Resource-Aware Raft for UAV Swarms via Online Q-Learning: A Systems-Level Adaptive Timeout Controller

Supplementary Material

Appendix X: Guarantees for Guardrailed Adaptation

Setting and Notation.: We consider the leader-coordinated controller (Secs. III–IV) that adapts Raft timers $a_t = (T_e, T_h)$ on a finite lattice $\mathcal{A} = \mathcal{A}_e \times \mathcal{A}_h$ with $\mathcal{A}_e = \{200, 250, 300, 350, 400, 500\}$ ms and $\mathcal{A}_h = \{50, 75, 100, 125, 150\}$ ms. Guardrails follow Eq.(8): smoothing parameter $\lambda \in (0, 1]$, single-step cap $\kappa > 0$, minimum dwell $\Delta_{\min} > 0$, and follower jitter magnitude $j > 0$. We work under Assumptions **Asm. 1–Asm. 4** from Sec. III (crash faults only; bounded delay tails; eventual strong connectivity; bounded clock drift). Let $T_e^{\min} = 200$ ms and $T_e^{\max} = 500$ ms.

Theorem 1 (Bounded controller-induced election attempts)**.** Fix any node i and any time window of length $T > 0$. Under Eq.(8) with dwell $\Delta_{\min}$, the controller can perform at most $\lceil T/\Delta_{\min} \rceil$ timer updates at node i in that window. Moreover, with the step cap κ and the floor $T_e^{\min} > 0$, the number of updates that *reduce* T_e by an amount sufficient to shorten time-to-timeout is also bounded by $\lceil T/\Delta_{\min} \rceil$. Consequently, there exists a finite function $B_1(T)$ such that the *expected* number of elections *attributable to controller-driven timer reductions* initiated by node i in a connected period of length T is $\le B_1(T)$.

Proof. By Eq.(8), updates are separated by at least $\Delta_{\min}$, hence at most $\lceil T/\Delta_{\min} \rceil$ updates occur in a window of length T. Each update changes T_e by at most κ (step cap) and $T_e \ge T_e^{\min}$ (floor), so the number of successive reductions is finite. Elections can be *attempted* only after a timeout; controller-driven timeout shortening events are thus bounded by the number of reductions, which is $\le \lceil T/\Delta_{\min} \rceil$. Under **Asm. 1–Asm. 4** and during connected periods, the probability that a reduction precipitates an election is bounded above by 1, so the *expected* count is bounded by the same linear function, which we denote $B_1(T)$. □

Theorem 2 (Safety invariants preserved)**.** Under **Asm. 1** and the guardrails in Eq.(8), adapting only (T_e, T_h) within fixed bounds leaves Raft's state-machine safety, log matching, and leader completeness invariants intact.

Proof. Raft's safety invariants depend on RPC semantics (AppendEntries/RequestVote), log indices/terms, and commit rules, not on specific timeout values. The controller leaves RPC handlers and log rules *unchanged*, modifying only timers within the fixed lattice $\mathcal{A}$; thus the standard Raft safety arguments apply verbatim. Guardrails neither alter message contents nor ordering rules, only the *timing* of when elections/heartbeats are triggered, which does not invalidate the safety invariants. □

Theorem 3 (Finite expected rounds to elect a leader in a connected component)**.** Consider any interval over which the communication graph is strongly connected (**A3**) and message delays are bounded (**A2**). Suppose follower timeouts include independent jitter with a continuous density supported on $[a_t^{(e)} - j,\, a_t^{(e)} + j]$. Then, conditional on initiating an election in that interval, the probability of electing a unique leader in one round is lower bounded by a constant $p_0 > 0$ that does not depend on time. Consequently, the expected number of election rounds to elect a leader is at most $1/p_0 < \infty$.

