Skate AMM: Unifying Liquidity in One State Machine

Siddharth Lalwani

Skate

March 5, 2025

Abstract

Conventional automated market makers (AMMs) are inherently limited by their monolithic design – each operates on a single blockchain, leading to fragmented liquidity, isolated pricing curves, and suboptimal capital utilization. In this paper, we introduce *Skate AMM*, a novel protocol that implements a stateless application design pattern to enable a single AMM to operate seamlessly across multiple virtual machine environments (e.g., EVM, Solana, MoveVM, TonVM), maintaining one canonical pricing curve across disparate blockchain ecosystems. Liquidity deposited on any supported chain is integrated into one canonical liquidity state, maintained exclusively on a dedicated **hub chain**. Although token custody remains decentralized on individual chains (the spokes), core pricing logic, liquidity state, and tick data are shared globally. This unified state is secured via EigenLayer's economic trust, wherein a quorum of AVS operators (restaked node operators subject to slashing) collectively attest to state updates by producing tasks on the hub that trigger callbacks on the periphery via a hub-and-spoke AVS model. We also introduce new mathematical formulations to describe the unified pricing curve and present an analysis of loss-versus rebalancing (LVR), showing how the Skate design can minimizes toxic flows and arbitrage losses.

CRYP2NOVA

Archived copy · cryp2nova.com

Contents

1 Context **3**

2 Stateless Application Pattern: Kernel–Periphery Model **3**
- 2.1 Kernel (Hub Chain) . . . 4
- 2.2 Periphery (Spoke Chains) . . . 4
- 2.3 EigenLayer AVS for State Synchronization . . . 4
- 2.4 Execution Network . . . 4
- 2.5 Skate token: Alignment and Security . . . 4

3 Protocol Mechanics and Swap Execution **5**
- 3.1 Skate AMM Representation . . . 5
- 3.2 Swap Execution Sequence . . . 6
- 3.3 Adding Liquidity Sequence . . . 7
- 3.4 Robust Security and Mitigation Strategies . . . 7

4 Liquidity Providers **8**
- 4.1 LP Benefits in a Unified AMM . . . 8
- 4.2 LP Fee Optimization and Global Fee Sharing . . . 8
- 4.3 Impact on FLAIR Metric . . . 10

5 Swap Size Limitations and Local Liquidity Constraints **10**
- 5.1 Chain-Specific Fee Component . . . 11
- 5.2 Incentives for Rebalancing Inventory . . . 11

6 Future Vision: Skate AMM v2 and Beyond **13**
- 6.1 Skate AMM v2: Cross-Chain Extensions . . . 13

7 Conclusion **13**

A Formal Derivations and Comparative Analysis **15**
- A.1 Unified Pricing and Liquidity Invariants . . . 15
- A.2 LVR Quantification . . . 15
- A.3 Dynamic Fee Adjustment Model . . . 15
- A.4 Equilibrium Model for Liquidity Movement . . . 16

A References **17**

Archived copy · cryp2nova.com

1 Context

The rapid expansion of layer-1 blockchains and their diverse virtual machine (VM) environments (e.g., the Ethereum Virtual Machine (EVM), Solana VM, TON VM, Move VM) has led to a fragmented decentralized finance (DeFi) landscape. Liquidity in traditional automated market makers (AMMs) is siloed on each chain, leading to price disparities for assets and a suboptimal user experience, especially on newer or less liquid networks. For example, Uniswap v3 forks deployed on separate chains must each bootstrap liquidity independently, often yielding shallower liquidity pools and higher slippage for traders. This fragmentation compels users to either navigate complex cross-chain interactions or accept suboptimal exchange rates, while liquidity providers (LPs) are forced to split their capital across isolated pools—diminishing fee revenues and increasing exposure to impermanent loss.

Skate provides infrastructure enabling the development of stateless applications, directly addressing the limitations of monolithic AMM designs that maintain separate liquidity states across different blockchains. Skate AMM leverages this infrastructure by implementing a stateless application model, consolidating liquidity into a single global state managed via a single *Kernel* smart contract deployed on a dedicated hub chain. Lightweight *Periphery* contracts deployed across each blockchain handle user interactions, Eigenlayer Autonomous Verifiable Service (AVS) signature verification, asset custody, and asset settlement but do not store pricing information. State consistency across chains is maintained through Skate's AVS, ensuring decentralized, trustless validation of all state updates.

This approach establishes a unified pricing curve and significantly enhances liquidity depth, reduces toxic flow for LPs, and maximizes capital efficiency. It provides substantial benefits for asset issuers:

- **Immediate Market Depth and Stability**: Unified pricing ensures assets experience immediate deep liquidity, reducing slippage and arbitrage.
- **Reduced Capital Requirements**: A single liquidity deposit achieves omnichain liquidity depth, minimizing the capital needed.
- **Operational Simplification**: Asset issuers can focus on strategic initiatives rather than managing fragmented liquidity pools across multiple chains.

