© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This is the accepted version of the following article:

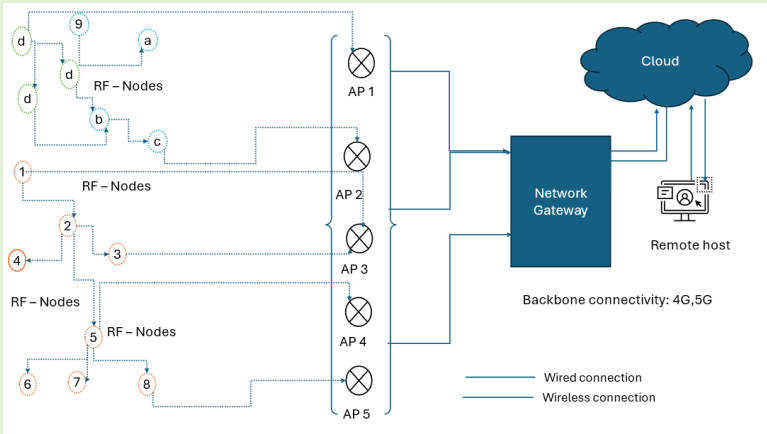
Kumar, K., & Kolberg, M. (2026). Hybrid Scalable Scheduling Reservation Protocol for Industrial IoT Networks. IEEE Sensors Journal. Advance online publication.

The final published version is available in IEEE Xplore:
https://doi.org/10.1109/jsen.2026.3713924

Hybrid Scalable Scheduling Reservation Protocol for Industrial IoT Networks

K. Kumar, and M. Kolberg, Sr., *Member, IEEE*

***Abstract*—IoT communication technologies are advancing with spectrum sharing and cooperation, with frequency hopping and enhanced communication reliability. These advancements play a crucial role in Industry 4.0. However, communication failures due to multipath fading and wireless interferences pose challenges, particularly in dense wireless sensor networks, leading to increased packet loss. Consequently, existing standards such as ZigBee, Bluetooth Low Energy (BLE), WirelessHART, and ISA 100.11a face challenges related to poor reliability and interoperability. 6TiSCH is an advanced wireless communication standard, allowing industrial sensor devices to interact seamlessly. This standard implements the Time-Slotted Channel Hopping (TSCH) protocol, which tackles multipath-fading and External Interferences (EIs) using channel-hopping capability. 6TiSCH offers a scalable environment using IPv6 over Low Power Wireless Personal Area Network (6LoWPAN), easing the strain on the underlying radio technology (IEEE 802.15.4). However, scalability has remained an open challenge with the existing 6TiSCH Scheduling Functions (SFs) posing trade-offs between reliability, energy consumption, and latency. In this paper, we present a Hybrid Scalable Scheduling Reservation Protocol (SSR), addressing important challenges with scalability, reliability, latency, and energy efficiency. The proposal, with its cake-slicing technique, network depth mapping, efficient traffic adaption, and congestion mitigation, shows potential for significant improvements over existing solutions like Minimal Scheduling Function (MSF) (RFC 9033). The experimental results show significant improvements over fixed hysteresis-based scheduling policies of MSF and demonstrate scalable performance using hundreds of IoT nodes within a subnet.**



***Index Terms*— Wireless Sensor Networks, IEEE 802.15.4, IPv6, 6LoWPAN, TSCH Scheduling, 6TiSCH Architecture**

I. Introduction

THE Internet of Things (IoT) is a paradigm, allowing data gathering using sensors and actuators on a real-time basis. From smart homes to smart cars, the applications of IoT are increasingly becoming popular [1]. Ericsson's report predicts that by 2029 there will be 38.9 billion connected IoT devices globally. Of these, 32.3 billion are expected to be short-range IoT devices [2]. This significant growth is driven by the enhanced capabilities of wireless networks, enabling spectrum sharing and cooperation with frequency diversity. Using these enhancements, IoT navigates key areas of the 4th industrial revolution (Industry 4.0) contributing to improved production and reduced cost of operation via automation. However, reliability cannot be guaranteed using resource-constrained IoT devices due to path loss, caused by wireless interferences and multipath-fading. While channel hopping addresses multipath fading with rapid switching of channels between transmitter and receiver [3], for industrial networks, Low Power Wide Area (LPWA) technologies cannot guarantee the required Quality of Service (QoS). This includes an acceptable latency with a threshold of about 1s (RFC 5673 [4]), and a high reliability. Traditional Low Power-Wireless Personal Area Network (LP-WPAN) technologies such as ZigBee [5], BLE [6], and ISA 100.11a [7] commonly use the IEEE 802.15.4 standard. These solutions use primitive channel-access protocols that do not implement channel-hopping, which leads to communication failures due to poor resilience to external interferences and multipath-fading. WirelessHART [8] implements channel-hopping. However, communication failures still occur due to the limited MTU capacity of IEEE 802.15.4 combined with a heavy User Datagram Protocol (UDP) payload of 127 bytes. To overcome payload interoperability and to improve performance, IPv6 Low Power Wireless Personal Area Network (6LoWPAN) [9] provided with compression and payload adaptation capabilities, was introduced.

In 2013, the Internet Engineering Task Force (IETF) and Institute of Electrical and Electronics Engineers (IEEE), jointly

K. Kumar., is with University of Stirling, Stirling, Scotland, FK9 4LA, United Kingdom. He is a tutor, and a PhD student in the Division of Computing Science and Mathematics, University of Stirling, UK(e-mail:kaushal.kumar@stir.ac.uk).

M. Kolberg is with the Faculty of Natural Sciences, University of Stirling, UK, (e-mail:mario.kolberg@stir.ac.uk).

rolled out Time-Slotted Channel Hopping (TSCH) over IEEE 802.15.4 as a MAC layer protocol [10]. This followed the introduction of the Routing Protocol for Low-Power and Lossy Networks (RPL) [11]. TSCH combines Time Division Multiple Access (TDMA) [12] and Frequency Division Multiple Access (FDMA) with channel hopping capability. The radio of IEEE 802.15.4 can "hop" about 1600 times per second among different channels with an approximate time slot duration of 10 ms, and 16 total channels within 2.4 GHz, each maintaining a coherent separating bandwidth (5 MHz). However, the implementation of TSCH has been hindered by scheduling issues. The reason is that the nodes reassess their routing position frequently seeking the shortest path to the root. This means the demand for TSCH cells per node is difficult to predict in advance using fixed patterns. This leads to an under- or over consumption of link-layer resources depending on traffic conditions. To improve communication performance, the 6TiSCH Operation (6Top*)* layer [13] was introduced to better integrate TSCH with the rest of the IPv6 LoWPAN stack. The resultant stack was named IPv6 over TSCH mode of IEEE 802.15.4e (6TiSCH) [14].

6TiSCH scheduling has evolved over time with the SF instructing the 6Top layer to perform operations such as adding or deleting a TSCH schedule. However, existing SFs are not sufficiently scalable and incur a trade-off between reliability, latency and battery life [15], especially in heavy traffic conditions [16]. Therefore, a new scheduling proposal is necessary to boost scalability and improve overall communication performance.

A. Contribution

This article introduces the Hybrid Scalable Scheduling Reservation Protocol (SSR) for 6TiSCH networks. The proposed solution addresses the scalability challenge without compromising reliability, latency, and energy efficiency. Our solution delivers scheduling with significant improvements compared to Minimal Scheduling Function (MSF), a recent scheduling draft (RFC 9033) [17]. We evaluated SSR using the 6TiSCH simulator [18] using varying traffic conditions and topologies. The findings indicate a lower latency (1s) and 99% reliability for networks of up to 100 nodes using a heavy traffic load, as well as enhanced battery life across different topologies. This article makes the following contributions:

1) **Fixed Adaption Delay**
   SSR rejects the concept of nodes having to wait for the fixed adaption window before adapting traffic. It allows packets to be assembled and fast-forwarded.
2) **Reliability and Battery Life**
   SSR ensures higher reliability by employing its network depth strategy (NDS) and dynamic traffic adaption strategy (DTS). That is, fewer relay nodes are given extra cells, leading to improved performance and battery life.
3) **Queue Optimization**
   SSR provides a dynamic packet aggregation approach for balancing queue conditions by implementing NDS and analytical measures. This allows the nodes closer to the root avoiding potential packet loss due to congestion without compromising network stability.
4) **Scalability**
   SSR delivers an improved communication performance for larger networks, leading to an improved scalability both in terms of network density and packet generation intervals.

The remainder of this paper is organized as follows: Section II describes the architecture reference model of 6TiSCH. Section III provides a thorough literature review of 6TiSCH scheduling solutions. Our proposal is then introduced in Section IV. The performance evaluation is presented in Section V benchmarking the results against MSF using varying packet generation intervals and deployment scenarios, followed by the scalability analysis. Section VI concludes the article and outlines future research directions.

II. 6TiSCH Architecture

The network architecture of 6TiSCH is illustrated in Fig. 1. The sensor nodes (motes) are labelled A-E in the hierarchical Wireless Sensor Network (WSN) tree. Each node implements the 6TiSCH protocol stack depicted on the right in Fig. 1. The nodes are battery powered sensors for data collection operating with small amounts of memory and processing capacity. The collected data is forwarded to the root node, which is responsible for storing the routing table and processing path computation algorithms. The transition of the data stream from the root node to a remotely connected application server necessitates internet connectivity, achieved by the 6LoWPAN Broader Router (6LBR). Consequently, remote applications can access the sensor readings on a real-time basis.

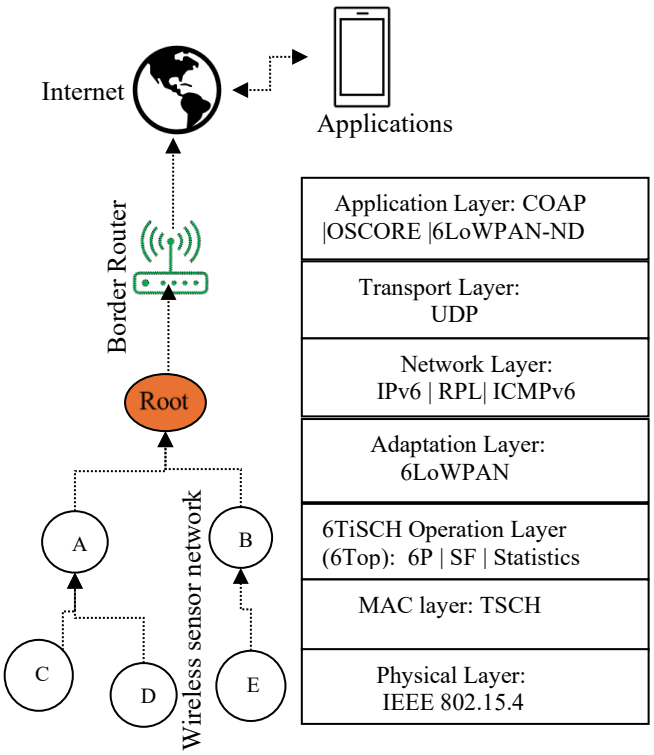


Fig. 1 The 6TiSCH Architecture reference model.

The application layer implements Constrained Application Protocol (CoAP) [19], which is a lightweight version of the Hypertext Transfer Protocol (HTTP) [20], which can run on sensor platforms. The protocol implements a RESTful approach allowing sensor nodes to sleep whenever there is no activity. This feature contributes towards enhanced power-savings. The application layer further implements Object Security for Constrained RESTful Environments (OSCORE) [21], which is a lightweight security protocol. The 6LoWPAN Neighbour Discovery (6LoWPAN-ND) [22] protocol exercises an additional control over adaptation and compression. It avoids multicasting or broadcasting to locate a node in a different subnet headed by 6LBR.

The transport layer implements UDP for end-to-end delivery of CoAP packets.

The network layer consists of IPv6 for addressing, RPL for routing, and Internet Control Message Protocol for IPv6 (ICMPv6) for reporting. The IPv6 addresses are compressed into local prefixes and later appended to the packet's header [23] by the root.

RPL (RFC 6550) is a lightweight protocol designed for routing over Low Power and Lossy Networks (LLNs). It organizes sensor nodes into a Destination Oriented Dynamic Acyclic Graph (DoDAG) Cluster Tree [24] where each node is assigned a rank based on the underlying link metrics, computed by the Objective Function (OF). Here, a rank value is a two-byte counter that increases and decreases following changes in the physical environment, and depicts distance in terms of number of hops to the root.

To keep the topology operational, RPL employs four main types of control messages: DIO (DODAG Information Object), DIS (DODAG Information Solicitation), DAO (DODAG Advertisement Object), and DAO-ACK (DODAG Advertisement Object Acknowledgement). As a new node joins the network, it connects to a parent node and then derives the rank from the parent's rank, limited by the maximum rank increase threshold (256). The role of DIO is to keep the network topology up to date, expected to be broadcasted at a defined probability (a value between 0.0 - 1.0). DIS beacons are used by the joining node for information solicitation prior to being assigned a rank. DAO messages are used by a source node to propagate or advertise path information upwards through the root using unicast messaging. DAO-ACK is used to acknowledge delivery of a DAO message.

RPL operates using one of its two pre-defined Mode of Operations (MoPs). Using the non-storing mode, sensor nodes initially send the data packets to the root node using forwarding instructions. The routing nodes are required to update their status regularly to maintain path accuracy, using unicast DAOs. The downside of this approach is that a longer path translates into a large routing header, which is attached to the packet before travelling downwards in the routing hierarchy. Using the other mode, storing mode, nodes are allowed to store routing information. This *source* routing is out of scope for this article because it requires the frequent broadcasting of messages leading to significant overhead. Currently, interoperability between MoPs is an open challenge [25].

RPL implements 'Trickle Timer' [26], introduced by IETF for suppression of control overheads and manipulation of Expected Duty Cycle (EDC) [27]. This mechanism is more suitable for synchronized networks because in asynchronized networks the DIOs are sent less frequently, making suppression unworthy.

