POD 0007: Review Findings and Verification Evidence

COMMITTED

CATEGORY
Review Record

STATUS
Committed

AUTHORS
Vikrant Rathore <vikrant@insan.ai>, Paxos Odin Contributors

CREATED
2026-09-16

LAST UPDATED
2026-09-17

LABELS
review, verification, correctness, benchmark

Abstract

This record preserves five review passes from September 16 and a recovery-storage follow-up from September 17. The first two passes repaired correctness defects and revised the public API. The third introduced the data-oriented ledger and rotating ownership; the fourth and fifth examined window boundaries, admission and recovery reports. The sixth records bounded recovery storage and matched performance evidence. Counts and timings remain attached to the run that produced them; the latest evidence does not retroactively change an earlier result.

The review used `paxos-zig` to identify observable behaviours and verification scenarios, not as source code to translate. These records claim neither universal superiority nor machine-checked correctness or complete branch coverage.

First Pass: Correctness Findings and Repairs

The revised library retains Odin's procedural, bounded state-machine design. The important improvements are recovery correctness, configuration isolation, explicit diagnostics, and executable verification.

Priority	Finding in the original Odin code	Repair and evidence
High	A new campaign retained an old leader proposal, which could reject the value required by the new quorum.	Clear election/proposal state at the campaign boundary. Verified by `review_campaign_discards_prior_term_proposals`.
High	Moving between recovery chunks discarded previously observed chosen/trim boundaries.	Preserve those boundaries across chunks and retain recovered evidence. Verified by `review_recovery_preserves_fences_across_chunks`.
High	Snapshot recovery erased votes above the snapshot prefix. Such votes can belong to an already chosen quorum.	Preserve the durable suffix and promise; reset volatile leadership state. Verified by `review_snapshot_preserves_votes_above_anchor`.

High	Retry scans repeatedly started at the same slots. A six-slot recovery with two-slot chunks stalled after slot two.	Rotate the bounded scan through the physical window. Verified by `review_multichunk_recovery_and_retry_progress`.
High	A leader behind its followers skipped their decided slots during retry without fetching those decisions.	Fetch an ahead peer's chosen prefix during retransmission. Verified by `review_leader_fetches_decisions_from_ahead_follower` (three-node partial-crash seed 77 reproduced the stall).
High	A locally committed stop sign was not observed until a later `step`; restoring a committed stop could reopen the epoch.	Observe all decision-producing wrapper operations and durable restoration. Verified by `review_stop_seal_restore_and_completed_history`.
High	An out-of-order committed stop behind a gap disappeared from the pending-stop check.	Committed and accepted stop entries both block further proposals. Verified by `review_out_of_order_chosen_stop_blocks_proposals`.
High	Log envelopes did not carry an epoch, leaving delayed-message isolation entirely to the transport.	Add `Log_Envelope`, `log_envelope`, and a checked `log_step` overload. A three-node handover test rejects an actual queued old-epoch message. Legacy core envelopes remain available for already isolated transports.
Medium	Replaying a later slot could fail behind an accepted-only cell even when a durable trim certificate had closed the old slot.	Allow reuse covered by the certified trim. Verified by `review_replay_reuses_certified_trimmed_vote`.
Medium	A raw memory comparison treated string addresses and inactive union storage as value data.	Use native Odin equality and its comparable-type constraint. Payloads must remain immutable and valid while referenced. The `non_comparable_value` fixture in `tools/check_contracts.py` checks the constraint.
Medium	An optional proposal was extracted before testing whether it existed.	Use Odin's `(value, ok)` extraction and return `Missing_Proposed_Value`.
Medium	Slot arithmetic could wrap in restoration, range construction, or learner advancement.	Use checked/saturating bounds and terminate at exhaustion. Verified by `review_slot_exhaustion_and_learner_wrap`.
Medium	Negative quorum overrides silently became majorities; failed membership initialization partially changed the destination.	Validate a local candidate, then assign it. Zero alone requests the default majority.
Medium	Live trim adoption ignored a conflicting record with the same trim ID.	Reject conflicting identities and regressions before mutation. Verified by `review_live_trim_rejects_conflicting_identity`.

