Autonomous Collision-free Scheduling in Low-Power Wireless Sensor Networks

Andras Pinter
Department of Computer Science
University of York
York, United Kingdom
alp567@york.ac.uk

Leandro Soares Indrusiak
School of Computing
University of Leeds
Leeds, United Kingdom
L.SoaresIndrusiak@leeds.ac.uk

Ian Gray
Department of Computer Science
University of York
York, United Kingdom
ian.gray@york.ac.uk

***Abstract*—Wireless Sensor Networks (WSNs) provide a cost-effective, scalable, and adaptable alternative to wired deployments, making them ideal for smart infrastructure applications. However, challenges related to Quality of Service (QoS) and reliability hinder their adoption in safety-critical use cases or industrial automation. Medium Access Control (MAC) optimisation plays a crucial role in addressing these limitations. This paper introduces the Autonomous Collision-free Protocol (ACP), a MAC scheduling mechanism designed to reduce contention and support high network throughput while ensuring energy efficiency. ACP achieves this through linear probing-based collision resolution, hashing-based autonomous scheduling enabling multi-slot allocation, and a MAC schedule cell deactivation mechanism. Simulation results indicate that ACP successfully addresses network contention, eliminating packet collisions due to schedule conflicts, and significantly improves throughput compared to existing protocols.**

Keywords—wireless sensor networks, autonomous medium access control scheduling, collision-avoidance

I. INTRODUCTION

Wireless sensor networks (WSNs) offer potential advantages over wired connections, including adaptability, cost-effective deployment, maintenance and scalability. These benefits have incentivised research on their application in smart infrastructures, such as environmental and agricultural monitoring, smart cities, traffic control and building automation [1], [2], [3].

However, challenges related to Quality of Service (QoS), reliability, performance and the provision of hard real-time guarantees continue to hinder large-scale adoption in safety-critical and industrial applications. Communication optimisation between network nodes through efficient medium access control (MAC) addresses these issues by minimising contention and ensuring improved resource allocation, latency and data throughput. The dynamic nature of practical use cases often requires adaptive operation to accommodate changing application needs and network topologies, making centralised, offline optimisation of communication schedules impractical. While distributed scheduling, requiring negotiation between neighbouring nodes to finalise their schedules, offers increased flexibility, the associated communication and energy overhead can become prohibitive in environments with frequently changing conditions, dense deployments, or high network traffic.

Reliability and performance issues arise from unpredictable environmental conditions, internal interference, and congestion due to limited bandwidth. Time-Slotted Channel Hopping (TSCH) [4], which combines Time- and Frequency Division Multiple Access, has proven effective in addressing these challenges. The combination of TSCH with autonomous scheduling, based solely on information locally available to each node, enables fast and efficient network communication. Building on Duquennoy et al.'s [5] influential proposal, which outlines a method for the assignment of transmission slots using node ID hashing, subsequent research has explored further potential within the autonomous scheduling paradigm. A key challenge is that hashing-based scheduling mechanisms can result in contention for the wireless medium and packet collisions due to hashing conflicts. Despite improvements and refinements to this approach, so far no technique has comprehensively eliminated internal interference caused by colliding transmission slots.

To address this, we propose an autonomous, collision-free MAC scheduling mechanism. This approach not only resolves slot contention but is also extended to improve resource allocation under high traffic demands and includes a mechanism for the deallocation of resources to maintain energy-efficient operation.

This paper is organised as follows: Section II provides an overview of autonomous MAC scheduling protocols in prior research. Section III presents the problem statement and objectives. The system model is defined in Section IV, while Section V introduces ACP, the Autonomous Collision-free Protocol, to address limitations of existing methods. The experimental setup and analysis of simulation results are presented in sections VI and VII. Section VIII concludes the paper with proposed future research directions.

II. RELATED WORK

A. *Autonomous MAC scheduling*

In contrast to centralised and distributed scheduling where communication schedules are generated by a single coordinator node or as a result of a negotiation process between neighbouring network nodes, in autonomous scheduling, nodes compute their schedules based solely on locally available information. Autonomous techniques commonly adopt time-slotted MAC on the Data Link Layer, dividing time into discrete slots assigned for either transmission (Tx) or reception (Rx).

Duquennoy et al. [5] introduce the concept of slot assignment based on hashed sender or receiver node IDs, implemented in the Orchestra protocol. While Orchestra defines a fast and low-overhead scheduling method, its design presents a few intrinsic limitations. In the receiver-based approach, a node is allocated a single shared Rx slot to receive packets from any neighbour, causing contention and frequent packet collisions. The sender-based approach reduces contention but requires each node to allocate an Rx slot for every potential transmitter, increasing the duty cycle. Furthermore, Orchestra operates on a single frequency channel, which exacerbates contention.

The ALICE protocol presented by Kim et al. [6] addresses these challenges by allocating communication slots to bidirectional links, incorporating both sender and receiver IDs along with the current Absolute Slot Frame Number (ASFN). This method therefore requires the re-computation of schedules at each superframe. Although ALICE mitigates recurring collisions, in the reported experiments the authors assume non-adaptive, low-traffic conditions, which may not reflect more dynamic scenarios. Furthermore, schedule regeneration after each superframe could be avoided with an effective collision resolution mechanism, thereby reducing the associated computational overhead. Allocation of a single Tx and Rx slot per link limits the protocol's effectiveness in high-traffic scenarios, especially near gateway nodes where congestion is more severe.

