Radial DEX:

The DEX Where Liquidity Earns While It Waits

A SOL-Native Yield-First Hub Architecture for Solana

May 11, 2026

Abstract

Radial DEX is presented — a SOL-native, yield-first decentralized exchange for Solana built around a single capital-efficient hub. Unlike conventional AMMs that keep all liquidity idle in swap pools, Radial deploys the majority ($\sim 80\%$) of hub capital directly to Solana validators via native staking, while retaining only the minimum on-chain buffer needed for parallel swap execution.

The on-chain buffer is sized by a deterministic worst-case bound, $C^{\text{active}} = T_{\max}^{(h)} \cdot K \cdot s_{\max}$, where $T_{\max}^{(h)} = 32$ (configurable `u8`) is the maximum parallel transactions per slot, $K = 10$ swaps per transaction, and $s_{\max}$ the per-swap cap. The remaining capital is delegated through a bucket model wrapping the Solana Stake Program (one stake account per bucket, decoupling activate/deactivate cadence from epoch boundaries). Settlement of intra-hub flows is deferred via an inter-bin claim ledger that records net flows between $B = 640$ parallel execution bins; physical capital movement occurs only when a bin's ledger entry exceeds a 20% threshold.

The Phase 1 design is oracle-free: prices emerge from each bin's reserve-implied ratio $P_n = a_b/R_b$, analogous to Meteora DLMM. Manipulation is defended by bin-traversal cost, per-transaction user safeguards (deadline-based invalidation, output slippage cap), and arbitrage closure.

For drawdowns deeper than the staking-bucket cadence can absorb, users can opt at withdrawal time to receive *rSOL* — a transferable liquid-staking receipt minted by the hub against its active validator stake. rSOL gives the LP an instant exit while letting physical SOL settle at the next epoch, decoupling exit UX from the validator deactivation window.

The active-ratio band $\rho \in [0.10, 0.30]$ and the 2σ operational buffer are calibrated against 150 consecutive days of on-chain indexed reserves across the six largest SOL-denominated Solana AMM architectures (Meteora DLMM, Meteora DAMM v2, Meteora Dynamic Bonding Curve, Raydium CLMM, Raydium AMM v4, Raydium CPMM). The aggregate dataset shows an 88–99% idle ratio and a worst observed 3-day drawdown of -28.4% on the most volatile architecture — empirically validating the yield-first capital allocation.

Contents

1 Introduction and Motivation

Solana hosts the second-largest decentralized exchange ecosystem after Ethereum, processing tens of billions in monthly swap volume across thousands of pools on Raydium, Orca, Meteora, and aggregators like Jupiter. Yet beneath this throughput lies a structural inefficiency: the vast majority of SOL sitting in AMM pools never participates in a trade on any given day. Over 150 consecutive days of on-chain indexed data covering the six largest SOL-denominated AMM architectures, the daily idle floor ranges from 88% to 99% of TVL (Table 5). This idle capital represents a massive opportunity cost in a blockchain ecosystem where SOL earns $\sim 6.8\%$ APY through native staking and 6.0–7.5% through liquid-staking tokens.

The idle-capital problem. Conventional AMMs (Uniswap V2/V3, Orca Whirlpool, Raydium CLMM and CPMM, Meteora DLMM and DAMM) require liquidity providers to deposit SOL into isolated pools, which must remain fully available for swaps at all times. On Solana, this means:

- A SOL-paired pool on a conventional Solana AMM with 10M SOL TVL keeps the full 10M in the pool contract, earning only swap fees ($\sim$ 1–3% APY). Meanwhile, the same SOL deposited as a native validator stake earns $\sim 6.8\%$.

- Multi-hop swaps via aggregators move intermediate SOL physically through wallets and ATAs at every hop, adding compute-unit overhead and forcing the intermediate to remain liquid on both sides.
- Across all six measured Solana AMM architectures, the aggregate mean idle floor is 150.8M SOL ($\sim$ \$30.2B at \$200/SOL). At 6.17% blended yield, this idle capital represents $\sim$ \$1.86B/year of foregone return (Section 11).

Three structural costs. Current designs incur three distinct costs that we eliminate:

1. **Idle capital.** Liquidity that could earn yield sits unproductively in swap pools, earning only the thin margin of swap fees.
2. **Physical intermediate transfers.** Multi-hop swaps move assets redundantly when a ledger update would suffice.
3. **Pair fragmentation.** For n active tokens, up to $\binom{n}{2}$ pools fragment liquidity, each requiring its own state account, rent, and LP deposits.

The Radial solution. Radial DEX addresses all three costs through a single insight: *a DEX does not need all of its capital on-chain at once.* The maximum Solana can process in a single slot is bounded:

$$C^{\text{active}} \geq T_{\max}^{(h)} \cdot K \cdot s_{\max},$$

where $T_{\max}^{(h)} = 32$ (default, configurable per-hub as `u8`), $K = 10$ swaps per transaction, and $s_{\max} = 31{,}250$ SOL per swap. For a 100M SOL hub, only 20M SOL (20%) is needed on-chain to handle any plausible single-slot demand. The remaining 80M SOL works in a single, narrow yield route: direct native validator staking via the bucket model of Section 8.5. This capital is not gone — it is tracked by an *inter-bin claim ledger* that records each bin's virtual SOL balance. When swaps execute, the ledger updates atomically; physical capital is reclaimed from the staking buckets only when a bin's claim drifts beyond a threshold. For drawdowns deeper than the staking-bucket cadence can absorb, an opt-in withdrawal-time mint produces *rSOL* — a transferable liquid-staking receipt against the hub's active stake (Section 9), giving the LP an immediate exit while letting physical SOL settle at the next epoch boundary.

Four design pillars.

1. **Yield-first capital allocation.** The hub's on-chain buffer is sized by the deterministic worst-case bound above, with the remainder deployed to native validator staking. The active ratio $\rho = V_{\text{vault}} / \sum a_i$ is maintained in $[0.10, 0.30]$ by a keeper, calibrated against the empirical σ_{daily}–3σ envelope observed in production AMMs.
2. **Parallel execution bins with inter-bin ledger.** The on-chain buffer is partitioned into $B = T_{\max}^{(h)} \cdot K$ independent bins, each its own account. Non-conflicting swaps execute in parallel. The inter-bin ledger defers physical settlement to threshold-driven batched rebalances.
3. **Inter-bin virtual netting.** Cross-token swaps through the SOL hub graph route via two ledger updates — spoke A $\to$ SOL bin $\to$ spoke B — without physically moving SOL between bins. SOL is the canonical anchor and the only Phase 1 hub.
4. **Empirically calibrated buffer.** Hub parameters are not set by intuition but by direct measurement: across 150 days and six AMM architectures, the worst-observed 3-day SOL drawdown is -28.4% on Meteora DLMM; the median protocol is bounded by a 3σ buffer. Radial's $\rho \in [0.10, 0.30]$ band is the operating range that empirically dominates this envelope (Section 11).

Price discovery without oracles (Phase 1). The current production target (Phase 1) derives swap prices from each bin's reserve-implied ratio $P_n = a_b/R_b$, analogous to Meteora DLMM's discrete-bin pricing. This eliminates the oracle manipulation surface for intra-hub flow entirely — there is no external feed to lag, front-run, or price-asynchronously manipulate. A Phase 2 follow-up may optionally introduce Pyth or Switchboard feeds for routing across multiple hubs once such hubs are governance-approved; the discussion is deferred to Section 14.

Prior art. Radial builds on Solana-native and Ethereum-native innovations. *Meteora DLMM (Dynamic Liquidity Market Maker) v1* pioneered bin-based liquidity with reserve-implied pricing and per-bin accounts for parallel execution. Radial generalizes the bin geometry and constant-sum invariant, replaces LP-chosen ranges with uniform hub deposits, and adds the inter-bin ledger so capital can earn native-staking yield while remaining trade-ready. *Meteora Dynamic Vaults* demonstrated that AMM idle capital can be programmatically routed to yield, though limited to per-pool lending vaults rather than hub-level direct staking. Radial extends the idea to the entire SOL hub with a single, transparent yield route (validator staking) instead of a multi-protocol lending aggregator.

Contributions. This paper presents:

- A **yield-first SOL hub** that deploys $\sim 80\%$ of LP capital to native validator staking while retaining deterministic solvency (Pillar 1, Section 8).
- The **inter-bin claim ledger** and lazy keeper settlement protocol, enabling $O(1)$ on-chain swap execution with deferred physical settlement (Pillar 2, Section 7).
- **rSOL withdrawal-time mint**, a tail-relief mechanism that converts a user withdrawal into a transferable liquid-staking receipt when the SOL buffer is locally exhausted, preserving instant LP exit without forcing premature unstaking (Section 9).
- An **empirical buffer analysis** of 150 days of on-chain reserves across six Solana AMM architectures, justifying the $\rho \in [0.10, 0.30]$ active-ratio band and the 2σ operational buffer (Section 11).

The remainder of this paper is organized as follows. Section 2 surveys prior art. Section 3 fixes notation. Section 4 defines the hub primitive. Section 5 specifies the execution-bin architecture. Section 6 develops intra-hub cross-token swaps. Section 7 formalizes the inter-bin claim ledger. Section 8 derives the yield-first capital allocation. Section 9 specifies the rSOL withdrawal mechanism. Section 10 treats manipulation resistance. Section 11 presents the empirical idle-capital evidence. Section 12 works numerical examples. Section 13 compares with alternatives. Section 14 and Section 15 discuss limitations and conclude.

2 Related Work

Prior art is surveyed along three axes:

2.1 Yield-Integrated DeFi

Meteora DAMM v1 + Dynamic Vaults (closest prior art). Meteora's DAMM v1 [4] pairs a constant-product AMM with *Dynamic Vaults*, a per-pool aggregator that programmatically routes the idle side of each pool to external yield strategies. An off-chain keeper monitors yield destinations and rebalances allocations, with per-protocol caps to limit concentration risk. Radial generalizes this per-pool idle-yield pattern into a *per-hub* architecture: instead of deploying yield from individual pool buffers, the entire SOL hub's idle capital ($\sim 80\%$ of TVL) is

yield-eligible, and the yield path is a single, transparent route — direct native validator staking (Section 8.5) — rather than a multi-strategy aggregator.

Yield aggregators. Vault protocols (Yearn-style on Ethereum, Kamino-vault-style on Solana) automatically deploy capital to the highest-yielding strategy within a curated whitelist. They operate as an opt-in overlay: a user deposits into a vault, the vault deposits into a strategy, fees and exit times are vault-specific. Radial integrates the same idea at the *protocol* level rather than as an overlay: every LP automatically benefits from idle capital deployment, no opt-in or vault wrapper is required, and the yield route is fixed by the hub's policy rather than chased across protocols.

2.2 Discrete-Bin and Hub-and-Spoke Topologies

Meteora DLMM and Trader Joe Liquidity Book. Both use discrete price bins with constant-sum invariants and geometric bin steps. Meteora DLMM [4] implements this on Solana with per-bin PDAs for Sealevel parallelism; Trader Joe's Liquidity Book [2, 3] did the same on Avalanche. Radial adopts the same bin geometry ($P_n = P_0(1+s)^n$) and constant-sum invariant, but differs in three ways: (i) *uniform LP deposit* — LPs deposit into the SOL hub, not into chosen bin ranges, eliminating range-management complexity; (ii) *execution bins* — all B bins are simultaneously active (no single active-bin contention point), distributing swap load uniformly; (iii) *idle yield routing* — the hub deploys idle capital to native staking, which DLMM does not do.

MakerDAO Peg Stability Module (PSM). The PSM [5] is the architectural ancestor of Radial's virtual hub. Each whitelisted stablecoin has a "join adapter" that mints DAI against deposits. PSM trades update accounting entries rather than physically transferring DAI between pools. Radial generalizes this principle from stablecoin-only pegs to all SPL tokens routed through the SOL hub, replacing the fixed peg with reserve-implied bin pricing.

2.3 Parallel Execution on Solana

Meteora DLMM per-bin PDAs. Meteora DLMM uses per-bin PDAs for parallel execution, achieving $\sim 50\times$ throughput over single-account designs on Solana. Radial shares this per-bin PDA architecture but removes the active-bin contention bottleneck: in DLMM, the active bin captures 30–50% of volume and serializes those swaps; Radial uniformly distributes swaps across all B bins via deterministic hash assignment.

Orca Whirlpool and Raydium CLMM. These use single tick-array accounts, serializing all activity on the pool. Radial's per-bin PDAs eliminate this bottleneck entirely.

2.4 Liquid Staking Tokens (LSTs) and Withdrawal Receipts

Marinade, Jito, Sanctum. Marinade [7], Jito [6], and the Sanctum LST network [8] demonstrate that SOL deposited into a staking protocol can be represented by a transferable receipt token, enabling instant secondary-market exit even though the underlying SOL is locked in a validator stake. Radial's rSOL withdrawal mint (Section 9) borrows this primitive but inverts the trigger: rSOL is issued only *at withdrawal time*, by user choice, when the hub's SOL buffer is locally tight — rather than at every deposit. The hub remains the canonical SOL custodian; rSOL is a Plan-B exit receipt rather than the default deposit token.

