Management of 6TiSCH Networks Using CORECONF: A Clustering Use Case

Fabian Graf, David Pauli, Michael Villnow, Thomas Watteyne

Management of 6TiSCH Networks Using CORECONF: A Clustering Use Case

Fabian Graf*, David Pauli*,†, Michael Villnow*, Thomas Watteyne
*Siemens AG, Erlangen, Germany
†Friedrich-Alexander-Universität, Erlangen, Germany

***Abstract*—Industrial low-power wireless sensor networks demand high reliability and adaptability to cope with dynamic environments and evolving network requirements. While the 6TiSCH protocol stack provides reliable low-power communication, the CoAP Management Interface (CORECONF) for runtime management remains underutilized. In this work, we implement CORECONF and introduce clustering as a practical use case. We implement a cluster formation mechanism aligned with the Routing Protocol for Low-Power and Lossy Networks (RPL) and adjust the TSCH channel-hopping sequence within the established clusters. Two use cases are presented. First, CORECONF is used to mitigate external Wi-Fi interference by forming a cluster with a modified channel set that excludes the affected frequencies. Second, CORECONF is employed to create a priority cluster of sensor nodes that require higher reliability and reduced latency, such as those monitoring critical infrastructure in industrial settings. Simulation results show significant improvements in latency, while practical experiments demonstrate a reduction in overall network charge consumption from approximately 50 mC per hour to 23 mC per hour, by adapting the channel set within the interference-affected cluster.**

***Index Terms*—6TiSCH, CORECONF, IEEE 802.15.4, Clustering.**

I. Introduction

Industrial Internet of Things (IIoT) devices, particularly in the form of Wireless Sensor Networks (WSNs), are ubiquitous in modern industrial environments. They play a crucial role in monitoring various entities within a factory, such as machinery, environmental conditions, and production processes. By providing real-time data and insights, these networks help ensure flawless operation, improve efficiency, and enable predictive maintenance, ultimately reducing downtime and operational costs. It is therefore essential to implement a system that users can rely on. A survey published by the International Society of Automation (ISA) reveals that the vast majority of respondents prioritize reliability when asked about the most important features in a WSN [1].

To meet stringent reliability requirements, wireless standards leveraging frequency diversity have proven highly effective in recent years. Traditional technologies such as Wi-Fi (Wi-Fi), Bluetooth and IEEE 802.15.4 [2] rely on this principle to enhance link reliability. The Time Slotted Channel Hopping (TSCH) mechanism in IEEE 802.15.4 offers several advantages, including resilience to external interference and multi-path fading through continuous frequency hopping, as well as improved energy efficiency by ensuring that each node precisely knows when to transmit, receive and sleep. IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) [3] – an open and standardized communication stack – builds upon this foundation, enabling reliable and deterministic wireless communication within a mesh topology, making it well-suited for diverse industrial applications [4].

Figure 1: Wireless network in a shop floor with a subset of nodes (red) requiring special management during runtime.

Industrial environments are often characterized by dynamic changes in both the physical surroundings and user requirements for the WSN. To ensure uninterrupted operation of the WSN, it is essential for users not to need to reinstall the deployment but instead manage the network seamlessly during runtime. This management includes various aspects, including adjusting wireless communication parameters in response to poor link quality, or reconfiguring node settings to adapt their role within the application. Fig. 1 illustrates a wireless mesh network deployed across different sections of a factory. Depending on evolving requirements, a user may need to intensify monitoring of a specific device, necessitating configuration changes for the subset of wireless sensors attached to it (marked in red). Another possible scenario involves an Application Performance Monitoring and Management (APM) system detecting performance degradation in certain nodes, prompting a management action to mitigate the issue and maintain optimal network performance.

In 6TiSCH networks, management actions are executed at the application layer, typically using the Constrained Application Protocol (CoAP) protocol [5]. To facilitate runtime management, the Internet Engineering Task Force (IETF) has defined the CoAP Management Interface (CORECONF) [6], a lightweight network management protocol that integrates the Yet Another Next Generation (YANG) data modeling

language [7] with CoAP, leveraging Concise Binary Object Representation (CBOR) [8] encoding for enhanced efficiency in constrained IoT and industrial environments. CORECONF enables seamless device configuration, monitoring, and control in low-power and lossy networks, while ensuring compatibility with existing YANG-based management frameworks. Consequently, CORECONF serves as a promising framework for managing configuration resources within the node application at runtime, enabling immediate responses to management demands, whether issued by the user or autonomously triggered by an APM system in reaction to changing performance metrics. However, CORECONF has not yet gained significant popularity, primarily due to the lack of implementations in existing 6TiSCH stacks, and the absence of practical use cases demonstrating its effectiveness in real-world scenarios.

We identified *clustering* as a practical use case for CORECONF. The concept involves dynamically grouping a set of nodes within the 6TiSCH network by modifying their YANG resources at runtime using CORECONF. Nodes within a designated cluster adjust specific application settings, such as sensor reading frequency or reporting intervals. Additionally, clustering enables adaptive tuning of critical network parameters, including Routing Protocol for Low-Power and Lossy Networks (RPL) parent selection mechanisms and TSCH channel-hopping sequences. This approach remains compatible with any Scheduling Function (SF) in use. From a networking perspective, clustering can help mitigate external interference or enhance reliability by prioritizing specific subsets of nodes.

This paper presents a scalable approach for clustering nodes in a 6TiSCH network using CORECONF, including a synchronized roll-out mechanism for cluster assignments. Alongside a detailed system architecture, the focus lies on an experimental evaluation based on two use cases, demonstrating the benefits of dynamically adapting network parameters at runtime to significantly reduce failed transmissions. In summary, the key contributions of this work are:

i Development of a clustering concept for 6TiSCH networks, incorporating adaptive RPL parent selection, cluster-based channel-hopping sequence adaptation, and a synchronized roll-out mechanism.
ii Implementation of a basic CORECONF interface in Contiki-NG [9].
iii Definition of a YANG datastore to support the proposed clustering concept.
iv Simulation-based evaluation of a use case where a subset of nodes is exposed to external interference from a Wi-Fi router.
v Simulation-based evaluation of a use case where a subset of nodes is prioritized, resulting in reduced traffic load, fewer transmissions, lower power consumption, and improved End-to-End (E2E) delay.
vi Experimental evaluation of the Wi-Fi interference mitigation use case in a real-world testbed including power consumption analysis.

The remainder of this article is organized as follows. Section II introduces the technical background on 6TiSCH and CORECONF. Section III discusses related work. Section IV presents a detailed theoretical concept of the clustering approach. Section V provides insights on the system architecture including the CORECONF interface, YANG data model and their practical implementation. Section VI explains the simulation setup in Contiki-NG's Cooja network simulator and analyses the results of two different use cases. Section VII evaluates the effectiveness of the proposed clustering approach in a practical experiment. Finally, Section VIII concludes the paper and provides thoughts on future research directions.

II. Technical Background

In this section, we briefly present the fundamentals of the 6TiSCH stack and CORECONF. Both standards form the basis of the network topologies investigated throughout this paper.

A. 6TiSCH: IPv6 over the TSCH mode of IEEE 802.15.4e

In addition to WiFi (IEEE 802.11) and Bluetooth (IEEE 802.15.1), another key standard in the IIoT landscape are Low-Rate Wireless Personal Area Networks (LR-WPAN), specified in IEEE 802.15.4 [2]. Since its introduction in 2003, IEEE 802.15.4 has undergone several revisions, with TSCH first introduced as an amendment (IEEE 802.15.4e [10]) in 2012 before being fully integrated into the standard in 2015. TSCH operates by ensuring that, when two neighboring nodes exchange frames, they do so on different frequencies for each transmission, enabling channel hopping. This mechanism enhances reliability by mitigating the effects of external interference and multi-path fading. If a transmission fails on one frequency, the retransmission anyway occurs on a different channel, increasing the likelihood of success [11], [12]. Additionally, IEEE 802.15.4 supports multi-hop mesh topologies, requiring strict time synchronization and a predefined communication schedule based on a global Absolute Slot Number (ASN). Typically, time is divided into 10 ms subslots, while the 2.4 GHz frequency band is split into 16 channels. This structured time-frequency matrix serves as the foundation for a SF, which dictates when each node transmits, receives, or enters sleep mode.

The IETF has established the 6TiSCH Working Group to enable IPv6-based communication over the TSCH mode of IEEE 802.15.4e. This group has focused on integrating IPv6 networking technologies, such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN), RPL, User Datagram Protocol (UDP), and CoAP, with TSCH to ensure reliable and deterministic wireless communication. To achieve this, 6TiSCH defines the 6top sub-layer, which facilitates scheduling, TSCH configuration, and network management [13]. A Minimal Scheduling Function (MSF) [14] has also been specified to ensure interoperability across different implementations. Additionally, the research community continues to develop scheduling mechanisms, aiming to provide a standardized framework for dynamic and efficient TSCH network operation.

Tabouche et al. [15] highlight recent advancements in TSCH SFs and optimization strategies tailored for industrial applications. Notably, the review shows that most proposed schedulers build upon two foundational frameworks: MSF and Orchestra.

The MSF [14], [16] is a distributed scheduling algorithm designed to minimize control overhead while ensuring adaptive and reliable communication. Each node autonomously decides when to allocate or de-allocate cells based on traffic requirements and observed link quality. Metrics such as Packet Delivery Ratio (PDR) and Expected Transmission Count (ETX) guide these decisions, enabling nodes to dynamically adjust their schedules. The MSF is particularly effective during network bootstrapping and in environments where traffic patterns are relatively predictable. By operating in a distributed manner, the MSF achieves scalability and robustness while maintaining deterministic behavior.

Orchestra [17] introduces an autonomous scheduling paradigm, where schedules are automatically derived from existing network protocols, such as routing and neighbor discovery. Autonomous schedulers are a special case of distributed/decentralized schedulers. Unlike centralized or distributed schedulers, Orchestra does not rely on explicit communication for schedule management. Instead, it integrates with routing and neighbor tables to allocate transmission opportunities. For example, unicast and broadcast traffic are assigned separate slotframes, which are dynamically adjusted as the network topology evolves. This autonomy simplifies deployment and maintenance, making Orchestra particularly attractive for scenarios with highly dynamic or unpredictable traffic patterns.

B. CORECONF: The CoAP Management Interface

The CORECONF protocol [6], formerly known as "CoMI", is a lightweight management interface optimized for resource-constrained scenarios setting up on CoAP, the application protocol used in 6TiSCH. Unlike Network Configuration Protocol (NETCONF) [18], which operates over Secure Shell Protocol (SSH) with Extensible Markup Language (XML) encoding, and RESTful Configuration Protocol (RESTCONF) [19], which uses Hypertext Transfer Protocol (HTTP) with JavaScript Object Notation (JSON), CORECONF employs CoAP over UDP with the compact CBOR data serialization format. This design significantly reduces both transport and parsing overheads, making it more suitable for low-power wireless links. Table I summarizes and compares the mentioned management interfaces [20].

Table I: Overview of management interfaces and corresponding application protocols.

Mgmt. Interface	NETCONF	RESTCONF	CORECONF
Message Format	XML	JSON	CBOR
Application Protocol	RPC	HTTP	CoAP
Transport Layer	SSH	TCP	UDP
Security Protocol	TLS	TLS	DTLS

CORECONF operates on data structures specified through the YANG modeling language, ensuring interoperability with existing network management ecosystems. To minimize wireless overhead in constrained environments, CORECONF offers an option to replace textual paths in YANG modules with numeric Schema Item iDentifiers (SIDs). These optimizations enable standardized management solutions in constrained environments, while maintaining full compatibility with existing models [20], [6]. The CORECONF architecture adopts a stateless client–server model optimized for efficient management operations in constrained environments. Clients and servers both rely on a shared YANG specification that defines the structure and semantics of manageable resources. A key architectural feature of CORECONF is its reliance on a *single* unified datastore that consolidates both configuration and operational state data. The datastore structure information is part of the YANG specification [7]. Clients interact with servers using well-defined CoAP methods (e.g., `GET`, `PUT`, `POST`, `FETCH`, `iPATCH`, `DELETE`). These requests and responses follow the proven request–response pattern that can be protected by security protocols (e.g., Object Security for Constrained RESTful Environments (OSCORE) [21] or Datagram Transport Layer Security (DTLS) [22]). Clients discover available data nodes and event streams via the YANG library or resource discovery mechanisms, ensuring that all interactions remain in sync with the server's data model. Another essential architectural element is the use of the Concise Binary Object Representation (CBOR) for encoding and transporting data, combined with SIDs. The rules specifying how YANG can be encoded into CBOR format are defined in RFC 9254 [23]. SIDs replace verbose textual YANG identifiers with numeric values, further reducing packet size compared to formats like XML or JSON.

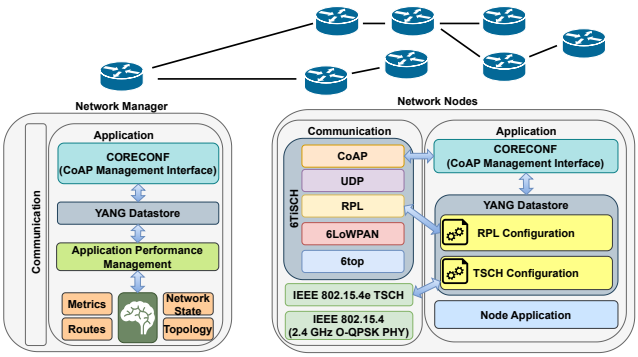

Figure 2: 6TiSCH network architecture including CORECONF.

