Quick6TiSCH: Accelerating Formation of 6TiSCH Networks with TSCH and RPL

Hongchan Kim*, Geonhee Lee*, Juhun Shin*, Jeongyeup Paek†, and Saewoong Bahk*

*Department of Electrical and Computer Engineering and INMC, Seoul National University, Seoul, Republic of Korea
†Department of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea
{hckim, ghlee, jhshin}@netlab.snu.ac.kr, jpaek@cau.ac.kr, sbahk@snu.ac.kr

2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS) | 979-8-3503-6399-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/MASS62177.2024.00020

***Abstract*—6TiSCH is an IPv6 protocol stack over *time-slotted channel hopping* (TSCH) mode of IEEE 802.15.4e and *routing protocol for low-power and lossy networks* (RPL), enabling low-power wireless multi-hop networking for Internet of Things (IoT). However, its network formation process involving TSCH bootstrapping, RPL topology formation, and TSCH resource scheduling, is often extremely slow due to its inherent rendezvous mechanism; the use of *TSCH common shared cells*. Common shared cell is the sole transmission path during network formation, and thus collisions are frequent in these scarce resources while clear channel assessment fails to resolve the issue. To address this problem, we propose *Quick6TiSCH*, a method for accelerating 6TiSCH network formation by prioritizing critical control messages and diversifying their transmission times within common shared cells to facilitate collision avoidance among nodes. Since negotiation is infeasible prior to network formation, *Quick6TiSCH* adopts an autonomous approach allowing each node to adaptively mitigate collisions based on its network join progress. Evaluation on real-world testbeds demonstrate a significant reduction in network formation time by 62.6% with transmission overhead similar to the baseline.**

***Index Terms*—6TiSCH, TSCH, RPL, IEEE 802.15.4, LLN**

I. Introduction

6TiSCH [1] is an IETF standard protocol stack that defines IPv6 over IEEE 802.15.4e *time-slotted channel hopping* (TSCH) [2]. Together with the *routing protocol for low-power and lossy networks* (RPL) [3]–[5], it enables low-power wireless multi-hop networking for resource constrained embedded devices. 6TiSCH has attracted significant attention for its applicability in various industrial and Internet of Things (IoT) applications [6]–[8]. However, from various studies on 6TiSCH (§III), it is well-known that the network formation process of 6TiSCH could progress extremely slowly. This directly affects the network availability as well as how quickly applications are executable on the network.

Formation of a 6TiSCH network encompasses the processes of joining at the TSCH layer (link), joining at the RPL layer (routing), and establishing bidirectional links. Subsequently, allocation of resources (TSCH cells) for data transmission [9]–[12] takes place. In TSCH, there is a *common shared cell* per every repeating group of timeslots (a.k.a slotframe, Fig. 1), a common channel resource automatically allocated and accessed by all nodes (details in §II-A). Since additional resource allocation (i.e., cells for unicast transmissions) is performed only after completion of network formation, all nodes communicate through this scarce resource during network formation. This can lead to severe collisions and congestion in the common shared cells, especially during the bootstrapping phase, resulting in transmission failures and re-transmissions leading to prolonged network formation time.

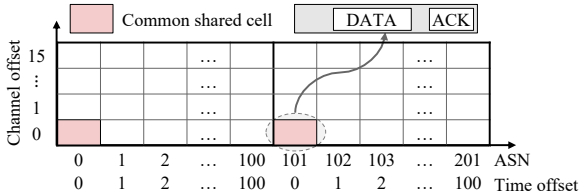


Fig. 1: Illustration of TSCH channel resources with a slotframe size of 101. There is only one *common shared cell* every 101 timeslots.

Despite frequent collisions in common shared cells during network formation, the inherent design of TSCH does not effectively resolve this issue. In TSCH, all nodes are time-synchronized and communicate in a slotted manner as shown in Fig. 2a. Thus, if multiple nodes attempt to transmit within a same timeslot, they start simultaneously as shown in Fig. 2b, resulting in a collision. Even with clear channel assessment (CCA) before transmission, the transmitters are unaware of each other due to the aligned start times, making collisions unavoidable. Consequently, severe collisions occur repeatedly within common shared cells, leading to inefficiency and delays during network formation process.

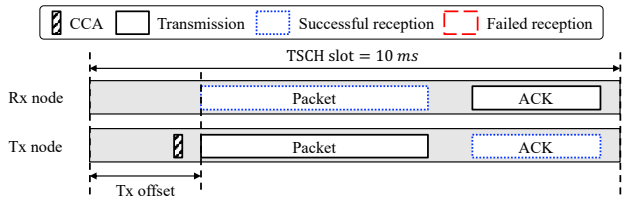
To address this problem, we propose *Quick6TiSCH* which enables transmission detection among nodes within common shared cells by diversifying the start times of transmissions. Specifically, *Quick6TiSCH* assigns different transmission time offsets to each node based on its network formation progress, transmission status, and priority of messages, to distribute the transmission start points across the time domain. This approach allows *Quick6TiSCH* to mitigate collisions in common shared cells and accelerates the network formation process.

We implement *Quick6TiSCH* on real IEEE 802.15.4 embedded devices using Contiki-NG [13], and evaluate on a large-

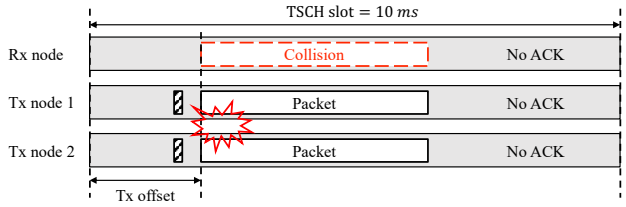
This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-2021-0-02048) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), also by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A5A1027646 & 2022R1A4A5034130).

S. Bahk and J. Paek are co-corresponding authors.

2155-6814/24/$31.00 ©2024 IEEE
DOI 10.1109/MASS62177.2024.00020



(a) Successful packet transmission within a TSCH timeslot.



(b) Packet collision within a TSCH timeslot.

Fig. 2: Unavoidable collision within a TSCH timeslot due to the common transmission (Tx) time offset.

scale topology in the FIT/IoT-LAB public LLN testbed [14]. The results demonstrate substantial improvements in network formation time by 62.6% with similar transmission overhead compared to the default 6TiSCH.

