Cross-Shard Atomic Polaris Cert: Two-Phase Commit via Parent-L1 QuasarCert

Zach Kelling

Lux Industries

2026-06-03

Abstract

LP-204 organizes state into Tier-2 chain-VMs running on a parent L1. When a transaction group's state touches multiple chain-VMs, the wave commits at each shard are independent under LP-209/LP-210 – there is no intrinsic atomicity across shards. LP-211 supplies that atomicity via two-phase commit anchored at the parent L1's existing QuasarCert: a coordinator dispatches per-shard portions; shards prepare via LP-210 Phase-4 staged overlays; when all shards prepare, the parent L1's next wave commits a cross-shard commit transaction whose finality cert is the cross-shard cert. ALL shards commit OR NONE. End-to-end latency at PQ-strict LAN: ∼260 ms under the projected GPU-accelerated cert (LP-203 GB10 sm_120 pairing rates measured 2026-06-04); ∼1326 ms on the measured CPU-only cert (`/tmp/quasar-results-spark.txt` matrix bench Spark N=64 PQ-strict 595 ms per-cert wall-clock). This paper expands on LP-211 (`lps/LP-211-cross-shard-atomic.md`) with the five new wire schemas (`0xE2`–`0xE7`), the atomicity proof sketch, and the load-bearing composition with LP-202's monotonic cert-tier upgrade – which lets shards commit early at their individual mode without waiting for the parent L1's target.

Contents

1 Introduction

This paper expands on LP-211 (`lps/LP-211-cross-shard-atomic.md`) with the five new wire schemas, the atomicity proof sketch, the load-bearing composition with LP-202's monotonic cert-tier upgrade, and the latency projections at LAN/WAN × PQ-strict/PQ-heavy.

1.1 What was wrong before

LP-204 organizes state into Tier-2 chain-VMs running on a parent L1. Each chain-VM has its own LP-208 mempool, LP-209 wave ordering, and LP-210 Block-STM executor. Per-shard finality is independent: each shard publishes its own QuasarCert per wave.

For a transaction group whose state touches multiple chain-VMs, this shard-independence is a correctness problem. If shard A's wave commits but shard B's wave aborts, the cross-shard transaction is half-applied: A's state has moved, B's has not. The protocol needs cross-shard atomicity.

1.2 What this LP solves

Cross-shard 2PC via a parent-L1 QuasarCert. A coordinator (one of the parent L1's validators, selected deterministically per cross-shard tx group) dispatches per-shard portions to shard executors, collects prepare-acks (LP-210 Phase-4 staged-prepare), and when all shards have prepared, submits a cross-shard commit transaction to the parent L1. The parent L1's next wave (LP-209) includes that commit tx; the wave's LP-217 cert is the cross-shard cert. Once the parent L1's cert finalizes, every shard's prepared overlay is committed atomically.

If any shard refuses to prepare, the coordinator submits a cross-shard abort transaction instead; every prepared shard unwinds its staged overlay via LP-202 `txn.Discard()`. ALL shards commit OR NONE.

1.3 Why Polaris cert, not a sidechain mechanism

A naive cross-shard atomicity mechanism uses a sidechain or a relay validator set. LP-211 instead uses the *parent L1's existing QuasarCert* as the atomic-commit anchor. The parent L1 already has a validator set; LP-204's chain-VMs inherit the parent's security automatically. The parent L1's cert at LP-217's chosen mode is the strongest cryptographic statement available on the network – using it for cross-shard commit means cross-shard finality is exactly as strong as the parent L1's finality at that mode.

No new validator set. No new bridge contract. No new dispute window. The cert that the parent L1 already publishes per wave serves as the cross-shard commit anchor. The shards prepare; the parent L1 commits; the cert covers the commit.

This is the canonical pattern from LP-204 "Cross-VM atomic memory" elevated to a wave-scale formal protocol.

1.4 Latency and throughput dividends

Latency. Per-shard waves continue at the LP-209 cadence (∼100 ms LAN under the projected GPU-accelerated cert; ∼711 ms on the measured CPU-only cert per the 2026-06-04 quasar-matrix-bench Spark N=64 PQ-strict row, `/tmp/quasar-results-spark.txt`). The cross-shard cert adds one parent-L1 wave plus the LP-217 cert mode timing (∼5 ms at PQ-strict projected GPU; ∼595 ms measured CPU). Total cross-shard latency: ∼260 ms (projected GPU) or ∼1326 ms (measured CPU) at LAN scale at PQ-strict.

