

CLA-Score: A Criticality Loss Age Packet Selection Policy for Heterogeneous Traffic in IEEE 802.15.4e TSCH Networks

Mo Ringbe Saynbe

Inria

Lille, France

mo-ringbe.saynbe@inria.fr

Carol Habib

Inria

Lille, France

carol.habib@inria.fr

Nathalie Mitton

Inria

Lille, France

nathalie.mitton@inria.fr

Abstract—Industrial Internet of Things (IIoT) networks, particularly those supporting Supervisory Control and Data Acquisition (SCADA) systems, rely on wireless standards derived from the Time-Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4e, such as WirelessHART and ISA-100.11a, for their determinism guarantees and robustness against interference. In these environments, the coexistence of periodic sensor readings and critical event-driven alarms creates a structural tension at the local queue of each sensor node (a constrained device generating and forwarding packets toward a sink): when packets of different traffic classes (alarms, monitoring, telemetry) compete for a transmission opportunity, the node-local selection policy directly conditions the network’s ability to meet the latency constraints of the most urgent flows. This work proposes CLA-Score (Criticality Loss Age score), a weighted scoring policy for packet selection at the sensor node level, combining three factors: the criticality of the traffic class, the observed loss rate, and the packet’s queuing age. The mechanism is fully node-local, requires no additional signaling, and remains compatible with the standard TSCH stack. We evaluate the policy on the SLICES-FR Lille testbed with 75 sensor nodes under heterogeneous traffic, and show that it improves the queue admission rate of critical alarm flows from 88% (FIFO) and 93% (strict priority) to 99% at low load, and from 89% and 92% to 97% at high load.

Index Terms—IIoT, IEEE 802.15.4e, TSCH, SCADA, packet selection, CLA-Score, quality of service, SLICES-FR.

I. INTRODUCTION

The Industrial Internet of Things (IIoT) demands reliable, low-latency, energy-efficient wireless communication, for which the IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) mode combines deterministic time-slotted access with channel hopping to resist industrial interference [1]. Yet real deployments mix *heterogeneous* traffic: periodic monitoring and telemetry coexist with sporadic, latency-critical events (alarms, faults) typical of Supervisory Control and Data Acquisition (SCADA) systems. When packets of different classes compete in a sensor node’s local queue, the node-local selection policy decides whether the most urgent flows meet their deadlines an alarm stuck behind older telemetry can miss its deadline even when radio capacity is available.

TSCH scheduling functions allocate cells by traffic demand: Orchestra autonomously from network state, the IETF 6TiSCH reference Minimal Scheduling Function (MSF) by adding or

removing dedicated cells from observed load, and On-demand Slotframe Tuning (OST) by tuning the slotframe length [1], [2]. Tuned for periodic flows, these adaptations act over many slotframes and absorb bursty event traffic poorly. Recent work likewise targets adaptive cell allocation [3]–[5], while *which packet* to serve first in the local queue is still left to simple heuristics: FIFO [1], which ignores class, or strict class-based priority [6], which ignores the packet’s dynamic state its queuing age and its class’s observed loss rate.

We introduce **CLA-Score**, a node-local weighted policy that selects the next packet from three factors class criticality, observed loss rate, and queuing age on a common scale, granting monotonic priority to aging critical packets without starving periodic flows. It requires no additional signaling and preserves the standard TSCH stack. On the SLICES-FR Lille testbed with 75 sensor nodes over 30 hours of live experiments, CLA-Score raises the queue admission rate of critical alarms to 97–99% a +4 to +11 percentage-point gain over FIFO and strict-priority baselines without degrading periodic flows.

The remainder is organized as follows: Section II reviews related work; Section III presents the system model; Section IV details the policy; Section V reports the evaluation on the SLICES-FR Lille testbed [7]; Section VI concludes.

II. RELATED WORK

We classify TSCH/6TiSCH transmission management by the level at which it operates cell allocation, traffic awareness, or queue-level packet selection and position our contribution.