The adaptation layer uses 6LoWPAN to compress IP headers (40 bytes) and divide a heavy UDP payload (approximately 1024 bytes) into smaller fragments based on the MTU size of IEEE 802.15.4 (127 bytes) [28]. On the receiving side, the original packet is re-formed using the same mechanism. Originally, fragmentation and assembly required each forwarding node to wait for the entire packet to be assembled before relaying new fragments to the next hop neighbor. This caused an unnecessary delay in forwarding an IPv6 packet. Virtual Reassembly Buffer (VRB) (RFC 8930) was introduced in 2020 by *Bormann et al.,* [29]. With this approach, the forwarding node can send a data packet without having to wait for the rest of the payload, optimizing the queuing delay. However, the assembly buffer can only store a maximum of 2 UDP packets at a time, causing packet loss when more than one node is sending packets concurrently [30]. The compressed 6LoWPAN fragments arrive at the 6Top layer. This layer collects network statistics often utilized by the scheduling function (SF). 6P [31] is a replication of 6Top, designed to provide flexibility with the TSCH-led scheduling designs.

The MAC-layer implements the TSCH protocol, which is a half-duplex scheduler. Its main responsibility is to maintain the scheduling module using *time cycle* and *channel-hopping* across network-wide communication. The channel-hopping in IEEE 802.15.4e is performed using Equation 1:

$$\mathrm{CH} = \mathrm{macHoppingSequenceList}\,(\mathrm{COUNTER}\ \%\ \mathrm{macHoppingSequenceLength}), \text{where}$$
$$COUNTER = ASN + Channel\ Offset \quad \ldots\ldots\ldots\ldots(1)$$

Equation 1 provides rapid switching of frequencies using the unique identifier Absolute Serial Number (ASN), which is increased by 1 each time a slot has elapsed. This is because the TSCH slotframe repeats itself over time. Typically, each slotframe is comprised of 101 slots and each slot is roughly 10 milliseconds long. Hence each slotframe cycle is roughly 1 second long. The Channel Offset represents a unique carrier frequency and macHoppingSequenceLength represents the channel density (maximum number of channels). By default, there are 16 channels available within the 2.4 GHz using IEEE 802.15.4e. A combination of *time slot* and *channel offset* forms a TSCH *cell*, which can be installed as a *hard* or *soft* cell. A *hard* cell is only configured once and cannot be added or deleted at run-time. A *soft* cell can either be installed as shared, or as a dedicated cell to a node for traffic adaption or managing network dynamics. However, collisions can still occur due to clock inaccuracy, which is an open issue.

The Physical layer is responsible for end-to-end radio transmission using IEEE 802.15.4. This standard, by default, uses Offset Quadrature Phase-shift Keying (O-QPSK) for modulation, Cyclic Redundancy Check (CRC) encoding for error detection, and radio module using 2.4GHz.

This article primarily focuses on scheduling functionalities implemented at the 6Top layer. In the next section, we review centralized, distributed, and hybrid scheduling functions.

III. Literature review

This section provides a comprehensive review of 6TiSCH scheduling capabilities based on the unicast-based (dedicated), broadcast-based, and hybrid scheduling SFs. The goal is to identify potential scheduling inconsistencies, which would limit the scalability of the 6TiSCH standard.

A. Unicast-based SFs

The term '*Unicast*' in TSCH scheduling refers to one-to-one communication where one cell is equal to 1 node. Such cells are dedicated to particular nodes and cannot be shared with any other node in the network. A dedicated schedule is contention-free with a sensor node either transmitting or receiving in a single timeslot.

Centralized scheduling benefits from one-hop signaling between the node and the root node that can compute an optimal schedule using the Path Computation Element (PCE). The PCE is a statistical measure helping to achieve an optimal schedule mapping. However, limitations are caused by change events triggering a node to relocate its entire schedule (cell bundle) to a new parent. While the routing topology is not static and overprovisioning is not provided in the PCE-based scheduling, connected nodes struggle to deal with heavy temporary traffic (bursty traffic) as each 6Top request requires separate signalling, which limits scalability.

Palattella et al., [32] proposed Traffic-Aware Scheduling Algorithm (TASA). TASA organizes nodes into a star topology where scheduling is performed by signaling. It uses *color-matching* patterns through the graph where each node directly engages with the root to add or delete cells. However, TASA has failed to address packet loss in densely populated networks of 100 nodes [16]. It generates a heavy signaling overhead, which not only limits the goodput but also limits scalability.

Faras & Dujovne [33] introduced a scheduling technique, which incorporates PCE in a model where multiple RPL instances are generated concurrently. This approach exploits a linear relationship between routing topology and schedule compactness. It provides a pre-computed schedule with each node directed by the central entity regarding the specific timing and channel allocation for transmission and reception. This approach too, suffers from high signalling overheads, imposing scalability limitations.

Distributed/Decentralized scheduling has played a crucial role in improving scalability. Such approaches use negotiations between nodes and use free cells through "bargaining", mediated by the 6Top layer. Several dedicated cells can be scheduled at a time by a node. The unicast-led distributed SFs have advantage of collision-free access and timely delivery. Consequently, a large number of contributions have adopted this idea [34].

Dujovene et al., [35] proposed Scheduling Function Zero (SF0), which provides extra cells to nodes based on the link quality metric. The physical layer provides two key link quality metrics including Received Signal Strength Indicator (RSSI), and Packet Delivery Ratio (PDR). SF0 allocates a schedule based on link metrics performance while monitoring each link passively (prior to providing extra cells). However, SF0 is a load-sensitive solution that drops packets in heavy traffic conditions.

Watteyne et al., [36] extended SF0 by proposing On The Fly (OTF) bandwidth reservation allowing a fixed number of cells to be added and deleted. The downside of this approach is that it causes recurrent wastage of cells especially with a higher threshold limit (a fixed set of cells) and a large number of collided cells. Conversely, a lower threshold value might cause packet loss due to poorly predicted traffic on the node. Hence, OTF potentially under- or over-estimates slot distribution. A strength of this approach is the low complexity. It has attracted a large number of contributions attempting to optimize the scheduling resources.

Chang et al., [37] proposed Low Latency Scheduling Function (LLSF) that implements a 'daisy-chaining' technique, allowing connected nodes to schedule transmission soon after reception. The algorithm filters the slotframe with low-latency slots. However, if a node fails to add a slot or there are no free slots available to add, the entire scheduling process is disrupted. LLSF does not scan poorly performing links or selects shorter links (closer in the slotframe). Longer links are prone to an increased delay [38]. Since a TSCH slotframe repeats itself, LLSF causes recurrent wastage due to fixed overprovisioning each time a node wants to add an additional cell [16]. This process exacerbates resource consumption under heavy traffic conditions, leading to the availability of fewer cells for the other nodes in the slotframe.

Daneel et al., [39] proposed Recurrent Scheduling Function (ReSF), aimed to address recurrent wastage using a separate traffic management algorithm. This algorithm reserves a series of slots along the path to the root. That way it is able to predict the success or failure of sending a message in advance. However, this could not be accomplished without monitoring of queue and link metrics predicting the relay between node and its neighbours. This approach suffers from high control overheads and packet loss under challenging traffic conditions [16]. And so, while this solved the wastage problem, the energy consumption remains at a same level as SF0 [35].

Duy et al., [40] proposed Enhanced SF0 (ESF0), which uses a 'best portioning' approach. This divides the slotframe into a number of equal-sized portions corresponding to the size of the TX buffer. Each portion then maintains a 'density' value indicating the maximum number of free slots available. To avoid collisions, a 'channel-change' algorithm is introduced, which continuously monitors the link's stability. If a channel exhibits a sustained poor PDR, ESF0 replaces this with a new channel. However, packets are dropped in heavy traffic conditions because dedicated slots are limited and ESF0 does not adapt traffic dynamically. ESF0 is criticized for inadequate portioning of the slotframe due to its uniform allocation.

Prieto et al., [41] is another threshold-based SF inspired by Proportional-Integral-Derivative (PID). In the context of 6TiSCH scheduling, this mechanism calculates an error value as a difference between actual traffic and expected traffic and adjusts allocation of cells based on the error rate. While it improves reliability in heavy traffic, the algorithm does not promise an interference-free communication [16]. The interferences remain due to overlapping channels between 6TiSCH and coexisting networks such as Wi-Fi 802.11 [42]. In addition, monitoring queue conditions actively impedes energy-savings for transparent allocation [16].

Wang et al., [43] proposed Enhanced Distributed Scheduling

Function (EDSF). This approach primarily focuses on how cells must be selected based on the transmission capability assigned by EDSF. This is another way of monitoring traffic through the slotframe [16]. EDSF assigns a TX probability value (PDR) per cell, however it relies on dedicated cells for both managing traffic and network topology, causing goodput loss. Collisions can also still occur due to poor synchronization of nodes accessing the same cells during high slotframe occupancy. This approach also requires frequent housekeeping to avoid network bottlenecks in challenging traffic conditions. The authors did not test scalability for dense and larger networks.

B. Broadcast-based SFs

6TiSCH offers an open access plan [44] to replace wired-like connectivity with a densely populated multi-hop IPv6 WSN, achieved by implementing a Decentralized, Broadcast-based Scheduler (DeBRaS) [45]. This scheduler offers an improved propagation with reference to randomly assigned broadcast cells. Single broadcast cells are not sufficient for a network-wide coverage [45], and poor propagation means fewer routes in the network. However, collision occurs frequently with DeBRaS when more than one node accesses the same cell, caused by clock drift [38]. If this happens the packet will be retransmitted via another route. The key limitation of DeBRaS-based scheduling is that as more broadcast cells are added, nodes rapidly consume battery current [45], leading to a shorter battery life. Currently, energy consumption is an open issue [45]. *Municio et al.,* [45] suggest an optimal limit for broadcast cells of 3 cells. Many SFs have adopted this limit.

Accettura et al., [46] propose Decentralized Traffic Aware Scheduling Algorithm in 6TiSCH Networks (DeTAS), an extension of TASA. Each node observes an even-numbered slot for the outgoing traffic and an odd-numbered slot for the incoming traffic, making an important distinction between packet payloads. In DeTAS, traffic burst is generated locally, while packets are upstreamed through the DeBRaS scheduler. This reduces dependency on the root. However, DeTAS faces criticism due to a high volume of collided cells and increased overheads, stemming from insufficient gaps between TX slots. Consequently, it causes congestion due to overlap within frequencies as more nodes are added in the network, hence DeTAS is only an effective solution for smaller or medium-sized networks with lower traffic volumes [47].

Kralevska et al., [48] propose Local Voting (LV), a graph-based approach that organizes the routing topology into a graph, enforcing $G= \{VE\}$ where V represents the set of nodes and E the connecting edges. While the graph theory may be seen as complex, these are amongst the best performing approaches exploiting multipath delivery. LV divides the traffic load equally amongst the nodes to minimize latency by simply monitoring queue length of the parent node. This provides an estimate of packets in the buffer of each node. If the load is unequal, the approach adds or deletes cells based on the connecting edges (E). Packets take multiple paths to the root. Generally, the queue length for all nodes is set to 100 packets [49]. The author of LV [48] did not specify an overprovisioning limit to deal with uncertain traffic conditions. In an extended version [50], the author used overprovisioning as a fixed threshold (the assumed maximum limit is 10 cells). However, both versions of LV lack traffic adaption and incur packet loss as the uniform distribution of cells using the pre-estimated threshold provides no concession toward optimizing bandwidth consumption.

Righetti et al., [51] proposed an Extension of OTF (E-OTF). It is used in both unicast and broadcast-based scheduling operations. E-OTF, instead of allocating cells to individual nodes, uses links. The algorithm schedules cells, which translates the traffic prediction along the root in advance. While taking into account factors like a node's duty-cycle (with respect to the slot occupancy) and queue status with congestion bonus, the author poorly defines these measures necessary to address temporary peaks of traffic. E-OTF also ignores poor performing links while making reservation along the root.

Wang and Fapojuwo [52] proposed Hysteresis-Free-OTF (HF-OTF), a decentralized and broadcast-based solution using OTF without any fixed threshold limit to meet traffic demand in the network. It autonomously computes overprovisioning cells per node. However, the proposal monitors queue occupancy to predict cells required by a node to successfully dispatch its payload. HF-OTF inherits OTF's and E-OTF's shortcomings in terms of energy-efficiency because the charge consumption is still high, even though it allows fewer cells at some nodes than others.

Our previously developed SF called Decentralized Broadcast-Based Scalable Scheduling Reservation Protocol (DeSSR) [53] offers an advantage in terms of improving scalability with comparatively low cell consumption when compared with DeBRaS-led SFs. However, energy consumption remains a challenge.

C. Hybrid SFs

Hybrid scheduling incorporates more than one scheduling design. Autonomous schedulers offer a competitive advantage in terms of reducing overheads. This includes EBs and routing beacons. On the other hand, negotiation-based, decentralized schedulers perform traffic adaption more efficiently.

Duaquennoy et al., [54] proposed *Orchestra*, which uses an autonomous slot. These slots are all time active slots, pre-computed in the slotframe. The autonomous cells follow traffic conditions (incoming or outgoing) on a node. Currently, this is a preferred approach for managing network dynamics for load-sensitive IPv6 subnets, extended by a number of SF. However, there are some common weaknesses with autonomous scheduling, in particular poor energy-efficiency and high latency [55].

Autonomous Link-based Cell Scheduling (ALICE) [56] performs link-based scheduling and out-performs *Orchestra* significantly. It adapts the traffic based on the topology hence allowing more cells to meet the demand. However, how many cells are sufficient is not defined by ALICE. The approach suffers from packet loss in variable traffic conditions [52].

Pratama et al., [57] proposes a Reinforced Learning (RL)-based SF employing machine-learning techniques. It extends the autonomous scheduling by studying the TX slots with the lowest transmission failure rates and selecting only those slots with a high PDR for the next set of packets to be transmitted. The solution relies on pre-computed cells, continuously active in every slotframe. This algorithm requires the accuracy of previously used slots (communication exchanges). Further it assumes a pre-configured schedule allocation, thus ignoring

sudden changes in traffic conditions based on displacements of nodes in routing tables and parent-changes, which triggers relocation of the entire cell bundle [16]. Furthermore, RL-SF is not an energy-efficient solution [16].

