SyncWave: Decentralised Firefly Synchronisation for Embedded Cognitive Agentic Loops

Michael Breza
Imperial College London
London, United Kingdom
michael.breza04@imperial.ac.uk

Luca Mehl
Imperial College London
London, United Kingdom
luca.mehl@gmail.com

Julie McCann
Imperial College London
London, United Kingdom
j.mccann@imperial.ac.uk

ABSTRACT

Swarms of embedded cognitive agents, devices that sense, infer, and act in tight perception-action loops using TinyML, need to align their cognitive cycles to achieve coordinated collective behaviour. Time synchronisation is a fundamental prerequisite for alignment, yet current solutions either require a dedicated reference node (PTP, NTP), or a designated flood initiator per round (Glossy), making them unsuitable for ad-hoc swarms operating without infrastructure. In this paper we propose the use of bio-inspired Pulse-Coupled Oscillator (PCO) synchronisation to provide synchronisation without reference nodes. We present SyncWave, a fully decentralised protocol derived from the Firefly-Gossip (FiGo) paradigm, extended with epoch-based maximum-time synchronisation and Trickle-inspired exponential backoff. We validate SyncWave on a large-scale IoT testbed (161 nodes, 7 hops, 2 s time-to-synchronise), on a 4-node single-hop bench measurement (488 μs pairwise accuracy), and in discrete-event simulations across four scenarios. Simulation results confirm that Time-to-Synchronise(TTS) $\leq$ 1 s in both dense (20-node, 3–4 hops) and multi-hop linear (8-node, 7 hops) topologies; a synchronisation margin ratio $\rho_s = \delta_{\text{sync}}/\tau_{\text{loop}} = 0$ across all tested cognitive loop periods $\tau_{\text{loop}} \in \{10\text{–}200\}$ ms; and re-synchronisation after a partition-merge event within $T_{\text{merge}} = 500$ ms, without a central reference node or infrastructure.

CCS CONCEPTS

• **Networks** → **Sensor networks**; • **Computing methodologies** → *Multi-agent systems*; • **Hardware** → *Embedded systems.*

KEYWORDS

pulse-coupled oscillators, time synchronisation, embedded AI, cognitive agents, TinyML, decentralised, swarm, firefly

1 INTRODUCTION

Embedded cognitive agents can collectively sense, infer, and act in ways that exceed the capability of individual unorganised devices. Collective organisation facilitates embedded device management and enables applications such as cooperative anomaly detection in sensor networks, distributed visual coverage in drone swarms, coordinated irrigation control in precision agriculture, and multi-implant biomedical coordination [10, 21].

The first requirement for any such collective behaviour is to agree on time, *temporal coordination*. Agents must align their perception-action cycles so that they sense the same world at the same moment, exchange inferences before acting, and apply actuation within a coherent shared time window. Logical clocks [14] order causally related events, but not concurrent events and could not show that two samples are in the same time-window or that actuation will meet a deadline. Recent EmbeddedAI and TinyML research suggests that as embedded intelligence evolves from isolated agent inference toward cooperative decentralized agent systems, higher-level coordination mechanisms—such as belief synchronization, consensus protocols, and semantic state coordination—become increasingly necessary [22]. These all build on a more basic prerequisite, a shared temporal reference, which is the focus of this work.

Current time synchronisation solutions are poorly matched to this requirement due to their lack of limited synchronisation accuracy, or dependence on a single point-of-failure. The Precision Time Protocol (PTP) and its variants elect a Best Master Clock and construct a master-slave hierarchy creating a single point of failure that fails under node loss or topology change [11]. The Network Time Protocol (NTP) [18] achieves only millisecond accuracy, insufficient for tightly coupled cognitive loops. Glossy [7] achieves sub-microsecond accuracy via synchronous transmissions but requires a *designated flood initiator* for each synchronisation round, a structural form of infrastructure dependency. GPS is unavailable indoors, underground, or in RF-denied environments. None of these solutions are infrastructure-free: all require a node that plays a special role that, if removed, degrades or halts synchronisation.

In this paper we present SyncWave, a Pulse-Coupled Oscillator (PCO) based fully decentralised synchronisation protocol that provides sufficient synchronisation accuracy and resilience to topology dynamics by extending the Firefly-Gossip (FiGo) paradigm [5] with three key mechanisms: epoch-based total ordering on time, maximum-time synchronisation (MTS) for guaranteed monotonic convergence, and Trickle-inspired exponential backoff that decouples synchronisation speed from radio energy consumption. We make the following contributions:

- **Cognitive loop timing analysis.** We discuss the synchronisation accuracy required for coordinated TinyML inference loops across a representative range of applications.
- **SyncWave protocol and formal properties.** We present the SyncWave algorithm with convergence proofs.
- **SyncWave evaluation.** We evaluate SyncWave using bench measurements, a large-scale testbed, and discrete event simulation.