The integration of CORECONF into the 6TiSCH network architecture is illustrated in Fig. 2. The network manager leverages its centralized knowledge to derive appropriate APM measures. This knowledge includes performance metrics, routing information, overall network state, and physical topology. Based on these insights, it can modify specific resources on individual nodes by issuing CoAP requests, thereby acting as a CORECONF client. Upon receiving such a request, the node processes it via its CORECONF interface, which updates the corresponding resource in the local YANG datastore. In our use cases, these resources are directly tied to configurations within the communication stack, such as RPL and TSCH.

III. Related Work

Practical implementations and use cases of CORECONF, including its predecessor "CoMI", are still rare in the literature. Nevertheless, Table II provides a summary of key related contributions. Additionally, we include studies that address use cases relevant to the focus of this article: clustering, RPL parent selection, scheduling and interference mitigation. The corresponding columns in the table indicate whether each work falls into these categories.

Existing applications of CORECONF in the literature primarily focus on two domains: protocol-related management and scheduling management. In the context of protocol-related management, SCHC [34] and SenML [38] are presented as use cases, both aiming to minimize payload size for wireless transmission. For scheduling management, Karaagac et al.propose a hybrid approach for constructing an SF, which is deployed via CoMI and enables time-critical message delivery [27], [28]. Similarly, Mohamadi et al. [44] introduce the management of "tracks", defined as sets of time-frequency resources across a 6TiSCH network between source and destination nodes, ensuring guaranteed throughput and bounded latency for traffic along the track.

As outlined in the introduction, this is the first work to present clustering as a use case for CORECONF. Although Koesnadi et al. [42] recently proposed cluster formation in 6TiSCH networks by adapting the SF cell allocation, existing clustering-related work generally goes hand in hand with modifications to RPL parent selection. However, these approaches differ from ours: we determine the preferred parent based on the assigned cluster ID, effectively controlling routing paths through the network. In contrast, related work often employs a multi-gateway architecture, where multiple cluster heads act as gateways. We, however, adhere to a single-gateway model. It is important to note that the specific algorithms for cluster and parent selection are beyond the scope of this work. For further details, we refer the reader to [32], [36], [41], [43].

Clustering can serve multiple purposes. In this article, we focus on mitigating external interference and forming a priority cluster. Prior research on interference mitigation happens through channel exclusion in the 6TiSCH schedule. Elsts et al. [24] propose adaptive channel selection algorithms based on RSSI and PDR. However, their approach is limited to individual nodes without considering global network awareness, and it lacks a structured mechanism for distributing the common exclusion-list. Instead, the exclusion-list is propagated by simply inserting it at the front of the regular message queue. Similarly, Kotsiou et al. [25] present a channel exclusion strategy based on PDR, where the exclusion-list is exchanged via the IEEE 802.15.4 IE field. Both approaches, along with other related work listed in [24], provide valuable input for designing our cluster formation strategy and constructing coordinated exclusion-lists. Nevertheless, the deployment of such configurations via CORECONF, as well as the concept of organizing nodes affected by external interference into dedicated clusters, are novel contributions of this work.

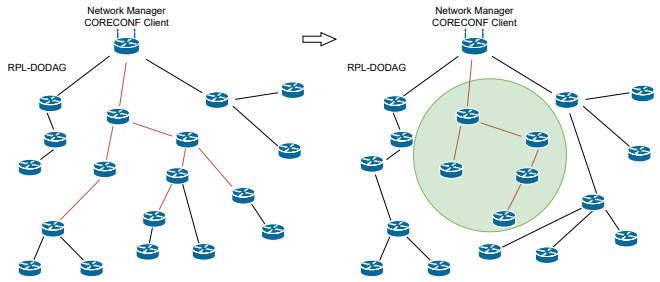

Figure 3: DODAG route refreshment respecting the cluster-membership.

IV. Dynamic Clustering of 6TiSCH Networks

In this section, we introduce dynamic clustering in 6TiSCH networks as a novel use case for CORECONF. The process of determining how clusters should be formed is application-specific and relies on detailed insights, such as those obtained through APM, which is beyond the scope of this work. Instead, our focus is on clustering from a networking perspective—specifically, how adaptive RPL parent selection and channel hopping sequence adjustments can potentially improve PDR and E2E latency. The framework is built to ensure conformance with existing IETF standards, while also providing the flexibility necessary to meet diverse application-level requirements. Additionally, we provide a theoretical explanation of the synchronized roll-out process to ensure robustness.

A. Adaptive RPL Parent Selection

The first mechanism introduces logical clusters by assigning cluster-IDs to groups of nodes, restricting RPL parent selection to within each cluster. This partitions the network into smaller Destination-Oriented Directed Acyclic Graph (DODAG) segments, keeping traffic localized. Cluster assignment is triggered via a CORECONF iPATCH request, which propagates through the network, updating neighbor mappings. As a result, nodes adjust their parent selection accordingly during route updates. Fig. 3 illustrates how clustering can isolate traffic, enhance fault tolerance, or separate real-time control loops from less critical data flows.

Cluster formation limits each node's parent selection, impacting RPL convergence time, especially during route rebuilds. This trade-off enables faster local decisions but risks isolating nodes if clusters are poorly chosen. The method for determining optimal clusters—whether algorithmic or manual—is beyond this work's scope. To prevent network isolation, a fallback mechanism allows nodes to first prioritize same-cluster parents and, if none are found, expand their search beyond cluster boundaries.

Furthermore, the proposed mechanism can be selectively applied to only specific traffic types, such as unicast traffic, which is widely supported by TSCH schedulers due to the ability to operate with multiple slotframes. This ensures that broadcasts and RPL-related traffic continue to function normally across cluster boundaries. As a result, changes in cluster memberships or fallback mechanisms can be carried

Table II: Related work on RPL parent selection, clustering, interference mitigation, scheduling and CORECONF.

Author	Year	Description	Evaluation	RPL Parent Selection	Clustering	Interference Mitigation	Scheduling	CORECONF
Elsts et al. [24]	2017	Adaptive interference mitigation is achieved by channel exclusion based on RSSI and PDR. A decentral local exclusion-list and central shared exclusion-list are used.	Cooja Simulator, TI CC2650			✓	✓	
Kotsiou et al. [25]	2017	The LABeL algorithm proposes the construction of exclusion-lists on the node based on PDR and their exchange via the IEEE 802.15.4 IE field.	FIT IoT-Lab platform with OpenWSN			✓	✓	
Mavromatis et al. [26]	2017	Dense TSCH networks are split in clusters. Each cluster has a cluster head that collects all the data forwards it to the sink node and thus improves RDC and the number of retransmissions.	Cooja Simulator, Zolertia Z1	✓	✓			
Karaagac et al. [27]	2018	A hybrid 6TiSCH schedule combines centrally planned deterministic cells with flexible, dynamically allocated cells within the same TSCH schedule. The schedule is rolled out via CoMI.	OpenMote-CC2538 with OpenWSN				✓	✓
Karaagac et al. [28]	2018	By centralized scheduling, the PCE can form tracks that prioritize certain nodes with time-critical data and provide minimized latency by optimizing the individual path. CoMI is used for schedule remote management.	OpenMote-CC2538 with OpenWSN				✓	✓
Boucetta et al. [29]	2019	SIM Algorithm uses Latin rectangles to construct a distributed scheduling function ensuring that a node has a different channel allocation pattern compared to his neighbors within a timeslot which avoids inter-nodes interference.	6TiSCH simulator			✓	✓	
Sinche et al. [30]	2019	CoMI is evaluated alongside other management protocols, with observations that "CoMI has still not taken off, while LwM2M is rapidly gaining ground."	-					✓
Cena et al. [31]	2020	The benefit of interference mitigation by classical channel-hopping is investigated.	OpenMote B running OpenWSN			✓	✓	
Daneels et al. [32]	2021	Each node chooses a preferred PHY and parent. Slot bonding combines different PHYs with different data rates in a TSCH schedule.	Zolertia RE Mote platform (revision B)	✓				
Moons et al. [33]	2021	SCHC [34] is used to compress IPv6 and upper-layer protocol headers. An SCHC registration mechanism at server nodes is evaluated within a LwM2M environment. CORECONF is acknowledged as a potential alternative.	Matlab					✓
Gaitán et al. [35]	2022	The formation of clusters around multiple gateways satisfies real-time requirements.	Synthetic generation of network graphs	✓	✓			
Kalita et al. [36]	2023	The EPS scheme allows nodes to monitor their parents' buffer occupancy via information embedded in EBs. When a node detects high buffer usage or repeated transmission failures, it proactively switches to a less congested parent from its RPL parent set.	FIT IoT-Lab platform with Contiki NG	✓				
Gudi [37]	2023	SenML [38], a data format for sensor measurements and device parameters, is natively supported by LwM2M. The existing SenML LwM2M model is mapped to a CORECONF-compliant YANG model. As part of this effort, the *C-CORECONF* library [39] has been developed and subsequently integrated into RIOT OS [40].	-					✓
Tabouche et al. [41]	2024	The RPL-LQ algorithm proposes RPL parent selection based on link quality.	Cooja Simulator	✓				
Koesnadi et al. [42]	2024	Cell allocation in 6TiSCH networks is optimized based on node density and spatial distribution using K-Means clustering to prioritizes traffic inside a cluster.	6TiSCH simulator		✓		✓	
Tabouche et al. [43]	2024	The RPL routing path is influenced by traffic load, with each node independently selecting its preferred parent in a decentralized manner.	Cooja Simulator, FIT IoT-Lab platform with Contiki NG	✓				
Mohamadi et al. [44]	2024	Determinism in 6TiSCH networks is achieved by a CORECONF interface that enables installation and management of "tracks".	OpenMote B running OpenWSN				✓	✓
Gaitán et al. [45]	2024	Nodes are chosen as gateways to cause minimal overlap between active flows in network. This clustering-aided multi-gateway designation satisfies real-time requirements.	UUniFast-like simulation	✓	✓			

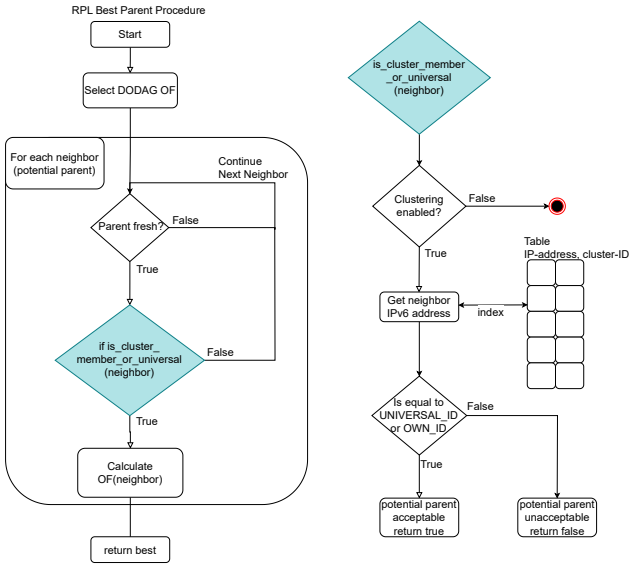

Figure 4: RPL preferred parent selection in a clustered 6TiSCH network.

out immediately, maintaining network robustness since all nodes retain the same neighborhood knowledge as they would in non-clustered networks.

Fig. 4 illustrates the integration of cluster-awareness into the RPL parent selection mechanism while preserving standard RPL behavior, including the configured Objective Function (OF) and its decision logic within clusters. The left side of the figure presents the default preferred parent selection in RPL-classic; the right side shows our cluster-aware modifications. The colored block highlights the adapted sub-procedure. Each node receives a cluster ID from the network manager, used to prioritize intra-cluster communication. When clustering is enabled, the function checks whether a neighbor's cluster ID matches the node's own or the universal ID, using a maintained mapping of IP addresses to cluster IDs. Only such neighbors are considered valid candidates for best parent selection. Crucially, the core OF logic remains unchanged. Our extension acts as a pre-filter to enforce cluster boundaries, allowing RPL's standard evaluation, based on rank and link quality, to proceed as usual. This modular design ensures that cluster-specific policies can be applied with minimal disruption to existing RPL configurations.

To prevent node isolation caused by the exclusion of all potential parents, e.g. due to overly restrictive cluster assignments, we introduce a fallback mechanism. If the list of acceptable parents is empty after applying the cluster-aware filter, the node re-executes the procedure shown in Fig. 4, this time bypassing the cluster-aware decision block. This ensures that nodes are not unintentionally isolated due to suboptimal cluster configurations.

B. Channel Hopping Sequence Adaptation

The second mechanism enhances TSCH channel-hopping by assigning different channel-set IDs to specific nodes or

Figure 5: Channel hopping sequence adaptation mechanism triggered at a specific target-ASN.

clusters, allowing for targeted frequency adaptation without disrupting the entire network. In standard TSCH implementations, all nodes follow a common channel-hopping sequence, which helps mitigate interference and multi-path fading. However, in scenarios where certain nodes experience persistent interference or require higher reliability, a more flexible frequency allocation can improve performance.

To achieve synchronized transitions, the network management entity issues a CORECONF iPATCH request specifying both a channel-set ID and a target-ASN. Upon receiving this request, the node updates its local YANG datastore and notifies its neighbors about the scheduled change. This ensures that all affected nodes switch to the new channel-hopping sequence simultaneously at the designated ASN. Fig. 5 illustrates this transition process.

This mechanism is fully compatible with existing TSCH scheduling strategies since it does not alter how slotframes or 6TiSCH Operation Sublayer 6top Protocol (6P) operations function. Instead, each node references its assigned channel-set ID rather than a globally shared set of frequencies. By modifying the default TSCH channel selection formula to use a subset $\mathcal{C}_{new}$ of available frequencies, interference-prone channels can be excluded, reducing packet loss and improving reliability:

$$\text{Channel}_{\text{freq}} = \mathcal{C}_{new}\Big((\text{ASN} + \text{ChannelOffset}) \bmod |\mathcal{C}_{new}|\Big), \tag{1}$$

where $|\mathcal{C}_{new}|$ is the size of the adapted channel set, and $\mathcal{C}_{new}(k)$ returns the k-th channel in that set. Thus, any channels $c \in \mathcal{C},\ c \notin \mathcal{C}_{new}$ are excluded from the hopping mechanism.