A. Adaptive Cell Allocation

In TSCH, time is divided into slots grouped into a periodic *slotframe*; a slot paired with a channel offset forms a *cell*, dedicated to one transmitter or shared by contention. A scheduler decides this allocation, and most recent work targets this problem. Urke *et al.* [3] extend the autonomous *Layered* scheduler to heterogeneous traffic by using shared cells as a signaling channel. Lee and Chung [4] adjust the MSF add/delete cycle from link quality and queue occupancy. Reinforcement-learning schedulers Jurado-Lasso *et al.* [5] (HRL-TSCH) and the neural bandit of Raghuwanshi *et al.* [8]

adapt the slotframe to multi-objective criteria at the cost of a training phase. These mechanisms improve aggregate *capacity* but leave untouched the choice among competing packets when a cell opens.

B. Traffic-Aware Service Differentiation

This category adds awareness of traffic characteristics (flow priority, deadlines, fairness) to the slotframe-level allocation. Tabouche *et al.* [9] survey traffic-aware scheduling functions and identify the lack of distributed mechanisms jointly accounting for criticality, loss rate, and queuing age. Farag *et al.* [6] (REA-6TiSCH) let alarms hijack reserved cells a cell-level scheme orthogonal to our queue-level selection, its priority queue matching our FIFO-3Q baseline. Bommisetty and Venkatesh [10] propose *CBPF*, a proportional-fairness scheme on shared cells, without explicit criticality classes.

C. Packet Selection and Priority Queues

Closest to our work, queue-level packet selection has received limited attention. Contiki-NG [1] serves its queue in FIFO order; when differentiation is added, it takes the form of *strict* class priority: Farag *et al.* [6] place each class in a separate queue and always serve critical packets first. Such approaches ignore the dynamic state of the packet its queuing age and the observed loss rate of its class and may starve periodic flows under heavy alarm load.

D. Positioning

Our work differs along three axes. (i) **Granularity:** to the best of our knowledge, no prior work performs packet selection at the queue level using a multi-dimensional score; existing schedulers operate at the orthogonal cell-allocation level [3]–[5]. (ii) **Multidimensionality:** unlike strict priority [6], our score jointly combines *criticality* (a priori importance), *loss rate* (current loss conditions of the class), and *queuing age* (remaining deadline margin), guaranteeing monotonic priority to aging critical packets without starving periodic flows. (iii) **Cost and compatibility:** the formulation is fully node-local and needs neither signaling nor training, unlike RL-based approaches [5], [8], and preserves the standard TSCH stack.

III. SYSTEM MODEL AND MOTIVATION

A. Network Topology and Traffic Classes

We consider a single-hop star topology with N sensor nodes transmitting to a single collector. Each sensor node maintains a local queue Q of capacity Q packets. Each packet descriptor is augmented with its class and its enqueue timestamp.

Each sensor node generates three concurrent flows representative of a SCADA workload, each with its own end-to-end deadline (measured from packet generation at the source to reception at the collector):

- **Class 1 (Cls1):** Real-time alarms. Arrivals follow a Poisson process of rate λ_1 . End-to-end deadline $D_1 = 1$ s.
- **Class 2 (Cls2):** Periodic monitoring. Period $T_2 = 5$ s, end-to-end deadline $D_2 = 5$ s.

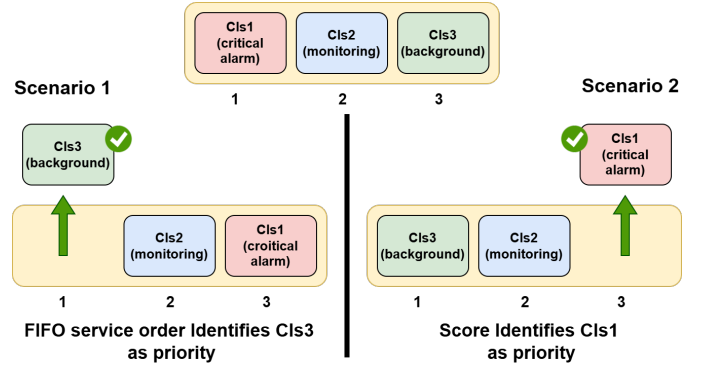


Fig. 1. Illustrative example. Three packets in the queue: Cls3 (background) is the oldest, Cls2 (monitoring) is intermediate, Cls1 (critical alarm) is the most recent. FIFO serves Cls3 first and delays the alarm; CLA-Score identifies Cls1 as the priority via $\tilde{c}_1$ and serves it immediately.

- **Class 3 (Cls3):** Slow telemetry. Period $T_3 = 30$ s, end-to-end deadline $D_3 = 15$ s.