2 BACKGROUND

2.1 Embedded Cognitive Agent Loops

A cognitive embedded agent runs a *perception-action loop*: it samples its sensors, runs an on-device inference model, communicates partial inferences with neighbours, and applies an action. This loop repeats with period τ_{loop}. In this paper we use *cognitive* in

a narrow sense: the perception step is a *learned* inference model (TinyML), not a fixed function or algorithm. This distinguishes our cognitive agents from classical reflex agents [23]. The value τ_{loop} is inference-bound, and quantifies synchronisation budget, a term we define below. In this work we do not claim higher-level reasoning, belief maintenance, or planning. For our definition of collective behaviour, all agents must be event synchronised, aligned with the same temporal reference, so that their sensing windows, inference phases, and communication slots are coordinated.

TinyML enables the inference step on microcontrollers [21]. Inference latency depends on model complexity and hardware. The MLPerf Tiny benchmark [2] reports latencies from 20 to 60 ms for keyword spotting on ARM Cortex-M class SoCs (e.g. ∼23 ms on a Cortex-M7 and ∼58 ms on a Cortex-M4); anomaly-detection and image-classification workloads measured in the tens-of-milliseconds; simpler wake-word or anomaly models showed single-digit millisecond latencies. Heavier vision CNNs have longer latencies: MCUNetV2 reports measured visual-wake-word latencies of roughly 185–660 ms on a Cortex-M4-class MCU (SRAM <320 kB, Flash <1 MB) [16, Fig. 7b]. We add to these inference costs the sensing and communication overhead for the calculation of τ_{loop}.

2.2 Synchronisation Budget for Cognitive Loops

The synchronisation budget, δ_{sync}, is the worst-case synchronisation error between any two agents, or nodes, who exchange inferences about the same environmental event. The budget is set by the application: for coherent collective inference, the temporal offset between agents must stay well inside one loop period so that fused inferences refer to the same event. We define the synchronisation budget as: $\delta_{\text{sync}} \ll \tau_{\text{loop}}$

Cheng et al. [6], fusing detections across microcontroller-class vision nodes, find that fusion stays coherent while the inter-node offset is within one inference cycle, and that past ∼5 FPS explicit timestamp alignment is needed to avoid fusing misaligned observations; federated learning applies the same principle, bounding update staleness [9]. These results imply $\delta_{\text{sync}} \ll \tau_{\text{loop}}$ and justify a bound that is a fraction of one cycle rather than a specific constant; we use $\delta_{\text{sync}} \leq \tau_{\text{loop}}/10$ as a conservative operating point.

We motivate synchronised loops with the following use case. Consider distributed visual coverage by a drone swarm, where each agent runs an on-board camera, radar or LIDAR. The swarm fuses images into a single picture of the monitored area. A residual synchronisation error δ_{sync} between two agents translates directly into a spatial disagreement: at a closing speed v, their observations of the same object differ in position by $v \cdot \delta_{\text{sync}}$. For agents moving at $5\,\text{m}\,\text{s}^{-1}$ running a $\tau_{\text{loop}} = 100$ ms detection loop, our $\tau_{\text{loop}}/10$ budget admits $\delta_{\text{sync}} \leq 10$ ms and so bounds the disagreement at 5 cm. An unsynchronised swarm drifting by one full loop period would instead disagree by half a metre, enough to report the error that one object is two. The measured single-hop $\delta_{\text{sync}} = 488\,\mu$s sits 20× inside this budget, a disagreement of under 3 mm.

Table 1 compares synchronisation protocols from literature. Network topology bounds SyncWave's performance, as shown in the evaluation.

Table 1: Synchronisation accuracy. SyncWave is a single-hop bench measurement (Section 5.1); all others reported by the cited sources, obtained on differing hardware and measurement scopes. Results are indicative rather than a controlled comparison.