In more advanced scenarios, multiple channel sets $\mathcal{C}_0, \ldots, \mathcal{C}_{N_{\text{Sets}}-1}$ can be defined to create logical or physical separation between clusters. These channel sets are designed to be mutually exclusive, ensuring that no frequency overlap occurs. First, each node determines which channel-set ID i to use based on the current ASN:

$$i = \text{ASN} \bmod N_{\text{Sets}}, \tag{2}$$

then picks the channel within $\mathcal{C}_i$:

$$\text{Channel}_{\text{freq}} = \mathcal{C}_i\big((\text{ASN} + \text{ChannelOffset}) \bmod |\mathcal{C}_i|\big). \tag{3}$$

This ensures that different clusters never transmit on the same frequency at the same ASN, even if they use the same overall range of N_{Channels}. Unlike static channel assignments, which could expose clusters to interference vulnerabilities,

the rotating channel-set mechanism maintains reliability by dynamically adapting to changing conditions. This approach preserves the robustness of 6TiSCH networks while providing resilience against interference and improving network efficiency.

Allowing different frequency-hopping sequences within a single 6TiSCH network presents challenges, particularly in enabling communication between nodes operating on different channel-set IDs. To address this, nodes must be aware of their neighbors' respective channel-set IDs. Given this requirement, a straightforward method can be defined to ensure agreement on which channel to use for communication:

During *receptions*, a node computes the expected channel for a given slot based on its own channel-set ID. When *transmitting* to a specific neighbor (e.g. its parent), the node temporarily adopts *the neighbor's channel-set* to determine the channel, ensuring that both sides remain on the same frequency. The network management entity must ensure that all nodes are informed of their neighbors' channel-set IDs.

As a result, no significant scheduling modifications (e.g. in 6P or RPL) are required. Instead, nodes simply adjust how they index into the channel set during transmission.

This procedure is illustrated in Fig. 6. At the start of each slot, the node first checks whether channel-hopping adaptation is enabled. If adaptation is disabled or the current slot is not a unicast slot, the node uses the default TSCH channel computation. Otherwise, it distinguishes between receive and transmit modes. In receive mode, the node selects its channel from its own channel-hopping sequence, based on a locally stored channel-set ID. In transmit mode, it uses the neighbor's channel-set ID, retrieved from an IP-to-channel-set mapping, to compute the channel. This mechanism supports interoperability across clusters with different channel-hopping sequences as shown in Fig. 7. In our case, all available channel-hopping sequences are compiled into the firmware at build time. This design allows defining multiple sequences without incurring any runtime overhead or requiring additional network traffic. That being said, there is still the possibility to put new channel hopping sequences into the YANG datastore if necessary.

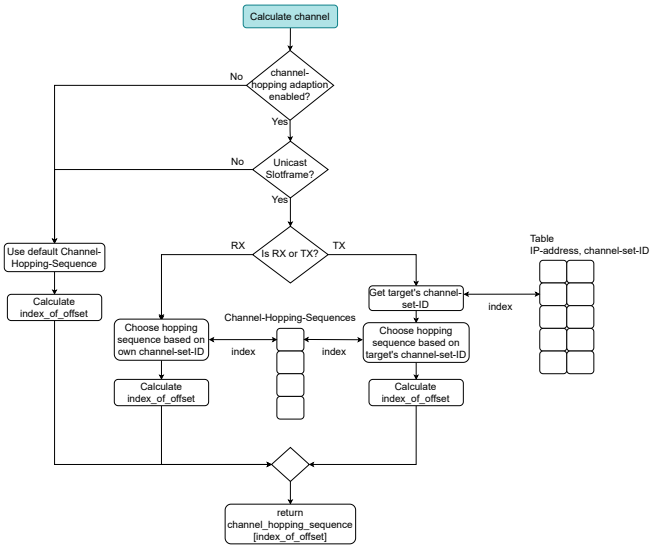

Figure 6: Modified channel calculation procedure considering cluster based channel-sets.

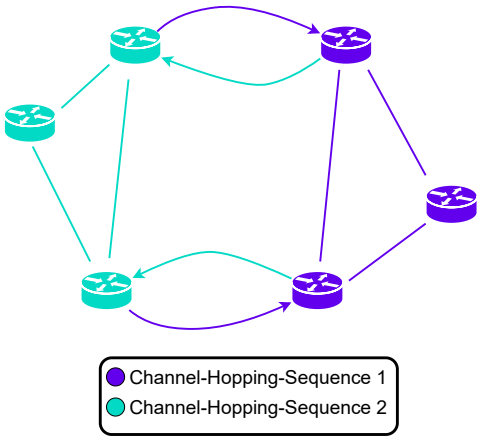

Figure 7: Communication across cluster boundaries by adopting the channel-set of the target neighbor.

C. Synchronized Roll-Out Mechanism

Both, clustering-based routing modifications and channel-set adjustments depend on precise synchronization to be effective. The runtime management system ensures this by linking each parameter update to a specific ASN, guaranteeing that all neighbors of the targeted node to adapt their configurations at exactly the same slot boundary. This synchronization is essential for maintaining consistency, particularly in scenarios involving time-sensitive traffic flows or mission-critical control loops.

Fig. 8 illustrates the synchronization mechanism. Each targeted node must inform all of its neighbors, not just parent and child nodes, about the received update and the target-ASN at which it will take effect. This ensures robustness in RPL and TSCH, even when a DODAG rebuild is triggered or when a group of nodes temporarily loses and then regains network connectivity. The roll-out is initiated by the central manager and carried out sequentially on a per-node basis to maintain consistency and ensure uninterrupted network operation throughout the update process.

V. System Architecture

By combining clustering-based routing with adaptive channel-hopping, the approach meets a wide range of industrial requirements, from ensuring high reliability in challenging environments to enforcing strict latency constraints and traffic isolation. This section delves into the practical realization of these concepts, beginning with an overview of the system architecture and progressing to detailed implementation aspects, including CORECONF, the unified YANG datastore, and the interface layer for synchronized updates.

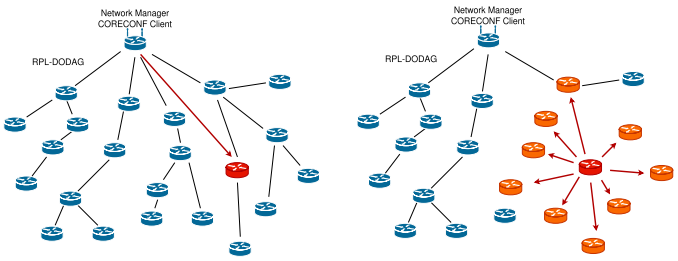

Figure 8: Synchronized Roll-Out Mechanism.

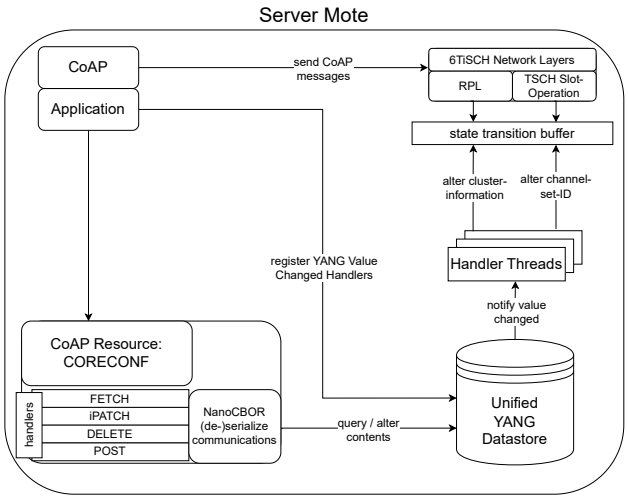

Figure 9: High-level architecture overview of CORECONF server nodes.

A. CORECONF Client-Server Architecture

1) CORECONF Server: The internal architecture of each CORECONF server node is illustrated in Fig. 9, highlighting the key modules responsible for handling management requests, storing configuration values, and applying changes to the RPL and TSCH layers.

The CoAP Layer serves as the entry point for management traffic, exposing CORECONF methods such as iPATCH, FETCH, DELETE, and POST through a CoAP resource. These operations enable external clients to update and retrieve configuration parameters. All clustering and channel-hopping parameters, along with other node-specific settings, are stored in a unified YANG-based datastore, which ensures consistency and type correctness by validating changes against predefined models. When a value change is registered, handler threads manage inter-node communication (e.g. notifying neighbors of a new cluster-ID) and coordinate update timing. Pending parameter changes, along with their respective target-ASN, are temporarily stored in the state transition buffer until the specified activation time is reached. The 6TiSCH network stack, including RPL for routing and TSCH for channel-hopping, queries the state transition buffer as needed. At the designated target-ASN, the updated parameters are seamlessly applied without disrupting ongoing slot operations. This modular architecture ensures a clear separation of concerns: the CORECONF resource handles management traffic, the YANG datastore enforces consistency, and the state transition buffer guarantees synchronized roll-outs. By decoupling timing logic from the core network stacks, the system remains both flexible and resilient. A sufficiently large timespan must be allocated between a configuration request and its scheduled activation to ensure that all participating nodes receive the update. While selecting a later target-ASN introduces some additional latency, it minimizes the risk of inconsistent network states by allowing nodes adequate time to synchronize. Since roll-out procedures are expected to occur infrequently—typically in response to longer-term environmental changes such as external interference—this added delay is generally negligible.

2) Client-Server Channel-Set-ID Assignment: Fig. 10 illustrates the installation process of a new cluster-ID or channel-set-ID using the proposed CORECONF-based management framework. The diagram focuses on a successful update scenario where all parties—the CORECONF client, the CORECONF server, and the server's neighbors—are reachable and acknowledge all messages.

The process begins when the CORECONF client sends an iPATCH request to the server's CoAP resource. This request specifies the new identifier (e.g. cluster-ID or channel-set-ID) along with a target-ASN that defines the exact timeslot for activation. Upon receiving the request, the server validates and updates the parameter in its YANG datastore. If the update is successful, the datastore triggers a value-changed event, signaling the handler thread to proceed. The server then responds to the client, confirming the update or reporting any errors.

Next, the handler thread asynchronously issues iPATCH requests to inform neighboring nodes of the pending change. These requests install a YANG-list entry mapping the node's IP address to its new cluster- or channel-set-ID, along with the target-ASN. The handler then notifies the client whether all neighbors were updated successfully.

At this point, the client evaluates whether enough time remains before the target-ASN is reached. If there is not enough time, the process ends. Otherwise, the client sends a POST request instructing the server to finalize the update.

During this finalization phase, the handler contacts all neighbors again and writes the updated value to a dedicated state transition buffer, along with the target-ASN. The neighbors do the same. The client is then informed of the final status. Only after receiving two separate confirmations all nodes consider the roll-out as active. The update is applied precisely at the target-ASN, integrating the new ID into the TSCH and RPL layers without disrupting ongoing slot operations. Nodes that do not receive a second confirmation will not apply the change, minimizing the risk of inconsistencies. The entire process runs in $\mathcal{O}(1)$ time and space complexity.

Despite the robustness of the two-step confirmation mechanism, a small risk of roll-out inconsistency remains. This may occur if one or more neighbor servers fail to acknowledge the second POST request, leaving the client uncertain whether those nodes have adopted the new channel set at the specified target-ASN. To address this, we implement a simple fallback mechanism. It monitors neighbors that did not send a second

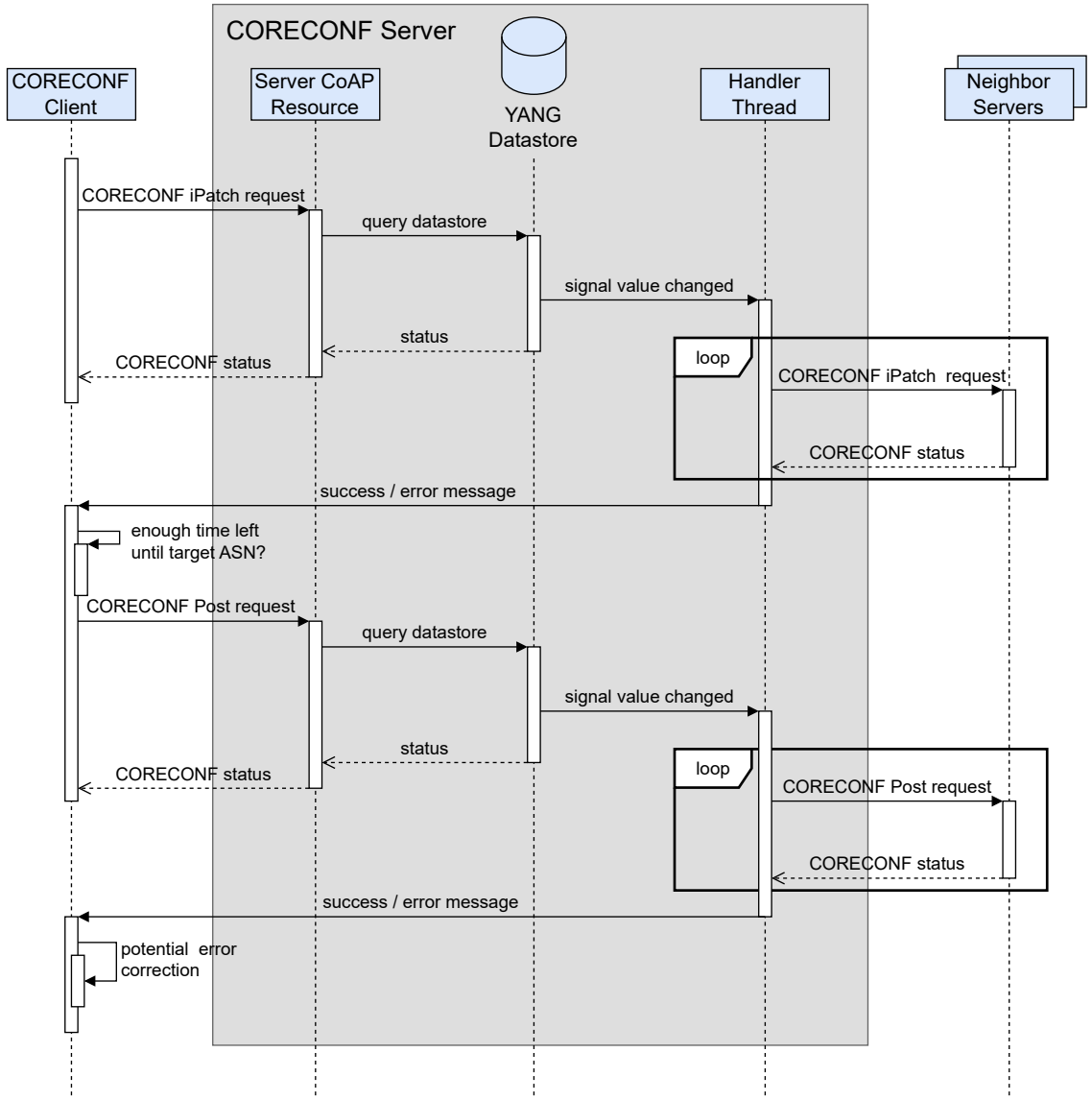

Figure 10: Flow diagram for a successful CORECONF client-server channel-set-ID assignment.

confirmation. Due to the two-step design, a node can only be operating with either the previous or the new channel set after the target-ASN. The fallback first tests communication using the new channel set. If a neighbor is found to be unreachable, the channel set is reverted to its previous value in the YANG datastore, and the assignment process is repeated.

