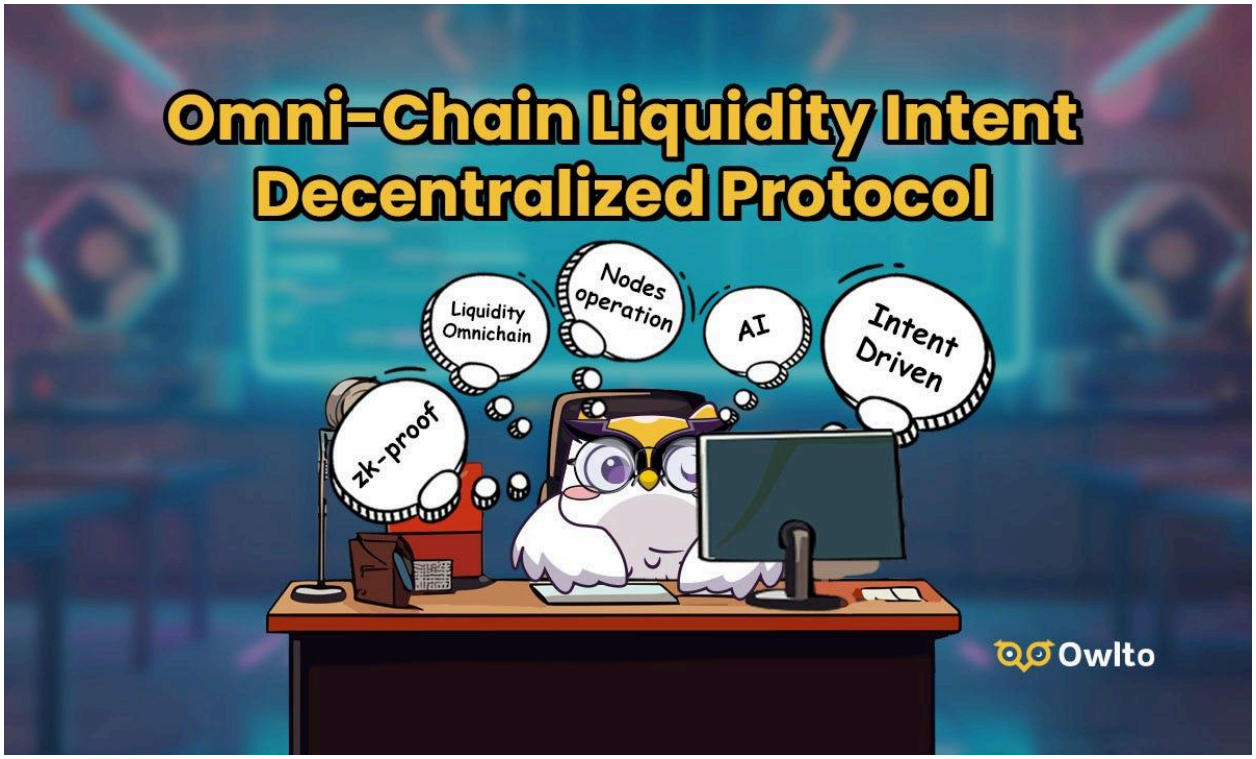
Owlto Finance White Paper: Omni-Chain Liquidity Intent Decentralized Protocol



Abstract

This white paper introduces a decentralized trading protocol based on omni-chain liquidity, intent, and AI optimization. The protocol integrates liquidity across multiple blockchains, utilizing smart contracts, zero-knowledge proofs, and AI technology to construct an efficient and secure cross-chain transactions platform. Its aim is to simplify cross-chain operations, lower trading costs, and enhance trading efficiency and security.

Keywords

Omni-Chain Liquidity, Cross-Chain Transactions, AI, Smart Contracts, Zero-Knowledge Proofs, Intent-Driven, Liquidity Prediction, Transaction split, Smart order routing, Cross-Chain Acceleration, Liquidity Bridge, Security Mechanisms, Node Incentives, Node Penalties, Path Optimization, Liquidity Aggregation, Trading costs, Decentralization, Efficient Execution.

Glossary

- Omni-chain liquidity: Integration of liquidity pools across multiple blockchains to form a unified liquidity pool, providing deeper liquidity, reducing trading slippage, and enhancing trading efficiency.

- Cross-chain transactions: The process of transferring and exchanging assets between different blockchains, aimed at simplifying cross-chain operations, reducing transaction costs, and achieving a seamless cross-chain transactions experience.
- Zero-knowledge proof: A cryptographic technique that allows one party (the prover) to convince another party (the verifier) that a statement is true without revealing any specific information. It is used to protect user privacy, prevent MEV attacks (from miners), and ensure transaction security.
- Intent-driven: A trading mode where users submit their trading intents, and the system automatically matches and executes transactions based on those intentions, without requiring users to specify a particular trading path.
- Liquidity prediction: Utilize AI algorithms to analyze historical data and current market conditions, predict changes in liquidity across different blockchains, proactively adjust the configuration of liquidity pools, and ensure that sufficient liquidity is available at any point in time to support cross-chain transactions.
- Transaction split: Split a user's transaction request into multiple small amounts and allocate them to different liquidity providers to obtain the best possible liquidity, minimize transaction fees, and meet liquidity requirements.
- Smart order routing: Utilize AI algorithms to analyze historical transaction data and current market conditions, select the optimal path to minimize transaction costs and time, and choose the best path among multiple available routes.
- Cross-chain acceleration engine: Utilize intent-driven, zero-knowledge proof, and smart contract technology to improve the efficiency and security of cross-chain transfers, lowering the costs of cross-chain transactions.
- Liquidity bridge: Achieve liquidity interconnectivity between different blockchains, ensuring efficient and secure circulation of funds during cross-chain transactions, and supporting various cross-chain bridging methods.
- Node: A computer or device operating in the blockchain networks, responsible for executing transaction validation, block generation, and consensus maintenance. In the cross-chain liquidity protocol, nodes also manage liquidity, transaction split, and path optimization.
- Liquidity aggregation: Consolidate liquidity sources across multiple blockchains to form a unified liquidity pool, ensuring flexibility and efficiency in cross-chain transactions while reducing trading slippage and price discrepancies.

Introduction

In current blockchain ecosystems, public chain ecosystems are increasingly thriving, with each having its unique liquidity pools. However, this decentralized liquidity presents numerous challenges for users trading across different chains. This protocol aims to integrate liquidity from various chains, providing a seamless cross-chain transactions experience, allowing users to trade across chains while leveraging omni-chain liquidity.

Decentralized exchanges (DEX) based on the AMM model offer superior liquidity compared to many centralized exchanges. The vigorous development of DEXs has brought sufficient depth

and better liquidity to the trading market. However, due to the inability to interconnect and share data and messages between public chains, liquidity cannot be shared across chains. Therefore, protocols are needed to facilitate cross-chain liquidity.