Protocol	δ_{sync}	Ref. node?
PTP [11]	∼1 μs	Yes (master)
Glossy [7]	∼1 μs	Yes (initiator)
SwarmSync [25]	127 μs	No
SyncWave	488 μs	**No**
FiGo [5]	∼500 μs	No
PCO [17]	480 μs	No
NTP [18]	∼1 ms–1 s	Yes (server)

3 RELATED WORK

Mirollo and Strogatz [19] proved PCO convergence for all-to-all excitatory coupling with monotone concave phase response functions. Werner-Allen et al. [30] translated this to WSNs with the Reachback Firefly Algorithm (RFA), which synchronised phase only (no epoch counter), leaving it susceptible to cycling between phases rather than converging monotonically.

Klinglmayr and Bettstetter [12] proved that *inhibitory* coupling outperforms excitatory coupling in dynamic networks and that lossy channels aid convergence; follow-on work extended the proofs to time-varying topologies [13]. Masood et al. [17] validated timestamped PCO on Z1 motes over IEEE 802.15.4, reporting 480 μs single-hop precision that degrades to 2.25 ms at diameter 5, but without an epoch counter or adaptive backoff. Schmidt et al. [24] showed a random fire phase is the most effective strategy for reducing collisions, and Brandner et al. [4] reached sub-microsecond precision via phase-rate equalization on FPGA radio hardware unavailable on commodity MCU+RTOS stacks.

The Firefly-Gossip protocol (FiGo) [5] combined firefly synchronisation with gossip dissemination for WSN management, and was formally verified using PRISM probabilistic model checking [29], proving synchronisation with probability 1.0 and steady-state synchronisation probability of 99.67% under temperature drift. SyncWave departs from FiGo in the update rule itself. FiGo averages phases circularly, so a node's time is pulled towards its neighbours' mean; the network-wide state can therefore oscillate, and convergence is not monotonic. SyncWave instead replaces averaging with *maximum consensus* over an epoch-extended timestamp: a node adopts a peer's time only when that time is strictly ahead. This makes $T_{\max}$ non-decreasing by construction (Theorem 1), converting an empirical convergence property into a guaranteed one, and it is what allows the total ordering that FiGo's phase-only representation cannot express. The cost is the fault sensitivity discussed in Section 4: maximum consensus amplifies an erroneously large timestamp where averaging would dilute it.

O'Keeffe, Hong and Strogatz [20] presented swarmalators: oscillators whose spatial and phase dynamics are mutually coupled, producing five emergent collective states including static synchrony and active waves. Barcis and Bettstetter [3] implemented swarmalators on wheeled robots and Crazyflie drones, demonstrating emergent space-time patterns in hardware. Their implementation

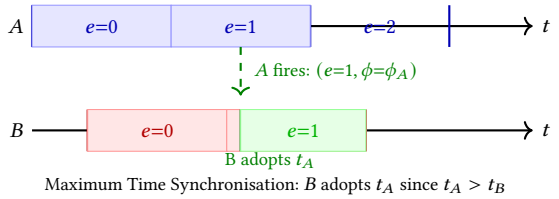

Figure 1: Maximum Time Synchronisation (MTS). When node A fires and $t_A > t_B$, node B adopts A's epoch and phase. The epoch counter provides total ordering on time and monotonic, non-cycling convergence.

relies on infrastructure: an OptiTrack motion-capture system (37 cameras) and the Crazyswarm centralised platform for drone coordination. SyncWave provides the wireless decentralised mechanism that makes it infrastructure-independent.

Vogell et al. identified *deadlocks* on star-graph topologies [26] and *chimera states* (partial synchronisation) [27] in PCO networks, failure modes whose conditions for SyncWave remain as open research questions.

Glossy [7] achieves sub-microsecond accuracy via synchronous transmissions (constructive interference) and forms the synchronisation backbone of recent distributed MCU learning systems [8]. Its accuracy is superior to SyncWave, but it requires a designated initiator for each flood round, making it centralised. SwarmSync [25] achieves 127 μs accuracy in a fully decentralised manner but is designed for *slow-changing topologies with resynchronisation intervals of minutes*, incompatible with the ms-timescale dynamics of mobile cognitive agent swarms.

4 THE SYNCWAVE PROTOCOL

SyncWave is a Pulse-Coupled Oscillator protocol for fully decentralised time synchronisation. Each node maintains a monotonically increasing *local time* $t = e \cdot \Phi + \phi$ where e is an epoch counter ($e \in \mathbb{N}$), ϕ is a phase counter ($\phi \in [0, \Phi)$, in μs), and Φ is the epoch period. This epoch-and-phase representation provides a total ordering on time, enabling maximum-time synchronisation. Algorithm 1 presents the core receive and fire loop. The full algorithm is symmetric across all nodes, no node has a special role.