2 Stateless Application Pattern: Kernel–Periphery Model

The Stateless Application Design Pattern utilized by Skate AMM addresses the inefficiencies inherent in maintaining independent liquidity states across multiple blockchains. This architecture is inspired by established client-server paradigms prevalent in traditional distributed computing systems, where lightweight client-side components interact with centralized servers responsible for computation-intensive tasks and unified state management. Similar to how traditional web applications leverage lightweight clients to delegate computation and data consistency responsibilities to central backend servers, Skate's model applies analogous principles to blockchain infrastructure. It distinctly separates local asset management and user interactions from a shared computational logic, responsible for global liquidity management, pricing mechanisms, and invariant enforcement deployed on a dedicated hub chain. Cross-chain interactions and state updates are ensured through decentralized and economically secured validation provided by Skate's AVS.

Archived copy · cryp2nova.com

2.1 Kernel (Hub Chain)

The Kernel smart contract is deployed on a designated hub chain and holds the global liquidity pools, pricing, and tick data. It implements the core AMM logic using concentrated liquidity invariants (as used in Uniswap v3) but applied to a unified multi-chain context. All trades and liquidity changes are computed in the Kernel; that is, the Kernel enforces the constant-product invariant and updates the global state accordingly. Importantly, the Kernel does not manage token custody; instead, it aggregates liquidity positions provided by users across all chains.

2.2 Periphery (Spoke Chains)

On each supported chain, a Periphery contract is deployed to serve as the user interface. The Periphery handles local token custody and records user intents (such as swaps or liquidity provision). It then forwards these intents to the Kernel via cross-chain messages; the Kernel's response is delivered as a callback to the Periphery through Skate's AVS in a hub-and-spoke model. Periphery contracts do not perform any pricing computations; they rely entirely on the Kernel's global state. This separation minimizes on-chain complexity and ensures that users across all chains experience a single, unified market.

2.3 EigenLayer AVS for State Synchronization

Skate leverages its EigenLayer's AVS to securely coordinate between the Kernel and Periphery contracts. AVS consists of a decentralized network of operators who, by restaking assets, provide a secure, trust-minimized method for attesting cross-chain messages. The protocol supports a dual-staking model, incorporating both restaked ETH and staked $SKATE tokens to enhance economic security. When a user submits a transaction to a Periphery contract, the event is relayed by AVS nodes to the Kernel. Likewise, when the Kernel processes a transaction, its output is attested by the AVS and returned as a callback to the appropriate Periphery. This mechanism guarantees that state updates and trade executions are validated by a quorum of economically incentivized operators.

2.4 Execution Network

The Execution Network within the Skate architecture consists of Executors acting as listeners for actions initiated by users on Periphery (spoke chain) contracts. These Executors are responsible for capturing user's intents, relaying them securely to the Kernel on the hub chain for execution. Executors ensure timely and accurate execution of state updates in the Kernel, maintaining consistency and responsiveness across the entire multi-chain environment.

In future iterations, Skate intends to support advanced, intent-centric designs by introducing precompile contracts on the hub chain. These precompiles will directly facilitate signature verification for user interactions originating from altVM environments, such as ed25519 signatures.

2.5 Skate token: Alignment and Security

The Skate token is a core component of the protocol's design, underpinning both economic security and governance within a decentralized framework. Its primary functions are as follows:

Archived copy · cryp2nova.com

- **Economic Security via Staking.** Skate tokens are staked within the EigenLayer dual-staking framework to serve as collateral for Skate's AVS. This mechanism economically incentivizes AVS operators to comply with protocol requirements, thereby ensuring the security of cross-chain operations.
- **DAO Governance.** Skate token holders engage in decentralized governance through the Skate DAO, enabling the community to vote on protocol modifications and upgrades.
- **Staker Incentives.** A portion of the trading fees generated by Skate AMM is allocated to Skate token stakers. This incentive mechanism rewards token holders in proportion to their stake and supports alignment with long term protocol growth.

3 Protocol Mechanics and Swap Execution

3.1 Skate AMM Representation

In Skate AMM, rather than deploying independent AMM contracts on each individual chain, the protocol enforces a single, unified AMM with a canonical liquidity state on a dedicated hub chain. Each supported chain hosts a lightweight Periphery contract for local token custody and user-facing interactions, but all core state—liquidity positions, pricing, and tick data—is consolidated in a central Kernel contract on the hub.

Whenever liquidity is deposited on any chain, these funds are immediately integrated into the global reserve balances for the relevant token pair. Let X and Y denote the total aggregated reserves of tokens X and Y, respectively, maintained by the Kernel. The constant-product invariant

$$X \cdot Y = L^2$$

governs all swaps, where L is the invariant liquidity parameter capturing the active liquidity range. This formulation parallels the mechanism employed by conventional concentrated AMMs (e.g., Uniswap v3) over a specified price interval, ensuring that any infinitesimal increase in one reserve mandates a corresponding decrease in the other to preserve $X \cdot Y$.

A principal motivation behind this unified design is to address cross-chain information asymmetry.