This protocol builds a cross-chain acceleration engine based on intent, zero-knowledge proofs, smart contract technology, and AI algorithms, dedicated to reducing transaction costs and improving transaction efficiency while ensuring decentralization and security. Given the real-time fluctuations in the trading market, we need to utilize AI for liquidity prediction and trading path optimization.

The two main challenges faced by omni-chain liquidity and trading are the complexity of cross-chain transactions and the liquidity fragmentation.

- Complexity of cross-chain transactions: Users face complex cross-chain transactions operations and high fees when trading across different chains.
- Market liquidity fragmentation: Liquidity pools on individual blockchains operate independently, resulting in a fragmented liquidity landscape that hampers effective utilization.

To address these issues, we introduce a full-chain liquidity trading protocol optimized with AI technology:

1. **Liquidity Prediction**

- By leveraging AI to predict liquidity changes across different blockchains, the protocol can proactively adjust the configuration of liquidity pools, ensuring sufficient liquidity to support cross-chain transactions at any given time.
- Users can access the total liquidity across the entire blockchain network.

2. **Transaction Split**

- Splitting trades to obtain the best quality liquidity across various chains.
- Users can access omni-chain liquidity for a trade initiated on any chains.

3. **Smart Order Routing**

- AI analyzes historical trading data and current market conditions to select the optimal path to minimize trading costs and time, allowing users to participate more flexibly in asset trading.
- This reduces execution costs and time.

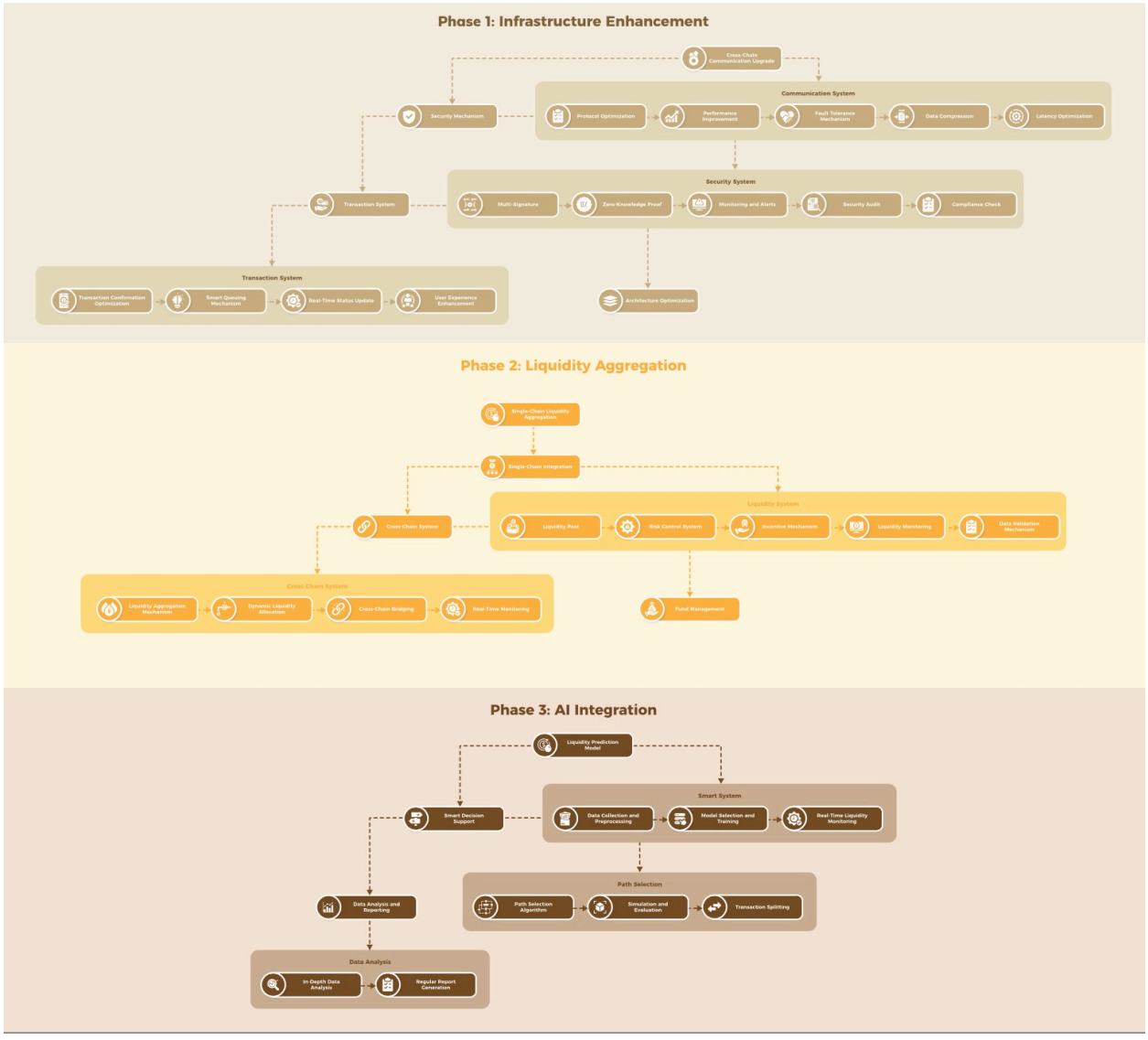
4. **Lowering Cross-Chain Transactions Fees**

- By leveraging intent on-chain and liquidity pools, we effectively lower cross-chain transactions fees.
- This enhances the efficiency of cross-chain transactions protocols and lower data transmission costs.

5. **Security and Privacy**

- The intent-driven protocol design avoids MEV attacks, enhancing user transaction privacy.
- Through intent on-chain and secure node verification, we address security validation issues in the underlying protocol and ensure transaction safety.

This protocol not only addresses the complexity of cross-chain transactions and insufficient market liquidity, but also significantly improves trading efficiency and security, providing users with an efficient and decentralized cross-chain transactions platform.



Background

With the rapid development of blockchain technology, more public chains are emerging and each with its unique ecosystem and liquidity pool. However, this dispersed liquidity brings the following challenges:

1. **Liquidity Fragmentation**: Each chain's liquidity pool exists independently, leading to fragmented liquidity that cannot be fully utilized.
2. **Complex Cross-Chain Transaction**: Users face complex operations and high fees when trading across different chains.
3. **Price Differences**: Due to the liquidity fragmentation, asset prices on different chains may vary greatly, affecting users' trading experience.
4. **Liquidity and Trading Are Always Changing**: On-chain liquidity and trading can change at any time, impacting user transactions, such as transaction failures or excessive slippage.