The Exclusive Cell Allocation (ECA) protocol [7] introduces the concept of schedule-awareness of neighbouring nodes through *local regulation*, a mechanism in which parent nodes assign sequential local indices to child nodes. The index assignment leverages existing routing and network control messages, thereby incurring no additional overhead, which meets the primary objective of the autonomous paradigm. While nodes can infer the presence of sibling nodes based on their local index, they remain unaware of nodes with higher indices.

One notable shortcoming of ECA is the delayed reassignment of local indices. When a child node disconnects, its index is only reassigned when a new node joins, resulting in temporarily blocked slots. Additionally, the protocol does not define or easily accommodate a mechanism for allocating more slots to adapt to increased network traffic. Any feasible further assignment utilising *local regulation* could lead to collisions or an unfair distribution of resources. For instance, in a scenario where initial Tx and Rx slots are allocated sequentially based on local indices, a node n_i would be unaware of slots assigned to nodes $n_j, n_k, \dots, n_x$ (where $j,k,\dots,x > i$), increasing the probability of conflicts. Similarly, allocating additional slots simultaneously with the first slots for each node might leave no available slots for nodes with a higher local index. By design, the parent node's schedule is excluded from consideration to prevent recursive schedule changes across the network tree hierarchy. However, this can lead to interference between nodes at different ranks. This trade-off avoids triggering cascading schedule updates but may introduce transmission conflicts. Furthermore, the protocol does not support deallocation of unused slots.

Chung et al. [8] outline a traffic-aware autonomous protocol, which also accounts for the routing distance from the network root to address potential traffic congestion and bottlenecks at lower ranks, closer to the root. Integrating with the Routing Protocol for Low-Power and Lossy Networks (RPL) [9], in IM-LAS the slotframe length for each node n (L_{SFn}) is determined at each RPL rank based on n's sub-tree size, the global slotframe length (L_{SF}), and the total network size N. This approach implicitly assumes a fixed network size, which may not hold true in all deployments. While proportional bandwidth allocation based on traffic demand is logical, the formulas for determining slotframe length (L_{SFn}) and Rx/Tx time offsets can lead to collisions, especially among sibling nodes with similar sub-tree sizes, resulting in comparable slotframe lengths. Furthermore, the allocation formulas increase the likelihood of clustering, as links are scheduled within a sub-slotframe smaller than the global slotframe. This compresses the time available for transmissions.

The A3 (Adaptive Autonomous Allocation) protocol, introduced by Kim et al. [10], incorporates a Load Estimation Algorithm, a "receive load estimation" scheme, where the receiver independently estimates sender traffic based on Exponentially Weighted Moving Average (EWMA). It also uses multi-slot scheduling to prevent under- or over-provisioning communication slots. A3's limitation lies in the design's strict emphasis on fully autonomous operation, in contrast to ECA for instance, where information distribution utilising network control messages which would be transmitted in any case allows the protocol to meet the underlying objective of autonomous scheduling and avoid communication overhead associated with medium access control. This constraint limits A3's accuracy, as it estimates traffic load rather than utilising direct data. Estimation-based allocation of slots may also lead to temporary misalignment between receiving and sender nodes as traffic fluctuates. This discrepancy highlights the need for an additional synchronisation mechanism between neighbouring nodes.

OST [11] combines Orchestra's receiver-based operation with on-demand slot allocation by utilising autonomous as well as periodic provisioning over multiple unicast slotframes. The approach seeks to harness the simplicity of ID-based slot assignment and leverage existing data and ACK frames along with a negotiating procedure to dynamically adjust the number of slots based on demand.

Practical WSN applications commonly rely on low-power, resource-constrained devices with limited battery life, making energy-efficient operation and effective slot management essential. Autonomous scheduling, combined with time-slotted MAC protocols and network-wide time synchronisation enables nodes to reduce radio duty cycles through low-power operation during inactive periods. However, under-provisioning can cause congestion, increased latency, buffer overflows, and degraded QoS, while over-provisioning results in higher duty cycles, contention, and wasted channel resources. Orchestra, ALICE, and ECA lack mechanisms for dynamically adjusting the number of reserved slots, meanwhile the statistical approach of A3 and distributed method of OST fail to achieve optimal results. In contrast, Chung et al. [5] propose a straightforward yet effective solution using a reserved bit in the IEEE 802.15.4 MAC frame to indicate the activation of subsequent slots in the superframe.

III. PROBLEM STATEMENT AND OBJECTIVES

Practical WSN use cases commonly involve the aggregation of sensor data toward gateway nodes in multi-hop hierarchical configurations. Limited throughput and congestion remain key practical challenges, particularly during high-traffic periods that may be triggered by equipment failures or emergencies. The spatial distribution of nodes may affect the severity of internal interference resulting in retransmissions, degraded latency and energy consumption. Responsiveness to changing bandwidth requirements is also essential for energy-efficient operation while addressing potentially non-uniform throughput demand across network segments.

Autonomous, hashing and link-based scheduling enable rapid adaptation to operational changes. However, their efficiency can be constrained by a strict emphasis on full autonomy. Integrating autonomous approaches with mechanisms for sharing local network state within clusters, such as the presence of nodes within transmission range or slot allocation needs, can enhance scheduling performance. An important objective is to achieve information exchange without incurring additional communication overhead and power consumption.

