telecom

MDPI

Article

Adaptive Execution Timing for Offline First Quorum Coordination in LoRa Mesh Networks

Francis Kagai [1], Philip Branch [1,*], Jason But [1] and Rebecca Allen [2]

[1] Department of Electrical, Robotics and Biomedical Engineering, Swinburne University of Technology, Melbourne 3122, Australia; fkagai@swin.edu.au (F.K.); jbut@swin.edu.au (J.B.)
[2] Department of Physics and Astronomy, Swinburne University of Technology, Melbourne 3122, Australia; rebeccaallen@swin.edu.au
* Correspondence: pbranch@swin.edu.au

Abstract

Emergency, remote, and infrastructure-constrained environments cannot always rely on cellular networks or continuous Internet connectivity. LoRa offers an alternative for direct, long-range communication between low-power devices without depending on nearby cellular infrastructure, but its low data rate and variable transmission time create new coordination challenges. When multiple LoRa nodes must agree on information, fixed timing can either introduce unnecessary delay or cause missed deadlines as radio conditions change. This paper introduces a cross-layer controller that adapts mesh coordination timing to LoRa time-on-air at runtime. A MAPE-K loop maps estimated airtime into transmission-slot spacing $S(c)$ and deadlines $D(c)$, coupled with quorum or full-participation finalisation and a bounded single-retry mechanism. The framework was implemented on four SX1276 nodes at 915 MHz and evaluated over 2514 coordination rounds under crash and omission faults. Three-of-four quorum coordination achieved a median latency of 2.1 s, compared with 5.3 s for full participation. Overall, 88.6% of rounds met the first adaptive deadline, 11.4% required one retry, and none exhausted the retry budget or violated the evaluated safety invariants. A replay-derived fixed-SF12 schedule produced a median completion time of 11.3 s, compared with 3.4 s for adaptive execution. These results show that LoRa airtime can serve as a practical runtime signal for adaptive coordination on constrained mesh nodes.

Keywords: adaptive timing; autonomic computing; crash and omission faults; emergency network; LoRa mesh; MAPE-K; quorum-based coordination

Check for updates

Academic Editor: Thiago C. Jesus

Received: 21 July 2026
Revised: 10 September 2026
Accepted: 16 September 2026
Published: 22 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

When disasters, infrastructure outages, or remote deployments remove reliable backhaul, nodes must still coordinate decisions despite intermittent links, constrained airtime, and volatile participation. These constraints are acute in LoRa-based deployments, where long transmission times and variable spreading factors make coordination completion highly sensitive to how slot spacing and round deadlines are timed. Central schedulers are frequently unavailable in offline-first field settings, so timing control must emerge from local interaction among constrained nodes.

Quorum-based coordination provides a lightweight agreement mechanism that tolerates limited participation loss without heavyweight consensus infrastructure. In this paper, a vote denotes one bounded coordination round keyed by a unique identifier: nodes

transmit on scheduled LoRa transmission slots until either a quorum threshold or a full-participation rule is satisfied, after which local state is cleared. Many deployments still fix slot spacing and deadlines offline [1]. When links improve, fixed timing wastes latency margin; when airtime or participation worsens, the same schedules become brittle. LoRa optimisation has focused mainly on PHY/MAC efficiency (spreading-factor adaptation, collision reduction, and energy-aware scheduling) [2–5], with far less attention to continuous retuning of coordination execution timing based on live airtime.

This paper designs, implements, and evaluates an adaptive execution-timing framework for quorum-based coordination on constrained LoRa mesh nodes. Rather than treating timing as a fixed deployment parameter, each node runs a local Monitor–Analyse–Plan–Execute–Knowledge (MAPE-K) loop that derives slot spacing $S(c)$ and round deadlines $D(c)$ based on LoRa time-on-air estimates, guard margins, and observed participation. Figure 1 summarises the four-node LoRa mesh, the opportunistic Wi-Fi/UDP gateway path, and the embedded MAPE-K loop on each node. The design is motivated by the Emergency Buddy System (EBS), where low-bitrate voice, lightweight ledger anchoring, and mesh coordination share the same airtime budget [6–8]. Prior EBS work addressed voice transport and gateway integration; this paper addresses the separate execution-timing problem that arises when verification and agreement must remain bounded under changing airtime.

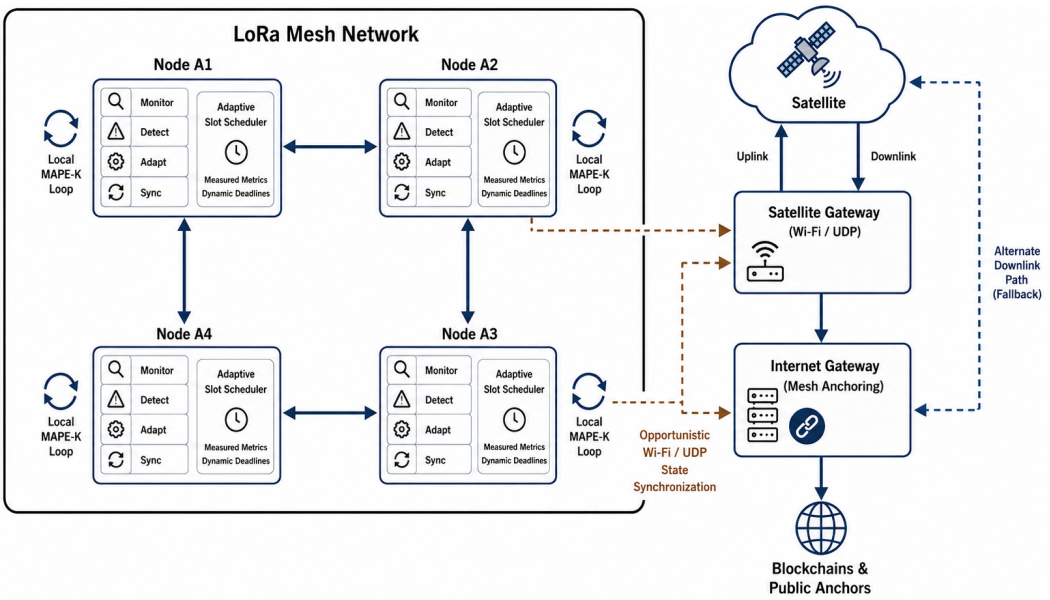


Figure 1. System architecture used throughout the evaluation: schematic of the four-node LoRa mesh, opportunistic Wi-Fi/UDP gateway path, and embedded MAPE-K loop on each node. Laboratory configuration used for instrumentation. Per-round slot spacing and deadlines are driven by measured airtime.

Scientific Contribution and Scope

It is useful to separate three levels of radio and coordination adaptation. Traditional LoRa adaptation modifies PHY or MAC variables such as spreading factor (SF), bandwidth (BW), coding rate (CR), and transmit power (P_t) to improve delivery, throughput, coverage, or energy [4,5,9–11]. The present framework instead takes the resulting PHY state $c = \langle SF, BW, CR \rangle$, derives time-on-air $\mathrm{ToA}(c)$, and uses that value to control the coordination layer:

$$\mathrm{ToA}(c) \rightarrow S(c) \rightarrow D(c) \rightarrow \text{quorum/full-participation execution.} \quad (1)$$

Therefore, the contribution is the cross-layer control relationship between observed PHY airtime and the execution timing of a distributed coordination process on constrained mesh nodes.

MAPE-K, adaptive radio scheduling, and quorum finalisation are established building blocks [12–14]. Existing approaches typically adapt PHY parameters for delivery, membership, or epoch-level timeouts independently; the present controller instead exposes PHY airtime as the runtime control signal for scheduled quorum execution. The MAPE-K loop sits inside the coordination execution path on each MCU and updates timing variables locally.

We implemented the framework on a four-node SX1276 LoRa mesh and evaluated 2514 coordination rounds under crash and omission faults with honest firmware. The results characterise latency, mode mix, and recovery on that testbed.

Our contributions are as follows.

(a) A runtime execution-timing framework that maps measured LoRa airtime (ToA(c)) into coordination scheduling variables $S(c)$ and $D(c)$ under constrained bandwidth.
(b) An embedded node-local MAPE-K loop that updates $S(c)$, $D(c)$, and finalisation policy from observed airtime and participation without adding coordination messages or changing agreement semantics.
(c) A bounded recovery architecture with a single active coordination instance, FIFO queue capacity $K = 10$, and at most one retry per round.
(d) Firmware implementation and laboratory validation on a four-node LoRa mesh across 2514 rounds under crash and omission faults with honest firmware.

The remainder of the paper is organised as follows. Section 2 reviews the related work. Section 3 defines the system model and timing framework. Section 4 presents the algorithms. Section 5 analyses complexity and overhead. Sections 6 and 7 describe the experimental setup and results. Section 8 discusses implications, and Section 9 outlines scalability and deployment considerations.

2. Background and Related Work

Autonomic computing and delay-tolerant networking (DTN) motivate embedded adaptation when operators are absent and connectivity is fragmentary [12,15–17], yet many autonomic deployments still assume edge–gateway paths absent in disconnected meshes.

Table 1 contrasts representative approaches along network model, reported scale, primary objective, timing adaptation, coordination support, and published latency, packet-delivery, and energy evidence. Inclusion criteria are: (i) relevance to intermittent or austere connectivity; (ii) support (or explicit absence) of distributed agreement under partitions; and (iii) device class compatible with constrained radios. Metrics are reported as published by the respective studies and are not interpreted as head-to-head benchmarks because payload size, topology, channel model, PHY configuration, and outcome definitions differ. Where a study does not report a metric under a comparable definition, the cell is marked N/R (not reported). DTN stacks emphasise store-and-forward delivery but rarely add a coordination layer for committed votes. IoT consensus treatments often keep phase timing fixed. Adaptive-consensus work typically varies who votes, when views change, or epoch-level timeouts [13,18–20], rather than per-round slot spacing from PHY airtime.

Quorum intersection and flexible voting sets are well studied in the Paxos family [14]. WAN-facing consensus separates site-local progress from wide-area latency [21]. Partial-synchrony models motivate configured timeouts relative to unknown but bounded delay [22]. The timing contribution here is orthogonal: fixed-threshold finalisation with continuous retuning of $S(c)$ and $D(c)$ from measured LoRa time-on-air.

Table 1. Comparative positioning of representative approaches. Latency, PDR, and energy cells report values only when the cited study publishes them under its own workload; N/R denotes not reported under a comparable definition. Metrics are not head-to-head benchmarks.

Approach	Network	Nodes	Primary Objective	Timing Adaptation	Coordination	Latency	PDR	Energy	Evidence
LoRa mesh routing [23]	LoRa mesh	field mesh	path selection/forwarding	MAC/network routing	none	N/R (coord.)	N/R (coord.)	N/R	laboratory/field
Adaptive LoRa MAC [10,11]	star/LPWAN	varies	delivery/duty cycle	SF, frame, duty cycle	none	N/R	N/R	N/R	simulation/analysis
Fine-grained LoRaWAN scheduling [5]	LoRaWAN-like	varies	collection reliability/energy	slot/SF schedule	none	N/R	N/R	N/R	simulation
BFT/IoT consensus [13,19,24]	LAN/WAN/IoT	varies	agreement	view/epoch timeout	quorum/BFT	N/R	N/R	N/R	simulation/laboratory
This work	LoRa mesh	4	quorum execution timing	ToA$\rightarrow S, D$	yes	2.1/5.3 s	N/R	N/R	laboratory

In Table 1, latency, PDR, and energy are filled only when a single published figure is available under a definition comparable enough to report without reinterpretation; otherwise the cell is N/R. For this work, coordination latency is reported directly; conventional PDR and joule-level energy were not measured, so those cells are N/R even though airtime is used elsewhere as a radio-activity proxy.

LoRa mesh routing and the proposed controller operate at different layers. Routing answers which path a packet should take; the timing controller answers how long the coordination process should wait and when it should finalise. A LoRa mesh routing protocol can transport the coordination messages used by the proposed framework [3,23,25]. The timing controller operates above routing and complements route discovery, forwarding, retransmission, and topology maintenance. DTN and LPWAN systems primarily address delivery [16,26,27]; the present work adds committed coordination timing on top of those delivery layers.