Proof. With independent continuous jitter, the probability that two followers' timeouts are *exactly* equal is zero; thus the earliest expiration (the minimum) is unique with probability 1. Under strong connectivity and bounded delays, there is a strictly positive probability $q > 0$ that the earliest candidate collects a majority of votes in that round (this follows from positive lower bounds on link success and the bounded-delay RPC exchange completing within the round). Therefore, the success probability per round is lower bounded by $p_0 \triangleq q > 0$, and the number of rounds to elect a leader is stochastically dominated by a geometric random variable with parameter p_0, yielding finite expectation $1/p_0$. □ □

Remark (Millisecond quantization). If device timers are quantized (e.g., to 1 ms), the "unique minimum" event need not hold with probability 1. In practice, OS scheduling and network/processing noise introduce sub-ms variability. If desired, one can add an ϵ-jitter (e.g., uniform on $[-\epsilon, \epsilon]$ with a continuous density) to restore the continuous case. Alternatively, with purely discrete uniform jitter on $\{-j, \dots, +j\}$ ms, the probability of exact ties remains bounded away from 1 for typical j; the success probability per round still admits a constant lower bound $p_0 > 0$ in connected intervals, so the finite-expectation conclusion stands.

Corollary (Non-explosive controller behavior). *Over any connected interval of length T, the expected number of controller-induced election* attempts *at a node is $\le B_1(T)$ (Theorem 1), and the expected number of* rounds *required to elect a leader is finite (Theorem 3). Hence, the controller* *is non-explosive: it cannot trigger an unbounded cascade of elections within finite time. Safety is preserved (Theorem 2).*

Scope: The bounds above concern *controller-induced* attempts; network faults can still precipitate elections, but **A2–A3** ensure that connected intervals occur and enable the liveness statement. These results align with the empirical stability diagnostics and reduced election activity reported in the main text.

I. Additional Experimental Figures (Experimental Setup)

This appendix supplements Sec. V with compact visual diagnostics. All figures use the bins/actions and guardrails defined in Secs. III–IV; each plot aggregates 25 seeds unless noted. Filenames correspond to the artifact package.

A. Election Timeout Usage

What it shows. Fig. 1 summarizes how often the controller selects each election-timeout lattice value $T_e \in \{200, \dots, 500\}$ ms across S1–S5 and swarm sizes N.
Why it matters. T_e governs leader turnover; longer timeouts reduce split votes but may delay failover.
Key observation. Under harsher RTT/loss (S3) and dense contention (S4), usage shifts toward the upper half of the lattice, reflecting the guardrails' bias for stability under Eq. 8; in stable S1, the mass centers near the nominal baseline.

B. Heartbeat Interval Usage

What it shows. Fig. 2 reports the distribution of heartbeat intervals $T_h \in \{50, 75, 100, 125, 150\}$ ms.
Why it matters. T_h trades commit responsiveness against control overhead and energy.
Key observation. The controller favors shorter T_h in mobile/harsh regimes (S2/S3/S5) to accelerate detection and recovery, while relaxing to longer T_h in S1 to curb overhead, consistent with the composite cost function in Sec. III.

C. Step-Cost Time Series

What it shows. Fig. 3 plots the windowed step-cost c_t over time for representative runs (S3, S5).
Why it matters. Decreasing level/variance indicates convergence to lower-latency, stable operation.
Key observation. Early spikes are damped by smoothing, bounded steps, and dwell (Eq. 8); as learning proceeds, c_t trends downward with fewer excursions, consistent with the cost-minimizing Q-backup in Eq. 7.

D. Exploration Schedule

What it shows. Fig. 4 tracks ε_t during training/deployment.
Why it matters. Cautious, decaying exploration reduces timer churn in steady regimes.
Key observation. ε_t decays toward $\varepsilon_{\min}$ with brief holds when the instability gate is closed, matching the "update when stable" logic in Algorithm 1.

E. Control-Message Mix

What it shows. Fig. 5 decomposes control-plane rate into AppendEntries (incl. heartbeats) and RequestVote.
Why it matters. Message composition explains overhead/energy trends in Sec. VII-C.
Key observation. Adaptive timers suppress vote bursts (fewer split elections) and avoid redundant heartbeats in dense settings, aligning with observed reductions in M and E.