Regarding convergence and complexity, the protocol guarantees consistent roll-out by requiring two explicit acknowledgments from each participating node before the new configuration becomes active. This ensures that all involved nodes synchronize precisely at the defined target-ASN. Convergence time depends primarily on the configured delay between initial update and the target-ASN, which is tunable based on network size and slotframe length to ensure sufficient synchronization margin. The target-ASN must be selected conservatively to ensure that all affected nodes and their neighbors have sufficient time to complete the two-step roll-out procedure, including the acknowledgment of the second confirmation message. This timing is critical to avoid inconsistencies in the network state at the moment of activation. However, determining an optimal target-ASN analytically is infeasible due to the large number of influencing factors, including the current application traffic load, slotframe lengths and scheduling density, the number of neighbors to inform, and the effective PDR across the neighbor links. Some parameters are dynamic and often interdependent, making a closed-form estimation impractical. Instead, we apply a conservative heuristic, initially allocating one minute guard time. If the roll-out fails, this interval is doubled with each subsequent attempt.

B. YANG Data Model

The YANG data model underlying this system is designed to capture all essential parameters for cluster and channel adaptations. It defines the structural layout of the unified YANG datastores used by all nodes in the network. As shown in Table III, the model consists of two main sections: cluster adaptation and channel adaptation. Each section includes an integer identifier, a target-ASN for activation, a list of neighbor entries (containing their IPv6 addresses and associated IDs), a boolean flag to enable or disable the mechanism, and an Remote Procedure Call (RPC) item to trigger the final roll-out process. By structuring the model this way, cluster- and channel-related functions remain logically separate while still being managed through a unified framework. The YANG SIDs shown in the table are manually selected examples.

C. Implementation Details

The described architecture is implemented on top of the Contiki-NG Operating System (OS) [9]. The OS includes an integrated 6TiSCH stack, combining TSCH scheduling with IPv6-based RPL routing. It supports two RPL variants: RPL-Classic, which maintains per-node routing entries for flexible topologies, and RPL-Lite, which simplifies routing by centralizing computations at the root. The optional Orchestra module automates TSCH scheduling based on topology and traffic patterns, eliminating manual slot assignments [17]. Contiki-NG also provides a lightweight CoAP library for application-level messaging, with optional features like block-wise transfers and observe notifications.

The CORECONF server's primary role is to accept incoming CoAP requests, decode their payloads with NanoCBOR [46], and invoke corresponding functions in the YANG datastore. It then returns appropriate success or error codes based on the datastore's response. The unified YANG datastore is designed to adhere to YANG's tree-based modeling paradigm while being lightweight enough to suit resource-constrained devices. It supports a subset of YANG data node types—specifically container, leaf, leaf-list, list, and RPC—each associated with a YANG-SID. Unlike traditional YANG NETCONF implementations, which can dynamically instantiate arbitrary YANG elements, our approach limits node creation to a predefined set of YANG items. This allows only value insertion or modification at runtime, such as appending entries to leaf-lists or updating existing leaves. By separating the datastore into static and dynamic portions, this design ensures that memory usage, code size, and runtime complexity are kept manageable for highly constrained embedded platforms.

To enable efficient node lookups, all static YANG items are stored in a single array, sorted by their SID. This approach allows for binary search to find a node in $\mathcal{O}(\log n)$ time, where n is the total number of static nodes. Given the typical scale of industrial networks, which generally involve tens to hundreds of nodes rather than thousands, this method provides sufficient performance. Once a node is located, its children or siblings are traversed in linear time, $\mathcal{O}(m)$, where m represents the number of descendants to process. Additionally, using integer array indices instead of raw pointers enhances safety by enabling boundary checks, thereby improving overall robustness. Array-based storage also simplifies persistence and re-initialization, as a block copy of the array preserves the entire static relationship among nodes.

VI. Simulation-Based Evaluation

The presented architecture is evaluated for two different use cases. At first, the simulation setup for both use cases is defined and then the obtained simulation results are analyzed and interpreted.

A. Simulation Setup

1) Contiki-NG and 6TiSCH Configuration: All experiments use Contiki-NG with the 6TiSCH stack enabled [9]. Nodes run the TSCH Medium Access Control (MAC) layer for deterministic slot allocation and channel hopping. The native Orchestra scheduler is used throughout, with default slotframe settings listed in Table IV. Orchestra's unicast slotframe length is a key factor, as all analyzed channel-hopping adaptations apply only to unicast traffic. Longer slotframes reduce energy consumption by allowing longer sleep periods but increase E2E latency due to fewer transmission opportunities. Shorter slotframes lower latency and improve throughput but consume more energy as nodes wake up more frequently.

The average end-to-end E2E delay is calculated as the mean time between the generation of a CoAP message at the source node and its successful reception at the DODAG root. Only successfully delivered messages are included in the calculation. Delay values are aggregated over 15 min intervals.

Simulations show that slotframe length significantly impacts E2E delay, particularly under high traffic [47]. Longer slotframes worsen delays by limiting retransmission chances after failed packets. Shorter slotframes increase transmission attempts but allow quicker recovery from failures, mitigating performance degradation. These trade-offs highlight the need for adaptive slotframe configurations based on traffic patterns.

For routing, the network uses Contiki-NG's RPL in storing mode, where each node maintains a routing table. To simplify analysis, RPL operates within a single DODAG. The DODAG Information Object (DIO) interval starts at 2^{12} ms (4.096 s) and doubles with each subsequent interval up to 1048 s, keeping control traffic low. Routes expire individually after 30 min, preventing full topology refreshes. Destination Advertisement Object (DAO) messages propagate routing data upward, ensuring downward connectivity.

Simulated CoAP traffic varies by scenario, with configurable message intervals. Blocking mode ensures reliable delivery by waiting for acknowledgments but increases latency, while non-blocking mode prioritizes speed at the cost of reliability. These settings allow evaluation under diverse traffic patterns and reliability requirements. Table IV summarizes the default simulation parameters.

2) Cooja Network Simulator Configuration: All results are generated using the Cooja network simulator, which compiles Contiki-NG firmware into Java-based emulated nodes. Unlike script-based or discrete-event simulators, Cooja executes the

Table III: YANG-Modeled Resources for Cluster and Channel Adaptation.

YANG Item	Type	SID	Description
`root container`	`container`	4000	Container for all items used by the proposed system.
Cluster-Adaptation Parameters			
`clustering-data`	`container`	4100	Container for clustering relevant items.
`cluster-id`	`leaf/int32`	4101	Identifier determining cluster-membership.
`target-ASN-cluster`	`leaf/int64`	4102	Target-ASN for updates.
`neighbor-cluster-map`	`list`	4103	Holds neighbor IPv6 addresses and their associated cluster IDs.
`cluster-adaption-enabled`	`leaf/bool`	4104	Enables or disables cluster-based adaptation.
`trigger-cluster-update`	`leaf/rpc`	4105	RPC to trigger the rollout of a cluster adaptation update.
Channel-Adaptation Parameters			
`channel-set-data`	`container`	4200	Container for channel-set adaption relevant items.
`channel-set-id`	`leaf/int32`	4201	Identifier for the channel set.
`neighbor-channel-map`	`list`	4202	Holds neighbor IPv6 addresses and their associated channel-set IDs.
`target-ASN-channel-set`	`leaf/int64`	4203	Target-ASN for channel-set updates.
`channel-set-adaption-enabled`	`leaf/bool`	4204	Enables or disables channel-set adaptation.
`trigger-channel-update`	`leaf/rpc`	4205	RPC to trigger the rollout of a channel adaptation update.

Table IV: Simulation parameters for use case evaluation.

Parameter	Value	Description
Slotframe Parameters		
EB	349 timeslots	Broadcast of EBs
Common Shared (CS)	31 timeslots	Routing broadcast traffic
Unicast (UC)	29 timeslots (Use Case 1) 97 timeslots (Use Case 2)	Unicast traffic
Root (R)	13 timeslots (Use Case 1) 23 timeslots (Use Case 2)	Root traffic
RPL Parameters		
DIO Interval	2^{12} ms = 4.096 s	Time before first DIO broadcast
DIO Interval Doublings	8	RPL DIO doublings
Maximum DIO Interval	$4.096\text{ s} \cdot 2^8 \approx 1048$ s	Max DIO interval
Route Lifetime	30 minutes	Time before invalidating a route
CoAP Parameters		
Message Interval	$[0, 10]$ s (Use Case 1) $[0, 60]$ s (Use Case 2)	Frequency of CoAP message generation
Traffic Type	Blocking (Use Case 1) Non-blocking (Use Case 2)	Determines whether messages wait for acknowledgment
Network Parameters		
Number of Nodes	41 (Use Case 1) 45 (Use Case 2)	Total number of nodes in the network

same binaries as real Internet of Things (IoT) hardware, ensuring seamless portability between simulation and deployment with minimal adaptation.

Cooja provides cycle-accurate CPU emulation down to the microsecond level, enabling precise analysis of latency, transmission scheduling, and interference. While it simplifies peripheral abstractions like radio transceivers, it supports scalable simulations, expediting large-scale experiments. This work employs the Unit Disk Graph Medium (UDGM) for radio modeling, which approximates circular transmission and interference ranges. However, since our evaluation primarily focuses on networking mechanisms rather than precise signal propagation effects, we opted for this model to ensure computational efficiency without compromising the validity of our results. The nodes' default transmit power is set to 0 dBm for UDGM. Though less precise than advanced models, UDGM ensures deterministic communication boundaries, simplifying reproducibility in interference studies.

Contiki-NG logs per-slot TSCH activity when `LOG_LEVEL_DBG` or `TSCH_LOG_PER_SLOT` is enabled, capturing ASN, channel offset, physical channel, and transmission status. Logs include source/destination MAC addresses, packet details, and success or failure codes. For unicast transmissions, drift corrections indicate timing adjustments based on received acknowledgments. Python scripts process these logs offline to extract key performance metrics, including transmission attempts, collision-induced failures, and E2E delays.

B. Use Case 1: Mitigation of External Interference

1) Topology: In IIoT environments, Wi-Fi (IEEE 802.11) predominantly operates on overlapping channels in the 2.4 GHz ISM band—notably channels 1 (2412 MHz), 6 (2437 MHz), and 11 (2462 MHz). These channels significantly overlap with IEEE 802.15.4 channels (11–26), leading to potential interference that can degrade communication reliability. To maintain deterministic performance in mission-critical IIoT applications, robust interference-avoidance strategies are essential.

To evaluate the dynamic channel adaptation mechanism (Section IV-B) under realistic interference conditions, we design a simulation scenario in Cooja that emulates Wi-Fi interference patterns.

The network shown in Fig. 11 comprises 45 nodes categorized into three distinct groups.

- 1 CORECONF client node, serving as both network manager and DODAG root.
- 40 CORECONF server nodes, forming a multi-hop mesh network.
- 4 jammer nodes, positioned to transmit random packets with higher transmit power, simulating strong Wi-Fi interference from channel 1, causing noise on IEEE 802.15.4 channels 11–14.

The experimental protocol consists of three phases.

In Phase 1, the network operates without interference, establishing baseline performance metrics. Server nodes transmit blocking CoAP requests with randomized inter-transmission intervals of [0,10] seconds. A small unicast slotframe length combined with high traffic ensures that interference effects are clearly visible. Blocking CoAP requests further highlight the impact on both transmission and reception events.