B. Motivation

When a transmission cell opens, the sensor node must decide which packet from its local queue to send. Although seemingly secondary, this decision becomes critical under mixed SCADA traffic: as illustrated in Fig. 1, a critical alarm (Cls1) with 500 ms of queuing age may end up blocked behind older but less urgent telemetry (Cls3, 15 s) and monitoring (Cls2, 5 s) packets, missing its 1 s deadline even though radio capacity is sufficient. The reference TSCH implementation in Contiki-NG [1] serves the queue in *strict FIFO* order, with no distinction of class, queuing age, or observed loss rate. Under heterogeneous load, this choice mechanically produces three undesirable effects:

- critical packets wait behind older background packets in the queue;
- a packet that has already missed its deadline keeps occupying the queue and consuming transmission opportunities;
- no information about the observed loss rate (admission refusals and queue evictions observed at the node) is used to adjust the decision.

IV. CLA-SCORE: WEIGHTED PACKET SELECTION

This section presents CLA-Score (Criticality–Loss–Age score), the proposed selection policy designed to address the three limitations identified in Section III-B. The decision relies on a single score that explicitly separates the *observed facts* (measured online) from the *application belief* (fixed at design time).

A. Scoring Model

To each packet p waiting in the local queue Q of a sensor node, of class $i \in \mathcal{C}$, the following score is associated:

$$\text{score}(p) = \underbrace{w_a \cdot a_i + w_b \cdot b_p}_{\text{observed facts}} + \underbrace{\tilde{c}_i}_{\text{belief}}, \quad (1)$$

where:

- $a_i \in [0, 1]$ is the observed loss rate for class i , estimated online (Section IV-B);
- $b_p \in [0, 1]$ is the normalized queuing age of the packet (Section IV-C);
- $\tilde{c}_i \in [0, 1]$ is the normalized class criticality (Section IV-D);
- $w_a, w_b \in [0, 1]$ satisfy $w_a + w_b = 1$.

This score reflects a simple intuition: a packet should be served sooner when it belongs to a more critical class, when its class is currently experiencing losses, or when it is approaching its deadline. At each transmission opportunity, the packet with the highest score is selected:

$$p^* = \arg \max_{p \in \mathcal{Q}} \text{score}(p), \quad (2)$$

where $\mathcal{Q}$ denotes the local queue of the sensor node.

Facts/belief architecture. The *observed facts* term aggregates two measurements collected online by the node: the observed loss rate of the class (a_i) and the queuing age of the packet (b_p). As a convex combination ($w_a + w_b = 1$) of two values in $[0, 1]$, it remains bounded in $[0, 1]$. The *application belief* term $\tilde{c}_i \in [0, 1]$ encodes the priority *a priori* assigned to the class by the designer and dictates the nominal order in the absence of perturbation.

Starvation avoidance. Since the observed facts term is bounded by 1, a low-priority packet in a critical situation (high loss rate, deadline nearly reached) can raise its score by at most 1, enough to overtake the criticality gap separating it from the most critical class (recall $\tilde{c} = (1.00, 0.20, 0.10)$). No class is therefore permanently disadvantaged: the nominal order is deviated only when the situation requires it, ensuring monotonic priority to aging critical packets without starving periodic flows.

B. Loss Rate Estimation

The observed loss rate a_i is estimated online *at the node level* from local loss events. We define a loss as any packet of class i that fails to be transmitted from this node, either because it has been *refused at admission* into the queue (full buffer), or because it has been *removed from the queue* before transmission (eviction on deadline overflow, or to free space for a higher-priority packet). Radio transmission failures are not directly accounted for: they are handled by TSCH retransmissions and affect a_i only indirectly, when they lead to a deadline overflow.