F. Seed-to-Seed Reproducibility (S3)

What it shows. Fig. 6 scatters per-seed medians (L, R) in S3 with 95% CI ellipses.
Why it matters. Tight clustering supports robustness claims beyond single-seed artifacts.
Key observation. Points cluster without heavy tails; paired nonparametric tests in Sec. VII confirm improvements vs. baselines.

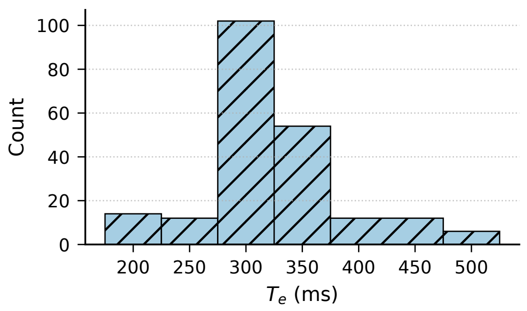

Fig. 1. **Election-timeout lattice usage.** Empirical frequency of $T_e \in \{200, \dots, 500\}$ ms across scenarios S1–S5 and $N \in \{10, 20, 40, 70, 100\}$. Concentration shifts toward longer T_e under higher RTT/loss (S3) and dense contention (S4), consistent with the controller's latency–stability trade-off.

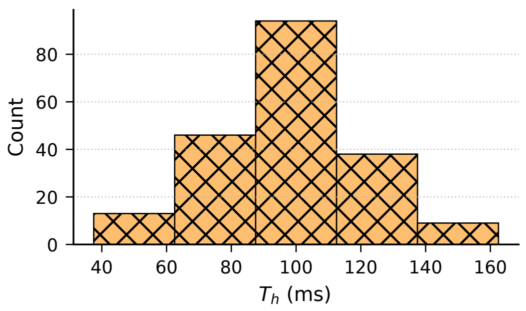

Fig. 2. **Heartbeat-interval lattice usage.** Distribution of $T_h \in \{50, 75, 100, 125, 150\}$ ms by scenario. The controller prefers shorter T_h in mobile/harsh regimes (S2/S3/S5) to reduce recovery latency, and relaxes to longer T_h in stable S1 to save overhead/energy.

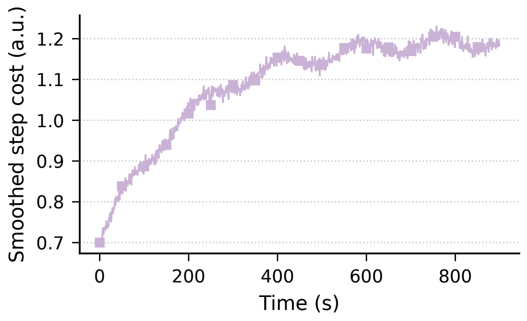

Fig. 3. **Windowed step cost c_t over time** (representative runs in S3 and S5). Guardrails (Eq. 8) damp spikes early; as learning proceeds the median and variance of c_t decrease, reflecting lower latency/recovery and fewer split votes.

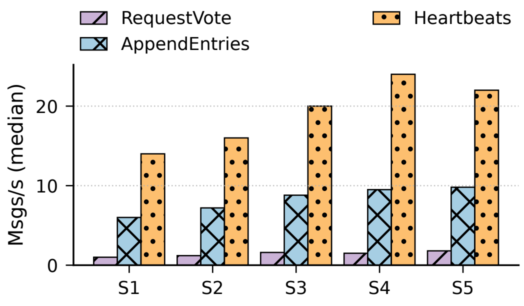

Fig. 5. **Control-plane message composition.** Per-node rates of AppendEntries (incl. heartbeats) and RequestVote by scenario and method. Adaptive timers reduce vote traffic and avoid redundant heartbeats, explaining overhead and energy gains (Sec. VII-C).