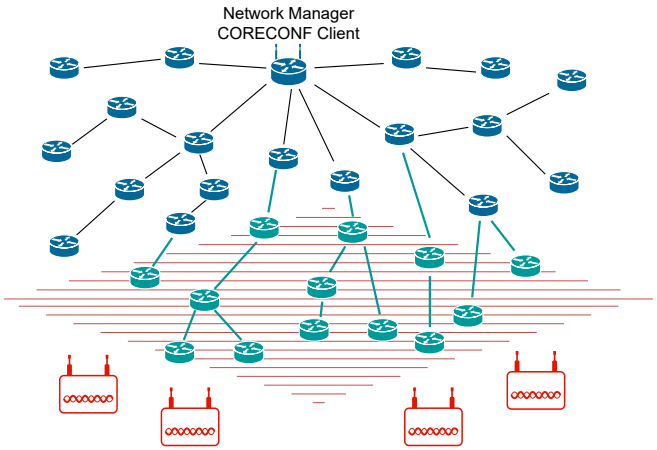

Figure 11: Example network topology including a node subset affected by interfering jammer.

In Phase 2, the jammer nodes activate their radio, introducing sustained interference on IEEE 802.15.4 channels 11–14. Since every successful TSCH transmission receives an Acknowledgment (ACK), failed transmissions are identified by comparing their increase against the baseline. This allows us to quantify the interference impact.

In Phase 3, the network manager detects interference and initiates channel-set adaptation, instructing affected nodes to avoid jammed frequencies.

As illustrated in the heatmap in Fig. 12, transmissions initially follow the default 16-channel hopping scheme but later exclude channels 11–14 once the update is applied. The heatmap clearly highlights this adaptation, as transmissions shift to interference-free frequencies, ensuring network resilience.

2) Results: To evaluate the network's behavior under realistic external interference, we present two complementary perspectives of the same experimental dataset. First, Fig. 13 illustrates failed transmission attempts in 15-minute intervals, split into: (1) the entire network (blue bars) and (2) a subset of nodes within the jammer's interference range (red stacked bars). Second, Fig. 14 shows the total number of transmissions attempted by all nodes (including retransmissions) and the average E2E delay—measured from when a CoAP message leaves its source node until it reaches the DODAG root node. Both figures highlight the three mentioned distinct operational phases:

In Phase 1, during the first 360 min, the network operates without external jamming. In Fig. 13, the failed transmission counts are relatively low for both the network-wide and interference-affected groups, indicating uniform access to all 16 channels. Collisions are minimal, especially among nodes farther from the root. A parallel trend appears in Fig. 14, where the total number of transmissions (blue bars) remains within a moderate range (roughly 5900–6300 transmissions per 15-minute interval), while the average E2E delay (orange line) fluctuates around a stable baseline of approximately 0.38 s. The balanced channel usage and lack of external interference during this period result in relatively efficient data delivery times.

In Phase 2, starting at minute 360, four jammer nodes start transmitting on Wi-Fi channel 1, overlapping with IEEE 802.15.4 channels 11–14, simulating heavy Wi-Fi activity. In Fig. 13, there is a sharp increase in failed transmission attempts, primarily from the subset of nodes within the jammer's range (red portion). Packets transmitted on jammed frequencies consistently fail, and successful transmission occurs only by chance or when the default TSCH mechanism jumps to a frequency outside the affected Wi-Fi channel 1 region. At the same time, Fig. 14 shows a significant rise in the total number of transmissions, climbing from around 6100 to nearly 7000 attempts in some intervals. The network responds to higher interference by triggering more retransmissions, resulting in a higher overall traffic load. Consequently, the average E2E delay spikes sharply, exceeding 0.44 s in some intervals, as nodes queue and retry packets affected by the jamming activity. Despite this increase in retransmissions and delay, the PDR remains above 99%, demonstrating the resilience of 6TiSCH networks.

In Phase 3, starting at minute 420, the CORECONF management system issues a command to exclude the jammed frequencies (channels 11–14) from the hopping sequence of the affected nodes. This adaptation is reflected in Fig. 13, where the number of failed transmissions drops significantly, and the red portion approaches baseline levels. The blue portion increases slightly, indicating a minor uptick in internal collisions due to the reduced channel set (from 16 to 12 channels) for the adapted nodes, which raises the probability of intra-network collisions. In Fig. 14, this adaptation is evident as well, with a decrease in total transmissions from the Phase 2 peak, as fewer retransmissions are required once the jamming frequencies are avoided. The average E2E delay improves, returning to around 0.38–0.39 s, signaling faster delivery of CoAP messages now that high-loss frequencies are no longer used. However, the slight elevation above the original baseline illustrates the trade-off between excluding problematic channels and narrowing overall channel diversity, particularly in dense topologies where fewer frequencies must be shared among nodes.

These figures highlight the substantial impact of external interference on both retransmission rates and E2E delays in 6TiSCH networks. Dynamic runtime adaptation of channel sets helps mitigate much of the jamming-induced disruption, but narrowing the available channel resources can lead to increased internal collisions. The results show a 67% reduction in failed transmission attempts and a slight decrease of over 10% in average E2E delay after adaptation. These improvements are heavily influenced by network configuration, with spatial topology and slotframe lengths being key factors. Longer slotframes result in longer retransmission delays but also reduce intra-network collisions. This interplay between interference avoidance and channel diversity emphasizes the importance of intelligent, standards-based management via CORECONF to maintain reliability in mission-critical IIoT scenarios, especially those sharing spectrum with Wi-Fi deployments.

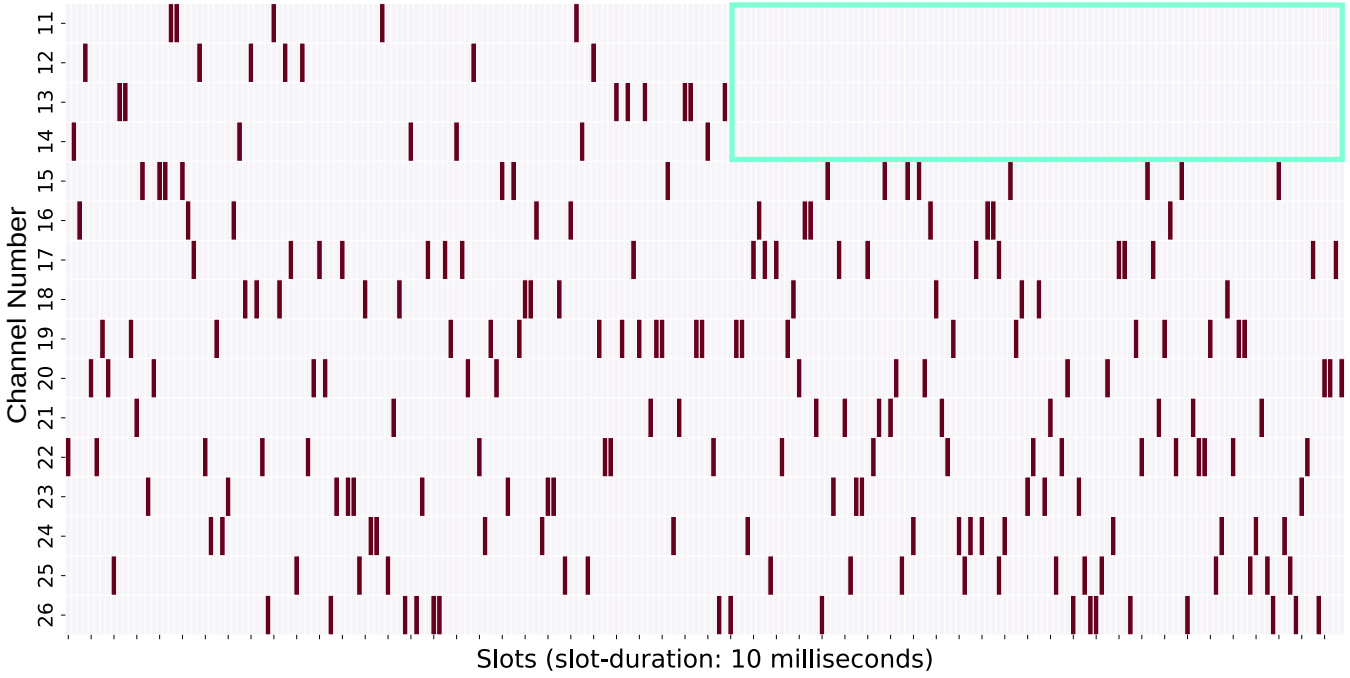

Figure 12: Transmission heatmap for nodes within jammer interference range.

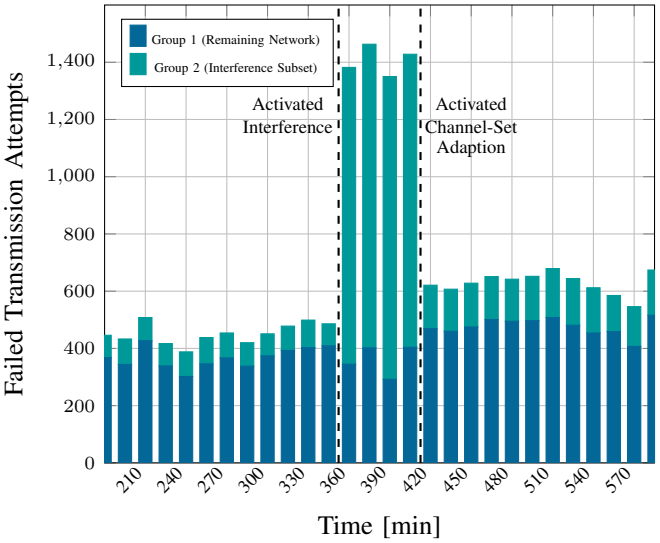

Figure 13: Number of failed transmission attempts before/ after jammer activation and before/ after channel-set adaptation in affected cluster.

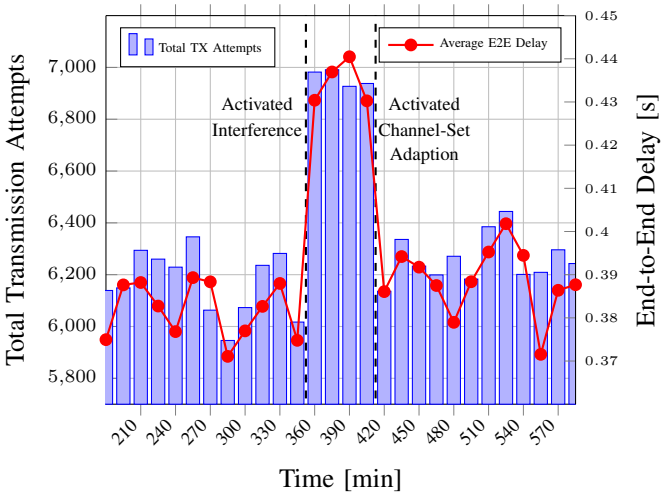

Figure 14: Total transmission attempts and end-to-end delay over time.

C. Use Case 2: Priority Cluster

1) Topology: This use case examines the advantages of configuring a high-priority "cluster" with nodes operating on a distinct routing tree and channel-hopping set. By logically and physically isolating this priority cluster from the main DODAG, the goal is to minimize congestion in critical data paths and ensure low-latency communication for mission-critical traffic. While the absolute length of slotframes is not the primary factor influencing network performance in this scenario, the relationship between traffic volume and unicast slotframe lengths plays a significant role in shaping the network's behavior. Insufficient traffic volume or overly short slotframes may prevent the network from reaching the necessary conditions to effectively validate the proposed mechanism. To account for this, the experimental parameters were manually fine-tuned—such as testing different message intervals and slotframe lengths—and monitoring the resulting queue usage of the TSCH nodes. This allowed us to introduce moderate congestion at nodes near the root, ensuring consistent packet forwarding without reaching saturation thresholds. The configuration in Table IV produces observable forwarding queue utilization, while avoiding complete buffer overflow.

The network shown in Fig. 15 comprises 45 nodes catego-

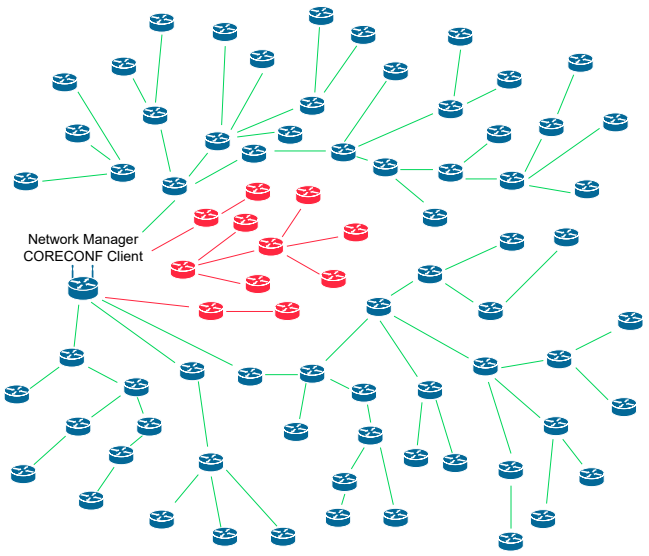

Figure 15: Adaptive RPL parent selection isolates the priority cluster.

rized into three distinct groups.

- 1 CORECONF client node acting as the cluster-management entity and the DODAG root.
- 11 CORECONF server nodes designated as the "priority cluster".
- 33 CORECONF server nodes remain forming the default (non-priority) network.

Initially, all nodes participate in a single RPL instance using a common 16-channel hopping sequence. Each node generates non-blocking CoAP requests at randomized intervals within a [0-60] s range. After the baseline period, the CORECONF client updates the RPL parent-selection parameters for the priority cluster and assigns it an exclusive 8-channel hopping schedule. As a result, non-priority nodes are no longer able to route traffic through the priority cluster, alleviating congestion for those nodes.

2) Results: Fig. 16 illustrates the total number of transmission attempts and the average E2E delay for priority cluster nodes, aggregated into 15-minute intervals.

Two distinct phases emerge from the data. In the first phase (pre-adaptation), until minute 300, priority cluster nodes forward both their own traffic and transit packets from non-priority nodes. As the network load increases, these nodes become congested, leading to a high volume of transmissions and increased queueing delays. The average E2E latency peaks at nearly 1.2 s in certain intervals.