Recent LoRa work adapts spreading factor, power, and duty-cycle policy to improve throughput, coverage, or latency under imperfect SF orthogonality [4,9–11,28]. Cross-layer and fine-grained scheduling schemes (e.g., FREE) target reliable energy-efficient collection in LoRaWAN-like settings [5]. These results are complementary: they optimise delivery airtime and collisions, whereas this paper maps the resulting airtime into coordination slot spacing and deadlines for scheduled quorum rounds. Table 1 therefore positions these approaches by control objective and operating assumptions rather than as head-to-head experimental benchmarks. Quorum completion is also distinct from packet delivery ratio: a coordination round can succeed despite individual packet omissions, so PDR is reported as N/R for this work.

State-machine replication typically relies on view-based rounds and configured timeouts [13,18,19,21,22,24,29]. LoRa/LPWAN research studies scalability and duty-cycle limits with a focus on delivery rather than agreement on committed decisions [3,25–27,30]. DTN architectures deliver under partitions without mesh-level vote finalisation [16,17,31].

The contribution lies in the runtime integration of PHY-derived airtime with scheduled quorum execution and bounded recovery [14]. Evaluation uses a deployed laboratory prototype with measured airtime and embedded scheduling constraints (Sections 6 and 7).

Prior EBS publications established low-bitrate voice over LoRa and satellite–LoRa emergency infrastructure [6–8]. Offline-first Byzantine-fault-tolerant agreement with token-gated security in the same low-bitrate mesh setting is treated in [32]. The present work is the complementary scheduling and timing layer that retunes per-round $S(c)$ and $D(c)$ from measured airtime and evaluates dual-mode finalisation under crash and omission faults.

3. System Model

We model a low-power mesh in which each vote is one time-bounded coordination instance with explicit start, deadline, and cleanup so that intermittent links cannot grow unbounded local state. The following subsections fix topology, timing, and fault assumptions used later in the algorithms and evaluation.

3.1. Network, Timing, and Fault Management

Table 2 summarises notation. The architecture in Figure 1 is used throughout: an embedded management loop on each node and an opportunistic Wi-Fi/UDP path to a gateway, with the LoRa radio layer represented abstractly rather than as a geographic map.

Nodes are resource-constrained devices with a single application processor that may also serve best-effort gateway traffic; mesh timing is therefore modelled without assuming a dedicated quiet host. The graph, clock, and loss model below capture the quantities the

adaptive scheduler uses. Latency, loss, and contention on each link vary with environment and duty cycle.

Table 2. Notation summary.

Symbol	Definition
n	Number of participating nodes
f	Quorum-notation parameter (used to relate q to standard quorum expressions); this paper evaluates crash/omission only
q	Quorum threshold for quorum-mode finalisation; in our four-node experiments $q = 3$ (three-of-four)
n_{slots}	Transmission slots per vote round counted in $D(c)$ and D_{static}; in the four-node deployment $n_{\text{slots}} = n = 4$ (1)
$S(c)$	Slot spacing for LoRa config c
$D(c)$	Round deadline for config c
K	Queue capacity (fixed: 10)
r	Retry count (max: 1)
Δ_{clk}	Design-time bound on tolerable clock misalignment (configured target < 200 ms; not measured on the USB-instrumented testbed)
$\text{ToA}(c)$	LoRa time-on-air for the nominal vote payload under config c
$\text{ToA}_{\text{sig}}(c)$	LoRa time-on-air for compact signed control or status frames under c (used for slot interval Δ in Algorithm 2)
W_{perf}	Full-participation (4/4) wait window (3Δ)

The notation in Table 2 defines the quantities used to describe the architecture and execution model below. In particular, n, q, $S(c)$, $D(c)$, K, and Δ_{clk} map directly to participant membership, quorum formation, slot scheduling, deadline enforcement, queueing, and clock tolerance in Figure 1.

Let $G = (N, E)$ be a directed graph where each node $i \in N$ has a LoRa interface and each link $(i, j) \in E$ has delay $\delta_{ij}(t)$ and loss probability $p_{ij}(t)$ [2]. Nodes are half-duplex, use fixed transmit power P_t, and choose physical parameters $c = \langle BW, SF, CR \rangle$ (bandwidth, spreading factor, coding rate) [3,33].

Each node keeps a local clock. The timing model treats misalignment relative to a reference schedule as bounded by the configured parameter Δ_{clk} [34]. LoRa time-on-air $\text{ToA}(c)$ dominates packet duration; queueing and propagation add smaller but non-negligible jitter under contention. Let $M_{ij}(\tau)$ be the set of payloads successfully delivered from i to j within time τ.

Together, G, c, and Δ_{clk} supply the inputs on which $S(c)$, $D(c)$, and the MAPE loop operate; the next subsection fixes the round semantics built on top.

3.2. Coordination and Timing Framework

Coordination proceeds in discrete rounds, each initiated by a node acting as a temporary leader. A round comprises n_{slots} transmission slots determined by the channel-adaptive schedule $S(c)$ and bounded deadline $D(c)$:

$$S(c) = \lceil \text{ToA}(c) \rceil + t_{\text{sw}} + t_g, \tag{2}$$

$$D(c) = n_{\text{slots}} S(c) + t_{\text{lead}} + t_{\text{gw}} + t_{\text{margin}}. \tag{3}$$

For a homogeneous validator set of size n, we take $n_{\text{slots}} = n$ so each vote round assigns one deterministic LoRa transmission slot per participant before quorum aggregation.

Each node maintains a single active vote $v = \langle \text{id}, t_{\text{start}}, t_{\text{deadline}}, r \rangle$ with retry flag $r \in \{0, 1\}$. Votes that arrive while another is active are stored in a bounded FIFO queue Q

of capacity $K = 10$. This bounds queue occupancy and per-vote retries; worst-case waiting time for a queued vote depends on predecessor service times (Section 4.4).

The algorithmic design assumes a hybrid fault framework appropriate for intermittently connected LoRa mesh networks [35,36]. Nodes may experience crash faults, in which a node stops transmitting or fails silently, as well as omission faults caused by packet loss due to channel interference or buffer overflow. In addition, timing faults are considered: local clock drift can lead to slot misalignment within the configured Δ_{clk} envelope used in the analysis.

The coordination layer exposes a quorum decision interface and enforces deterministic timing bounds through $S(c)$, $D(c)$, bounded queueing, and a single retry. The fault model considers crash and omission behaviour on intermittently connected links; adversarial Byzantine behaviour is out of scope for the implementation level properties stated here and for the evaluation in Section 6.

A round is successful if at least q nodes agree on the same vote outcome within the deadline $D(c)$. If the quorum condition is not satisfied within the specified deadline, the protocol initiates a bounded retry mechanism, allowing at most one retry per vote.

3.3. Baseline Model: Static Slot Scheduling (SSS)

Three distinct datasets are used in the evaluation as shown in Table 3:

(A) Adaptive measured: 2514 hardware rounds under the MAPE-K controller (primary validation trace).
(B) Fixed SF10 executed: a separate hardware run under fixed SF10 on identical device class
(C) Static SF12 replay: counterfactual application of fixed worst-case SF12 timing to the recorded adaptive event sequence

Table 3. Summary of evaluation datasets used for adaptive versus static timing comparisons.

Dataset	Type	Role
Adaptive (n = 4)	measured hardware	primary latency, mode mix, retries, invariants
Fixed SF10	measured hardware	executed fixed-configuration reference
Static SF12 (SSS)	trace replay	timing-rule sensitivity on the adaptive event log

To compare against a conservative non-adaptive schedule without collecting a second corpus, we define static slot scheduling (SSS) that fixes the LoRa configuration to c_{SF12} (SF12 on 125 kHz with the same coding and payload assumptions as the firmware) and uses worst-case airtime for every round. Static spacing and deadline are

$$S_{\text{static}} = \lceil \text{ToA}(c_{\text{SF12}}) \rceil + t_{\text{sw}} + t_g, \tag{4}$$

$$D_{\text{static}} = n_{\text{slots}}\, S_{\text{static}} + t_{\text{lead}} + t_{\text{gw}} + t_{\text{margin}}. \tag{5}$$

Both the adaptive scheduler and SSS are grounded in the same time-on-air quantity $\text{ToA}(\cdot)$: the adaptive controller computes $(S(c), D(c))$ from $\text{ToA}(c)$ using the current radio configuration c, whereas SSS fixes c to c_{SF12} and therefore uses one conservative $\text{ToA}(c_{\text{SF12}})$ for all rounds. The key distinction is whether the timing rules track per-round PHY settings (adaptive, executed) or impose a fixed ToA schedule (static). Section 7.2 reports three complementary comparison points: (i) the measured adaptive validation trace, (ii) an executed fixed-configuration static baseline (SF10) measured on hardware, and (iii) a replay-derived fixed-SF12 worst-case timing baseline (SSS) applied counterfactually to the adaptive trace. Trace replay then recomputes each validation round's counterfactual completion time by substituting $(S(c), D(c))$ with $(S_{\text{static}}, D_{\text{static}})$ while preserving the logged initiation order,

per-slot transmission attempts, and finalisation events (quorum formation and retries). Medians and tail latencies are replay-derived; the measured adaptive row is taken directly from the same 2514-round trace. This baseline is a counterfactual timing comparison that holds the observed event sequence fixed while changing only the slot/deadline rules; it does not, by itself, constitute a second long-run executed with static SF12 timing on the nodes.

Trace replay quantifies timing-rule sensitivity on an identical logged event sequence, and it does not model closed-loop feedback that would arise under an actually executed static policy. In particular, (i) longer slot spacing can change collision/loss patterns, (ii) different deadlines can change when retries occur and thus alter subsequent transmissions, and (iii) queue dynamics can change when earlier rounds take longer and overlap with application and gateway activity. For these reasons, we use replay to characterise timing-policy sensitivity and a conservative upper-bound schedule under fixed events, while treating executed baselines as the source of measured static-configuration behaviour.

A companion study evaluates executed fixed configuration behaviour of the same four-node LoRa mesh stack under a fixed high-reliability PHY setting (SF10 with 120-byte payloads), including measured quorum and full-participation latencies, success rate, and tail latency. The SF12 SSS replay here answers a complementary question for the MAPE validation corpus: how pessimistic completion times become if every slot in the same 2514-round log assumes worst-case SF12 airtime. The two baselines are not interchangeable—they answer different questions under different PHY assumptions—but together they separate (i) measured fixed-configuration operation from (ii) counterfactual worst-case static timing on the adaptive event log. Executing a long-run static-SF12 schedule on hardware for the same workload remains a useful extension.

3.4. Problem Definition and Design Rationale

We seek a self-managing mesh coordination layer with: (i) safety—at most one committed outcome per vote id (conflicting outcomes are not committed); (ii) conditional liveness under crash/omission within the tested quorum model (a valid round eventually completes when enough participants remain reachable); and (iii) low-latency L and bounded waiting under the queue and retry limits of Section 4.4 on duty-cycled, memory-limited devices. Availability denotes whether enough participants are reachable to form the quorum. Central orchestration is ruled out by the deployment model.

Three constraints drive the design. Delays swing with SF, interference, and MCU load, so each node must sense and retune locally. A capacity-K queue, explicit $D(c)$, and $r \leq 1$ retries cap state and avoid retransmission storms when several nodes recover together. The MAPE loop exports measurements directly into the slot planner, rather than through a separate management plane that might itself be unreachable.

Concurrent active rounds are deliberately prohibited. Allowing concurrency would require per-round slot allocation, concurrent quorum state, duplicate suppression across active identifiers, multiple timeout contexts, additional RAM, arbitration between radio transmissions, and a substantially more complicated safety analysis. Most importantly, SX1276 communication is half-duplex and the physical channel remains serialised even if logical votes execute concurrently. The architecture therefore permits only one active vote and queues the rest. Serialisation is a design choice that trades throughput for bounded memory, deterministic radio scheduling, and simpler safety semantics. The design can introduce head-of-line blocking during bursts and is consequently intended for low-rate coordination.

Section 4 formalises these constraints as executable algorithms.