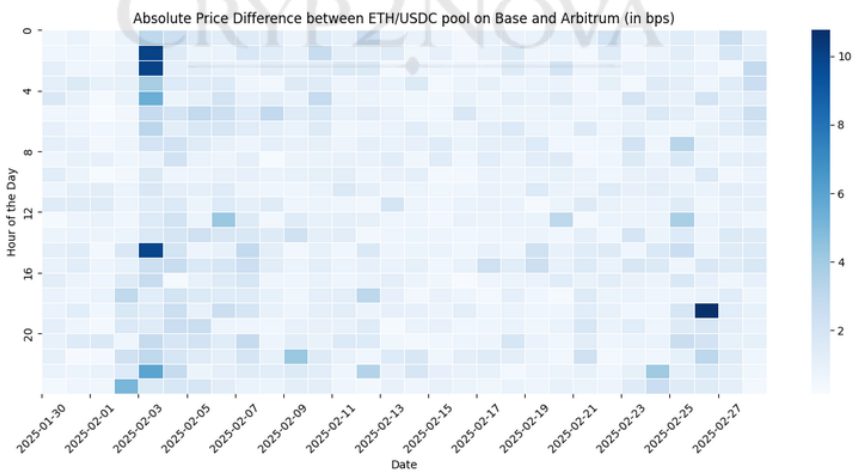


Figure 1: Absolute Price Difference between the biggest ETH/USDC pool on Base and Arbitrum(in bps)

Archived copy · cryp2nova.com

Such discrepancies open windows for arbitrage and create inefficient price quotes for the same asset pair across chains. Empirical data from the largest 5 bps ETH/USDC pools on Base and Arbitrum, for instance, shows that the absolute price difference may vary from nearly 0 to over 40 basis points at certain time intervals. These spikes, reflected by changes in color intensity in the heatmap, typically arise when one chain's pool lags in updating to the global market rate. In a fragmented architecture, sophisticated traders can exploit these short-lived disparities, disadvantaging less informed participants (Passive LPs in this case) and degrading overall market efficiency.

3.2 Swap Execution Sequence

When a user initiates a swap via a Periphery contract (for example, on Chain A), the process is as follows:

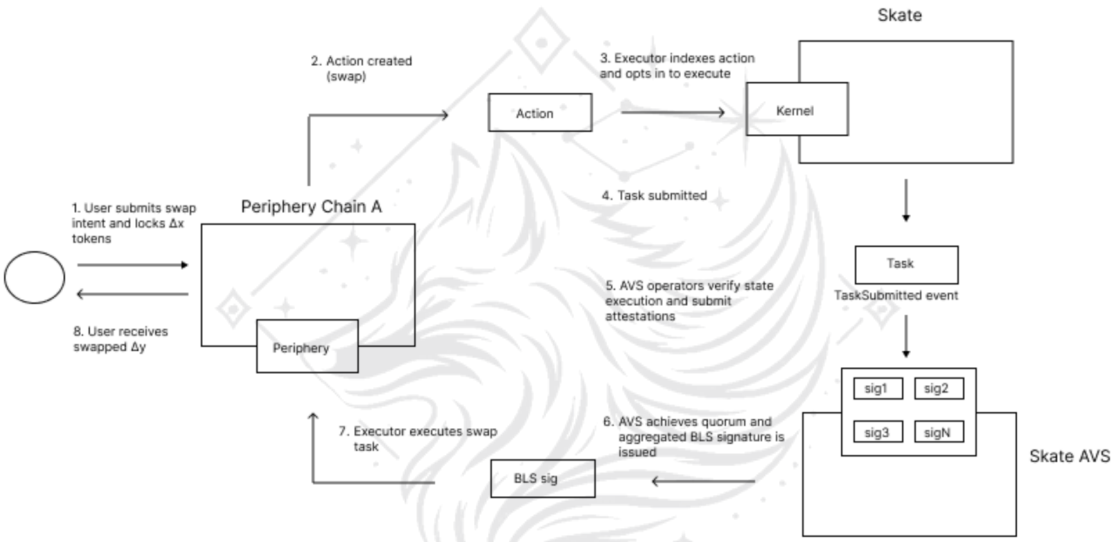


Figure 2: User Interaction flow for a swap on Skate AMM

1. **Swap Intent Submission:** The user submits a swap intent by staging and locking their input tokens in the Periphery contract.
2. **Kernel Processing:** An executor retrieves the swap intent and forwards it to the Kernel on the hub chain. The Kernel calculates the swap output using the global liquidity pool. For a trade with an input Δx of token X, the invariant

$$\big(X_{\text{total}} + \Delta x\big)\Big(Y_{\text{total}} - \Delta y\Big) = X_{\text{total}} \cdot Y_{\text{total}}$$

is used to compute the output:

$$\Delta y = Y_{\text{total}} - \frac{X_{\text{total}} \cdot Y_{\text{total}}}{X_{\text{total}} + (1-\gamma)\Delta x},$$

where γ denotes the trading fee. For swaps crossing tick boundaries, the Kernel applies piecewise computations based on novel tick-based mathematics analogous to those used in conventional concentrated AMMs.