Medium	A learner could be marked campaign-enabled, and learner restoration ignored the trim anchor.	Preserve learner role restrictions and restore anchored frontiers.
Medium	Promise range bounds and equal-ballot conflicting values were insufficiently checked.	Validate the requested chunk and reject conflicting values. Verified by `review_promise_reordering_deduplication_and_validation`.
Medium	The simulator ignored errors, did not require missing decisions to converge, and did not crash during writes.	Fail on unexpected errors; simulate partial durable prefixes and pre-durable Accept delivery; independently observe durable voting quorums; require stable-period progress and convergence.
Medium	The CLI returned success after a failed compiler/test command; `make` could reuse a stale CLI.	Propagate failure and track CLI source prerequisites. `tools/check.py` tests failure with a deliberately failing compiler stand-in.
Low	Benchmark JSON was malformed because braces were interpreted by the format string parser.	Serialize a typed report with `core:encoding/json` and parse the output during checks.

Additional API completion: replicated-log learner decision, reconnection, and catch-up procedures now forward through the wrapper; learner decisions also update sealing state. Stop-sign initialization is safe when input slices alias the destination, and does not retain stale unused storage.

Odin Design Choices

- Keep ordinary procedures, parameterized structs, native `bit_set`, tagged unions, `Maybe`, slices, and explicit error returns. No class or method emulation was introduced.
- Use whole-struct initialization for nodes and learners so new fields receive deterministic defaults without additional reset loops.
- Use native optional/union extraction and `or_return` where they make the transition easier to follow.
- Keep the accept/commit/recovery transitions explicit. Shared helpers are limited to repeated invariants such as stop observation and bounded slot arithmetic.
- Preserve existing entry points and the caller-owned `Effects` model. Normal consensus transitions allocate no heap storage. Test harnesses and host simulations may allocate.
- Support positive non-power-of-two windows with modulo indexing. The 128-voter regression also exercises reuse of a three-slot window and acknowledgements across the 64-bit boundary. (Superseded in the third pass: the window must be a power of two and `review_hundred_twenty_eight_voters` runs on a four-slot window.)
- Retain the native voter bit-set bound of 128. Zig's larger configurable membership bound is not matched by this design. (Superseded in the third pass: `MAX_SUPPORTED_MEMBERS` is 65535.)

Elm-Style Error Contract

Every non-success `Error` has a descriptive title, a plain-language explanation, and a `Hint:` with a corrective action. The test enumerates the entire enum, so adding a code without a specific explanation fails verification. Capacity assertions and durability-order failures also contain corrective hints.

Hints name the actual Odin controls (`CHUNK_SLOTS`, `WINDOW_SLOTS`, `set_campaign_enabled`) and distinguish retryable backpressure from safety incidents. For example, a pending stop does not authorize handover,

an old-epoch message must not be relabeled, and failed writes must never be confirmed merely to clear the durability gate.

Call `explain_error(err)` at the host boundary. It returns a static string; formatting does not add allocation or I/O to consensus transitions. Include the operation, node, slot, and journal/message trace in host logs where available.

First-Pass Verification and Behavioral Comparison

Run the complete reproducible check from the repository root:

```
make check
# Broader deterministic matrix:
python3 tools/check.py --seeds=100 --steps=10000
```

The completed broad run of the first pass passed **45 tests in both debug and optimized builds**, **972 election cases within the test suite**, **nine compiler/durability contract checks**, and **300 simulations totaling 3,000,000 fault steps and 16,371 partial-write crashes**. The example, benchmark JSON, and CLI failure checks also passed on `odin dev-2026-09-nightly:a2fb372`. (The second pass raised the test count to 48 and the fixture inventory to eight compile-fail and eight durability runs; see below.)

The check includes debug and optimized tests, expected compiler failures, subprocess durability misuse tests, fault simulations, the counter example, benchmark JSON validation, and CLI failure propagation. Artifacts are built in a temporary directory so stale binaries cannot make a failed build appear to pass.