2.5 Positioning

Radial's contribution is not a novel AMM formula. It is the *synthesis* of three proven ideas — discrete-bin pricing (Meteora DLMM), idle-capital yield routing (Meteora Dynamic Vaults), and per-bin Sealevel parallelism — unified under a single SOL-hub primitive with an inter-bin claim ledger that decouples swap execution from physical capital settlement, plus an opt-in withdrawal-time liquid-staking receipt (rSOL) for tail-event relief. Each component exists in production; Radial is the first to combine them in a yield-first SOL-native architecture where $\sim 80\%$ of hub capital is delegated to validators while only the deterministic worst-case buffer sits on-chain for swap execution.

3 Notation

Definition 1 (Hub state)**.** Each hub $h \in \mathcal{H}$ stores

$$h = (V_{\text{vault}},\ I_{\text{stake}},\ F,\ \{a_b\}_{b=1}^{B},\ S_r,\ R(t)),$$

where V_{vault} is the on-chain vault balance (native lamports or wSOL), I_{stake} is the SOL delegated to native validator stake buckets, F is the accumulated fee pool, $\{a_b\}$ are the per-bin hub-asset claims, and $(S_r, R(t))$ track the outstanding rSOL supply and current redemption rate (Section 9). The global solvency invariant is

$$V_{\text{vault}} + I_{\text{stake}} \ = \ \sum_{b=1}^{B} a_b \ + \ S_r \cdot R(t).$$

With no outstanding rSOL ($S_r = 0$) this reduces to the simpler $V_{\text{vault}} + I_{\text{stake}} = \sum_b a_b$ used in most of the analysis.

Definition 2 (Bin state)**.** Each execution bin b stores

$$\text{bin}_b = (R_b,\ a_b,\ L_b,\ \text{status}_b),$$

where R_b is the token reserve (in token units), a_b is the hub-asset claim, L_b is the inter-bin ledger entry, and $\text{status}_b \in \{\textsc{Active}, \textsc{Draining}, \textsc{Settling}\}$.

Definition 3 (Active ratio)**.** The active ratio of hub h is

$$\rho_h = \frac{V_{\text{vault}}}{\sum_{b=1}^{B} a_b} \in [0, 1].$$

The keeper maintains $\rho_h \in [\theta_L, \theta_H]$ with $\theta_L = 0.10$, $\theta_H = 0.30$, and target $\rho^* = 0.20$.

Remark 4 (No pool-level invariants). Unlike conventional AMMs, Radial does not maintain per-pool conservation invariants ($x \cdot y = k$, constant-sum, etc.). The only invariant is the hub-level ledger conservation $\sum_b L_b = 0$. Swaps are priced through each bin's reserve-implied ratio $P_n = a_b / R_b$ of the active bin, which is why per-pool invariants are unnecessary.

4 Topology and the Hub Primitive

Radial's trade graph is built around an abstract *Hub primitive*. In Phase 1 the active hub set is the single canonical SOL hub: SOL is native to the chain, holds the deepest aggregate liquidity (Section 11), and has the most mature yield route (validator staking). The hub primitive itself is general; the choice to operate a single hub is a Phase 1 scope decision and is revisited in Section 14.

Symbol	Meaning	Unit
$\mathcal{H}$	Set of hubs	—
B	Number of execution bins per hub ($= 640$)	—
R_b	Token reserve of bin b	smallest token unit
a_b	Hub-asset claim of bin b	lamports or stable unit
L_b	Inter-bin ledger entry for bin b	lamports or stable unit
V_{vault}	On-chain vault balance	lamports or stable unit
I_{stake}	SOL delegated to native validator stake buckets	lamports
F	Accumulated fee pool	lamports or stable unit
ρ	Active ratio $V_{\text{vault}}/\sum a_b$	dimensionless
θ_L, θ_H	Keeper rebalance thresholds	dimensionless
C^{active}	Deterministic active buffer (Definition 22)	lamports
$T_{\max}^{(h)}$	Max parallel swaps per slot for hub h, u8 configurable (default 32)	—
$K_{\max}$	Max swaps per transaction ($= 10$)	—
$s_{\max}^{(h)}$	Per-swap max input size for hub h	lamports or stable unit
f	Swap fee	bps
S_r	Outstanding rSOL supply (Definition 26)	rSOL units
$R(t)$	rSOL redemption rate (Definition 26)	dimensionless
P_n	Reserve-implied bin price (a_b/R_b)	dimensionless

Table 1: Notation used throughout.

Operation	$\Delta\sum a_b$	ΔV_{vault}	ΔI_{stake}	$\Delta(S_r \cdot R)$	$\Delta\sum L_b$
Swap (hub asset in/out)	0	$\pm\Delta$	0	0	0
LP join	$+\Delta$	$+\Delta$	0	0	0
LP exit (SOL path)	$-\Delta$	$-\Delta$	0	0	0
LP exit (rSOL path)	$-\Delta$	0	0	$+\Delta$	0
rSOL burn (queued)	0	0	$-\Delta$	$-\Delta$	0
Keeper stake deploy	0	$-\Delta$	$+\Delta$	0	0
Keeper bucket deactivate	0	$+\Delta$	$-\Delta$	0	0

Table 2: Per-operation effect on conserved quantities. The invariant $V_{\text{vault}} + I_{\text{stake}} = \sum_b a_b + S_r \cdot R(t)$ and $\sum_b L_b = 0$ are preserved by all operations.

4.1 Hub primitive

Definition 5 (Hub primitive)**.** A hub is a tuple

$$h = (\text{mint},\ \text{vault},\ \text{stake\_buckets},\ \text{rsol\_mint},\ B,\ \kappa,\ \text{status},\ \text{governance}),$$

where:

- mint is the SPL token mint identifying the hub asset (native SOL, wrapped as `wSOL` for SPL-token compatibility where required);
- vault is the on-chain account holding the hub's active SOL buffer;
- stake_buckets $= \{B_1, B_2, \ldots, B_N\}$ is the set of native-stake buckets that hold the deployed portion of hub capital (Section 8.5);
- rsol_mint is the SPL mint authority for the rSOL withdrawal receipt token (Section 9);
- B is the number of execution bins (default $B = 640$);
- κ is the per-hub safety multiplier on the worst-case-bound buffer;

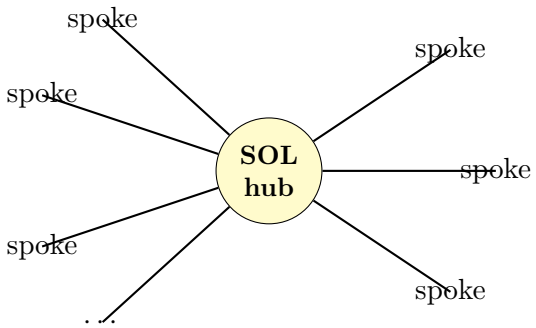

Figure 1: Phase 1 architecture overview. A single SOL hub serves as the routing nexus for every spoke token. Each spoke connects to the hub through its own execution-bin pool; spoke-to-spoke swaps route through the hub via the inter-bin ledger (Section 6).

- status $\in \{\textsc{Bootstrap}, \textsc{Active}, \textsc{WindDown}\}$;
- governance is the authority for parameter changes, spoke admission, and wind-down.

Remark 6 (Phase 1 is single-hub). Throughout this paper the active hub set is $\mathcal{H} = \{\text{SOL}\}$. Every spoke token $t \in \mathcal{S}$ routes through this one hub. The data structures and lifecycle below are written generally so that a future Phase 2 extension can introduce additional hubs (e.g., stablecoin hubs) without protocol redesign, but Phase 2 is out of scope for the implementation discussed in this paper.

4.2 Hub set and topology

The Phase 1 trade graph is a star: every spoke $t \in \mathcal{S}$ has a single edge to the SOL hub.

Definition 7 (Phase 1 trade graph)**.** The Phase 1 Radial trade graph is the simple undirected graph $\mathcal{G} = (V, E)$ with

$$V = \{\text{SOL}\} \cup \mathcal{S}, \qquad E = \{(t, \text{SOL}) : t \in \mathcal{S}\}.$$

4.3 Reachability and routing

Proposition 8 (Reachability)**.** *For any two distinct spoke vertices $u, v \in \mathcal{S}$, the shortest path through $\mathcal{G}$ is exactly length 2 via SOL. Every spoke-to-spoke trade is therefore a deterministic two-hop swap through the SOL hub.*

This is the central simplification of single-hub Phase 1: routing is constant and trivially deterministic.

4.4 Lifecycle and governance

- **Genesis:** $\mathcal{H} = \{\text{SOL}\}$, $\mathcal{S} = \emptyset$. The SOL hub is bootstrapped with an LP deposit that funds the initial worst-case buffer.
- **Adding a spoke:** Governance proposes a new spoke token with bin-step parameters. After an N-day timelock and a bootstrap-TVL deposit, the spoke transitions $\textsc{Bootstrap} \to \textsc{Active}$.
- **Wind-down:** A spoke may be retired by governance; outstanding LP positions are settled at the wind-down spot price. The SOL hub itself cannot be wound down.

4.5 Authority model and emergency controls

The hub recognizes three distinct authority roles, all enforced on-chain:

- **Admin.** Holds the configuration and lifecycle authority for the hub (initialization, parameter updates, stake-bucket lifecycle, paused/unpaused flag, wind-down). Admin transfer is *two-step*: the current admin proposes a successor (PROPOSE), and the successor must explicitly accept (ACCEPT) before the transfer takes effect. This prevents a single failed transaction — including a typo, a compromised RPC endpoint forging a destination, or a key lost between propose and accept — from stranding the admin role.
- **Keepers.** A whitelisted set of off-chain operators authorized to call the rebalance and stake-bucket handlers (deposit, withdraw, claim-yield, reconcile, native-stake bucket activate/deactivate, rSOL queue settlement). The whitelist is admin-managed.
- **Users.** Any signer can call the user-facing handlers (swap, add liquidity, remove liquidity, withdraw with rSOL option); no whitelist is required.

Emergency stop. The hub carries a *paused* flag, settable by the admin. When set, all user-facing handlers reject. Fund-moving keeper handlers (rebalance, stake-bucket activate/deactivate, rSOL queue settlement) also reject. Read-only and reconciliation keeper handlers (sync, reconcile, status reads) remain operative so that forensic accounting can proceed during an incident without losing accurate balance state. Unpause is the inverse admin action.

5 Reserve-Priced Execution Bins

Radial bins serve a dual role: they are *parallel execution slots* for concurrent swap processing on Solana's Sealevel runtime, and they are *reserve-implied price points* on a geometric grid. Each bin holds token and hub-asset reserves subject to a constant-sum invariant at a fixed bin price. The price is derived entirely from the bin's position on the grid — no oracle is consulted — while the Sealevel scheduler dispatches swaps across disjoint bin write sets within a single slot. Unlike conventional discrete-bin AMMs (Meteora DLMM, Trader Joe LB) where LPs choose a price range $[n_{\min}, n_{\max}]$ and deposit capital into specific bins, Radial LPs deposit into the hub. The protocol distributes deposits *uniformly* across all bins. There is no per-LP range selection, no weighted distribution, and no position migration. The result is a fully populated price grid where every bin is simultaneously active and independently executable.

5.1 The bin as a priced execution slot

Definition 9 (Reserve-priced execution bin)**.** An execution bin is its own on-chain account storing a tuple

$$\text{bin}_b = (x_b,\, y_b,\, L_b,\, \text{status}_b),$$

where x_b is the token reserve (in token units), y_b is the hub-asset reserve (SOL, denominated in lamports or wSOL units), L_b is the cumulative ledger delta (see §7), and $\text{status}_b \in \{\text{ACTIVE}, \text{DRAINING}, \text{SETTLING}\}$ indicates the bin's lifecycle stage. Each bin carries a *fixed geometric price*

$$\boxed{P_b \;=\; P_0 \cdot (1+s)^b}, \qquad b \in \{0, 1, \ldots, B-1\},$$

where $s \in (0,1)$ is the bin step (governance-tunable per hub, default $s = 10^{-3}$ for 10 bps spacing) and P_0 is the hub's bootstrap price. The price P_b is denominated in hub-asset units per token unit — it is a property of the bin's index, not an external feed.

Remark 10 (No oracle). Because P_b is fixed by the geometric grid and the invariant is enforced by bin reserves, the swap price is *reserve-implied*. The protocol never reads a Pyth price, a TWAP accumulator, or any external oracle during swap execution. Price discovery is entirely endogenous: users observe the bin grid and trade at the bin price that matches their desired execution rate.

5.2 Per-bin invariant: constant-sum

Each bin's reserves satisfy a constant-sum constraint at the bin's fixed price:

$$\boxed{x_b \cdot P_b + y_b \;=\; k_b},$$

where k_b is the bin's total value denominated in hub-asset units. Within a single bin, the invariant is linear: a user who deposits Δ_{in} token units receives $\Delta_{\text{in}} \cdot P_b$ hub-asset units (after fees); conversely, depositing Δ_{in} hub-asset units yields Δ_{in}/P_b token units. There is no slippage *inside* a bin — the price is exactly P_b for any trade that the bin's reserves can accommodate.