In the current blockchain ecosystem, users want to simplify transaction processes, achieve asset conversions, and minimize friction costs without submitting specific transaction instructions directly. Users only need to submit a request containing their trading intent, and the system will automatically match and execute transactions based on these requests.

By integrating liquidity pools across various chains into an omni-chain liquidity pool, we can significantly enhance trading efficiency and user experience:

1. **Liquidity Depth**: Provide deeper liquidity, reducing transaction slippage and price discrepancies.
2. **Seamless Cross-Chain Transaction**: Users can perform seamless transactions between different chains, simplify cross-chain operations and lower transaction costs.
3. **Price Consistency**: Omni-chain liquidity reduces price discrepancies for assets across different chains, providing more consistent pricing.
4. **Smart Prediction and Optimization**: Based on AI algorithms, we predict liquidity changes and trading needs, adjusting liquidity pool configurations in advance. Additionally, AI helps find the optimal trading paths, minimizing costs and time.
5. **Security and Transparency**: Smart contracts automate and decentralize transactions, ensuring safety and transparency.

Challenges of Omni-Chain Liquidity

Certainly, omni-chain liquidity transaction currently faces some challenges:

- Accuracy: User-submitted transactions need to be accurate and complete. Any errors or incomplete information can lead to transaction failures or delays.
- Privacy Protection: While on-chain trading increases transaction transparency, it may also expose users' transaction intentions, leading to privacy risks.

- Execution Efficiency: Execution depends on complex computations and verifications, which may increase system complexity and computational costs, especially when real-time monitoring of liquidity changes is required.
- Security: Transaction transmission and storage require high security to prevent tampering or theft.

Encryption technologies and secure protocols must be employed to ensure the secure transmission and storage of intentions.

The protocol aims to meet users' omni-chain trading needs while addressing the aforementioned challenges.

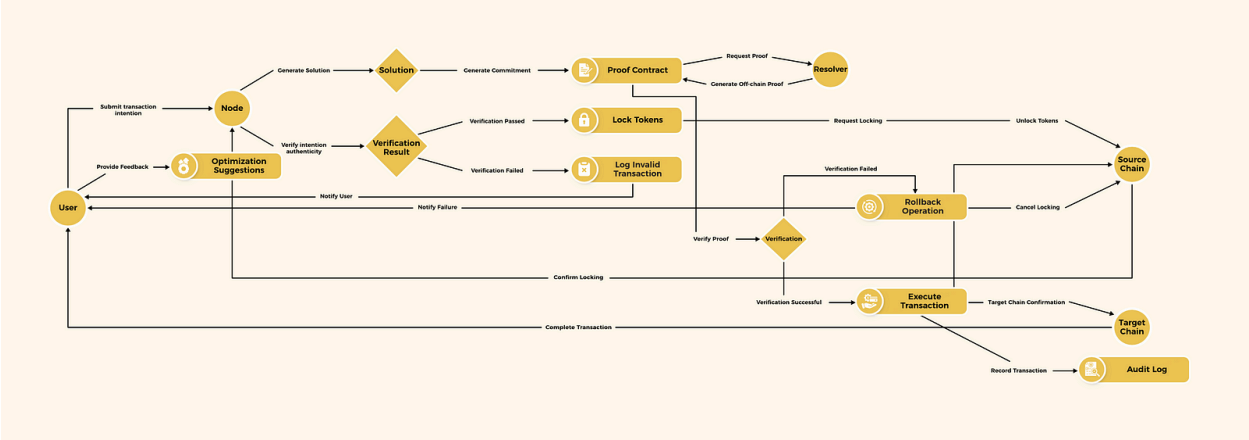
Omni-Chain Liquidity Protocol

This protocol aims to integrate liquidity from various chains, providing a seamless cross-chain transactions platform that allows users to trade across different chains. Specifically, it includes the following core components:

- Cross-Chain Liquidity Pool: Integrates liquidity pools from various chains into a omni-chain liquidity pool, providing deeper liquidity.
- Cross-Chain Transactions Protocol: An efficient cross-chain transactions protocol that allows users to perform seamless transactions between different chains and access the best liquidity through AI and algorithms.
- Liquidity Bridge: Facilitates liquidity interoperability between different chains through a liquidity bridge, ensuring smooth transaction executions.

The protocol uses zero-knowledge proofs for information transmission and employs a cross-chain intent trading mechanism, allowing users to seamlessly complete information transmission during cross-chain transactions without waiting for the delays associated with traditional data transfer and authentication. In this process, interactions between users and solvers can be completed on the target chain, enhancing transaction efficiency. The simplified steps are as follows:

1. Users interact with the protocol on the target chain to convey their intentions and acquire tokens. The trading protocol generates a solution based on the user's intent.
2. A commitment is generated via zero-knowledge proof, containing the solver's proof of user intent execution without disclosing specific execution details.
3. Nodes forward this commitment to the proof contract on the source chain.
4. Upon receiving the user's transaction request, the protocol splits the transaction according to a transaction splitting algorithm.
5. The split transactions are executed on the target chain based on routing.
6. Nodes generate off-chain proofs based on commitments and submit them to the proof contract on the source chain.
7. After verification, tokens on the source chain are unlocked and delivered to the protocol.



AI Optimization

The protocol utilizes AI algorithms to predict liquidity changes and trading demands, adjusting liquidity pool configurations in advance. At the same time, by optimizing trading paths through AI, transaction costs and time are minimized. The specific process is as follows:

- Liquidity Prediction: The AI analyzes historical data and current market conditions to predict liquidity changes across different chains, adjusting liquidity pool configurations dynamically.
- Path Optimization: By using AI algorithms to find the best trading paths, users are ensured to obtain the highest quality liquidity while minimizing transaction costs and time.
- Data Analysis and Reporting: Conducting in-depth analysis of cross-chain transactions data to generate detailed reports and insights, helping protocol developers and users better understand market dynamics and liquidity conditions.