Area represented in Zig's library tests	Odin evidence
Ballot ordering, membership, flexible quorums	Existing protocol tests; 972 enumerated election scenarios across all six intersecting three-member quorum pairs and all six initial response orders
Highest-vote recovery, duplicate/reordered promises	Election matrix and targeted recovery regressions
Duplicate acknowledgements	A write-quorum-of-three test repeats one peer's acknowledgement five times and requires the third voter
Multi-Paxos proposals, batches, inherited-prefix gating	Existing consensus tests; atomic batch backpressure and inherited-prefix tests
Chunked recovery, ring reuse, full windows	Retry-progress tests, three-slot/128-voter test, and host-consumption simulation
Trim identities, replay, snapshot restoration	Regression tests for retained suffix votes, trim conflicts, anchored cell reuse, and overflow
Durability gates and safe pre-durable messages	Debug/optimized subprocess misuse tests and partial-write crash simulation
Catch-up, reconnect, heartbeat, Nack	Log wrapper tests, Nack test, recovery tests, and stable-period simulation
Learner ordering, duplicates, conflicts, window wrap	Existing learner tests and maximum-slot regression
Sealing, replayed stop signs, handover	Single-node and three-node reconfiguration tests, learner stop tests, pending/replaced stop tests

Host-certified configuration isolation	Checked epoch envelope test rejects a delayed prior-configuration message without writes or messages
Compiler rejection of invalid capacities	`tools/check_contracts.py`

The simulation records values chosen by durable vote quorums, even if no leader announces a commit. It models crashes before, during, and after the write prefix; pre-durable Accept transmission; drops, duplication, reordering, link partitions, restart, bounded window reuse, and host-served evicted history. A stable period must decide a fresh proposal and apply every known chosen entry on every node. This is finite executable evidence, not exhaustive exploration of all executions.

Host Contracts and Remaining Limits

1. Consume one transition's effects in order before advancing that node. Persist required writes, confirm actual durability, then send gated messages and expose decisions according to the host durability contract. The gate tracks an effects batch; it cannot prevent a host from deliberately bypassing the contract with another buffer or direct field mutation.
2. Use `Log_Envelope` across handovers, or provide equivalent authenticated epoch isolation in the transport. A bare `Envelope` carries no configuration identity.
3. Install only certified snapshots, with the matching durable application state and retained journal suffix. `begin_recovery` preserves durable suffix votes rather than erasing them. Restoring already trimmed stop history requires the host's persisted configuration metadata.
4. `is_leader_caught_up` reports prefix progress. It is **not** a lease, quorum read barrier, or authorization for linearizable reads during a partition. The library does not implement leader leases or Fast Paxos.
5. Payload equality must be deterministic and reflexive. Prefer fixed, self-contained commands or IDs. Reference-bearing comparable values borrow host storage; do not mutate or free it while protocol state or effects retain it. Float commands containing raw NaN representations are unsuitable for native equality without explicit host canonicalization.
6. Compile-time checks cover exercised initialization boundaries, not arbitrary manual construction or mutation of all public Odin structs. All peers must use compatible recovery chunk sizes. Batching remains coupled to `CHUNK_SLOTS`.
7. The sibling Zig project includes formal models, more compiler-rejection fixtures, reconfiguration simulations, and application/host integrations. Those were not copied or claimed as completed Odin verification. No refinement proof connecting this implementation to a formal model has been established, and line/branch coverage percentages were not measured.

Performance Measurement

First pass

`docs/review-benchmark.json` records the original Odin `HEAD` sources and revised sources with the same current benchmark driver, compiler, build flags, 32,768 values, five inner samples, and three alternating outer repetitions. The amortized cost stayed approximately 113-122 ns/value on that machine, with small differences in both directions. This supports retaining the correctness repairs without claiming a demonstrated speedup. These numbers are neither network/fsync latency nor a comparison against a compiled Zig library.

The local comparison used `../paxos-zig/src/{protocol,replicated_log,learner,bit_set,host_managed,errors,root}.zig`, its embedded test inventory, compiler/misuse fixture inventory, and simulator/verification layout. The Zig compiler was not available in `PATH`; no cross-language performance ranking is claimed.

Second pass

Benchmark on the second-pass machine (AMD Ryzen 7 5800H, Linux, ZFS journal, 131,072 values per in-memory mode, median of five samples): 140-145 ns per committed value in memory across the five modes; 26.8 ms per value with one fsync per commit round (six barriers per value) and 3.39 ms with group commit over eight values (0.75 barriers per value). The earlier book table that ranked this library against other implementations was removed because it had not been measured; a recorded four-way run replaced it (see the comparison below).