Our contributions can be summarized as follows.

- We identify the root cause of 6TiSCH network formation delays that CCA fails to mitigate.
- We propose *Quick6TiSCH*, a method to accelerate network formation in 6TiSCH networks.
- We implement *Quick6TiSCH* on real embedded devices, and evaluate in a sizeable public testbed to demonstrate significant performance improvement.

The remainder of this paper is organized as follows: We discuss the background and motivation of this work in §II, and review related prior work in §III. We then present the design of our proposed scheme in §IV, and evaluate our proposal in §V. Finally, we conclude the paper in §VI.

II. Background and Motivation

We first briefly introduce the TSCH and RPL protocols, and then describe the 6TiSCH network stack, its formation process, and the problem and motivation behind this work.

A. Time-Slotted Channel Hopping (TSCH)

TSCH [2] is a medium access control (MAC) protocol standardized in IEEE 802.15.4e. By synchronizing the network and enabling time-slotted communication, TSCH enhances reliability and energy efficiency. Its channel hopping further improves resilience to external interference and fading by leveraging channel diversity. Fig. 1 illustrates an example of TSCH's resource map operation.

TSCH divides time into *timeslots*, typically set to 10 ms each, allowing for an exchange of a maximum-sized (128 bytes) frame and an acknowledgement (ACK) of up to 70 bytes [15]. As shown in Fig. 2a, TSCH nodes by default begin transmission at the same time offset (Tx offset) from the start of a timeslot. Each timeslot is assigned an *absolute slot number (ASN)* which starts at zero when the network begins and increments sequentially. A collection of timeslots forms a *slotframe*, which repeats over time and serves as a scheduling unit. The number of timeslots within a slotframe is the slotframe length (L_{SF}) (e.g. 101 in Fig. 1). The *time offset* (t_o) represents the relative position of a particular timeslot within a slotframe and is calculated as,

$$t_o = mod(ASN, L_{\mathrm{SF}}). \tag{1}$$

To determine the channel for hopping, TSCH uses a *channel offset* (c_o). There are up to 16 channels, and the channel for each timeslot is calculated based on the channel offset as,

$$Channel = List_{\mathrm{c}}[mod(ASN + c_{\mathrm{o}}, sizeof(List_{\mathrm{c}}))] \tag{2}$$

where $List_{\mathrm{c}}$ is a set of channels to be used, and $sizeof(List_{\mathrm{c}})$ is the number of channels in $List_{\mathrm{c}}$. As ASN increases, each timeslot with a specific c_o hops over different channels. Even with the same ASN timeslot, different c_o values result in selection of different channels.

TSCH standard defines time-slotted communication and channel hopping, but leaves *resource scheduling*–the selection of when (t_o) and on which channel (c_o) each device communicates–as an open problem. To fill this gap, various TSCH schedulers have been proposed [9]–[12]. One of the simplest is the 6TiSCH minimal configuration (6TiSCH-MC) [16], which autonomously allocates a single resource shared by all nodes (i.e., *a common shared cell*) at a fixed time and channel offsets in every slotframe (e.g., zero for both in Fig. 1). It is important to note that the slow network formation problem we aim to solve is largely independent of the scheduler used. During the network formation process, as there are typically no other scheduled resources, most packet transactions occur in commonly shared cells similar to the common shared cell in 6TiSCH-MC. This means that the problem can occur regardless of the scheduler type. Therefore, without loss of generality, we select 6TiSCH-MC for our study.

In a TSCH network, synchronization information is propagated through a control message called Enhanced Beacon (EB). Starting from the TSCH coordinator, all TSCH joined nodes periodically transmit EB messages. A new node not yet joined to the TSCH network waits for EB broadcasted by already joined nodes while scanning all available channels. Upon receiving an EB, the node extracts TSCH network information (including synchronization) and joins the network.

B. Routing protocol for low-power and lossy networks (RPL)

RPL, the IETF standard *IPv6 routing protocol for low-power and lossy networks*, is tailored for resource-constrained IoT devices [3]–[5], [17], [18]. As a distance-vector protocol, RPL forms a tree-like routing topology called a *destination-oriented directed acyclic graph* (DODAG). Nodes broadcast *DODAG Information Object* (DIO) messages to share and update routing information. Upon receiving a DIO message, a node selects a neighboring node as its *parent* for its upward

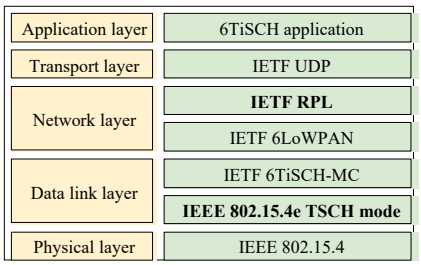


Fig. 3: An example of the 6TiSCH network stack.

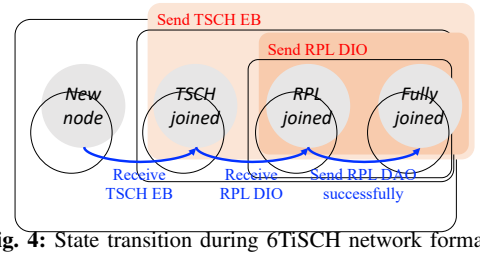


Fig. 4: State transition during 6TiSCH network formation.

route towards the root. The node then sends a *Destination Advertisement Object* (DAO) message to this parent to establish the downward path, creating a bidirectional link. If the node needs to change its parent node, it sends a DAO message to the new parent and a *no-path DAO* message to the old parent to update the forwarding tables. A node can trigger DIO transmissions from its neighbors by sending a *DODAG Information Solicitation* (DIS) message. Upon receiving a DIS message, nodes minimize their DIO transmission intervals to quickly send DIO messages.

C. 6TiSCH stack and network formation process

As aforementioned, 6TiSCH [1] is a network stack designed for low-power wireless multi-hop IPv6 networking over IEEE 802.15.4e TSCH mode with RPL. Fig. 3 presents the 6TiSCH stack considered in this work. The physical layer of 6TiSCH is IEEE 802.15.4, with TSCH and RPL serving as the link layer and routing layer, respectively. On top of the TSCH, 6TiSCH-MC allocates a common shared cell as shown in Fig. 1. To support IPv6 communication, the 6LoWPAN adaptation layer is added above the link layer. UDP is commonly used as the transport layer protocol for the 6TiSCH stack.