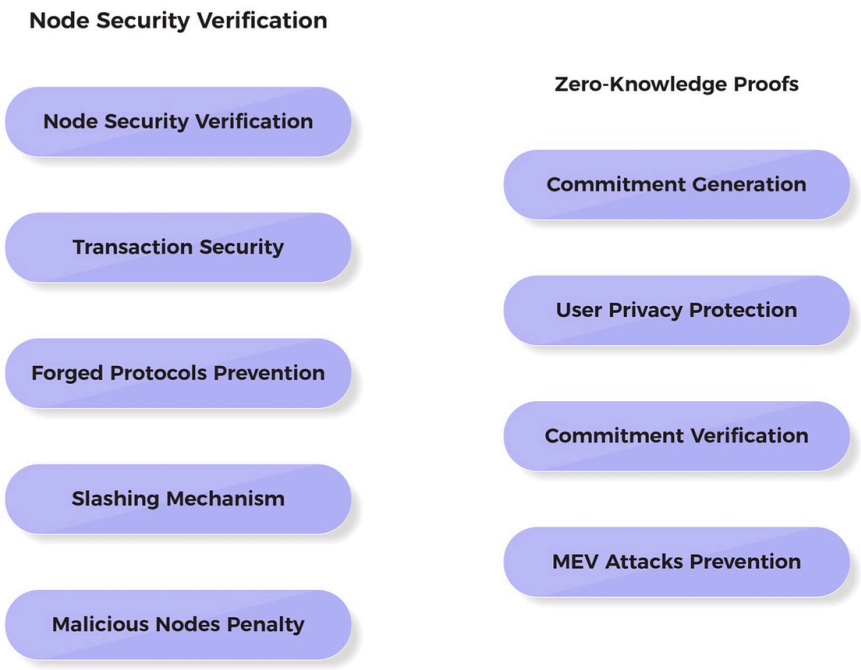
Security Mechanisms

Zero-knowledge proofs ensure that specific information is not disclosed, while verifiers can confirm that user intent has been executed correctly. If there is an attempt to transmit forged protocols, the generated commitments will fail verification, ensuring the security of the entire transaction process. Specific implementations include:

- Generating a hash of the solution through the keccak256 algorithm, which serves as the commitment data for zero-knowledge proofs.
- Forwarding this commitment to the proof contract on the source chain.
- Other validating nodes generate zero-knowledge proofs based on the commitment and submit them to the proof contract. After verification, the token is unlocked.

Zero-knowledge proofs are also applied in the verification stage, preventing MEV attacks and enhancing overall system security through zero- knowledge proofs and intent-on-chain design.

Node operations are based on decentralized consensus, and we adopt a staking mechanism for nodes. If fraudulent data is transmitted, the staked tokens will be subject to penalties (slashing mechanism). This design ensures that nodes maintain honest behavior in a decentralized environment, avoiding malicious actions.



The protocol is a fully decentralized trading protocol that acts as a router for user intentions and on-chain liquidity. On the whole, the platform acts as a router for user intentions and on-chain liquidity, responsible for matching users' cross-chain trading intentions and distributing their transactions to ensure them obtain the optimal liquidity. The protocol also needs to provide decentralized security mechanisms to ensure the safety of users' funds and usage.

Intent-Centric Omni-Chain Liquidity Trading

In traditional trading processes, transaction signatures specify a specific computation path, and validators execute computations based on that path, incentivized by transaction fees to complete the process. However, an intent-centric trading model does not adhere to a specific path but allows for any path that can achieve the expected result while satisfying user constraints.

In the intent-centric omni-chain liquidity trading model, the user's trading process undergoes fundamental changes. Unlike traditional trading that relies on specific computational paths and clear execution steps, the intent-centric model allows users to define their expected trading results, while the protocol optimizes and manages the specific execution paths through nodes.

This model aims to enhance the efficiency of cross-chain liquidity usage and provide users with a more flexible and efficient trading experience.

User Intent and Protocol Nodes

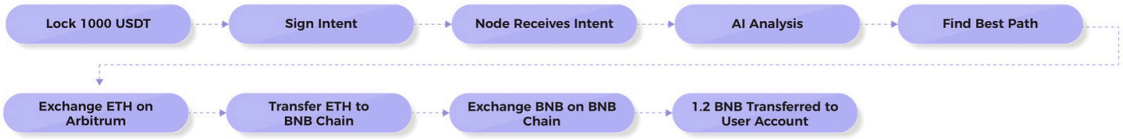
In this framework, users only need to sign a trading intent, expressing their expectations for the final outcome. Protocol nodes play a crucial role in this model, with primary responsibilities including:

- Cross-Chain Liquidity Assessment and Selection: Protocol nodes assess different liquidity providers across multiple blockchain networks. These providers can include decentralized exchanges (DEX), market makers, etc.
- Path Optimization and Execution: Based on user-defined constraints (such as minimum received amount, trading costs, execution speed, etc.), protocol nodes select the optimal cross-chain liquidity path through smart routing and algorithms, ensuring that the trading results align with user intent.
- Security Assurance: Protocol nodes ensure the safety of user funds through smart contract mechanisms. During the transaction process, user assets are custodied in smart contracts, and funds are only released to liquidity providers upon successful transaction completion, reducing risk.

Example of Intent-Centric Omni-Chain Liquidity Transaction

Suppose a user wants to convert 1000 USDT on Ethereum to BNB on BNB Chain and requires at least 1.2 BNB. This intent may involve multiple chain liquidity providers and complex cross-chain paths. The specific process is as follows:

1. Submission of User Intent: The user locks 1000 USDT on Ethereum through a smart contract and signs an intent indicating they expect to receive at least 1.2 BNB on BNB Chain.
2. Path Optimization by Protocol Nodes: After receiving the user's intent, After receiving the user's intent, the protocol nodes use AI algorithms to analyze historical data and current market conditions, searching for the best liquidity path across the entire chain. The nodes might first exchange USDT for ETH on Arbitrum, then transfer ETH to the BNB Chain, and finally exchange ETH back to BNB on the BNB Chain. This multi-chain operation can more effectively utilize liquidity pools and exchange rate advantages on different chains, ultimately helping users achieve their desired trading outcomes.
3. Path Execution and Settlement: Once the optimal path is selected, the transaction execution begins. The AI algorithm not only helps nodes find the best trading path but also minimizes trading costs and time, ensuring users receive optimal liquidity. Then, the 1000 USDT on Ethereum is released by the smart contract and completes the cross-chain transfer according to the pre-set path. After the transaction is completed, 1.2 BNB on the BNB Chain is transferred into the user's designated account.



Advantages of this Model

- Integration of Omni-Chain Liquidity: Through protocol nodes, the system can integrate various liquidity sources across chains, including market makers, liquidity pools, and lending protocols. This allows users to leverage different liquidities on multiple chains, ensuring the flexibility and efficiency of cross-chain transactions.
- Safety and Custody Mechanism for Funds: User assets are custodied by smart contracts during the transaction process and funds are only released when the transaction completes and meets user expectations. This mechanism effectively reduces the risk of fund loss or theft, ensuring safety and transparency.
- Optimized User Experience and Cost Reduction: Users do not need to directly participate in the complex operations of multi-chain transactions; protocol nodes automatically select the optimal path for them, enhancing the convenience of cross-chain transactions. Additionally, through smart path optimization, the system maximizes liquidity utilization, reduces trading costs, and minimizes efficiency losses caused by on-chain friction.
- AI-based: AI algorithms can predict changes in liquidity and trading demands, adjust liquidity pool configurations in advance, and ensure that the system always has sufficient liquidity to support cross-chain transactions at any point in time. At the same time, AI algorithms can find the best trading paths, minimize transaction costs and time, and ensure that users obtain the highest quality liquidity.