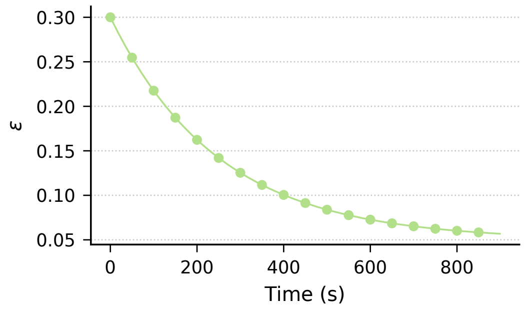

Fig. 4. ε**-greedy schedule.** ε_t decays to $\varepsilon_{\min}$ with occasional holds during instability (gate closed). This limits unnecessary timer churn once conditions stabilize, aligning with the safety gate in Algorithm 1.

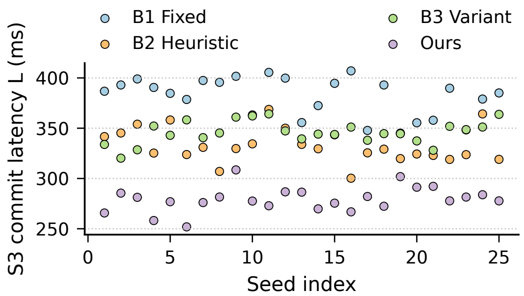

Fig. 6. **Reproducibility across seeds (S3).** Each point is a seed's (L, R) median; ellipses show 95% CIs. Tight clustering indicates robust behavior under harsh conditions; paired nonparametric tests in Sec. VII, confirm significance vs. baselines.

G. Simulator vs. HIL Microbench

What it shows. Fig. 7 compares simulator estimates to controller-in-the-loop (HIL) measurements for per-tick wall time, CPU %, RAM MB, and run energy on a Jetson-class SoC.
Why it matters. Validates that the controller's compute/energy footprint transfers to embedded hardware.
Key observation. Points lie close to the diagonal with modest scatter; discrepancies are discussed in Sec. VII-I as external-validity limits of the simulator.

H. State-Coverage Heatmap (S5)

What it shows. Fig. 8 visualizes visitation frequency over discretized telemetry bins in mixed stress S5.
Why it matters. Adequate coverage supports the use of a finite action lattice and tabular $Q(s,a)$.
Key observation. Coverage is broad without pathological sparsity; frequently visited cells align with mobility/loss regimes in Table III, indicating that the controller observes enough diversity to learn stable timer choices.

I. Reproducibility Notes

All figures here are derived from the same configuration space used in Sec. V: controller tick Δ_{ctrl}, feedback window W, action lattices for (T_e, T_h), and bins $\phi(\cdot)$. We fix random seeds per scenario and provide plotting scripts and YAML configs in the artifact to regenerate Figs. 1–8.

J. Methods and Reproducibility (Details)

All experimental results (Secs. V–VII) are generated with a fixed configuration that we summarize here for completeness. Mobility uses a $1\times1\,\text{km}^2$ area with altitude $[80, 150]\,\text{m}$; Gauss–Markov $\alpha{=}0.85$ with $\sigma_\theta{=}10^\circ$ and scenario-specific speeds (S1: 5 m/s; S2: 10 m/s; S3: 15–20 m/s; S4: 10 m/s; S5: mixed); Random Waypoint runs use a 3 s pause. Connectivity is a disk graph with radius $r{=}300\,\text{m}$. Delay τ is log-normal with per-scenario medians/shapes (S1: 25 ms/0.35; S2: 40 ms/0.50; S3: 80 ms/0.70; S4: 60 ms/0.60; S5: drawn per run), and loss follows a Gilbert–Elliott process tuned to Table III (e.g., S3:

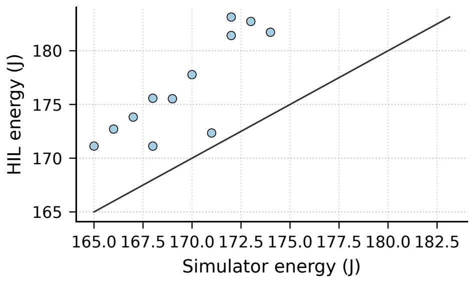

Fig. 7. **Simulator estimates vs. HIL measurements.** Scatter comparing per-tick wall time, CPU %, RAM MB, and run energy on a Jetson-class SoC (HIL) against simulator estimates under matched traces. Close-to-diagonal points indicate the compute/energy footprint modeled in Sec. V transfers to embedded hardware.