Traffic Aware Elastic Slotframe Adjustment (TESLA) [58] extends Orchestra [54] using RX-based slots adjusting the slotframe based on incoming traffic. It employs a piggyback technique to reduce overheads. However, it is not an overhead free approach since it utilizes 6top negotiations [31].

Chang et al., [17] proposed MSF (RFC 9033) that adopts autonomous slots with OTF using fixed threshold-based overprovisioning. MSF is a generic solution that uses a hash function to compute slot offsets from an Extended Unique Identifier (EUI-64) of a node as shown below:

$$\text{slot} = 1 + \text{hash}(\text{EUI64}, \text{Length}(\text{slotframe} - 1) - 1) \ldots..(2)$$
$$\text{ch0ffset} = \text{hash}\ (\text{EUI64}, \text{ChannelDesity}) \ldots\ldots\ldots\ldots\ldots\ldots(3)$$

The rules of adding, deleting, and relocating a TSCH cell by MSF is subject to node's status: either a node is switching parent, or the node has been met with collisions in the negotiated slotframe, or the node requires more cells to deal with incoming and outgoing traffic.

For traffic adaption, each node maintains two counters; one for TX cell and one for RX cells. These counters are NUMCELLELAPSED (number of 6top transactions), and NUMCELLUSED (number of successful 6Top transactions).

TX and RX cells from the negotiated slotframe are dedicated unicast cells, which are only added or deleted when MAX_NUMCELL are equal to NUMCELLELAPSED and when NUMCELLUSED exceeds ¾ of MAX_NUMCELL. Here MAX_NUMCELL depicts the total volume of cells used by a node since the beginning of the slotframe cycle. It is by default set to 32 for distributed SF such as SF0, OTF, LLSF, etc. MSF would add or delete cells when MAX_NUMCELL = NUMCELLELAPSED. A cell is deleted if NUMCELLUSED < ¼ of the MAX_NUMCELL. MSF does not address traffic efficiently due to fixed limits specified for incoming and forwarding traffic.

Chang et al., [59] extends MSF to further improve traffic adaption by proposing a strategy that can add or remove multiple cells simultaneously. This leads to minimum delay as 6Top exchanges are fairly bulky. With more cells added at the same time, demanding nodes avoid triggering frequent request for addition or deletion of cells unless the estimate is biased. Notably, this solution for bursty traffic proposes MAX_NUMCELL = 4. For steady traffic the delay must equal to 32 cells (MAX_NUMCELL= 32). These limits, once declared, do not vary over time. Hence the scheduler uses these limits causing a potential mismatch with the actual traffic levels. Clearly, fixed thresholds do not work well as actual traffic is difficult to predict in a dynamic topology.

YSF [60] is an extension to MSF that uses cell selection as a means to improve the latency performance and efficient queue utilization. Cells are assumed to be closer together at the top-end of the slotframe. YSF requires more frequent housekeeping to mitigate collisions, which negatively affects energy consumption.

Valdovinos et al., [61] introduced Channel Ranking Scheduling Function (CRSF), which listens to channels while measuring the link-quality using RSSI and PDR in addition to signal to background noise on channel; this is performed by *Kamran* filter, used for channel-selection by predicting channel quality or availability of communication channels over time. In terms of traffic adaption, this solution monitors the total number of packets in the queue minus any scheduled payload (available cells by the node) prior sending for channel clearance. However, channel monitoring is delay sensitive, and the selection of the best available channel can further delay the packet arrival in the TX buffer. Furthermore, its active monitoring provokes rapid energy consumption.

Tanaka et al., [60] introduced SF-Fastboot as an extension of MSF. The proposal exploits the asymmetry between the root and sensors following the *SmartMesh IP* design [62], allowing shorter joining delay. This approach allows different beacon intervals for different nodes. A joining node will hear more frequently from the root as it is given a higher priority towards frequent beaconing. The overheads generated at the root are ignored because the root is assumed to be a powered device. SF-Fastboot faces notable challenges in the network due to scaling. Using unicast beacons, joining the network can take longer when actually the opposite is assumed for broadcast-based joining approaches. Further, the broadcast-based joining is prone to frequency overlapping and queue congestion. The packet loss is more pronounced at the early stages of network formation, which is exacerbated by SF-Fastboot.

IV. Proposed Solution

The literature review concludes that SFs using dedicated cells for both managing control traffic and data packets are well-suited for lightweight traffic scenarios offering better energy efficiency and enhanced battery life with minimal strain on network resources. Broadcast-based scheduling is scalable as long as it is feasible to replace batteries frequently. Hybrid scheduling, although complex, reduces control overhead while managing network topology and application data transfer. SFs incorporating autonomous or hybrid scheduling lack balance between frequent predictions and queue optimization. Thus, scalability remains an open challenge due to inaccuracy in deciding how many cells are needed per node or per link. Crucially, this is not defined by the TSCH scheduler. This complexity is heightened when using a single broadcast cell for propagation, leading to fewer routes in the network. Using these routes, the connected nodes aim to associate themselves with a parent whose distance is shorter than their existing parents, determined by link-quality metrics such as RSSI, and PDR. Generally, a neutral cell allocation to meet traffic demand results in poor performance. On the other hand, actively monitoring queue or cell status incurs a high volume of signaling overheads. Our proposal aims to predict the near-accurate estimates at runtime. The approach is a hybrid technique and employs an analytical measure and four strategies.

A. The Cake-slicing technique

Cake-slicing is an analytical measure, used to divide a finite resource into portions. The distribution follows the *hop-count* metric, defined by Objective Function Zero (OF0) in RPL. The

metric assigns each node with a two byte counter (rank), while organizing the connected nodes into a hierarchy. The rank follows a normal distribution with deviation of 256. Hence, the overall network depth represents the maximum value of rank, translated into maximum number of hops excluding outliers.

This technique takes two key variables: S represents the maximum slotframe length and N represents the number of iterations depending on network depth.

This algorithm for cake-slicing is shown in Figure 2. This algorithm is implemented at the root level, enabling easy access to the routing table. Furthermore, input parameters are computed by the root node without the need for extra signalling. The key statistics produced as output of the cake-slicing technique is accessed by each node via the 6Top layer. While, the cake-slicing is in use, the proposal does not monitor queue conditions, or cell lists, or any other node resources as this would require extra signalling and cause extra traffic in the network. The algorithm has a complexity of O(log N).

Algorithm 1 Cake-slicing

SLICER(S N)

1. Initializations
   $T \leftarrow N$
   $\beta \leftarrow 0$, $\alpha = S$
2. Loop while T>=1:
   - Compute: $O_T \leftarrow |S / T^2|$
   - Update S: $S \leftarrow \{ ((1 - 1/T) * S) + ((S/T) - O_t) \}$
   - Accumulate β: $\beta \leftarrow \beta + O_T$
   - Update T: $T \leftarrow T-1$
3. **Condition:**
   - **if** $\alpha < \beta$ **:**
     $O_t \leftarrow (O_t - (\beta - \alpha))$
   - Otherwise
     $O_t \leftarrow (O_t + (\alpha - \beta))$

Return O

Fig. 2. Cake-slicing algorithm.

The example below shows the step-by-step execution of the algorithm for N = 4 and S = 101. O represents a list of numbers after processing S, and N using T iterations. The variable α stores the initial value of S, which is compared with β (cumulative total of list O). The example computes O_T, updates S and β, before decreasing T by 1. This process repeats as long as T >= 1. Before providing a final output, it ensures the sum of all elements in O, denoted as β, must be equal to the original slotframe length (α). For this example, both α and β are 101. Typically, variance occurs based on the probability of obtaining non-integer values after divisions. The algorithm returns O as final output.

Step by Step Execution:

Iteration 1 (T = 4):

- Compute: $\boldsymbol{O_4} = \frac{S}{T^2} = \frac{101}{4^2} = 6$
- Update S: $\mathbf{S = (1 - \frac{1}{4})\, 101 + (\frac{101}{4} - 6)}$
  $S = 75.75 + (25.25 - 6) = 95$
- Update T and β: $\boldsymbol{\beta = 6}$, $T = \mathbf{3}$

Iteration 2 (T = 3):

- Compute: $\boldsymbol{O_3} = \frac{S}{T^2} = \frac{95}{3^2} = \mathbf{10}$
- Update S: $\mathbf{S = (1 - \frac{1}{3})\, 95 + (\frac{95}{3} - 10)}$
  $S = 63.33 + (31.67 - 10) = 85$
- Update T and β: $\boldsymbol{\beta = 6 + 10 = 16}$, $T = \mathbf{2}$

Iteration 3 (T = 2):

- Compute: $\boldsymbol{O_2} = \frac{S}{T^2} = \frac{85}{2^2} = \mathbf{21}$
- Update S: $\mathbf{S = (1 - \frac{1}{2})\, 85 + (\frac{85}{2} - 21)}$
  $S = 42.5 + (42.5 - 21) = 64$
- Update T and β: $\boldsymbol{\beta = 16 + 21 = 37}$, $T = \mathbf{1}$

Iteration 4 (T = 1):

- Compute: $\boldsymbol{O_1} = \frac{S}{T^2} = \frac{64}{1^2} = \mathbf{64}$
- Update S: $\mathbf{S = (1 - \frac{1}{1})\, 64 + (\frac{64}{1} - 64)}$
  $S = 0 + (64 - 64) = 0$
- Update T and β: $\boldsymbol{\beta = 37 + 64 = 101}$, $T = \mathbf{0}$

Final Output : O= [64, 21, 10, 6]

B. Network Depth Strategy (NDS)

The NDS is used to translate a node's position into 'hop distance' based on the routing metrics of RPL (rank). The algorithm of NDS is shown in Figure 3.Here R is a variable representing next hop rank according to RPL, initialized to 512. $prev_hop_{rank}$ contains the rank value of the previous hop. It is initially assigned 256 for R being 512; The rank increases in line with the maximum rank increase threshold, by default set to 256 by RPL. The variable *hop* is assigned 1 and it increases iteratively. The network depth is represented by D, initially assigned 4. NDS operates based on the node's rank, calculated using RSSI and PDR. The next step repeats for as long as R is smaller than *maximum_rank* (in the rank distribution excluding outliers). If $prev_hop_{rank}$ is found to be less than rank of the node, which must be less than $R + (O_{hop})^{hop}$ then the loop terminates where this function returns *hop*. The value of *hop* increases by 1 under each iteration when the condition is not met. R is incremented by 256 (maximum increase threshold) and $prev_hop_{rank}$ becomes the sum of R and square of the corresponding value in O, indexed by *hop* identifier.

Algorithm 2 Network Depth Strategy

NDS (rank)

1. Initializations:
   R←512, hop←1, O ← Output list from algorithm1
   $prev_hop_{rank} \leftarrow 256$ D← network depth
2. Iteration Until Rank Condition is Met:
   While R < maximum_rank, perform:
   - if :
     $prev_hop_{rank} < rank < R + (O_{hop})^{hop}$
     Then, terminate loop.
   - Otherwise:
     $prev_hop_{rank} \leftarrow R + (O_{hop})^{hop}$
     hop← hop + 1
     R← R + 256
3. if hop > D
   Then, assign hop randomly between 1 and 4
4. Return hop

Fig. 3. NDS algorithm.

The step by step execution of the algorithm 2 is as follows:

Hop by Hop Execution:

Lets compute the hop distance using O= [64, 21, 10, 6] from cake-slicing:

hop = 1:

- $prev_hop_{rank}$ = 256
- O_1 = 64
- R = 512
- Rank interval= [257 … 512 + 64^1]
- If node's rank falls in this range, return hop = 1.

hop = 2:

- $prev_hop_{rank}$ = 512 + 64^1 = 576
- O_2 = 21
- R = 512+256= 768
- Rank interval = [577 … 768 + 21^2]
- If node's rank falls in this range, return hop = 2.

hop = 3:

- $prev_hop_{rank}$ = 768 + 21^2 = 1209
- O_3 = 10
- R = 768+256 = 1024
- Rank interval = [1210 … 1024 + 10^3]
- If node's rank falls in this range, return hop = 3.

hop = 4:

- $prev_hop_{rank}$ = 1024 + 10^3 = 2024
- R = 1024 + 256 = 1280
- O_4 = 6
- Rank interval = [2025 … 1280 + 6^4]
- If node's rank falls in this range, return hop = 4.

The example above illustrates NDS over 4 iterations. As the rank of a node falls within the rank interval, the function returns the corresponding hop distance of the node from the root. In a scenario when the hop value is higher than D, we will randomly assign a number from 1 to 4. This may occur when a node has subsequently joined the network or is leaving the network due to local repair in which the node assigns itself an infinitely higher number and leaves the network.

C. Cell Selection Strategy

CSS focuses on TX cell slot selection whereas RX slots are not added reactively. The strategy formulates subsequent transmission of received packets and avoids scanning the entire slotframe. The algorithm of CSS is shown in Figure 4. R is an input variable indicating the number of cells required. D represents network depth, S contains maximum slotframe length, O contains the output from cake-slicing algorithm, X indicates number of free slots, INDEX is a counter, increased to avoid overlapping between the portions when finding free slots. S_X indicates the number of free slots available in the slotframe and S^O_{INDEX} contains the number of slots in a controlled portion (driven by O), hop is assigned 1 and increases by 1 in each iteration of the loop.

Algorithm 3 Schedule Selection

CSS (R)

1. Initializations
   D←Network Depth, S← slotframe length
   O← Output from cake-slicing algorithm
   R← Required Slots, X ← free cell, hop← 1
   S_X← Selected free slots in a given portion
   INDEX← 1
   S^O_{INDEX}← controlled portions with starting index
2. Loop
   Do
   SELECT FREE CELLs X ? S^O_{INDEX}
   COUNTER ← COUNTER + X
   INDEX← INDEX + O_{hop}
   hop ← hop + 1
   While hop <= D or COUNTER < R
3. **If:**
   COUNTER >= R,
   Return S_X
   Otherwise:
   SELECT FREE CELLS X ? S
   Return S_X

Fig. 4.Cell Selection Strategy (CSS) Algorithm.