4.1 Maximum Time Synchronisation

Upon receiving a message $(id_{msg}, e_{msg}, \phi_{msg})$, a node computes $t_{msg} = e_{msg} \cdot \Phi + \phi_{msg} + \hat{c}$ (where $\hat{c}$ is a propagation constant) and its own current time $t = e \cdot \Phi + \phi$. If $t_{msg} > t$, the node updates:

$$e \leftarrow \lfloor t_{msg}/\Phi \rfloor, \quad \phi \leftarrow t_{msg} \bmod \Phi \tag{1}$$

MTS provides *monotonic* convergence: the network-wide maximum time $T_{\max}(t) = \max_i T_i(t)$ is non-decreasing, preventing the cycling behaviour of circular-averaging schemes such as FiGo. A node that is *ahead* reduces its firing interval to inform behind neighbours; a node that is *behind* reduces its interval to catch up. Figure 1 illustrates the mechanism.

4.2 Randomised Firing and Exponential Backoff

SyncWave decouples the firing timer ψ from the epoch timer ϕ. The next firing time ψ_{fire} is drawn uniformly from a firing interval I: $\psi_{\text{fire}} \sim \mathcal{U}(0, I)$. This randomisation spaces transmissions to minimise packet collisions without any coordination, an approach independently validated by Schmidt et al. [24].

The firing interval adapts via exponential backoff (inspired by the Trickle algorithm [15]):

- After each firing the interval grows, $I \leftarrow \min(\beta \cdot I,\ I_{\max})$, slowing broadcasts as the network settles.
- Whenever a node detects desynchronisation (t_{msg} differs from t by more than ϵ), ResetInterval sets $I \leftarrow I_{\min}$, triggering rapid broadcast dissemination. As this can fire mid-interval, an interval containing a conflicting message is effectively cut short.

This mechanism makes SyncWave simultaneously fast during synchronisation and energy-efficient during maintenance: nodes converging from random initial phases fire aggressively at rate $\approx 1/I_{\min}$, while a fully synchronised network reduces to near-silent operation at rate $\approx 1/I_{\max}$.

4.3 Message Suppression

In dense networks, SyncWave optionally suppresses redundant broadcasts. A node tracks how many *unique* neighbours have fired with similar times (within ϵ of its own) in the current interval, counting these in c. If $c \geq k$ (message suppression threshold), the node cancels its next broadcast. Unlike naïve suppression, SyncWave counts *unique senders* and resets the count at each firing interval, avoiding suppression in multi-hop scenarios where bridge nodes must relay information.

4.4 Formal Properties

The analysis rests on three structural assumptions, (A1) bounded maximum degree $\Delta = O(1)$ (in-range nodes per receiver bounded independently of n); (A2) k, $I_{\min}$, $I_{\max}$ constant in n, firing times $\psi_{\text{fire}} \sim \mathcal{U}(0, I)$, and suppression never starving a frontier, each uninformed node adjacent to S has ≥ 1 synchronised neighbour permitted to transmit per interval, which holds because an *ahead* node resets rather than suppresses; and (A3) no clock drift for Theorem 1 (drift reintroduced for Theorem 2).

Lemma 1 (Frontier advance). *Let S hold the network maximum time $T_{\max}$, and call the not-yet-synchronised nodes adjacent to S the* frontier. *Let u be such a frontier node. Under (A1)–(A3), in any firing interval u receives $T_{\max}$ with probability at least the constant $p_{\min} = (1 - \epsilon/I_{\min})^{\Delta} > 0$.*

Proof. A neighbour in S holds a larger time, so it resets toward $I_{\min}$ (ResetInterval) and fires once per window of length $I_{\min}$ at a time uniform on $(0, I_{\min})$. Delivery fails only if one of the $\leq \Delta$ other in-range transmitters overlaps the vulnerable window of width ϵ, each with probability $\leq \epsilon/I_{\min}$; hence success probability $\geq (1 - \epsilon/I_{\min})^{\Delta}$. The exponent is the *local* degree Δ, not $n - 1$, so $p_{\min}$ does not vanish as the network grows. □