For the 6TiSCH network to fully operate, *network formation*, which includes joining processes across multiple layers, must occur. Fig. 4 illustrates the overall 6TiSCH network formation process modeled as a state transition diagram. 6TiSCH nodes transition sequentially through four essential states: *new node*, *TSCH joined*, *RPL joined*, and *fully joined*.

Initially, in the *new node* state, a node waits for TSCH EB messages broadcasted by nodes in the *TSCH joined* or subsequent states. Upon receiving an EB, the node joins the TSCH network, moving into the *TSCH joined* state with TSCH communication capability but only on the common shared cells. The node then listens for RPL DIO messages broadcasted by nodes in the *RPL joined* or subsequent states.

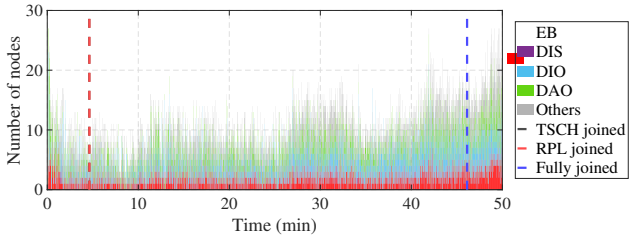


Fig. 5: Time evolution of transmissions within common shared cells, and the time when the state transition of all nodes completed.

This allows the node to join the RPL network and establish an upward route, moving into the *RPL joined* state. In the *RPL joined* state, the node sends a DAO message to its RPL parent to establish a downward route, finally transitioning to the *fully joined* state where a bidirectional unicast link is established.

As described, the transitions between join states rely on the exchange of specific control messages including EB, DIO, and DAO. The prerequisite conditions for state transitions are the reception of EBs and DIOs, and the successful transmission of DAOs, as shown in arrows in Fig. 4. To enable other nodes' state transitions, a node must transmit EB or DIO messages according to their states. For example, nodes in states after *TSCH joined* must send EB messages, and nodes in states after *RPL joined* must send DIO messages, as depicted in red in Fig. 4. In short, to ensure swift 6TiSCH network formation, it is crucial that the necessary packets are exchanged promptly for each node to successfully complete its state transitions.

D. Slow 6TiSCH network formation problem

We have argued that 6TiSCH network formation, occurring within the common shared cells accessed by all nodes, is prone to severe collisions. Due to its operational characteristics, TSCH cannot effectively handle these collisions, potentially leading to slow network formation. Here we validate this through experiments on a real testbed with 83 nodes (Fig. 8) using 6TiSCH-MC scheduler with a common shared slotframe size of 101. Fig. 5 illustrates the number of nodes attempting transmissions in the common shared cell over time, with different colors representing various types of messages. The vertical dashed lines indicate the time taken for all nodes to reach each state in Fig. 4 at least once. The result reveals that immediately after network initiation, a huge number of nodes engage in packet transmissions within the common shared cell. In some cells, more than 20 nodes attempt to transmit packets simultaneously. However, successful reception rate for these packets is notably low. Consequently, it takes more than 45 minutes for all 83 nodes to reach the *fully joined* state.

Addressing collisions within the common shared cell and accelerating the network formation process is crucial to make 6TiSCH network practically useful. This motivates us to propose *Quick6TiSCH*, a method that enables each node to adaptively mitigate collisions by prioritizing critical control messages and diversifying their transmission times within common shared cells. This autonomous approach, based on

each node's network join progress, results in swift 6TiSCH network formation.

III. Related Work

There have been several prior research efforts aimed at improving 6TiSCH network formation.

Vallati et al. [19] proposed dynamic resource allocation (DRA) of common shared cells based on control packet load to mitigate collisions. However, this approach necessitates negotiation between nodes. Additionally, it converts other scheduled cells into common shared cells, resulting in decreased data throughput and increased energy consumption. Vucinic et al. [20] found that EB transmissions can cause congestion in common shared cells and suggested a low EB rate to alleviate this issue at the cost of delayed state transitions. C2DBI [21] dynamically adjusts the EB generation interval to address congestion, while Kalita et al. [22] dynamically adjust the priority of control packets to prevent delays caused by the highest priority given to EBs. TRGB [23] divides the common shared cells into three types and differentiates packets transmitted within each type of common shared cell. The goal is to reduce collision probability, but this approach lead to resource scarcity issues as resources are divided into thirds. None of these studies investigated the fundamental causes of delay in network formation: the unavoidable collision problem within the common shared cell due to closely aligned transmission start times.

The concept of assigning distinct offsets to transmission times within TSCH slots has been explored in previous studies [24], [25]. In [24], transmission offset differentiation is utilized in dedicated cells, allowing nodes that do not own the dedicated cell to utilize it when the owner is not transmitting packets, with the goal of reducing latency. DualBlock [25] introduces multiple offsets in shared cells for unicast transmission, aiming to alleviate collisions. However, these studies focus on mitigating collisions during data transmission phase rather than the network formation phase.

IV. Proposed Scheme

We propose *Quick6TiSCH*, a method to accelerate 6TiSCH network formation. We describe the diversification of Tx time offsets within the common shared cell, and then explain how formation-critical packets are autonomously prioritized based on formation progress, message type, and transmission status. We also outline the supplementary features of *Quick6TiSCH*.

A. Diversification of transmission time offsets

In default TSCH, transmission start times within a common shared cell are closely aligned, preventing nodes from detecting each other's transmissions and causing unavoidable collisions as shown in Fig. 2b. To address this issue, *Quick6TiSCH* diversifies Tx time offsets illustrated in Fig. 6. While Fig. 2b accounts for the longest ACK (70 byte) filling the 10 ms slot, Fig. 6 assumes a more typical ACK length (20 byte).