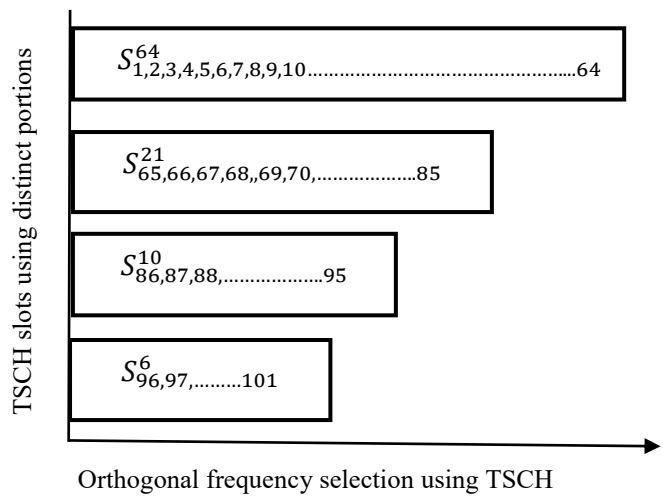


Fig. 5.Cell selection across negotiated slotframe using an example where O = {64,21,10,6}, S= 101, and D= 4.

The algorithm shown in Fig. 4 illustrates the TX slot selection using CSS. The search only expands when there are not enough free cells available in a controlled slotframe portion S^O_{INDEX}. The slotframe portions are visualized in the example in Fig. 5 where each portion contains an INDEX value indicating the starting slot number and it can not exceed the maximum number of slots allowed in O. For example for $O = 64$, the algorithm searches for a free slot from 1 to 64 slots. If the number of required slots (R) is not met, then the search expands through subsequent controlled portions as INDEX becomes a sum of total number of slots in the preceding slotframe portion. For $O = 21$, the porting starts from 65 and ends with 85. For $O = 10$, start and end points of the slotframe portion are 86 and 95, and finally for $O = 6$, the start and end points are 96 and 101. These are formulated based on the algorithm 1. In terms of channel selection, the TSCH protocol performs channel hopping between time slots, allowing nodes to avoid interference by not overlapping on the same carrier frequencies.

Once these free cells are located across the S^O_{INDEX}, the CSS algorithm terminates (the maximum number of free cells as *COUNTER* is greater than or equal to *R*). However, the

scheduler implements a *random* selection if the required number of slots (R) is not met and all portions are scanned.

D. Dynamic Traffic Adaption Strategy

DTS is an extension of MSF traffic adaption policy, which uses variables NUMCELLELAPSED and NUMCELLUSED to measure TX utilization. These variables are discussed in Section III in MSF's review. DTS offers more cells to the nodes within closer proximity to the root (for example hop 1) than to nodes located further away from the root. A node can seek to add or delete cells only when the adaption delay is equal to the number of times the NUMCELLELAPSED is updated for a given slotframe cycle.

This delay is characterized based on the cake-slicing output in such a way that hop 1 will have a lower adaption delay than hop 2, allowing nodes within the periphery of hop 1 to add and delete cells more frequently than that of hop 2. Hence, the traffic adaption is a function of adaption delay where computation of how many cells (X) to add or delete is performed separately by using algorithm 4, shown in Fig. 6. The algorithm predicts key estimates crucial to address congestion on particular nodes in response to fewer routes being provided by a single *minimal cell*.

The schedule allocation using DTS is purely a statistical exercise formulating DTS threshold as $(\boldsymbol{S - O_{hop_{id}}})/100$. This value is compared against the TX utilization threshold as $\frac{NUMCELLUSED}{NUMCELLELAPSED}$. The rationale behind this comparison is that NUMCELLUSED is typically lower compared to the NUMCELLELAPSED [17] and the utilization of TX cells can vary at different hop levels in the RPL topology. The algorithm only adds cells if the TX utilization threshold is higher than the DTS threshold. In that case, X is computed using the value for the corresponding hop in the output of cake-slicing, O, and dividing by 2 the bitwise right shift **(>>)** of 3. To prevent cell wastage, the algorithm deletes cells if the TX utilization falls below a fixed threshold of 0.25. The value of X is computed in the same manner as it is during cell addition.

Algorithm 4 Schedule Allocation

DTS (Node)

D←Network Depth, S← slotframe length
O← Output from cake-slicing algorithm
hop← **Node**'s hop distance from NDS
NUMCELLUSED ← number of successful TX cells
NUMCELLELAPSED ← number of total TX cells
X← NUMBER OF CELLS TO ADD

Condition for Adding TX cells:

If $\frac{NUMCELLUSED}{NUMCELLELAPSED} > \frac{(S-O_{hop})}{100}$, then:

$$X \leftarrow round\ (\frac{(O_{hop})\gg 3}{2})$$

Condition for Deleting TX cells:

If $\frac{NUMCELLUSED}{NUMCELLELAPSED} < 0.25$, then:

$$X \leftarrow round\ (\frac{(O_{hop})\gg 3}{2})$$

Update NUMCELLUSED and NUMCELLELAPSED

Fig. 6.Traffic adaption using DTS.

Example 1: Step by Step Execution of Algorithm 4:

- Slotframe Length: S= 101
- Network Depth: D = 4
- Cake-Slicing Output: O={64, 21, 10, 6}

Hop By Hop Schedule Allocation Decisions:

TABLE I
EXAMPLE 1 HOP BY HOP SCHEDULING DECISIONS

hop	TX utilisation threshold: $\frac{NUMCELLUSED}{NUMCELLELAPSED}$	DTS threshold: $\frac{(S-O_{hop})}{100}$	Decision
1	0.60	(101-64)/100 = 0.37	ADD
2	0.47	(101-21)/100 = 0.80	Do nothing
3	0.77	(101-10)/100 = 0.91	Do nothing
4	0.15	(101-6)/100 = 0.95	DELETE

Compute X to ADD/DELETE:

- **Hop 1** where 0.60>0.37**:** $X = \frac{(O_1)\gg 3}{2} = \frac{64\gg 3}{2} = \frac{8}{2} = 4$ (ADD)
- **Hop 2 and Hop 3:** $TX\ utilisation\ threshold$ for hop 2 = 0.47, and hop 3= 0.77. These values are not greater than the DTS threshold for hop 1 = 0.80, and hop 2= 0.91 respectively hence do nothing
- Hop 4 where 0.15 is not greater than 0.95 and it is less than 0.25 hence delete cells**:** $X = \frac{(O_4)\gg 3}{2} = \frac{6\gg 3}{2} = \frac{0.75}{2} = rounded(0.375) = 0$ (DELETE)

E. Queue optimization

Queue management often suffers from a concurrency risk, with multiple nodes sending packets to the same destination. Queue Optimization Strategy (QOS) aims to minimize the risk of concurrent forwarding by applying a Packet Aggregation Strategy (PAS) mechanism as a traffic soothing measure. The algorithm is shown in Figure 7.

Algorithm 5 SSR Packet Assembly Strategy

SSR_PAS (node, S, D)
D←PREDICT (D)
S← Maximum Slotframe Length
PP← node.preffered parent
rank← PP.rank
Hop_id← NDS(rank)
O← Output from cake-slicing algorithm
PKT← packets in parent node's buffer
IN← PKT.RX, FWD← PKT.TX, SELF←PKT.GEN
T← $\frac{(O_{hop_id})}{100}$
do
Assert node is not root
$\boldsymbol{TX_{QUEUE} \leftarrow FWD + (SELF \times T) + (IN \times 1 - T)}$
While nodes in PP list
Return PP. $\boldsymbol{TX_{QUEUE}}$

Fig. 7. Queue Optimization strategy.

Traffic Prioritization: Nodes closer to the root receive a larger proportion of incoming packets (IN) from neighbour nodes, generate comparatively less traffic themselves (SELF), and their traffic includes a larger proportion of forwarded packets (FWD). Here both FWD and SELF packets are assigned TX cells while receiving packets are using RX cells. Conversely, at hop 2, nodes are expected to attract even more

incoming (IN) than hop 1 and generate less of SELF packets and have smaller number of forwarding traffic due to availability of more routes. As the hop distance increases, nodes are programmed to allow most of the incoming payload straightaway and add less of SELF and FWD payload.

Threshold Calculation: Each hop in the network employs PAS to determine its Packet Aggregation Threshold (T). This threshold, ranging from 0.00 to 1.01, represents a percentage share of the slice (O) allocated to that hop. The algorithm adjusts the threshold dynamically according to network conditions and traffic load. The algorithm shown in Figure 7 computes threshold limit T for each hop using PAS. It overcomes some of the shortcomings of RPL routing by allowing load-balancing under challenging traffic conditions.

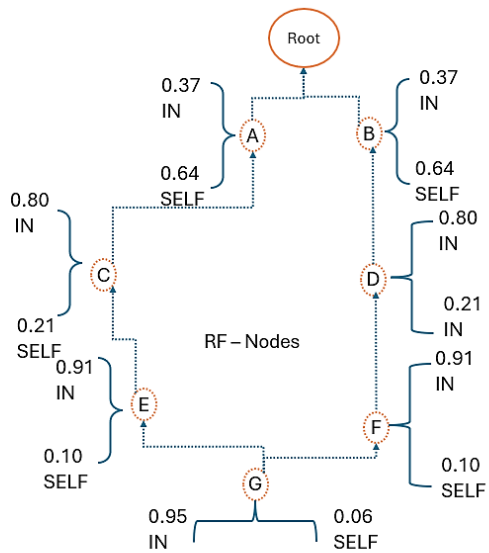


Fig. 8. Example of Packet Aggregation at various hops in the routing topology. Example shows overall payload assembly in the TX queue where maximum slotframe length (S)= 101, and the depth of network topology (D) = 4.

The diagram in Figure 8 shows PAS articulation at different level of RPL hierarchy: Node A and Node B at hop 1 must relay 0.37 of the incoming payload (IN) and 0.64 from their own (SELF). Since the SELF payload may contain beacons to maintain clock synchronization and keep the topology operational, it is vital to prioritize such payload at this level (closer to the root). Node C and Node D relay 0.80 of IN packets and 0.21 of SELF packets. The topology is more populated and dense at this stage, and nodes are benefitting from MSF-like traffic adaptation, allowing a maximum of 1 additional cell if the TX utilization increases rapidly. Further, Nodes E and Node F at hop 3 may receive the Enhanced Beacons (EBs) with a delay compared to Node C and Node D, causing frequent clock drift; hence, Node E and Node F relay 0.91 of their IN packets and 0.10 of their SELF packets. Lastly, Node G is the most distant node, which moves less frequently in the topology due to increased distance from the root. It may have to reset its clock each time it synchronizes with the network after leaving and rejoining. At hop 4, PAS is upstreaming 0.95 of IN packets and 0.06 of SELF payload.

V. Performance Evaluation

The 6TiSCH simulator [18] is specifically designed for testing larger IPv6 WSN, allowing variations in traffic conditions and deployment topologies making it an ideal environment for our experimentation. This section evaluates SSR and uses MSF for benchmarking. Each experiment simulates the functionalities of OpenMote CC2538 [63]. This device embeds a *temperature sensor*, *motion detector* and *light sensor*. The device uses a 32 bit Cortex-M3 Microcontroller, using clock frequency of 32 MHz, and a maximum drift time of 40 ppm (parts per million) [64]. Once the nodes have joined the network, Keep Alive (KA) beacons are sent every 10s ensuring routing path accuracy and synchronization across the network. OpenMote operates using 32 kB RAM and 512 kB of flash memory. The battery capacity is 2821.5 milliampere-hour (*mAh*). The device is compatible with IP, and TSCH mode of IEEE 802.15.4, operating at 2.4GHz. The TSCH slotframe is configured with 101 time slots and 16 channels. The slot duration is set to 10ms. The sender prepares the payload after confirming the availability of the channel [30]. It then configures its radio as per the mentioned channel offset, then initiates the end-to-end transmission. After some time, the sender's radio is turned off and the same slot is used for reception by the receiver. After successfully receiving the payload, the receiver node sends an acknowledgement and turns off the radio. Typically, this process consumes less than 1% of the slot duration (10s), and is referred to as duty-cycle. The consumption depends on the duty-cycle variations, caused by delayed acknowledgements. Longer delays are harmful as they are leading to unsuccessful operation. Extended listening causes power wastage. Consequently, each device implements timers deciding the length of period the radio is on [30]. The MAC layer allows a maximum of 5 *retry* attempts before the packet is removed from the TX queue. The energy consumption model defined by *Vilajosana et al.,* [65] is used to calculating the amount of charge consumed per slotframe cycle. This current version of the 6TiSCH simulator has replaced its propagation model with 'K7' connectivity traces [66], derived from several testbed experiments. With the use of connectivity traces, it offers more realistic results over extended slotframe cycles and performs well in terms of simulation speed and scenario testing.

To compute an average battery life, we translated the value for energy consumption to battery life in years using the following equation:

$$\frac{AA_{mAh} \times 10^3}{avg_{current\mu A}} \quad \ldots\ldots (4)$$

AA_{mAh} represents a capacity of the battery and $avg_{current\mu A}$ represents average current consumed per node (as a sum of all states [65]). To convert the charge consumed by a node, the output from Equation (4) must be divided by 24 (number of hours per day) and then by 365 (days per year). During bootstrapping, each node is sending a minimum of 15 beacons to nearby devices and awaits responses. Based on the received responses, the node computes the link's PDR which must not be less than 0.5. The full joining process is explained in [30]. The EBs are generated at the probability of 0.33 [67] as the

network experiences no significant difference in RSSI and PDR despite 2/3rd reduction, is contributing to reduced energy consumption.

For routing, a non-storing mode of RPL (RFC 6550) is employed, which, by default, uses Objective Function Zero (OF0). DIO messages are dispatched every 10s, while the DAO messages, are sent every 60s. 6TiSCH uses a housekeeping mechanism that checks for collided cells every 30s, while collisions in advertisement/shared slots are handled with the default back-off mechanism provided by IEEE 802.15.4.

Each experiment simulates network behavior using an event-detection model. The results are averaged over a number of simulation runs. For each run, the duration of the experiment is measured through the number of slotframe cycles. Each slotframe cycle lasts approximately 1s. Refer to Table 3 for the configuration parameters.