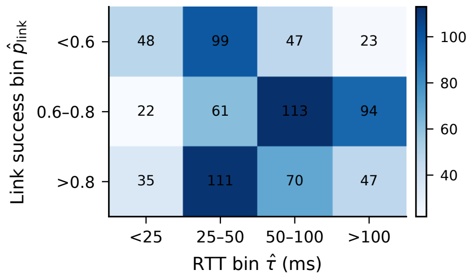

Fig. 8. **State coverage in mixed stress (S5).** Heatmap of visitation frequency over discretized telemetry bins (rows) and scenarios/windows (columns). Broad coverage without pathological sparsity supports the use of a finite action lattice and tabular $Q(s, a)$ for online adaptation.

p_G=0.005, p_B=0.15, $P_{G\to B}$=0.10, $P_{B\to G}$=0.30). Traffic is Poisson appends at $\lambda \in \{5, 10, 20\}$ entries/s (256 B payload), leader batching b=4; RequestVote is 96 B and AppendEntries control is 128 B+payload. Failures: leader crashes every 120 s (down 10 s) and each follower has 5% crash chance per 900 s run; clocks drift ±100 ppm, timers are ms-granular. Telemetry uses a 3 s success window and EMA smoothing (β_{EMA}=0.3); bins for $\hat{\tau}$, $\hat{p}_{\text{link}}$, CPU%, memory, and SOC match Sec. III. Guardrails (Sec. IV) use λ=0.5, κ=100 ms, $\Delta_{\min}$=5 s, follower jitter j=20 ms, and a neighborhood of ±1 lattice step per knob; instability I_t is elections/min over W_I=60 s, with safe-mode freeze if I_t>6/min or $\text{Var}(a_t)$>(75 ms)^2. RL hyperparameters: γ_{RL}=0.9, η_0=0.2 decaying as $1/(1+0.005t)$, ε_0=0.30 decaying by 0.993 to $\varepsilon_{\min}$=0.05; warm start Q_0 uses T_e=$k_e\hat{\tau}$, T_h=$k_h\hat{\tau}$ with (k_e, k_h)=(3.5, 1.5) (clamped). Energy integrates radio and CPU power: P_{tx}=1.2 W, P_{rx}=0.9 W, $P_{\text{cpu,active}}$=2.0 W, $P_{\text{cpu,idle}}$=1.2 W; airtime uses R_{phy}=12 Mbps and MTU 1200 B; residual SOC is against a 60 Wh budget. Protocol: 60 s burn-in is excluded; we report per-seed medians and across-seed median with 95% BCa CI (B=2000), do

TABLE I
ADDITIONAL PARAMETERS FOR REPRODUCIBILITY (SETUP AND CONTROLLER).