Throughput. Each shard's waves run independently except for the cross-shard txs, which by construction are a minority of the workload (in Liquidity-style traffic: $\sim 5\%$). The 95% intra-shard waves commit at the LP-210 single-shard rate; the 5% cross-shard waves pay the 2PC tax.

2 Algorithm

2.1 Step 0 – coordinator selection

A cross-shard tx group $T = \{T_1, \ldots, T_K\}$ where each T_j is the per-shard portion targeting shard s_j. The coordinator is a parent-L1 validator selected deterministically:

$$\text{coordinator}(T) = \text{parent\_L1.validators}\big[\,\text{hash}(T.\text{id}) \bmod N\,\big] \tag{1}$$

where $T.\text{id} = \text{sha256}(\text{canonical\_bytes}(T))$. The selection is a pure function of the tx group; every shard executor agrees on the coordinator without coordination.

2.2 Step 1 – dispatch

The coordinator gossips T via the parent L1's LP-201 DHT, schema `0xE3` (`CrossShardDispatch`). Each shard s_j whose NodeID is subscribed to the parent L1 receives T and dispatches T_j into its local mempool.

2.3 Step 2 – per-shard prepare

Each shard s_j executes T_j under LP-209/LP-210 normally:

1. T_j enters the shard's LP-208 DAG mempool.
2. The shard's LP-209 wave commits a wave containing T_j.
3. LP-210 Phase 4 detects that T_j is cross-shard (`T_j.txEnvelope.kind == cross_shard`) and stages the overlay instead of committing.
4. The shard publishes a *prepare-ack*: schema `0xE4` (`CrossShardPrepareAck`) containing $(T.\text{id}, s_j, T_j.\text{state\_diff\_hash}, \text{shard\_wave\_cert})$.

The `shard_wave_cert` is the per-shard LP-217 cert at the shard's configured mode – proof that the shard committed to executing T_j at the shard's per-wave consensus.

2.4 Step 3 – coordinator collect

The coordinator subscribes to prepare-acks on the parent L1's gossip layer. It collects acks until either:

- **All K acks arrive** within `prepare_timeout_ms` → emit cross-shard commit tx (Step 4).
- **Any shard times out or refuses** (returns refuse-ack schema `0xE5`) → emit cross-shard abort tx (Step 4').

`prepare_timeout_ms` defaults to $5\times$ the slowest shard's wave cadence (typical: 500 ms LAN, 2000 ms WAN). The timeout is conservative – under healthy operation, all acks arrive within $1\times$ wave cadence.

2.5 Step 4 – cross-shard commit tx at parent L1

The coordinator submits to the parent L1's LP-208 mempool:

```
CrossShardCommit {
    group_id:     [32]byte  T.id
    shards:       []ShardCommit  // per-shard:
                                 // (chain_id, state_diff_hash,
```

```
                                    //  wave_cert)
    coordinator:  NodeID
    coord_sig:    []byte  // BLS or PQ per parent L1 cert mode
}
```

The parent L1's next LP-209 wave includes this tx. The wave's LP-217 cert covers the cross-shard commit. Once the parent L1's cert finalizes:

1. Each shard observes the parent L1's cert (gossip schema `0xE6 CrossShardCommitNotify`).
2. Each shard verifies the cert covers T.id and that the cert's leg composition satisfies the shard's cert-mode policy.
3. Each shard applies its staged overlay via LP-202 atomic commit.

2.6 Step 4' – cross-shard abort tx at parent L1

If any shard refused or timed out, the coordinator submits:

```
CrossShardAbort {
    group_id:     [32]byte  T.id
    reason:       enum {timeout, shard_refused, ...}
    coordinator:  NodeID
    coord_sig:    []byte
}
```

The parent L1's next wave includes this tx; the cert covers the abort; each shard observes via gossip and unwinds its staged overlay via LP-202 `txn.Discard()`. The cross-shard tx group is dropped; per-shard waves move forward independently from the next round.