TABLE III
CONFIGURATION PARAMETERS

Name	Parameters
Deployment Area	[2*2] km
Communication patter	MP2P
Node's buffer	[10] packets
VRB Table Length	[2] rows
Radio Frequency Band	[2.4]GHz
Maximum available channels	16
Max retransmission attempt	5
SSR's MAX_NUMCELLS	Not fixed
Housekeeping period	[30]s
RPL's Objective Function	OF0
DIO's interval	[10]s
DAO's interval	[60]s
Broadcast Beacon probability	0.33
Keep Alive (KA) messages	[10]s
Packet Delivery Ratio	0.5
Maximum number of packets sent	15
Battery capacity	[2821.5] mAH
Radio sensitivity	[-97]decibel-milliwatts
Number of nodes	[50,60,70,80,90,100]
Topology	Random or linear
MAX_NUMCELLS	[4,16,32,64,100]
Slotframe Length	[33, 101] slots
Duration per run	[66.7, 530]minutes
Number of runs	[20, 100]
Packet generation interval	[0.1, 0.5, 1.0, 2.0, 20]s

A. Analysis of Traffic Adaption Delay and Optimization with Higher Traffic Load

This experiment is focused on testing the impact of fixed traffic adaption delays, introduced by MSF. The results are compared against SSR, which does not use a fixed value for MAX_NUMCELLS. The simulation settings shown in Table 4 show any varied setting for the first experiment based on values in Table 3.

TABLE IV
6TISCH SIMULATOR CONFIGURATION PARAMETERS

Name	Parameters
Number of nodes	[**50**]
Topology	Random
MAXNUMCELLS	[**4,32,64,100**] cells
Slotframe Length	[**33**] slots
Duration per run	[**66.7**]minutes
Number of runs	[100]
Packet generation interval	[**1.0**]s

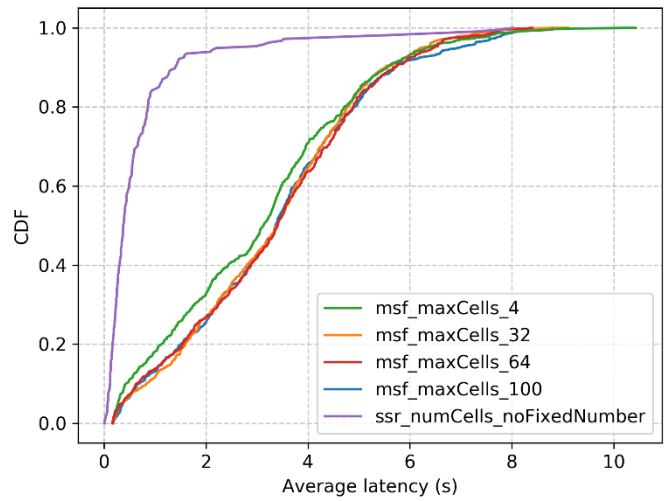


Fig. 9. Comparing average latency delay.

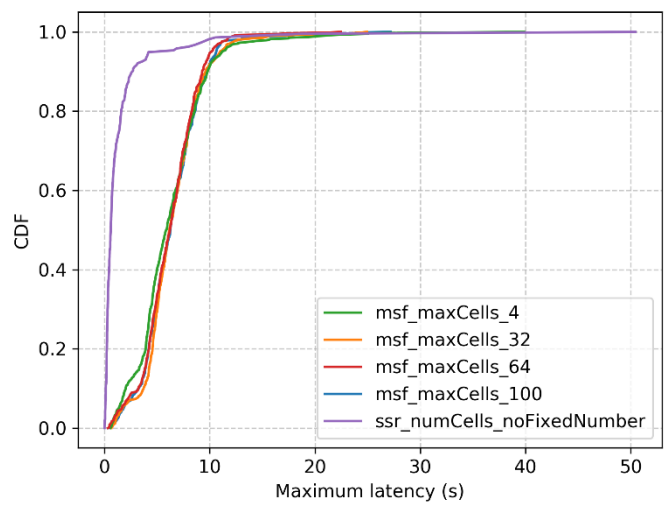


Fig. 10. Comparing maximum latency of MSF and SSR.

Typically, latency fluctuates depending on the size of the slotframe. The Cumulative distribution function (CDF) of MSF and SSR in Fig. 9 and 10 exhibit latency as a function of MAX_NUMCELLS. For SSR, Figure 9 is showing an average latency of about 1s at a 90% probability. Figure 10 indicates a maximum latency of just over 1s at 90% probability. In contrast, an average latency for MSF is between *3s* and *5s* at a probability of 90%. Its maximum latency is reaching close to *10s*. Clearly, SSR is outperforming MSF significantly.

Fig. 11 shows a average PDR of 99% for SSR whereas MSF achieves between 68% – 77% depending on the adaption delays. The primary causes include congestion, attributed to MSF's fixed allocation policy (allowing only 1 cell per node) against higher demand and the number of times the maximum retransmission limit is reached, preventing new packets entering the TX buffer. Congestion and retransmissions are shown in Fig. 12 and 13, respectively.

In a 6TiSCH network, if transmission delays are significant and all retransmission attempts are exhausted, the packet is dropped from the TX queue. Poor propagation conditions can also increase the retransmission rate due to limited route availability. However, this drawback does not affect MSF

alone, as MSF typically establishes longer multi-hop routes to the root compared to SSR (see Fig. 17). As a result, distant nodes differ in their ability to receive broadcasts over extended source routes. TSCH handles communication across these longer routes with a non-zero probability, but the use of higher TX power is restricted under the 'K7' connectivity model (as implemented in the simulator) to prevent favoring poorly performing nodes. Consequently, MSF experiences additional packet loss due to exhausted retries more frequently than SSR, as illustrated in Figure 13.

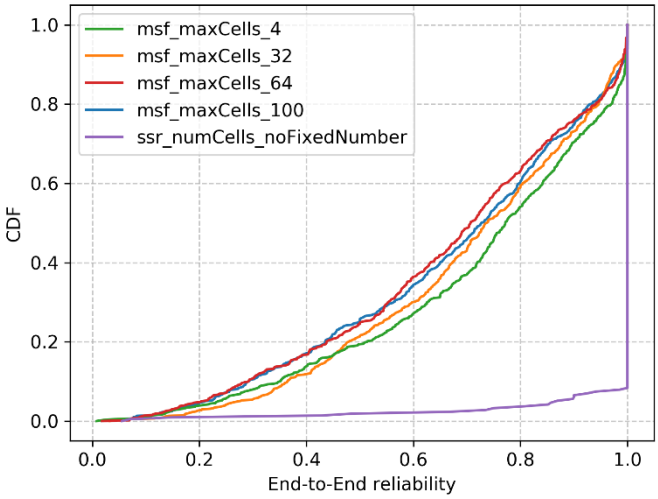


Fig. 11. Comparing end to end reliability.

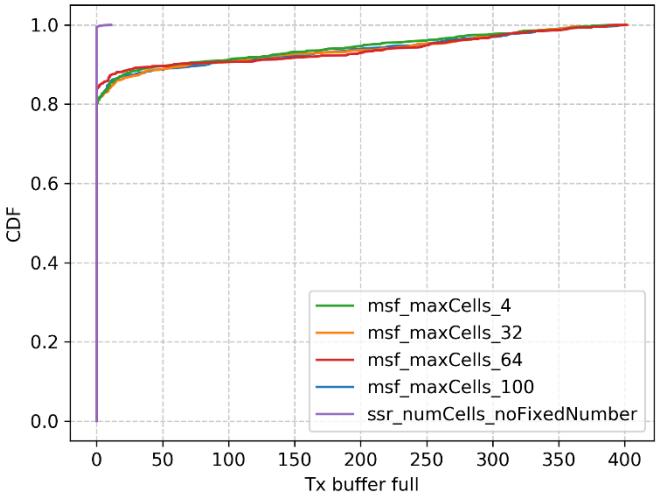


Fig. 12. Comparing congestion-induced loss.

In terms of battery life, frequent retransmissions impedes energy savings especially when using a smaller slotframe. Fig. 16 is showing the CDF of battery life, with MSF performing poorly due to exhausted *retries*, and a higher TX cell utilization, followed by the higher number of failed transactions (6P errors). Notably, SSR shows an improved battery life by delivering most packets within the agreed *retries* limit, and thus consuming a lower number of cells than MSF with fewer 6P failures. This is shown in Fig. 13, 14, and 15.

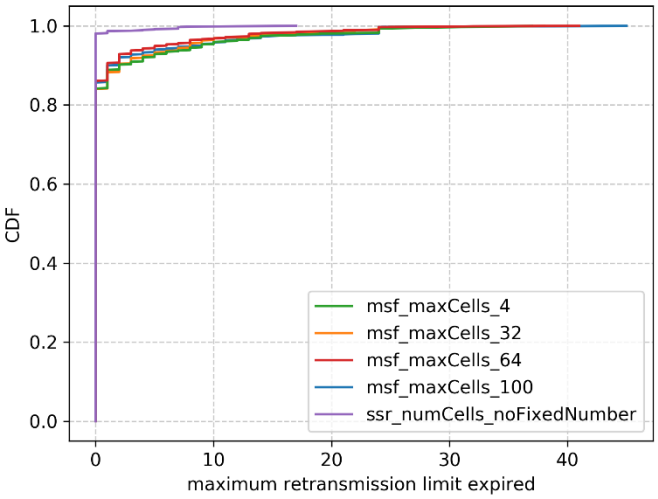


Fig. 13. Comparing loss due to exhausted retries.

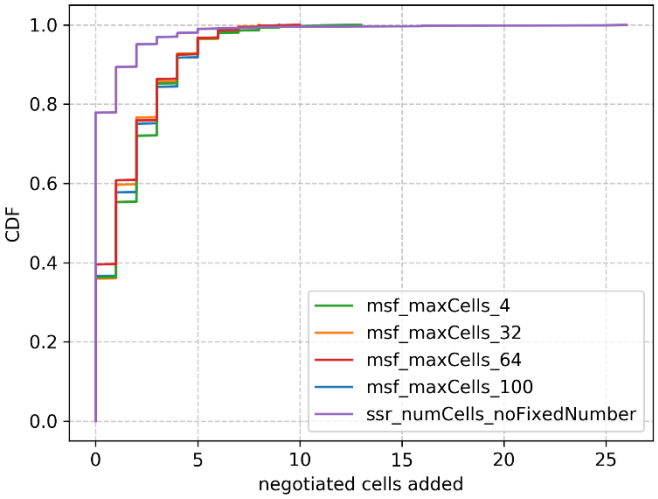


Fig. 14. Comparing Tx cell utilization by both SFs, 33 slots.

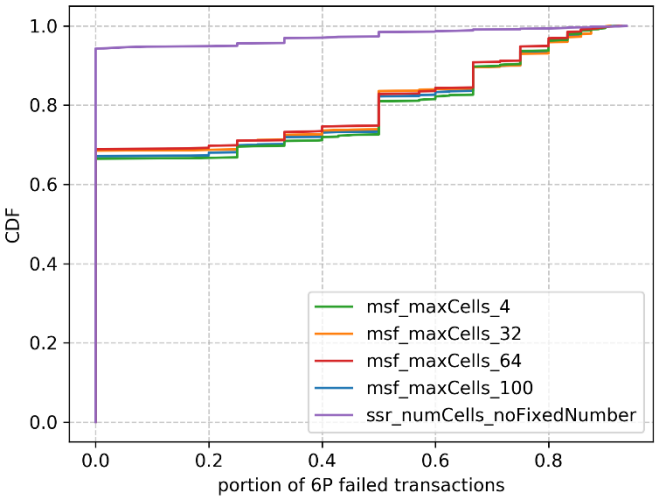


Fig. 15. Comparing average portions of failed 6P transactions.

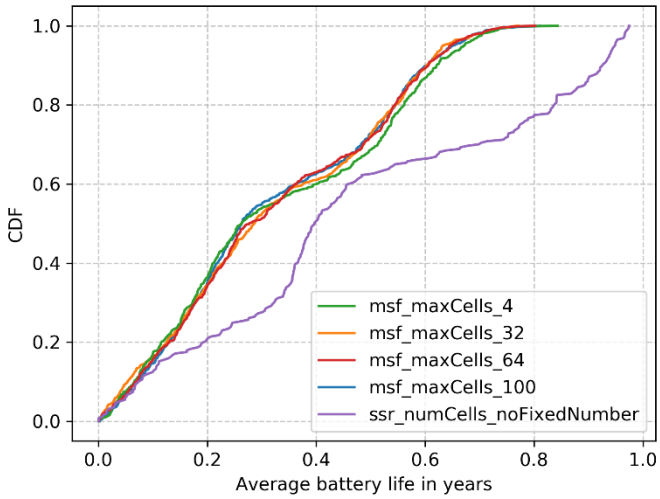


Fig. 16. Comparing average battery life per node, 33 slots.

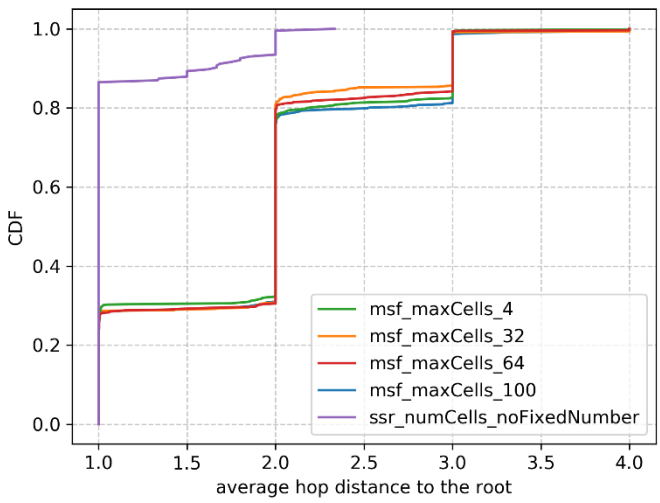


Fig. 17. Comparing average hop distance for MSF and SSR.