Archived copy · cryp2nova.com

3. **AVS Attestation:** Skate's EigenLayer AVS operators verify the Kernel's state update and produce an aggregated BLS signature attesting to the valid execution of the swap.

4. **Settlement on the Periphery:** The Kernel's attested result is sent back via an AVS callback to the originating Periphery contract on Chain A. The Periphery contract verifies the aggregated signature and transfers Δy of token Y to the user, updating its local state accordingly.

Future iterations of Skate AMM will support seamless, atomic cross-chain swaps via AVS callbacks, allowing users to swap token X on one chain for token Y on another without manual bridging, while preserving robust security and fee structures.

3.3 Adding Liquidity Sequence

When adding liquidity, users interact with the Periphery contract by specifying a tick range and depositing token amounts in the corresponding ratio.

1. **Liquidity Intent Submission:** The user stages one or both tokens on the Periphery contract, locking them to create an addLiquidity *Action*.

2. **Kernel Processing:** An Executor listens to the Action and forwards it to the Kernel on the hub chain. The Kernel updates the global tick bitmap and calculates the liquidity using conventional constant product concentrated AMM equation:

$$L = \Delta x \cdot \frac{\sqrt{P_b} \cdot \sqrt{P_c}}{\sqrt{P_b} - \sqrt{P_c}},$$

$$L = \frac{\Delta y}{\sqrt{P_c} - \sqrt{P_a}},$$

   with the lesser of the two values being assigned to the user.

3. **AVS Attestation:** EigenLayer AVS operators verify the Kernel's state update and produce an aggregated signature attesting to the successful execution of the liquidity Intent.

4. **Settlement on the Periphery:** An executor picks up the resulting Task and settles the user on the Periphery contract, allowing withdrawal of any unused tokens. The difference between the staged and utilized amounts reflects minor price movements during execution.

3.4 Robust Security and Mitigation Strategies

Skate AMM incorporates comprehensive security measures to address potential attack vectors and ensure robust operation across chains.

1. **Cross-Chain Front-Running:** Unified state management in the Kernel prevents price disparities across chains, effectively neutralizing cross-chain front-running advantages. All swap intents are processed in a single global queue, reducing risks to those typical of conventional AMMs.

2. **Price Manipulation and Spoofing:** Since liquidity is aggregated globally, any attempt to manipulate the price must overcome the entire pool's depth. Localized imbalances are immediately arbitraged away, with tick-based updates enforcing consistent pricing.

Archived copy · cryp2nova.com

3. **Blockchain Reorganizations:** Swap and liquidity intents are executed only after the originating chain achieves sufficient confirmation. In the event of a reorganization, pending swaps are canceled or reversed via a two-phase commit protocol, ensuring atomicity and preventing double-spend or asset loss.
4. **Denial-of-Service and Network Stability:** Although the Kernel is a centralized component, its design incorporates redundancy and secure message relaying. Confirmation timeouts and off-chain relayers mitigate risks from network partitioning or transient DoS attacks, maintaining state integrity and protecting user funds.
5. **Liquidity Provider Safeguards:** LP funds remain secure as Periphery contracts maintain custody, and state updates occur only after verifiable, attested actions. The AVS attestation process further protects against unauthorized modifications, ensuring a fair balance between user and provider interests.

4 Liquidity Providers

4.1 LP Benefits in a Unified AMM

By providing liquidity to the unified pool, LPs capture fees from trading activity across all chains. Instead of splitting capital among multiple isolated pools, LPs deposit once into the global pool, earning a pro-rata share of total fee revenue. This yields:

- **Higher Fee Revenue:** Aggregated trading volume results in greater fee collection.
- **Reduced Arbitrage Losses:** A single, unified price minimizes price discrepancies, reducing the losses arbitrageurs extract.
- **Enhanced Capital Efficiency:** The effective liquidity is greater, resulting in lower slippage for large trades.

Moreover, because the global state informs pricing on every chain, LPs earn fees from trading volume across all chains even if their liquidity is only physically present on one chain. In equilibrium, only enough liquidity is needed locally to support swaps in both directions – if one asset pair has deep liquidity on even one chain, that liquidity is effectively available globally.