In Fig. 6, packet transmission of Tx node 2 begins after a delay of ΔT compared to Tx node 1, enabling Tx node 2 to perform CCA after Tx node 1's transmission has started. This strategic delay enables Tx node 2 to detect Tx node 1's transmission. Then, Tx node 2 could attempt its packet transmission in a later slot by applying the backoff mechanism [2]. Consequently, the Rx node successfully receives the packet from Tx node 1 and sends back an ACK. In this way, TSCH nodes can detect transmissions from nodes with smaller offsets, avoiding collisions and facilitating successful packet delivery.

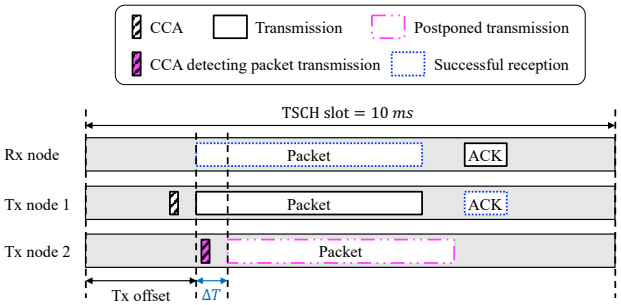


Fig. 6: Collision avoidance through diversification of transmission time offsets in each synchronized timeslot.

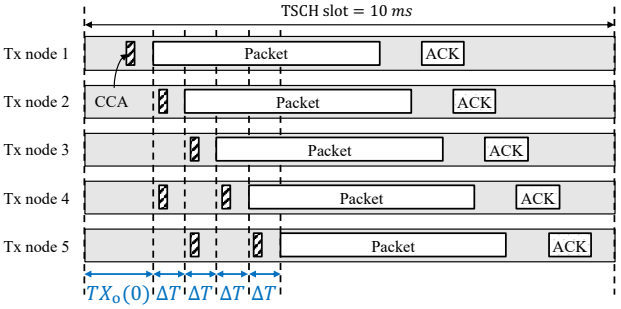


Fig. 7: *Quick6TiSCH*'s transmission offset diversification.

During 6TiSCH network formation process, multiple nodes may simultaneously access a single common shared cell. To enhance the effectiveness of the diversification despite this simultaneous access, *Quick6TiSCH* needs to distribute transmission start times across the time domain as widely as possible. Fig. 7 illustrates our proposal on this, showing five different transmission time offsets.

The default Tx offset (in Fig. 2) specified in the standard [2] is set with a generous margin, allowing flexibility for adjusting the actual transmission start time forward. Additionally, since typical ACK length (20 bytes) is much shorter than the maximum size (70 bytes) [15], the transmission start time can also be adjusted backward relative to the default TSCH configuration. Based on these two observations, by spreading transmission start times both forward and backward, *Quick6TiSCH* could support five distinct Tx time offsets as illustrated in Fig. 7. The earliest transmission begins at $TX_o(0)$ at the slot's outset, with subsequent transmissions spaced at intervals of ΔT.

By performing CCA once or twice before starting transmis-

TABLE I: Conditions for the formation-critical packets

State	TSCH EB	RPL DIO	RPL DAO
New node	N/A	N/A	N/A
TSCH joined	Up to k EB packet(s)		
RPL joined		Up to k DIO packet(s)	DAO packet(s) until successful delivery
Fully joined			N/A

sion, each preceding packet transmission becomes detectable to following transmissions. For example, Tx node 5, with the largest Tx time offset can detect transmissions from Tx nodes 1 and 2 via its first CCA, and transmissions from Tx nodes 3 and 4 through its second CCA. In this manner, *Quick6TiSCH* avoids collision in common shared cells, which is impossible in the default TSCH.

B. Autonomous critical control packet prioritization

Quick6TiSCH determines packet criticality, and autonomously prioritizes critical control packets at each node based on network formation progress, message type, and transmission status.

Identification of critical packet: *Quick6TiSCH* identifies the packets required for state transitions in Fig. 4 as formation-critical packets. Transitioning from the *new node* to the *TSCH joined* state requires receiving TSCH EB messages, while transitioning from the *TSCH joined* to the *RPL joined* state requires receiving RPL DIO messages. Consequently, nodes in the *TSCH joined* state or beyond must send some EB packets, and nodes in the *RPL joined* state or beyond must send both EB and DIO packets. Additionally, transitioning to the *fully joined* state requires sending an RPL DAO message and receiving the corresponding ACK.

Table I summarizes the conditions outlined above for formation-critical packets. Given *Quick6TiSCH*'s principle of autonomous operation, we simplify the conditions for EB and DIO. For nodes in the *TSCH joined* state or later, up to k EB packets are considered formation-critical. Similarly, for nodes in the *RPL joined* state or later, up to k DIO packets are considered formation-critical. For nodes in the *RPL joined* state, DAO packet(s) are considered formation-critical until successful delivery. Then, *Quick6TiSCH*'s autonomous packet prioritization for those critical packets follows a two-step procedure: (1) network-level prioritization, and (2) node-level prioritization.

Network-level prioritization: To ensure critical packets are prioritized for transmission throughout the network, *Quick6TiSCH* separates offsets for critical and non-critical packets: critical packets are assigned smaller offsets, while non-critical packets receive larger offsets. *Quick6TiSCH* implements this as follows.

$$TX_{\mathrm{o}}^{non} = TX_{\mathrm{o}}(0) + \Delta T \cdot (N-1). \tag{3}$$

$$TX_{\mathrm{o}}^{crit}(n) = TX_{\mathrm{o}}(0) + \Delta T \cdot n, \text{ s.t. } 0 \le n < N-1. \tag{4}$$

TX_{o}^{non} and $TX_{\mathrm{o}}^{crit}(n)$ denote the transmission time offsets for non-critical packets and critical packets, respectively. N is the number of offsets (5 in our study). $TX_{\mathrm{o}}(0)$ is the earliest Tx time offset, and ΔT is the time interval between distinct offsets (Fig. 7). Through Eq. (3), non-critical packets are assigned the largest offset. According to Eq. (4), critical packets are assigned smaller offsets, thus always having higher priority over non-critical packets.

In this way, *Quick6TiSCH* prioritizes critical packets without any negotiation and in a fully autonomous manner. Whenever a critical packet transmission occurs, the transmission of non-critical packets is deferred. In the absence of critical packet transmissions, non-critical packets are transmitted in the same manner as the default TSCH operation.