Four-way comparison on one machine

After a Zig 0.16 toolchain was found on the host, `tools/bench_compare.py` ran this library, paxos-zig 0.7.0, OmniPaxos 0.2.2, and LibPaxos3 sequentially and recorded `bench/results/latest.json` (AMD Ryzen 7 5800H with Radeon Graphics; the figures in this paragraph are the `0.1.0` run at commit `728b4b6` and are kept as history). Per committed value: three voters with 8-byte values 145 ns here against 114 ns for paxos-zig; five voters 185 ns against 208 ns; 1 KiB values 1,091 ns against 2,744 ns; OmniPaxos 1,121 ns one at a time and 86 ns with sixty-four in flight; LibPaxos3 2,265 ns. Durable modes: 27.15 ms and 3.58 ms here, 26.49 ms and 3.37 ms for paxos-zig. The book and README now read their benchmark tables from the recorded file.

Second Pass: API, Harness, and Documentation Review

This pass compared the library against `paxos-zig` 0.7.0 feature by feature (options, types, every public procedure, the error set, the simulator, the compile-fail and misuse fixtures, and the book's structure), then reworked the Odin surface for idiomatic use and brought every document back into agreement with the code. All claims below were verified by `make check`.

Parity with the Zig library

Area	Zig 0.7.0	Odin 0.1.0 after this pass
Messages, writes, effects, host requests	9 / 4 / 4 / 1	Same set and fields
Chunked recovery, fences, no-op filling, memory floor, trim anchors, term base gate	yes	yes
Flexible quorums	compile-time options	runtime overrides at `membership_init`, validated
Priorities, tick intervals, campaign disable	`Options`	`Node_Options` (zero means default)
Replicated log, stop signs, `pendingStopSign`, `initFromStop`	yes	yes, plus `Log_Envelope` and the checked `step`
Learner	yes	yes
Error explanations	42 (`Trimmed` unexplained)	every value of `Error` explained (42 besides `.None`); the test enumerates the enum
Simulator crash points	3	3 (`Crash_Point`)

Simulator actions	deliver, tick, propose (+batch), crash, restart, cut, heal, reconnected	same set
Simulator oracles	promise monotonic, decided monotonic, agreement, validity, convergence	agreement (at the vote level), validity, promise monotonic, vote-below-promise, contiguity, liveness probe, convergence
Reconfiguration simulation	3 scenarios x 16 seeds	3 scenarios x 16 seeds (`tests/test_reconfiguration_sim.odin`)
Compile-fail fixtures	14	8 (`tools/check_contracts.py`), covering every Odin-side contract
Misuse fixtures (both build modes)	3	4 (adds "zero value is ready")
Durable benchmark	fsync per value	`--durable`: fsync per commit round, sync and group-of-eight
Membership bound	65535	128 (native `bit_set`), by design; 65535 since the third pass
Formal model	yes	none; not claimed anywhere

API changes (breaking for the pre-release naming)

- `NodeId` becomes `Node_Id`; CamelCase error values become Ada_Case (`.Not_Leader`). `NodeId` stays as an alias (removed in `0.2.0`).
- `node_init_with_priority`, `node_restore_with_priority`, `node_restore_at` and their log counterparts are replaced by `Node_Options` and default parameters: `paxos.init(&node, id, membership, paxos.Node_Options{priority = 2})`, `paxos.restore(&node, id, membership, durable, floor)`.
- `Effects` now takes the same five parameters as `Node`; the compiler rejects a mismatch, and the capacities are the exact per-transition maxima (`2*CHUNK+1` writes, `MEMBERS*CHUNK + 2*MEMBERS + 1` messages, `WINDOW+1` committed). For the benchmark's 4096-slot window this shrinks a batch from about 1.5 MB to about 100 KB.
- The zero value of `Effects` is valid (`writes_pending` replaces `writes_confirmed`), so the enforced gate can no longer be bypassed by re-initialising a batch by accident; `effects_init` is the documented unchecked reset for abandoned batches.
- `Host_Managed_Node` is gone; `Node(..., .Host_Managed)` is the audited exception and the four host rules are on `Durability_Gate`.
- Proc groups cover both receivers: `campaign`, `propose`, `propose_batch`, `step`, `tick`, `reconnected`, `request_catch_up`, `learn_chosen`, `restore`, `continue_at`, `begin_recovery`, `advance_memory_floor`, `install_chosen_trim`, and every query.
- `replicated_log_init_from_stop(id, stop, stop_slot, anchor)` and `replicated_log_continue_at(id, configuration_id, membership, floor, anchor)` put the configuration id after the node id and the floor before the anchor everywhere.
- `Node` and `Learner` carry `where intrinsics.type_is_comparable(Value)`; the remembered no-op is `Maybe(Value)`; `stop_sign_validate_members` is public for host decoders.