Proposition 11 (Zero in-bin slippage)**.** *For any swap that completes without exhausting the receive-side reserve of a single bin b, the effective price equals P_b exactly, fees aside.*

Proof. The constant-sum invariant gives $\Delta x_b \cdot P_b + \Delta y_b = 0$. A token-to-hub-asset swap satisfies $\Delta y_b \,/\, \Delta x_b = P_b$; a hub-asset-to-token swap satisfies $\Delta x_b \,/\, \Delta y_b = 1/P_b$. Since P_b is fixed, the ratio depends only on b, not on trade size. □

Because the protocol distributes deposits uniformly across all B bins, every bin starts with the same total value $k_b = \text{cap}_b$ (see §8). The initial split between x_b and y_b follows from the constant-sum constraint: for a bin at price P_b, the reserves are

$$x_b^{(0)} = \frac{\text{cap}_b}{2\,P_b}, \qquad y_b^{(0)} = \frac{\text{cap}_b}{2}.$$

After swaps, reserves drift, but k_b changes only through fee accumulation (which increases k_b) or keeper-led rebalancing (which resets the bin's baseline).

Remark 12 (Uniform deposit, non-uniform reserves). All bins receive the same initial value cap_b, but the *composition* differs by price level. A high-price bin ($P_b \gg P_0$) holds fewer token units x_b and more hub-asset units y_b; a low-price bin ($P_b \ll P_0$) holds more tokens and less hub asset. This is natural — a bin that prices the token higher needs fewer tokens to represent the same total value. The uniform value allocation ensures that no bin is a bottleneck: every price point on the grid carries equal capital.

5.3 Swap execution model

Each swap on Radial is routed to a starting bin via deterministic hash assignment. The swap consumes input from that bin's reserves at the bin's fixed price. If the swap does not exhaust the receive side, it completes as an in-bin trade (zero slippage). If the swap exceeds the bin's reserves on the receive side, it *traverses* to the adjacent bin in the direction of trade. A swap is routed to a starting bin by a deterministic hash that combines the slot's blockhash, the transaction's index within the slot, and the instruction's index within the transaction:

$$b_0 \;=\; \text{hash}(\texttt{slot\_hash} \,\|\, \texttt{program\_tx\_index} \,\|\, \texttt{instruction\_idx}) \bmod B.$$

The `slot_hash` is read from the `recent_blockhashes` sysvar and is unspoofable within the slot; combining it with the per-slot transaction and instruction indices removes the `tx_signature` grinding vector present in signature-based bin assignment. If b_0 is in DRAINING status on the receive side, traversal advances deterministically to the next bin with available reserves. The net

input after the swap fee, $r = (1-f)\,\Delta_{\text{in}}$, is consumed in-bin at the fixed price P_b until either r is exhausted or the bin's receive-side capacity is depleted; in the latter case the traversal advances to the adjacent bin in the direction of trade. Once the swap settles, the inter-bin ledger entries L_b on the touched bins are updated (Section 7) and the gross output is compared against the user's slippage bound $\Delta_{\text{out}}^{\min}$. Because each swap writes only to the bins it touches, two swaps with disjoint traversal paths have no writable-set intersection.

Remark 13 (Effective price across traversal). For a swap traversing bins $b_0, b_1, \ldots, b_k$, the user's effective price is the volume-weighted mean of the bin prices encountered:

$$P_{\text{eff}} = \frac{\sum_{i=0}^{k} r_i \cdot P_{b_i}}{\sum_{i=0}^{k} r_i} \quad \text{(token-in)}, \qquad P_{\text{eff}} = \frac{\sum_{i=0}^{k} r_i}{\sum_{i=0}^{k} r_i / P_{b_i}} \quad \text{(hub-asset-in)}.$$

The effective price is bounded by P_{b_0} and $P_{b_k} = P_{b_0} \cdot (1+s)^{\pm k}$, giving a worst-case slippage of $k \cdot s$ relative to the entry bin. For an in-bin trade ($k = 0$), slippage is exactly zero.

5.4 Bin lifecycle

A bin transitions through three states:

Active: Both sides non-empty ($x_b > 0$ and $y_b > 0$). Swaps may traverse into the bin from either direction. Most bins are in this state under normal load.

Draining: One side has been exhausted ($x_b = 0$ or $y_b = 0$). The bin accepts swaps only on the side with remaining reserves. This is the DLMM-equivalent of a bin that has been "crossed" by the market — the price has moved past this grid point. The keeper observes the draining bin during periodic rebalancing (see §7) and restores both sides by injecting capital from the yield-route pool.

Settling: The bin's ledger delta $|L_b| > \tau \cdot \text{cap}_b$. The keeper initiates physical settlement: reclaiming capital from yield routes or deploying excess. Settlement resets the bin's baseline and returns it to ACTIVE with restored reserves.

Because all bins receive equal capital, the draining pattern is symmetric: under a rising token price, high-index bins receive token-inflow and hub-asset-outflow; under a falling price, low-index bins receive hub-asset-inflow and token-outflow. The geometric grid ensures that price moves of $(1+s)^n$ drain exactly n bins on one side, leaving the others fully active.

5.5 Per-bin curve selection

The derivation above (constant-sum invariant at fixed bin price P_b, zero in-bin slippage by Proposition 11) describes the *default* bin behavior. The implementation extends this with a per-bin curve-type flag that selects how reserves price within a bin:

- **Constant-price (default).** The bin enforces the constant-sum invariant $x_b \cdot P_b + y_b = k_b$ at the fixed grid price $P_b = P_0 \cdot (1+s)^b$. This is the DLMM-analog mode used throughout this section and in the derivations of §5.2–§5.3; all of those statements apply to bins flagged constant-price. In practice this is the configuration appropriate for the vast majority of pairs.
- **Constant-product (alternative).** For specific tail-liquidity classes where the implementer prefers a continuous in-bin curve (e.g., long-tail tokens with sparse external reference prices), a bin may instead enforce the Uniswap V2-style constant-product invariant $x_b \cdot y_b = k_b$. In this mode the in-bin trade is not zero-slippage — the realized price slips along the constant-product curve — and the bin's geometric grid price P_b serves as the bin's initial pricing anchor rather than a strict in-bin clearing price. The bin traversal logic of Section 5.3 is structurally identical: the bin reports its in-bin capacity and the swap engine advances when the capacity is exhausted.

The curve-type flag is set at bin initialization and is not changed by user-initiated operations. Constant-price bins satisfy all in-bin slippage statements in this section; constant-product bins relax the zero-slippage claim (Proposition 11) to the standard constant-product slippage formula but otherwise interact identically with the inter-bin ledger and the parallel-execution scheduler.

5.6 Bin count and hub sizing

The total number of bins B is fixed per hub:

$$B = T_{\max} \cdot K = 64 \cdot 10 = 640,$$

matching the deterministic worst-case bound of §8. Each bin receives an equal share of the on-chain buffer:

$$\text{cap}_b = \frac{C_h^{\text{active}}}{B}.$$

For a \$100M SOL hub with \$20M on-chain buffer, $\text{cap}_b = 31{,}250$ SOL per bin. The bin step s is governance-tunable per hub. Recommended values follow the same regime as Meteora DLMM:

Pair class	s **(bps)**	± 100**-bin range**
Hard-peg	1	$\pm 1\%$
Soft-peg / LST	5–10	± 5–10%
Slow-drift volatile	50	$\pm 50\%$
Wide-volatility	100+	$\pm 100\%$+

Table 3: Recommended bin step s by hub class. Tighter s gives finer near-anchor granularity; wider s covers broader price ranges. $B = 640$ bins with $s = 10$ bps covers a total price range of $(1.001)^{640} \approx 1.90\times$; with $s = 50$ bps the range expands to $(1.005)^{640} \approx 24\times$.

Example 14 (SOL hub with \$100M TVL, $s = 10$ bps)**.** A SOL hub with $\mathcal{L}(\text{SOL}) = \$100\text{M}$. Deterministic buffer: \$20M SOL. With $B = 640$ bins, each bin holds \$31,250 in total value. The geometric grid spans $(1.001)^{639} \approx 1.90\times$, so the highest-price bin sits at $P_{639} \approx 1.90 \cdot P_0$, about 90% above the bootstrap price. In a single 400 ms slot, 64 parallel transactions each executing 10 in-bin swaps can consume up to $64 \cdot 10 \cdot 31{,}250 = \$20\text{M}$ — the entire on-chain buffer. Under normal load ($\sim$ 1–5% of capacity), no rebalancing is triggered.

Example 15 (rSOL/SOL hard-peg pool, $s = 1$ bp)**.** A future rSOL/SOL mini-pool (Section 9) configured with the hard-peg recipe: $s = 1$ bp gives a grid covering $(1.0001)^{639} \approx 1.066\times$, a narrow $\pm 3.3\%$ range appropriate for an asset whose protocol redemption rate $R(t)$ drifts slowly with epoch yield. Within this band the bin grid prices rSOL against SOL while the rSOL holder remains free to redeem at the protocol floor; the bin pool serves users who want instant SOL without burning rSOL.

5.7 Relation to discrete-bin AMMs

Radial's reserve-priced execution bins share the geometric grid and constant-sum invariant of discrete-bin AMMs (Meteora DLMM, Trader Joe LB), but differ in three fundamental ways:

- **No LP range selection.** In DLMM, each LP chooses a bin range $[n_{\min}, n_{\max}]$ and deposits capital into specific bins. This creates uneven liquidity, migration costs when the price drifts, and inventory risk for LPs who guess the wrong range. Radial LPs deposit into the hub; the protocol distributes capital uniformly across all bins. Every price point carries equal capital, and no LP ever needs to migrate.

- **All bins are active.** Conventional discrete-bin AMMs maintain a single *active bin index* n^* that advances as trades drain one side. Bins above n^* hold only token; bins below hold only hub asset. In Radial, *every* bin is simultaneously active because uniform capital distribution ensures both sides exist at every price level. Parallel execution requires this: only fully-populated bins can serve independent swap flows in the same slot.
- **No weights.** DLMM pools track per-bin LP share distributions, fee allocations, and position accounting. Radial bins are uniform: each holds the same capital, pays the same fee rate, and follows the same ledger protocol. Per-bin LP shares do not exist — LPs hold a single hub-level claim.

The reserve-priced execution bin design transforms the AMM problem from "where should LPs concentrate liquidity?" (the DLMM question) and "what price should the market clear at?" (the oracle question) into a unified question: "how much capital must stay on-chain at each price point for parallel swaps, and how can the rest earn yield?" This is the core innovation of Radial.

6 Cross-Swap via Virtual Hub Netting

A cross-swap from token A to token B is realized as the deterministic shortest path through the hub graph. In Phase 1 the hub set is $\mathcal{H} = \{\text{SOL}\}$ (Remark 6), so the shortest path is always the same shape: $A \to \text{SOL} \to B$, two ledger updates with no physical SOL movement, generalizing MakerDAO's PSM accounting to the swap-frequency regime.

6.1 Atomic cross-swap protocol

Each leg of the cross-swap uses the bin-swap procedure of Section 5.3 to execute against a binned pool. The source leg swaps ΔR_A of token A into SOL via the (A, SOL) binned pool at the bin price P_n implied by the active bin's reserves; the destination leg swaps SOL into ΔR_B of token B via the (B, SOL) binned pool, also at its bin's reserve-implied price. The intermediate SOL amount ΔS does not physically move: the source pool's claim on the SOL hub buffer increments by ΔS, and the destination pool's claim decrements by the same amount on the second leg. Both legs share the same SOL hub vault, so the net hub-asset displacement is zero. Per-bin writes $(R_{b_s}, a_{b_s}, R_{b_d}, a_{b_d})$ commit atomically; the inter-bin ledger entries L_{b_s}, L_{b_d} are updated; the user's output is checked against the slippage bound $\Delta R_B^{\min}$ and the transaction reverts as a whole if the bound is violated.

6.2 Conservation of hub liquidity

Proposition 16 (Hub conservation)**.** *The two-leg cross-swap preserves both the global invariant* $V_{vault} + I_{stake} = \sum a_b$ *and the physical hub vault balance* V_{vault}*.*

Proof. Step 1 increments the source pool's claim a_{b_s} by ΔS. Step 3 decrements the destination pool's claim a_{b_d} by $\Delta S' = \Delta S \cdot (1 - f_A)(1 - f_B)$ after fees. The net change to $\sum a_b$ is $\Delta S - \Delta S' = \Delta S \cdot [1 - (1 - f_A)(1 - f_B)]$, which equals the fee retained in the system (added to the protocol treasury or LP fee pool). Critically, V_{vault} is unchanged because no physical SOL leaves the vault; the two ledger updates settle against the shared buffer. The invariant holds. □

6.3 Fee composition

Lemma 17 (End-to-end effective fee)**.** *The effective fee paid by the user in a two-leg cross-swap is*

$$f_{eff} = 1 - (1 - f_A)(1 - f_B) = f_A + f_B - f_A f_B.$$

For $f_A, f_B \ll 1$*, this is approximately* $f_A + f_B$*.*

6.4 Single-hub cross-swap semantics (Phase 1)

Because all Phase 1 spokes share the SOL hub (Remark 6), every cross-swap between two spokes is *truly ledger-only*: no physical SOL moves. The source pool's ledger entry L_{b_s} increments, the destination pool's L_{b_d} decrements, and V_{vault} is unchanged. This is the regime described in the two-leg cross-swap; only the per-pool ledger entries change.