Algorithm 1 SyncWave: Message Receive and Fire Loop

```
e ← 0; φ ← 0; I ← I_min
ψ ← 0; ψ_fire ← U(0, I)
c ← 0; heard ← {}
while True do
    if Receive(id, e′, φ′) then
        t_msg ← e′ · Φ + φ′ + ĉ
        t ← e · Φ + φ
        if t_msg > t then
            e ← ⌊t_msg/Φ⌋; φ ← t_msg mod Φ
            if t_msg − t > ϵ then
                ResetInterval                          ▷ large gap: fire fast
            else if c < k and id ∉ heard then
                c ← c + 1; heard ← heard ∪ {id}
            end if
        else
            if t − t_msg > ϵ then
                ResetInterval                          ▷ sender is behind: inform
            else if c < k and id ∉ heard then
                c ← c + 1; heard ← heard ∪ {id}
            end if
        end if
    end if
    if ψ ≥ ψ_fire then                                 ▷ Fire timer expired
        if c < k then
            Transmit(id, e, φ)
        end if
        I ← min(β · I, I_max)
        ψ_fire ← U(0, I); ψ ← 0
        c ← 0; heard ← {}
    end if
    φ ← φ + 1; ψ ← ψ + 1
    if φ ≥ Φ then
        φ ← 0; e ← e + 1
    end if
end while
```

Theorem 1 (Convergence). *Under (A1)–(A3), in a connected network of n nodes with diameter D, SyncWave converges to a single time value with probability 1 in expected $O(D \log n)$ firing intervals.*

Proof. *Termination.* By the MTS update (1) a node only adopts a *strictly larger* time, so $T_{\max}(t)$ is non-decreasing and S never shrinks. The configuration is an absorbing Markov chain with sole absorbing state $S = V$; by Lemma 1 some node joins S each interval with probability $\geq p_{\min} > 0$, so absorption occurs with probability 1.

Time bound. A node at hop-distance $h \leq D$ is reached only after $T_{\max}$ crosses h successive (monotone) frontiers. For one frontier with m uninformed nodes, let R_t be the number still uninformed after t intervals ($R_0 = m$). Each interval informs a Binomial$(R_t, p_{\min})$ subset, *many* nodes at once, not one[1], so $\mathbb{E}[R_t] \leq (1-p_{\min})^t m$ and

$$\mathbb{E}[X] = \sum_{t \geq 0} \Pr[R_t \geq 1] \leq \sum_{t \geq 0} \min\{1, (1-p_{\min})^t m\} = O(p_{\min}^{-1} \log m).$$

[1]This treats the R_t uninformed nodes as informed independently each interval. Suppression correlates which synchronised neighbours transmit, so the events are mildly dependent; the independent model is the standard first-moment idealisation and the $O(\log n)$ order is unaffected, though it is an approximation rather than a per-realisation guarantee.

As $p_{\min}$ is constant (Lemma 1) this is $O(\log n)$ per frontier, *not* $\Theta(\sqrt{n})$: the $\sqrt{n}$ arises only when simultaneous senders grow with frontier size, degrading per-interval success and forcing a regime split at $\sqrt{n}$; suppression caps senders at the constant k, fixing $p_{\min}$ and removing that factor. Summing over the $\leq D$ monotone frontiers ($\sum_d m_d \leq n$) gives $O(p_{\min}^{-1} \sum_d \log m_d) = O(D \log(n/D)) = O(D \log n)$; monotonicity prevents the frontiers from interfering, so the bound does not compound. □

Theorem 2 (Bounded Drift Error). *With per-node drift $|\rho_i| \leq \rho_{\max}$, after convergence SyncWave maintains pairwise synchronisation within $B \leq 2D\rho_{\max} I_{\max} = O(D\rho_{\max} I_{\max})$.*

Proof. A synchronised network backs off to $I_{\max}$, so a laggard re-adopts its max-holding neighbour's time at most $I_{\max}$ after the previous adoption; between adoptions their clocks separate at relative rate $\leq 2\rho_{\max}$, giving single-hop error $\leq 2\rho_{\max} I_{\max}$ before MTS resets it. A node D hops from the max-holder relays through $\leq D$ intermediaries, each (worst case) at the end of its own resync window, so the per-hop errors add: $B \leq 2D\rho_{\max} I_{\max}$. Adoption of the maximum jumps instantaneously onto the larger value, resetting rather than compounding error, so accumulation stays linear in D. □

These properties distinguish SyncWave from FiGo (whose circular averaging produces cycling) and from RFA (no epoch counter; convergence not monotonic) and make it suitable for the continuous, safety-critical operation of cognitive agent swarms.