Intent Data Encoding

The intent data structure simplifies user interactions and optimizes on-chain storage and computational efficiency. The user-submitted intent includes the initiator's address, intent data, and signature, with the specific structure as follows:

```
struct UserIntent {
    address sender;           // User address initiating the intent
    uint256 amount;           // Transfer amount
    string targetChain;       // Target chain
    uint256 minReceiveAmount; // Minimum receive amount
    uint256 deadline;         // Transaction deadline
    bytes additionalData;     // Additional data
    bytes signature;          // User signature for the intent
}
```

- sender: The address of the user to initiate the intent.

- amount: The transfer amount, indicating the quantity of assets the user wishes to transfer.
- targetChain: The target chain where the user wants to transfer assets.
- minReceiveAmount: The minimum amount the user wishes to receive on the target chain.
- deadline: The deadline by which the user wants the transaction to be completed.
- additionalData: Other supplementary data, possibly including transaction notes or specific transaction conditions.
- signature: The user's signature on the intent, ensuring data authenticity and immutability.

Protocol nodes parse the intent off-chain and generate optimal paths and solutions through off-chain computation, reducing on-chain interaction costs and improving transaction confirmation speed.

Transaction Split

In the omni-chain liquidity trading model, protocol nodes optimize trade execution through dynamic splitting and liquidity aggregation. Upon receiving a user's cross-chain intent, nodes rely on AI algorithms to analyze historical data and current market conditions, predict liquidity changes, and choose the optimal path. Trades are optimized according to algorithms and, if necessary, split into multiple smaller amounts for execution across various paths. This mechanism achieves minimum costs and maximum liquidity utilization through dynamic splitting and liquidity aggregation:

- Multi-Path Selection: Protocol nodes assess multiple paths that can partially meet trading needs rather than relying on a single liquidity pool. By evaluating liquidity resources across multiple chains, nodes can utilize liquidity provided by various decentralized exchanges or liquidity pools.
- Liquidity Aggregation: Based on path cost and efficiency, protocol nodes rank different paths and select the optimal ones, ensuring maximum liquidity and execution efficiency through these paths.
- Split Execution: Protocol nodes split the transaction into several smaller parts and execute them at the same time in different chains, ensuring optimal utilization of transaction liquidity and cost minimization.

This mechanism effectively addresses the liquidity needs of large trades while enhancing the execution efficiency of smaller trades, preventing any single chain or liquidity provider from becoming bottlenecked.

Transaction Split Algorithm

The goal of the transaction split algorithm is to allocate trades to multiple liquidity providers to minimize total fees while ensuring funding needs are met.

Input Parameters:

- T: Total amount requested for cross-chain transfer by the user.
- c_i: Maximum available liquidity that liquidity provider i can handle.
- f_i: Fee rate charged by liquidity provider i.
- x_i: Amount allocated from liquidity provider i.

Objective Function:
We aim to minimize the total transaction fee *F* , given by the formula:

$$F = \sum_{i=1}^{n} f_i \cdot x_i$$

Constraints:
1. Satisfy the user's total funding requirement:

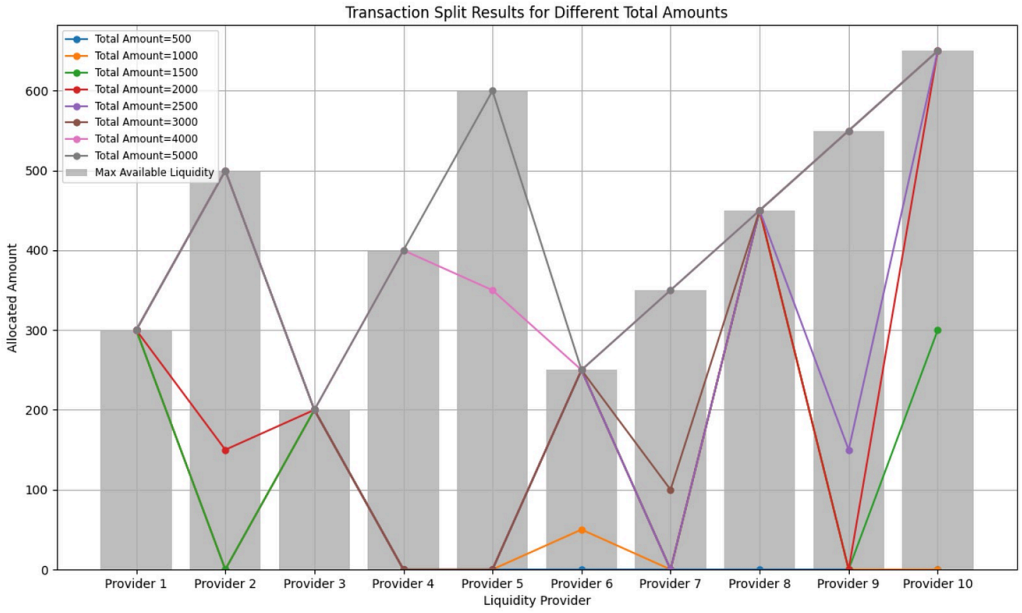
$$\sum_{i=1}^{n} x_i = T$$

2. The allocation amount for each liquidity provider cannot exceed its available funds:

$$0 \leq x_i \leq c_i \quad \forall i \in \{1, 2, \ldots, n\}$$

Algorithm Steps:

1. Sort all liquidity providers by fee *fi* from low to high.
2. Allocate funds starting from the provider with the lowest fee until the provider's fund limit *ci* is reached.
3. If the first provider is insufficient to meet the total requirement *T* , continue allocating to the next higher-cost provider until the user's requirement is met.



Obtaining liquidity from different providers through transaction split to achieve optimal liquidity.

Liquidity Prediction Model

Protocol nodes use AI algorithms to predict fund flows for different chains and liquidity pools. Given the time series $L(t)$ of liquidity data within the time interval $[t0, tn]$, nodes perform prediction and optimization through the following steps:

1. Feature Extraction and Modeling:
Use a sliding window method to extract features from $L(t)$, with window size w representing the liquidity state of the past w timepoints. Nodes learn the liquidity change trend through the following regression model:

$$\hat{L}(t+\Delta t) = \alpha_0 + \sum_{i=1}^{p} \alpha_i L(t-i)$$

where $\alpha 0, \alpha 1, \ldots, \alpha p$ are model parameters, and $L\hat{}(t + \Delta t)$ is the predicted liquidity value for the next time step.