3 Wire Schemas

LP-211 introduces six new ZAP wire schemas, allocated in the LP-208 extension range (`0xE2`–`0xE7` = LP-208/209/210/211 cluster):

Schema	Hex	Purpose	Fields
`CrossShardTxGroup`	`0xE2`	the tx group bytes (canonical)	`group_id [32]byte, parts []CrossShardPart` where each part = `(chain_id uint32, tx_bytes []byte)`
`CrossShardDispatch`	`0xE3`	coordinator → shards via DHT (Step 1)	`group_id, coordinator NodeID, tx_group_bytes []byte`
`CrossShardPrepareAck`	`0xE4`	shard → coordinator (Step 2)	`group_id, chain_id uint32, state_diff_hash [32]byte, wave_cert []byte`
`CrossShardRefuseAck`	`0xE5`	shard → coordinator (Step 2 refusal)	`group_id, chain_id uint32, reason enum`
`CrossShardCommitNotify`	`0xE6`	parent L1 → shards (Step 4)	`group_id, parent_cert []byte`
`CrossShardAbortNotify`	`0xE7`	parent L1 → shards (Step 4')	`group_id, parent_cert []byte`

Table 1: LP-211 wire schemas at IDs `0xE2`–`0xE7`.

Schemas `0xE0`–`0xE1` are reserved for LP-208 DAG header/parent-list wire formats per LP-208's schema allocation. Schema `0xE0`–`0xEF` as a whole is allocated to the rollup-VM envelope range by LP-218, but with LP-208/LP-211 taking `0xE0`–`0xE7`, leaving `0xE8`–`0xEF` free for LP-218 rollup envelopes (`0xE0`–`0xE3` per LP-218 itself for `RollupBatchTx`, `RollupChallengeTx`, `RollupExitTx`, `RollupGenesisTx`). The schema allocation map at the umbrella (LP-200) reconciles the overlapping ranges.

3.1 No collisions with prior allocations

Range	Owner	LP
`0xC0`–`0xCB`	chains-VM wire (12 VMs)	LP-186
`0xD0`–`0xDF`	ZAP P2P transport	LP-201
`0xE0`–`0xE1`	DAG header/body	LP-208 (older draft alloc)
`0xE2`–`0xE7`	cross-shard wire	LP-211 (this LP)
`0xE8`–`0xEF`	rollup-VM envelopes	LP-218
`0xF0`–`0xFF`	DAG header/body (current)	LP-208

Table 2: Cross-LP schema allocation, reconciled at the umbrella LP-200.

4 Atomicity Proof Sketch

The atomicity claim: for any cross-shard tx group T, after the parent-L1 cert decisively covers either `CrossShardCommit`(T.id) or `CrossShardAbort`(T.id), every shard $s_j \in T$'s shards has either committed T_j or unwound T_j, and these outcomes match.

4.1 Argument

1. Each shard's LP-210 Phase 4 staging is local – the staged overlay exists only in the shard's runtime, content-addressed by the per-shard wave cert.
2. The shard's wave cert (Step 2 ack) commits the shard to executing T_j at the shard's per-wave finality. If the shard later refuses to apply the staged overlay on Step 4 commit-notify, the shard is byzantine and slashable on the parent L1 (Step 4 commit-notify is the cryptographic instruction; refusal contradicts the shard's own wave cert).
3. The parent-L1 cert is either commit or abort, never both – wave ordering at the parent L1 is total (LP-209), so the `CrossShardCommit`(T.id) or `CrossShardAbort`(T.id) tx commits at exactly one parent-L1 wave height with a single cert.
4. Therefore: the parent-L1 cert is the unique decision oracle. Every shard's local outcome is a deterministic function of the parent-L1 cert (commit $\rightarrow$ apply staged overlay; abort $\rightarrow$ LP-202 unwind). A shard that diverges is byzantine and slashable.

4.2 Eventually-cryptographic atomicity

The atomicity is *eventually-cryptographic*: there is a transient window between Step 2 prepare and Step 4 commit during which a shard's staged overlay exists but is not yet committed. During that window, the shard's external API MUST NOT expose the staged state – queries return pre-T_j state. This is the standard 2PC invisibility property; LP-210's Phase-4 staging guarantees it by construction.

4.3 Monotonic cert-tier upgrade across shards

LP-202's "Cert leg degradation tiers" allows a QuasarCert to be observable at a lower mode than the chain's configured target, when slower legs fail to aggregate within the timeout. The cert is monotone: once a leg arrives, the cert is observable at the higher tier; legs never retract.

Cross-shard interaction. The parent-L1 cross-shard commit cert is observable at progressively higher modes as its legs arrive. A relying party (a shard executor watching for commit-notify) applies its *own* cert-mode policy against the parent-L1 cert. The shard does not need

to wait for the parent L1 to reach the parent L1's configured target mode; it only needs to wait for its own configured mode.