Node-level prioritization: In addition to the *network-level prioritization*, *Quick6TiSCH* incorporates *node-level prioritization*. This prioritization ensures that critical packets, as determined by Table I based on the node's network formation progress and message type, are sent first from the buffer.

C. Further collision avoidance for critical control packets

In addition to prioritization, *Quick6TiSCH* enables collision avoidance, a capability absent in default TSCH. The key to achieving collision avoidance lies in nodes selecting different transmission time offsets. In Eq. (4), we specified the range of transmission time offsets for critical packets, and there are multiple offsets (4 in our study) available for collision avoidance. The challenge lies in distributing the critical packets across these multiple offsets. It is important to note that relying on node negotiation is impractical during formation as the common shared cell–the sole transmission path–experiences severe congestion. Instead, *Quick6TiSCH* uses random offsets to autonomously spread nodes across different transmission time offsets as $n = \mathit{random}(N-1)$. Then, *Quick6TiSCH* lets the transmission for critical packets begin at the randomly selected offset, $TX_{\mathrm{o}}^{crit}(n)$.

Meanwhile, if a node repeatedly selects large offsets while other nodes choose smaller ones, the node with the larger offset will continuously defer its transmission, potentially resulting in delivery failure for extended time. To address this issue, *Quick6TiSCH* introduces *prioritization based on transmission status*, particularly focusing on the *number of transmission postponements*. As the count of postponements increases, *Quick6TiSCH* gradually reduces the upper bound for selecting a random transmission time offset, thereby assigning smaller offsets and increasing the priority. To this end, *Quick6TiSCH* defines a constant P, representing the number of postponements after which the upper bound of the offset is reduced. If the number of postponements for a specific packet is p, *Quick6TiSCH* selects the random number for the transmission time offset as,

$$n = \mathit{random}(\max(0, N-1-\lfloor p/P \rfloor)) \tag{5}$$

Then, *Quick6TiSCH* determines the Tx time offset for critical packets according to Eq. (4). This approach ensures that packets are eventually assigned the earliest offset (i.e., $TX_{\mathrm{o}}(0)$),

allowing them to be transmitted regardless of the presence of other packet transmissions.

D. Supplementary features of Quick6TiSCH

The standard TSCH does not account for detection of other packets through CCA. Thus, to enable *Quick6TiSCH* to function effectively with the aforementioned features, we need to enhance the retransmission and backoff mechanisms, as well as the management policy for packets awaiting transmission.

Legacy retransmission and backoff mechanisms: In default TSCH, retransmission mechanism ensures successful unicast packet delivery. If a transmission fails, it is retransmitted, incrementing the retransmission count by 1. When this count reaches the limit (i.e., *macMaxFrameRetries* as defined in [2]), the packet is dropped. If the channel is detected as busy through CCA before transmission starts, the retransmission count is incremented, and retransmission will be performed later without attempting transmission within the current cell. Backoff mechanism resolves contention within the common shared cell during retransmission. Each retransmission increases the backoff exponent (*BE*) by 1 (up to a maximum), and a random backoff counter is selected from the range $[0, 2^{BE} - 1]$. This counter decreases at every scheduled cell until it reaches 0; then the next retransmission is performed.

For broadcast packets, there is no notion of transmission failures. If CCA detects a busy channel and refrains from transmitting a broadcast packet, the retransmission occurs immediately in the next cell without any backoff or consideration of retransmission counts.

Modification on retransmission and backoff mechanisms: When *Quick6TiSCH* detects packets from other nodes for collision avoidance, incrementing the retransmission count would be inappropriate, as the detection does not signify transmission impossibility but rather intentional postponement. Therefore, we propose modifying the TSCH operation for *Quick6TiSCH*, ensuring that when a busy channel is detected via CCA within the common shared cell and no transmission is attempted, the retransmission count remains unchanged. Despite maintaining an unchanged retransmission count, *Quick6TiSCH* allows the backoff mechanism to operate as usual. This is necessary because the detection of packets from other nodes within a common shared cell indicates a need for contention resolution. *Quick6TiSCH* introduces a random backoff mechanism for broadcast packets too, extending the behavior previously applied only to unicast packets.

In default TSCH, backoff counter decreases per every scheduled slot (cell), triggering retransmission when it hits 0. During network formation, cells other than the common shared cell might be scheduled. In such cases, the backoff counter decreases not only in the common shared cell but also in the other cells. This additional decrease could diminish *Quick6TiSCH*'s congestion mitigation efficacy. For instance, if the counter hits 0 before the next common shared cell, retransmission would occur immediately without effectively resolving contention. To address this issue, we propose an additional per-slotframe backoff operation. It updates the per-slotframe backoff exponent/counter alongside the original backoff, but decreases the counter only at the common shared cell. Access to the common shared cell is granted only when the per-slotframe backoff counter is 0. This ensures that the backoff counter, set based on congestion in the common shared cell, reflects its evolution only, enabling *Quick6TiSCH* to more effectively mitigate congestion within the common shared cell.

Policy for packets awaiting transmission: *Quick6TiSCH* prevents collisions within common shared cell by refraining from attempting packet transmission when other transmissions are detected. However, this advantage comes with a downside: packets awaiting transmission must be buffered, potentially leading to longer storage times compared to the default TSCH. Given the typically small buffer sizes of 6TiSCH devices, this could result in overflow. To address this issue, *Quick6TiSCH* includes a packet management policy. Specifically, *Quick6TiSCH* replaces buffered packets with newly created packets of the same type when their sequence does not matter, such as TSCH EB and RPL DIO. By doing so, *Quick6TiSCH* not only avoids collisions in common shared cells but also prevents buffer overflow and ensures the delivery of latest packets, facilitating swift network formation.

V. Evaluation

We evaluate *Quick6TiSCH* against state-of-the-art studies for fast 6TiSCH network formation on a large-scale testbed.