Protocol changes

- A follower that receives a heartbeat above its promise now promises (a safe write) and observes the leader instead of ignoring it; previously it would time out and start a needless election after a dropped Prepare.
- Peer progress is learned from every message that carries it (prepare, promise range, learn, accepted, nack, heartbeat), matching the Zig core.
- Tick counters saturate instead of wrapping.
- `effects_reset` clears lengths only; an intermediate version zeroed the whole struct and cost a 75x slowdown, which the benchmark caught.

Verification run (second pass, 0.1.0)

`make check` on `odin dev-2026-09-nightly:a2fb372` at the end of the second pass: style (the Zen constraints, `-vet -strict-style` on every package), 48 tests in debug and optimized builds, 8 compile-fail and 8 durability fixtures, 60 simulations of 10,000 steps (1, 3, and 5 voters, 20 seeds each), the counter example, the benchmark JSON contract, and CLI failure propagation.

Documentation

The book was rewritten chapter by chapter against the final sources (foundations with the invariants and the greatest-vote proof, the single-decree trace with crash points, Multi-Paxos with chunked recovery, the library and its host contract, features, style, three worked systems, evidence with the real test inventory and the measured benchmark, a desk reference with the full error list and exercise answers, and a paper-to-code conformance appendix). POD 0002 and 0003 were corrected, POD 0004 stays a proposal, and POD 0005 (the API surface) and POD 0006 (reconfiguration and epoch isolation) were added. `README.md`, `README.ko.md`, and the Typst release notes describe the current API only; the API reference is Part VII of the book. This record replaces the former `docs/REVIEW.md`.

Third Pass: Ground-Up Data-Oriented Redesign

The third pass produced `0.2.0`. It kept the effect-machine contract, the error contract, the replicated log, the learner, and every test oracle, and replaced the core's data layout.

Motivation

A callgrind profile of the in-memory benchmark on the benchmark host, taken on the `0.1.0` sources, showed the 1 KiB workload dominated by two things: the `Message(Value)` union, whose every variant was padded to the size of a value-bearing message and copied whole at each hop, and the value copies themselves (into the proposal cell, each accept envelope, each acceptor cell, each write record, each commit envelope, and each committed entry). A phase-one or retransmission scan also loaded whole `Durable_Cell`s, values included, to read one ballot. None of this was visible in the 8-byte workloads, which is why the earlier passes had not seen it.

The new layout

POD 0009 records the design. In one paragraph: `Ledger(Value, WINDOW)` stores Lamport's `maxBal`, `maxVBal`, and `maxVal` as columns (`promised`, `promised_at`, `vote_ballot`, `value`) with a `state` byte and two bitmaps (`used`, `chosen`) over a power-of-two window; `Ballot` is one packed `u64`; `Node_Id` is `u16`; `Write_Vote`, `Write_Chosen`, `Committed`, and the three value-bearing messages carry `^Value` into the producing node's ledger, valid until that node's next transition; hosts copy at the journal and at the transport (`Journal_Record`, `Packet` in `tests/harness.odin`). `Trim_Anchor` lost its hash. The core is ten files (POD 0002). Rotating slot ownership (POD 0010) was added in the same pass as `Node_Options.rotating_ownership`, with `Prepare_Scope.Bounded`, `Write_Promise_At`, and `Nack_Message.slot` as its protocol footprint.

The ownership safety bug the simulator caught

The `0.1.0` rule let every `Accept_Message` leave before the sender's local barrier: an accept asks peers to persist a vote and claims nothing about the sender's own durability, and a restarted leader always campaigns at a fresh round. Under ownership an owner proposes at `ownership_ballot(owner) = ballot_make(0, 0, owner)`, which is the same ballot after every restart. With pre-durable delivery enabled in `sim/simulation.odin --ownership`, an owner could send its round-zero accept, crash at `Before_Writes` so its own `Write_Vote` was lost, restart, recompute `own_next` from a ledger that did not show the slot as used, and suggest a different value in the same slot under the same ballot. The vote oracle in `persist_sim_write` reported "ballot accepted two values in slot", which is a B1 violation per decree.

