiummiento P. (2016). Support for hybrid network in RPL. In *Proceedings of the 2016 IEEE international conference on smart grid communications (SmartGridComm'16)* (pp. 527–532).
33. Pignolet, Y.-A., Rinis, I., Dzung, D., & Karaagac, A. (2012). Heterogeneous multi-interface routing: Networking stack and simulator extensions. In *Proceedings of the 2012 IEEE 9th international conference on mobile ad-hoc and sensor systems (MASS'12)* (pp. 1–6).
34. Balmau, O., Dzung, D., Karaağaç, A., Nesovic, V., Paunovic, A., Pignolet, Y. A., & Tehrani, N. A. (2014). Evaluation of RPL for medium voltage power line communication. In *Proceedings of the 2014 IEEE international conference on smart grid communications (SmartGridComm'14)* (pp. 446–451).
35. Lemercier, F., & Montavont, N. (2018). Performance evaluation of a RPL hybrid objective function for the smart grid network. In *Proceedings of the 2018 international conference on ad-hoc networks and wireless (ADHOC-NOW'18)* (pp. 27–38).
36. Rady, M., Lampin, Q., Barthel, D., & Watteyne, T. (2021). g6TiSCH: Generalized 6TiSCH for agile multi-PHY wireless networking. *IEEE Access, 9*, 84465–84479
37. Tabaja, A., & Cohen, R. (2020). When the network of a smart city is not so smart. In *Proceedings of the 2020 IEEE conference on communications and network security (CNS'20)* (pp. 1–9).

Robbe Elsas is a Ph.D. candidate with the IDLab research group of Ghent University and imec. He received a B.Sc. (2017) and M.Sc. (2018) degree in Electronics and ICT Engineering Technology from the University of Antwerp, Belgium. In 2019 he started a Ph.D. fellowship at Ghent University, Belgium under a Special Research Fund (BOF) doctoral scholarship. He has since obtained a strategic basic research grant from the Research Foundation - Flanders (FWO) through which he continues his work at IDLab as a Ph.D. candidate in the field of multi-hop wireless sensor networks for use in highly dynamic/heterogeneous industrial environments.

Dries Van Leemput received his B.Sc. (2017) and M.Sc. (2018) in Electronics and ICT Engineering Technology, and M.Sc. in Electrical Engineering (2020) from Ghent University, Belgium. In 2020, he started as a Ph.D. candidate at Ghent University with the IDLab research group. His research topics are mainly focused on MAC protocol design for critical wireless systems.

Jeroen Hoebeke is an associate professor with the IDLab research group of Ghent University and imec. He conducts and coordinates research on deterministic and time-sensitive wireless communication, wireless network management, tighter application-network integration, (industrial) IoT connectivity and embedded communication stacks. His expertise has been applied in a variety of application domains such as logistics, Industry 4.0, building automation, healthcare and animal monitoring. He is particularly active in nationally funded projects as well as in defining, executing and managing such projects. He has also been involved in several EU-funded research projects and is author or co-author of more than 180 publications in international journals or conference proceedings.

Eli De Poorter is an associate professor with the IDLab research group of Ghent University and imec. He received his master degree in Computer Science Engineering from Ghent University, Belgium, in 2006, his Ph.D. degree in 2011 at the Department of Information Technology at Ghent University, and became professor at IDLab in 2015. Since 2017, he is also affiliated with the imec research institute. His team performs research on wireless communication technologies such as (indoor) localization solutions, wireless IoT solutions, machine learning for wireless systems, and sensor-based health and sport analytics. He performs both fundamental and applied research. For his fundamental research he is currently the coordinator of several research projects (SBO, FWO, GOA, etc.) and has over 200 publications in international journals or in the proceedings of international conferences.