A. Implementation and experiment setup

We implement *Quick6TiSCH*[1] on M3 board using Contiki-NG[2] [13]. For the baseline scheme, we utilize the 6TiSCH-MC [16] scheduler implementation in Contiki-NG. As comparison schemes, we implement *DRA* [19] and *TRGB* [23] in Contiki-NG. Slot length is set to 10 ms by default. For channel hopping, *Quick6TiSCH*, 6TiSCH-MC, and DRA uses four IEEE 802.15.4 channels (15, 20, 25, 26) while allowing TRGB to use all sixteen IEEE 802.15.4 channels as in their paper. While the others have a maximum EB sending interval of 16 seconds, TRGB has a 4-second interval (i.e., four times more EB messages generated) as in their original paper. At the routing layer, we employ RPL storing mode [26] and use MRHOF with ETX [27] as the objective function. DAO-ACK option in RPL is enabled. Tx power is set to -17 dBm.

For *Quick6TiSCH*, we implement five different transmission time offsets and corresponding CCA operation within the slot. To enable operation with multiple offsets while maintaining synchronization, the indices of the utilized offsets are encapsulated as an information element (IE) into the every packet transmitted via the common shared cell. Regarding the criteria for determining the criticality of EB and DIO messages (Table I), we set the parameter k to 2 considering the existence of nodes that fail to hear the first message. We utilize TSCH's *macMaxFrameRetries* parameter [2], which defines the maximum number of transmission attempts allowed for a

[1]https://github.com/Hongchan-Kim/Quick6TiSCH
[2]https://github.com/iot-lab/iot-lab-contiki-ng

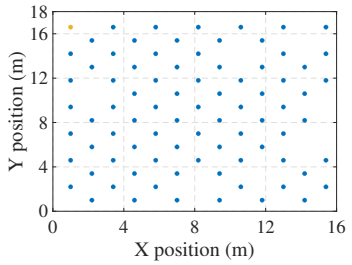
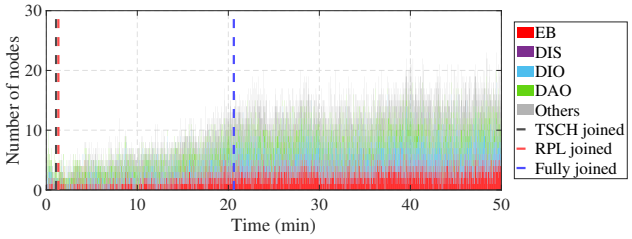
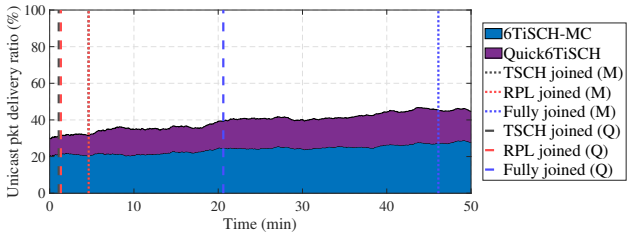


Fig. 8: Node deployment topology at Lille testbed. The node located at the upper left corner and marked in yellow serves as the root.



(a) Time evolution of transmissions within the common shared cell and the time when the state transition of all nodes completed for *Quick6TiSCH*



(b) Unicast packet delivery ratio within the common shared cell

Fig. 9: Time evolution of transmissions and unicast packet delivery ratio within the common shared cell for *Quick6TiSCH* and 6TiSCH-MC [16]. (M) and (Q) are state transition time for 6TiSCH-MC and *Quick6TiSCH*, respectively.

packet at the MAC layer, as the constant P for *prioritization based on transmission status*. In Contiki-NG, once a node enters the *TSCH joined* state upon receiving an EB message, it transmits a *keep alive* (KA) message to achieve clock drift correction at least once unless clock drift is corrected by transmission or reception of any packet. Failure to do so causes the node to revert back to the *new node* state. Therefore, we classify KA message as critical only before synchronization is achieved; once clock drift has been corrected at least once, it is considered non-critical.

We conduct experiments on the FIT/IoT-LAB testbed, a large-scale public testbed located in Lille. The physical deployment topology of the 83 nodes is illustrated in Fig. 8. The nodes are almost evenly distributed in a rectangular grid shape, forming a 3∼4 hop topology. With this setup, we measure the network formation performance for each scheme, ten times for each experimental case.

B. Performance of Quick6TiSCH: an overview

First, we examine whether *Quick6TiSCH* mitigates contention within the common shared cell during network formation and reduces the network formation time as intended.

Fig. 9 shows the result. Fig. 9a depicts the number of nodes attempting transmissions in the common shared cell over time, as well as the time at which all nodes transition to the *TSCH joined*, *RPL joined*, and *fully joined* states for *Quick6TiSCH*. *Quick6TiSCH* demonstrates time evolution distinct from that of 6TiSCH-MC (Fig. 5). Messages critical for state transitions, such as EB, DIO, and DAO, are transmitted in a prioritized manner, blocking transmissions of non-critical packets. Consequently, at the beginning of network formation, fewer packets are transmitted compared to 6TiSCH-MC, effectively mitigating congestion and avoiding collisions. This results in a swift transition to the *fully joined* state, taking 20.6 minutes, which is less than half of the 46.12 minutes shown in Fig. 5 for 6TiSCH-MC.

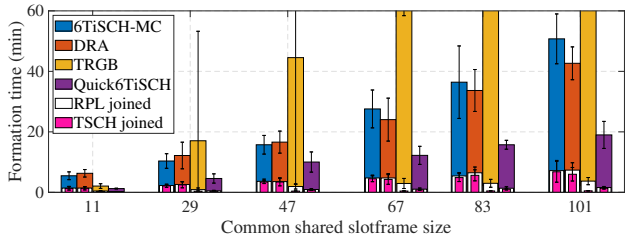
Fig. 9b illustrates the reason behind *Quick6TiSCH*'s swift network formation. Specifically, it shows the unicast packet delivery ratio within the common shared cell during network formation, for *Quick6TiSCH* (Fig. 9a) and 6TiSCH-MC (Fig. 5), respectively. Thanks to *Quick6TiSCH*'s prioritization of critical packets and collision avoidance, *Quick6TiSCH* consistently exhibits a higher delivery ratio compared to 6TiSCH-MC. We observe similar trends for broadcast packets as well. By successfully delivering packets, *Quick6TiSCH* quickly completes the state transitions necessary for network formation. Even after all nodes have transitioned to the *fully joined* state at least once, network formation-related packets may still be generated, for example, due to RPL topology maintenance or TSCH synchronization maintenance. In these cases, *Quick6TiSCH* maintains a higher packet delivery rate by prioritizing the critical packets and avoiding collisions.