Remark 18 (Cross-hub is Phase 2). If a future Phase 2 admits a second hub (Section 14.4), cross-hub swaps would require a physical hub-to-hub leg through a dedicated bin pool, and the ledger-only property would apply only within each hub's own spoke set. The Phase 2 extension is deliberately out of scope for the Phase 1 implementation discussed in this paper.

6.5 Deterministic routing

The path between any two tokens is fixed: every spoke-to-spoke trade is exactly two legs against the SOL hub. There is no off-chain path-finding, no oracle, and no aggregator quoting layer. The on-chain program reads the input mint, looks up the corresponding mini-pool, and executes the two-leg cross-swap with exactly two legs. This eliminates the stale-quote failure mode of multi-DEX aggregators.

6.6 Comparison to physical-transfer routing

A naive implementation would transfer ΔS hub-asset between bins, requiring two additional SPL transfers and CPI overhead. The two-leg cross-swap eliminates these transfers by updating ledger entries L_{b_s} and L_{b_d} on the hub vault.

7 Inter-Bin Claim Ledger

The inter-bin claim ledger is the accounting layer that enables Radial's yield-first capital allocation. It decouples the logical swap execution (which happens on bins) from the physical capital movement (which happens through yield routes). The ledger tracks which bin has lent capital to or borrowed capital from the hub-wide pool, enabling deferred settlement.

7.1 Motivation

In a naive design, every swap that shifts capital between bins would require a corresponding physical movement of SOL through the hub vault. When $\sim 80\%$ of hub capital is delegated to native validator stake (Section 8.5), this physical movement would require costly bucket-deactivation and re-staking on every swap, with an epoch lag of ~ 2 days in each direction — erasing the economic benefit of yield integration entirely. The inter-bin ledger avoids this by recording each bin's net position as a *virtual claim* against the hub vault. When a swap moves SOL from bin A to bin B, the ledger entries L_A and L_B are updated atomically. The physical SOL stays delegated to its validator. Only when a bin's claim drifts beyond a threshold does the keeper trigger physical settlement.

7.2 Ledger definition

Definition 19 (Inter-bin claim ledger)**.** For a hub h with B bins, define the per-bin claim ledger

$$L_b = a_b - a_b^{(0)}, \qquad b \in \{1, \ldots, B\},$$

where a_b is bin b's current hub-asset claim (e.g., SOL lamports for the SOL hub) and $a_b^{(0)}$ is the baseline allocation at initialization or last settlement. L_b measures the net SOL flow into ($L_b > 0$) or out of ($L_b < 0$) bin b since baseline. Swap fees are credited to a separate *fee pool*

account F, which is not part of any bin's claim a_b. This ensures that fee accrual does not perturb the ledger conservation invariant.

Proposition 20 (Hub-wide ledger conservation)**.** *For any sequence of swaps on hub h,*

$$\sum_{b=1}^{B} L_b = 0.$$

The ledger sum is zero because every swap that increments one bin's claim decrements another's, and swap fees are credited to the fee pool F rather than to any bin's claim.

Proof. Each swap on hub h has gross volume G (the total amount the user deposits into the source bin) and a fee f (e.g. 0.05% of G). The fee is deducted from the gross amount and credited to the fee pool F. Only the net amount $N = G - f$ flows through the inter-bin ledger. The net transfer moves N hub-asset units from source bin b_s to destination bin b_d. The ledger updates are

$$L_{b_s} \leftarrow L_{b_s} - N, \qquad L_{b_d} \leftarrow L_{b_d} + N.$$

The sum $\sum_b L_b$ changes by $-N + N = 0$. Because fees are routed to F and never recorded in a_b, they cannot accumulate in any L_b. Induction across all swaps maintains the invariant. □

The fee pool F is a separate on-chain account per hub. Fees can be distributed to LPs, stakers, or the protocol treasury through governance-defined mechanics outside the scope of the inter-bin ledger. Importantly, fee distribution operations (e.g. claiming or compounding) also go through F and do not touch the L_b entries, preserving the conservation invariant.

7.3 Lazy keeper settlement

Physical settlement — moving capital between the hub vault and the stake-bucket set — is deferred until a threshold is breached. The keeper triggers settlement on bin b when $|L_b| \geq \tau \cdot a_b^{(0)}$: if $L_b > 0$ the bin has accumulated excess hub-asset claim and a stake-bucket deactivation of size L_b is scheduled (settling at the next epoch boundary), while if $L_b < 0$ the bin owes claim to others and $|L_b|$ is deployed from the hub vault into a fresh or existing stake bucket. Either way the baseline is reset, $a_b^{(0)} \leftarrow a_b$. The deactivation window is masked from the user by the deterministic buffer (Definition 22) and, for tail events, by rSOL issuance (Section 9). When an LP withdraws capital from the hub, every bin touched by the LP's position is settled regardless of threshold so that the LP's fair share of fee-pool F accruals is included. A periodic keeper tick every Δ_{tick} slots applies the same rule with a relaxed threshold $\tau/2$ to catch slowly drifting bins.

Default parameters. $\tau = 0.20$ (a bin's ledger drift exceeding 20% of its baseline triggers settlement). $\Delta_{\text{tick}} = 100$ slots (~ 40 seconds). Both are governance-tunable per hub.

Example 21 (Ledger in action)**.** Consider a SOL hub with 100M SOL total claims, 20M on-chain buffer (C^{active}), $B = 640$ bins, each with baseline $a_b^{(0)} = 31,250$ SOL. Over a busy trading day, net SOL flows cause bin 127 to accumulate $+6,250$ SOL (net SWAP-in) and bin 412 to lose $-5{,}000$ SOL (net SWAP-out). Swap fees are credited to the fee pool F and do not appear in any bin's L_b. Ledger entries:

$$L_{127} = +6,250, \qquad L_{412} = -5,000, \qquad \sum_b L_b = 0.$$

Since $|L_{127}| = 6,250 = 0.2 \cdot 31,250 = \tau \cdot a_b^{(0)}$, bin 127 is at threshold. The keeper triggers settlement: it schedules deactivation of a 6,250 SOL portion from one of the stake buckets; at

the next epoch boundary the SOL lands in the hub vault and the keeper resets $a_{127}^{(0)} = 37{,}500$. Bin 412 with $|L_{412}| = 5,000 < 6,250$ is below threshold — no settlement yet. The keeper monitors. If activity continues and L_{412} reaches $-7{,}000$, settlement fires for both bins in a single batched epoch-aligned transaction: the keeper combines the bin-127 deactivation with the bin-412 activation into one batched stake-program flow.

7.4 Settlement guarantees

- **LP fairness:** On exit, the LP's share of yield-route earnings and the fee pool F is included in the payout. The ledger is settled for all bins touched by the LP's position.
- **Solvency:** The hub invariant $V_{\text{vault}} + I_{\text{stake}} + F = \sum_b a_b$ is preserved at all times. Note that F is held outside the bin claim system and does not appear in any L_b; it is an additional asset of the hub that backs LP claims alongside the vault and the stake buckets.
- **Stockout risk:** If a withdrawal requires more SOL from the hub vault than is currently available, the request takes one of two routes: (a) the user accepts an rSOL mint as their exit receipt (Section 9), instantly; or (b) the request enters the bucket-deactivation queue and settles at the next epoch boundary. The periodic tick at Δ_{tick} slots ensures the keeper observes drift quickly; the actual unstaking is epoch-paced.
- **Bounded settlement lag:** Worst-case latency between a bin crossing the threshold and the keeper *scheduling* settlement is $\Delta_{\text{tick}} = 40$ seconds; the actual on-chain stake-program movement settles at the next epoch boundary (~ 2 days). User-facing latency is masked by the buffer and rSOL.

7.5 Ledger as a primitive

The inter-bin claim ledger is a general primitive that can be composed with other protocols. Because it maintains $\sum_b L_b = 0$ at all times (with fees independently accumulated in F), any external protocol that can read the bin accounts can verify the hub's solvency. This enables:

- **Transparent proof of reserves:** Anyone can sum $\sum_b a_b$ and compare it to $V_{\text{vault}} + I_{\text{stake}} + F$.
- **Credit delegation:** An external protocol can accept a bin's claim as collateral, knowing the claim is backed by physical SOL in the hub vault and the stake-bucket set.
- **Cross-protocol composability:** The ledger format is hub-agnostic (Phase 1 instantiates it for SOL only), so any future hub admitted under Phase 2 (Section 14.4) plugs into the same composability surface.

8 Active / Idle Buffer Sizing

The central thesis of Radial is that a DEX does not need all of its capital available for swaps at once. Capital that is not needed for the worst-case on-chain demand should be deployed to yield. This is formalized by deriving the on-chain buffer as a deterministic worst-case bound, then routing the surplus to the SOL hub's single Phase 1 yield route: direct native validator staking (Section 8.5).

8.1 Prior art: Meteora Dynamic Vaults

The closest production implementation is acknowledged: Meteora's DAMM v1 with Dynamic Vaults already deploys idle pool reserves to yield strategies via an off-chain keeper that monitors yield destinations and rebalances allocations. Per-pool idle yield is a proven pattern. Radial

generalizes the idea from per-pool to per-hub: the entire SOL hub's idle capital (not just individual pool buffers) is yield-eligible, and the yield route is a single, transparent path (direct native staking) rather than a multi-strategy aggregator.

8.2 Deterministic worst-case floor

Solana's Sealevel runtime executes transactions in 400ms slots. Within a slot, the maximum number of parallel non-conflicting transactions is bounded by the number of hardware threads available to the validator. $T_{\max}^{(h)}$ is a per-hub governance parameter stored as a `u8` (unsigned 8-bit integer, range 0–255), defaulting to 32. Conservative production validators run 32 logical cores for compute; while each core can process multiple instructions within a single transaction, bounding parallelism at 32 threads provides a safety margin against contention, scheduler overhead, and worst-case blocking.

Definition 22 (Deterministic active buffer)**.** For the SOL hub, the on-chain active buffer is

$$\boxed{C^{\text{active}} \;=\; T_{\max}^{(h)} \cdot K_{\max} \cdot s_{\max}\ ,}$$

where

- $T_{\max}^{(h)}$ is the maximum parallel non-conflicting transactions per slot, a `u8` governance parameter defaulting to 32;
- $K_{\max} = 10$ is the maximum swap CPIs per transaction;
- $s_{\max}$ is the per-swap maximum input size, set as a governance parameter ($\sim 0.0625\%$ of expected hub TVL is the default).

No statistical assumption underlies this bound. It is the maximum SOL value that any admissible block can drain from the on-chain vault in a single slot: $T_{\max}^{(h)}$ parallel transactions each issuing $K_{\max}$ swaps each capped at $s_{\max}$ per swap.

Example 23 (SOL hub: 100M SOL total claims)**.** A SOL hub with $\mathcal{L}(\text{SOL}) = 100\text{M SOL}$. Governance sets $T_{\max}^{(h)} = 32$, $K_{\max} = 10$, $s_{\max} = 31{,}250$ SOL ($\sim 0.0312\%$ of hub TVL). Then

$$C^{\text{active}} = 32 \cdot 10 \cdot 31{,}250 = 10{,}000{,}000 \text{ SOL} = 10\% \text{ of } \mathcal{L}(\text{SOL}).$$

Target active ratio $\rho^* = 0.10$. The remaining 90M SOL is yield-eligible and is delegated to validators via the bucket model (Section 8.5).

8.3 Empirical calibration of ρ

The default active-ratio band $\rho \in [0.10, 0.30]$ is not chosen by intuition. It is the operating range that empirically dominates the worst observed flow drawdowns across the six largest SOL-denominated Solana AMM architectures, measured over 150 consecutive days of on-chain reserves (Section 11). The lower bound, $\rho = 0.10$, equals the deterministic worst-case bound of Definition 22 for the default governance parameters; the upper bound, $\rho = 0.30$, exceeds the worst observed 3-day drawdown on the most volatile measured architecture (Meteora DLMM, -28.4%) by a 1.7 pp margin.

Remark 24 (2σ operational policy, 3σ tail). For ordinary operation, the keeper targets $\rho^* = 0.10$ with the empirical 2σ envelope: across the measured protocols, the day-over-day 2σ flow ranges from 3.6% (DBC) to 15.4% (DLMM). Empirical 3σ events — which occur on $\sim 1\%$ of days for the highest-volatility architectures — are absorbed by the rSOL withdrawal mint (Section 9) rather than by preemptive unstaking, eliminating the deepest tail of opportunity cost.