We give a brief fault model. Monotonicity can be problematic: because MTS adopts the maximum, one node reporting an erroneously large time dominates the swarm, and the maximal rule that guarantees convergence propagates the error. An averaging rule will dilute a faulty value, maximum consensus amplifies it. We assume crash faults only. Bounding the epoch jump accepted in one interval to the plausible advance $D \cdot \hat{c}$ confines a single fault to a bounded offset, at the cost of slower partition-merge recovery. Byzantine-tolerant maximum consensus is future work.

5 EVALUATION

5.1 Testbed Validation

We evaluated SyncWave on the FIT IoT-LAB testbed [1] using IoT-LAB M3 nodes (ARM Cortex-M3, AT86RF231 radio, IEEE 802.15.4) running RIOT-OS. Table 2 lists the protocol parameters used.

Dense topology (161 nodes, 7 hops): At an average degree of 37.6, SyncWave converges in **2 s**. For comparison, the cluster-based consensus methods CMTS and CCTS are evaluated on 20 nodes and converge in hundreds of broadcast rounds [28]; post-sync backoff drives the broadcast rate to near-zero.

Linear (multi-hop) topology (25 nodes, 12 hops): SyncWave converges within 3 s despite the extreme diameter (all 140 trials converging), scaling approximately as $O(D \log n)$, consistent with the formal bound.

Synchronisation accuracy: Measured with a Rigol1074Z logic analyser on 4 nodes in a laboratory bench test (GPIO pins on epoch boundary), SyncWave achieves **488 μs** worst-case pairwise error, comparable to FiGo (~500 μs) and well within the cognitive loop synchronisation budget.

Table 2: SyncWave parameters: testbed (FIT IoT-LAB) and Simulation. $\hat{c}$ differs because the simulation fires `received_packet` at exactly `airtime` after TX start; no software-stack overhead exists.

Symbol	Parameter	Testbed	Sim	Unit
Φ	Epoch period	1	1	s
$I_{\min}$	Min. firing interval	75,000	75,000	μs
$I_{\max}$	Max. firing interval	300	60	s
$\hat{c}$	Propagation constant	9,000	4,000	μs
k	Suppression threshold	2	2	—
β	Backoff base	2	2	—
ϵ	Desync threshold	20,000	20,000	μs

Table 3: Simulation results (n nodes, seed 42, 60 s or 90 s).

Scenario	n	TTS	δ_{sync}	BC/node
Dense (100×100 m, ~3–4 hops)	20	1 s	0 μs†	12
Linear chain (7 hops)	8	1 s	0 μs†	12
Partition-merge (T_{merge})	20	500 ms*	4,840 μs	5

*After bridge connection at $t = 30$ s. †Exact (integer) agreement: no modelled drift, exact delay compensation.

Network-wide accuracy and its measurement limit: We report single-hop accuracy from the bench measurement above rather than from the testbed because the M3 serial logging path timestamps output with an alternating ±8 ms error, giving an effective resolution of ±16 ms for any pairwise comparison derived from the logs. Any network-wide error we could quote from those logs would therefore sit at or below the noise floor of the measurement, and would bound rather than resolve the true error. Establishing network-wide and per-hop accuracy at scale requires hardware timestamping below this floor, and is left to future work.

5.2 Simulation for Cognitive Loop Workloads

We implement SyncWave as a discrete-event simulation and run four scenarios using a UDGM channel ($r_{\text{tx}} = 40$ m, $r_{\text{intf}} = 80$ m, loss-free). Simulation parameters match the testbed except $\hat{c} = 4{,}000$ μs (Table 2). The metric $\rho_s = \delta_{\text{sync}}/\tau_{\text{loop}}$ is sampled every 500 ms; TTS is the first sample where max pairwise error < 21 ms. Table 3 summarises all four scenarios.

Dense and linear. Both reach $\delta_{\text{sync}} = 0$ μs at the first post-TTS sample (Table 3). With no simulator clock drift and exact delay compensation, the epoch+MTS rule drives every node to the same integer time. On real hardware, drift and delay-estimation error reintroduce the 488 μs single-hop error of Section 5.1. Post-sync backoff cuts the rate to fewer than one broadcast per 60 s.

Partition-merge. Two clusters of 10 nodes run independently for 30 s; when a bridge node connects them at $t = 30$ s, the offset falls below the 21 ms threshold at the first post-bridge sample, and $T_{\text{merge}} = 500$ ms confirms the $O(D' \log n)$ bound (one $I_{\min}$ window).