4. Algorithms

Notation used below is summarised in Table 2. Algorithm 1 is the vote engine: one active vote, FIFO queue of capacity K, and at most one retry. Algorithm 2 is the MAPE-K controller that samples radio and node metrics on a fixed interval and updates the timing parameters consumed by Algorithm 1. The subsections present cleanup rules, safety and liveness properties (Section 4.4), and the autonomic controller.

Algorithm 1 Channel-adaptive coordination with queue and single retry. Only `OnQuorumReached` commits via `FinaliseVote`; deadline expiry aborts an attempt or the vote without committing

Require: $s \in \{\perp, v\}$, FIFO queue Q of capacity K
Require: $c, n_{\text{slots}}, t_{\text{sw}}, t_g, t_{\text{lead}}, t_{\text{gw}}, t_{\text{margin}}$

Event `OnReceiveVoteInit`(x):
if $s = \perp$ **then**
  $v \leftarrow \langle x, t_{\text{now}}, t_{\text{now}} + D(c), 0\rangle$
  `StartVote`(v); $s \leftarrow v$
else if $x = s.\text{id}$ **then**
  `MergeFragments`(s)
else if $|Q| < K$ **then**
  `Enqueue`$(Q, \langle x, t_{\text{now}}, 0, 0\rangle)$
else
  `SendBusy`(x) {Send NACK, sender backs off}
end if

Event `OnQuorumReached`$()$:
`FinaliseVote()`; `CleanupVote()`; $s \leftarrow \perp$
if $Q \neq \emptyset$ **then**
  $u \leftarrow$ `Dequeue`(Q)
  $u.t_{\text{start}} \leftarrow t_{\text{now}}$; $u.t_{\text{deadline}} \leftarrow t_{\text{now}} + D(c)$
  `StartVote`(u); $s \leftarrow u$
end if

Event `OnDeadlineTick`$()$:
if $s \neq \perp$ **and** $t_{\text{now}} \geq s.t_{\text{deadline}}$ **then**
  if $s.r = 0$ **then**
    `AbortAttempt()` {no commit; close failed attempt}
    $s.r \leftarrow 1$
    $s.t_{\text{start}} \leftarrow t_{\text{now}}$; $s.t_{\text{deadline}} \leftarrow t_{\text{now}} + D(c)$
    `RescheduleSlots`$(S(c))$
  else
    `AbortVote()`; `CleanupVote()`; $s \leftarrow \perp$ {no commit; drop after retry}
    if $Q \neq \emptyset$ **then**
      $u \leftarrow$ `Dequeue`(Q); $u.t_{\text{start}} \leftarrow t_{\text{now}}$; $u.t_{\text{deadline}} \leftarrow t_{\text{now}} + D(c)$
      `StartVote`(u); $s \leftarrow u$
    end if
  end if
end if

4.1. Deterministic Cleanup with Bounded Queue and Single Retry

To ensure predictable recovery under intermittent links, the coordination engine maintains at most one active vote at a time. Incoming votes are queued in a bounded FIFO buffer of capacity K. Each vote $v = \langle \text{id}, t_{\text{start}}, t_{\text{deadline}}, r\rangle$ carries a retry flag $r \in \{0, 1\}$. A vote may be retried once using updated timing derived from the channel configuration c.

Algorithm 2 Autonomic controller for offline-first quorum coordination

Require: Local radio metrics; deterministic ToA model; payload byte lengths `pktBytes` and `sigPktBytes`; leader view-controller; gateway optional
Ensure: Self-monitor, self-adapt, self-heal under intermittent links

State: link metrics $\mathcal{M}$, slot base Δ_0, slot interval Δ, view v, window W_{perf}

Procedure `Monitor`():
  $\mathcal{M} \leftarrow$ SAMPLE(`RSSI,SF,BW,CR,queueLoad,latency`)
  `ToA` $\leftarrow$ SEMTECHTOA(`SF,BW,CR,pktBytes`)
  `sigToA` $\leftarrow$ SEMTECHTOA(`SF,BW,CR,sigPktBytes`)

Procedure `Analyze`():
  $\Delta_0 \leftarrow$ `ToA` + `guard` + `switch` + `proc`
  $\Delta \leftarrow$ `sigToA` + `guard` + `switch`
  $W_{\text{perf}} \leftarrow 3\Delta$ {wait window for 4/4}

Procedure `Plan`(v):
  ASSIGNDETERMINISTICSLOTS(Δ_0, Δ)
  if QUORUMREACHED() **and** $\neg$LEADERFROZEN() **then**
    FREEZESIGNERS(); ELECTLEADERDET(); $v \leftarrow 0$
  end if
  if LEADERTIMEOUT(v) **then**
    $v \leftarrow v + 1$ {view change for failover}
  end if

Procedure `Execute`():
  SCHEDULETX(); RUNSCHEDULEDTX()
  if PERFECT4OF4WITHIN(W_{perf}) **then**
    FINALISEPERFECT()
  else if WINDOWEXPIRED() **and** QUORUMREACHED() **then**
    FINALISEQUORUM()
  end if
  if ISLEADER() **then**
    SENDTOGATEWAYORDEFER()
  end if
  BROADCASTCONSENSUSSTATUS()
  ENFORCEDEADLINECLEANUP()

while True **do**
  MONITOR(); ANALYZE(); PLAN(v); EXECUTE()
end while

The slot spacing and deadline are computed from the LoRa time-on-air parameters (closed-form airtime from the PHY parameterisation of the deployed transceiver family [3,37]):

$$t_{\text{sym}}(BW, SF) = \frac{2^{SF}}{BW}, \tag{6}$$

$$S(c) = \lceil \text{ToA}(c) \rceil + t_{\text{sw}} + t_g, \tag{7}$$

$$D(c) = n_{\text{slots}} S(c) + t_{\text{lead}} + t_{\text{gw}} + t_{\text{margin}}. \tag{8}$$

4.1.1. Queue Design Rationale

FIFO was selected because it preserves arrival ordering, supports $O(1)$ insertion and removal, requires no priority metadata, behaves deterministically, and is appropriate where every vote has the same service semantics. It also avoids starvation introduced by priority scheduling. Queue capacity $K = 10$ is an implementation parameter selected for the evaluated MCU workload. Memory for the queue scales as $M_Q \propto KM_v$, where M_v is the

per-vote state footprint. Each queued predecessor is a full coordination round that may consume up to two deadline windows (first attempt plus one retry), so queue waiting is service-time dependent. Increasing K therefore improves burst tolerance but increases memory and worst-case waiting under backlog.

The queue provides bounded buffering with capacity $K = 10$. Safety is maintained by enforcing a single active vote and limiting the retry count to at most one ($\leq$1). Timing functions $S(c)$ and $D(c)$ adapt automatically with LoRa bandwidth and spreading factor, thereby stabilising slot allocation under channel variation.

When queue capacity is exceeded, the node issues a `SendBusy` notification (NACK) to the vote initiator to signal overload; the initiator then applies exponential backoff (initial delay 500 ms, maximum 4 s) before retrying, which prevents vote loss without unbounded congestion.

4.1.2. Relation to TCP-Style Exponential Backoff

TCP-style exponential backoff reacts mainly to inferred congestion or failed delivery by increasing retransmission intervals. The primary timing adaptation in this work is different: it proactively derives the schedule from ToA(c) before retransmission is necessary. The two mechanisms therefore solve different problems. The adaptive scheduler maps PHY state to airtime and then to slot spacing and deadlines (ToA$(c) \rightarrow S(c) \rightarrow D(c)$). Backoff maps overload or failure to a longer wait before retry. Both are used: $S(c)$ and $D(c)$ constitute the main adaptive timing mechanism, while exponential backoff after `SendBusy` is secondary overload control.

4.2. Correctness and Bounded Progress

Correctness is stated using three distinct terms. Safety: at most one committed outcome is associated with each vote identifier. Liveness: a valid round eventually completes under stated reachability assumptions. Availability: enough participants are reachable to form the quorum.

Safety under omission: Packet loss removes messages from the delivered sets $M_{ij}(\tau)$; under the stated authentication and quorum-intersection assumptions, omission alone cannot fabricate a second authenticated decision. Consequently, for any packet-loss process, omission alone does not violate uniqueness of a committed value.

Conditional bounded liveness: Liveness cannot be guaranteed under arbitrary permanent packet loss. If fewer than q participants remain mutually reachable within $D(c)$, quorum cannot form. Bounded progress therefore requires eventual delivery from at least q eligible participants within the configured deadline or retry envelope.

Safety is enforced through a set of invariants that constrain node behaviour throughout the coordination process. At most one active vote may exist within a round at any given time, and vote cleanup must complete before the initiation of any subsequent vote. Each vote is retried at most once, which bounds fault recovery and prevents unbounded retransmission, while FIFO ordering of queued votes preserves the arrival sequence across distinct identifiers.

Together, these invariants prevent two quorums from deciding conflicting outcomes for the same vote identifier. Because any retry reuses the same identifier and deadline structure, stale or duplicate state cannot be committed after cleanup. For equal-sized quorums over a membership of n, intersection requires $2q > n$; in the evaluated four-node deployment, $2(3) > 4$, so any two three-of-four quorums intersect. Under authenticated messages and the exclusion of Byzantine equivocation, that intersection condition supports the implementation-aligned uniqueness argument used here.

The algorithm design ensures liveness when at least a quorum of nodes remains reachable within bounded delay $D(c)$. If temporary disruptions prevent quorum formation, pending votes are held in the queue Q and retried after timeout. The bounded queue capacity K prevents indefinite growth of uncommitted local state and bounds the number of predecessor rounds that an accepted queued vote can encounter.

Formally, let T_{wait} be the waiting time of a newly enqueued vote behind at most K predecessors, and let $T_{\text{completion}}$ be the time from enqueue (or start) until that vote either commits, exhausts its single retry, or is aborted. Under Algorithm 1, each active vote may occupy at most two deadline windows ($r \leq 1$). A conservative implementation-aligned bound is therefore

$$T_{\text{wait}} \leq 2K\,D(c), \qquad T_{\text{completion}} \leq 2(K+1)\,D(c),$$

neglecting local processing and the secondary `SendBusy` backoff, which are outside the LoRa schedule model. These bounds assume every predecessor consumes its full two-attempt envelope; they are deterministic upper bounds under that assumption, not probabilistic expectations. The design therefore claims bounded queue occupancy, bounded retries, and service-time-dependent waiting under the stated attempt limits.

4.3. Assumptions and Decision Semantics

To avoid over-interpreting the implementation-aligned properties below, we make explicit the assumptions and the meaning of finalisation used in this paper.

Each coordination instance is keyed by a unique vote identifier id and targets a single outcome value associated with that identifier (e.g., a hash-committed payload). A node invokes `FinaliseVote()` only when either (i) a quorum of q authenticated participant messages is accumulated (quorum mode, three-of-four in the evaluation), or (ii) all n participants contribute within the bounded wait window W_{perf} (full-participation mode). Deadline expiry instead invokes `AbortAttempt()` (first miss) or `AbortVote()` (after the single retry); those paths close local attempt state without committing an outcome. Once finalised, `CleanupVote()` removes local state for that identifier and prevents later messages for the same id from being committed after cleanup.

The arguments in Section 4.4 are stated for the evaluated crash/omission setting with honest firmware, where (i) messages are authenticated and vote identifiers are not forged; (ii) non-faulty nodes follow the specified queue, deadline, and retry rules ($K = 10$, $r \leq 1$); (iii) participants share a consistent membership list for the round; and (iv) equal-sized quorums satisfy the intersection condition $2q > n$ (here $2{\cdot}3 > 4$) [14]. Byzantine equivocation, replay attacks, and adversarial scheduling are outside the experimental scope of this paper.

4.4. Implementation Invariants and Recovery Bounds

We state compact properties that mirror the implemented invariants and bounds under the evaluated crash and omission model.

Proposition 1. *Safety under omission, per vote id: At most, one committed value is associated with each vote identifier.*

Invariant I1 enforces a single active vote; retries reuse the same identifier and deadline envelope; FIFO queueing with cleanup before the next activation prevents overlapping decisions for the same id. Under authenticated messages and quorum intersection assumptions, two conflicting quorums for the same id cannot both complete without violating those invariants. Omission of messages from $M_{ij}(\tau)$ cannot by itself create a conflicting authenticated vote. This is an implementation-aligned argument for the stated assumptions.