2. Liquidity State Update:

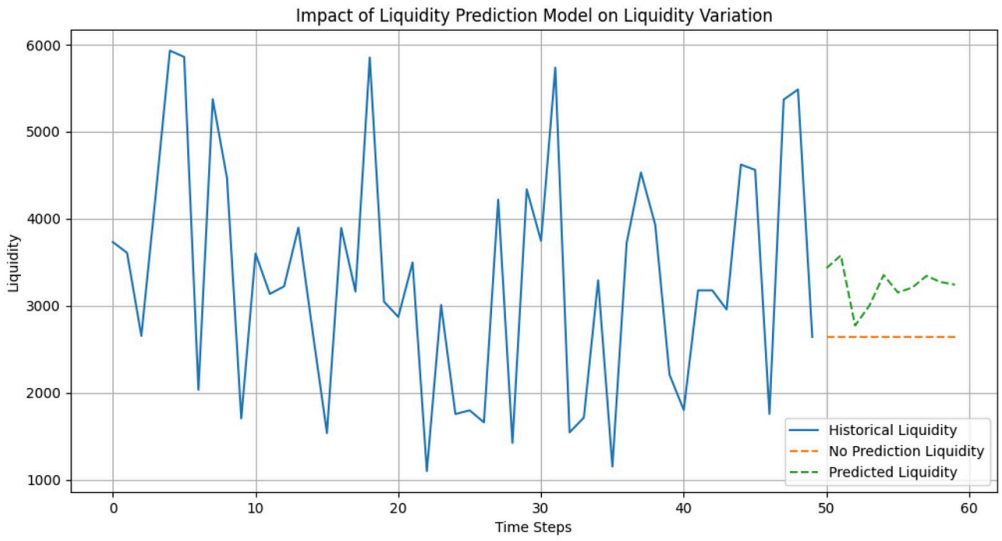
At each time step, the system dynamically adjusts the fund pool configuration based on the predicted liquidity distribution $L\hat{}(t + \Delta t)$ to minimize liquidity risk and transaction costs in cross-chain trading.

3. Optimization Objective:

Define *C*(*t*) as the liquidity cost function on different chains, with the goal of minimizing the total system cost *J* while meeting cross-chain trading needs:

$$J = \int_{t_0}^{t_n} C(t)dt$$

Through such time series prediction and optimization, protocol nodes can adapt to market changes in advance, dynamically adjust fund pool configurations, and ensure maximum liquidity utilization efficiency.



Liquidity changes over time, and both liquidity and transaction prices fluctuate. We use deep learning to predict liquidity based on historical data.

Route Selection Mechanism

In the intent-centric full-chain liquidity trading model, transactions undergo splitting and path optimization to maximize trading efficiency and reduce costs. Protocol nodes integrate full-chain liquidity resources and use intelligent algorithms to determine the optimal trading path, ensuring user trading needs are met.

Protocol nodes are responsible for the following steps in path selection based on user trading intents:

- **Path Filtering**: Protocol nodes identify multiple available liquidity pools, decentralized exchanges (DEXs), and other liquidity sources on the chain to meet user trading intents.
- **Sorting and Selection**: For each path, nodes prioritize based on fees, execution time, liquidity depth, and other factors to ensure the selection of the path with the lowest trading cost and highest execution efficiency.

For example, if a user wants to convert USDT on Ethereum to BNB on BNB Chain, protocol nodes will integrate liquidity across multiple chains, such as exchanging on the Arbitrum intermediary chain and then transferring to BNB Chain to ensure the transaction meets user expectations.

Routing Algorithm

The routing algorithm is used to select the optimal path among multiple available ones, minimizing total costs associated with cross-chain fees and execution time.

Input Parameters:

- P_j: The j-th path.
- F_j: Total fees for path P_j.
- T_j: Total execution time for path P_j.
- α: User preference parameter, determining the weight of fees and time, $0 \leq \alpha \leq 1$.

Objective Function:
We aim to minimize the total cost *Cj* for path *Pj* , given by the formula:

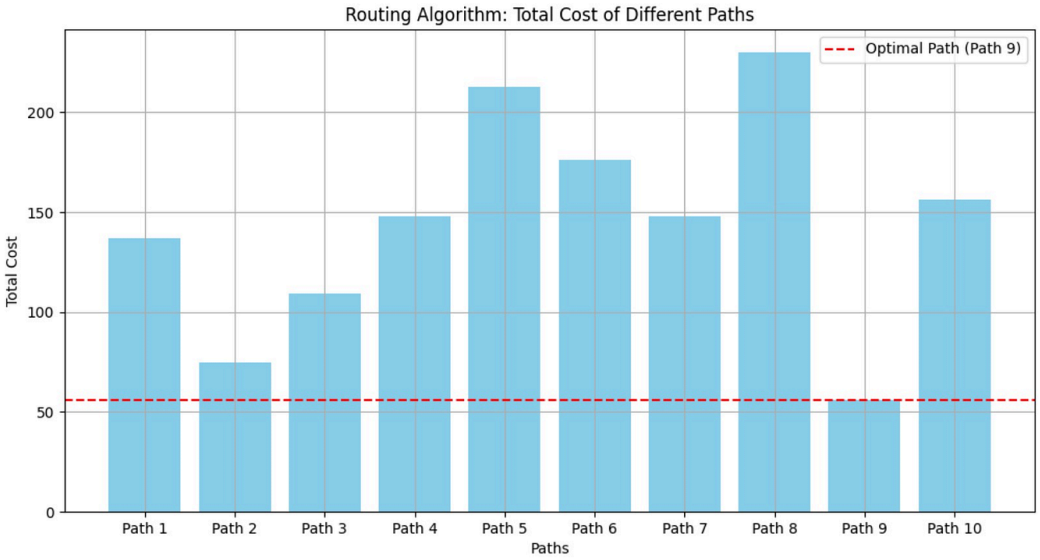
$$C_j = \alpha \cdot F_j + (1 - \alpha) \cdot T_j$$

Constraints:

1. The total funds of all liquidity providers on path *Pj* need to meet the user's funding requirement:

$$\sum_{i \in P_j} c_i \geq T$$

2. Users can set maximum tolerance limits for fees or time.



Selecting the optimal trading execution path to minimize total costs.

Finding the Best Trading Path through AI Algorithms

1. **Features**
Extract key features affecting trading paths from historical trading data, including path fees, execution time, liquidity depth, market volatility, etc. Let these features be $X = \{x1 , x2 , \dots , xn \}$, where *xi* is the feature vector for each path.

2. **Supervised Learning Model**
Use regression and neural network algorithms to predict the cost of trading paths. The goal is to find the optimal path *Pj* and minimize the objective function *Cj* :

$$C_j = \alpha \cdot F_j + (1 - \alpha) \cdot T_j$$

where *Fj* is the path fee, *Tj* is the execution time, and *α* is the user preference weight.