At each arrival of a packet of class i , we observe a Bernoulli variable $X_n^{(i)} \in \{0, 1\}$ where $X_n^{(i)} = 1$ if the packet is rejected or evicted. To obtain a causal and smoothed estimate, we draw inspiration from the Random Early Detection (RED) mechanism [11], which uses an exponentially weighted moving average (EWMA) to estimate the average queue length. Transposed to our context:

$$a_i^{(n)} = (1 - \beta_i) \cdot a_i^{(n-1)} + \beta_i \cdot X_n^{(i)}, \quad (3)$$

TABLE I
PARAMETRIZATION OF THE THREE TRAFFIC CLASSES.

i	Class	Traffic	D_i	c_i	$\tilde{c}_i$
1	Critical	Sporadic Poisson	1 s	10	1.00
2	Monitoring	Periodic 5 s	5 s	5	0.20
3	Background	Periodic 30 s	15 s	2	0.10

with initialization $a_i^{(0)} = 0$. The update is *event-driven* (triggered at each arrival) rather than periodic, which prevents any drift during silent periods for infrequently activated classes.

We index the smoothing factor on the class ($\beta_1 = 0.9$ for critical, $\beta_2 = 0.5$ for monitoring, $\beta_3 = 0.2$ for background): a high value gives strong weight to recent observations and makes the estimator highly reactive, while a low value smooths out transient fluctuations. A separate estimator is maintained per class for a total cost of three scalars per node.

C. Queuing Age

The queuing age b_p is the fraction of the deadline already consumed by packet p of class i :

$$b_p = \min \left(\frac{t_{\text{now}} - t_{\text{arr}}(p)}{D_i}, 1 \right), \quad (4)$$

where $t_{\text{arr}}(p)$ is the enqueue timestamp and D_i is the end-to-end deadline of class i (i.e., D_1 , D_2 , or D_3 as defined in Section III-A). A freshly enqueued packet has $b_p \approx 0$, a packet at its deadline $b_p = 1$. Unlike a_i , b_p is recomputed at every transmission opportunity through SCORE, so a packet's score rises continuously toward its deadline the anti-starvation pressure that prevents indefinite delay of lower-priority packets.

D. Criticality Normalization

Each class i is assigned a scalar bias $c_i \in \mathbb{R}$, configured *a priori*. To make this bias homogeneous with the other factors in $[0, 1]$, we normalize it by the maximum bias across classes:

$$\tilde{c}_i = \frac{c_i}{\max_{j \in \mathcal{C}} c_j}. \quad (5)$$

This normalization maps the most critical class to 1 while preserving the relative magnitude of the lower classes, which keeps the criticality scale directly interpretable for the application designer and comparable with the observed-facts term. With the chosen biases $c = (10, 2, 1)$, we obtain $\tilde{c} = (1.00, 0.20, 0.10)$.

Table I summarizes the chosen parametrization. The deadlines D_i are aligned with the ISA-100.11a classification of industrial traffic [12], which defines the critical class below 1 s, periodic supervision around 5 s, and slow logging/monitoring at several seconds. The default weights are set to $(w_a, w_b) = (0.5, 0.5)$ to balance loss-rate reactivity and starvation avoidance.

Algorithm 1 CLA-Score packet selection at node n

Require: Local queue $\mathcal{Q}$ of capacity Q ; per-class $\{a_i, \tilde{c}_i, D_i, \beta_i\}$; weights $w_a + w_b = 1$

- 1: **function** SCORE(p, t)
- 2: $i \leftarrow \text{class}(p)$; $b_p \leftarrow \min(\frac{t - t_{\text{arr}}(p)}{D_i}, 1)$
- 3: **return** $w_a a_i + w_b b_p + \tilde{c}_i$
- 4: **end function**
- 5: **procedure** ONARRIVAL(p_{in} , class i)
- 6: **if** $|\mathcal{Q}| < Q$ **then**
- 7: enqueue p_{in} ; $a_i \leftarrow (1 - \beta_i) a_i$
- 8: **else**
- 9: $p_{\text{min}} \leftarrow \arg \min_{p \in \mathcal{Q}} \text{SCORE}(p, t_{\text{now}})$
- 10: **if** $\text{SCORE}(p_{\text{in}}, t_{\text{now}}) > \text{SCORE}(p_{\text{min}}, t_{\text{now}})$ **then**
- 11: evict p_{min} ; enqueue p_{in}
- 12: $j \leftarrow \text{class}(p_{\text{min}})$; $a_j \leftarrow (1 - \beta_j) a_j + \beta_j$
- 13: **else**
- 14: reject p_{in} ; $a_i \leftarrow (1 - \beta_i) a_i + \beta_i$
- 15: **end if**
- 16: **end if**
- 17: **end procedure**
- 18: **procedure** ONCELLOPEN
- 19: **return** $\arg \max_{p \in \mathcal{Q}} \text{SCORE}(p, t_{\text{now}})$
- 20: **end procedure**