4.2 LP Fee Optimization and Global Fee Sharing

Consider a cross-chain AMM operating over N chains, where trading activity and fee accrual are measured on a per-block basis. Let Q_i denote the trading volume on chain i during a block, and f_i the fee rate applied on that chain. Thus, the fee revenue accrued on chain i in that block is:

$$F_i \;=\; f_i\, Q_i.$$

An LP allocates a total liquidity budget M across the N chains. Denote by ℓ_i the liquidity allocated to chain i, subject to

$$\sum_{i=1}^{N} \ell_i \;=\; M.$$

We assume that trading volume Q_i depends on both ℓ_i (with higher liquidity attracting more volume) and f_i (with higher fees potentially reducing volume); that is,

$$Q_i \;=\; Q_i(\ell_i, f_i),$$

Archived copy · cryp2nova.com

where Q_i is increasing in ℓ_i and decreasing in f_i. Consequently, the fee revenue on chain i is given by

$$F_i(\ell_i, f_i) \;=\; f_i\, Q_i(\ell_i, f_i).$$

An LP's revenue from chain i is proportional to the liquidity ℓ_i they provide relative to the total active liquidity on that chain, denoted by L_i^{active}. Thus, the LP's per-block revenue from chain i is

$$\Pi_i \;=\; F_i(\ell_i, f_i) \cdot \frac{\ell_i}{L_i^{\text{active}}}.$$

In the Skate AMM, however, a unified liquidity state is maintained, so that fee revenue is shared globally. Let L^{active} denote the total active liquidity across all chains. Then, the LP's global fee revenue in a block is given by

$$\Pi_{\text{LP}} \;=\; \frac{M_{\text{LP}}}{L^{\text{active}}} \sum_{i=1}^{N} F_i(\ell_i, f_i)$$

where M_{LP} is the LP's liquidity allocated in the active price range. This formulation implies that the LP's revenue is a pro-rata share of the total fee revenue, regardless of which chain generates the trades.

Define the marginal fee revenue per unit liquidity on chain i as:

$$v_i(\ell_i, f_i) \;:=\; \frac{\partial}{\partial \ell_i}\Big[f_i\, Q_i(\ell_i, f_i)\Big] \;=\; f_i\, \frac{\partial Q_i(\ell_i, f_i)}{\partial \ell_i}.$$

In an optimal allocation, LPs will reallocate their liquidity across chains such that the marginal revenue is equalized:

$$v_1(\ell_1^*, f_1) \;=\; v_2(\ell_2^*, f_2) \;=\; \cdots \;=\; v_N(\ell_N^*, f_N) \;=\; \lambda,$$

where λ is the Lagrange multiplier corresponding to the liquidity constraint. This condition, derived via optimization theory, ensures that no LP can increase its overall fee income by shifting liquidity from one chain to another.

Moreover, the fee rates f_i are dynamically adjusted to help achieve this equilibrium. If, for instance, chain k exhibits a relatively higher marginal revenue v_k (due to, say, a surge in trading volume), then reducing f_k slightly can attract additional volume and drive v_k down toward the equilibrium level. Conversely, if v_m is too low on chain m, increasing f_m can boost fee income per unit volume, thereby raising v_m. Through these adjustments, the system maintains:

$$v_1 \approx v_2 \approx \cdots \approx v_N,$$

ensuring that liquidity is optimally utilized across the network.

Global Fee Sharing vs. Siloed AMMs. In conventional, siloed AMMs, LP revenue on chain i would depend solely on F_i and the LP's fraction of L_i^{active}:

$$\Pi_i^{\text{silo}} \;=\; \frac{\ell_i}{L_i^{\text{active}}}\, F_i(\ell_i, f_i).$$

This dependence on local trading volume means that an LP's earnings can be significantly lower if that particular chain experiences low volume.

Archived copy · cryp2nova.com

In contrast, Skate AMM aggregates fee revenue across all chains, such that the LP's per-block revenue is:

$$\Pi_{\text{LP}} \;=\; \frac{M_{\text{LP}}}{L^{\text{active}}} \sum_{i=1}^{N} f_i \, Q_i(\ell_i, f_i).$$

Since

$$\sum_{i=1}^{N} f_i \, Q_i(\ell_i, f_i) \;>\; f_j \, Q_j(\ell_j, f_j) \quad \text{for any individual chain } j,$$

the expected revenue in Skate AMM is higher than that in any monolithic AMM. Additionally, the probability that the LP earns zero fees in a given block is reduced because it requires all chains to simultaneously have zero trading volume.

Conclusion. The mathematical analysis above demonstrates that, under optimal liquidity allocation and dynamic fee rate adjustments, the Skate AMM's unified model yields higher expected fee revenue and lower income volatility for LPs compared to siloed, single-chain AMMs. This global fee sharing mechanism, which equalizes the marginal fee revenue across chains, ensures that LPs are probabilistically better off participating in Skate AMM.

4.3 Impact on FLAIR Metric

The FLAIR (Fee Liquidity-Adjusted Instantaneous Return) metric, originally introduced by Uniswap Labs, measures the fee return that a liquidity provider (LP) earns relative to the liquidity they deploy. Skate AMM's unified liquidity model fundamentally alters how fee returns are measured by the FLAIR metric. Traditionally, FLAIR quantifies an LP's fee income relative to their liquidity contribution in a single, isolated pool. In contrast, Skate AMM aggregates liquidity across chains into one global state, eliminating the need for separate FLAIR scores per chain. With a single liquidity bitmap governing the entire system, fees earned from trades on any chain contribute to a consolidated fee revenue, thereby enhancing the overall fee return on LP capital.