3. **Dynamic Optimization and Prediction**

Combine historical data and use LSTM time series models to predict future liquidity changes *L^*(*t*) and market state *M*(*t*). Use the predicted liquidity *L^*(*t*) and current state *S*(*t*) as inputs to adjust path selection and splitting rules, ensuring:

$$\sum_{i=1}^{n} c_i(t) \geq T$$

i.e., ensuring each selected path meets the user's total requirement T.

4. **Deep Reinforcement Learning**

Use Q-learning to deep reinforcement learning further optimize path selection selection. At each trading step *t*, the intelligent agent selects action *at* (i.e., path or splitting strategy) based on the current market state *S*(*t*) and prediction *M*(*t* + 1) to maximize reward *R*(*t*), i.e., minimize trading costs and execution time. Update the strategy $\pi(a \mid s)$ through the reward function *R*(*t*) at each step:

$$R(t) = -C_j(t)$$

Iteratively adjust the path selection strategy to gradually learn the optimal decision.

Optimizing trading paths based on AI.

By splitting algorithms and multi-path selection, intent transactions are split across multiple chains and executed simultaneously to improve liquidity utilization and trading speed.

- **Transaction Splitting Algorithm**: For user transactions, split requests into multiple parts and allocate them to different liquidity providers. This splitting algorithm optimizes based on each provider's fees, liquidity depth, and available quota to ensure minimal transaction fees and meet liquidity needs.
- **Intelligent Routing Algorithm**: Use user preference weight α to perform weighted calculations between fees Fj and execution time Tj , selecting the best cross-chain path. The algorithm dynamically evaluates market conditions in real-time to ensure users get the optimal trading path.

Transaction splitting and intelligent routing optimization are the core of the protocol, jointly forming the core logic of cross-chain trading. Through algorithmic solutions, the best path selection and fund allocation are achieved, maximizing user trading experience under efficient and low-cost conditions.

In the intent-centric full-chain liquidity trading model, protocol nodes serve as core components, parsing and executing user cross-chain trading intents in real-time. Nodes optimize trading processes through full-chain liquidity resource integration, historical data analysis, market conditions, and AI algorithms, achieving liquidity aggregation and split execution which minimize trading costs and maximize execution efficiency.

Protocol Nodes

Protocol nodes continuously collect and parse user intents through off-chain computation, combining historical data, market conditions, and AI algorithms to optimize cross-chain trading paths and provide multi-path trading solutions to ensure efficient execution of cross-chain transactions:

- **Multi-Path Trading Execution**: Protocol nodes use intelligent routing algorithms to integrate liquidity sources across multiple chains and dynamically optimize trading paths. Nodes combine decentralized exchanges and liquidity pools, using intelligent routing algorithms to provide users with the best path, ensuring low costs and efficient execution.
- **Complex Strategy Execution**: Based on the complexity of the transaction, protocol nodes can use splitting and aggregation strategies to schedule funds and execute arbitrage operations on different chains to achieve the best trading results.

The intent-centric full-chain liquidity trading model enhances trading execution efficiency through protocol nodes' path optimization and liquidity integration. Transactions undergo splitting and path optimization, achieving optimal liquidity utilization, reducing trading costs, and increasing the success rate of cross-chain operations.

In the protocol based on full-chain liquidity trading, node behavior is crucial to the overall stability and security of the system. To encourage honest and efficient node operations while preventing malicious behavior or operational errors, the protocol introduces strict **node penalty mechanisms** and diverse **node incentive mechanisms**. This not only ensures the normal operation of the system but also provides nodes with sustainable economic returns.

Node Incentives

To encourage active participation and honest execution by nodes, the protocol designs attractive incentive mechanisms. Nodes not only earn additional rewards through staking but also receive fee shares and token incentives for successfully executing trading paths.

In the protocol's economic model, nodes participate in transaction verification and liquidity management by staking protocol tokens. As participation and contribution increase, nodes receive protocol tokens as rewards. This token incentive is distributed based on the transaction volume processed by the node, the liquidity provided, and the activity within the system. The token reward distribution formula can be described as:

$$R = \alpha \times V$$

where *R* is the node's token reward, *α* is the incentive coefficient, and *V* is the total transaction volume processed by the node. This mechanism ensures sustainable economic incentives for active and honest nodes, promoting the healthy development of the full-chain liquidity ecosystem.

Nodes also earn fee shares from successfully executed transactions. Cross-chain transactions involve liquidity and resources across multiple chains, and nodes optimize trading paths and liquidity allocation, earning a certain percentage of fees as rewards. This reward mechanism is as follows:

$$F = \beta \times \text{Fees}$$

where *F* is the node's fee reward, and *β* is the fee share ratio. Nodes can earn substantial economic returns by efficiently utilizing user cross-chain transactions, further incentivizing them to improve operational efficiency and comply with protocol regulations.

Node Penalty Mechanism

The node penalty mechanism ensures that all participating nodes maintain honest and efficient behavior through strict economic measures. If nodes violate the protocol's behavior standards, their staked assets will be slashed. These violations include but are not limited to the following categories:

- **Double Signing**: If a node generates conflicting signatures for the same transaction on different chains, it is considered malicious behavior, resulting in significant slashing of staked assets.
- **False Liquidity Reporting**: Submitting false liquidity information or forging trading paths will trigger slashing directly.
- **Failure to Execute Transactions on Time**: If a node fails to complete transactions on time or does not follow the protocol's optimal path planning, it will also be penalized.

The slashing of a node's staked assets S is calculated as follows:

$$\Delta S = f(p) \times S$$

where ΔS is the slashed amount, and $f(p)$ is an increasing function of the violation severity p. This function adjusts based on the severity of the violation, from minor operational errors to severe malicious behavior (such as double signing), with increasing slashing magnitude.

Through the dual constraints and incentives of the **node penalty mechanism** and **node incentive mechanism**, the protocol ensures the efficiency, security, and stability of the full-chain liquidity trading system. The node incentive mechanism and penalty mechanism together form the protocol's incentive structure, ensuring that nodes maintain honest and efficient behavior while earning economic benefits. If nodes engage in violations, they face slashing of staked assets and loss of future token incentives and fee shares. This mechanism uses economic leverage to encourage nodes to play an active role in the system and continuously provide high-quality liquidity support for users.

Technical Architecture

1. **Cross-Chain Liquidity Pool**:

- Integrating liquidity pools across multiple chains to construct full-chain liquidity aggregation.
- Managing liquidity pools through smart contracts to ensure fund security and efficient utilization across chains.
- AI analyzes historical data to predict liquidity trends and dynamically adjust fund pool configurations.
- Providing real-time liquidity monitoring and reporting to ensure the transparency and auditability of liquidity pools.