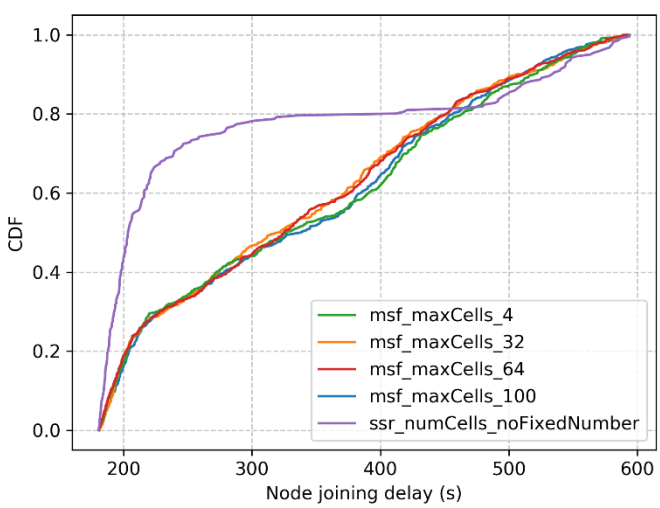


Fig. 18. Comparing node joining delay using 33 slots.

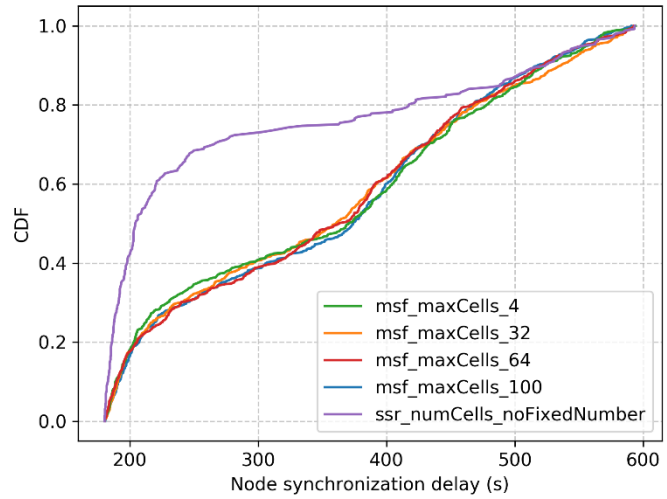


Fig. 19. Comparing synchronization delay using 33 slots.

Figure 17 provides insights into how the random topology translates into larger number of hops, due to poor traffic adaption. It demonstrates that SSR achieves a more compact and efficient routing topology.

In remotely operated WSN, nodes frequently connect and disconnect, and if connection to the root is lost, the entire network hierarchy is lost. Therefore, a lower joining delay is preferred to quickly resume data streaming. Fig. 18 and 19 present joining and synchronization delays caused by both SFs. SSR exhibits a shorter joining and synchronization delay for most nodes. The key cause is a more compact topology, allowing nodes to join the network at a shorter distance to root. An increased distance (Fig. 17) provokes propagation loss where nodes receive broadcast beacons at delay.

B. Scalability Analysis

This section assesses scalability using a varying density of nodes per installation as the packet generation is set to 0.5 (120 ppm). The load factor has been nearly doubled in comparison to our previous experiment (Section VI A). The CDF plots for both SFs are kept side by side to allow comparison. The configuration parameters are presented in Table 5.

TABLE V
6TISCH SIMULATOR CONFIGURATION PARAMETERS

Name	Parameters
Number of nodes	[**50,60,70,80,90,100**]
Topology	Random
MAXNUMCELLS	[**32**] cells
Slotframe Length	[**101**] slots
Duration per run	[66.7]minutes
Number of runs	[100]
Packet generation interval	[**0.5**]s
Housekeeping period	[30]s
Keep Alive (KA) messages	[10]s
Packet Delivery Ratio	0.5
Maximum number of packets sent	15
Deployment area	[2 * 2] km

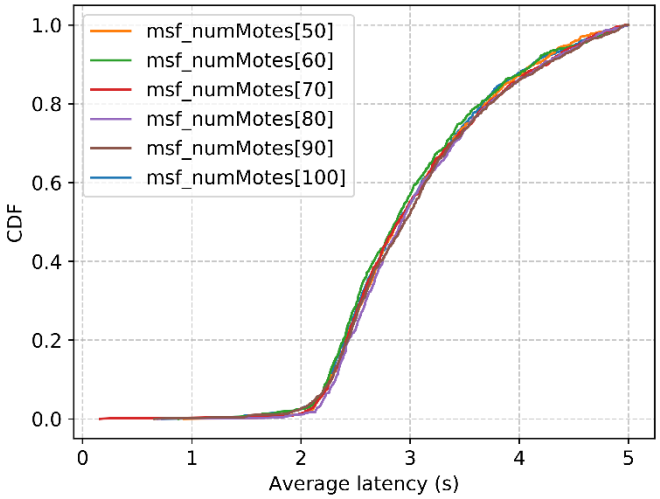


Fig. 20. Comparing MSF's latency using 101 slots, 120ppm.

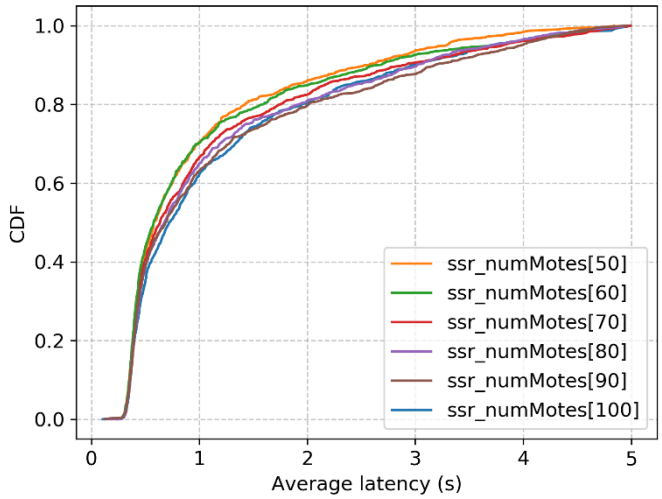


Fig. 21. Comparing SSR's latency using 101 slots, 120ppm.

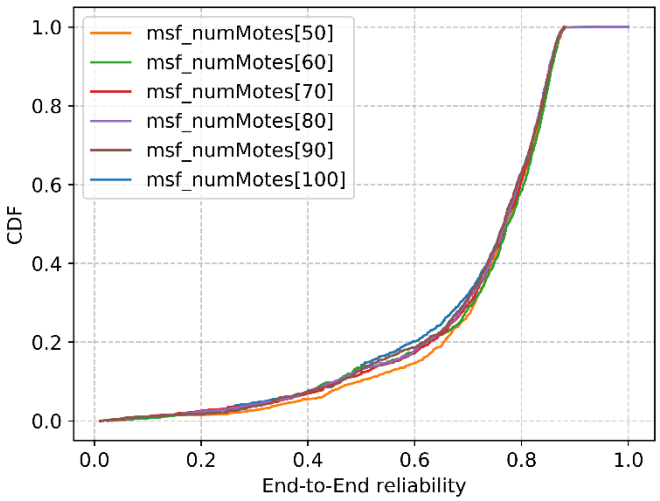


Fig. 22. Comparing MSF's reliability using 101 slots, 120ppm.

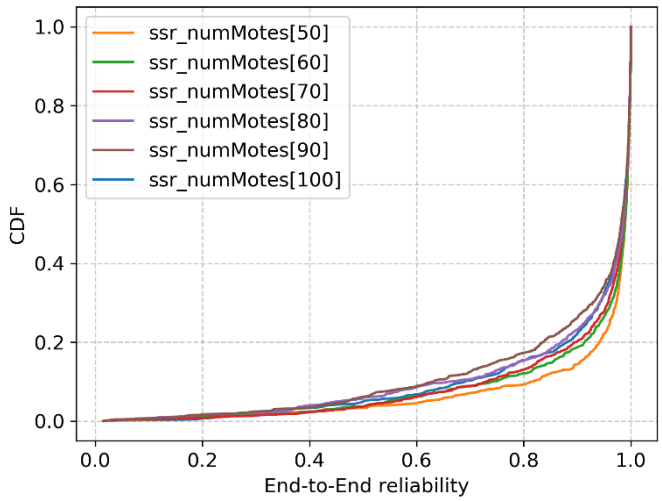


Fig. 23. Comparing SSR's reliability using 101 slots, 120ppm.

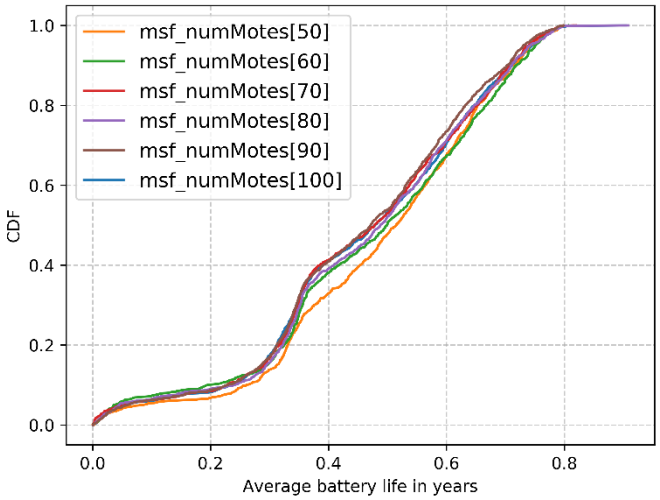


Fig. 24. Comparing MSF's battery life using 101 slots, 120ppm.

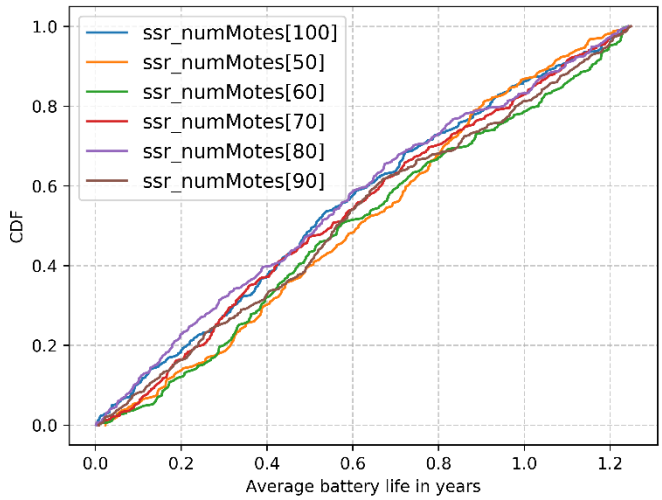


Fig. 25. Comparing SSR's battery life using 101 slots, 120ppm.

Figures 20 and 21 compare the average latency of MSF and SSR. Here, SSR consistently maintains a lower latency, close to *1s* for the majority of the nodes. In contrast, MSF shows a latency to about 3s. MSF suffers from high latency due to poor traffic adaption.

MSF is not coping well with the heavy periodic traffic demand. Some nodes experience temporary spikes in traffic, and MSF responds by evenly distributing cells based on TX utilization and a fixed adaptation delay. This delay is set to 32 cells, meaning a node must wait until 32 6Top cell requests have been attempted, whether successful or not, before it can adjust to changing traffic conditions (NUMCELLELAPSED = 32). The effect of this adaptation delay is discussed further in Section V(A).

SSR consistently adapts to real-time traffic demands across each hop in the topology, even under medium contention. Its ability to promptly allocate cells through source routing and proactive scheduling helps mitigate latency and sustain packet delivery performance. In contrast, MSF exhibits higher payload loss, approximately 20% more, across all network densities, as shown in Figure 22. This highlights the superior reliability of SSR, further supported by the results presented in Figure 23.

SSR frequently adds comparably more additional cells to fewer nodes (only those close to the root) and comparably fewer cells are given to a larger number of nodes located further away from the root. This has led to a 99% reliability for identical settings. When a node experiences a temporary traffic peak, more cells are added to handle the load. However, nodes often change parents. When this happens, the TSCH scheduler moves cells to better-suited nodes. Near the end of each slotframe cycle, any extra cells are removed. These cells are then reallocated at the start of the next cycle, depending on how long the experiment runs. Figures 24 and 25 show SSR nodes last longer than MSF nodes, with 80% of SSR nodes reaching about 1 year of battery life, compared to just over 0.6 years for 80% of MSF nodes. The results suggest that SSR significantly outperforms MSF for the given challenging traffic conditions across presented scenarios.

C. Analysis using Linear Topology

This section focuses on the controlled deployment of a large-scale network using a linear topology. Linear deployment is popular for reduced interferences from nearby nodes. The aim is to observe the impact of various periodic traffic patterns by evaluating both SFs in terms of stable, sustainable, and scalable performance. The updated configuration settings for this experiment (based on Table 3) are shown in Table 6.

TABLE VI
6TISCH SIMULATOR CONFIGURATION PARAMETERS

Name	Parameters
Number of nodes	[**100**]
Topology	**Linear**
MAXNUMCELLS	[32] cells
Slotframe Length	[101] slots
Duration per run	[66.7]minutes
Number of runs	[100]
Packet generation interval	[**0.1,0.5,1.0,2.0**]s

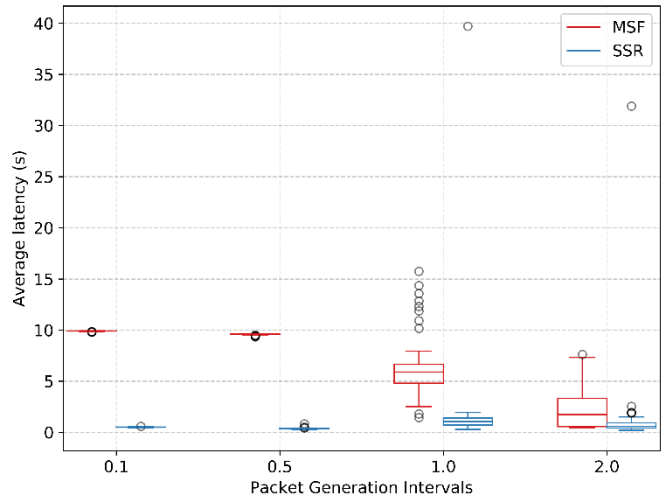


Fig. 26. Average latency as a function packet generation interval.