In real-time criticality-aware applications, maintaining QoS across varying criticality levels requires efficient resource allocation to ensure timely delivery of high-priority data while minimising disruptions to lower-priority services. This paper addresses these challenges with the following objectives:

1. Propose a mechanism for distributing network state information within local node clusters using Information Element fields in IEEE 802.15.4 frames.
2. Define and evaluate an autonomous, collision-free protocol, ACP, which incorporates this mechanism and supports allocation of a preconfigured number of extra transmission slots.
3. Define and evaluate a mechanism for dynamically adjusting active slot allocations to meet traffic demands while ensuring energy efficiency.
4. Evaluate compliance with reliability and timing requirements, assuming the deadline for each forwarded data packet is known.

IV. System model

We extend the system model described in Pinter et al. [12], considering a WSN configured in a multi-hop, tree topology with a single sink and multiple, distributed field devices (Fig. 1), denoted $N = \{n_1, n_2, \ldots, n_k\}$. For each node $n_p \in N$, the local cluster C_p is defined as $C_p = \{n_p, (n_{p,1}, n_{p,2}, \ldots, n_{p,m})\}$, where m is the number of child nodes connected to n_p. Nodes are resource constrained devices, equipped with half-duplex radio transceivers and limited transmission range, necessitating peer-to-peer packet switching. Each node communicates with one neighbouring node per time slot, either transmitting or receiving, as determined by its MAC schedule.

The IEEE 802.15.4 protocol [13] defines 16 non-interfering frequency channels in the 2.4GHz band available for transmission, although it is assumed that not all of these are accessible to the network due to factors such as external interference or regulatory constraints: $C\ (1 < C < 16)$.

Medium access control scheduling adopts TSCH. Communication is time-synchronised and occurs within discrete time slots with sufficient duration to allow for the transmission of a single packet and its acknowledgment (L_{slot} = ~ 10ms). Nodes communicate at different frequencies in each time slot, following a predetermined hopping sequence. It is assumed that clock drift between nodes is negligible ($L_{drift} \ll L_{slot}$). This is a reasonable assumption since, in the MAC scheduling protocols selected for our experimental setup, beacon frame transmissions are used for clock synchronisation between nodes [5], [6].

We assume the use of RPL (Routing Protocol for Low-Power and Lossy Networks) for network management and

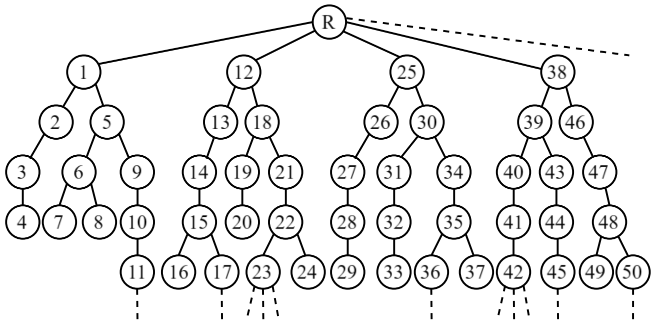

Fig. 1. Partial routing topology in the experiment setup. The network consists of a single RPL DODAG (one root node) and 125 field devices.

packet routing [9]. RPL network topologies form Directed Acyclic Graphs (DAGs), which can be partitioned into one or more Destination-Oriented DAGs (DODAGs), each associated with a sink node. Lower Rank values indicate closer proximity to the root, establishing a partial order of node positions within the DODAG.

Each node n_k executes a set of tasks, $T_k = \{\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,l}\}$, defined by phase ($Ph_{k,l}$), period ($P_{k,l}$), relative deadline ($D_{k,l}$, where $D_{k,l} \leqslant P_{k,l}$), and criticality ($\Phi_{k,l}$). Tasks release jobs that generate application data, transmitted as packets over a multi-hop path to the network sink. Packets, generated on a node or received from descendant nodes for forwarding, are placed in a queue and selected for transmission using criticality monotonic scheduling. In alignment with Chen et al. [14], it is assumed that each node can store a minimum of 50 packets. Packets selected for forwarding are dequeued before transmission. Therefore, packets lost due to collisions are not subject to re-transmission. Upward traffic flow is assumed to be dominant for application data.

V. Proposed Approach

A. *Disseminating network state information*

Strict interpretation of autonomy in MAC scheduling, meaning schedules are generated based solely on locally available or computable information may limit performance optimisation. Probabilistic methods, utilising time-series data or information about recent traffic patterns to infer the network's current state may mitigate issues like contention and improve resource allocation [10]. However, leveraging up-to-date data directly provided to nodes by their neighbours enhances decision-making accuracy. A common approach to achieve this is through control messages, essential for network management and transmitted independently of the adopted MAC scheduling method [7], [8]. This relaxed definition of autonomous scheduling retains the key objective of the autonomous paradigm: allowing nodes to generate their own schedules without incurring the communication overhead inherent in centralised or distributed scheduling.

ID hashing-based autonomous scheduling is vulnerable to contention. However, we argue that collision resolution among sibling nodes in a tree topology is feasible if:

1. All nodes in a local cluster Cp know each other's IDs, as well as the ID of the parent node n_o of n_p, enabling collision-free scheduling at Cp's RPL rank.

2. Any collision-resolution decisions that cause deviation from n_p's default schedule, are known to all its child nodes, ensuring they can avoid contention with transmissions at lower RPL ranks.