The fix is in `pre_durable_next`: only accepts whose `ballot_round` is above zero may leave before the barrier; a round-zero suggestion waits, because the owner's own vote is the only durable record that the instance was used. The reasoning is written on `Pre_Durable_Iterator` in `src/effects.odin` and in POD 0003. The simulator, not a unit test, found this; the ownership mode now runs in every `tools/check.py` invocation for that reason.

Fourth Pass: Adversarial Review of the Window and Ownership

An independent read of the 0.2.0 sources with throwaway tests found eight defects, all fixed in the library with a regression test each (`tests/test_window_review.odin`):

- **The live window was unbounded above.** `claim_live` tagged a free cell with any slot, so a lagging follower could hold slot $s + W$ in the cell of a live slot s; the pass-through in `record_commit` could then release two decisions through one shared value in a single transition (the record and the entry of the first carried the second's value). Fix: `claim_live` admits only slots in the interval `(memory_floor, memory_floor + WINDOW_SLOTS]`, and the pass-through runs at most once per transition. This also removed two ways an owner could be wedged (`.Window_Overrun` forever from a far-ahead cell; `own_next` below the floor after commits and a floor advance) and the case where a revoker's own bounded promise could fail after it had already changed role.
- **A revoked suggestion was lost when the owner itself voted the revoker's accept.** `on_accept` overwrote the owner's round-zero vote before `record_commit` could see it, and a revocation the owner started had cleared `lead_slot`. Fix: `on_accept` queues the resubmission at the overwrite, keyed on the vote's ballot alone.
- **An ownership tick could overflow `Effects.messages`** when resubmits, skips, and retransmissions to every peer landed in one batch. Fix: one chunk of proposals per tick; retransmission only on a quiet tick.
- **A stale duplicate `Accepted` returned `.Missing_Proposed_Value`** (an incident error) after its cell had been reused. Fix: stale acknowledgements are ignored.
- **A leader fenced by a peer that then died never re-ran phase one.** Fix: a leader whose inherited gap stalls for `election_timeout_ticks` campaigns again; a deposed leader waits a full timeout before campaigning; a candidate that loses a decree to a higher ballot in `resolve_chunk` steps down instead of surfacing `.Not_Leader` from `tick`.
- Minor: `ledger_replay_fold` rejects a `Write_Promise_At` for slot zero; `send_accept` reports `.Not_Leader` rather than silently returning a slot it did not propose; `propose_owned` steps over an own slot a revoker's promise reached first.

The duplicate-release defect and the best-effort resubmission defect required schedules the seeded simulator had not produced in its first hundred runs: a partition longer than the window, and an owner that hears the revoker's accept before its commit.

Fifth Pass: Batches, Quorum Combinations, and Recovery Reports

A third independent review reproduced four defects with an isolated program; each is fixed with a regression test in `tests/test_batch_review.odin`:

- **An ownership tick could overrun the writes buffer** with read quorum three, write quorum one, and chunk two: each skip decided at once (two writes), and the stall timeout started a revocation (one promise per slot) in the same transition. A revocation now gets a transition of its own: it starts only on a tick that proposed nothing.
- **Ownership batches stopped working once the floor advanced**: `node_propose_batch` computed occupancy from the single-leader `next_slot`, which ownership never moves, and the unsigned subtraction wrapped. Ownership admission now runs on the owner's own frontier before any single-leader arithmetic.
- **A rejected ownership batch could leave a vote behind**: `own_slots_available` estimated the batch's span arithmetically while the proposals stepped over revoked slots. It now probes every target slot through `own_slot_probe`, which is also what `next_usable_own_slot` uses, without mutating anything, and the batch is admitted whole or not at all.
- **Recovery reported a false conflict** when an acceptor outside the deciding quorum reported an older, losing vote after another had reported the decision. `on_promise` now lets only a second decision with a different value contradict a decision; both report orders are tested.