Worked example: parent L1 configured PQ-heavy, shards A and B configured PQ-strict.

t (ms)	Event	Parent-L1 cert observable at	Shard A acts?	Shard B acts?
0	Commit tx in parent-L1 wave	n/a	no	no
100	Parent-L1 BLS leg aggregates	PQ-off	no (A wants $\geq$ PQ-strict)	no
105	Parent-L1 Corona leg aggregates	PQ-fast	no	no
110	Parent-L1 Pulsar leg aggregates	PQ-strict	**yes – A commits**	**yes – B commits**
155	Parent-L1 Magnetar leg aggregates	PQ-heavy	(already committed)	(already committed)

Table 3: Monotonic cert-tier observability across shards.

The shards commit at PQ-strict observability (their configured mode), not at the parent L1's target PQ-heavy. The parent L1 still waits for Magnetar before finalizing at its target – but the shards do not.

Architectural decision. Shards may commit early at their individual mode without waiting for the parent L1's target mode. This follows from LP-202's monotonic-upgrade invariant – the parent-L1 cert is the same struct at every observation point; only the leg-set membership grows. A shard acting on an early observation cannot be wrong, because the cert at the lower mode is a strict subset of the cert at the higher mode.

The cross-shard 2PC commit point is the minimum-mode-observability of the parent-L1 cert across all participating shards, not the parent L1's configured target. Latency is bounded by the slowest-configured shard, not by the parent L1's configured target.

5 Latency

LAN, PQ-strict mode at every layer, 40-validator parent L1 with 4-shard fan-out. Two columns: the projected GPU-accelerated path and the 2026-06-04 quasar-matrix-bench measured CPU-only path.

End-to-end. Projected GPU path: ∼260 ms at PQ-strict LAN. WAN, PQ-strict mode: ∼600 ms–800 ms (RTT-dominated; LP-209 latency triples under WAN). PQ-heavy mode under projected GPU: add ∼150 ms to each cert step; total ∼400 ms LAN, ∼1000 ms WAN.

Measured CPU path (matrix bench Spark N=64, `/tmp/quasar-results-spark.txt`): ∼1326 ms cross-shard finality at PQ-strict LAN. PQ-heavy CPU cert wall-clock 1326 ms (matrix bench) pushes per-shard wave to ∼1442 ms; parent L1 adds another ∼1442 ms; total cross-shard PQ-heavy LAN ∼2900 ms. The CPU-only path is the floor until LP-203 GPU wrappers (Magnetar/Pulsar/Corona verify) ship; the GB10 kernels themselves are operational (BLS pairing 49 197/s at 19 μs/pair, keccak256 11.38 Mops/s).

The dominant cost is the **two sequential LP-209 wave commits** – per-shard prepare wave plus parent-L1 commit wave. Optimizations:

- **Speculative commit at PQ-fast.** If the shard cert is PQ-fast and the parent-L1 cert is also PQ-fast under the *projected* GPU-accelerated path, both certs land within ∼5 ms LAN;

Event	GPU proj.	CPU meas.	Source
Cross-shard tx submitted to coordinator	0 ms	0 ms	client
Coordinator dispatches to shards via DHT	∼10 ms	∼10 ms	LP-201 DHT
Per-shard mempool admit	∼25 ms	∼25 ms	LP-208 tx receive
Per-shard wave commits T_j with prepare-staged	∼120 ms	∼711 ms	LP-209 PQ-strict latency
Prepare-acks arrive at coordinator	∼130 ms	∼721 ms	+ 10 ms gossip
Coordinator submits CrossShardCommit	∼131 ms	∼722 ms	one mempool insert
Parent L1 wave commits CrossShardCommit	∼251 ms	∼1317 ms	+ LP-209 wave PQ-strict
Shards observe commit-notify and apply staged overlay	∼260 ms	∼1326 ms	+ 10 ms gossip + LP-202 $O(1)$

Table 4: End-to-end cross-shard finality at PQ-strict LAN. GPU column is projected (LP-209 latency assuming the projected ∼5 ms GPU-accelerated cert from LP-203 GB10 sm_120 pairing rates measured 2026-06-04). CPU column is grounded in the matrix-bench-measured cert wall-clock at Spark N=64 PQ-strict 595 ms (`/tmp/quasar-results-spark.txt`); LP-209 LAN wave $\approx$ 116 ms pre-cert + 595 ms cert $\approx$ 711 ms.

total cross-shard latency drops to ∼150 ms. On the measured CPU-only path PQ-fast cert wall-clock is 585 ms (matrix bench Spark N=64), so this optimization is GPU-gated.