At minute 300, the priority-cluster mechanism is applied, marking the beginning of the second phase (post-adaptation). From this point, priority cluster nodes are logically and physically isolated, preventing non-priority traffic from routing through them. As a result, transmission attempts drop sharply, with a 70% reduction in general transmissions. The average E2E delay also decreases, stabilizing around 0.7 s – roughly 40% lower than before the adaptation. This demonstrates how offloading non-critical traffic alleviates congestion for priority

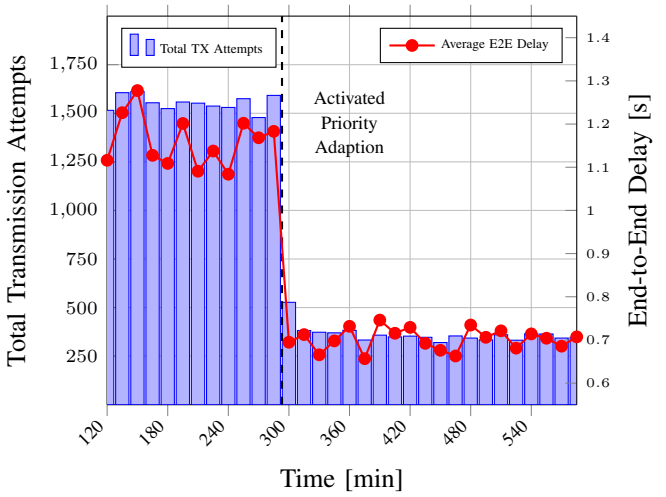

Figure 16: Number of total transmission attempts and average E2E delay in the priority cluster before/ after the priority cluster formation.

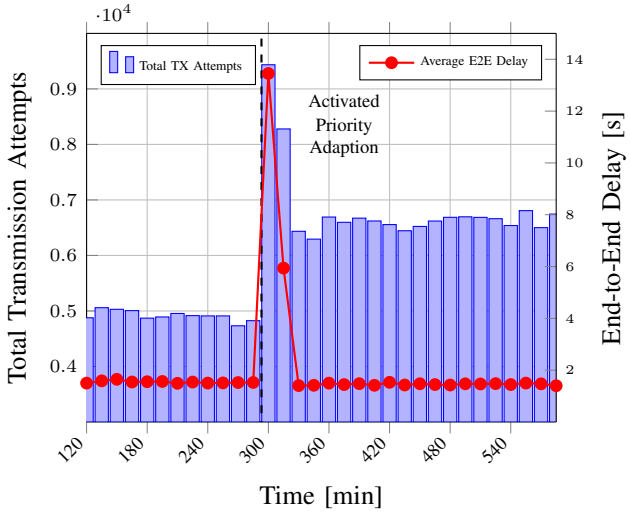

Figure 17: Number of total transmission attempts and average E2E delay in the remaining network before/ after the priority cluster formation.

traffic, confirming the effectiveness of isolating high-priority nodes to maintain lower latency under heavy network load.

Fig. 17 presents similar data for the non-priority nodes.

Once the priority cluster begins isolating, non-priority nodes must select new parents, with the process starting around minute 300 and completed by minute 330. Some nodes initially struggle to find a new parent, resulting in time-outs and lost messages. However, by minute 330, the network stabilizes, and total transmissions among non-priority nodes rise as transit traffic is redistributed. Despite this, E2E delay remains stable—below 2.0 s throughout Phase 2—indicating sufficient routing capacity outside the priority cluster. Certain nodes required up to 30 min to re-establish stable routing, a scenario specific to this simulation. Extensive simulations showed that such disconnections are highly dependent on network topology. In cases where low-priority nodes had multiple parent

options, no disruptions occurred. In more isolated topologies, delays in route re-establishment were observed. To address this, network managers could redefine cluster boundaries or assign the special cluster-ID: 0, allowing affected nodes to join any cluster. This scenario was deliberately chosen to highlight both the benefits of priority clustering and its potential negative effects on the broader network.

The key observations highlight several important aspects of the priority-cluster mechanism. After the cluster adaptation, priority nodes experience a significant reduction in transit traffic, which results in a decrease in E2E delay. This drop in congestion allows for more efficient communication within the priority cluster. Despite rerouting traffic away from the priority cluster, the non-priority nodes in the network exhibit minimal changes in E2E delay, suggesting that there is sufficient capacity and alternate routes available to handle the traffic. Furthermore, by isolating both routing relationships and frequency channels, the system demonstrates scalability, allowing mission-critical nodes to maintain reliable, low-latency communication even under heavy network load.

The size of the prioritized cluster was deliberately set to 11 nodes to represent a realistic subset of critical devices in an industrial deployment. This number ensures a meaningful traffic volume while maintaining a manageable scope for synchronized reconfiguration. To assess the impact of cluster size, we conduct additional simulations with smaller and larger clusters. The results show that smaller clusters yield more pronounced improvements in E2E delay due to reduced transit traffic and lower congestion. In contrast, larger clusters introduce more significant routing disruptions during the transition phase, as more nodes have to reconfigure their paths simultaneously. These findings suggest that the effectiveness of the clustering mechanism depends on a balance between cluster size, traffic load, and the availability of alternative routing paths.

Overall, the priority-cluster mechanism shows how selectively isolating nodes, both in terms of routing and channel hopping, can significantly benefit critical data flows without causing severe disruptions to the broader network. The mechanism proves most effective in situations where priority nodes are congested or face high interference and when there are alternative routes of similar quality available for non-priority traffic. If these conditions are not met, the benefits may not be as substantial, and isolating the priority cluster could even result in negative effects on the non-priority network. This approach is particularly useful in industrial scenarios where guaranteed service quality is essential, as it allows for precise control over high-priority versus best-effort traffic. However, forming priority clusters must be done carefully and under the right conditions to ensure that the benefits outweigh any potential drawbacks.

VII. Experimental Evaluation

Finally, we assess the effectiveness of the proposed clustering approach through experimental evaluation. At first, we describe the hardware and testbed setup, followed by an overview of the 6TiSCH configurations used during the experiment. Subsequently, we analyze the experimental results by examining the power profile of a 6TiSCH node running Contiki NG. This analysis serves to quantify the benefits of the proposed scheme in mitigating external interference.

A. Experimental Setup

1) Hardware: For experimental evaluation we use the nRF52840 DK. The nRF52840 is a versatile, low-power System-on-Chip (SoC) from Nordic Semiconductor, designed for wireless communication in IoT applications. It features a 64 MHz ARM Cortex-M4 processor with Floating-Point Unit (FPU), 1 MB of flash memory, and 256 kB of RAM, offering ample resources for complex embedded tasks. The chip supports multiple wireless protocols, including Bluetooth Low Energy (BLE), IEEE 802.15.4, ANT, and 2.4 GHz proprietary protocols. Its integrated radio and advanced power management make it well-suited for constrained, battery-operated devices. Contiki NG natively supports the nRF52840 as a hardware target, allowing the implementation to be easily deployed by simply switching the compile target from Cooja to nRF52840.

Another valuable feature of the nRF52840 DK is the inclusion of dedicated measurement pins, which allow seamless connection to Nordic Semiconductor's Power Profiler Kit (PPK)-II as illustrated in Fig. 18. The PPK-II is a versatile power measurement tool designed specifically for analyzing the energy consumption of embedded devices. It offers high-resolution current measurements across a wide dynamic range—from 200 nA up to 1 A, making it suitable for ultra-low-power as well as active radio operation analysis. With a sampling rate of up to 100 kHz, the PPK-II can capture fine-grained power consumption patterns, making it ideal for evaluating the radio utilization and sleep phase durations. The PPK-II can operate in different modes. In ampere-meter mode, the Device Under Test (DUT) is powered externally (for instance via USB) and in source-meter mode the DUT is powered by the measurement pins and the supply voltage can be adjusted via the monitoring application. On the nRF52840 DK, we first need to cut a solder bridge, which normally connects the DK's power supply to the nRF52840 chip. By cutting this bridge, the current path is rerouted through the measurement pins, allowing the PPK-II to measure the current drawn by the SoC exclusively.

In addition to its current measurement capabilities, the PPK-II also features an integrated logic analyzer that can be connected to the device's General Purpose Input/Output (GPIO) pins. By toggling a GPIO pin at specific points in the software, it is possible to precisely mark the start and end of operations, enabling accurate correlation between software events and power consumption.

2) Testbed: The testbed environment is set up in an office space where external interference can be precisely controlled. The network topology, illustrated in Fig. 19, features numerous walls and doors that the wireless signal must traverse, introducing realistic propagation challenges. The network consists of 10 nodes in total, including the network manager. To generate external interference, a Wi-Fi router is placed in the

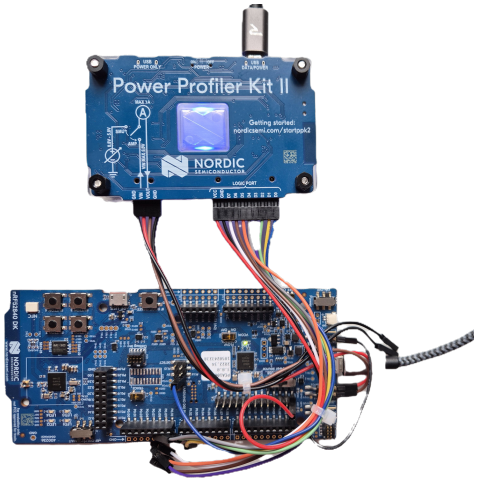

Figure 18: Measurement setup with PPK-II and nRF52840 DK.

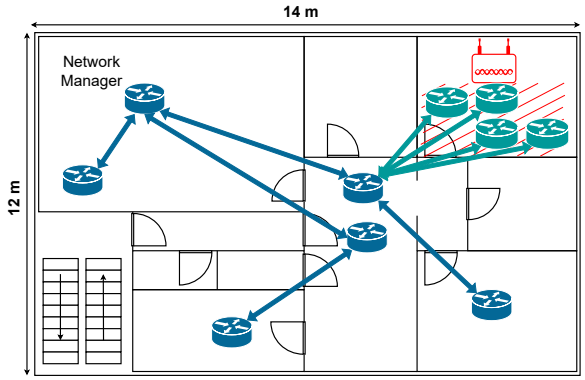

Figure 19: Testbed topology and preferred parent relations.

upper right room. It is configured to operate at maximum transmit power on IEEE 802.11 channel 1, which overlaps with IEEE 802.15.4 channels 11 to 14. As shown in Fig. 20, four nRF52840 nodes are positioned directly adjacent to the Wi-Fi router. This scenario is not merely artificial. It reflects a realistic situation, e.g. a machine in a factory equipped with a Wi-Fi transmitter, while being monitored by sensors mounted directly on the device. Once activated, the Wi-Fi router generates significant interference, particularly due to the high traffic load induced by a concurrently running streaming application. This setup ensures that these four nodes experience substantial external interference. All testbed nodes are powered via USB, and their serial output is logged for subsequent analysis.

The DODAG routing topology can be reconstructed from the collected logs, with each node's preferred parent indicating the uplink path that data packets follow toward the network manager. Prior to the channel-set adaptation, no clusters are formed, and RPL relies solely on the calculated rank to determine parent selection. The nodes that will later form the cluster affected by Wi-Fi interference initially have a hop-depth of 2, which means that they communicate with the manager via one intermediate node.

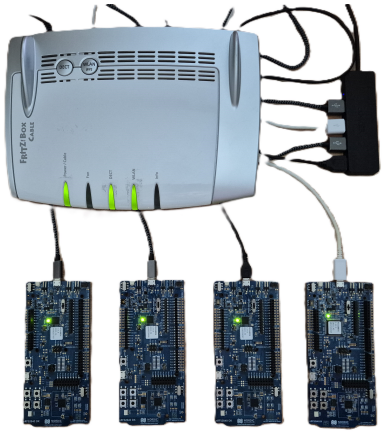

Figure 20: nRF52840 DKs arranged as 6TiSCH nodes exposed to external interference.

Table V: Parameters for experimental evaluation.

Parameter	Value	Description
Radio Parameters		
Transmit Power	-12 dBm	TX Power of IEEE 802.15.4 radio
Receiver Sensitivity	-100 dBm	RX Sensitivity of nRF52840
Slotframe Parameters		
Enhanced Beacon	397 timeslots	Broadcast of EBs
Common Shared (CS)	31 timeslots	Routing broadcast traffic
Unicast (UC)	17 timeslots	Unicast traffic
Root (R)	7 timeslots	Root traffic
RPL Parameters		
DIO Interval	2^{12} ms = 4.096 s	Time before first DIO broadcast
DIO Interval Doublings	8	RPL DIO doublings
Maximum DIO Interval	$4.096\text{ s} \cdot 2^8 \approx 1048$ s	Max DIO interval
Route Lifetime	30 minutes	Time before invalidating a route
CoAP Parameters		
Message Interval	5 s	Frequency of CoAP message generation
Traffic Type	Non-Blocking	Determines whether messages wait for acknowledgment
Network Parameters		
Number of Nodes	10	Total number of nodes in testbed

3) Contiki-NG and 6TiSCH Configuration: The configuration parameters used in the experiment are summarized in Table V. The nRF52840 allows the transmit power to be adjusted between -40 dBm and +8 dBm. While the default setting is 0 dBm, we deliberately reduced it to -12 dBm for the experiment. This adjustment was made to encourage the formation of a mesh topology. At 0 dBm, most nodes would establish direct links to the manager, resulting in a star topology. By lowering the transmit power, we ensure that the network includes at least one intermediate hop, allowing us to properly evaluate the effectiveness of our scheme under multi-hop conditions. The receiver sensitivity is -100 dBm for IEEE 802.15.4 mode.