Proposition 2. *Conditional bounded liveness: If at least q nodes are reachable within deadline $D(c)$ for a vote, then that vote completes or exhausts its single retry within a service-time-dependent envelope bounded by*

$$T_{\text{completion}} \leq 2(K+1)\, D(c)$$

when, at most, K predecessors occupy the FIFO and each predecessor may consume two deadline windows. A vote either commits through `OnQuorumReached`*, waits one bounded retry after* `AbortAttempt`*, or is removed by* `AbortVote` *after the retry is exhausted; queue waiting is therefore driven by full-round service times. If fewer than q participants remain reachable within the deadline or retry envelope, progress is not claimed.*

The system therefore provides bounded, implementation aligned properties. Empirical checks in Section 7 show the invariants held on all 2514 validation rounds under crash and omission faults.

4.5. Autonomic Controller for Mesh Coordination

Algorithm 2 implements the autonomic controller for EBSC in intermittently connected networks. The controller follows the MAPE-K paradigm, integrating monitoring, adaptation, and recovery directly into the coordination execution path without reliance on centralised coordination.

Table 4 summarises what each MAPE-K stage changes. Unlike conventional adaptive LoRa MACs that retune SF, BW, CR, or power for delivery [10,11], the loop here consumes the resulting PHY state and retunes coordination timing variables $S(c)$, $D(c)$, and W_{perf}.

Table 4. MAPE-K stages in the embedded coordination controller.

Stage	Role in This Implementation
Monitor	Sample RSSI, SF, BW, CR, queue load, and latency
Analyse	Compute ToA, Δ_0, Δ, and W_{perf}
Plan	Assign transmission slots; evaluate quorum; handle leader timeout
Execute	Schedule transmissions; finalise; send gateway summary; cleanup
Knowledge	Retain protocol state, recent measurements, and timing parameters

In the monitoring phase, each node continuously observes radio and system-level metrics, including RSSI, spreading factor, bandwidth, coding rate, queue occupancy, and observed latency. Rather than relying on empirical transmission delays, the controller derives deterministic airtime estimates from the closed-form LoRa time-on-air expression in the radio parameters [3,37], allowing scheduling decisions to reflect real hardware behaviour under strict duty-cycle constraints.

The analysis and planning phases translate these observations into bounded timing and coordination decisions. The base slot Δ_0 aggregates the vote-payload airtime $\text{ToA}(c)$ with guard, switching, and processing delays, while the inter-slot cadence Δ uses a second Semtech airtime variable `sigToA`, which implements $\text{ToA}_{\text{sig}}(c)$ for the shorter signed control frames on the mesh. A bounded performance window specifies the tolerance for achieving full agreement before falling back to quorum-based execution. Deterministic slot assignments are generated, and view state is managed to support leader continuity and failover. When the current leader becomes unresponsive, a view-change operation advances the view index and enables deterministic leader rotation.

Execution enforces the resulting schedule by transmitting votes and verification messages in assigned slots. Full agreement triggers immediate finalisation, while partial but sufficient quorum completion finalises under quorum mode. Finalised summaries

are forwarded opportunistically to gateways when available, and each node performs deterministic cleanup to bound local state growth.

The knowledge component of the MAPE-K loop maintains protocol state, timing bounds, and recent execution outcomes, enabling each node to reason about progress and recovery locally. By embedding deterministic timing control, leader rotation, and bounded recovery actions within this autonomic cycle, the system achieves predictable coordination on low-bitrate, delay-tolerant links with quorum-mode and full-participation finalisation as discussed in Section 3.

5. Complexity and Overhead

Per round, monitoring and analysis are $O(1)$ in the number of scalar metrics sampled; planning and signature verification are $O(n)$ per node for n participants as shown in Table 5. Aggregated over the mesh, signature checks imply $O(n^2)$ local work, whereas protocols such as PBFT [13] additionally incur $O(n^2)$ messages across phases; here, the LoRa schedule keeps messages linear in n per round while verification cost remains the dominant quadratic term but parallelises per node.

Memory is $O(n)$ for participant state plus $O(nK)$ for signatures buffered with at most $K = 10$ queued votes, so queue RAM is bounded by construction. The MAPE-K loop runs on the same microcontroller as coordination; on the validation build it executed at a 50 ms interval and contributed under 2% CPU utilisation on the Arduino Uno R4 while sharing the core with LoRa and Wi-Fi tasks (Section 6). Adaptation adds no extra on-air control-plane messages beyond the scheduled vote and status frames already required by the coordination path, so communication overhead remains $O(n)$ transmissions per round. Energy consumption in joules was not power-instrumented. Time-on-air is therefore reported as a radio-activity proxy: adaptive timing reduces unused waiting relative to worst-case SF12 provisioning and is expected to reduce airtime-related energy expenditure under that proxy.

Table 5. Asymptotic resource scaling for one scheduled transmission opportunity per participant per round.

Resource	Scaling
Scheduled transmissions	$O(n)$
Deadline $D(c)$	$O(nS(c))$
Per-node verification	$O(n)$
Mesh-wide verification	$O(n^2)$
Participant state	$O(n)$
Queue-related state	$O(nK)$

6. Experimental Evaluation

We evaluate the firmware on a four-node LoRa mesh with opportunistic gateway backhaul, focusing on latency, mode mix, timeouts, and invariant checks under a controlled RF setting. Interpretation limits are collected in Section 6.2.

6.1. Experiment Design

The firmware evaluated here implements EBSC as the offline-first quorum coordination layer within EBS. Evaluation covers scheduled votes, quorum or full-participation finalisation, and bounded retries under crash and omission faults with honest firmware. As shown in Table 6, a four-node LoRa mesh was built from Arduino Uno R4 WiFi boards (nodes A3–A6), each hosting a Semtech SX1276 LoRa transceiver on a 915 MHz ISM band module (125 kHz and 250 kHz channel options, spreading factors SF7–SF12, coding rate 4/5 in validation logs) and sharing a single-core firmware image that multiplexes LoRa,

Wi-Fi, and UDP tasks. An opportunistic Raspberry Pi 4 gateway provided best-effort Wi-Fi/UDP backhaul on port 20240; it was not required for safety or liveness. Every node ran one firmware build (Arduino IDE 2.3.3) combining Algorithms 1 and 2; the MAPE loop updated Δ_0, Δ, and W_{perf} in the same task set as coordination using (2) and (3) with $n_{\text{slots}} = 4$. Gateway stalls of up to 10 s were observed after Wi-Fi drops during development, confirming that mesh timing cannot assume a quiet host.

The SF12 static comparison is a trace replay applied to the adaptive event log (Table 3). The measured SF10 baseline reports 100% success, 2.7 s quorum latency, 5.2 s full-participation latency, and P99 6.6 s under its fixed PHY workload (Table 8).

For experimental observability, all nodes were connected via USB serial links (115,200 baud) to a central logging workstation running Windows 10. This configuration provides synchronised, high resolution event traces across nodes and supports the timing and invariant analyses reported in Section 7.

Communication followed a hybrid model combining WiFi UDP and LoRa mesh propagation. Nodes attempted UDP communication with the gateway when connectivity was available and fell back to LoRa-based forwarding when disconnected or partitioned. The gateway listened on UDP port 20240 for finalised blocks and disseminated configuration updates when reachable. In mesh-only operation, configuration commands and initiation signals propagated via a hop-limited store-and-forward mechanism designed to prevent flooding.

Table 6. Experimental and baseline configuration used in the evaluation.

Parameter	Adaptive Evaluation	SF10 Executed Baseline
Nodes	4	4
MCU	Arduino Uno R4 WiFi	same
Radio	SX1276	same
Band	915 MHz ISM	same
Bandwidth	125/250 kHz	fixed with SF10 run
Spreading factor	SF7–SF12	SF10
Coding rate	4/5	4/5
Payload (baseline)	as logged	120 B
Gateway	Raspberry Pi 4	same class
Firmware environment	Arduino IDE 2.3.3	same stack family
Logger host	Windows 10	same class
Serial logging	115,200 baud	same class
MAPE interval	50 ms	N/A (fixed PHY)
Queue capacity K	10	10
Retries	$\leq$1	$\leq$1
t_{sw}	50 ms	same margin family
t_g	100 ms	same margin family
t_{lead}	200 ms	same margin family
Execution type	measured hardware	measured hardware

Autonomic fault management was exercised through a distributed MAPE-K loop running locally on each node at a fixed 50 ms interval. On the validation firmware build, that control interval added under 2% CPU utilisation on the Arduino Uno R4 while sharing the core with LoRa and Wi-Fi tasks. To minimise control overhead, nodes exchanged only essential information, including outcomes, participant counts, and critical timing measurements. Gateway reachability was determined through adaptive probing, with nodes operating fully autonomously when the gateway was unavailable. This design ensured that adaptation and recovery behaviour were evaluated under both connected and partitioned conditions without reliance on centralised coordination.

6.2. Limitations of Experimental Validation

The evaluation uses a four-node testbed with high-rate USB serial instrumentation and co-located LoRa radios in a controlled indoor laboratory. This setting supports detailed timing and invariant checking, and it fixes the scope of what is measured.

- Clocking and synchronisation. USB serial logging introduces implicit clock alignment through the host; independent untethered oscillator drift is not measured.
- RF environment. The controlled indoor laboratory does not capture outdoor multipath, strong external interference, mobility, or geographic dispersion.
- Scale. Topology size $n = 4$ does not characterise denser meshes or large validator sets; Section 9 provides analytical scaling arguments only.
- Fault model. Experiments exercise crash and omission faults with honest firmware. Byzantine equivocation, adversarial scheduling, and intentional jamming are out of scope.
- Baselines. The executed static reference is a fixed-SF10 hardware run. The SF12 static comparison is a replay counterfactual on the adaptive event log, not a long-run executed SF12 campaign on identical events.
- Energy. Energy consumption in joules was not measured. Airtime is observed as a radio-activity proxy.
- External systems. Related-work positioning is by objective and assumptions (Section 2); third-party adaptive LoRaWAN and consensus stacks were not reimplemented on this testbed.

Accordingly, experimental conclusions are limited to the four-node testbed, crash and omission faults, timing and recovery behaviour, and the evaluated hardware configuration. Larger-scale performance, joule-level energy behaviour, intentional jamming, and direct comparisons with third-party LoRa mesh implementations remain outside the present validation.

Within this defined setting, the paper establishes the behaviour of the adaptive timing controller and bounded recovery mechanisms on real embedded hardware.

6.3. Experiment Execution

Rounds were initiated via UDP commands to nodes with gateway connectivity, while mesh-only nodes received initiation messages through LoRa propagation. Each node executed deterministic slot scheduling and single-retry logic as specified in Algorithm 1, maintaining autonomous operation regardless of gateway availability. To minimise network load, only essential coordination metadata was transmitted during execution, including vote identifiers, outcomes (full-participation or quorum), and participant counts. LoRa configuration updates were disseminated through the mesh to ensure consistency even under partial connectivity.

Following an initial optimisation and stabilisation phase, the validation study comprised 2514 coordination rounds executed on the controlled testbed. The evaluation covered natural link variability, rounds with varying numbers of participating nodes (three to four), and execution both with and without active gateway backhaul to assess fully mesh-autonomous operation. Multiple LoRa configurations were tested to examine the impact of channel parameters on latency, including spreading factors from $SF = 7$ to $SF = 12$ and bandwidths of 125 kHz and 250 kHz.

Operational margins were selected after laboratory experimentation on the target hardware rather than from analytic defaults alone. Extended bring-up runs were used to separate Wi-Fi/LoRa contention effects, validate queue and single-retry cleanup under fault injection, and settle the timing margins $t_{\mathrm{sw}} = 50\,\mathrm{ms}$, $t_g = 100\,\mathrm{ms}$, and $t_{\mathrm{lead}} = 200\,\mathrm{ms}$ together with $K = 10$ and $r \leq 1$, chosen to accommodate SX1276 airtime, half-duplex

switching, and single-core scheduling on the Uno R4 while keeping deadlines tight enough to limit queue buildup. The reported validation results use one frozen firmware revision at these parameters; Section 7 analyses the 2514-round post-tuning window. First-deadline satisfaction ρ and latency statistics in that window characterise steady-state behaviour of the selected configuration, not exploratory development logs.

