POD 0004: Fast-Path Leader Leases and Linearizable Read Verification

DISCUSSION

CATEGORY	**STATUS**
Protocol Extension	Open for Discussion
AUTHORS	**CREATED**
Paxos Odin Contributors <team@insan.ai>	2026-09-16
LAST UPDATED	**LABELS**
2026-09-17	consensus, reads, leases, performance

Abstract

A lease could allow a leader to answer a read without a network exchange, provided it can prove that no competing write can complete during that read. This record remains **Open for Discussion and unimplemented**. The earlier tick-only sketch was insufficient: arbitrary host ticks supply neither a clock bound nor a safe expiry rule. The obligations below must be discharged before implementation.

Status and Implementation Boundary

The core has no lease and no freshness-guaranteed read API. `is_leader_caught_up` reports progress through an inherited prefix; it cannot prove current leadership. Local decided-value queries may return an old prefix. Heartbeats have no counted lease acknowledgement. `Node_Options` tick intervals control retries and elections, not real-time authority. POD 0011 must not infer a lease from these queries.

A host can design a read barrier through consensus and wait until the application has applied the required prefix. Its proof must state how concurrent writes and client acknowledgements relate to that barrier. Merely reading local state or observing a heartbeat supplies no such proof.

Terminology and Scope

A **linearizable read** returns a state consistent with some instant between its invocation and response, respecting operations that completed before invocation. A **grant** restricts a voter's future behaviour for an interval. A **lease** is the leader's evidence that enough restrictions remain valid to exclude conflicting progress for the entire read.

The proposed extension initially concerns single-leader Multi-Paxos. Clock drift, process suspension, restart, quorum intersection, and persistence are part of its correctness model, not details that can be delegated without a contract. The existing core's safety needs no timing bound; this extension would add one.

Required Design and Proof Obligations

Time and delayed replies

Specify a monotonic-clock model, drift bounds, and behaviour across process or machine suspension. A leader must derive a conservative deadline from the grant request's start and the model's bounds. Starting

a fresh full interval when a reply arrives is unsafe: the reply may have been delayed until the grant expired. Every read must check a deadline that remains valid through its linearization point. Counts of calls to `tick` alone cannot establish elapsed time.

Quorums and all competing transitions

Flexible Paxos requires read/write intersection. It does not require two read quorums to intersect. A claim that only one partition can hold a read quorum is therefore invalid. Choose grant quorums and restrictions, then prove that every competing write path intersects a live restriction, including a previously prepared leader issuing phase-two accepts. Fencing only new elections or global Prepare messages is insufficient; Accept, heartbeat adoption, bounded prepares and recovery need explicit treatment.

Crash recovery

A voter that forgets a live grant on restart can violate it immediately. Specify either durable grant recovery with a clock model that survives restart, or a conservative restart quarantine with proved bounds. A vague guard interval that "covers restart time" is not enough. A recovered leader must discard stale lease authority, and the resulting writes and barriers must conform to POD 0003.

Applied state and membership

An eligible leader must have applied the complete prefix required by the read, not merely learned its own write. Configuration changes must fence old leases before a new configuration can acknowledge conflicting work. The specification must identify the read's linearization point and its relation to application state, stop signs and client completion.

Rotating Ownership

Rotating ownership features several independent proposers and no persistent `Role.Leader`. A revoker returns to follower status after driving its assigned range. Neither a leader hint nor a caught-up prefix grants exclusive authority. This proposal does not specify an ownership lease; that would require a separate proof of the relevant writer restrictions.

Validation and Acceptance Gates

Model bounded clock drift, delayed grants, pauses, crash/restart and handover. Enumerate supported quorum combinations, including disjoint read quorums. Test reads against completed client histories, with faults immediately before deadline checks and responses. Add deterministic counterexamples for receipt-based expiry, forgotten grants and phase-two writes by a previously prepared competitor.

Tests supplement the proof; they cannot establish an unspecified clock model. Until these proof obligations are resolved, this record remains in discussion, and no public `can_serve_local_read` capability will be added to the core or Python SDK.

Alternatives Considered

A consensus barrier avoids importing lease clocks but adds communication latency. A host-specific lease can exploit a known environment but must publish its timing and durability assumptions.

Open Questions

Which environment and quorum family should the first proposal support? Can its assumptions be checked operationally? Which transition fences and restart policy provide a complete argument? These questions precede message layout and API naming.

References

- POD 0002: current architecture and effect boundary.
- POD 0003: durability and restart contracts.
- POD 0006: stop signs and epoch isolation.
- POD 0008: the current timing-independent safety argument.
- POD 0010: rotating slot ownership.