As in the simulation, the Orchestra scheduler is used. We stick to its default slotframe parameters, and the slot length is set to 10 ms. To generate network traffic, a CoAP application is deployed on each node, transmitting messages at a fixed interval of 5 s.

B. Experimental Results

1) Understanding the Power Profile: We begin the analysis of the results with a brief primer on how to interpret the

measured power profile. We also remark that our application is built on Contiki NG's `examples/6tisch/simple-node` and may therefore serve as common reference for nRF52840 nodes running the Contiki NG 6TiSCH stack.

Prior to joining the network, the node keeps its radio continuously active, listening for incoming EBs frames. In the power profile, this state is characterized by a high current draw of around 16 mA with downward spikes each 10 ms. These spikes represent channel switch events, following the channel hopping sequence defined in the software. In our setup, the node initially cycles through the full IEEE 802.15.4 channel set, spanning channels 11 to 26.

Once the node has successfully joined the network, its average current consumption drops significantly to approximately 0.9 mA. The corresponding current profile is shown in Fig. 21, which displays a series of periodic peaks. For clarity, the figure excludes additional CoAP traffic. These periodic peaks are determined by the Orchestra slotframe configuration. By toggling a GPIO pin at the start of each radio transmission and correlating this with the node's log output, we can easily identify the operations related to EBs. Following the emission of an EB, two types of periodic peaks emerge: one spaced at 310 ms and another at 170 ms. Each of these peaks has a duration of about 4 ms, during which the radio is active in receive mode. These intervals directly correspond to the slotframe settings defined in Table V. Shared slots occur every 31 timeslots (310 ms), and unicast slots every 17 timeslots (170 ms). During these slots, the node is able to receive unicast messages from its neighbors. It is important to note that, although the duration of the EB slotframe is fixed (3.97 s), the transmission of EBs is not guaranteed in every slotframe. In practice, we observe that EBs frames are sent at intervals that are integer multiples of the defined slotframe length, such as at 11.91 s or 15.88 s.

The frequency of radio wake-up operations has a significant impact on the overall energy consumption of a node. Consequently, slotframe lengths should be carefully selected based on the expected traffic patterns of the application. In addition to periodic receive operations, the application traffic itself is a major contributor to energy usage. Therefore, minimizing packet length and avoiding retransmissions are critical for energy efficiency. Fig. 22 provides a detailed view of a CoAP packet transmission at a transmit power of 0 dBm. At the beginning of the slot, the radio wakes up, causing the current draw to rise to approximately 12 mA. The transmission then begins and lasts for about 3.31 ms, during which the current peaks at 15.9 mA. After the transmission, the current briefly returns to around 12 mA before rising again to approximately 16 mA, indicating the reception of the ACK. Following this, the current oscillates around 5 mA due to serial console output activity, before the processor returns to sleep mode.

The duration of the transmit operation observed in Fig. 22 closely aligns with the analytically calculated value. The 2.4 GHz Offset Quadrature Phase-Shift Keying (O-QPSK) PHY of IEEE 802.15.4 provides a data rate $R = 250$ kbps, which is defined as

$$R = \frac{L}{t}.$$

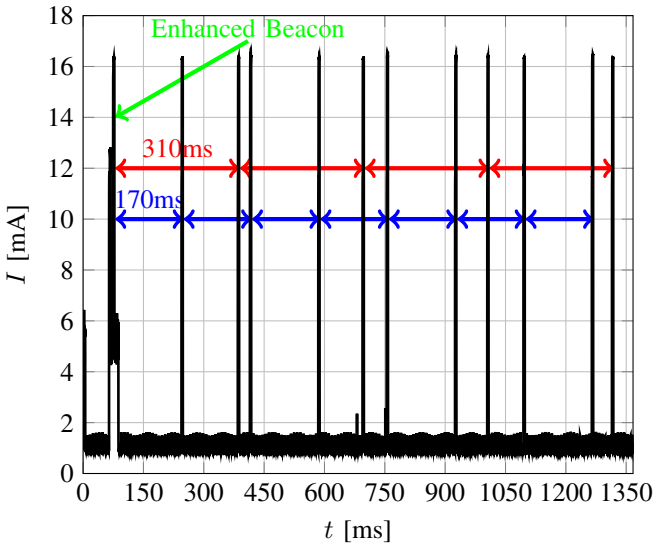

Figure 21: Current draw of a nRF52840 running Contiki-NG as 6TiSCH root node with Orchestra Scheduler and default slotframe periods.

From the log data, we identified that the MAC Protocol Data Unit (MPDU) is 95 B long. Including the PHY overhead, consisting of the preamble, Start Frame Delimiter (SFD), and PHY header, an additional 6 B are added, resulting in a total packet size of $L = 101$ B. This corresponds to 808 bits transmitted over the air. At a rate of 250 kbps, the theoretical transmission time t is approximately 3.23 ms. The measured duration of 3.31 ms differs by only 80 μs. Similar levels of deviation are observed for other packet lengths as well, confirming the consistency of the transmission behavior.

Finally, we analyze the current consumption at various transmit power levels, as shown in Fig. 23. Results indicate that the radio is most energy-efficient at 0 dBm, which represents a sweet spot in the power-current curve. Reducing the transmit power below this point yields only marginal energy savings, while significantly compromising communication range. Although the choice of -12 dBm in our experiment is clearly suboptimal from a range perspective, it was intentionally selected to enforce a mesh topology even within a confined test environment. In contrast, increasing the transmit power beyond 0 dBm leads to a steep rise in current consumption. For instance, at 8 dBm, the current draw during transmission more than doubles to 37.7 mA, making it unsuitable for low-power, battery-operated devices. For the subsequent energy analysis, we calculate the total charge consumed during a complete transmit operation—from radio ramp-up to the end of ACK reception. At 0 dBm, the charge consumed is 78 μC, while at -12 dBm it drops slightly to 70 μC. This minor difference underscores the limited benefits of reducing transmit power below 0 dBm in terms of energy efficiency.

2) Mitigation of External Interference: Fig. 24 shows the number of retransmissions across all network links, recorded in 15-min intervals. Prior to the activation of interference, the retransmission count remains consistently low. Based on the

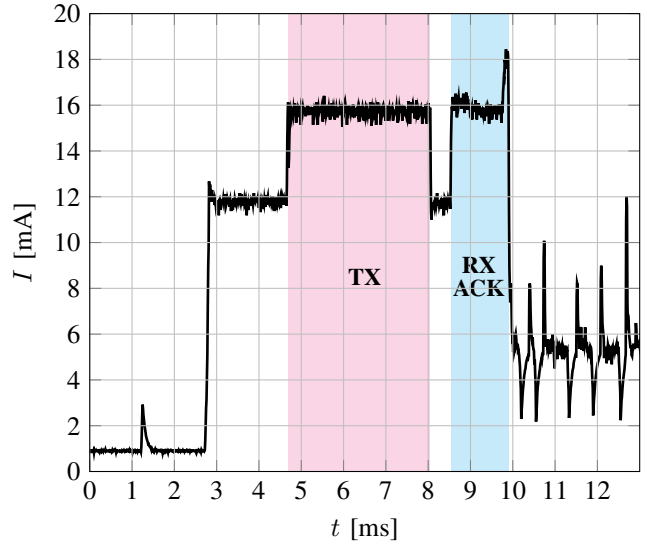

Figure 22: Current draw of a nRF52840 running Contiki-NG and transmitting a CoAP packet (101 Bytes) at 0 dBm.

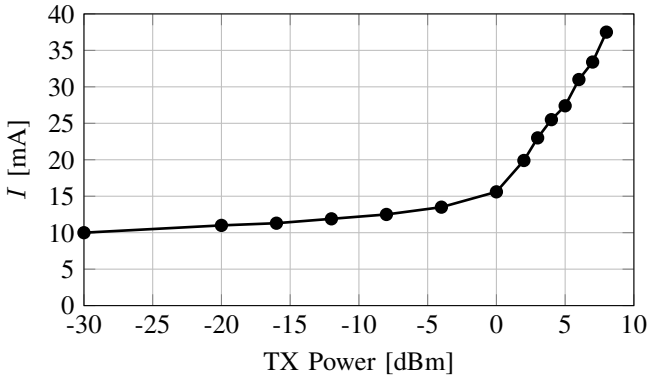

Figure 23: Current draw of the nRF52840 during transmit operation in Contiki NG for different power levels.

topology in Fig. 19, the network consists of one manager, four nodes located in the interference-affected area, and five nodes in the unaffected portion of the network. During this initial phase, no noticeable differences in retransmission rates are observed between the two groups.

After the activation of interference, the number of failed transmission attempts increases sharply. This increase is caused by the disruptive nature of Wi-Fi traffic, which introduces error bursts [48], [49] that interfere with IEEE 802.15.4 reception. These error bursts lead the demodulator to misinterpret symbols, resulting in failed packet decoding. Interestingly, this trend closely mirrors the simulation results presented in Section VI, and aligns well with the packet loss rates reported in [50]. A clear distinction between the two node groups becomes apparent: for the subset exposed to interference near the Wi-Fi router, the number of retransmissions increases by a factor of approximately four compared to the pre-interference phase. However, even nodes in the unaffected portion of the network experience a noticeable rise in retransmissions, as the Wi-Fi signal operates at higher power and therefore can still penetrate all rooms of the testbed environment.

Once the interference-affected subset forms a cluster that operates on a modified channel hopping sequence, which excludes IEEE 802.15.4 channels 11 to 14, the number of retransmissions within this group drops significantly. This channel set adaptation ensures that nodes within the cluster, as well as their parent node, communicate exclusively on the interference-free channels. As a result, retransmission rates return to levels observed prior to the activation of interference. The network performance remains stable, as the Wi-Fi traffic continues to operate on IEEE 802.11 channel 1, which overlaps with the excluded channels. In practice, Wi-Fi routers rarely change channels automatically, further supporting the robustness of this approach. Nonetheless, the CORECONF interface enables dynamic reconfiguration at runtime, allowing the network to respond quickly should interference patterns change.

Lastly, we evaluate the impact of the channel set adaptation from an energy consumption perspective. As shown in the previous section, retransmissions are a major contributor to increased power consumption, ultimately reducing the network's operational lifetime. In our testbed setup, we measured that a single retransmission consumes approximately 70 μC. This value remains consistent, as the CoAP traffic in the network consistently generates packets of about 100 B. Fig. 25 illustrates the accumulated charge consumption due to retransmissions across the entire network. During the first hour, when no external interference is present, the total charge drawn by all nodes amounts to 22 mC. However, once interference is introduced, the slope of the curve increases sharply, reflecting a significant rise in energy usage. After activating the channel set adaptation within the interference-affected cluster, the curve begins to flatten, indicating a return to the lower energy consumption rate observed before interference. This highlights the effectiveness of the adaptation mechanism in restoring energy efficiency under challenging conditions.

VIII. Conclusion

This work introduces a novel application performance management system for 6TiSCH networks using CORECONF. The research contributes to enhancing reliability and performance of wireless IIoT systems by changing configuration parameters during runtime without the need to re-install the system.

We therefore created a resource-efficient CORECONF implementation optimized for embedded systems, addressing the need for standardized management in IIoT networks while adhering to IETF standards.

Furthermore, we designed a management interface that bridges CORECONF with 6TiSCH operations, allowing dynamic configuration of network parameters, such as RPL parent-selection and TSCH channel-hopping sequences. This enables granular control at both the node and cluster levels, with synchronized changes across the network.

We explore two use cases: dynamic channel-set adaptation to face external interference and establishing priority-based transmission clusters. These are validated through Cooja simulations, demonstrating successful interference mitigation, reduced collisions, and improved E2E delay for priority clusters.

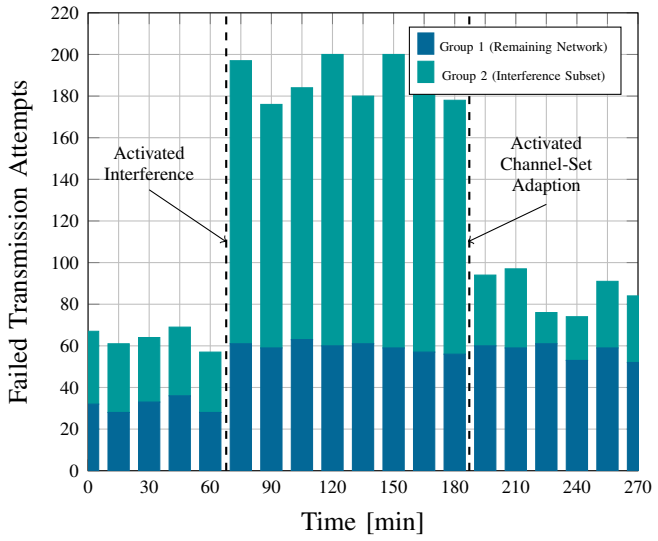

Figure 24: Experimental evaluation by number of failed transmission attempts before/ after Wi-Fi activation and before/ after channel-set adaptation in affected cluster.

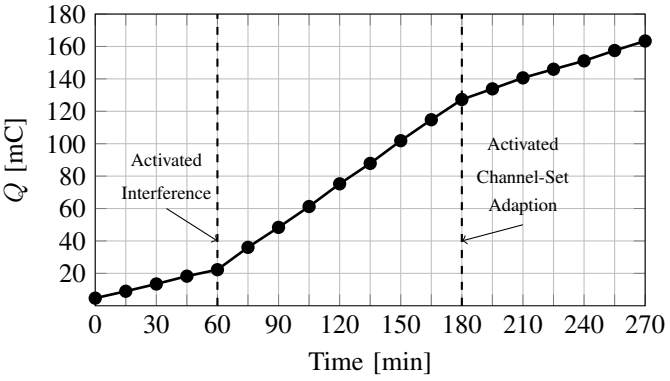

Figure 25: Charge consumption due to retransmissions accumulated from the whole network.