6.4. Runtime Safety Invariant Checks

The firmware asserts invariants I1–I4 on every round during validation. Table 7 records violations observed over the 2514-round window; the checks provide implementation evidence complementary to the informal arguments in Section 4.4.

No violations were observed. Together with the static bounds on K and r, this supports the claim that the deployed binary matches the intended safety shell under the exercised fault model.

Table 7. Runtime enforcement of safety invariants across 2514 coordination rounds.

Invariant	Violations	Max Observed	Status
I1: Single active vote	0	1 vote	Verified
I2: Cleanup before next	0	–	Verified
I3: Single retry max	0	1 retry	Verified
I4: FIFO ordering	0	–	Verified

6.5. Outcome Measures

Table 8 defines the metrics captured during experiments and used to evaluate coordination performance in Section 7 while Table 9 presents the outcome measures used in the validation analysis. Metrics are extracted from instrumented firmware logs: the coordination engine (Algorithm 1) and MAPE loop (Algorithm 2) share one runtime image, so samples reflect single-core scheduling, LoRa variability, and intermittent WiFi.

Table 8. Executed fixed-configuration baseline (SF10) measured on identical hardware (separate run, not replay-aligned with the 2514-round adaptive validation trace). Success rate applies only to the SF10 workload.

Metric	Value	Configuration
Success rate	100%	SF10, four nodes
Three-of-four quorum latency	2.7 s	SF10, CR 4/5
Four-of-four latency	5.2 s	SF10, CR 4/5
End-to-end latency (P99)	6.6 s	SF10, 120 B payloads

Table 9. Outcome measures used in the validation analysis.

Measure	Symbol	Definition
Coordination latency	L	Time from vote initiation to finalisation
First-deadline satisfaction rate	ρ	Fraction of rounds that satisfy the first adaptive deadline $D(c)$ without invoking the single-retry path

A multi-threaded serial logger records 18 timing channels from all four nodes while matching vote events online; SF7–SF12 and 125/250 kHz settings are swept as described above. Section 7 reports the 2514-round post-tuning validation window; development and bring-up traces were used for parameter selection and are excluded from those headline statistics.

In the validation trace, $\rho = 0.886$ (88.6% of rounds), complementary to the adaptive first-deadline timeout rate reported with the results depicted in Table 11. After warm-up,

measured slot edges remained within ± 2 ms of the planner schedule under the USB-logged setup, consistent with the scheduled timing model used to derive $S(c)$ and $D(c)$.

6.6. Artefacts and Reproducibility

The results in Section 7 are obtained from instrumented firmware logs collected on the four-node testbed for the 2514-round post-tuning validation window. The corresponding USB serial traces, processed tables, and analysis scripts used to generate the reported figures and tables are publicly available; see the Data Availability Statement. The evaluation characterises implementation correctness, bounded retry and queue behaviour, timing performance, and adaptive versus fixed-timing sensitivity on this platform.

7. Results and Analysis

We structure the results as follows: mode-level latency, trace-replay comparison to static SF12 scheduling (Section 3.3), per-node summaries, distribution and correlation views, timeout and recovery behaviour, gateway sensitivity, and a synthesis tied to Table 11.

7.1. Participation-Mode Analysis

The framework supports two adaptive finalisation modes based on node participation. Table 10 presents the breakdown across 2514 validation rounds. Figure 2 visualises the same mixture.

Table 10. Coordination latency by participation mode over the measured 2514-round adaptive validation window (seconds).

Mode	Threshold	Rounds	Median (s)	90th %ile (s)	95th %ile (s)
Quorum (3/4)	75%	1580	2.1	2.8	3.2
Full participation (4/4)	100%	934	5.3	6.4	7.2
Overall	–	2514	3.4	6.4	7.2

Figure 2 is constructed from the post-tuning 2514 coordination rounds. For each round the latency $L = t_{\text{finalise}} - t_{\text{init}}$ is placed into the histogram. Rounds are classified by recorded terminal state: quorum mode if three-of-four contributions caused finalisation (1580 rounds), and full participation if four-of-four contributions were received within W_{perf} (934 rounds). The histogram uses 30 equal-width bins over the observed latency range, matching the analysis script used to generate the figure.

The histogram shows a faster mass corresponding to quorum finalisation and a slower mass corresponding to full participation, with a small tail attributable to bounded single-retry recovery.

The system achieved three-of-four quorum finalisation (a three-node threshold in our four-node setting) in 62.9% of rounds (1580/2514) with median latency of 2.1 s. Full participation (4/4) occurred in 37.1% of rounds (934/2514) with median latency of 5.3 s, reflecting the additional bounded wait time specified by the window W_{perf} in Algorithm 2. Interpreting these statistics, the scheduler is designed to expose an explicit latency–completeness trade-off: when a quorum is reached quickly the round can finalise early, while rounds that keep waiting for 4/4 finalisation are intentionally slower but more complete in the sense of including all participants' contributions under honest firmware. These results should be read within the evaluated crash/omission setting, not as evidence of adversarial Byzantine tolerance.

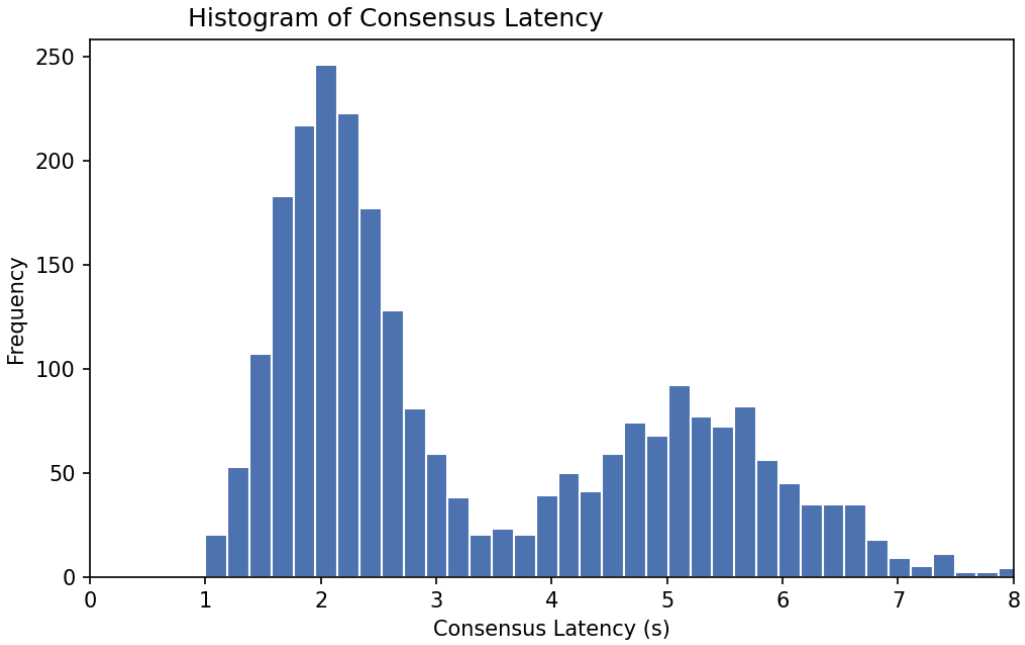


Figure 2. Histogram of measured coordination latency L (seconds) across 2514 adaptive validation rounds. Bars aggregate all rounds; the bimodal shape reflects 1580 quorum-mode and 934 full-participation finals. Bin count: A total of 30 equal-width bins. Data are measured hardware logs, not replay.

7.2. Trace-Replay Comparison to Static Slot Scheduling

This section separates three notions of baseline so the evaluation can be interpreted without conflating them: (i) Adaptive measured: The 2514-round validation trace reported throughout Section 7; (ii) Static executed (fixed configuration): A separate hardware run under fixed SF10 on the same class of devices (Table 8); and (iii) Static replay (fixed-SF12 worst case): The SSS counterfactual of Section 3.3, replaying the adaptive event log under $(S_{\text{static}}, D_{\text{static}})$ (Table 11).

SSS replay is a timing-policy sensitivity analysis under a frozen event sequence. It is not a substitute for a long-run executed static-SF12 campaign: closed-loop collisions, retry timing, and queue dynamics would differ under an actually executed static policy (Section 3.3). Table 11 is treated as an upper-bound style counterfactual and Table 8 as the executed fixed-configuration reference.

Table 8 summarises the executed fixed-SF10 run. Table 11 contrasts the measured adaptive controller against the SSS replay. Adaptive medians match Table 10. The SF10 reference uses a favourable fixed PHY and 120-byte payloads, so deadlines carry more slack than many adaptive rounds that sweep SF7–SF12; it is not a head-to-head reliability comparison on identical events. The adaptive trace reports $\rho = 0.886$ (11.4% first-deadline miss rate) with all misses recovered by a single retry.

Table 11 reports the median latency, 90th percentile latency, and first-deadline timeout rate for the measured adaptive controller and the trace-replayed static SF12 schedule (SSS) over the same 2514-round event log. The adaptive values are measured, whereas the SSS values are replay-derived.

Table 11. Measured adaptive performance versus trace-replayed static SF12 baseline.

Model	Median (s)	P90 (s)	Timeout Rate
Static SSS (SF12 replay)	11.3	12.0	5.1%
Adaptive (measured)	3.4	6.4	11.4%

Under the replayed static schedule, each slot assumes worst-case SF12 airtime, so median completion time increases toward the fixed-cycle duration implied by $(S_{\text{static}}, D_{\text{static}})$. The lower replay timeout rate reflects that D_{static} is looser than many adaptive per-round deadlines that triggered retries in the measured trace; it should not be read as evidence that static SF12 execution would necessarily yield fewer timeouts on hardware. Consistent with Section 3.3, we interpret Table 11 as a timing-policy sensitivity analysis under identical logged events (different timing rules), not as an executed head-to-head benchmark.

7.3. Timing and Finalisation Sensitivity

Two comparisons use the validation trace and the replay baseline from Section 3.3: static timing rules versus adaptive $(S(c), D(c))$, and quorum versus full-participation finalisation.

Table 11 replaces $(S(c), D(c))$ with $(S_{\text{static}}, D_{\text{static}})$ on the same 2514-round event log. Relative to the measured adaptive controller, the static-schedule replay increases median completion time from 3.4 s to 11.3 s, relative to the replay-derived worst-case SF12 timing baseline on the same logged event sequence, isolating sensitivity to the timing rules updated by the MAPE-K loop.

Table 10 conditions on observed participation in the same trace: three-of-four quorum mode has median latency 2.1 s, while full participation has median latency 5.3 s. The gap quantifies the additional wait when all participants must be included, holding the airtime-derived timing model fixed.

The replay baseline changes timing rules while holding the logged event sequence fixed; Table 10 conditions on observed participation rather than enforcing a fixed policy. A full factorial study (static timing $\times$ forced finalisation mode) would require additional executed runs or counterfactual logs. Here, the results depicted in Table 11 attribute completion-time sensitivity to removing adaptive timing, and the results depicted in Table 10 attribute the latency component tied to the finalisation policy.

7.4. Analytical Regimes for Timing Adaptation

The adaptive controller improves completion time by reducing conservative margin in the timing rules that govern unused airtime. This yields strong, system-level regime insights directly from the timing model.

From (2), slot spacing decomposes as

$$S(c) = \lceil \text{ToA}(c) \rceil + (t_{\text{sw}} + t_g),$$

so timing adaptation changes the dominant term $\text{ToA}(c)$ when PHY settings vary. A fixed worst-case schedule instead provisions S_{static} using $\text{ToA}(c_{\text{SF12}})$ for every round (4). The resulting sensitivity is governed by the ratio

$$\frac{S_{\text{static}}}{S(c)} \approx \frac{\text{ToA}(c_{\text{SF12}}) + t_{\text{sw}} + t_g}{\text{ToA}(c) + t_{\text{sw}} + t_g},$$

which increases when $\text{ToA}(c)$ is much smaller than the worst-case airtime.