Two contract items from the same review: resubmission is documented as best effort and a dropped resubmission is counted (`resubmits_dropped`); and the membership is now sorted by `membership_init`, so ownership order is ascending id on every node whatever order the host listed the members in, which retires the rule that every host must pass the same order. The benchmark harness now fails on a transition error, on a queue overflow, and on any node that did not decide every value, so a dropped message cannot flatter a number.

Consolidated Historical Verification

`make check` runs `tools/check_style.py` (the Zen constraints of POD 0001: file, line, and procedure limits), `odin check -vet -strict-style` on `tests`, `sim`, `bench`, `cli`, and `examples/counter.odin`, then:

- 69 `@(test)` procedures in `-debug` and `-o:speed` (`grep -c "@(test)" tests/*.odin`), including the five `ownership_*` scenarios, `review_thousand_voters_reach_quorum`, the bit-set tests, and the `election_matrix_preserves_chosen_values` enumeration of 972 cases;
- 9 compile-fail fixtures (adds `window_not_power_of_two`) and 4 durability fixtures built in both profiles (8 runs) in `tools/check_contracts.py`;
- 120 seeded simulations of 10,000 steps: 1, 3, and 5 voters, 20 seeds each, in the single-leader mode and again in the ownership mode (60 + 60);
- the four reconfiguration scenarios over 16 seeds each (one under rotating ownership, where decisions above the seal are abandoned), the counter example, the benchmark JSON contract (eleven rows, now including the two `owned-3n` modes), and CLI failure propagation.

`INVARIANT_CHECKS` (`#config(PAXOS_INVARIANT_CHECKS, ODIN_DEBUG)`) compiles `node_assert_valid` into the debug test profile, where it checks the cell tags, the vote-below-promise rule, and the bitmap consistency after every lifecycle change.

Benchmark figures for 0.2.0

`make bench-compare` (`tools/bench_compare.py`) reran this library, `paxos-zig`, OmniPaxos, and LibPaxos3 sequentially on the `0.2.0` sources and rewrote `bench/results/latest.json` (recorded 2026-09-16T23:53:16Z on the same AMD Ryzen 7 5800H host, after the fifth-pass fixes); the `0.1.0` figures

quoted under "Four-way comparison" above survive only in that paragraph. Those figures belong to the September 16 harness. Current matched comparisons and profiles are recorded separately in POD 0009. Against the `0.1.0` run, the 1 KiB workload, where the copies were the cost, fell from 1,091 ns to 505 ns per value one at a time; the three-voter 8-byte workload, where the transition logic is, moved from 145 ns to 148 ns; five voters from 185 ns to 191 ns. The file also carries the two `owned-3n` rows for rotating ownership (161 ns and 154 ns) and the durable rows (27.49 ms and 3.71 ms). The window bound added in the fourth pass costs one compare per claim and is inside the noise of these rows. POD 0009 states what the redesign was expected to change and what it was not.

Sixth Pass: Recovery Storage and Matched Evidence

The implemented recovery scratch now scales with chunk capacity. Range checks precede narrowing, selection freezes before phase two, and retries retain it. Sparse retransmission wraps once without repeating cells. Campaigns and bounded revocations durably reserve their own ballot before sending Prepare. These fixes and their limits are recorded in POD 0009 and the book's measurement chapters.

`bench/results/recovery-validation.json` records 79 tests in debug and optimized builds, 720 seeded simulations of 10,000 steps, nine compile-failure fixtures, and four durability fixtures in both builds, with style and vet checks. The default check uses 240 simulations; the archived extended run uses 720.

`recovery-matched-20260917.json` contains 90 workload/implementation rows with nine samples each; the profiles and static memory CSVs are adjacent. Three-member, 1 KiB node-plus-effects storage fell from 633,120 to 433,176 bytes at W256/C64. Some workloads became faster, some remained inconclusive, and other libraries still lead some rows. These measurements establish no universal speed ranking.

References

- Odin language overview (`https://odin-lang.org/docs/overview/`): procedures, parameterized types, unions, equality, and native containers.
- Lamport, Leslie. "The Part-Time Parliament." ACM TOCS, 1998: agreement, durable records, and progress assumptions.
- Lamport, Leslie. "Paxos Made Simple." ACM SIGACT News, 2001: quorum intersection and preservation of previously chosen values.
- POD 0002, 0003, 0005, and 0006 for the specifications the repairs feed into.