8.4 Active ratio and the keeper

The hub maintains an active ratio $\rho = V_{\text{vault}} / \sum_i a_i$, where V_{vault} is the on-chain vault balance and $\sum_i a_i$ is the sum of all pool claims. An off-chain keeper monitors ρ and rebalances when it exits the operating band $[\theta_L, \theta_H]$, targeting the midpoint $\rho^* = (\theta_L + \theta_H)/2$. When $\rho > \theta_H$, the excess $V_{\text{vault}} - \rho^*\Sigma$ is deployed to a fresh or existing stake bucket (decreasing V_{vault} and increasing I_{stake}). When $\rho < \theta_L$, the deficit $\rho^*\Sigma - V_{\text{vault}}$ is scheduled for deactivation from the bucket set, settling at the next epoch boundary. Every rebalance preserves the conservation identity $V_{\text{vault}} + I_{\text{stake}} + S_r \cdot R(t) = \sum_i a_i$. The v1 defaults are $\theta_L = 0.05$, $\theta_H = 0.15$, $\rho^* = 0.10$; the wider $[0.10, 0.30]$ band is the upper envelope governance may activate during periods of elevated empirical σ.

8.5 Yield route: native validator staking

The Phase 1 implementation deploys a single, transparent yield route: direct native validator staking, with the Solana Stake Program as the only counterparty. SOL is delegated through a *bucket model* — the hub maintains a small fixed number of stake accounts ("buckets"), each with a configurable size (default $\sim$ 100,000 SOL per bucket). Buckets are activated and deactivated independently, decoupling the per-bucket epoch cadence from individual user activity. Yield accrues to the hub directly through epoch rewards; no third-party LST issuer, no lending market, and no off-chain aggregator sits between the hub and the validator.

Hub	Integrated yield route (Phase 1)
SOL	Native validator staking only, via the bucket model. Withdrawals from a stake bucket are epoch-aware ($\sim$ 2 days); rSOL mint (Section 9) provides an instant-exit alternative for users on tail-event days.

Table 4: Phase 1 hub-to-yield-route mapping. The bucket model is the only default route; no LST wrapper and no lending market is active by default. The hub is governance-extensible: additional yield routes (e.g., a curated LST set) may be admitted by time-locked proposal once an extended adapter set is published.

Remark 25 (Why native staking, not LST). Native staking, at the time of writing, yields $\sim$ 6.8%; the deepest Solana lending markets yield $\sim$ 3.0–3.5% for SOL. The yield differential ($\sim 1.9\times$) means lending — the typical idle-capital route on existing yield-integrated AMMs — carries a $\sim$ 50% yield haircut for the same nominal capital. Direct staking also eliminates the LST issuer and the lending-market depositor base as counterparties; the only failure mode is validator slashing, mitigated by the bucket model's diversification across multiple validators.

8.6 Withdrawal-time mitigations

The SOL hub provides three layers of withdrawal handling, ordered from immediate to delayed:

1. *Buffer drain.* Withdrawals smaller than the on-chain buffer settle in the requesting block.
2. *rSOL mint (user opt-in).* Withdrawals exceeding the local buffer can be served instantly as rSOL — a transferable claim on the hub's active stake, redeemable for SOL on the next epoch or sold immediately on the secondary market (Section 9).
3. *Bucket deactivation queue.* Withdrawals routed through the keeper's bucket-deactivation flow settle at the next epoch boundary, with the keeper sized bucket count and bucket size to minimize the expected deactivation lag (Section 8.3).

The combination eliminates the need for a fully-liquid emergency cushion: tail withdrawals are absorbed by rSOL rather than by overprovisioning the on-chain buffer, recovering the 52% opportunity cost that a static 3σ buffer would impose on the most volatile architecture (Table 6).

8.7 POLP architecture and yield distribution

Native-staking yield accrued to the SOL hub is routed by default to a *protocol-owned liquidity pool* (POLP) which the protocol uses to provide additional swap liquidity inside the hub's mini-pools (token/SOL pairs). POLP-provided liquidity is priced at the start of each block, following the proportional-AMM convention used by GMX v2, Drift BAMM, and other deterministic-price systems — there is no continuous AMM curve on the POLP side, only a per-block snapshot price. Six invariants govern this distribution:

- **Permissioned POLP deposits.** Only the protocol may add capital to the POLP; user deposits are routed exclusively to the standard LP pool. The dominant POLP feed source is yield reconciliation (positive yield delta from the native staking adapter); secondary feeds (e.g., reserved swap fees) are protocol-discretion.
- **Configurable LP/protocol split.** A per-hub parameter $\beta \in [0, 1]$ (governance, default $\beta = 1$) determines the fraction of *positive* yield delta routed to POLP; the complement $(1 - \beta)$ accrues to V_{vault} and lifts per-share LP value. With the default $\beta = 1$, LPs do not receive yield through per-share appreciation; LP claims on yield-derived value are purely indirect, mediated by the protocol's discretionary POLP redistribution.
- **Asymmetric loss policy.** Negative yield delta (slashing loss) falls entirely on V_{vault}; POLP serves as a one-way protocol-side reserve and is never drawn down to absorb LP losses. LPs accept this asymmetry in exchange for the chance of POLP-derived rewards (next invariant) and IL protection.
- **Discretionary POLP distribution.** Accrued POLP capital is redistributed to LPs at the protocol's discretion. The default heuristic prioritises the longest-holding LPs — providing both a holding incentive and built-in compensation for impermanent loss (IL) accumulated over the holding period — but the protocol retains full authority to adjust weights, gate distributions, or redirect POLP capital to other protocol-level uses (e.g., insurance top-ups, liquidity bootstraps for new spoke admissions). LPs hold a *contingent claim* on POLP rewards, not a per-share entitlement.
- **LP token semantics unchanged.** LPs continue to receive standard SPL LP tokens whose redemption value tracks $V_{\text{vault}} + I_{\text{stake}}$ (principal). LP tokens remain fully transferable and composable; the contingent POLP claim travels with the token holder.
- **Keeper bookkeeping invariant.** Because keeper deposit/withdraw operations on the native-staking adapter preserve principal accounting unchanged, no on-chain reconciliation of yield-implied principal increase is required when capital cycles in and out of stake buckets; yield is captured exclusively via explicit reconciliation calls, and the keeper never needs to adjust deposit or withdraw amounts to add yield-implied gains.

This design separates two concerns that are typically conflated in DeFi yield aggregators: *capital efficiency* (where idle capital is parked) and *LP reward economics* (how earnings are distributed). Routing the entirety of yield to POLP and redistributing via a discretionary, duration-prioritised mechanism side-steps the impossible-fairness problem of statically dividing single-asset yield (e.g., SOL staking rewards) across thousands of mini-pools that each have heterogeneous TVL, volume, and holding profiles. Deferring the fairness decision to redemption time — when the relevant per-LP holding history and protocol context (loss events, IL incidence, hub utilisation)

are known — lets the protocol resolve the trade-off retrospectively rather than committing to a brittle a-priori formula.

8.8 Mini-pool architecture

The hub provides infrastructure (yield routing, liquidity reserve); user-facing token exchange takes place in *mini-pools*, each pairing one non-hub token with SOL. Token-to-token swaps not directly involving SOL route through two mini-pools, with the hub serving as a virtual common denominator: no physical SOL movement occurs during a cross-pool swap, only ledger updates to each mini-pool's claim on the hub's buffer.

Pricing. Each mini-pool has its own geometric bin grid (with a governance-set bin step). Bin pricing reuses the hub's intra-pool model (Section 5); the bin grid is independent per mini-pool, allowing different volatility profiles for different token pairs.

Cross-pool swap. A user swap from token A to token B (both with mini-pools on the SOL hub) consumes A in mini-pool A, computes the equivalent SOL amount through A's bin pricing, applies the symmetric ledger update to mini-pool B's claim on the hub, and releases B in mini-pool B at B's bin price. The hub's physical balance is unchanged — the aggregate hub-claim across all mini-pools is preserved across the swap.

Single-pair swap. A direct exchange against one mini-pool does involve the hub's physical balance: swapping SOL into the mini-pool's token grows the hub's idle balance and the mini-pool's claim symmetrically; the reverse direction shrinks both. Cross-pool swaps net these claims without touching the hub balance.

Mini-pool LP withdraw. A pair LP burns their LP token and receives a pro-rata share of the mini-pool's token plus an optional SOL share paid from the hub's idle balance (clamped to availability, with rSOL fallback per Section 9).

9 rSOL: Withdrawal-Time Liquid Staking Receipt

Native validator staking gives the SOL hub the highest sustained on-chain yield available for SOL ($\sim$ 6.8% APY, Remark 25), but at a cost: deactivating a stake bucket takes one epoch ($\sim$ 2 days) before the SOL becomes spendable. Ordinary withdrawal flow is unaffected — the deterministic buffer of Definition 22 absorbs all expected daily demand, and the keeper preemptively deactivates buckets ahead of the empirical 2σ envelope (Section 8.3). What *is* affected is the deep tail: 3σ events on the most volatile architectures consume up to 52% of available capital in opportunity cost if absorbed by a static buffer alone (Table 6).

The rSOL withdrawal mint is the mechanism that lets Radial size the operational buffer at the 2σ envelope while still serving tail withdrawals instantly. It is an opt-in, withdrawal-time issuance of a transferable liquid-staking receipt against the hub's active validator stake.

9.1 Motivation and design goals

Three goals shape the design:

1. **Default path stays simple.** A user who deposits and later withdraws receives SOL the way they expect, in the same block, as long as the hub's buffer covers it. rSOL is never forced.

2. **Tail events do not force preemptive overprovisioning.** Without rSOL, the hub would need a 3σ buffer to handle the worst observed 3-day drawdowns; this would tie up 11–52% of capital depending on architecture (Table 6). With rSOL, the operational buffer can target 2σ and the residual tail is served through receipt issuance.

3. **No new yield route.** rSOL is backed by the same native validator stake that the hub already operates (Section 8.5). It is a *claim wrapper*, not an additional protocol exposure.

9.2 Token definition

Definition 26 (rSOL)**.** rSOL is an SPL token whose mint authority is the SOL hub PDA. Its supply is denoted $S_r(t)$, and the protocol maintains a redemption rate

$$R(t) \;=\; \frac{V_{\text{vault}}(t) + I_{\text{stake}}(t)}{S_r(t) + \tau(t)},$$

where $V_{\text{vault}}(t)$ is the on-chain hub SOL balance, $I_{\text{stake}}(t)$ is the total SOL delegated through stake buckets, and $\tau(t)$ is the implied SOL-equivalent of all outstanding LP claims that are not rSOL. At genesis $S_r(0) = 0$ and $R(0) = 1$ by construction. $R(t)$ is monotonically non-decreasing in expectation: it rises as epoch staking rewards accrue, and is unchanged by ordinary swap or deposit flow.

Remark 27 (Invariant). The accounting invariant is

$$\underbrace{V_{\text{vault}}(t)}_{\text{on-chain SOL}} + \underbrace{I_{\text{stake}}(t)}_{\text{delegated SOL}} = \underbrace{\tau(t)}_{\text{LP-token-backed SOL}} + \underbrace{S_r(t)\cdot R(t)}_{\text{rSOL-backed SOL}}.$$

Every SOL held or staked by the hub is claimable by exactly one of: (a) a regular LP token, (b) a unit of rSOL, or (c) a pending bucket-deactivation queue entry (treated as still-staked until the epoch settles). The invariant is checked at the end of every keeper handler and at every user withdrawal.

9.3 Mint flow

A user-initiated withdrawal carries a preference parameter that selects one of three settlement paths: *SOL-only* (legacy, transfers SOL from the buffer or reverts with `InsufficientBuffer` when the buffer cannot cover the requested amount), *rSOL-only* (opt-in, mints fresh rSOL against the user's LP claim and always succeeds), and *SOL-or-rSOL* (default, takes the SOL path when $V_{\text{vault}} \geq$ `amount` and falls back to rSOL otherwise). The handler first checks that the requested amount `amount` does not exceed the user's LP-token entitlement E in SOL units. On the rSOL path the protocol mints $\texttt{amount}/R(t)$ rSOL to the user, leaves V_{vault} untouched, and decrements the LP-backed quantity $\tau(t)$ by `amount` to preserve the conservation identity of Remark 27. On the SOL path the buffer is debited as usual. After dispatching either path the handler re-evaluates the invariant and reverts on any mismatch.

No keeper intervention required. An rSOL mint is purely an accounting operation: the hub's SOL position on validators is unchanged, the user's LP token is burned, and a corresponding rSOL position is created. The keeper learns about the new outstanding rSOL claim only through the eventual redemption (next subsection); no preemptive stake deactivation is needed.

9.4 Redemption flow

A holder of rSOL has two exit paths:

1. **Burn at the hub.** The user submits an on-chain burn of b rSOL. The hub records a pending claim of $b \cdot R(t)$ SOL and adds it to the next bucket-deactivation batch; the keeper deactivates a bucket of at least the required size; after one epoch the user receives $b \cdot R(t_{\text{settle}})$ SOL — denominated at the redemption rate *at settlement*, so any staking rewards accrued in the intervening epoch flow to the redeemer. This path is the canonical exit; the user takes no peg risk and earns the epoch yield.