E. Algorithm

Algorithm 1 is executed independently by each sensor node n , using only locally available information. It is triggered at two distinct events in a packet's lifecycle. On the *arrival* of a new packet, the node decides whether to admit it; if the queue is full, the lowest-score packet is evicted, and the per-class loss estimator a_i is updated via Eq. (3). At each *transmission opportunity*, the node recomputes the score of every queued packet and selects the highest-scoring one. Because the queuing age b_p depends on the current time, it is implicitly refreshed at every decision through the SCORE function; no separate update step is needed. The computational cost is dominated by score evaluation over the queue, i.e., $O(|\mathcal{Q}|)$ per decision; with the small queue sizes typical of constrained IIoT nodes, this cost remains negligible compared to radio operations.

V. IMPLEMENTATION AND EVALUATION**A. Implementation**

CLA-Score is implemented in Contiki-NG by modifying the queue management module (`tsch-queue.c`). The packet descriptor is augmented with two fields (class and enqueue timestamp), and the selection function is replaced by the computation of Equation (1). The additional memory cost is limited to three scalars per node (one estimator per class) and two bytes per queued packet.

B. Baseline Policies

We compare CLA-Score against two baselines representing the simplest class-aware queue management strategies. **FIFO-1Q** preserves the standard single-queue TSCH implementation

TABLE II
DIMENSIONS ACCOUNTED FOR BY EACH POLICY.

Dimension	FIFO-1Q	FIFO-3Q	CLA-Score
Class hierarchy	✗	✓	✓
Queuing age (dequeue)	✗	✗	✓
Queuing age (eviction)	✓	✗	✓
Loss-rate priority	✗	✗	✓
Starvation-free	✓	✗	✓

of Contiki-NG [1], and evicts the head packet only if it has missed its deadline. This policy handles stale packets but ignores both class criticality and the observed loss rate. **FIFO-3Q**, inspired by the multi-queue priority scheme of [6], partitions the queue into three sub-queues (one per class) with strict priority: no packet from a lower class is served as long as a critical packet is waiting. This policy respects the class hierarchy but introduces a starvation risk for lower classes under heavy alarm load, and ignores both the queuing age and the observed loss rate. Table II summarizes the dimensions accounted for by each policy.

C. Experimental Setup and Metrics

We evaluate the three policies on the SLICES-FR Lille testbed (formerly FIT IoT-LAB) [7] with 75 IoT-LAB M3 nodes (STM32F103REY, ARM Cortex-M3, 64 KB SRAM, 72 MHz) arranged in a single-hop star topology: one static collector and 74 sensor nodes, all within radio range. Each M3 node is equipped with several on-board sensors: a 3-axis accelerometer and magnetometer (LSM303DLHC), an ambient-light sensor (ISL29020), and a barometric pressure/temperature sensor (LPS331AP); the three traffic classes carry, respectively, accelerometer (class 1), light (class 2), and pressure/temperature (class 3) readings, so that each class maps to a distinct physical sensor on every node. Each node runs Contiki-NG [13] configured with IEEE 802.15.4e TSCH. The TSCH slotframe comprises 101 slots of 10 ms (slotframe duration 1.01 s) over 4 channels (hopping sequence TSCH_HOPPING_SEQUENCE_4_4), and the MAC layer performs at most 5 retransmission attempts per frame.

As mainline Contiki-NG provides no complete MSF, we implemented one per RFC 9033 [2] on the 6top sublayer. Standard MSF sizes negotiated cells from *occupancy* alone, adding or removing a cell when the used-cell count crosses a threshold. This is blind to deadline violations: a critical packet that expires in the queue is dropped *before* its cell opens, so it never raises the occupancy count, and the under-served flow keeps its allocation unchanged. We therefore add a *deadline-aware* trigger: in-queue deadline drops are counted, and a node that exceeds a small threshold issues a 6P ADD to acquire capacity exactly when critical traffic misses its deadlines, rather than waiting for average occupancy to rise. This trigger is part of the common scheduling layer and is applied identically under all three policies (FIFO-1Q, FIFO-3Q, and CLA-Score), so that the comparison isolates the effect of the queue selection policy alone.