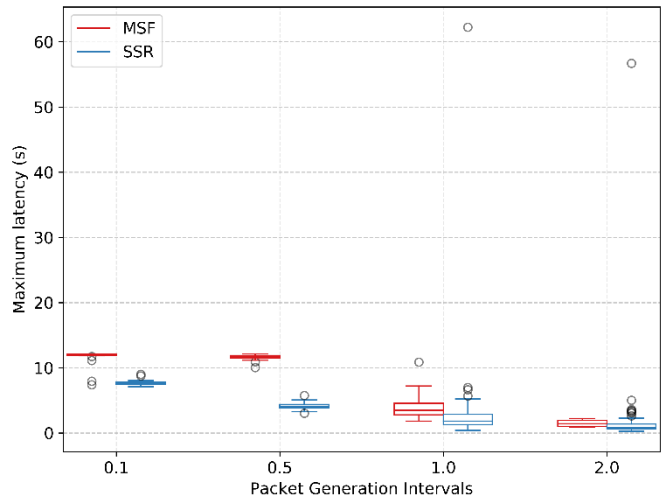


Fig. 27. Maximum latency as a function packet generation interval.

Figures 26 and 27 compare the average and maximum latencies of MSF and SSR across packet generation intervals of 0.1, 0.5, 1.0, and 2.0s. MSF consistently exhibits higher latency in both metrics. In contrast, SSR maintains a relatively constant average latency, while its maximum latency decreases as the packet generation interval increases. The primary causes include poor traffic adaption and inefficient queue handling. This has further impact on packet delivery in a hop by hop manner as some nodes are more occupied than other and MSF simply fails to address this, instead it allows nodes to add more negotiated cells unanimously.

Figure 28 illustrates reliability for SSR and MSF. The boxplot indicates that SSR is experiencing hardly any packet loss. Only under extremely challenging conditions, at a packet generation interval of 0.1s (600 ppm) load, its performance drops slightly to about 96%, remaining stable at 99% for all other configurations. MSF demonstrates equally high reliability for packet generation intervals of 1.0s and 2.0s. However, for higher loads MSF's reliability declines more noticeably to just over 80% for the most challenging setting.

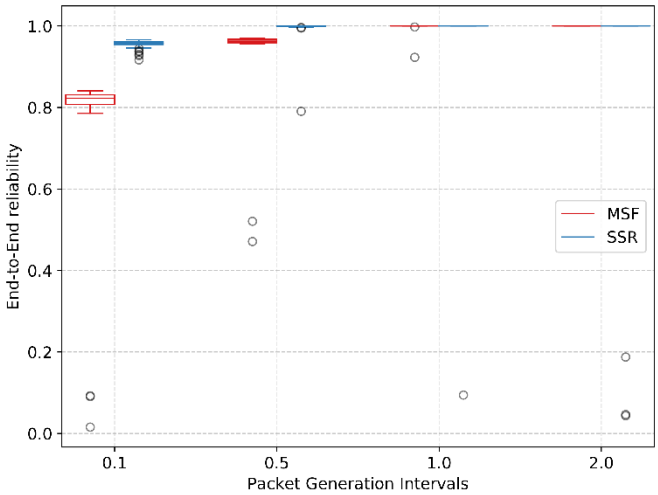


Fig. 28. Upstream reliability as a function packet generation interval.

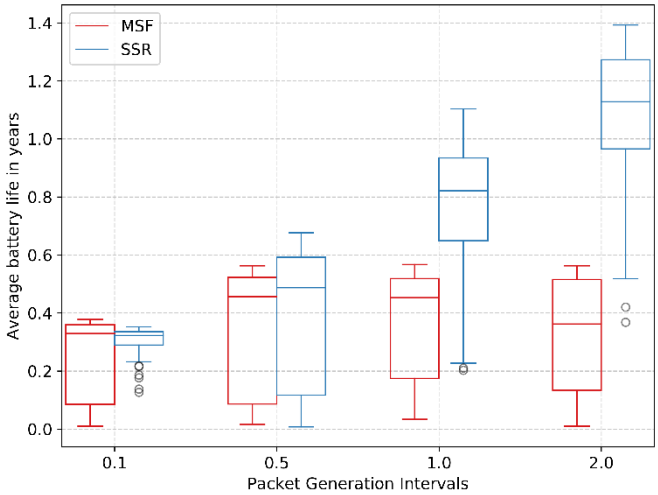


Fig. 29. Battery life as a function packet generation intervals.

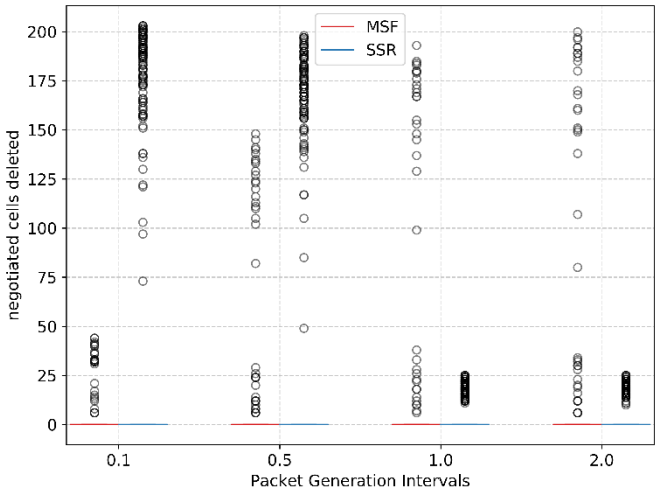


Fig. 30. 6Top DELETE as a function packet generation interval.

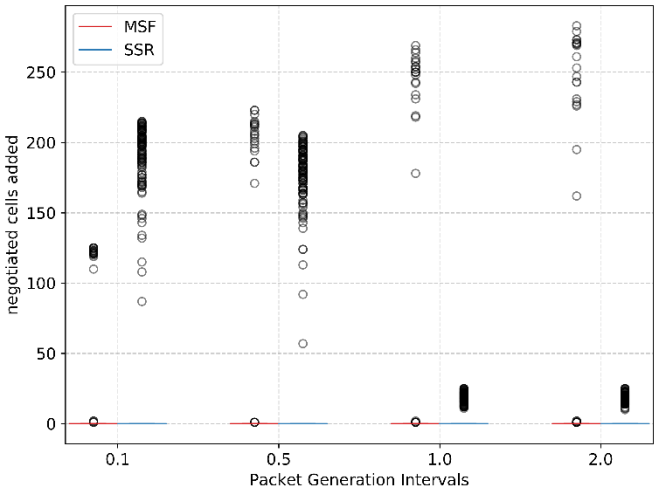


Fig. 31. 6P ADD transactions as a function packet generation interval.

As far as battery life is concerned, Figure 29 demonstrates that, with a linear topology, SSR offers a linear proportional increase in network age as traffic load decreases. With a packet period of 2.0s, the battery life of SSR could exceed a year. Under challenging traffic conditions of 60 ppm, the battery lifespan decreases to approximately just over 0.8 years. The lifespan declines to 0.5 years and bottoms out at 0.3 years at 120 ppm and 600 ppm, respectively. In contrast, MSF exhibits a poorer battery life compared to SSR, showing no lifespan improvement with increasing packet generation intervals. Key reasons are higher overheads and fixed traffic adaption rules. This is evident from Figures 30 and 31, which illustrate the average volume of cells deleted and added from the negotiated slotframe where cells are not shared. The boxplot reveals a significant number of outliers, attributed to the non-uniform addition and deletion of cells by MSF and SSR due to varying traffic demands across the network. The data shows that the number of 6P cycles for deleting cells exceeds those for adding cells in both SFs. Notably, SSR triggers a minimum number of outliers at packet period interval of 1.0 (60ppm) and 2.0 (30 ppm), which positively impacts battery life efficiency. As traffic load increases to 120ppm (packet period= 0.5) and 600 ppm (packer period=0.1), SSR aggressively adds and deletes cells to maintain higher reliability, however, interestingly, not for all nodes (as they are mostly shown as outliers). Crucially, for the lower packet densities, SSR adds and deletes significantly fewer cells than with the higher packet densities. In the case of MSF, the consumption of cells follows a lower bound (adding only 1 cell at a time) under the specified loads.

Conclusion

This article presents a novel scheduling function, SSR, implementing TSCH mode of the IEEE 802.15.4 technology. It uses the cake-slicing technique where the heuristics produced are dynamically adopted across key decisions made under scheduling, as detailed in Section IV.

Fixed-Adaption Delay and Queue Utilization: Section VI (A) highlighted that fixed-adaption delay does not alleviate high consumption and queue utilization issues. MSF drops a significant proportion of the payload at a traffic load of 60

ppm. In contrast, SSR rejects the concept of fixed delay and threshold for traffic adaptation, offering 99% reliability using only about 1/3rd of the total available bandwidth.

Reliability and Battery Life: The results in Section VI (B) showed that SSR overcomes limitations encountered by MSF in terms of reliability, latency and battery life in large-scale implementations with a single access point.

Large-Scale Network Performance: The experiment in Section IV(C) tested the performance of a large-scale network using a linear topology. SSR exhibited a linear impact on the consumption of link-layer resources. As traffic load increased to their highest levels (600ppm), it registered only a small drop in reliability, compared to a 15% drop by MSF. Additionally, SSR had shown an improved latency and longer battery life at 30ppm and 60ppm. The linear increment in network longevity make SSR an ideal solution for remotely deployed networks encountering challenging traffic conditions.

The comprehensive evaluation concludes that SSR is an ideal scheduling function for various industrial projects involving periodic data streaming, regardless of linear or random deployment. SSR's hybrid scheduling approach significantly enhances network performance, and efficiency, providing scalability and making it a suitable solution for high-traffic, and adaptive data-rate based applications.

References

[1] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," CISCO, 2020.

[2] Ericsson, "IoT connections outlook," ericsson.com, 2023.

[3] T. Watteyne, S. Lanzisera, A. Mehta and K. J. Pister, "Mitigating Multipath Fading through Channel Hopping in Wireless Sensor Networks," in *2010 IEEE International Conference on Communications*, Cape Town, SA, 2010.

[4] K. Pister, T. Phinney, p. Thubert and S. Dwars, "Industrial Routing Requirements in Low-Power and Lossy Networks," *IETF:RFC 5673,* pp. 1-25, 2009.

[5] Connectivity standards alliance, "zigbee The Full Stack Solution for All Smart Devices," 2024. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/. [Accessed 6 2024].

[6] wikipedia, "Bluetooth Low Energy," 1 2 2021. [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth_Low_Energy.

[7] The International Society of Automation, "ISA 100.11A Wireless Systems for Industrial Automation: Process Control and Related Applications," ISA, 4 May 2011. [Online]. Available: https://standards.globalspec.com/std/1413497/ISA%20100.11A. [Accessed 10 07 2019].

[8] A. N. Kim, F. Hekland, S. Petersen and P. Doyle, "When HART goes wireless: Understanding and implementing the WirelessHART standard," in *in 2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Hamburgh,germany , 2008.

[9] N. Kushalnagar, G. Montenegro and C. Schumacher, "IPv6 over LowPower Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, Goals," RFC 4919, 2007.

[10] T. Watteyne, M. R. Palattella and L. A. Grieco, "Using IEEE802.15.4e Time-Slotted Channel Hopping((TSCH) in the Internet of Things (IoT): ProblemStatement," *Internet Engineering Task Force, IETF,* 2015.

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

[12] wikipedia, "Time-division multiple access," 6 2024. [Online]. Available: https://en.wikipedia.org/wiki/Time-division_multiple_access.

[13] Q. Wang, X. Vilajosana and T. Watteyne, "6TiSCH Operation Sublayer (6top)," 4 06 2014.

[14] P. Thubert, "draft-thubert-6tisch-architecture," *IETF Draft,* 2019.

[15] A. R. Urke, Ø. Kure and K. Øvsthus, "A Survey of 802.15.4 TSCH Schedulers for a Standardized Industrial Internet of Things," *Sensors,* vol. 22, no. 1, 2022.

[16] R. M. Senouci, B. Djamaa and A. Tabouche, "Traffic-Aware Reliable Scheduling in TSCH Networks for Industry 4.0: A Systematic Mapping Review," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 25, no. 4, pp. 2834-2859, 2023.

[17] T. Chang, X. Vilajosana, M. Vucinic and S. Duaquennoy, "6TiSCH Minimal Scheduling Function (MSF)," *RFC 9033, IETF,* 2021.

[18] E. Municio, G. Daneels, M. Vucinic, S. Latre, J. Famaey, Y. Tanaka, K. Brun, k. Muraoka, X. Vilajosana and T. Watteyne, "Simulating 6TiSCH network," *telecommunication journal,* pp. 1-17, 2018.

[19] C. C. A. S. Bormann, "CoAP: An application protocol for billions of tiny Internet nodes," *IEEE Internet Computing,* vol. 1, no. 2, pp. 62-67, 2012.

[20] HTTP, "HTTP," Wikipedia, 11 11 2024. [Online]. Available: https://en.wikipedia.org/wiki/HTTP#cite_note-4. [Accessed 14 11 2024].

[21] ericsson, "OSCORE: A look at the new IoT security protocol," 7 11 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/11/oscore-iot-security-protocol.

[22] P. Thubert, E. Nordmark, S. Chakrabarti and C. E. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Networks (6LoPWAN) Neighbor Discovery," *RFC 8505,* 11 2018.

[23] P. Thubert, "ROLL: Internet Draft," IETF ROLL, 11 06 2012. [Online]. Available: https://tools.ietf.org/html/draft-thubert-roll-asymlink-02. [Accessed 1 10 2019].

[24] B. Pavkovic, F. Theoleyre and A. Duda, "Multipath opportunistic RPL routing over IEEE 802.15.4," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems MSWiM '11*, NY, USA, 2011.

[25] O. Lova, F. Theoleyre, T. Watteyne and T. Neol, "The love-hate relationship between IEEE 802.15.4 and RPL," *IEEE Communications Magazine ,* vol. 55, no. 1, pp. 188-194, 2017.

[26] P. Levis, T. Clausen, j. Hui, O. Gnawali and j. Ko, "The trickle algorithm: IETF RFC 6206," *IETF draft,* pp. 1-11, 2011.