This integrated model means that the effectiveness of an LP is measured not only by the fee yield on the chain where they provide liquidity, but also by the trading volume on other chains. In effect, LPs earn fees across the entire ecosystem, which raises the global fee return compared to fragmented models. Dynamic fee mechanisms further optimize this process by adjusting fees based on local liquidity imbalances, incentivizing rebalancing that boosts fee collection.

Further theoretical research on the impact of dynamic fee adjustments on FLAIR performance is underway and will be detailed in a forthcoming study.

5 Swap Size Limitations and Local Liquidity Constraints

A notable challenge in a multi-chain context is that although the unified global state offers substantial liquidity, the execution of swaps on an individual chain is constrained by the locally available liquidity (denoted L_{local}). That is, while the pricing mechanism reflects the total aggregated liquidity, each swap operation is limited to drawing from the tokens available on the chain where the transaction is performed. For example, if Chain A holds only \$100 worth of token X despite the global pool aggregating \$1100, the swap executed on Chain A is restricted to that \$100 inventory.

Archived copy · cryp2nova.com

In version 1, LPs are permitted to withdraw their entire liquidity only if L_{local} is adequate to satisfy their position; otherwise, only a partial withdrawal is feasible. This constraint is regarded as advantageous, as it motivates LPs to initiate cross-chain rebalancing—or encourages systematic bots to perform such adjustments—through the modulation of fees. Moreover, the dynamic fee mechanism, described in detail below, further promotes the reallocation of liquidity to locations where it is most needed, thereby ensuring that although the swap capacity on each chain is limited by L_{local}, the effective pricing depth accurately reflects the total global liquidity.

5.1 Chain-Specific Fee Component

The dynamic, chain-specific fee in Skate AMM will be designed to convert local liquidity imbalances into economic incentives that prompt systematic participants to replenish the scarce asset. This fee is applied whenever local liquidity is replenished, whether through swap flows that bring in the underrepresented token or through additional liquidity provision and is tailored to each chain's conditions.

The fee mechanism is charged in two distinct ways:

1. **Swapping in Scarce Assets:** In Skate AMM, a chain-specific fee mechanism is used to mitigate local liquidity imbalances. Each chain monitors its own token inventories and computes a local imbalance metric, Δ_i. A dynamic fee function $\Phi_i(\Delta_i)$ is then applied to adjust the effective fee. If f_0 is the base fee of the pool, the effective fee on chain i is given by:

$$f_i \;=\; f_0 \cdot \Phi_i(\Delta_i).$$

   This dynamic fee increases when the asset being swapped out is scarcer, thereby deterring further depletion and encouraging swaps that replenish the local pool.

2. **Adding Liquidity with Scarce Assets:** When an LP adds liquidity on a chain where a token is scarce, the fee premium earned on subsequent trades serves as an incentive. Specifically, if an LP provides liquidity at an out-of-range position where the dynamic fee premium is defined as

$$f_i^{\text{premium}} = f_i(q) - f_0,$$

   and the LP expects to hold this position for a period T incurring a holding cost C_h per unit liquidity, then the strategy is profitable provided that:

$$f_i^{\text{premium}} \times T > C_h.$$

   This condition indicates that the additional fee income compensates for the opportunity cost of providing liquidity in a scarce environment.

Note: The exact formulation and calibration of the chain-specific fee mechanism remain active areas of research. The equations and conditions delineated above serve an illustrative purpose and are amenable to further refinement in light of empirical observations and the dynamic evolution of market conditions.

5.2 Incentives for Rebalancing Inventory

The dynamic, chain-specific fee in Skate AMM is applied whenever a chain's local liquidity is replenished—either through swaps that bring in the scarce token or by adding liquidity directly. This fee is designed to incentivize systematic participants to replenish the asset

Archived copy · cryp2nova.com

that is underrepresented in the local liquidity pool, without necessarily requiring positions to be extremely out-of-range.

We consider two primary ways to get incentivized:

Mode 1: Out-of-Range Liquidity Provisioning. Systematic players may choose to provide liquidity at price ranges outside the current active band. These *out-of-range* positions do not incur impermanent loss risk since they remain inactive unless the market moves dramatically. The incentive is driven by the chain-specific fee premium. Formally, let C_h denote the holding cost per unit liquidity per unit time and T the expected holding period. If the additional fee premium earned per unit liquidity, defined as

$$f_i^{\text{premium}} = f_i(q) - f_0,$$

satisfies

$$f_i^{\text{premium}} \times T > C_h,$$

then it is profitable for systematic players to add out-of-range liquidity. In this way, the fee mechanism converts local scarcity into an economic incentive to replenish the underrepresented asset.

Mode 2: Round-Trip Cross-Chain Rebalancing (v2). In the envisioned Skate AMM v2, enhanced cross-chain swap functionality will enable systematic bots to perform round-trip swaps that directly rebalance liquidity across chains. The process unfolds as follows:

Step 1: Initiate Cross-Chain Swap: On Chain A—where an asset (e.g., ETH) is scarce—the bot initiates a swap converting ETH to USDC. The elevated dynamic fee on Chain A, $f_A(q)$, reflects the local scarcity.