Cognitive-loop margin. For $\tau_{\text{loop}} \in \{10, 20, 50, 100, 200\}$ ms, $\rho_s = \delta_{\text{sync}}/\tau_{\text{loop}} = 0$ across all periods in the dense scenario. Even using the conservative partition-merge error ($\delta_{\text{sync}} = 4{,}840$ μs) as the worst case, $\rho_s < 0.10$ holds for $\tau_{\text{loop}} \geq 50$ ms and $\rho_s = 0.048$ for $\tau_{\text{loop}} = 100$ ms, well within the cognitive loop budget in simulation and at one hop on hardware; multi-hop hardware validation awaits timestamping below the testbed's measurement floor. Extension to mobility models and energy measurement is ongoing.

6 CONCLUSION

We presented SyncWave, a fully decentralised PCO-based time synchronisation protocol extending the firefly-gossip paradigm with epoch-based maximum time synchronisation and exponential backoff for the synchronisation of embedded cognitive inference loops. On real 802.15.4 hardware it achieves 488 μs single-hop pairwise accuracy and, on 161 nodes over 7 hops, 2 s TTS; Simulations show TTS ≤ 1 s and $T_{\text{merge}} = 500$ ms after a partition-merge, all without any reference node or infrastructure.

ACKNOWLEDGMENTS

The authors thank the FIT IoT-LAB team for testbed access and Luca Seimon Mehl for his foundational work on the SyncWave protocol.

REFERENCES

[1] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frédéric Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. 2015. FIT IoT-LAB: A Large Scale Infrastructure for the Internet of Things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 459–464. https://doi.org/10.1109/WF-IoT.2015.7389098

[2] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, Urmish Thakker, Antonio Torrini, Peter Warden, Jay Cordaro, Giuseppe Di Guglielmo, Javier Duarte, Stephen Gibellini, Videet Parekh, Honson Tran, Nhan Tran, Niu Wenxu, and Xu Xuesong. 2021. MLPerf Tiny Benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*. arXiv:2106.07597

[3] Agata Barcis and Christian Bettstetter. 2020. Sandsbots: Robots That Sync and Swarm. *IEEE Access* 8 (2020), 218752–218764. https://doi.org/10.1109/ACCESS.2020.3041543

[4] Guenther Brandner, Udo Schilcher, and Christian Bettstetter. 2016. Firefly Synchronization with Phase Rate Equalization and its Experimental Analysis in Wireless Systems. *Computer Networks* 97 (2016), 74–85. https://doi.org/10.1016/j.comnet.2015.12.011

[5] Michael Breza and Julie A. McCann. 2017. Polite Broadcast Gossip for IoT Configuration Management. In *3rd International Workshop on Sensors and Smart Cities*.

[6] Chieh-Tung Cheng, Mustafa Aslanov, and Eiman Kanjo. 2026. Tiny Collaborative Inference for Occlusion-Robust Object Detection. *arXiv preprint* (2026). arXiv:2606.02894

[7] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient Network Flooding and Time Synchronization with Glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 73–84. https://doi.org/10.1145/1993581.1993592

[8] Alexander Gräfe, Fabian Mager, Marco Zimmerling, and Sebastian Trimpe. 2025. RockNet: Distributed Learning on Ultra-Low-Power Devices. *ACM Transactions on Cyber-Physical Systems* (2025). https://doi.org/10.1145/3773282 arXiv:2510.13320.

[9] Baran Can Gül, Stefanos Tziampazis, Nasser Jazdi, and Michael Weyrich. 2025. SyncFed: Time-Aware Federated Learning through Explicit Timestamping and Synchronization. *arXiv preprint* (2025). arXiv:2506.09660

[10] Mohit Gulati, Koen Zandberg, Zhen Huang, Gerhard Wunder, Cedric Adjih, and Emmanuel Baccelli. 2024. TDMiL: Tiny Distributed Machine Learning for Microcontroller-Based Interconnected Devices. *IEEE Access* 12 (2024), 167810–167826. https://doi.org/10.1109/ACCESS.2024.3492921

[11] IEEE. 2019. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (IEEE 1588-2019). https://doi.org/10.1109/IEEESTD.2020.9120376

[12] Johannes Klinglmayr and Christian Bettstetter. 2012. Self-Organizing Synchronization with Inhibitory-Coupled Oscillators: Convergence and Robustness. *ACM Transactions on Autonomous and Adaptive Systems* 7, 3 (2012), 30:1–30:28. https://doi.org/10.1145/2348750.2348757