2. **Secondary-market sale.** rSOL is a standard SPL token and is freely transferable. A user who prefers instant SOL can sell rSOL on an external venue (a SOL hub spoke mini-pool for rSOL, or a third-party AMM that supports the token) at whatever discount the secondary market demands. This path is instant but introduces peg risk: the user pays the spread between $R(t)$ and the market clearing price.

9.5 Economic analysis

Peg dynamics. The protocol redemption rate $R(t)$ is a hard floor: the burn-at-the-hub path always pays $R(t_{\text{settle}})$ SOL per rSOL within one epoch, with the only risk being a validator slashing event on the hub's bucket set (Section 14). Secondary-market prices therefore have a one-epoch arbitrage anchor; the market price can dip below $R(t)$ only by an amount that compensates for the one-epoch holding cost. In the typical case where the secondary market has $\gtrsim$ 1,000 SOL of depth on rSOL/SOL, the empirical peg discount is bounded by the burn-redeem holding cost, generally a few tens of basis points.

Buffer relief and yield uplift. Each rSOL mint defers exactly one withdrawal that would otherwise force a stake bucket to deactivate. In aggregate, the hub can therefore operate at the empirical 2σ buffer (rather than 3σ) without compromising solvency. From Table 6 the buffer-relief yield uplift is 14–30 pp on the most volatile architectures (DLMM, AMM v4) and 4–8 pp on the rest. On a 100M SOL hub at 6.8% APY, a 10 pp uplift in deployed capital corresponds to $\sim$ 680,000 SOL per year of additional yield — \$136M at \$200/SOL.

User-facing trade-off. An rSOL exit is strictly weakly preferable to the alternative of waiting at the keeper's deactivation queue: the user holds a transferable, yield-bearing claim instead of a non-transferable IOU. The only path that gives the user nothing they did not already have is *Sol-or-revert* with insufficient buffer; under default `SolOrRsol` preference, the user always gets an instant settlement (in SOL or in rSOL), preserving deterministic withdrawal UX.

9.6 Risk model

Three risks are inherited or introduced by rSOL:

- **Validator slashing.** rSOL holders share, pro rata, in any slashing event on the bucket set. This is the same exposure that all LP token holders already carry; rSOL inherits it without amplifying it.

- **Secondary-market depth.** If the rSOL/SOL secondary market is thin during a tail event, users who prefer instant SOL face a wider peg discount. The fallback — burn-at-hub — still resolves within one epoch at $R(t_{\text{settle}})$.

- **Mint-supply overflow.** The hub enforces $S_r(t) \cdot R(t) \leq I_{\text{stake}}(t)$ at every mint to guarantee solvency on the burn path. A withdrawal preference of `Rsol` reverts if the cap is reached; this is the hard ceiling on outstanding rSOL claims.

9.7 Integration with the keeper

The keeper rebalance protocol of Section 8 is unchanged for the buffer-side branches; one additional branch handles outstanding rSOL claims:

- Each pending rSOL burn settles via the same bucket-deactivation flow that BucketDeactivate$(\cdot)$ uses for direct $\rho < \theta_L$ rebalances; the keeper batches outstanding rSOL burns and ordinary deactivations into a single epoch operation.
- The keeper does not pre-mint rSOL: minting is strictly user-driven through the withdrawal handler. The keeper's only responsibility on the rSOL path is to ensure that an outstanding burn claim has a corresponding pending deactivation.

9.8 Implementation status

The rSOL mechanism is specified here as a Stage 18 roadmap item against the current `radial-dex` implementation. The deterministic buffer, the keeper rebalance algorithm, and the native-staking bucket model (Stages 1–17) are all in production; the rSOL extension extends the withdrawal handler and the keeper without altering the core inter-bin ledger or the swap path. Section 14 discusses the deployment sequence.

10 Manipulation Resistance

Radial derives its swap price from the reserve-implied ratio of each bin's token reserves, analogous to Meteora DLMM and other discrete-bin AMMs. Because price is a function of the bin's on-chain state rather than an external feed, the manipulation surface is fundamentally different from oracle-dependent designs.

10.1 Threat model

Definition 28 (Bin-depletion attack)**.** An attacker executes a trade that depletes a bin's token reserve in the direction of a target price, then captures the displaced price in a subsequent trade against the same or a neighbouring bin.

Definition 29 (Sandwich-style attack)**.** A trader observes a pending transaction T_n that moves the active bin's reserves, then places transactions T_{n-1} and T_{n+1} to extract value from the price movement caused by T_n. This is the standard AMM sandwich pattern.

10.2 Defence mechanisms

(1) Bin traversal cost. Because price moves only by consuming reserves within a bin, any attempt to move the price significantly must traverse multiple bins. Each bin transition incurs a swap fee, creating a cumulative cost that grows linearly with the price impact. For a target price movement of ΔP, the attacker pays at least $f \cdot \Delta P$ in fees across the traversed bins.

(2) Minimum-TVL gate (Phase 2 proposed). A program-wide minimum-TVL gate is described in the original design: a hub would be inactive for swaps until its TVL exceeds a configurable floor (e.g., \$1M or equivalent), with only LP join/exit operations permitted below the gate. This mechanism is *not* enforced in the Phase 1 implementation; below-gate hubs would still be open to swaps. The gate is a Phase 2 candidate that would raise the per-attack capital floor without dependence on an external feed; in Phase 1 the same goal is approached through arbitrage and per-transaction user safeguards (next paragraphs).

(3) Arbitrage closes drift. If Radial's price drifts above an external venue's, an arbitrageur buys cheaper externally and sells to Radial (or vice versa), capturing the difference minus fees. The arbitrageur's profit is the externalization of the manipulation tax: every manipulator pays the next arbitrageur.

(4) Per-transaction user safeguards. Independent of the gate, every user-initiated swap on Radial is parameterized with three caller-supplied guards enforced by the program at the start of each handler:

- **Deadline slot.** The caller specifies a slot number after which the transaction must revert. Validator-side reordering or relay delays that push the transaction past the deadline cause an automatic revert, foreclosing the standard "delayed-execution" MEV pattern in which an attacker holds the transaction until a favorable bin state appears.
- **Output slippage bound.** The caller specifies a minimum acceptable output ($\Delta_{\text{out}}^{\min}$). If the realized output is below this bound, the swap reverts. This caps the price impact a sandwich attempt can inflict on a single user before they refuse the trade.
- **Intermediate hub-leg bound (cross-bin / cross-token).** For two-leg cross-token swaps that net through the hub asset, the caller specifies a minimum acceptable hub-leg intermediate. If the first leg yields below this bound, the second leg does not execute. This closes the "leg-1 squeeze" pattern where an attacker manipulates one leg's bins between the legs of a multi-step user swap.

None of these guards depend on an external price feed; they constrain the per-user execution surface against worst-case adversarial ordering within the slot.

Combined effect. An attacker materially moving the price must commit at least the level set by arbitrage closure (the lower bound) and pay cumulative bin-traversal fees, while the per-user safeguards above prevent extracting that value at the expense of any specific user. Arbitrageurs close any exploitable gap within the same slot, and the cost of a sustained manipulation campaign typically exceeds the profit for all but the deepest-pocketed attackers.

10.3 Reserve-implied pricing

Radial does not rely on external oracles for price discovery. Instead, the bin price P_n for bin b is the reserve-implied ratio

$$P_n = \frac{\text{reserve of hub-asset}}{\text{reserve of token}}$$

computed directly from the bin's on-chain state (R_b, a_b). This is the same pricing mechanism used by Meteora DLMM and Trader Joe LB; it is manipulation-resistant because the price is derived from reserves that the attacker must physically move. In the Phase 1 implementation, the absence of any oracle for intra-hub pricing eliminates the oracle-manipulation attack surface for the dominant flow entirely. There is no external feed to lag, front-run, or price-asynchronously manipulate across multiple slots. (A Phase 2 follow-up may opt into Pyth or Switchboard feeds for cross-hub legs across heterogeneous mints; the discussion is in §14.)

11 Empirical Idle-Capital Analysis

The motivating premise of Radial — that the vast majority of capital sitting in Solana AMM pools never participates in trade — is validated against 150 consecutive days of on-chain indexed data covering the six largest SOL-denominated AMM pool architectures: Meteora DLMM, Raydium CLMM, Meteora DAMM v2, Raydium AMM v4, Meteora Dynamic Bonding Curve

(DBC), and Raydium CPMM. The dataset spans 2025-12-11 to 2026-04-30 (Q4 2025 – Q1 2026), aggregating per-protocol daily minimum and maximum SOL reserves across all active pools.

11.1 Methodology

For each protocol π and each day d in the window, two reserve aggregates are computed:

- $\min_\pi(d)$ — the minimum total SOL reserve held across all active π pools, sampled every block and taken as the daily lower envelope.
- $\max_\pi(d)$ — the corresponding daily upper envelope.

The *idle floor* is the day's $\min_\pi(d)$: capital that did *not* move during the entire 24-hour interval. The *idle ratio* is $\min_\pi(d)/\max_\pi(d) \in [0,1]$: a value close to 1 means most of the day's TVL is sitting still; a value close to 0 means liquidity is heavily transacted. Day-over-day relative changes $\Delta \min_\pi(d) = (\min_\pi(d) - \min_\pi(d{-}1))/\min_\pi(d{-}1)$ are used to estimate volatility and tail events.

11.2 Per-protocol summary

Across the 150-day window the six protocols exhibit consistently high idle ratios:

Protocol	**Mean idle (M SOL)**	**Idle ratio**	**σ daily**	**Worst 3d**	**K_σ req.**
Meteora DLMM	61.9	88%	7.68%	−28.4%	2.14
Raydium CLMM	60.0	95%	3.64%	−18.2%	2.88
Meteora DAMM v2	12.1	98%	1.84%	−17.1%	5.38
Raydium AMM v4	10.0	98%	2.83%	−24.7%	5.05
Meteora DBC	3.4	96%	1.79%	−1.9%	0.61
Raydium CPMM	3.4	99%	4.17%	−16.4%	2.27
Total / range	**150.8**	**88–99%**	—	**−28.4%**	—

Table 5: Per-protocol empirical buffer characteristics over Q4 2025 – Q1 2026 ($n = 150$ days). K_σ is the minimum sigma multiple of σ_{daily} required to cover the worst observed 3-day drawdown; values above 3 indicate the worst event sits in a $> 3\sigma$ tail relative to typical daily volatility.

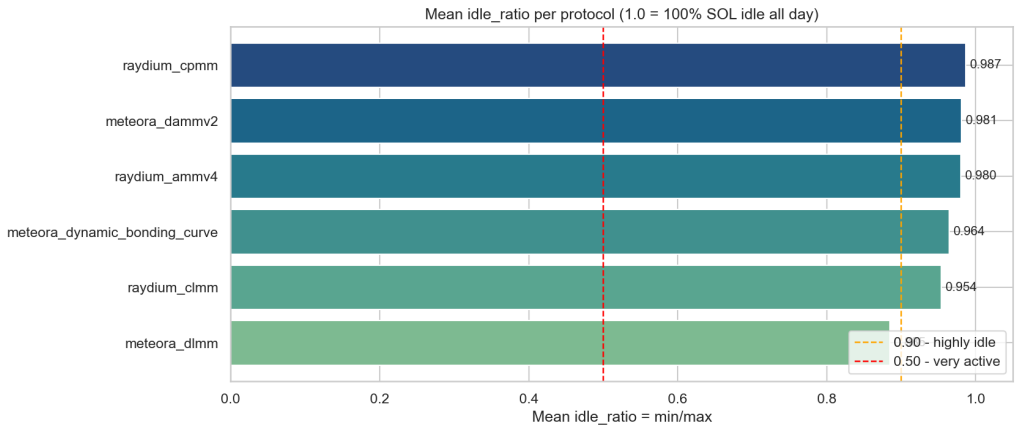

Figure 2: Mean idle ratio per protocol over the 150-day window. All six Solana AMMs hold $\geq 88\%$ of their SOL TVL idle on a daily basis; the long-tail protocols (DAMM v2, AMM v4, DBC, CPMM) approach 99%.

11.3 Market totals

The aggregate mean idle floor across all six protocols is 150.8M SOL — approximately $30.2B at $200/SOL — sitting unused on the median day. At the blended SOL yield assumption used throughout this paper (6.17% APY, blending native staking and reference LST routes), this idle capital represents an annual opportunity cost of 9.30M SOL ($\sim$ $1.86B). Meteora DLMM and Raydium CLMM alone account for 81% of this idle floor.

Remark 30 (Why the idle floor matters for Radial). The *minimum* daily SOL reserve — not the average — is the quantity that determines how much capital can be safely deployed to yield without ever interrupting a swap. By construction, $\min_\pi(d)$ is the floor that held throughout day d; any SOL above this floor moved at some point during the day. Radial's yield-first allocation rule mirrors this exactly: keep an on-chain buffer that covers the worst observed intra-window drawdown, deploy the rest. The empirical idle ratio of 88–99% across six unrelated AMM architectures is direct evidence that a 20–30% active buffer (Radial's default $\rho \in [0.10, 0.30]$) leaves substantial yield headroom.