Formally, let ID_p represent the set of IDs known to all nodes in C_p, and n_o be the parent node of n_p. Then $\forall\, n_{p,i} \in C_p$, $ID(n_{p,i}) \in ID_p$ and $ID(n_o) \in ID_p$. Let CR_p represent the set of collision-resolution probes for n_p. Then $\forall\, n_{p,i}\ CR_p$ is known. CR_p is defined as a set of two-tuples: $CR_p = \{(s_p, r_p) \mid \forall\, s_p \in$ slots assigned to $n_p\}$, where s_p represents the position of the slot in the allocation sequence and r_p represents the number of probes conducted to find the next available slot, indicating a deviation from the default slot assignment due to collision.

The IEEE 802.15.4 standard defines a variable-sized Enhanced Beacon (EB) frame, adaptable to specific application requirements. Considering a typical payload range of 72-116 bytes accounting for link-layer framing and assuming 16-bit short addressing, a single EB frame can encode 30 to 50 node IDs, along with the collision resolution set CR_p for the parent node n_p of cluster C_p. Aligned with the standard, as well as the recommendations for peer-to-peer, multi-hop TSCH networking in [4], [15], the approach follows the scheduling method described in [5], [6]. EB frames are broadcast at preconfigured intervals by all intermediate nodes, while this is not required for reduced-function devices, which are always leaf nodes.

B. *Network management and Medium Access Control*

ACP adopts hashing and link-based cell allocation proposed by Duquennoy et al. and Kim et al. [5] [6], extending these with a collision avoidance mechanism to reduce contention. Additionally, the protocol facilitates the assignment of extra slots and dynamic slot de-activation to address bandwidth limitations while preserving energy efficiency.

Orchestra-based protocols maintain a hierarchical organisation of three traffic planes – TSCH, RPL, and Application planes in priority order – dedicated respectively to the transmission of Enhanced Beacon (EB) frames for time synchronisation and network association, RPL broadcast control messages for routing and network management and unicast packet switching for application data transfer. The TSCH and RPL planes are assigned a single dedicated channel each to avoid the interference across different types of traffic. ACP adopts multi-channel operation on the Application plane. Slotframe lengths differ across traffic planes and are coprime to avoid persistent collisions of overlapping slots. When addressing overlaps, priority is given to traffic scheduled in the highest-priority plane.

a) Default slot assignment

Directional link-based scheduling, described in Kim et al. [6] significantly improves network performance compared with Orchestra's sender- and receiver-based techniques [5], which suffer from high duty cycle operation and contention. Adopting this refined approach, in ACP the default time and channel offsets for a directional link between nodes n_k and n_l are determined as

$$t_{k,l} = mod(\ hash(\alpha\ ID(n_k)+ID(n_l)),\ SL_{APP}) \quad (1)$$

$$c_{k,l} = mod(\ hash(\alpha\ ID(n_k)+ID(n_l)),\ C_{APP} - 1)+1 \quad (2)$$

where the coefficient α is used to differentiate traffic directions. SL_{APP} represents the slotframe length and C_{APP} denotes the number of channels assigned to the Application plane. One notable difference between ALICE and ACP is that ALICE incorporates the Absolute Slotframe Number (ASFN) in the cell allocation formulas to avoid recurring overlaps of scheduled cells across traffic planes. Due to its collision-avoidance mechanism this is not required for ACP.

b) Collision resolution and multi-slot allocation

A key limitation of the Orchestra and ALICE protocols is that slot assignment is restricted to a single Tx and Rx slot for each link, significantly constraining network throughput. ACP addresses this problem by iteratively allocating a preconfigured number of cells for each link. During the initial iteration, default slots are allocated to all nodes. Subsequent assignments use linear probing, incrementing both time and channel offsets by 1 until an unused slot is identified. This approach ensures fair allocation. Importantly, the collision resolution mechanism is decoupled from the protocol itself and can be configured independently. While linear probing leads to clustering of scheduled cells in the slot table, it is chosen as the default approach due to its simplicity and low computational overhead.

Fig. 2 presents the pseudocode for computing the Application plane schedule, which consists of three phases:

1. Parent Node Schedule Generation: Nodes, excluding the root and those lacking a direct link to it, generate their parent node's (n_p) schedule, ensuring transmissions do not interfere with traffic at lower RPL ranks. Collision resolution may result in deviations from n_p's default schedule. Therefore, resolution probes are recorded and shared with descendant nodes. This information is encoded in Enhanced Beacon frames.
2. Sibling Schedule Generation: Nodes iteratively generate their own and their siblings' schedules for links with their shared parent node. During each iteration, nodes are assigned a single Tx and Rx slot in the order of their sorted IDs.
3. Descendant Schedule Generation: Finally, nodes allocate Tx and Rx slots for transmissions with their descendants in a similar fashion. This process is recursively applied across the entire network.

The awareness of nodes of their parent's schedule means that updates may trigger recursive schedule changes propagating down the network tree. This approach differs from the ECA protocol's method, which localises the impact of network changes and implicitly assumes low contention across neighbouring RPL ranks [7].

ACP is designed to be more robust in dense deployments and high traffic scenarios, where adaptivity is crucial. While recursive updates may occur, they are expected to have negligible performance impact with autonomous scheduling. Compared to the ALICE protocol, which recomputes schedules across all nodes during each superframe to avoid recurring collisions, ACP significantly reduces the frequency of schedule updates, even under dynamic operating conditions.

C. Dynamic slot activation

While allocating extra slots increases network bandwidth, energy efficiency remains crucial for WSNs, typically comprising low-power, resource-constrained devices. ACP follows an approach similar to that of Chung et al. [7]. The IM-LAS protocol uses the “MorePacketToSend” flag, encoded in a reserved bit of the IEEE 802.154 data frame, to indicate whether subsequent cells in the current superframe should be activated.