[13] Johannes Klinglmayr, Christian Bettstetter, Marc Timme, and Christoph Kirst. 2017. Convergence of Self-Organizing Pulse-Coupled Oscillator Synchronization in Dynamic Networks. *IEEE Trans. Automat. Control* 62, 4 (2017), 1606–1619. https://doi.org/10.1109/TAC.2016.2594094

[14] Leslie Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM* 21, 7 (1978), 558–565. https://doi.org/10.1145/359545.359563

[15] Philip Levis, Thomas Clausen, Jonathan Hui, Omprakash Gnawali, and JeongGil Ko. 2011. *The Trickle Algorithm.* RFC 6206. IETF. https://doi.org/10.17487/RFC6206

[16] Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. 2021. MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. arXiv:2110.15352

[17] Wasif Masood, Johannes Klinglmayr, and Christian Bettstetter. 2014. Experimental Evaluation of Pulse-Coupled Oscillator Synchronization in IEEE 802.15.4 Networks. In *Proceedings of the 4th ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications (DIVANet), co-located with MSWiM*. 145–151. https://doi.org/10.1145/2656346.2656360

[18] David L. Mills, Jim Martin, Jack Burbank, and William Kasch. 2010. *Network Time Protocol Version 4: Protocol and Algorithms Specification.* RFC 5905. IETF. https://doi.org/10.17487/RFC5905

[19] Renato E. Mirollo and Steven H. Strogatz. 1990. Synchronization of Pulse-Coupled Biological Oscillators. *SIAM J. Appl. Math.* 50, 6 (1990), 1645–1662. https://doi.org/10.1137/0150098

[20] Kevin P. O'Keeffe, Hyunsuk Hong, and Steven H. Strogatz. 2017. Oscillators that Sync and Swarm. *Nature Communications* 8 (2017), 1504. https://doi.org/10.1038/s41467-017-01190-3

[21] Partha Pratim Ray. 2022. A Review on TinyML: State-of-the-Art and Prospects. *Journal of King Saud University – Computer and Information Sciences* 34, 4 (2022), 1595–1623. https://doi.org/10.1016/j.jksuci.2021.11.019

[22] Maurício Darabas Ronzani, Italo Firmino Da Silva, Jim Lau, Roberto Rodrigues Filho, Fabrício Ourique, and Alison R Panisson. 2026. Towards a BDI Architecture for Cooperative Agents on Resource-Constrained Microcontrollers.. In *ICAART (1)*. 294–301.

[23] Stuart Russell and Peter Norvig. 2010. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

[24] Jörg F. Schmidt, Udo Schilcher, Alexander Vogell, and Christian Bettstetter. 2023. Using Randomization in Self-Organized Synchronization for Wireless Networks. *ACM Transactions on Autonomous and Adaptive Systems* 18, 3 (2023), 9:1–9:27. https://doi.org/10.1145/3607193

[25] Mrinal V. Shenoy and K. R. Anupama. 2018. Swarm-Sync: A Distributed Global Time Synchronization Framework for Swarm Robotic Systems. *Pervasive and Mobile Computing* 44 (2018), 1–30. https://doi.org/10.1016/j.pmcj.2017.10.007

[26] Alexander Vogell, Udo Schilcher, and Christian Bettstetter. 2020. Deadlocks in the Synchronization of Pulse-Coupled Oscillators on Star Graphs. *Physical Review E* 102, 6 (2020), 062211. https://doi.org/10.1103/PhysRevE.102.062211

[27] Alexander Vogell, Udo Schilcher, Jörg F. Schmidt, and Christian Bettstetter. 2024. Chimera States in Pulse-Coupled Oscillator Systems. *Physical Review E* 110, 5 (2024), 054214. https://doi.org/10.1103/PhysRevE.110.054214

[28] Zhaowei Wang, Peng Zeng, Mingtuo Zhou, Dong Li, and Jintao Wang. 2017. Cluster-Based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks. *Sensors* 17, 1 (2017), 141. https://doi.org/10.3390/s17010141

[29] Matt Webster, Michael Breza, Clare Dixon, Michael Fisher, and Julie A. McCann. 2020. Exploring the Effects of Environmental Conditions and Design Choices on IoT Systems Using Formal Methods. *Journal of Computational Science* 47 (2020), 101231. https://doi.org/10.1016/j.jocs.2020.101231

[30] Geoffrey Werner-Allen, Geetika Tewari, Ankit Patel, Matt Welsh, and Radhika Nagpal. 2005. Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects. In *Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*. 142–153. https://doi.org/10.1145/1098918.1098934