a) *Queue parametrization*: The per-neighbor queue capacity is $Q = 8$ packets for FIFO-1Q and CLA-Score. For FIFO-3Q, each class has a separate circular buffer of $Q_k = 4$ packets (12 packets in total). This asymmetric allocation is imposed by the `ringbufindex` structure of Contiki-NG, which enforces power-of-two buffer sizes, and by the 64 KB SRAM budget of the M3 node combined with a neighbor table provisioned for 84 entries.

b) *CLA-Score parameters*: The score weights are fixed at $w_a = w_b = 0.5$. The criticality biases and EWMA smoothing factors follow the parametrization introduced in Section IV: $\tilde{c} = (1.00, 0.20, 0.10)$ and $(\beta_1, \beta_2, \beta_3) = (0.9, 0.5, 0.2)$.

c) *Experimental protocol*: We stress the policies under two load regimes by varying the Cls1 arrival rate over $\lambda_1 \in \{0.5, 1\}$ pkt/s. Each of the six scenarios (3 policies $\times$ 2 loads) is repeated over 5 independent runs of 60 minutes on the live SLICES-FR Lille testbed, and results are averaged across runs and sensors.

d) *Metric*: We report the *Queue Admission Rate* (QAR), defined as the fraction of packets generated by the application that are not dropped by the local queue policy before transmission:

$$\text{QAR}_i = 1 - \frac{N_{\text{deadline}}^{(i)} + N_{\text{full}}^{(i)} + N_{\text{rej}}^{(i)} + N_{\text{evict}}^{(i)}}{N_{\text{gen}}^{(i)}}, \quad (6)$$

where, for class i , $N_{\text{gen}}^{(i)}$ is the number of generated packets, and the numerator sums the four loss causes controlled by the queue policy: deadline expiration in queue, full-buffer drop, admission refusal (the REJECT branch in Algorithm 1, when an incoming packet's score is below the minimum-score packet already in queue), and eviction. QAR isolates the effect of the queue policy itself, independently of the downstream radio scheduling, which is held identical across policies. We complement QAR with additional metric: the loss decomposition by cause, which attributes each lost packet to one of the four causes above.

D. Results

a) *Traffic generation*: Each UDP frame carries an 11-byte payload comprising a class tag, a 32-bit sequence number, and the sensor sample, enabling exact per-packet accounting. Over all runs, the sensor nodes generated 6.49 million packets, distributed as 77.7% Cls1 (5.04 M), 19.0% Cls2 (1.23 M), and 3.3% Cls3 (0.21 M). In aggregate, this offered load reaches ≈ 47 pkt/s at $\lambda_1 = 0.5$ and ≈ 74 pkt/s at $\lambda_1 = 1$. The single-radio collector can receive at most one packet per 10 ms slot, i.e., a physical ceiling of ≈ 100 pkt/s (≈ 360 kpkt/h). The four channels do not raise this ceiling, since the sink listens on a single channel at a time. The two regimes therefore load the network to about 47% and 74% of its physical capacity, respectively. This alarm-dominated mix, together with the near-saturation $\lambda_1 = 1$ regime, is a deliberate stress condition that maximizes inter-class contention and exposes the differences between selection policies.

TABLE III
PER-CLASS QUEUE ADMISSION RATE (QAR) AND CLS1 LOSS BREAKDOWN, AS PERCENTAGES OF GENERATED PACKETS, POOLED OVER 5 RUNS OF 60 MIN (ONE FIFO-1Q RUN AT $\lambda_1 = 0.5$ PKT/S EXCLUDED). BEST QAR PER ROW IN BOLD. RUN-TO-RUN STANDARD DEVIATION IS $\leq 3.6\%$ ($\leq 0.9\%$ FOR CLA-SCORE ON CLS1).