In contrast, ACP keeps cells active by default, and nodes indicate the number of cells that can be deactivated based on their current traffic load. This approach ensures more robust operation. In IM-LAS, the failure to receive the first packet within a superframe leads to the deactivation of all subsequent cells, potentially increasing network latency. By considering the current schedule, ACP calculates and informs the receiving node of the number of slots that can be safely deactivated according to current traffic demands. This approach is particularly effective when application data is generated by periodic tasks.

VI. EXPERIMENTAL SETUP

The evaluation framework is adapted from Pinter et al. [12], with significant modifications to network topology and node density, as well as adjustments to traffic periodicity. ACP’s performance is assessed in a simulated environment using VisualSense, a suite of software packages for WSN modelling that extends the Ptolemy II discrete event simulation engine [16], [17], [18], [19]. It is compared against two baseline scheduling methods: Orchestra and ALICE. Performance, network stability, and energy efficiency are evaluated based on end-to-end Packet Delivery Ratio (PDR), latency, topology changes induced by internal interference, and average duty cycle. The following subsections outline the experimental setup across the Data Link, Network, and Application layers of the protocol stack.

A. Data Link layer

All evaluated protocols operate across three traffic planes: TSCH, RPL, and Application. By default, Orchestra assigns a single frequency channel per plane. For a more balanced comparison, we adopt Kim et al.’s enhancement, allocating four channels to Application Plane traffic (C_{APP} (4)) while reserving dedicated channels for EB frame transmission and RPL broadcast messages in the TSCH and RPL planes. Replicating the setup in [5], the simulations use the following slotframe lengths: $SL_{TSCH} = 397$, $SL_{RPL} = 31$, $SL_{APP} = 47$.

Orchestra allocates cells based on each node’s ID and its RPL neighbour relationship, following either receiver- or sender-based scheduling. In receiver-based mode (O-RB), a shared slot is assigned to node n_k at time offset

$$t_{Rx} = mod(hash(IDn_k), SL_{APP}) \quad (3)$$

Algorithm 1 Generate Application Frame Schedule

```
function generateSchedule(int slotCount)
    crp ← getCRp()                 // Collision resolutions for n_p
    ids ← getIDs()                 // n_o and cluster IDs
    nodeId ← getNodeId()           // this node's ID
    rplRank ← getRank()            // this node's hop distance from root
    slotTable ← new SlotTable()

    // Generate n_o - n_p link schedule
    if rplRank > 1 then
        for i ← 0 to slotCount do
            npResolution = ← crp.get(i)
            if npResolution != null then
                // n_p's schedule deviates from default, resolve collision
                slotTable.addSlots(ids.get(n_o), ids.get(n_p), npResolution)
            else
                // Use default allocation
                slotTable.addSlots(ids.get(n_o), ids.get(n_p))
            end if
        end for
    end if

    // Generate n_p - siblings' schedules
    orderedIds ← ids.getSortedSiblingIds()
    cr ← new CollisionResolutions()    // This node's collision resolutions
    for siblingId in orderedIds do
        resolution ← slotTable.addSlots(ids.get(n_p), siblingId)
        if siblingId == nodeId then
            cr.add(resolution)
        end if
    end for

    // Generate n_p- siblings schedules
    orderedChildIds ← getChildIDs()
    for childId in orderedChildIds do
        slotTable.addSlots(nodeId, childId)
    end for

    return slotTable
end function
```

Fig. 2. Pseudocode of the ACP application frame scheduling algorithm.

for packet reception from any neighbour. However, a key limitation of this method is contention, which is particularly evident in dense networks and during high-traffic periods, as multiple neighbours may attempt to communicate with the a node simultaneously [6], [7], [8]. To ensure an unbiased comparison with ACP’s collision-avoidance approach, Orchestra’s sender-based scheduling mode (O-SB) is used as the baseline. In O-SB a shared slot is allocated for packet transmission at time and channel offsets

$$t_{Tx} = mod(hash(IDn_k), SL_{APP}) \quad (4)$$

$$c_{Tx} = mod(hash(IDn_k), C_{APP}) \quad (5)$$

This method enables neighbouring nodes to determine transmission and reception times and frequencies with node n_k. While O-SB reduces contention, each node transmits only once per slotframe, increasing latency. Tx and Rx time slots for EB frames follow a similar allocation based on IDn_k and SL_{TSCH}, while RPL messages are broadcast in a predefined shared slot. ALICE and ACP use the cell allocation approaches described in the previous section. ACP is evaluated in two configurations, with 2 and 4 Tx and Rx slots allocated *per link* in a node’s schedule, denoted as ACP 2 and 4 respectively.

B. Network layer

The simulated sensor network consists of a single RPL DODAG instance with 124 field devices and a single root node (Fig. 1). We evaluate ACP’s ability to mitigate internal interference caused by MAC schedule clashes within a single cluster. Hashing-based autonomous scheduling is particularly sensitive to the spatial distribution of nodes, as it lacks the benefits of centralised or distributed scheduling, where nodes receive globally optimised schedules from a single coordinator node or negotiate to resolve conflicts. Let ρ, R, N and A denote node density, transmission range, the total number of nodes and network area respectively. Network density is then measured using the Communication Range-Based Density (CRD) as follows:

$$CRD = \rho \times R^2 \tag{6}$$

$$\rho = \frac{N}{A} \tag{7}$$

In our setup $CRD \approx 19.59$. RPL's adaptive behaviour is replicated in the simulator. Network nodes assess link health based on unacknowledged transmissions. When nodes become unavailable due to internal interference and packet collisions, network repair is triggered. During network bootstrap, joining nodes select parent nodes with the lowest RPL Rank and the fewest descendant nodes.