C. Performance of Quick6TiSCH: a comparative study

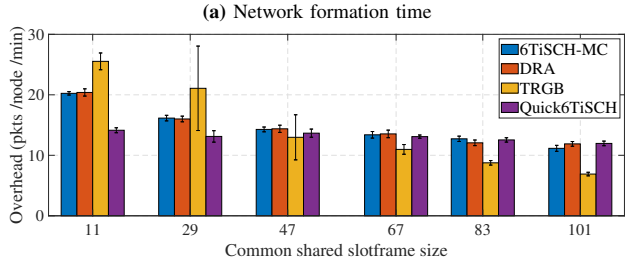
We compare *Quick6TiSCH* against other state-of-the-art studies for fast 6TiSCH network formation, DRA [19] and TRGB [23], in terms of network formation time, control overhead, and duty cycle.

Network formation time: Fig. 10a plots the network formation time. In each case, the outermost bar represents the point at which all nodes have reached the *fully joined* state at least once. The inner white and pink bars represent the points at which all nodes have reached the *RPL joined* and *TSCH joined* state, respectively, at least once for the corresponding scheme. Commonly, as the slotframe size increases, the common shared cell becomes scarcer, leading to more collisions and longer network formation times.

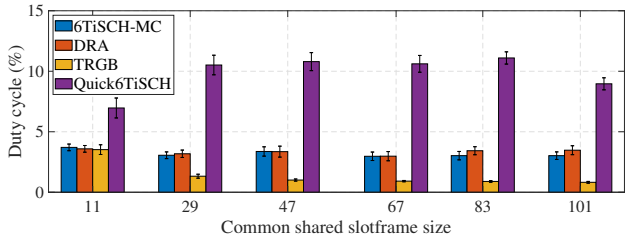
Quick6TiSCH significantly outperforms the other three schemes in terms of network formation time regardless of the slotframe size. This improvement is attributed to *Quick6TiSCH*'s autonomous prioritization of formation-critical packets, which effectively mitigates collisions in the common shared cell and ensures prompt delivery of the critical packets.



(a) Network formation time



(b) Control overhead during network formation



(c) Duty cycle during network formation

Fig. 10: Network formation time, overhead, duty cycle for different slotframe sizes. Above a slotframe size of 67, TRGB [23] shows a long formation time exceeding 120 minutes, indicating malfunctioning. We measure the overhead and duty cycle up to that point.

In the case of DRA, network formation is not much faster than 6TiSCH-MC; rather, it is similar or slower. In the case of TRGB, it achieves faster network formation than 6TiSCH-MC only when the slotframe size is small (e.g., 11). However, as the slotframe size increases, the time to reach *fully joined* state increases drastically, even exceeding 120 minutes.

Both DRA and TRGB rely on negotiation between nodes which hinders their network formation performance. DRA fails to operate effectively due to severe congestion in the common shared cell which prevents successful negotiation. As a result, DRA rarely adjusts resources according to traffic demands. Even more, the scarce adjustments are not coordinated among neighboring nodes, causing a mismatch in resource allocation between neighboring nodes leading to transmission failures. These transmission failures accumulate and ultimately increase network formation time. As a result, DRA does not perform better than 6TiSCH-MC.

TRGB also suffers from inefficiencies due to its negotiation-based approach. It divides the common shared cell into three time-domain resources: Red cell for broadcasting routing packets, and Green and Blue cells for transmitting/receiving other types of packets. While the Red cell is always available, access to the Green/Blue cells for transmission depends on indications from the RPL parent. Until then, access to Green/Blue cells is limited to reception only. During network formation, routing topology changes can render a node unable to use Green/Blue cells, requiring new indications. Since a node's access to Green/Blue cells depends on its parent node's availability, even a single topology change can render these cells unusable for all descendant nodes until new indications propagate downward, causing delays. Consequently, the Red cell becomes the sole functioning communication path. However, in practical 6TiSCH networks, messages other than broadcast routing information must also use the Red cell, worsening its congestion. For instance, a no-path DAO for a non-RPL parent node must be sent via the Red cell due to uncertainty in Green/Blue cell reception availability. This results in fewer successful message transactions within the Red cell, leading to failures not only in TRGB negotiation but also in synchronization. Overall congestion significantly prolongs network formation times. Thus, despite configuring TRGB to use more channels and generate more EB messages than other schemes, TRGB fails to achieve its intended performance.

The discussion above reinforces our argument favoring autonomous methods to facilitate 6TiSCH network formation. The slow pace of 6TiSCH network formation is primarily due to severe congestion in common shared cells. Attempting to resolve this congestion through negotiations over already congested shared cells only exacerbates the issue. Therefore, autonomous methods like *Quick6TiSCH* should be prioritized.

Overhead during network formation: Fig. 10b plots the overhead (i.e., average number of control packets) during network formation. We count all control messages until all nodes reached the *fully joined* state at least once. Then we normalized the number of control packets by the number of nodes and the formation time. As shown in the figure, *Quick6TiSCH* incurs a smaller or similar control overhead compared to other schemes. TRGB shows a steeper decrease in overhead compared to other schemes as the slotframe length increases. However, this does not indicate that TRGB operates more efficiently. Instead, it reflects TRGB's inability to utilize resources due to failed negotiations and subsequent packet transmission failures. Consequently, nodes fail to exchange packets, lose synchronization, and revert to the *new node* state. This prevents the generation of control messages that would typically occur in subsequent states, reducing the measured overhead.

The results show that as the common shared slotframe size increases, the overall amount of control overhead decreases. Generally, generation of periodic broadcast control packets in the 6TiSCH network decreases over time [28]. Additionally, generation of on-demand control packets also becomes less frequent as network formation progresses. As the slotframe size lengthens, the formation time becomes longer. As a result, the time average amount of control overhead decreases. We note that *Quick6TiSCH* achieves the shortest network formation time, meaning the generation rate of control packets during network formation would be the highest among all

schemes. Despite this, *Quick6TiSCH* showing similar overhead to other schemes indicates that *Quick6TiSCH* effectively utilizes the control overhead, resulting in efficient and swift network formation.