11.4 Buffer sigma requirements

The minimum sigma multiple required to cover the worst observed 3-day drawdown (K_σ column, Table 5) ranges from 0.61 (DBC — the worst event sits well within typical noise) to 5.38 (DAMM v2 — the worst event is a deep tail relative to ordinary days). High-K_σ protocols have low day-to-day σ but were occasionally hit by concentrated single events; low-K_σ protocols either have persistently choppy flows or never experienced an event outside their typical band.

For Radial's SOL hub, which aggregates flow across many spoke tokens, the relevant figure is the cross-protocol envelope: a $K = 3$ sigma buffer simultaneously covers the worst observed 3-day drawdown of every protocol except DAMM v2 and AMM v4, whose worst-case tails exceed any practical static-sigma buffer.

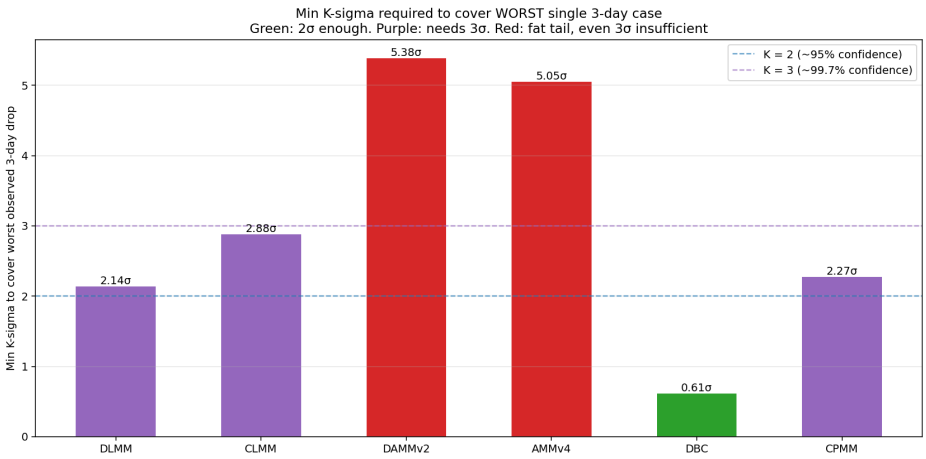

Figure 3: Minimum K_σ required to cover each protocol's worst observed 3-day drawdown. Green: $K \leq 2$ (typical noise covers the tail); purple: $K \leq 3$ (mild tail); red: $K > 3$ (fat tail relative to daily volatility — a static-sigma buffer is impractical, motivating dynamic relief).

11.5 Opportunity cost across buffer strategies

Sizing the buffer too generously wastes the very capital this paper aims to deploy. The empirical opportunity cost — the fraction of mean idle capital reserved as buffer instead of earning yield — is tabulated for three strategies:

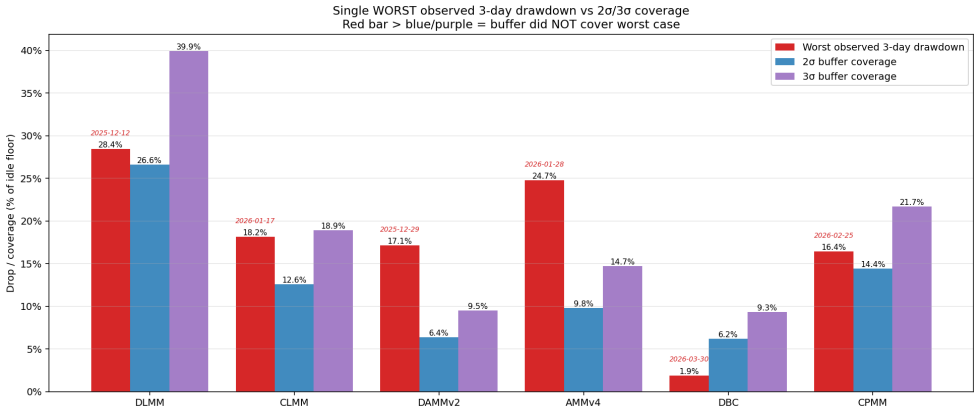

Figure 4: Worst observed 3-day drawdowns versus 2σ and 3σ buffer coverage envelopes. Red bars annotate the historical worst case and date. Meteora DLMM is the only protocol whose worst 3-day move (-28.4% on 2025-12-12) sits within a 3σ band; DAMM v2 and AMM v4 would require a $5\sigma+$ static buffer to cover their worst events, which is impractical and motivates the rSOL withdrawal-relief mechanism (Section 9).

Protocol	**Perfect oracle**	**2σ buffer**	**3σ buffer**
Meteora DLMM	20.2%	38.9%	52.2%
Raydium CLMM	7.6%	17.4%	23.7%
Meteora DAMM v2	3.7%	8.3%	11.5%
Raydium AMM v4	3.9%	11.8%	16.7%
Meteora DBC	3.2%	9.0%	12.1%
Raydium CPMM	2.7%	15.8%	23.0%

Table 6: Opportunity cost (% of idle floor reserved as buffer) by strategy. Perfect oracle is a clairvoyant upper bound; 2σ and 3σ are realistic operational policies. Values derived from on-chain indexed reserves.

A 3σ buffer recovers between 48% (DLMM, the hardest case) and 97% (CPMM) of capital for productive deployment. The remainder of this paper assumes a 2σ operational buffer on the SOL hub, with the rSOL withdrawal-relief mechanism (Section 9) absorbing residual tail events.

11.6 Implications for the Radial design

The empirical analysis directly informs the design:

1. **Active ratio range** $[0.10, 0.30]$**.** The default ρ range is set at the observed 1–3σ envelope: a 10% floor accommodates the σ_{daily} band, while a 30% ceiling absorbs the worst three-day drawdown observed on the most volatile architecture (Meteora DLMM, -28.4%).

2. **rSOL as tail relief.** For events beyond 3σ — which empirically occur on $\sim 1\%$ of days for DLMM-class protocols — the rSOL withdrawal mint defers physical SOL settlement until the next epoch, eliminating the need to pre-stake-down deeply (Section 9).

3. **Single-hub sufficiency.** The cross-protocol idle floor of 150.8M SOL exceeds, by an order of magnitude, the deepest single drawdown event. A single SOL hub holding even a modest share of cross-protocol flow has more than enough buffer capacity for the worst observed historical events.

12 Numerical Examples

Two concrete examples illustrate the inter-bin ledger conservation rule and the rSOL withdrawal mechanics.

12.1 Inter-bin ledger in action

Example 31 (Ledger settlement)**.** SOL hub with $B = 640$ bins, each $a_b^{(0)} = 31{,}250$ SOL (20M / 640). Over 24 hours: Bin 127: net +8,000 SOL ($L_{127} = +8{,}000$). $|L_{127}|/a_{127}^{(0)} = 8{,}000/31{,}250 = 0.256 > \tau = 0.20$. Keeper schedules deactivation of an 8,000 SOL portion from one of the stake buckets; at the next epoch boundary the SOL transfers into the hub vault and the keeper resets $a_{127}^{(0)} = 39{,}250$. Bin 412: net −12,000 SOL ($L_{412} = -12{,}000$). $|L_{412}|/a_{412}^{(0)} = 12{,}000/31{,}250 = 0.384 > \tau$. Keeper activates a fresh 12,000 SOL stake bucket from the hub vault, then resets $a_{412}^{(0)} = 19{,}250$. Verification: $\sum L_b$ after both settlements is $0 + 0 = 0$. The settlements batch into a single epoch-aligned stake-program transaction.

12.2 rSOL withdrawal: tail-event scenario

Example 32 (rSOL mint during a tail withdrawal)**.** A SOL hub at steady state: $V_{\text{vault}} = 20\text{M}$ SOL, $I_{\text{stake}} = 80\text{M}$ SOL, no outstanding rSOL ($S_r = 0$, $R(t) = 1.04$ after several months of accrued staking rewards). A market shock triggers 25M SOL of withdrawal requests over a few hours, exceeding V_{vault} by 5M. The keeper has already deactivated as much as the bucket cadence allows in this epoch. The 5M shortfall is absorbed as follows. Users who entered with preference `SolOrRsol` (the default) and whose individual requests cannot be served in full from the buffer receive their balance as rSOL: $5\,\text{M}/R(t) = 5\,\text{M}/1.04 \approx 4.808\,\text{M}$ rSOL minted across the affected users. The hub's accounting:

$$\begin{aligned} V_{\text{vault}} &\to 0 \\ I_{\text{stake}} &\to 80\,\text{M} \quad \text{(unchanged — no preemptive deactivation)} \\ S_r &\to 4.808\,\text{M} \\ S_r \cdot R(t) = 5.0\,\text{M SOL backing} &\quad \le I_{\text{stake}}. \end{aligned}$$

Users who accept rSOL hold a transferable, yield-bearing claim; users who prefer to wait enter the bucket-deactivation queue and receive SOL after the epoch boundary at $R(t_{\text{settle}})$. No preemptive overprovisioning of the buffer was required to absorb the tail.

13 Comparative Analysis

For the Solana cohort, the 150-day on-chain dataset of Section 11 provides directly measured idle ratios and worst-case 3-day drawdowns across the six largest SOL-denominated AMM architectures, contrasted with the Radial design target. Idle ratio is the median-day fraction of SOL TVL that does not move; "Capital earning yield" is the fraction of pool capital actively deployed to a native-staking or LST route during the window.

14 Discussion and Limitations

14.1 Limitations

Oracle dependency. The Phase 1 design — the current production target — is oracle-free. Swap rates are reserve-implied from each bin's geometric grid position (Section 5), and intra-hub cross-token routing (Section 6) is settled atomically via the inter-bin ledger without consulting

Solana design	Mean idle ratio	Worst 3d drawdown	Capital earning yield
Meteora DLMM	88%	−28.4%	0%
Raydium CLMM	95%	−18.2%	0%
Meteora DAMM v2	98%	−17.1%	0% (pool reserves)
Raydium AMM v4	98%	−24.7%	0%
Meteora DBC	96%	−1.9%	0%
Raydium CPMM	99%	−16.4%	0%
Radial v1 (target)	—	covered by 3σ buffer + rSOL	**$\sim$ 80%**

Table 7: Empirical idle ratio and worst observed 3-day drawdown across the six largest SOL-denominated Solana AMM architectures over Q4 2025 – Q1 2026, contrasted with the Radial design target. All six measured designs scored zero on yield deployment during the window.

any external feed. This eliminates the canonical oracle-manipulation surface (lag, front-running, confluence attacks) entirely. A Phase 2 follow-up may introduce optional Pyth or Switchboard feeds to support routing across multiple hubs under specific topologies (e.g., if a future stablecoin hub is admitted and a hub-to-hub bin pool needs to be priced against an external reference). Should Phase 2 adoption proceed, two failure modes apply only to the affected cross-hub legs: (i) feed outage halts those particular routes; (ii) feed manipulation propagates to those routes. Intra-hub swaps, the dominant flow, remain unaffected.

No LP price-range selection. The elimination of LP-side weights and bin ranges is by design, but removes a degree of freedom that sophisticated LPs value. A market maker who wants to concentrate liquidity at a specific price point cannot do so on Radial — all LPs are pooled and execution prices follow the reserve-implied geometric grid (Section 5). The yield supplement carried by the SOL hub's native staking allocation compensates for the inability to earn tighter bid-ask spreads via active range management.

Epoch-locked withdrawal latency. Native validator staking has a $\sim$ 2 day deactivation window: SOL delegated to a stake bucket is unspendable until the next epoch boundary closes. The deterministic buffer (Definition 22) and the 2σ operational policy (Remark 24) absorb expected daily flow, and the rSOL withdrawal mint (Section 9) absorbs tail events without forcing preemptive overprovisioning. A residual edge case — a sustained, market-wide withdrawal pressure exceeding the rSOL secondary-market depth — would manifest as an rSOL peg discount; the protocol-guaranteed burn-at-hub path still resolves at $R(t)$ within one epoch.

rSOL peg. rSOL holders who want instant SOL via the secondary market accept whatever discount the market clears at. The protocol redemption rate $R(t)$ is a floor (Section 9), but during a tail event the market may price rSOL/SOL at a meaningful spread to $R(t)$. This is the same trade-off all liquid-staking receipts face under stress conditions; Radial inherits it without amplifying it.

MEV and front-running. The inter-bin ledger records aggregate flows but does not prevent individual swap-level MEV. A searcher observing a pending swap to bin b_s can front-run with a swap that depletes a_{b_s}, manipulating the effective rate. Phase 1 mitigations enforced on-chain: per-transaction user safeguards (deadline slot, output slippage bound, intermediate hub-leg bound), bin traversal cost (cumulative fees), and arbitrage that closes any exploitable drift. The inter-bin ledger additionally makes any manipulation transparent and costly to sustain. A minimum-TVL gate raising the per-attack capital floor is a Phase 2 candidate (Section 10).