Component	**Value / Setting**
Area / Altitude	$1 \times 1\,\text{km}^2$; $[80, 150]$ m
Mobility	Gauss–Markov α=0.85, σ_θ=10°; speeds per S1–S5
Connectivity	Disk radius r=300 m
RTT model	Log-normal; S1: (25 ms, 0.35), S2: (40 ms, 0.50), S3: (80 ms, 0.70), S4: (60 ms, 0.60)
Loss model	Gilbert–Elliott; e.g., S3: p_G=0.005, p_B=0.15, $P_{G\to B}$=0.10, $P_{B\to G}$=0.30
PHY rate	R_{phy}=12 Mbps; MTU 1200 B
Workload	$\lambda \in \{5, 10, 20\}$ entries/s; entry 256 B; batch b=4
Failures	Leader: 120 s period, 10 s down; follower: 5%/run
Telemetry EMA	β_{EMA}=0.3; link window 3 s
Bins	CPU: $< 30/30$–$60/ > 60\%$; Mem: $< 512/512$–$1024/ > 1024$ MB; SOC: $> 60/30$–$60/ < 30\%$
Guardrails	λ=0.5, κ=100 ms, $\Delta_{\min}$=5 s, jitter j=20 ms
Neighborhood	±1 lattice step per knob (diagonals allowed)
Instability	I_t over W_I=60 s; freeze if I_t>6/min or $\text{Var}(a_t)$>(75 ms)^2
RL	γ=0.9, η_0=0.2 decays $1/(1 + 0.005t)$, ε_0=0.30 → 0.05
Energy model	P_{tx}=1.2W, P_{rx}=0.9W, $P_{\text{cpu,act}}$=2.0W, $P_{\text{cpu,idle}}$=1.2W
Run protocol	Burn-in 60 s; BCa B=2000; Wilcoxon+Holm; 25 seeds
ns-3	v3.39; 802.11ax HE 20 MHz; YansWifiPhy; shadowing on; 50 Hz traces
HIL	Jetson-class ARM64; perf governor; `tegrastats` 10 Hz; ext. power meter 5 Hz

paired Wilcoxon with Holm correction, and run 25 fixed seeds per scenario. Realism slice (Sec.V-H): ns-3 v3.39 (802.11ax HE 20 MHz, YansWifiPhy, LogNormalShadowing, 50 Hz traces) and a controller-in-the-loop HIL on a Jetson-class ARM64 SoC (Ubuntu 20.04/22.04, Python 3.9.21; performance governor; `tegrastats` at 10 Hz; optional external USB power meter at 5 Hz); control tick Δ_{ctrl}=1 s with ms timer resolution. Software: SimPy 4.1.1, NetworkX 3.2.1, NumPy/Pandas/Matplotlib pinned in the artifact. Additional parameter values and instrumentation are enumerated in Table I; figure-level diagnostics appear in Appendix I.

K. *Measurement Definitions*

Commit latency (L). Time from the leader receiving an append to receipt of the majority-commit acknowledgment; excludes client think-time and includes exactly one retransmission cycle if triggered.

Fault recovery (R). Time from an injected leader crash to the first new committed entry under the new leader; application appends are paused for 3 s after a crash to avoid confounding spikes.

Overhead (M). Per-node rate of control RPCs (AppendEntries including heartbeats and RequestVote); data payload bytes are excluded; duplicate sends are counted once per attempt.

Instability (I). Elections per minute over a 60 s sliding window (and $\text{Var}(a_t)$ as secondary).

Energy (E). E_{tx}+E_{rx}+E_{cpu} integrated over the run; message energy uses bytes-on-air at 12 Mbps and modeled $P_{\text{tx/rx}}$, CPU energy uses measured per-tick controller time.

L. *Randomization and Run Protocol*

Each run uses a fixed *global seed* with independent RNG streams for mobility, channel loss, failure injection, application arrivals, and RL exploration; streams are replicated across methods to enable paired tests. Ties in $\arg\min_a Q(s,a)$ are broken by selecting the action with minimal $\|a - a_{t-1}\|_\infty$ (then lexicographic). Unless stated (ablation A2), the Q-table is reset per run; warm starts use $T_e{=}k_e\hat{\tau}$, $T_h{=}k_h\hat{\tau}$ with $(k_e, k_h){=}(3.5, 1.5)$ and clamping to the lattices. Numeric guards use float32 storage, clipping Q-updates to $[-10^6, 10^6]$, and skipping updates on undefined costs. Leader failures occur every 120 s (10 s down); follower failures are Bernoulli 0.05 once per run, non-overlapping with leader downtime. Application appends pause for 3 s after each leader crash.