[27] o. Iova, g. p. Picco, t. istomin and c. kiraly, "RPL: the routing standard for the internet of things... Or Is IT?," *IEEE communication Magzine,* vol. 54, no. 12, pp. 16-22, 2016.

[28] Z. Shelby and C. Bormann, 6LoWPAN: The Wireless Embedded Internet, Wily publishing, 2010.

[29] C. Bormann, T. Watteyne and P. Thubert, "RFC 8930 On Forwarding 6LoWPAN Frangments over a Multi-hop IPv6 Network," *IETF,* 2020.

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

[31] Q. Wang, X. Vilajosana and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)," *IETF RFC 8480,* 2018.

[32] M. R. Palattela, N. Accettura, M. Dohler, L. A. Grieco and G. Boggia, "Traffic aware scheduling algorithm for reliable low power multii-hop IEEE 802.15.4e Networks," in *IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, Sydney, NSW, Australia, 2012.

[33] Á. A. Farías and D. Dujovne, "A queue-based scheduling algorithm for PCE-enabled Industrial Internet of Things networks," in *2015 Sixth Argentine Conference on Embedded Systems (CASE)*, 2015.

[34] F. Righetti, C. Vallati and G. Anastas, "Performance Evaluation of Adaptive Autonomous Scheduling Functions for 6TiSCH Networks," *IEEE Access,* vol. 9, pp. 127576-127594, 2021.

[35] D. Dujovne, L. A. Grieco, M. R. Palattela and N. Accettura, "6TiSCH 6Top schedulling function Zero," *IETF-6TiSCH Draft,* pp. 1-9, 2016.

[36] T. Watteyne, m. r. palattela, Q. Wang, K. Muraoka, n. Accettura, D. Dujovne, a. l. Grieco and t. Engel, "On-the-fly bandwidth reservation for 6TiSCH wireless industrial network," *IEEE sensor journal,* pp. 1-11, 2016.

[37] T. Chang, T. Watteyne, Q. Wang and X. Vilajosana, " LLSF: low latency scheduling function for 6TiSCH networks," in *Proceedings of International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2016.

[38] R. T. Hermeto, G. Antoine and T. Fabrice, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," *Computer Communication,* vol. 114, pp. 84-105, 2017.

[39] G. Daneel, B. Spinnewyn, S. Latre and J. famaey, "ReSF: recurrent Low-Latency scheduling in IEEE 802.15.4e TSCH networks," *Ad-Hoc networks,* vol. 69, pp. 80-88, 2018.

[40] P. T. Duy, T. Dinh and Y. Kim, "Distributed cell selection for schedulling function in 6TiSCH networks," *Compter Standard & Interfaces,* vol. 53, pp. 80-88, 2017.

[41] D. M. Prieto, T. Chang, X. Vilajosana and T. Watteye, "Distributed PID-based Scheduling for 6TiSCH Networks," *IEEE Communications Letters,* vol. 20, no. 5, pp. 1006-1010, 2016.

[42] F. Rincon, T. Watteyene and Y. Tanaka, “On the Impact of WiFi on 2.4 GHz Industrial IoT Networks,” in *2018 IEEE International Conference on Industrial Internet (ICII)*, 2018.

[43] H. Wang, W. Yang, Y. Cao and Z. Zhang , “EDSF: Efficient Distributed Scheduling Function for IETF 6TiSCH-based Industrial Wireless Networks,” *Springer LInk,* vol. 27, pp. 2244-2253, 2022.

[44] M. Vucinic, M. P. Djurisic and T. Watteyne, “SODA: 6TiSCH Open Data Action,” in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, Porto, Portugal, 2018.

[45] E. Municio and S. Latre, “Decentralized broadcast-based scheduling for dense multihop TSCH networks,,” in *MobiArch '16 Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, NY, USA, 2016.

[46] N. Accettura, M. R. palettela, G. Boggia, M. Dohler and L. A. Grieco, “Decentralized traffic aware scheduling for multi-hop low power lossy networks in the Internet of Things,” in *WoWMoM*, 2013.

[47] A. Aijaz and U. Raza, “DeAMON: a decentralized adaptive multi-hop scheduling protocol for 6Tisch wireless networks,” *IEEE Sensors Journal ,* vol. 17, no. 20, pp. 1-10, 2017.

[48] K. Kralevska, D. Vergados, Y. Jiang and M. Angelos, “A Load balancing algorithm for resource allocation in IEEE 802.15.4e Networks,” in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Athens, Greece, 2018.

[49] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia and M. Dohler, “Decentralised Traffic Aware Schedulling in 6TiSCH network: design and experimental evaluation,” *IEEE internet of things journal,* vol. 2, no. 6, pp. 455-470, 2015.

[50] D. J. Vergados a, K. Kralevska, Y. Jiang and A. Michalas, “Local voting: A new distributed bandwidth reservation algorithm for 6TiSCH networks,” *Computer Networks,* vol. 180, 2020.

[51] F. Righetti, C. Vallati, G. Anastasia and S. K. Das, “Analysis and improvement of the on-the-fly bandwidth reservation algorithm for 6tisch,” in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018.

[52] H. Wang and A. O. Fapojuwo, “Design and Performance Evaluation of a Hysteresis-Free On-the-Fly Scheduling Function for 6TiSCH,” *IEEE Internet of Things Journal,* vol. 8, no. 13, pp. 10499 - 10508, 2021.

[53] K. Kumar and M. Kolberg, “DeSSR: A Decentralized, Broadcast-Based Scalable Scheduling Reservation Protocol for 6TiSCH Networks,” *IEEE Internet of Things Journal,* vol. 11, no. 7, pp. 12728-12744, 2023.

[54] S. Duaquennoy, B. A. nahas, T. Watteyne and O. Landsiedel, “Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH,” in *ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Seoul, South Korea, 2015.

[55] A. Paventhan, D. Dashini, h. krishna, N. Pahuja , M. Khan and A. Jain, “Experimental evaluation of IETF 6TiSCH in the context of Smart Grid,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT),*, 2015.

[56] S. Kim, H. -S. Kim and C. Kim, “Autonomous Link-based TSCH Cell Scheduling,” in *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Montreal, cananda, 2019.

[57] Y. H. Pratama and S. Chung, “RL-SF: Reinforcement Learning based Scheduling Function for Distributed TSCH Networks,” in *2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Beijing, China, 2022.

[58] S. Jeong, J. Paek, H. -S. Kim and S. Bahk, “TESLA: Traffic-aware elastic slotframe adjustment in TSCH networks,,” *IEEE Access,* vol. 7, pp. 130468-130483, 2019.

[59] T. Chang, m. Vucinic, X. Vilajosana, D. Dujovne and T. Watteyne, “6TiSCH minimal scheduling function: Performance evaluation,” *Internet Technology letters,* vol. 3, no. 4, 2020.

[60] Y. Tanaka, P. Minet, T. Watteyne and F. Teraoka, “Accelerating 6TiSCH Network Formation,” in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2021.

[61] I. . A. Valdovinos, P. E. F. Millan, J. A. Perez-Diaz and C. Vargas-Rosales, “Distributed Channel Ranking Scheduling Function for Dense Industrial 6TiSCH Networks,” *Sensors,* vol. 21, no. 5, p. 1593, 2021.

[62] T. Watteyene, J. Doherty, J. Simon and K. Pister, “Technical overview of SmartMesh IP,” in *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013.

[63] T. Chang, P. Tuset-Peiro and X. Vilajosana, “OpenWSN & OpenMote: Demo'ing a Complete Ecosystem for the Industrial Internet of Things,” in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, London, UK, 2016.

[64] X. Vilajosan, P. Tuset, T. Watteyne and K. Pister, “OpenMote: OpenSource Prototyping Platform for the Industrial IoT),” in *7th International Conference AdHocHets 2015*, 2015.

[65] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang and K. J. Pister, “A realistic Energy Consumption Model for TSCH Networks,” *IEEE Sensors,* vol. 14, no. 2, pp. 482-489, 2014.

[66] Y. Tanaka, K. Brun-Laguna and T. Watteyne, “Demo: Simulating a 6TiSCH Network using Connectivity Traces from Testbeds,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, france, 2019.

[67] M. Vučinić, T. Watteyne and X. Vilajosana, “Broadcasting strategies in 6TiSCH networks,” *Internet Technology letter,* 2017.

[68] B. Buurman, J. KAMRUZZAMAN, G. Karmakar and S. Islam, “Low-Power Wide-Area Networks: Design Goals,Architecture, Suitability to Use Cases and Research Challenges,” *IEEE Access,* pp. 17179- 17220, 2020.

[69] U. Raza, P. Kulkarni and M. Sooriyabandara, “Low Power Wide Area Networks: An Overview,” in *in IEEE Communications Surveys & Tutorials*, 2017.

[70] N. Kushalnagar, G. Montenegro and C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs):Overview, Assumptions, Problem Statement, and Goals,” *RFC 4919, IETF,* 2007.

[71] The International Society of Automation, “ISA 100.11A Wireless Systems for Industrial Automation: Process Control and Related Applications,” in *ISA*, 2011.

[72] D. Dujovne, T. Watteyene, X. Vilajosana and P. Thubert, “6TiSCH: deterministic IP-ena-bled industrial internet (of things),” *IEEE Comm. Mag,* vol. 52, no. 12, pp. 36-41, 2014.

[73] T. Watteyne, M. R. Palattella and L. A. Grieco, “Using IEEE802.15.4e Time-Slotted Channel Hopping((TSCH) in the Internet of Things (IoT): ProblemStatement,” *Internet Engineering Task Force, IETF,* 2015.

[74] P. Thubert, “draft-thubert-6tisch-architecture,” *IETF Draft,* 2019.

[75] Q. Wang, X. Vilajosana and T. Watteyne, “6TiSCH Operation Sublayer (6top),” 4 06 2014.

[76] P. Thubert, X. Vilajosan, C. Schmitt and J. Farkas, “Reliable and Available Wireless Technologies,” *IETF WG,* 2023.

[77] E. Municio, G. Daneels, M. Vucinic, S. Latre, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana and T. Watteyne, “Simulating 6TiSCH network,” *Telecommuni-cation Journa,* pp. 1-17, 2018.

[78] P. Thubert, “ROLL: Internet draft,” *Internet Eng. Task Force,* p. 2020, 9 1 2019.

[79] P. Thubert, E. Norkmark, S. Chakrabarti and C. Perkins, “Registration Extensions for IPv6 over Low-Power Wireless Personal,” *RFC 8508, IETF,* 2018.

[80] C. Bormann, A. Castellani and Z. Shelby, “CoAP: An application protocol for billions of tiny Internet nodes,” *IEEE Internet Computing,* vol. 1, no. 2, pp. 62-67, 2012.

[81] X. Vilajosana, T. Watteyne, T. Chang, m. Vucinic, S. Duquennoy and P. Thubert, “IETF 6TiSCH: A Tutorial,” *Communications Surveys and Tutorials, IEEE Communications Society,* 2019.

[82] G. Selander, J. Mattsson, F. Palombini and L. Sei, “Object Security for Constrained RESTful Environments (OSCORE),” *IETF RFC 8613,* 2019.

[83] B. Ghaleb, A. Al-Dubai, E. Ekonomou, A. Alsarham, Y. Nasser, L. Mackenzie and A. Boukerche, “A survey of limitation and enhancement of the IPV6 routing protocol for low-power and lossy networks: A focus on core operations,” *IEEE Communication Survey and Tutorials (Early Access),* pp. 1-31, 2018.

[84] M. Almistarihi, M. Al-shuruman and A. Qusaimat, “Tree based dynamic address autoconfiguration in mobile ad hoc network,” *computer networks,* vol. 55, no. 8, pp. 1894-1908, 2011.

[85] S. E. Deering and A. Conta, “Generic Packet Tunneling in IPv6 Specification,” *IETF RFC 2473,* 12 1998.

[86] B. Ghaleb, A. Al-Dubai, E. Ekonomou, A. Alsarham, Y. Nasser, L. Mackenzie and A. Boukerche, “A survery of limitation and enhancement of the IPV6 routing protocol for low-power and lossy networks: A focus on core operations,” *IEEE communication survey and tutorials,* vol. 21, no. 2, pp. 1607 - 1635, 2018.

[87] É. Vyncke , “IPv6 over Networks of Resource-constrained Nodes WG (6lo),” IETF, 18 3 2024. [Online]. Available: https://datatracker.ietf.org/doc/charter-ietf-6lo/. [Accessed 10 8 2024].

[88] P. Thubert, “A Routing Protocol for Low-Power and Lossy Networks (RPL) Destination-Oriented Directed Acyclic Graph (DODAG) Configuration Option for the 6LoWPAN Routing Header,” IETF, 4 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9035. [Accessed 10 8 2024].

[89] A. Morell, X. Vilajosana, T. Watteyne and J. L. Vicario, “Label-Switching over IEEE 802.15.4e Networks,” *Telecommunication,* vol. 24, no. 5, pp. 458-476, 2013.

[90] D. Papadimitriou, “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control,” IETF: RFC 4328, 1 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4328.html#:~:text=RFC%204328%3A%20Generalized%20Multi%2DProtocol,709%20Optical%20Transport%20Networks%20Control. [Accessed 12 1 2024].

Kaushal Kumar was born in Nagla Bihari Village, Mathura, UP, India in 1983. He received the MSc in computer science from the University of Stirling, UK, in 2008, and the MBA from Edinburgh Napier University, UK, in 2013. He is a tutor in the computer science department, and a student who is currently pursuing the Ph.D. degree in computer science at the University of Stirling, Stirling, UK. Kaushal has published 1 paper in IEEE IoT journal

Mario Kolberg is a professor of computing science and Associate Dean at the University of Stirling. His research interests include Peer-to-Peer overlay networks, Wireless Sensor Networks, and Internet of Things. Mario is editor of the bi-annually published Consumer Communication Networking Series within the IEEE Communications Magazine. Mario is also on the editorial Board of the Springer Journal 'Peer-to-Peer Networking and Applications' and has a long-standing involvement with the IEEE CCNC conference series. Dr. Kolberg has published more than 60 papers in leading journals and conferences. He is a member of a number of international conferences and program committees on networking and communications. He is a Senior Member of the IEEE and holds a PhD from the University of Strathclyde, UK.