λ_1	Class	QAR (%)		
		FIFO-1Q	FIFO-3Q	CLA-Score
0.5 pkt/s	Cls1	88.2	93.3	98.7
	Cls2	90.0	99.0	98.6
	Cls3	90.7	100.0	99.9
1.0 pkt/s	Cls1	88.7	92.8	97.0
	Cls2	90.3	96.2	91.9
	Cls3	91.1	96.2	91.9
Cls1 loss breakdown (% of generated Cls1)				
0.5 pkt/s	Deadline	2.6	6.6	1.3
	QFull	9.2	0.1	0.0
	Total	11.8	6.7	1.3
1.0 pkt/s	Deadline	1.9	6.7	3.0
	QFull	9.4	0.5	0.0
	Total	11.3	7.2	3.0

b) *Per-class queue admission rate*: Table III and Fig. 2 report the per-class QAR for the three policies under both load regimes. CLA-Score provides the strongest protection of the *critical* class (Cls1): it admits 98.7% of Cls1 at $\lambda_1 = 0.5$ pkt/s and 96.9% at $\lambda_1 = 1$ pkt/s, +10.7 and +8.4 percentage points (pp) over the class-blind FIFO-1Q, and +5.5 and +4.8 pp over FIFO-3Q. These critical-class gains exceed one standard deviation in every regime, so they are consistent across the five runs.

Relative to FIFO-1Q, CLA-Score matches or exceeds every class: at $\lambda_1 = 0.5$, it improves all three classes by +9 to +11 pp; at $\lambda_1 = 1$, the lower classes are statistically on par (Cls2 +1.6 pp, Cls3 +0.8 pp, both within run-to-run variance).

Relative to FIFO-3Q, the result is a deliberate trade-off. CLA-Score leads on the critical class in both regimes. On the non-critical classes, the two policies are essentially tied at $\lambda_1 = 0.5$ (−0.4 and −0.1 pp, within noise), and FIFO-3Q takes a clear lead only at high load (−4.0 pp on Cls2 and −4.3 pp on Cls3 at $\lambda_1 = 1$). This is the expected cost of criticality-driven prioritization: under saturation, CLA-Score reallocates admission capacity toward Cls1 at the expense of the lower classes.

FIFO-3Q's strict priority, in contrast, over-serves the lower classes it even tops Cls2 and Cls3 marginally at low loa (98.9% and $\approx 100\%$) yet never matches CLA-Score on the critical traffic. FIFO-1Q, lacking any class differentiation, yields a near-uniform 88–91% across all classes and is consistently the weakest option for the critical class.

c) *Loss decomposition*: Loss causes differ sharply by policy (Table III): FIFO-1Q loses critical packets mainly to buffer overflow (QFull, $\sim 9\%$ of Cls1), whereas FIFO-3Q and CLA-Score never overflow and lose almost only to deadline expiration. CLA-Score attains the lowest total Cls1 loss (1.3/3.0% vs 11.8/11.3% for FIFO-1Q and 6.7/7.2% for

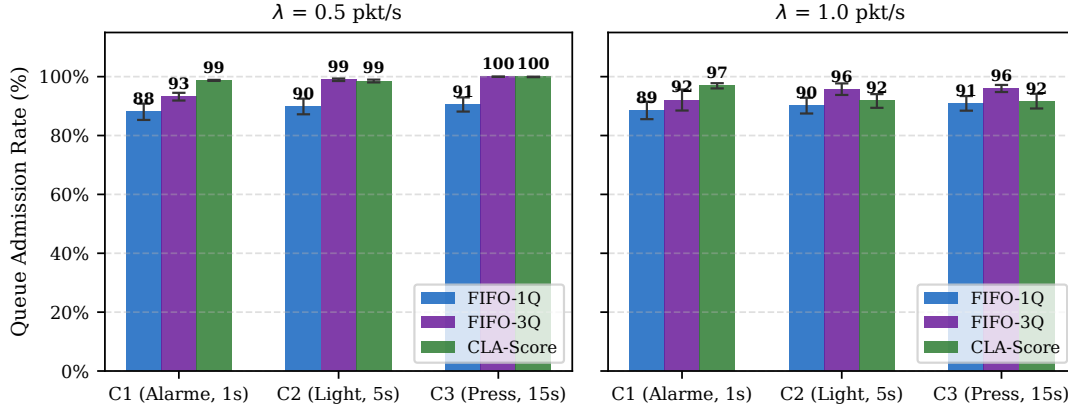


Fig. 2. Queue Admission Rate per class for the three policies under both load regimes. CLA-Score consistently delivers the highest QAR across all classes and loads, with the largest gain on the critical class (C1s1).