C. Application layer

Network performance is assessed under three scenarios outlined in Table 1. High traffic conditions (H1, H2) simulate severe loads where MED and LO criticality flows may be affected, as criticality monotonic scheduling prioritises real-time (HI criticality) flows. Conversely, the moderate traffic scenario (M) evaluates system behaviour under a manageable load, where HI criticality flows experience minimal impact, and QoS for lower criticality flows is expected to gradually improve.

All tasks are periodic, generating bursty traffic patterns that test the protocols' adaptability to handle high demand while avoiding resource overprovisioning to ensure energy efficiency. In all cases, packet deadlines align with the period of their generating tasks ($D_i = P_i$). To prevent exact overlapping of job releases for periodic tasks, a small random offset was applied to the phase of each task. Packets are dequeued when selected for transmission and are not subject to retransmissions in case of collisions. These are noted as unacknowledged frames by the sender node.

TABLE I. EXPERIMENTAL NETWORK TRAFFIC SCENARIOS

Network traffic scenarios	Scenario ID	Task periods (sec) [a]		
		P_{HI}	P_{MED}	P_{LO}
High traffic	H1	2.5	1.25	2.5
High traffic	H2	7.5	3.75	7.5
Medium traffic	M	15	7.5	15

a. Packets' relative deadlines align with the period of the task that generated them ($D_i = P_i$)

VII. Performance Evaluation

A. End-to-end PDR

Criticality monotonic packet scheduling does not employ early packet drop strategies described in [12], so end-to-end PDR is adjusted for packets that reach the gateway (DODAG root) within their deadlines. Provisioning additional Tx and Rx cells in ACP schedules significantly improves throughput compared to Orchestra and ALICE. This improvement is consistent across all scenarios, with MED and LO criticality flows benefiting earlier as traffic conditions improve (Fig. 3).

Simulation results highlight a key distinction between node- and link-based scheduling, particularly during traffic bursts. Both ALICE and ACP allocate at least one Tx cell *per link* in the application frame, increasing the likelihood of clearing medium- and low-criticality packet queues between HI-criticality job releases. In contrast, O-SB assigns a single transmission cell per node, increasing the probability that a HI-criticality packet gets queued before the next scheduled Tx slot and selected for transmission. Consequently, ALICE achieves a mixed distribution of criticality in delivered packets, whereas O-SB attains a higher end-to-end PDR for HI-criticality flows.

Another notable finding is that ACP 4 did not outperform ACP 2 in any of the scenarios. This is due to the combination of network density and ACP's fair cell allocation among clusters and descendant nodes. By default, ACP assigns the same number of Tx and Rx cells to each neighbouring link, potentially leading to over- or under-provisioned nodes depending on spatial distribution and the resulting network topology. This highlights the need for demand-based, proportional cell allocation. It is worth noting that this issue only arises in highly dense clusters and during high traffic.

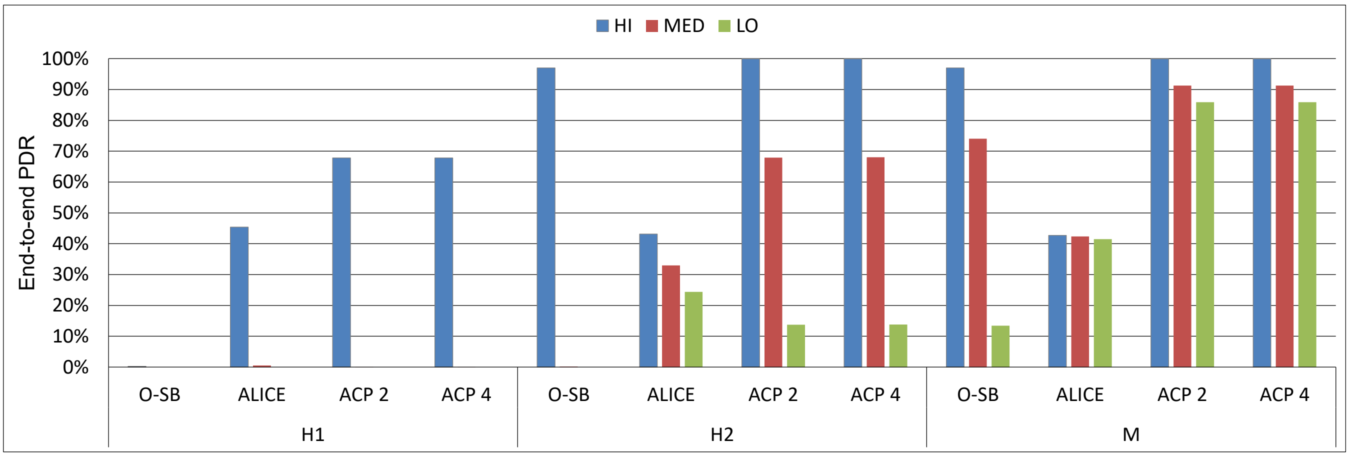

Fig. 3. End-to end PDR. Link-based scheduling may lead to mixed distribution of delivered packet criticality. ACP's fair slot allocation may lead to sub-optimal, yet fully provisioned schedules in high density clusters. This leads to no performance improvement in ACP 4 compared with ACP 2.