Step 2: Bridge Transfer: The resulting USDC is bridged to Chain B, where the same asset is abundant.

Step 3: Intra-Chain Rebalancing Swap: On Chain B, the bot executes an intra-chain swap converting USDC back to ETH, thereby restoring the ETH inventory on Chain A indirectly.

The incentive for the bot to perform this round-trip rebalancing is captured by the inequality:

$$f_A^{\text{premium}} \times q > C_{\text{rt}},$$

where f_A^{premium} is the fee premium on Chain A (i.e., the excess fee due to scarcity), q is the trade size, and C_{rt} represents the combined cost of the round-trip swap (including fees on the destination chain and bridging costs). When this condition holds, the dynamic fee premium compensates for the costs, thereby incentivizing cross-chain rebalancing.

In summary, the chain-specific fee mechanism in Skate AMM not only compensates LPs for local liquidity imbalances but also creates two distinct incentive modes for systematic participants. In both modes, the goal is to replenish the scarce asset on a chain—ensuring that even though swap execution is limited by local liquidity, the effective pricing depth benefits from the aggregated global state. In equilibrium, the model implies that each chain requires sufficient liquidity to support bidirectional swaps, with the liquidity on any one chain potentially reflected across the network.

Archived copy · cryp2nova.com

6 Future Vision: Skate AMM v2 and Beyond

6.1 Skate AMM v2: Cross-Chain Extensions

Skate AMM v1 implements a unified liquidity state with intra-chain swaps based on a global pricing curve. In Skate AMM v2, the protocol is extended to support atomic cross-chain swaps. The technical objectives for v2 are as follows:

Objective 1) **Atomic Cross-Chain Swaps:** In v2, a swap initiated on one chain (e.g., Chain A) will result in the output being delivered atomically on a different chain (e.g., Chain B). Specifically, the Kernel performs all pricing computations using the unified liquidity state, and the swap result is relayed as a callback via the AVS mechanism to the Periphery contract on the destination chain. This design preserves atomicity across chains while maintaining consistency in security, execution, and fee models with v1. Enhanced cross-chain coordination and robust inventory management are incorporated to mitigate latency.

Objective 2) **Enhanced Rebalancing via Round-Trip Swaps:** Systematic participants are enabled to execute cross-chain round-trip swaps that earn dynamic fee premiums while rebalancing local token inventories. For instance, if a token (e.g., ETH in an ETH/USDC pair) is underrepresented on Chain A, a participant can:

- **(i)** Execute a cross-chain swap converting ETH on Chain A to USDC on Chain B, thereby increasing the ETH inventory on Chain A via fee incentives.
- **(i)** Subsequently, perform an intra-chain swap on Chain B to convert USDC back to ETH, rebalancing the exposure on Chain B.

The profitability of such operations is determined by the fee differentials exceeding the round-trip and bridging costs.

Objective 3) **Inter-Chain Liquidity Provisioning:** v2 facilitates liquidity provisioning across chains by allowing users to add liquidity on one chain and remove it on another. This mechanism optimizes capital allocation and liquidity management by leveraging the unified global liquidity state, independent of the chain on which the tokens are physically held.

These extensions refine the unified global state concept by integrating atomic cross-chain swap execution and advanced rebalancing strategies. The design ensures that local liquidity constraints are efficiently managed via dynamic fee adjustments and inter-chain coordination, thereby preserving the global pricing mechanism established in v1.

7 Conclusion

Skate AMM provides a novel approach to decentralized liquidity provision by unifying liquidity across multiple blockchains into a single global state. Through its stateless Kernel–Periphery architecture, Skate enables intra-chain swaps with pricing derived from a unified liquidity pool, dramatically reducing slippage, improving capital efficiency, and mitigating losses due to liquidity fragmentation. By leveraging EigenLayer's AVS for secure cross-chain

Archived copy · cryp2nova.com

communication, Skate provides a robust framework where LPs earn fees from a global pool and arbitrage opportunities are minimized.

The mathematical and game-theoretic analysis presented indicates that a unified AMM can enhance fee revenue efficiency, reduces Loss-Versus-Rebalancing (LVR), and enhances the overall user experience compared to fragmented Uniswap v3 forks. Additionally, asset issuers benefit by no longer having to bootstrap liquidity separately in each chain, thereby reducing capital requirements and ensuring deep liquidity across ecosystems.

While Skate AMM v1 validates the concept with intra-chain swaps and a global pricing state, our vision for Skate AMM v2 is to enable direct cross-chain swaps, further dissolving barriers between networks and optimizing liquidity rebalancing. This evolution, underpinned by dynamic fee adjustments and coordinated via AVS, will pave the way for a truly omnichain liquidity protocol—unlocking a new era of interoperable, scalable, and efficient DeFi.