LoRa airtime grows rapidly as data rate decreases (higher spreading factor, lower bandwidth). Adaptation helps most in regimes where (i) the node frequently operates at better-than-worst-case PHY settings (e.g., lower SF or higher BW than the SF12/125 kHz bound used by SSS), and (ii) PHY settings vary across rounds so that a single static schedule would have to provision for the worst case. When the system already runs near the worst-case airtime for extended periods, adaptive timing has limited headroom because $\text{ToA}(c) \approx \text{ToA}(c_{\text{SF12}})$.

When $\mathrm{ToA}(c) \gg (t_{\mathrm{sw}} + t_g)$, slot time is airtime-dominated and timing adaptation directly translates into shorter slots and deadlines. When $\mathrm{ToA}(c)$ becomes comparable to fixed overheads $(t_{\mathrm{sw}} + t_g)$, the marginal benefit of further reducing airtime shrinks because $S(c)$ approaches an overhead floor. This predicts diminishing returns for adaptation at high data-rate PHY settings.

Table 10 shows a second, orthogonal regime effect: when one participant is delayed or absent, quorum finalisation avoids waiting for full participation and reduces latency. In rounds where all participants respond promptly, the policy difference collapses and both modes converge toward the same airtime-limited schedule.

7.5. Latency Distribution by Node

Figure 3 groups measured coordination latency by observing node under common protocol settings, testing whether execution behaviour exhibits node-specific bias. Small sample differences can arise from role or slot position within a round, local MCU scheduling jitter, differing packet-receipt timing, temporary link variability, Wi-Fi coexistence, and different mixes of quorum versus full-participation observations. Table 12 aggregates latency by node. A Kruskal–Wallis test on per-round samples grouped by node gives $H = 2.14$, $df = 3$, and $p = 0.543$ [38]. The similar medians and interquartile ranges across nodes, together with the non-significant Kruskal–Wallis result, provide no evidence of node-specific latency differences under this mix of quorum and full-participation rounds.

Distribution of measured coordination latency by observing node across all 2514 validation rounds. The solid line denotes the median, the box the interquartile range (IQR), and the whiskers the most extreme observations within 1.5 × IQR. The dashed green line denotes the mean. Outliers beyond the whiskers are suppressed. The comparison assesses potential node-specific latency bias under common protocol settings.

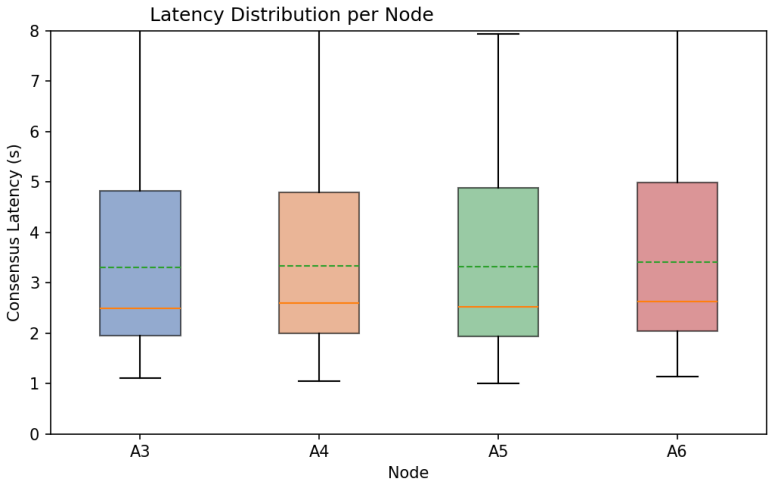


Figure 3. Measured coordination latency distributions across observing nodes over 2514 validation rounds.

Table 12. Per-node coordination latency statistics derived from all 2514 measured rounds. 95% CI denotes the confidence interval for the median as reported by the analysis pipeline.

Node	Median (s)	95% CI	IQR (s)	P90 (s)	P95 (s)
A3	3.38	[3.31, 3.45]	2.15	6.41	7.15
A4	3.42	[3.35, 3.49]	2.18	6.38	7.09
A5	3.35	[3.28, 3.42]	2.12	6.45	7.22
A6	3.45	[3.38, 3.52]	2.20	6.42	7.18
Overall	3.40	[3.33, 3.47]	2.16	6.42	7.16

Per-node latency distributions are bimodal across A3–A6, reflecting the system's dual finalisation policy: faster three-of-four quorum rounds (median 2.1 s) and slower full-participation rounds (median 5.3 s). The interquartile ranges in Table 12 are therefore largely attributable to this intentional mixture of execution modes.

Whisker lengths remain bounded across nodes, with 90th-percentile latencies below 6.5 s in this trace.

7.6. Latency Cumulative Distribution

To complement the per-node latency spread, we summarise the cumulative distribution of latency over the full set of measured rounds. This view quantifies the proportion of rounds completed within specific latency thresholds and complements the mode breakdown in Table 10. Figure 4 summarises the aggregated distribution.

Figure 4 makes the tail behaviour explicit: most rounds complete within a few seconds, and the remaining probability mass corresponds to full-participation waiting and the bounded retry path.

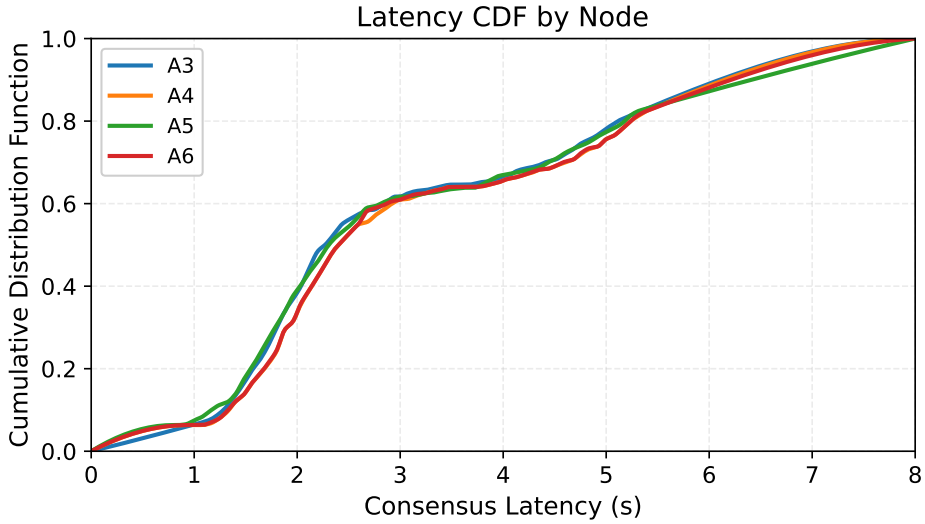


Figure 4. Empirical cumulative distribution of measured coordination latency (seconds) over the 2514-round adaptive validation window (aggregated across nodes). Data are measured hardware logs.

The resulting CDF summary indicates how quickly rounds finalise across nodes, including both quorum-based and full-participation outcomes. The rise around 5.3–5.5 s aligns with the concentration of full-participation (4/4) rounds (37.1% of rounds), while the remaining mass reflects faster quorum finalisation and recovery cases. Latencies remain bounded in this dataset, with 90% of rounds completing within 6.4 s and 95% within 7.16 s; the small tail beyond 7 s corresponds to rounds that invoke the bounded single-retry path. Together with the results depicted in Table 10, this supports the interpretation that adaptive finalisation yields a consistent mixture of execution modes within the exercised fault model and controlled conditions.

7.7. Correlation Between Metrics

We evaluated Pearson correlations across per-round log vectors (latency to finalisation, deadline-driven retries, gateway Wi-Fi events, and recovery/cleanup counters). Latency shows only weak association with retry-triggering timeouts ($r = -0.11$), indicating that timeout occurrence alone does not explain the dominant latency variation. The bimodal latency structure arises from the dual-mode finalisation policy (quorum versus full-participation), and the bounded recovery design (at most one retry, bounded queue

$K = 10$) prevents unbounded backlog-driven delay. Figure 5 visualises the correlation structure.

Figure 5 summarises these relationships: latency correlates primarily with participation-mode and airtime-driven timing effects, while recovery/control-plane events are more strongly coupled among themselves than to end-to-end completion time in this dataset.

Strong correlations are observed among control-plane and recovery-related events, including Wi-Fi connectivity changes and enforced cleanup actions. No strong correlation is observed between latency and these control-plane events in this dataset, which is consistent with recovery activity not dominating end-to-end round completion time under the evaluated conditions.

Latency is primarily driven by timing variance and bounded retry behaviour, while queueing delay and recovery overhead remain negligible in this dataset.

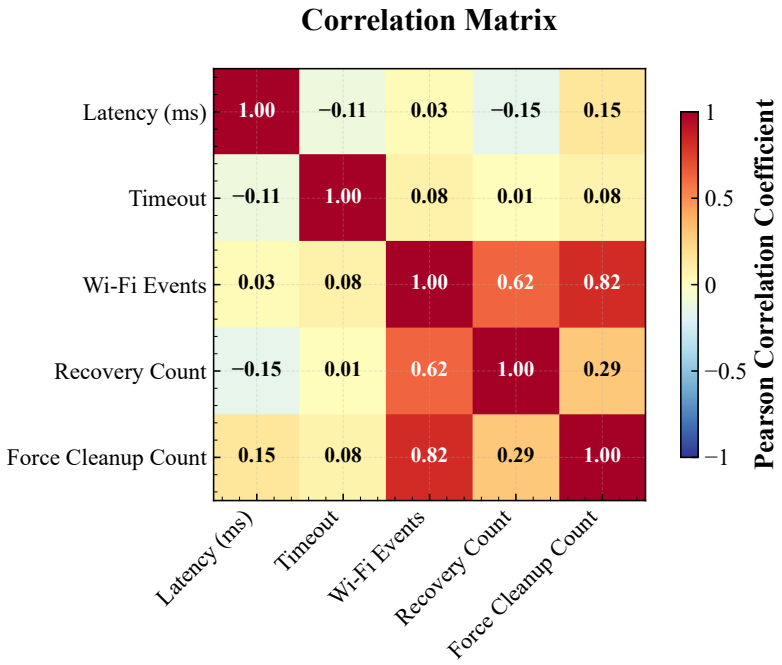


Figure 5. Pearson correlation matrix over per-round metrics derived from the measured 2514-round validation trace. Cell values are Pearson r; colour encodes signed correlation magnitude.

7.8. Timeout Density and Recovery Behaviour

We summarise deadline-miss (timeout) events observed before the bounded single-retry path completes. Timeout events occurred in 287 rounds (11.4%), with all cases recovered via a single retry in this dataset. Figure 6 plots their distribution over the validation window.

Figure 6 shows that deadline misses are episodic rather than persistent, consistent with the design goal of bounded, self-stabilising recovery rather than oscillatory behaviour.

Timeout events are relatively infrequent, occurring in 287 out of 2514 rounds (11.4%), indicating that the majority of rounds complete within the initial deadline without invoking recovery. All timeout events in this dataset were resolved through the bounded single-retry mechanism, resulting in zero failures due to exhausted retries. This is consistent with the intended role of the retry policy in Algorithm 1: bounded recovery without repeated retransmission.

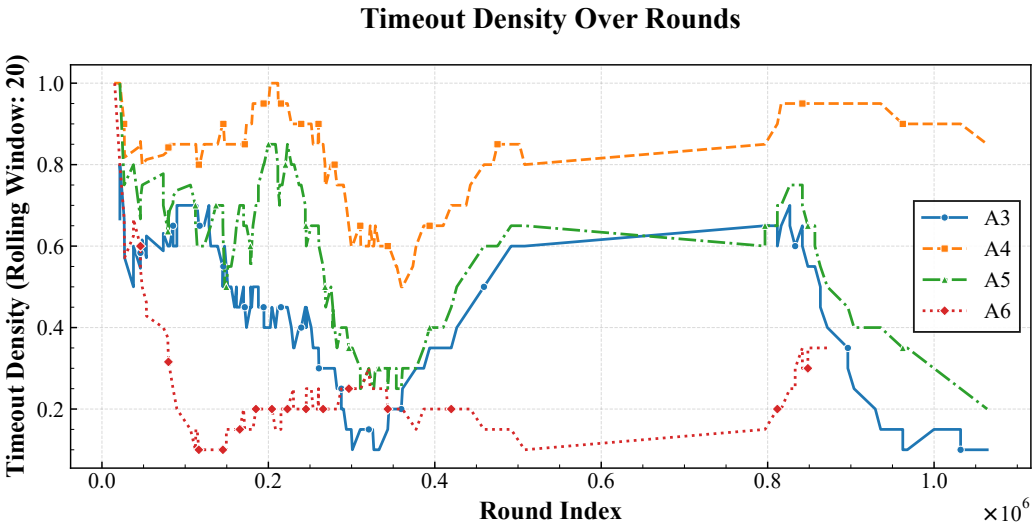


Figure 6. Timeout density over the 2514-round measured validation window (287 deadline misses, 11.4%). Curve: rolling count of timeout events with window size 20 rounds. The vertical scale is restricted so sparse episodic events remain visible. Timeout events are handled by the bounded single-retry mechanism.