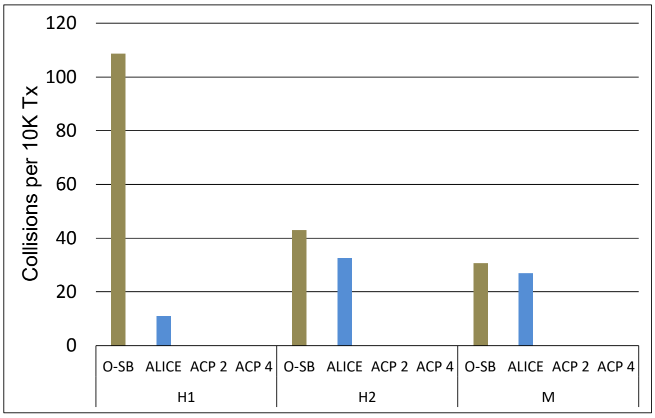

Fig. 4. Normalised packet collisions. Contention is significantly mitigated by link-based scheduling, introduced with ALICE. ACP, however, facilitates full schedule allocation without leading to increased collisions.

B. Network contention and collision avoidance

We evaluate ACP's effectiveness in reducing contention caused by overlapping MAC schedules within node clusters. It is important to note that link-based scheduling, as introduced with ALICE, already significantly mitigates contention. The key advantage of ACP's collision-resolution method is that it facilitates the allocation of more slots, thereby greatly improving throughput without increasing contention. In our simulations, we consider packet collisions arising from overlapping MAC schedules. To provide a fair comparison, the collision count is normalised by the transmission count, as link-based protocols allocate more links in a single frame for the same number of neighbouring nodes. Fig. 4 shows that while ALICE mitigates contention in all simulation scenarios, ACP completely eliminates it. Orchestra's node-based scheduling can potentially lead to recurring collisions, which is particularly evident in the first scenario (H1).

C. Energy efficiency

Contention-free scheduling of multiple cells effectively improves throughput. However, in resource-constrained WSNs, overprovisioning may prove prohibitively energy-inefficient. ACP's dynamic cell deactivation mechanism enables rapid adaptation to varying network traffic within a single superframe, avoiding the synchronisation lag seen in estimation-based schemes like the one introduced by Kim et al. [10]. Although simulation results show that ACP operates at a higher average duty cycle across the network (Fig. 5), it is important to consider the corresponding increase in throughput. We define the normalised per-packet energy cost (ε_N) using the formula below, which offers a more precise measure of energy efficiency by factoring in duty cycle (DC), total timeslots during the simulation (N_{TS}), and throughput, quantified by the number of packets delivered within their deadlines (P_d):

$$\varepsilon_N = (N_{TS} \times DC) / P_d \tag{8}$$

ACP's cell deactivation mechanism effectively balances energy efficiency with accommodating high traffic, outperforming ALICE in all traffic scenarios (Fig. 6). While Orchestra achieves better per-packet energy cost under moderate traffic, this comes at a cost of significantly lower throughput.

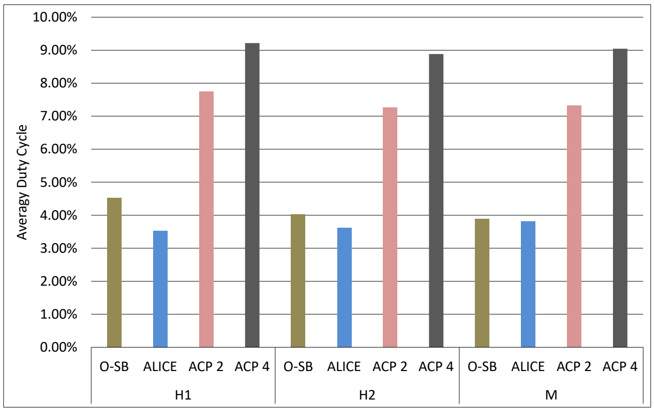

Fig. 5. Average Duty Cycle. ACP operates at a higher energy utilisation rate to accommodate high traffic demand.

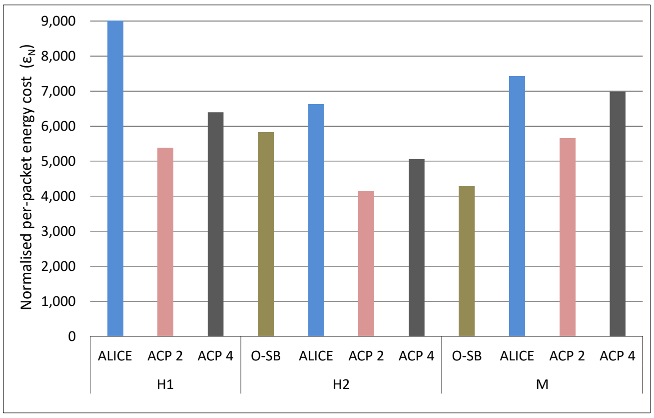

Fig. 6. Normalised per-packet energy cost. The O-SB protocol demonstrated significantly higher per-packet energy cost in scenario H1 due to recurring collisions that impacted throughput. While this highlights a limitation of node-based autonomous scheduling, the result is excluded from the chart to prevent bias in the comparison. ACP achieves better per-packet energy cost than ALICE in all scenarios and outperforms O-SB under heavy traffic demand.