Finally, the external interference scenario is evaluated in a real-world testbed. The results closely align with those from the simulation, showing a substantial reduction in retransmissions within the cluster exposed to Wi-Fi interference. To quantify the energy impact, the power profile of an nRF52840 running the Contiki NG 6TiSCH stack is analyzed, highlighting the cost of retransmissions in terms of charge consumption. Deploying the channel-set adaptation within the affected cluster reduces the overall network's charge draw from approximately 50 mC per hour to 23 mC per hour, significantly extending battery lifetime.

Future work could explore more sophisticated cluster formation algorithms. While priority clusters may be defined by the specific goals of a given user application, dynamically forming clusters in response to external environmental conditions remains an open research challenge. In this work, we demonstrate adaptation to external interference, but similar techniques could be applied to other unpredictable factors by tuning network parameters at the cluster level. For example, clustering based on traffic load is another compelling use case. Our CORECONF implementation provides a flexible foundation for such extensions, particularly through enhancements to the existing YANG model.

ACKNOWLEDGMENT

This document is issued within the frame and for the purpose of the OpenSwarm project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101093046. Views and opinions expressed are however those of the author(s) only and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] M. Hatler, "Wireless Sensor Networks: Expanding opportunities for Industrial IoT," *InTech - ISA's Flagship Publications*, Oct. 2017. [Online]. Available: https://www.isa.org/intech-home/2017/september-october/features/expanding-opportunities-for-industrial-iot

[2] IEEE Computer Society, "IEEE Standard for Information technology–Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," *IEEE Std 802.15.4-2024 (Revision of IEEE Std 802.15.4-2020)*, pp. 1–967, 2024.

[3] P. Thubert, *An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)*, Internet Engineering Task Force (IETF) Std. RFC9030, 2021.

[4] X. Vilajosana, T. Watteyne, M. Vučinić, T. Chang, and K. S. J. Pister, "6TiSCH: Industrial Performance for IPv6 Internet-of-Things Networks," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1153–1165, 2019.

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

[6] M. Veillette, P. Van der Stok, A. Pelov, A. Bierman, and C. Bormann, *CoAP Management Interface (CORECONF)*, Internet Engineering Task Force (IETF) Std. Draft-IETF-CORE-CoMI-19, 2024.

[7] M. Björklund, *The YANG 1.1 Data Modeling Language*, Internet Engineering Task Force (IETF) Std. RFC7950, 2016.

[8] C. Bormann and P. Hoffman, *Concise Binary Object Representation (CBOR)*, Internet Engineering Task Force (IETF) Std. RFC8949, 2020.

[9] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[10] IEEE Computer Society, "IEEE Standard for Information technology–Local and metropolitan area networks– Specific requirements- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY)-Amendment 1: MAC Sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Standard 802.15.4-2011)*, pp. 1–225, 2012.

[11] J. Muñoz, P. Muhlethaler, X. Vilajosana, and T. Watteyne, "Why Channel Hopping Makes Sense, even with IEEE802.15.4 OFDM at 2.4 GHz," in *Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, Nov. 2018, pp. 1–7.

[12] T. Watteyne, A. Mehta, and K. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," in *6th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, Tenerife, Canary Islands, Spain, Oct. 2009, pp. 116–123.

[13] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 595–615, 2020.

[14] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, *6TiSCH Minimal Scheduling Function (MSF)*, Internet Engineering Task Force (IETF) Std. RFC9033, 2021.

[15] A. Tabouche, B. Djamaa, and M. R. Senouci, "Traffic-Aware Reliable Scheduling in TSCH Networks for Industry 4.0: A Systematic Mapping Review," *IEEE Communications Surveys and Tutorials*, vol. 25, no. 4, pp. 2834–2861, 2023.

[16] T. Chang, M. Vučinić, X. V. Guillén, D. Dujovne, and T. Watteyne, "6TiSCH minimal scheduling function: Performance evaluation," *Internet Technology Letters*, vol. 3, no. 4, 2020.
[17] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Seoul, South Korea, 11 2015.
[18] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, *Network Configuration Protocol (NETCONF)*, Internet Engineering Task Force (IETF) Std. RFC6241, 2011.
[19] A. Bierman, M. Björklund, and K. Watsen, *RESTCONF Protocol*, Internet Engineering Task Force (IETF) Std. RFC8040, 2017.
[20] M. G. Bhat, S. Bhattacharjee, C. Gündoğan, K. Alexandris, and A. Gogolev, "CORECONF, NETCONF, and RESTCONF: Benchmarking Network Orchestration in Constrained IIoT Devices," *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 13 082–13 090, 2024.
[21] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, *Object Security for Constrained RESTful Environments (OSCORE)*, Internet Engineering Task Force (IETF) Std. RFC8613, 2019.
[22] E. Rescorla and N. Modadugu, *Datagram Transport Layer Security (DTLS) Version 1.2*, Internet Engineering Task Force (IETF) Std. RFC6347, 2012.
[23] M. Veillette, I. Petrov, A. Pelov, C. Bormann, and M. Richardson, *Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)*, Internet Engineering Task Force (IETF) Std. RFC9254, 2022.
[24] A. Elsts, X. Fafoutis, R. Piechocki, and I. Craddock, "Adaptive channel selection in IEEE 802.15.4 TSCH networks," in *Global Internet of Things Summit (GIoTS)*, 2017, pp. 1–6.
[25] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Label: Link-based adaptive blacklisting technique for 6tisch wireless industrial networks," in *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, Miami Beach, USA, 2017, pp. 25–33.
[26] A. Mavromatis, G. Z. Papadopoulos, X. Fafoutis, A. Goulianos, G. Oikonomou, P. Chatzimisios, and T. Tryfonas, "Link quality and path based clustering in IEEE 802.15. 4-2015 TSCH networks," in *22nd IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, 2017, pp. 798–803.
[27] A. Karaagac, I. Moerman, and J. Hoebeke, "Hybrid schedule management in 6TiSCH networks: The coexistence of determinism and flexibility," *IEEE Access*, vol. 6, pp. 33 941–33 952, 2018.
[28] A. Karaagac, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Time-critical communication in 6TiSCH networks," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Barcelona, Spain: IEEE, Apr. 2018, pp. 161–166.
[29] C. Boucetta, B. Nour, H. Moungla, and L. Lahlou, "An IoT scheduling and interference mitigation scheme in TSCH using Latin rectangles," in *IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6.
[30] S. Sinche, D. Raposo, N. Armando, A. Rodrigues, F. Boavida, V. Pereira, and J. S. Silva, "A survey of IoT management protocols and frameworks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1168–1190, 2019.
[31] G. Cena, C. G. Demartini, M. G. Vakili, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluating and modeling IEEE 802.15.4 TSCH resilience against Wi-Fi interference in new-generation highly-dependable wireless sensor networks," *Ad Hoc Networks*, vol. 106, p. 102199, 2020.
[32] G. Daneels, D. Van Leemput, C. Delgado, E. De Poorter, S. Latré, and J. Famaey, "Parent and PHY selection in slot bonding IEEE 802.15. 4e TSCH networks," *Sensors*, vol. 21, no. 15, p. 5150, 2021.
[33] B. Moons, E. De Poorter, and J. Hoebeke, "Device discovery and context registration in static context header compression networks," *Information*, vol. 12, no. 2, p. 83, 2021.
[34] A. Minaburo, L. Toutain, and R. Andreasen, *Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)*, Internet Engineering Task Force (IETF) Std. RFC8824, 2021.
[35] M. G. Gaitán, D. Dujovne, J. Zuñiga, A. Figueroa, and L. Almeida, "Multigateway Designation for Real-Time TSCH Networks Using Spectral Clustering and Centrality," *IEEE Embedded Systems Letters*, vol. 15, no. 2, pp. 97–100, 2022.
[36] A. Kalita, A. Hazra, and M. Gurusamy, "Efficient schemes for improved performance in 6TiSCH networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM)*, New York, USA, 2023, pp. 1–6.
[37] M. Gudi, "CORECONF Applications for IoT Systems," Master's thesis, IMT Atlantique, Bretagne/ Pays de la Loire, 2023. [Online]. Available: https://drive.google.com/file/d/1rCWjvA-BDDnr4E6MNmh4FDIlq1AtWx9s/view
[38] C. Jennings, Z. Shelby, J. Arkko, A. Keränen, and C. Bormann, *Sensor Measurement Lists (SenML)*, Internet Engineering Task Force (IETF) Std. RFC8428, 2018.
[39] M. Gudi, "CCoreconf For Embedded Devices (and RIOT)," IETF, Tech. Rep., 2024. [Online]. Available: https://datatracker.ietf.org/meeting/interim-2024-t2trg-02/materials/slides-interim-2024-t2trg-02-sessa-ccoreconf-for-embedded-devices-riot-00
[40] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Torino, Italy, 2013, pp. 79–80.
[41] A. Tabouche, B. Djamaa, M. R. Senouci, G. A. Elaziz, and O. E. Ouakaf, "A Link Quality Aware Parent Selection Mechanism for Enhanced RPL Routing in 6TiSCH Networks," in *2nd International Conference on Electrical Engineering and Automatic Control (ICEEAC)*, 2024, pp. 1–6.
[42] F. X. K. Koesnadi and S.-H. Chung, "Optimizing Cell Allocation in 6TiSCH Networks based on Node Density Using K-Means Clustering," in *14th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Beijing, China, 2024, pp. 1–4.
[43] A. Tabouche, B. Djamaa, M. R. Senouci, O. E. Ouakaf, and A. G. Elaziz, "TLR: Traffic-aware load-balanced routing for industrial IoT," *Internet of Things*, vol. 25, 2024.
[44] M. Mohamadi, Q. Lampin, and M. Dumay, "CoAP-Based Remote Network Management Model for Deterministic 6TiSCH Networks," *IEEE Internet of Things Journal*, vol. 11, no. 23, pp. 37 509–37 524, 2024.
[45] M. G. Gaitán, L. Almeida, P. M. D'orey, P. M. Santos, and T. Watteyne, "Minimal-overlap centrality for multi-gateway designation in real-time TSCH networks," *ACM Transactions on Embedded Computing Systems*, vol. 23, no. 1, pp. 1–17, 2024.
[46] K. Zandberg, "NanoCBOR," https://github.com/bergzand/NanoCBOR, accessed: 2024-11-25.
[47] J.-H. Kwon, E.-J. Kim, J. Park, Y. Lim, Y.-S. Kim, and D. Kim, "Impact of Slotframe Length on End-to-End Delay in IEEE 802.15.4 TSCH Network," in *IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Bangkok, Thailand, Jun. 2019, pp. 111–112.
[48] F. Graf, T. Watteyne, F. Maksimovic, and M. Villnow, "Bit- and Symbol-Error Patterns in IEEE 802.15.4 TSCH Mode," in *29th IEEE Symposium on Computers and Communications (ISCC)*, Paris, France, 2024.
[49] F. Barac, M. Gidlund, and T. Zhang, "Scrutinizing bit-and symbol-errors of IEEE 802.15. 4 communication in industrial environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 7, pp. 1783–1794, 2014.
[50] R. Musaloiu-E and A. Terzis, "Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks," *International Journal of Sensor Networks*, vol. 3, no. 1, pp. 43–54, 2008.

Fabian Graf received the master's degree in electrical engineering from the Technical University of Munich (TUM), Munich, Germany in 2021 with specialization in communications engineering. He joined Siemens AG in 2019 and worked in different R&D teams on embedded systems.

Since 2023, he is pursuing the Ph.D. degree with the Sorbonne University, Paris, France. The thesis is based on a cooperation between Siemens AG, Erlangen, Germany and the Inria Institute, Paris, France. His current research interests include low-power wireless networks, related monitoring and management approaches and IIoT networking standards.

David Pauli received the masters's degree in computer science from the Friedrich-Alexander-University Erlangen-Nuremberg, Germany in 2025. He specialized in communication networks and embedded real-time systems. In January 2025, he completed his master's thesis at Siemens Foundational Technologies, in cooperation with the Chair of Computer Networks and Communication Systems at FAU. This paper is based on the thesis work. His research interests include embedded systems software architecture and optimization strategies for industrial communication systems. David is currently working as a software engineer for Siemens Mobility, developing communication networks for locomotives.

Michael Villnow is a Senior Project Manager at Siemens Foundational Technologies in the department Research and Pre-development. In 2008, Michael received a Diploma degree in Mechatronics from Friedrich-Alexander University of Erlangen-Nuremberg (FAU). Between 2009 to 2012 he worked as a research assistant in joint collaboration of FAU and Siemens AG. Content of his studies was applied optics with focus on fiber-optic sensor technologies, especially optical accelerometer. Michael joined Siemens finally in 2012. Since 2015 he is a project manager for IoT development projects. In 2017 he became Project Portfolio Manager responsible for the strategic core topic "sensor nodes & networks" and leads the company core technology of smart field devices.

Thomas Watteyne is an insatiable enthusiast of embedded technologies. He is a Principal Engineer at Analog Devices, in Boston. He is currently on a sabbatical from a Senior Research Director position at Inria in Paris in the AIO research team, working on robotic swarms and low-power wireless. In 2023-2024, Thomas served as the scientific coordinator of the Horizon Europe OpenSwarm project. Between 2013 and 2022, Thomas co-chaired the IETF 6TiSCH working group to standardize how to use IEEE802.15.4e TSCH in IPv6-enabled mesh networks. He was a postdoctoral research lead in Prof. Kristofer Pister's team at the University of California, Berkeley. Between 2005 and 2008, he was a research engineer at France Telecom, Orange Labs. He holds a PhD in Computer Science (2008), an MSc in Networking (2005) and an MEng in Telecommunications (2005) from INSA Lyon, France.