14.2 Open problems

1. **Keeper-lag risk.** Formalizing the worst-case spot drift achievable when the keeper's threshold-triggered rebalance lags the underlying flow.
2. **Optimal bucket sizing.** Determining the optimal number and size of stake buckets as a function of expected daily withdrawal volume and rSOL secondary-market depth.
3. **Cross-program composability.** Designing safe interfaces for other Solana programs to compose Radial's swap as a primitive.
4. **Dynamic buffer sizing.** Adapting C^{active} based on observed swap volume rather than hard-coding it at $T_{\max} \cdot K \cdot s_{\max}$.
5. **Adaptive ρ.** An online policy that widens ρ during periods of elevated empirical σ and tightens it during calm regimes, calibrated against the 150-day rolling envelope.

14.3 LP risk matrix

Risk vector	Description
Slashing risk (native staking)	Validator misbehavior on a stake bucket reduces the underlying lamports. Mitigated by validator diversification across multiple buckets (Section 8.5).
rSOL peg risk	Secondary-market rSOL/SOL price below the protocol redemption rate $R(t)$ during tail-event withdrawals. The burn-at-hub path is unaffected; only users who require instant SOL within the same block bear this discount. Bounded above by the one-epoch holding cost (Section 9).
Epoch-deactivation latency	Native staking has a $\sim$ 2 day deactivation window. Mitigated by the deterministic buffer ($\rho \in [0.10, 0.30]$) and rSOL tail relief. Residual exposure: simultaneous market-wide withdrawals exceeding rSOL secondary-market depth.
MEV / sandwich	Per-transaction safeguards (deadline slot, output slippage bound, intermediate hub-leg bound) cap the value extractable from any single user; bin traversal cost and arbitrage close residual drift. Residual exposure is validator-level reordering within a slot.
Keeper failure	If keepers are offline, ρ drifts outside its band and bins do not get rebalanced; LP redemption is constrained by instantly-recallable balance. Mitigated by keeper whitelist redundancy and the emergency-paused flag (admin can quiesce the system pending recovery).
Admin compromise	Misuse of the admin authority could rotate keepers, register a hostile spoke, or set parameters that drain value. Mitigated by the two-step admin handoff (Section 4.5); a Phase 2 governance time-lock is planned.
Yield volatility	Native staking yields fluctuate with the global validator-reward emission schedule and the network-level inflation parameter; projected returns are not guaranteed.
Oracle outage (Phase 2 only)	Conditional: applies only to cross-hub legs that may opt into an external price feed in a Phase 2 follow-up. Phase 1 swaps are reserve-implied and are not affected by feed availability.

Table 8: Primary risk vectors for Radial LPs (Phase 1 unless otherwise noted).

14.4 Phase 2: multi-hub extension

The hub primitive (Definition 5) is general; Phase 1 instantiates it once with SOL as the canonical hub. A Phase 2 follow-up may admit additional hubs — the natural candidates are

USD-denominated stablecoin hubs (e.g., USDC, USDT) — to extend Radial's reach to flows that today would be served by external stable-AMMs. Such a Phase 2 extension would introduce three additional concerns absent from the Phase 1 SOL-only scope:

1. **Hub-to-hub routing.** A swap between a SOL-hub spoke and a stablecoin-hub spoke would require either an internal hub-to-hub bin pool or an external oracle leg.
2. **Hub-specific yield routes.** Stablecoin hubs would need a yield route different from native staking (lending markets and DeFi yield aggregators are the most natural candidates).
3. **Cross-hub solvency invariants.** The current single-hub solvency proof (Remark 27) would need to be extended to a per-hub invariant set plus an explicit non-pollution rule on cross-hub claims.

None of these is technically novel — multi-hub yield-integrated DEXs have been previously described in the design space — but each adds enough surface area that they are deliberately excluded from the production Phase 1 scope to keep the manipulation analysis (Section 10) and the empirical buffer calibration (Section 11) sharp.

14.5 Phase 2 planned

The following items are scoped for post-launch iteration. They are not part of Phase 1 but are concrete enough to have implementation paths sketched:

1. **Optional external price feeds.** Pyth or Switchboard integration for cross-hub legs once Phase 2 multi-hub is admitted (Section 14.4).
2. **TWAP overlay.** A time-weighted average price snapshot maintained on-chain as an optional secondary manipulation check for cross-hub flow.
3. **Adaptive ρ policy.** A keeper-side controller that widens ρ during periods of elevated empirical σ.
4. **Governance time-lock on admin actions.** A delayed-execution wrapper around the existing admin handlers, raising the cost of admin-compromise scenarios.

14.6 Future work (research)

The following items are research directions, not part of Phase 1 or the immediate Phase 2 scope:

1. **Active-bin sharding.** For very large hubs ($>$ 1B TVL), subdividing each execution bin into K sub-bin accounts for K-way throughput multiplication.
2. **Dynamic fee on volatility.** An EMA of realized volatility adjusts the swap fee upward during high-volatility regimes.
3. **Cross-chain hub.** Extending the hub topology across adjacent chains via SOL-pegged wrappers.
4. **Validator-set governance.** A keeper-side controller for the bucket-to-validator mapping, periodically rebalancing toward higher-performing or lower-slashing-risk validators.

15 Conclusion

Radial DEX has been presented — a SOL-native, yield-first decentralized exchange for Solana built around the insight that a DEX does not need all of its capital on-chain. By sizing the on-chain buffer as a deterministic worst-case bound ($T_{\max}^{(h)} \cdot K \cdot s_{\max}$), Radial delegates the majority ($\sim 80\%$) of hub capital directly to Solana validators via native staking, while retaining enough liquidity to handle any single-slot demand. The remaining tail of withdrawal pressure — empirically a $\sim 1\%$ of days for the most volatile measured architecture — is absorbed by an opt-in withdrawal-time mint of rSOL, a transferable liquid-staking receipt against the hub's active stake.

The design rests on four innovations:

1. *Yield-first capital allocation* (Section 8) routes the SOL hub's idle capital to a single transparent route — direct native validator staking through the bucket model — with a keeper maintaining the active ratio ρ in $[0.10, 0.30]$.
2. *Execution bins with the inter-bin claim ledger* (Sections 5 and 7) decouple swap execution from physical capital movement, deferring settlement to threshold-driven, epoch-aligned batched rebalances.
3. *rSOL withdrawal-time mint* (Section 9) absorbs tail withdrawals without forcing preemptive overprovisioning of the buffer, recovering the 52% opportunity cost a static 3σ buffer would impose on the most volatile architecture.
4. *Empirically calibrated buffer* (Section 11) sizes the ρ band against 150 days of on-chain reserves across six Solana AMM architectures, validating the 20–30% active-buffer target with directly measured idle ratios of 88–99%.

The Phase 1 architecture uses reserve-implied bin pricing (Section 10), eliminating the oracle dependency for intra-hub flow and removing the associated manipulation surface entirely. Price is derived from each bin's on-chain reserve ratio $P_n = a_b/R_b$, analogous to Meteora DLMM. Manipulation resistance comes from bin traversal costs (cumulative fees across bins), per-transaction user safeguards (deadline slot, output slippage bound, intermediate hub-leg bound), and arbitrage pressure. A minimum-TVL gate is documented as a Phase 2 candidate, not enforced in the current implementation.

The per-bin write-set architecture leaves the Sealevel scheduler free to dispatch non-conflicting swaps in parallel, configurable per-hub as a `u8` throughput parameter (default $T_{\max}^{(h)} = 32$). The inter-bin ledger keeps each per-swap write footprint minimal, enabling low-cost swap execution.

Hub-level yield depends on the prevailing native staking APY (currently $\sim 6.8\%$ on the Solana validator set) and the protocol/LP split parameter (Section 8.7); applied to the empirical idle floor of Section 11, this corresponds to a ~ 5.4 pp APY uplift on the deployed portion relative to leaving the same SOL TVL idle in a conventional pool. The design is implementable on current Solana infrastructure — the Phase 1 reference implementation is in security audit, with mainnet target Q1 2027, and a Phase 2 roadmap (optional external price feeds, TWAP overlay, multi-hub extension to admit stablecoin hubs, adaptive ρ policy, and governance timelock) is documented in Section 14.

A Inter-Bin Ledger Derivation

The inter-bin ledger conservation is derived from first principles. For a hub with B bins, let a_b be bin b's hub-asset claim. At initialization, $a_b^{(0)} = C^{\text{active}}/B$. After M swaps, each swap m transfers Δ_m from source bin s_m to destination bin d_m. The claim after all swaps is:

$$a_b = a_b^{(0)} + \sum_{m:s_m=b} \Delta_m - \sum_{m:d_m=b} \Delta_m.$$

The ledger entry is $L_b = a_b - a_b^{(0)}$. Summing across all bins:

$$\sum_b L_b = \sum_b \left(\sum_{m:s_m=b} \Delta_m - \sum_{m:d_m=b} \Delta_m \right) = \sum_m \Delta_m - \sum_m \Delta_m = 0.$$

Thus $\sum_b L_b = 0$ is an identity under any sequence of swaps.

Example 33 (Ledger balance verification)**.** A SOL hub with $B = 3$ bins, each $a_b^{(0)} = 10,000$ SOL. Sequence of three swaps:

- Swap 1: bin 1 $\to$ bin 2, $\Delta = 500$ SOL.
- Swap 2: bin 2 $\to$ bin 3, $\Delta = 200$ SOL.
- Swap 3: bin 3 $\to$ bin 1, $\Delta = 100$ SOL.

After swaps: $a_1 = 10,000 - 500 + 100 = 9,600$; $a_2 = 10,000 + 500 - 200 = 10,300$; $a_3 = 10,000 + 200 - 100 = 10,100$. Ledger: $L_1 = -400$, $L_2 = +300$, $L_3 = +100$. Sum: $-400 + 300 + 100 = 0$.

B Glossary

Active ratio (ρ) The fraction of total hub claims held as on-chain buffer balance. Maintained in $[0.10, 0.30]$ by the keeper.

Execution bin A parallel swap execution slot holding (R_b, a_b, L_b).

Hub The protocol-managed asset pool serving as the routing nexus for a class of tokens. In Phase 1 a single hub exists, anchored to SOL.

Buffer The hub's on-chain working liquidity held in wrapped-SOL (and analogous SPL-token) accounts. The buffer can carry any token routed against the hub.

Inter-bin claim ledger $\{L_b\}$ recording net hub-asset flows between bins since last settlement. Satisfies $\sum_b L_b = 0$.

Keeper An off-chain bot that monitors ρ and $\{L_b\}$, triggering rebalancing when thresholds are breached.

Yield route The on-chain pathway through which idle hub capital earns return. In Phase 1 the only active route is direct native validator staking via the bucket model.

Yield-first capital allocation The design principle that the majority of hub capital should be deployed to yield routes, with only the deterministic worst-case buffer kept on-chain.

Phase 1 / Phase 2 Phase 1 is the current production target: a single SOL hub with oracle-free reserve-implied pricing and native validator staking as the sole yield route, augmented by the rSOL withdrawal mint (Section 9) for tail-event relief. Phase 2 refers to a follow-up that may add optional external price feeds, a TWAP overlay, and a multi-hub extension; mechanisms tagged *Phase 2 proposed* are documented but not enforced in the current implementation.

Paused flag Per-hub emergency-stop bit (admin-settable) that rejects user-facing operations and fund-moving keeper operations while leaving read / sync / reconcile operations available for forensic accounting.

2-step admin handoff Admin transfer is executed as *Propose* (current admin nominates successor) followed by *Accept* (successor explicitly claims the role). Prevents stranding the role through a single failed transaction.

POLP Protocol-owned liquidity pool. The hub-level pool fed by yield delta and redistributed to LPs at the protocol's discretion (§8.7).

Native staking The SOL hub's default and only Phase 1 yield route: SOL beyond the buffer is delegated directly to validators through a bucket model wrapping the Solana Stake Program, rather than wrapped into a third-party staking token. Withdrawals are epoch-aware ($\sim$ 2 days).

rSOL A transferable SPL token minted, at user choice, when an LP withdrawal cannot be served from the hub's on-chain buffer. Backed by the hub's active validator stake at the protocol redemption rate $R(t)$; redeemable via burn-at-hub (one epoch) or sale on the secondary market (Section 9).

References

[1] Elsts, A. *Liquidity math in Uniswap v3 — Technical note.* 2021. https://atiselsts.github.io/pdfs/uniswap-v3-liquidity-math.pdf

[2] Trader Joe Labs. *Joe v2 Liquidity Book Whitepaper.* 2022. https://github.com/traderjoe-xyz/LB-Whitepaper/blob/main/Joe%20v2%20Liquidity%20Book%20Whitepaper.pdf

[3] Trader Joe Labs. *Liquidity Book v2.1 Specification.* 2024. https://docs.traderjoexyz.com/concepts/concentrated-liquidity

[4] Meteora. *Dynamic Liquidity Market Maker (DLMM) Documentation.* https://docs.meteora.ag/overview/products/dlmm/what-is-dlmm

[5] MakerDAO. *MIP-29: Peg Stability Module.* https://mips.makerdao.com/mips/details/MIP29

[6] Jito Labs. *JitoSOL Staking Integration Documentation.* https://www.jito.network/docs/jitosol/jitosol-liquid-staking/for-developers/staking-integration/

[7] Marinade Finance. *Marinade Liquid Staking Documentation.* https://docs.marinade.finance/

[8] Sanctum. *Sanctum Validator LST Network.* https://app.sanctum.so/