FIFO-3Q): its admission control suppresses overflow and its score serves urgent packets earlier. Eviction and admission refusal remain negligible ($\leq 0.9\%$, CLA-Score only). The bottleneck is thus *scheduling and admission*, not raw buffer capacity.

d) *Summary*: CLA-Score raises critical-alarm admission to 98.7%/97.0% (low/high load), i.e. +5 to +11 percentage points over the FIFO and strict-priority baselines, while matching or exceeding FIFO-1Q on the periodic and background classes; the only trade-off is a ~ 4 pp drop on these classes versus FIFO-3Q at high load. The loss decomposition attributes the critical-class gain to fewer in-queue deadline expirations and the absence of overflow, confirming that the multi-factor score reorders the queue toward the most urgent packets. The effect holds under both loads, with low run-to-run variance ($\sigma \leq 0.9\%$ on the critical class).

VI. CONCLUSION AND FUTURE WORK

We presented CLA-Score, a weighted packet selection policy for IEEE 802.15.4e TSCH networks under heterogeneous SCADA traffic. It combines three factors in a single per-packet score: class criticality, observed loss rate, and queuing age separating *observed facts* from *application belief* so that the nominal class order is deviated only when the runtime situation requires it. The mechanism is fully node-local, needs no signaling, and preserves the standard TSCH stack. On the SLICES-FR Lille testbed with 74 sensor nodes over 30 hours of live experiments, CLA-Score raises the queue admission rate of critical alarms to 98.7%/97.0% (low/high load) +4 to +11 percentage points over the FIFO and strict-priority baselines while matching or exceeding the class-blind FIFO-1Q on the periodic and background flows, with low run-to-run variance ($\sigma \leq 0.9\%$ on the critical class).

Future work: We plan to (i) calibrate the weights w_a, w_b and smoothing factors β_i online via lightweight reinforcement learning, (ii) extend the policy to the cell-allocation level by adapting the CSMA backoff to per-packet urgency, and (iii) evaluate on multi-hop industrial topologies.

REFERENCES

- [1] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomo, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 2017, pp. 11–18.
- [2] "RFC 9033: 6TiSCH Minimal Scheduling Function (MSF)," <https://dl.acm.org/doi/book/10.17487/RFC9033>, accessed: 2026-05-25.
- [3] A. R. Urke, Ø. Kure, and K. Øvsthus, "Autonomous Flow-Based TSCH Scheduling for Heterogeneous Traffic Patterns: Challenges, Design, Simulation, and Testbed Evaluation," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 2357–2372, 2023.
- [4] J.-H. Lee and S.-H. Chung, "Adaptive Cell Scheduling and Negotiation Techniques for 6TiSCH Networks Under Bursty Traffic," *Sensors*, vol. 25, no. 5, p. 1418, Jan. 2025.
- [5] F. F. Jurado-Lasso, C. Orfanidis, J. F. Jurado, and X. Fafoutis, "Hrltsch: A hierarchical reinforcement learning-based tsch scheduler for iiot," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 6, pp. 2102–2118, 2024.
- [6] H. Farag, S. Grimaldi, M. Gidlund, and P. Österberg, "REA-6TiSCH: Reliable Emergency-Aware Communication Scheme for 6TiSCH Networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1871–1882, Feb. 2021.
- [7] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "Fit iot-lab: A large scale open experimental iot testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 459–464.
- [8] P. Raghuvanshi, O. L. A. López, N. B. Mehta, H. Alves, and M. Latva-Aho, "Neural Network-Based Bandit: A Medium Access Control for the IIoT Alarm Scenario," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 7511–7524, 2024.
- [9] A. Tabouche, B. Djamaa, and M. R. Senouci, "Traffic-Aware Reliable Scheduling in TSCH Networks for Industry 4.0: A Systematic Mapping Review," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2834–2861, 2023.
- [10] L. Bommisetty and T. G. Venkatesh, "Contention-Based Proportional Fairness (CBPF) Transmission Scheme for Time Slotted Channel Hopping Networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1289–1300, Feb. 2023.
- [11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.
- [12] M. Raza, N. Aslam, H. Le-Minh, S. Hussain, Y. Cao, and N. M. Khan, "A critical analysis of research potential, challenges, and future directives in industrial wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 39–95, 2018.
- [13] "Contiki-NG," <https://iot-lab.github.io/docs/os/contiki-ng/>, accessed: 2026-05-26.