VIII. Conclusion and Future Work

The proposed autonomous scheduling protocol, ACP, integrates three fundamental mechanisms to enable high-throughput, energy-efficient, and collision-free MAC scheduling: (1) dissemination of network state information incurring no additional overhead beyond the transmission of Enhanced Beacon frames, (2) a collision resolution mechanism that allocates multiple transmission cells within a node's schedule, and (3) a slot deactivation mechanism. Simulation results demonstrate that ACP effectively eliminates contention among neighbouring nodes within a cluster caused by MAC schedule conflicts, significantly improving throughput while maintaining energy-efficiency.

Despite these contributions, the following aspects of autonomous scheduling require further research. ACP's approach of allocating a predefined number of cells per link may lead to starvation at higher ranks in dense clusters if parent nodes' schedules are fully provisioned, or under-utilisation of bandwidth if insufficient extra cells are scheduled. Additionally, equal allocations among cluster nodes may result in over- or under-provisioning, which highlights the need for demand-based allocation on a per-link basis.

While ACP mitigates contention within a cluster, it does not address network-internal interference among neighbouring nodes that are not aware of each other. Furthermore, the proposed linear probing-based collision resolution method results in cell clustering within a node's schedule, potentially exacerbating internal interference. Alternative techniques, such as quadratic probing with gradually increased probe distances, may mitigate this effect, while a more robust mechanism for encoding and disseminating additional network information via control frames may help manage interference across clusters.

In its current form, ACP is most suitable for networks configured in a tree topology with a single root and predominantly upward traffic flow. To prevent collisions at lower RPL ranks, nodes consider only their parent's schedule. Future extensions could incorporate all receiving nodes, accommodating more diverse traffic patterns.

References

[1] A. Pagano, D. Croce, I. Tinnirello, and G. Vitale, "A Survey on LoRa for Smart Agriculture: Current Trends and Future Perspectives," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3664–3679, 2023, doi: 10.1109/JIOT.2022.3230505.

[2] M. Rychlicki, Z. Kasprzyk, M. Pełka, and A. Rosiński, "Use of Wireless Sensor Networks for Area-Based Speed Control and Traffic Monitoring," *Applied Sciences*, 2024, [Online]. Available: https://api.semanticscholar.org/CorpusID:273307242

[3] W. S. Jang and W. M. Healy, "Wireless sensor network performance metrics for building applications," *Energy and Buildings*, vol. 42, no. 6, pp. 862–868, 2010, doi: https://doi.org/10.1016/j.enbuild.2009.12.008.

[4] T. Watteyne, M. R. Palattella, and L. A. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement," Internet Engineering Task Force, Request for Comments RFC 7554, May 2015. doi: 10.17487/RFC7554.

[5] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," Nov. 2015. doi: 10.1145/2809695.2809714.

[6] S. Kim, H. Kim, and C. Kim, "ALICE: autonomous link-based cell scheduling for TSCH," Apr. 2019. doi: 10.1145/3302506.3310394.

[7] J. Shin, H. Kim, J. Paek, and S. Bahk, "ECA: Exclusive Cell Allocation for Autonomous Scheduling in Time-Slotted Channel Hopping," in *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2023, pp. 350–358. doi: 10.1109/MASS58611.2023.00050.

[8] H.-J. Lee and S.-H. Chung, "Link-Based Autonomous Cell Scheduling for IEEE 802.15.4e TSCH With Improved Traffic Throughput," *IEEE Sensors Journal*, vol. 21, no. 19, pp. 22104–22112, 2021, doi: 10.1109/JSEN.2021.3104858.

[9] R. Alexander *et al.*, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force, Request for Comments RFC 6550, Mar. 2012. doi: 10.17487/RFC6550.

[10] S. Kim, H.-S. Kim, and C. Kim, "A3: Adaptive Autonomous Allocation of TSCH Slots," in *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, in IPSN '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 299–314. doi: 10.1145/3412382.3458273.

[11] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-Demand TSCH Scheduling with Traffic-Awareness," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 69–78. doi: 10.1109/INFOCOM41043.2020.9155496.

[12] A. Pinter, L. S. Indrusiak, and I. Gray, "Evaluation of Early Packet Drop Scheduling Policies in Criticality-Aware Wireless Sensor Networks," in *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, 2024, pp. 1–9. doi: 10.1109/ISORC61049.2024.10551348.

[13] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020, doi: 10.1109/IEEESTD.2020.9144691.

[14] X. Chen, X. Shi, and Y. Wang, "Packet Scheduling Algorithm Based on Priority Adjustment in Wireless Sensor Networks," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2022, pp. 1632–1639. doi: 10.1109/ITAIC54216.2022.9836768.

[15] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," no. 8180. in Request for Comments. RFC Editor, May 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8180

[16] P. Baldwin, S. Kohli, E. A. Lee, and X. Liu, "VisualSense: Visual Modeling for Wireless and Sensor Network Systems," University of California, Berkeley, UCB/ERL M04/08, Apr. 2004. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/759.html

[17] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "Modeling of Sensor Nets in Ptolemy II," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, in IPSN '04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 359–368. doi: 10.1145/984622.984675.

[18] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. [Online]. Available: http://ptolemy.org/books/Systems

[19] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng, "Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II)," EECS Department, University of California, Berkeley, UCB/EECS-2008-28, Apr. 2008. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-28.html