2. **Cross-Chain Trading Protocol**:

- Providing a seamless cross-chain trading protocol with automated trading to ensure transaction transparency and decentralization.
- Using AI to optimize trading execution, predict liquidity, and reduce transaction costs.

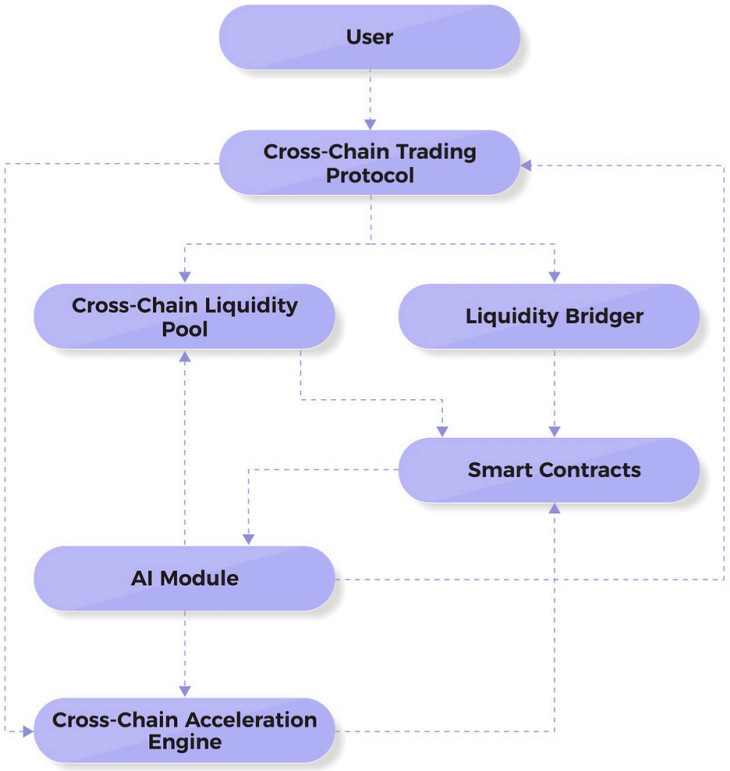
- Supporting various trading types, including limit orders, market orders, and stop-loss orders, to meet different user needs.

3. **Liquidity Bridge**:

- Achieving liquidity bridging between chains to ensure efficient and secure fund flow during cross-chain transactions.
- Reducing security risks through liquidity leakage prevention mechanisms.
- Supporting various cross-chain bridging methods, including atomic swaps and lock-release mechanisms, to ensure flexibility and security in cross-chain operations.

4. **Cross-Chain Acceleration Engine**:

- Using intent-driven, zero-knowledge proofs, and smart contract technology to improve the efficiency and security of cross-chain transfers and reduce costs.
- Dynamically adapting to on-chain liquidity changes to ensure full-process decentralization.
- Integrating AI algorithms to optimize trading paths and liquidity configurations in real-time, ensuring efficient system operation.



Conclusion

This protocol constructs an efficient and secure cross-chain acceleration engine by integrating intent, zero-knowledge proofs, smart contracts, and AI technology, aiming to address the current complexity of cross-chain trading and insufficient market liquidity. Through transaction splitting algorithms and intelligent trade routing algorithms, the protocol can obtain the best liquidity across multiple chains and select the optimal path to minimize transaction costs.

Core Advantages and Features:

1. **Liquidity Depth**: Integrating liquidity pools across various chains to provide deeper liquidity and reduce slippage.
2. **Seamless Cross-Chain Trading**: Allowing users to trade seamlessly between different chains, simplifying cross-chain operations and reducing transaction costs.
3. **Price Consistency**: Reducing asset price differences on different chains through full-chain liquidity, providing more consistent prices.
4. **Security and Transparency**: Achieving automated and decentralized trading through smart contracts, ensuring transaction security and transparency.
5. **Efficient Cross-Chain Transfers**: Using intent, zero-knowledge proofs, and smart contract technology to reduce cross-chain transfer costs and improve cross-chain efficiency.
6. **AI Optimization**: Using AI algorithms to predict liquidity changes and trading demand, optimizing trading paths to ensure users obtain the best liquidity.

Application Scenarios:

1. **Cross-Chain Exchanges**: Providing a seamless cross-chain trading platform that allows users to trade between different chains.
2. **Liquidity Aggregators**: Integrating liquidity pools across various chains to provide deeper liquidity and reduce slippage.
3. **Cross-Chain Payments**: Achieving payments between different chains through cross-chain liquidity, simplifying cross-chain payment operations.
4. **Intelligent Trade Optimization**: Using AI to predict market changes and optimize trading paths, enhancing trading efficiency and reducing costs.

Through these technologies, the protocol not only addresses the complexity of cross-chain trading and insufficient market liquidity but also significantly enhances trading efficiency and security, providing users with an efficient and decentralized cross-chain trading platform.

References

1. Sergey Nazarov, “Chainlink: A Decentralized Oracle Network,” Chainlink White Paper, 2017. [Online]. Available: https://chain.link/whitepaper
2. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor, “Making Smart Contracts Smarter,” Proceedings of the 2016 ACMSIGSAC Conference on Computer

and Communications Security, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2978309
3. Arjun Bhuptani, “Connext: Scalable Cross-Chain Communication,” Connext White Paper, 2021. [Online]. Available: https://connext.network/docs/whitepaper.pdf
4. Dan Robinson, Georgios Konstantopoulos, “Optimistic Rollups: Scaling Ethereum Smart Contracts,” 2020. [Online]. Available: https://research.paradigm.xyz/rollups.pdf
5. Harry Kalodner, Steven Goldfeder, Alishah Chator, Malte Möser, Arvind Narayanan, “Arbitrum: Scalable, private smart contracts,” 2018. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-kalodner.pdf
6. Alex Evans, “Liquidity Mining: A User-Centric Approach to Bootstrapping Liquidity,” Placeholder, 2020. [Online]. Available: https://www.placeholder.vc/blog/2020/9/1/liquidity-mining
7. Michael Egorov, “Curve Finance: A Deep Dive into Automated Market Makers,” Curve Finance, 2020. [Online]. Available: https://curve.fi/files/CurveFinanceWhitepaper.pdf
8. Ashleigh Schap, “Uniswap v3: A Flexible AMM for DeFi,” Uniswap, 2021. [Online]. Available: https://uniswap.org/whitepaper-v3.pdf
9. “Machine Learning for Financial Market Prediction,” Journal of Financial Data Science, 2020. [Online]. Available: https://jfds.pm-research.com/content/early/2020/01/22/jfds.2020.1.1