- **Pipelined cross-shard groups.** Multiple in-flight cross-shard groups overlap their wave commits via LP-205 pipelining; aggregate cross-shard throughput remains at the parent-L1 wave rate. Under projected GPU: $\sim$ 10 waves/sec $\times$ $\sim$ 100 cross-shard txs/wave = $1000\,\mathrm{cross-shard\,tx/s}$ floor. Under measured CPU: $\sim$ 1.7 waves/sec (matrix bench PQ-strict N=64) $\times$ $\sim$ 100 = $170\,\mathrm{cross-shard\,tx/s}$ floor.

5.1 Pipelined cross-shard groups (worked example)

Parent-L1 wave	t=T	t=T+100 ms	t=T+200 ms
W_k – contains CrossShardCommit(T_A)	committed	settled	settled
W_{k+1} – contains CrossShardCommit(T_B)	preparing	committed	settled
W_{k+2} – contains CrossShardCommit(T_C)	still collecting prepare-acks	preparing	committed

Table 5: Three cross-shard tx groups, three overlapping parent-L1 wave commits.

Aggregate cross-shard throughput at LAN scale $\approx$ parent-L1 wave rate $\times$ (parent-L1 wave capacity / `CrossShardCommit` tx size). Under projected GPU-accelerated cert (LP-203): $\sim$ 1000 cross-shard groups/sec at PQ-strict, $\sim$ 500 at PQ-heavy. Under measured CPU-only cert (2026-06-04 matrix bench Spark N=64): $\sim$ 170 cross-shard groups/sec at PQ-strict (595 ms cert wall-clock), $\sim$ 75 at PQ-heavy (1326 ms cert wall-clock).

LP	Composition
LP-079	Q-Chain finality blocks (where cross-shard commits ultimately land)
LP-182	Consensus wire / QuasarCert (the cert that anchors the cross-shard commit)
LP-201	ZAP P2P (DHT layer that dispatches large tx groups)
LP-202	ZAP Pipelining and Atomic Unwind (the atomic-commit and monotonic cert-tier-upgrade contracts cross-shard 2PC depends on)
LP-203	GPU-native verify (cross-shard latency floor is the QuasarCert floor, which inherits from LP-203's GB10 sm_120 pairing rates)
LP-204	Network of blockchains (Tier-2 chain-VMs are the shards)
LP-205	Pipelined block production (parent-L1 wave pipelining for concurrent cross-shard groups)
LP-208	DAG mempool / header DAG (per-shard mempool that receives cross-shard parts; equivocation gate)
LP-209	Mysticeti-style total-order on DAG (per-shard and parent-L1 wave commits)
LP-210	Block-STM optimistic parallel execution (per-shard staged-prepare in Phase 4)
LP-217	Cert Profile Modes (per-shard and parent-L1 cert modes)

Table 6: Composition of LP-211 with sibling LPs.

6 Composition With Sibling LPs

7 Failure Modes

8 Activation Marker

```
activates:      2025-12-25T16:20:00-08:00
activates-unix: 1766708400
```

Applies to cross-shard tx group atomicity from the genesis block of the new final Lux network onward.

9 Backwards Compatibility

None. The pre-Quasar Edition Lux network has no cross-shard primitive and is out of scope.

References

Failure	Trigger	Effect	Recovery
Coordinator offline	crash, partition	prepare-acks accumulate but no commit tx submitted	shards' staged overlays time out via local watchdog; LP-202 unwind; tx group dropped; client re-submits
One shard refuses prepare	validity failure, byzantine	coordinator receives `CrossShardRefuseAck`; aborts via Step 4'	every prepared shard unwinds; tx group dropped
Coordinator equivocates	byzantine	parent-L1 LP-209 wave ordering admits only one (first-seen); LP-208 equivocation gate suppresses the second	atomicity preserved by parent-L1 wave ordering
Parent L1 partitions before committing	infrastructure	shards' staged overlays time out; LP-202 unwind	client re-submits when partition heals
Shard commits before parent L1 cert finalizes at shard's mode	bug	observable as stale read; shard exposes uncommitted state	bug in the shard's executor, not in LP-211
Cross-shard tx too large for one wave	>1 MiB	LP-201 chunked DHT fetch handles arbitrary-size tx groups; coordinator dispatches `(group_id, contentHash)`	works by construction

Table 7: Failure modes.