The observed timeout rate provides a quantitative baseline for bounded recovery under the tested conditions. Together with the single-retry path, these results show that transient timing violations were handled predictably in this dataset and did not cascade into persistent failure.

7.9. Impact of Gateway Connectivity: Wi-Fi Disturbance vs Timeout Rate

To examine whether gateway connectivity disturbances coincide with mesh deadline misses, we compare the number of Wi-Fi events within each log segment with the corresponding mean timeout rate. Coordination safety and liveness do not depend on gateway connectivity (Section 3). Figure 7 shows this relationship across the measured validation window.

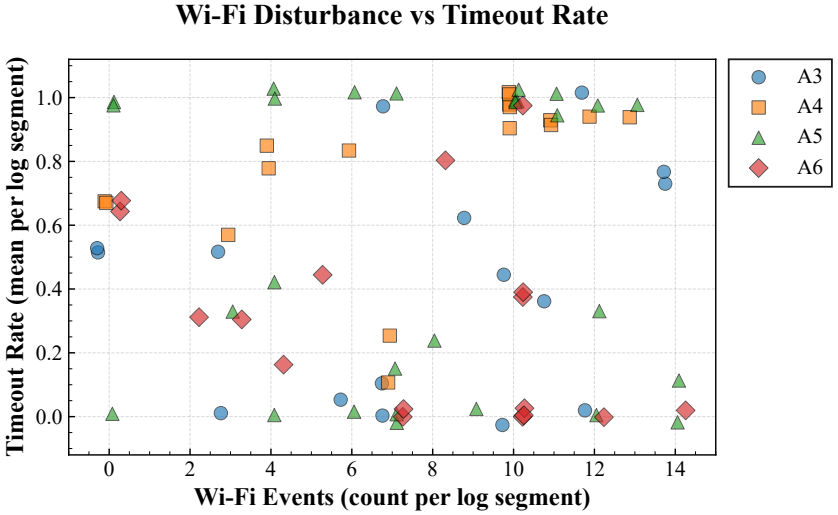


Figure 7. Relationship between gateway Wi-Fi events and mesh timeout rate across the measured validation window.

The observed association is weak (Pearson $r = 0.18$, $r^2 = 0.032$), with Wi-Fi event frequency accounting for approximately 3.2% of the observed variation in timeout rate. Given the large sample size ($n = 2514$), statistical significance alone would not imply

substantial practical importance. The weak association is consistent with the offline-first design, in which gateway connectivity is opportunistic and does not lie on the critical path for mesh round completion. Within this dataset, timeout behaviour is more strongly associated with mesh airtime and retry-policy effects than with gateway Wi-Fi activity.

7.10. Summary

Table 11 provides a conservative upper bound on completion times under worst-case airtime provisioning by replaying the validation event log with a fixed-SF12 static schedule (SSS). In that upper-bound analysis, SSS yields a higher median completion time than the measured adaptive trace, while the adaptive controller maintains the bimodal 2.1 s/5.3 s behaviour reported in Table 10. The ratio between the replayed SF12 median and the measured adaptive median is reported as a timing-policy sensitivity under identical logged events. Together with the executed fixed-SF10 reference point in Table 8, the evaluation separates measured adaptive behaviour from both (i) executed fixed-configuration operation and (ii) upper-bound replay under worst-case provisioning. Across all 2514 validation rounds under the evaluated conditions, 62.9% of rounds used quorum finalisation and 37.1% waited for full participation; 88.6% met the first adaptive deadline, 11.4% used one retry, and none exhausted the retry budget. Invariant checks I1–I4 remained clean, Kruskal–Wallis found no node bias ($p = 0.543$), and Wi-Fi jitter stayed a weak correlate of timeouts ($r^2 = 0.032$).

8. Discussion

8.1. Interpretation of Timing Adaptation

The four-node USB-logged setup shows that a MAPE-K scheduler around a lightweight scheduled quorum path can reduce completion time relative to the replay-derived worst-case SF12 timing baseline on the same logged event sequence while preserving bounded queue and retry behaviour under crash and omission faults.

Table 11 isolates timing policy by replaying an identical event sequence and therefore supports a timing-sensitivity comparison under fixed logged events. Table 8 reports executed fixed-configuration behaviour under a favourable PHY on identical hardware. Together they separate measured adaptive behaviour from executed fixed-configuration operation and from a conservative replay upper bound.

Airtime-driven adaptation matters because LoRa packet duration dominates slot spacing on constrained meshes: when $\mathrm{ToA}(c)$ shrinks, unused waiting under a fixed worst-case schedule becomes avoidable without changing quorum semantics.

8.2. Relationship to LoRa MAC, Routing, and Backoff

Adaptive LoRa MAC mechanisms retune radio parameters for delivery and energy [5,10,11]; mesh routing selects forwarding paths [23]; TCP-style backoff lengthens retries under overload. The present controller sits above those mechanisms: it maps observed airtime into $S(c)$ and $D(c)$ for scheduled quorum execution and uses exponential backoff only as secondary overload control after `SendBusy`. Routing can carry the coordination messages; the timing layer complements that forwarding path.

Read with prior EBS work on offline-first voice and gateway integration [6–8], the results motivate treating wait time on the air as a first-class control variable when voice and ledger traffic share a duty cycle. Where gateways buffer finalised summaries or ledger artefacts for later backhaul, confidentiality planning must also account for harvest-now, decrypt-later exposure over the storage interval [39]. Applicable deployment assumptions include: small validator sets (n on the order of the evaluated four-node mesh or modestly larger neighbourhoods), honest participants under crash/omission, opportunistic gateway

backhaul, and acceptance of quorum finalisation when full participation is slow. Outdoor propagation, mobility, untethered clock drift, congestion under dense interference, and Byzantine behaviour remain unvalidated (Section 6.2).

8.3. Scalability and Deployment Envelope

With one scheduled transmission opportunity per participant, $D(c) = O(nS(c))$ (Section 9). The deployment envelope is therefore small to moderate validator neighbourhoods, with hierarchical local quorums as the practical path for larger meshes.

8.4. Security Implications and Physical-Layer Denial of Service

The implementation invariants (I1–I4) and single-retry bound address deterministic cleanup under omissions and crashes. Byzantine equivocation, forged vote identifiers, and adversarial scheduling remain outside the evaluated fault model and could violate the informal arguments in Section 4.4. Congestion that systematically violates the configured Δ_{clk} envelope is likewise outside the measured regime.

Intentional RF jamming primarily affects availability under the stated assumptions:

$$\text{jamming} \rightarrow \text{packet omission} \rightarrow \text{deadline miss} \rightarrow \text{retry} \rightarrow \text{quorum failure}. \tag{9}$$

The controller can compensate for ordinary airtime variation and transient omissions, but it cannot guarantee progress against sustained intentional RF denial of service. Channel diversity, frequency hopping, spatial redundancy, or alternate backhaul remain future architectural options.

Section 9 summarises expected behaviour beyond $n = 4$, including how $S(c)$ and $D(c)$ scale with validator-set size and duty-cycle constraints.

9. Scalability and Deployment Considerations

The evaluation in this paper uses $n = 4$ nodes to enable end-to-end instrumentation and controlled fault injection. Unlike many coordination studies that report only simulation-based scalability, this work validates timing, recovery, and invariant behaviour on a deployed prototype with real LoRa airtime and embedded scheduling constraints. The following discussion is an analytical scalability treatment rather than an extrapolation of measured latency to larger n.

Under scheduled transmission with n_{slots} slots per round, the deadline in (3) grows approximately linearly with the number of scheduled slots:

$$D(c) = n_{\text{slots}} S(c) + t_{\text{lead}} + t_{\text{gw}} + t_{\text{margin}}.$$

In the simplest mapping where $n_{\text{slots}} \propto n$ (one transmit opportunity per participant per phase), both round duration and worst-case completion time increase as $O(n\,S(c))$. Because $S(c)$ is dominated by the airtime term $\text{ToA}(c)$ in low-rate regimes, increasing n at a fixed PHY setting can quickly drive $D(c)$ beyond operationally acceptable limits. The adaptive controller mitigates this by tracking the current $\text{ToA}(c)$ and selecting spacing/deadlines that avoid provisioning every round for a worst-case PHY. However, it does not remove the fundamental linear dependence of scheduled airtime on the number of required transmissions per round.

Table 5 summarises the companion complexity bounds: $O(n)$-scheduled transmissions per round, $O(n)$ per-node verification, $O(n^2)$ mesh-wide verification, $O(n)$ participant state, and $O(nK)$ queue-related state. The practical deployment envelope is therefore small-to-moderate validator sets, with hierarchical local quorums for larger networks.

In unlicensed LPWAN bands, regulatory duty-cycle limits and fair-use constraints bound the feasible number of transmissions per device per hour [26,33]. Increasing n increases aggregate on-air load even when individual nodes transmit infrequently. When the channel is airtime-limited, two effects become dominant: (i) more contention and collisions reduce effective delivery probability, and (ii) deadlines must grow to accommodate longer schedules, which increases application-perceived latency. Practically, larger validator sets benefit from reducing per-round message count (e.g., smaller payloads, fewer phases, or aggregated signatures where feasible) and from structuring coordination so that only a subset of nodes is required to transmit in every round.

For $n > 4$, two regimes emerge. In a reliability-dominated regime (higher SF and lower BW), airtime dominates $S(c)$, so both adaptive and static schedules lengthen substantially; in this regime, quorum finalisation is particularly valuable because it avoids waiting for all participants. In a higher-rate regime (lower SF or higher BW), fixed overheads ($t_{\mathrm{sw}} + t_g$ and processing time) become a larger fraction of $S(c)$, and the marginal gain from further airtime reductions diminishes, as discussed in Section 7.4. Regardless of regime, larger n increases the probability that at least one node is delayed or temporarily unreachable; therefore, the mixture of quorum versus full-participation rounds is expected to shift toward quorum completion in the absence of strong synchronisation and stable links.

The proposed design is best suited to small-to-moderate validator sets where quorum semantics are acceptable and where per-round scheduling can be kept short relative to application deadlines. For larger meshes, a practical path is hierarchical coordination: use local quorums within neighbourhoods or clusters, then propagate committed summaries across clusters using store-and-forward. This preserves offline-first operation while avoiding $O(n)$ on-air coordination at the full network scale.

A staged programme—requiring experiments beyond this paper—could (i) repeat the present instrumentation on eight- or twelve-node indoor meshes to measure growth in $D(c)$ and timeout rates, (ii) introduce controlled geographic separation and untethered clocks outdoors, and (iii) feed empirical airtime and participation distributions into discrete-event simulators so larger topologies are explored with parameters grounded in prototype measurements. Simulation remains useful for sweeping policy choices, but the four-node trace provides anchor points for slot spacing, retry rate, and quorum-mode mix that pure simulation studies often lack.

Outdoor propagation and interference, battery budgets, larger validator sets, executed long-run static SF12 baselines, and adversarial Byzantine behaviour are left to future field and adversarial testbeds.

10. Conclusions

This paper presented a cross-layer execution-timing controller for offline-first quorum coordination on constrained LoRa mesh nodes. The contribution is the closed-loop mapping from measured LoRa time-on-air into per-round scheduling variables $S(c)$ and $D(c)$, dual-mode finalisation, and bounded retry and queue behaviour with FIFO capacity $K = 10$ and at most one retry, realised through an embedded MAPE-K loop on each node.