M. Telemetry, Guardrails, and Neighborhood

Controller tick $\Delta_{\mathrm{ctrl}}{=}1$ s. The RTT proxy $\hat{\tau}$ is the median over a 5 s window with EMA smoothing $\beta_{\mathrm{EMA}}{=}0.3$. Link success $\hat{p}_{\mathrm{link}}$ uses a 3 s sliding success ratio (EMA 0.3). Guardrails (Eq. 8) use smoothing $\lambda{=}0.5$, step cap $\kappa{=}100$ ms, and dwell $\Delta_{\min}{=}5$ s; followers add uniform jitter $j{=}20$ ms to T_e. The action neighborhood $\mathcal{A}_{\mathrm{nbr}}$ permits ± 1 lattice step per knob (diagonals allowed) with boundary clipping.

N. Messaging and Batching

RequestVote is 96 B; AppendEntries control is 128 B plus log payload; a fixed 28 B header is added for link/IP. The leader batches up to $b{=}4$ entries per AppendEntries; heartbeats are zero-payload AppendEntries. Message counts (M) include retransmissions as distinct attempts; payload bytes are excluded by design.

O. Clocking and Timers

All timers use a monotonic clock with 1 ms resolution and bounded drift ($\pm$100 ppm). Election timers at followers are randomized with $j{=}20$ ms uniform jitter around the recommended T_e; heartbeat interval is set exactly to T_h at the leader.

P. ns-3 and HIL Details

ns-3. Version 3.39, 802.11ax HE 20 MHz, YansWifiPhy, Friis+LogNormalShadowing; traces recorded at 50 Hz and replayed into the simulator (no new RPC types).
HIL microbench. Controller-in-the-loop on a Jetson-class ARM64 SoC (Ubuntu 20.04/22.04, Python 3.9.21) with performance governor; metrics via `tegrastats` at 10 Hz; optional external USB power meter at 5 Hz. The Raft runtime runs locally with timer hooks; radio dynamics are injected from the same traces.

Q. Statistical Procedure and Artifacts

We discard the first 60 s as burn-in. For each metric we compute per-seed medians and then report the across-seed median with 95% BCa CIs ($B{=}2000$ resamples). Significance uses paired Wilcoxon tests vs. B1/B2/B3 with Holm correction; effect sizes are $r{=}Z/\sqrt{n}$. The artifact includes code (commit hash), environment YAML with pinned versions, the seeds list (1001–1025), and scripts to regenerate all figures and tables.
Seeds. We use 25 fixed seeds per scenario: `1001, 1002, ..., 1025`; identical seeds across methods enable paired tests.

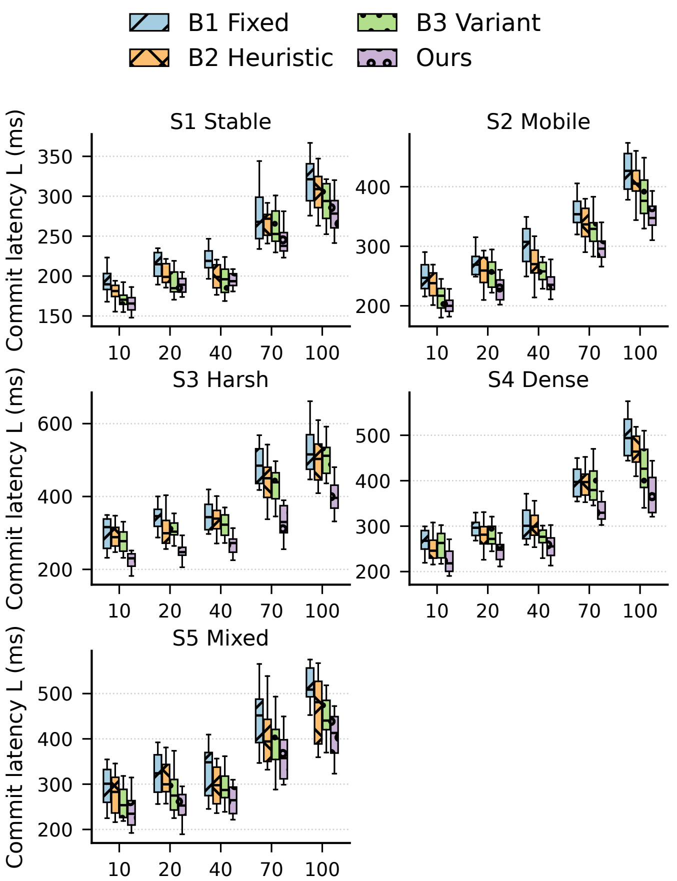

Fig. 9. Per-N commit latency L across S1–S5 (25 seeds per cell). Boxes show medians and 5–95% whiskers.