Energy efficiency (duty cycle) during network formation: Fig. 10c illustrates the average duty cycle among nodes during network formation. In default TSCH, nodes typically remain in sleep mode and wake up only at a single point within a slot for radio activities like packet transmission or reception. In contrast, *Quick6TiSCH* allows radio activities at up to five different positions within a slot. From the perspective of a transmitter node, the single wakeup per transmission remains unchanged. But there may be instances where an additional CCA is required, potentially increasing the duty cycle. From the perspective of a receiver, monitoring five potential radio activity regions increases duty cycle slightly. However, this increase significantly reduces network formation time (Fig. 10a), a cost worth paid. Furthermore, it is important to note that increased duty cycle need not be permanent. After completing network formation, *Quick6TiSCH* may return to default TSCH common shared cell operation.

On the other hand, TRGB exhibits a significantly lower duty cycle compared to other schemes above a slotframe size of 29. This is because TRGB fails in negotiation as slotframe size increases, rendering its two-thirds resources (i.e., Green and Blue cells) unavailable most of the time. While TRGB nodes still attempt reception within these resources, since most nodes cannot use them, there are no packets to receive and the reception period ends prematurely, resulting in a much lower duty cycle. Thus, TRGB's low duty cycle does not indicate efficiency but rather highlights its ineffective operation.

VI. Conclusion

This work proposed *Quick6TiSCH* to accelerate the formation of 6TiSCH networks. *Quick6TiSCH* assigns higher priority to messages critical for network formation depending on the state of each node, and enhances efficiency by mitigating collisions and congestion during network formation. Through implementation on real embedded devices and extensive evaluation on a sizable testbed, we have successfully validated the effectiveness of *Quick6TiSCH* in significantly reducing network formation time. These findings contribute to comprehensive understanding and optimization of 6TiSCH network formation, enabling faster and more reliable deployment.

References

[1] 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, 2019.

[2] R. Heile, R. Alfvin, P. Kinney, J. Gilb, and C. Chaplin, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," IEEE, Tech. Rep., 2012.

[3] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012.

[4] H.-S. Kim, J. Ko, D. E. Culler, and J. Paek, "Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, 2017.

[5] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load Balancing under Heavy Traffic in RPL Routing Protocol for Low Power and Lossy Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, Apr. 2017.

[6] C. Orfanidis, P. Pop, and X. Fafoutis, "Active Connectivity Fundamentals for TSCH Networks of Mobile Robots," in *18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2022, pp. 191–198.

[7] A. Elsts, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "TSCH Networks for Health IoT: Design, Evaluation, and Trials in the Wild," *ACM Trans. on Internet of Things*, vol. 1, no. 2, pp. 1–27, 2020.

[8] Z. Zhang, S. Glaser, T. Watteyne, and S. Malek, "Long-Term Monitoring of the Sierra Nevada Nnowpack Using Wireless Sensor Networks," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 185–17 193, 2020.

[9] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM conference on embedded networked sensor systems (SenSys)*, 2015, pp. 337–350.

[10] S. Kim, H.-S. Kim, and C. Kim, "ALICE: Autonomous Link-based Cell Scheduling for TSCH," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN)*, 2019.

[11] S. Kim, H.-S. Kim, and C.-k. Kim, "A3: Adaptive Autonomous Allocation of TSCH Slots," in *International Conference on Information Processing in Sensor Networks (IPSN)*, 2021, pp. 299–314.

[12] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-Demand TSCH Scheduling with Traffic-awareness," in *IEEE Conference on Computer Communications (INFOCOM)*, 2020, pp. 69–78.

[13] 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.

[14] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 459–464.

[15] H. Kim, G. Lee, J. Shin, J. Paek, and S. Bahk, "Slot-size Adaptation and Utility-based Packet Aggregation for IEEE 802.15.4e Time-Slotted Communication Networks," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 382–16 397, May 2024.

[16] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8180

[17] H. Kim, H.-S. Kim, and S. Bahk, "MobiRPL: Adaptive, robust, and RSSI-based mobile routing in low power and lossy networks," *Journal of Communications and Networks*, vol. 24, no. 3, pp. 365–383, 2022.

[18] M. Park, G. Jeong, O. Gnawali, and J. Paek, "RPL Objective Function for Multihop PLC Network," *Journal of Communications and Networks*, vol. 25, no. 1, pp. 131–139, Feb. 2023.

[19] C. Vallati, S. Brienza, G. Anastasi, and S. K. Das, "Improving Network Formation in 6TiSCH Networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 98–110, 2018.

[20] M. Vučinić, T. Watteyne, and X. Vilajosana, "Broadcasting Strategies in 6TiSCH Networks," *Internet Technology Letters*, vol. 1, no. 1, 2018.

[21] A. Kalita and M. Khatua, "Channel Condition Based Dynamic Beacon Interval for Faster Formation of 6TiSCH Network," *IEEE Transactions on Mobile Computing*, vol. 20, no. 7, pp. 2326–2337, 2020.

[22] ——, "Opportunistic Transmission of Control Packets for Faster Formation of 6TiSCH Network," *ACM Transactions on Internet of Things*, vol. 2, no. 1, pp. 1–29, 2021.

[23] ——, "Time-Variant RGB Model for Minimal Cell Allocation and Scheduling in 6TiSCH Networks," *IEEE Transactions on Mobile Computing*, 2023.

[24] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Hybrid Timeslot Design for IEEE 802.15.4 TSCH to Support Heterogeneous WSNs," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1–7.

[25] J. Park, H. Kim, H.-S. Kim, and S. Bahk, "DualBlock: Adaptive Intra-Slot CSMA/CA for TSCH," *IEEE Access*, vol. 10, 2022.

[26] J. Ko, J. Jeong, J. Park, J. A. Jun, O. Gnawali, and J. Paek, "DualMOP-RPL: Supporting Multiple Modes of Downward Routing in a Single RPL Network," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 2, pp. 39:1–39:20, Mar 2015.

[27] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," *RFC 6719*, Sep 2012.

[28] P. Levis and T. H. Clausen, "The Trickle Algorithm," *Internet Eng. Task Force, RFC 6206*, Mar. 2011.