On a four-node SX1276 mesh at 915 MHz, 2514 measured coordination rounds under crash and omission faults with honest firmware yielded median latency 2.1 s in three-of-four quorum mode and 5.3 s under full participation (pooled median 3.4 s). Across the validation window, 88.6% of rounds met the first adaptive deadline, 11.4% completed through one retry, zero rounds exhausted the retry budget, and invariants I1–I4 recorded zero violations. Relative to the replay-derived SF12 static schedule on the same event log (11.3 s median), adaptive timing reduced unused waiting while remaining comparable in

latency character to an executed SF10 fixed-configuration reference (2.7 s/5.2 s quorum/full-participation latency).

The results show that LoRa airtime can serve as an execution-control signal for higher-layer coordination on constrained devices. The controller sits above adaptive LoRa MAC, mesh routing, and TCP-style backoff, which address delivery paths or overload recovery, and uses airtime to schedule quorum execution.

These conclusions apply to the evaluated four-node laboratory platform under crash and omission faults with honest firmware. Future work includes larger topologies, untethered outdoor timing, power-instrumented energy budgets, executed long-run static SF12 baselines, and adversarial evaluation.

Author Contributions: Conceptualization, F.K. and P.B.; methodology, F.K.; software, F.K.; validation, F.K., P.B., J.B. and R.A.; formal analysis, F.K.; investigation, F.K.; resources, P.B. and R.A.; data curation, F.K.; writing—original draft preparation, F.K.; writing—review and editing, F.K., P.B., J.B. and R.A.; visualization, F.K.; supervision, P.B., J.B. and R.A.; project administration, F.K.; funding acquisition, P.B. and R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the SmartSat Cooperative Research Centre (CRC), whose activities are funded by the Australian Government's CRC Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are publicly available at https://github.com/f2ka07/adaptive-timing-data/ (accessed on 21 July 2026). The repository contains the USB serial traces, processed tables, and analysis scripts for the 2514-round post-tuning validation window used to generate the figures and tables reported in this manuscript.

Acknowledgments: The authors acknowledge Mark Rice of Safety from Space for expert technical advice and for the provision of technology that supported the experimental work reported in this paper. The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results. Generative AI tools were used for language editing and improving the clarity and readability of the manuscript. All scientific content was developed, reviewed, and verified by the authors, who take full responsibility for the final manuscript.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAPE-K	Monitor–Analyse–Plan–Execute–Knowledge
LoRa	Long Range (chirp-spread-spectrum radio)
LoRaWAN	LoRa Wide Area Network
EBS	Emergency Buddy System
EBSC	EBS coordination layer (quorum timing firmware)
DTN	Delay-Tolerant Networking
PHY	Physical Layer
MAC	Medium Access Control
ToA	Time on Air
SSS	Static Slot Scheduling (fixed-SF12 timing rules)
SF	Spreading Factor
BW	Bandwidth
CR	Coding Rate
BFT	Byzantine Fault Tolerance

MCU	Microcontroller Unit
NSM	Network and Service Management

References

1. Beke, T.; Dijk, E.; Ozcelebi, T.; Verhoeven, R. Time Synchronization in IoT Mesh Networks. In *Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC)*; IEEE: New York, NY, USA, 2020; pp. 1–8. [CrossRef]
2. Li, C.; Cao, Z. LoRa Networking Techniques for Large-Scale and Long-Term IoT: A Down-to-Top Survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
3. Sun, Z.; Yang, H.; Liu, K.; Yin, Z.; Li, Z.; Xu, W. Recent Advances in LoRa: A Comprehensive Survey. *ACM Trans. Sens. Netw.* **2022**, *18*, 67. [CrossRef]
4. Reynders, B.; Meert, W.; Pollin, S. Power and Spreading Factor Control in Low Power Wide Area Networks. In *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*; IEEE: Paris, France, 2017; pp. 1–6. [CrossRef]
5. Abdelfadeel, K.Q.; Zorbas, D.; Cionca, V.; Pesch, D. FREE—Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN. *IEEE Internet Things J.* **2020**, *7*, 669–683. [CrossRef]
6. Kagai, F.; Branch, P.; But, J.; Allen, R. Voice Over LoRa. In *Proceedings of the 2024 International Conference on Information Networking (ICOIN)*; IEEE: Ho Chi Minh City, Vietnam, 2024; pp. 526–531. [CrossRef]
7. Kagai, F.; Branch, P.; But, J.; Allen, R.; Rice, M. Rapidly Deployable Satellite-Based Emergency Communications Infrastructure. *IEEE Access* **2024**, *12*, 139368–139410. [CrossRef]
8. Kagai, F.; Branch, P.; But, J.; Allen, R.; Rice, M. Performance Evaluation of Low-Bitrate Voice Using Spread Spectrum Techniques for Satellite-Based Emergency Communication. *Comput. Netw.* **2025**, *270*, 111560. [CrossRef]
9. Lim, J.T.; Han, Y. Spreading Factor Allocation for Massive Connectivity in LoRa Systems. *IEEE Commun. Lett.* **2018**, *22*, 800–803. [CrossRef]
10. Bouazizi, Y.; Benkhelifa, F.; ElSawy, H.; McCann, J.A. SF-Adaptive Duty-Cycled LoRa Networks: Scalability, Reliability, and Latency Tradeoffs. *IEEE Trans. Commun.* **2024**, *73*, 1042–1057. [CrossRef]
11. Alahmadi, H.; Bouabdallah, F.; Al-Dubai, A.; Ghaleb, B. A Novel Autonomous Adaptive Frame Size for Time-Slotted LoRa MAC Protocol. *IEEE Trans. Ind. Inform.* **2024**, *20*, 12284–12293. [CrossRef]
12. Kephart, J.O.; Chess, D.M. The Vision of Autonomic Computing. *Computer* **2003**, *36*, 41–50. [CrossRef]
13. Castro, M.; Liskov, B. Practical Byzantine Fault Tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI'99)*; OSDI: New Orleans, LA, USA, 1999; pp. 173–186.
14. Howard, H.; Malkhi, D.; Spiegelman, A. Flexible Paxos: Quorum Intersection Revisited. In *Proceedings of the 20th International Conference on Principles of Distributed Systems (OPODIS 2016)*; Leibniz International Proceedings in Informatics (LIPIcs); OPODIS: Dagstuhl, Germany, 2017; Volume 70, pp. 25:1–25:14. [CrossRef]
15. Huebscher, M.C.; McCann, J.A. A Survey of Autonomic Computing—Degrees, Models, and Applications. *ACM Comput. Surv.* **2008**, *40*, 7. [CrossRef]
16. Cerf, V.; Burleigh, S.; Hooke, A.; Torgerson, L.; Durst, R.; Scott, K.; Fall, K.; Weiss, H. *RFC 4838: Delay-Tolerant Networking Architecture*; IRTF: Reston, VA, USA, 2007. [CrossRef]
17. Fall, K. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03)*; ACM: Karlsruhe, Germany, 2003; pp. 27–34. [CrossRef]
18. Buchman, E. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. Master's Thesis, University of Guelph, Guelph, ON, Canada, 2016.
19. Yin, M.; Malkhi, D.; Reiter, M.K.; Gueta, G.G.; Abraham, I. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC'19)*; ACM: Toronto, ON, Canada, 2019; pp. 347–356. [CrossRef]
20. Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; Zeldovich, N. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)*; ACM: Shanghai, China, 2017; pp. 51–68. [CrossRef]
21. Ailijiang, A.; Charapko, A.; Demirbas, M.; Kosar, T. WPaxos: Wide Area Network Flexible Consensus. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 211–223. [CrossRef]
22. Dwork, C.; Lynch, N.A.; Stockmeyer, L. Consensus in the Presence of Partial Synchrony. *J. ACM* **1988**, *35*, 288–323. [CrossRef]
23. Pueyo Centelles, R.; Meseguer, R.; Freitag, F.; Baig Viñas, R.; Navarro, L. A Minimalistic Distance-Vector Routing Protocol for LoRa Mesh Networks. *IEEE Access* **2024**, *12*, 128941–128962. [CrossRef]
24. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [CrossRef]

25. Bor, M.C.; Roedig, U.; Voigt, T.; Alonso, J.M. Do LoRa Low-Power Wide-Area Networks Scale? In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'16)*; ACM: Valletta, Malta, 2016; pp. 59–67. [CrossRef]
26. Adelantado, F.; Vilajosana, X.; Tuset-Peiró, P.; Martinez, B.; Melià-Seguí, J.; Watteyne, T. Understanding the Limits of LoRaWAN. *IEEE Commun. Mag.* **2017**, *55*, 34–40. [CrossRef]
27. Centenaro, M.; Vangelista, L.; Zanella, A.; Zorzi, M. Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. *IEEE Wirel. Commun.* **2016**, *23*, 60–67. [CrossRef]
28. Maurya, P.; Singh, A.; Kherani, A.A. A Review: Spreading Factor Allocation Schemes for LoRaWAN. *Telecommun. Syst.* **2022**, *80*, 449–468. [CrossRef]
29. Xu, J.; Wang, C.; Jia, X. A Survey of Blockchain Consensus Protocols. *ACM Comput. Surv.* **2023**, *55*, 278. [CrossRef]
30. LoRa Alliance. LoRaWAN FUOTA Process Summary: Technical Recommendation TR002 v1.0.0. Technical Recommendation TR002, LoRa Alliance, 2019. Available online: https://lora-alliance.org/wp-content/uploads/2020/11/tr002-fuota_process_summary-v1.0.0.pdf (accessed on 21 July 2026).
31. Rodrigues, J.J.; Soares, V.N. An Introduction to Delay and Disruption Tolerant Networks (DTNs). In *Advances in Delay-Tolerant Networks (DTNs)*, 2nd ed.; Rodrigues, J.J., Ed.; Woodhead Publishing Series in Electronic and Optical Materials; Woodhead Publishing: Cambridge, UK, 2021; pp. 1–20. [CrossRef]
32. Kagai, F.; Branch, P.; But, J.; Allen, R. Offline-First BFT Consensus with Token-Gated Security for Blockchain in Low-Bitrate Mesh Networks. *Blockchain Res. Appl.* **2026**, 100514. [CrossRef]
33. Kjendal, D. LoRa-Alliance Regional Parameters Overview. *J. ICT Stand.* **2021**, *9*, 35–46. [CrossRef]
34. Li, L.; Dai, Y. Time Synchronization Strategy and Implementation of LoRa Wireless Ad Hoc Network. In *Proceedings of the International Conference on Electronic Information Engineering and Computer Technology (EIECT 2021)*; SPIE: Kunming, China, 2021; Volume 12087. [CrossRef]
35. Driscoll, K.; Hall, B.; Sivencrona, H.; Zumsteg, P. Byzantine Fault Tolerance, from Theory to Reality. In *Proceedings of the Computer Safety, Reliability, and Security: 22nd International Conference (SAFECOMP 2003)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2788, pp. 235–248. [CrossRef]
36. Zhong, W.; Yang, C.; Liang, W.; Cai, J.; Chen, L.; Liao, J.; Xiong, N. Byzantine Fault-Tolerant Consensus Algorithms: A Survey. *Electronics* **2023**, *12*, 3801. [CrossRef]
37. Semtech Corporation. SX1276/77/78/79—137 MHz to 1020 MHz Low Power Long Range Transceiver. Datasheet, 2020. Document Revision Dated 26 May 2020. Available online: https://cdn.sparkfun.com/assets/7/7/3/2/2/SX1276_Datasheet.pdf (accessed on 21 July 2026).
38. MacFarland, T.W.; Yates, J.M. Kruskal–Wallis H-Test for Oneway Analysis of Variance (ANOVA) by Ranks. In *Introduction to Nonparametric Statistics for the Biological Sciences Using R*; Springer: Cham, Switzerland, 2016; pp. 177–211. [CrossRef]
39. Kagai, F.; Branch, P.; But, J.; Allen, R. Harvest-Now, Decrypt-Later: A Temporal Cybersecurity Risk in the Quantum Transition. *Telecom* **2025**, *6*, 100. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.