R. Per-Size Performance Breakdown

To complement the pooled results reported in Fig. 5 of the main paper and the scaling view in Fig. 9, we present per-size breakdowns for all scenarios (S1–S5) and swarm sizes $N \in 10, 20, 40, 70, 100$ across the compared methods (B1 Fixed, B2 Heuristic, B3 Variant, and Ours). Each cell aggregates results from 25 independent seeds. Boxplots indicate medians with 5–95% whiskers, with outliers omitted for clarity in print.
Commit latency. Figure 9 reports the per-N distributions of commit latency L. Across all scenarios and sizes, the adaptive controller shifts the median downward relative to B1 and narrows dispersion. Gains are modest in the stable regime (S1) at small N, and grow with mobility/loss and size: in S3 and S5 the gap widens monotonically with N (notably at $N \geq 70$), consistent with the controller's ability to stabilize leadership and tune heartbeats under stress.
Fault-recovery time. Figure 10 shows the per-N distributions of recovery time R (from leader failure to first new commit). Trends mirror latency: the controller yields shorter recovery windows in all scenarios, with the largest relative improvements in S3/S5 and at larger N. Heuristic variants (B2/B3) recover some benefit in S1/S2, but lag under harsher conditions due to over/under-reaction to rapid RTT/loss changes.

The per-size views corroborate the main claims: (i) improvements persist across all N and scenarios; (ii) relative gains are largest under combined stress (S5) and higher densities; and

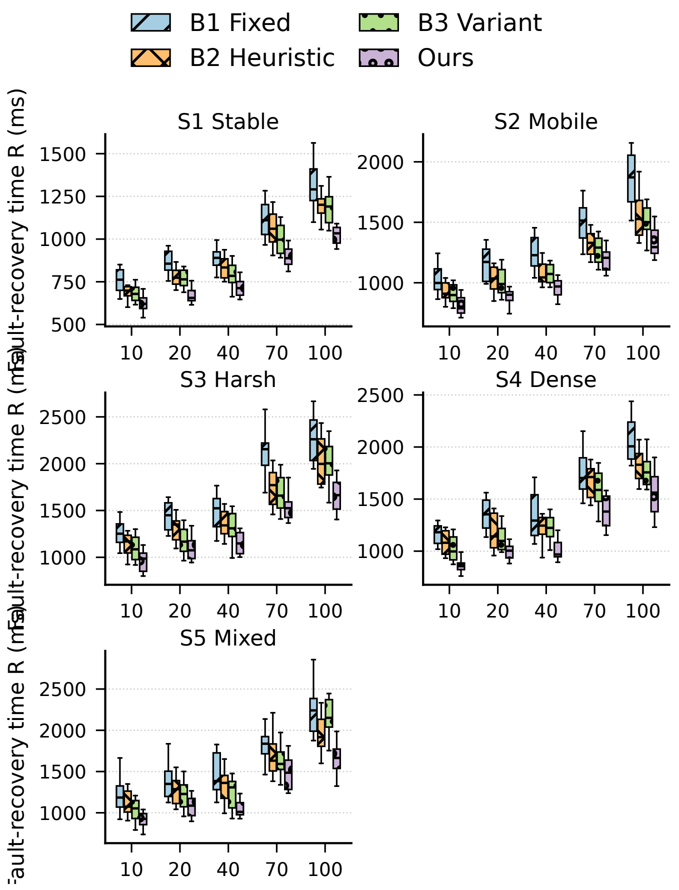

Fig. 10. Per-N fault-recovery time R across S1–S5 (25 seeds per cell). Boxes show medians and 5–95% whiskers.

(iii) dispersion reduces with the controller, indicating fewer split votes and more stable leadership.