Archived copy · cryp2nova.com

A Formal Derivations and Comparative Analysis

A.1 Unified Pricing and Liquidity Invariants

Skate AMM uses the same pricing invariants as Uniswap v3 for a given price range. Let x and y be the token quantities within the active tick range and L the liquidity parameter. The invariant is:

$$x \cdot y = L^2.$$

For a swap with input Δx (post-fee), the updated reserve of token X is $x + \Delta x$ and the new reserve of token Y is given by:

$$y' = \frac{L^2}{x + \Delta x}.$$

Thus, the output amount is:

$$\Delta y = y - \frac{L^2}{x + \Delta x}.$$

In Skate, these computations are executed on the global state, where:

$$x = X_{\text{total}}, \quad y = Y_{\text{total}},$$

ensuring that all trades benefit from the aggregated liquidity positions.

A.2 LVR Quantification

Let V_0 be the initial value of an LP's position and V_t its value at time t without continuous rebalancing. Let R_t be the hypothetical value if the LP had continuously rebalanced. Then, the Loss-Versus-Rebalancing (LVR) is defined as:

$$\text{LVR}_t = R_t - V_t.$$

For a constant-product market maker, if the price follows a geometric Brownian motion with volatility σ, the impermanent loss is approximately proportional to σ^2. In a unified pool with total liquidity L, the effective price impact is proportional to $\Delta x / L$. In contrast, if liquidity is fragmented into N pools (each with L/N), the slippage for a trade of size Δx becomes:

$$\frac{\Delta x}{L/N} = \frac{N \Delta x}{L},$$

i.e., roughly N times worse. Therefore, the efficiency gain is approximately:

$$\rho = \frac{\text{LVR}_{\text{Skate}}}{\text{LVR}_{\text{fragmented}}} \approx \frac{1}{N}.$$

A.3 Dynamic Fee Adjustment Model

Let I_i^X and I_i^Y denote the local token inventories on chain i. Define the local fractions:

$$w_i^X = \frac{I_i^X}{\sum_j I_j^X}, \quad w_i^Y = \frac{I_i^Y}{\sum_j I_j^Y}.$$

The imbalance on chain i is:

$$\Delta_i = w_i^X - w_i^Y.$$

Archived copy · cryp2nova.com

We define a fee multiplier as:

$$F_i(\Delta_i) = 1 + k \cdot |\Delta_i|,$$

with $k > 0$. Consequently, for a swap of size q on chain i that removes token X (when X is scarce, i.e., $\Delta_i < 0$), the effective fee is:

$$f_i(q) = f_0 \cdot F_i(\Delta_i),$$

where f_0 is the base fee. This dynamic fee mechanism compensates LPs for potential LVR and incentivizes rebalancing by effectively lowering fees for trades that help restore balance.

A.4 Equilibrium Model for Liquidity Movement

Let $L_i(t)$ denote the liquidity on chain i at time t and let $F_{i\to j}(t)$ denote the flow of liquidity from chain i to chain j. Then:

$$\frac{dL_i}{dt} = \sum_{j\neq i} F_{j\to i}(t) - \sum_{j\neq i} F_{i\to j}(t).$$

Assuming LPs shift liquidity when the marginal revenue difference exceeds a cost c_{ij}, let the marginal revenue per unit liquidity be:

$$v_i = \frac{\Pi_i}{L_i},$$

where Π_i is the fee revenue from chain i. Then, we can model:

$$F_{i\to j}(t) = \beta \max\{v_j(t) - v_i(t) - c_{ij}, 0\},$$

with β a rate constant. Equilibrium is reached when:

$$v_1 = v_2 = \cdots = v_N = v^*,$$

ensuring efficient liquidity allocation in accordance with trading demand.

CRYP2NOVA

Archived copy · cryp2nova.com

A References

References

[1] Adams, Hayden, et al. "Uniswap: Protocol for Decentralized Exchange." 2020.

[2] Milionis, Jason, C. Moallemi, T. Roughgarden, A. Zhang. "Automated Market Making and Loss-Versus-Rebalancing." arXiv:2208.06046 (2024).

[3] Milionis, Jason, et al. "LVR: Quantifying the Cost of Providing Liquidity to AMMs." a16z blog, Sept 19, 2022.

[4] Skate Documentation. "Skate AMM Overview and Design." 2024.

[5] Skate Team. "How Skate is Connecting the altVM Future." Medium, 2024.

[6] Aanes, J.M., et al. "Automated Market Makers for Cross-chain DeFi and Sharded Blockchains." arXiv:2309.14290 (2025).

[7] Milionis, Jason, Xin Wan, and Austin Adams. "FLAIR: A Metric for Liquidity Provider Competitiveness in Automated Market Makers." 2025.

[8] Biconomy Team. "Self Balancing Cross-Chain Liquidity Pools." 2022.

[9] Catalyst Team. "Optimising LP Performance Part 2: Dynamic Fees." 2023.

[10] EigenLayer. "EigenLayer: An Introduction to the EigenLayer AVS Ecosystem." 2023.

[11] Thorchain.org. "How Thorchain Works." (n.d.).

CRYP2NOVA

Archived copy · cryp2nova.com