Zenlink Whitepaper v0.6

Preface

Generally speaking, Defi (decentralized finance) include a wide meaning, and can be roughly divided into four types:

1. DEX, like Kyber and Uniswap.
2. Loan category, like Compound and Lend.
3. Derivatives and prediction markets are represented by GNO and SNX.
4. Oracle and other types, representatives include BAND and REN.

Among them, DEX is the most eye-catching. After the ups and downs of the past three years (2017~2019), a substantial outbreak comes in 2020. In 2017, there was only one decentralized exchange (IDEX) whose annual transaction volume was less than $5 million. In 2018, DEX trading volume achieved explosive growth, with trading volume reaching $2.7 billion. In 2019, DEX trading volume shrank slightly but still exceeded $2.5 billion.
In 2020, DEX gets rapid development, and its Q1 trading volume ($2.3 billion) almost equaled the full-year trading volume of 2019. Total trading volume in the Q2 risen to a record high, $3.7 billion. We expect DEX to maintain this development trend in the second half of the year and develop rapidly.

Compared with hot Defi and DEX, there is also the officially released 3rd generation blockchain project, Polkadot. Compared with the existing blockchain network, Polkadot network has several obvious advantages, including heterogeneous sharding, scalability, upgradeability, transparent governance, and cross-chain composability.
Heterogeneous sharding network: Polkadot is essentially a sharding blockchain, but each shard is a parachain, which means that it connects multiple chains in a network, allowing them to process transactions in parallel while sharing the security provided by the underlying relay-chain.
Scalability: By bridging multiple dedicated chains into a sharded network, Polkadot allows multiple transactions to be processed in parallel. It solves the performance bottleneck of the blockchain network. In the future, through nested relay chains, the number of parachains (shards) in the network can be further expanded.
No-fork upgrade: Polkadot enables the blockchain to upgrade with no-forks. These upgrades are achieved through Polkadot's transparent on-chain governance system.
On-chain governance: All DOT holders can make propose or vote on existing proposals. They can also help select board members who represent passive stakeholders in the Polkadot governance system.
Cross-chain composability: Polkadot's cross-chain composability and message passing allow shards to communicate, exchange value and share functions, and can interact with existing blockchain networks or encrypted assets.

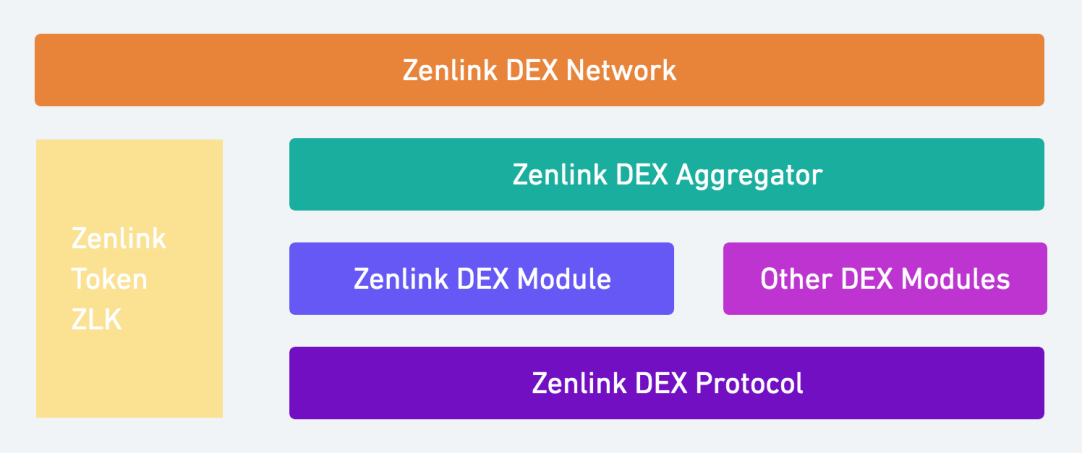
Based on the prediction of the further growth of the DEX ecosystem in the future and the rapid development of the public blockchain technology, we propose a Polkadot network-based, high-liquidity, upgradeable, cross-chain DEX network, Zenlink.

Overall

Zenlink is committed to building a new generation of cross-chain DEX network. By integrating the Zenlink DEX Module, Zenlink can enable parachains to quickly possess DEX capabilities and share liquidity with other parachains; Zenlink DEX aggregator can link all DEX DApps on Polkadot. Users can not only complete the exchange easily and quickly, but also enjoy a low slippage transaction experience; Zenlink's native token ZLK provides a fair and transparent governance mechanism and reasonable value capture methods to incentivize ecological users to participate in long-term network development.

Zenlink is a cross-chain DEX network based on Polkadot. In general, Zenlink consists of the following parts:

1. Zenlink Protocol: The top-level unified general DEX protocol.
2. Zenlink DEX Module: A Module on Substrate Runtime Module Library (SRML) layer according to the Zenlink Protocol standard. Parachains can integrate with it quickly, so that be able to get the DEX function, even share the liquidity with other DEX on other parachains.
3. Zenlink DEX Aggregator: A simple and user-friendly entrance of DEX world which is able to connect with most of DEX on Polkadot, so that user can do one-click trade with multiple DEX on low slippage.
4. Zenlink Token: The native token of Zenlink DEX Protocol which can be used to distribute liquidity benefits, governance, etc
5. Zenlink DEX Network: A decentralized exchange network composed of Zenlink DEX Module, Zenlink DEX Aggregator and other DEX Application on parachains.



Zenlink Ecosystem

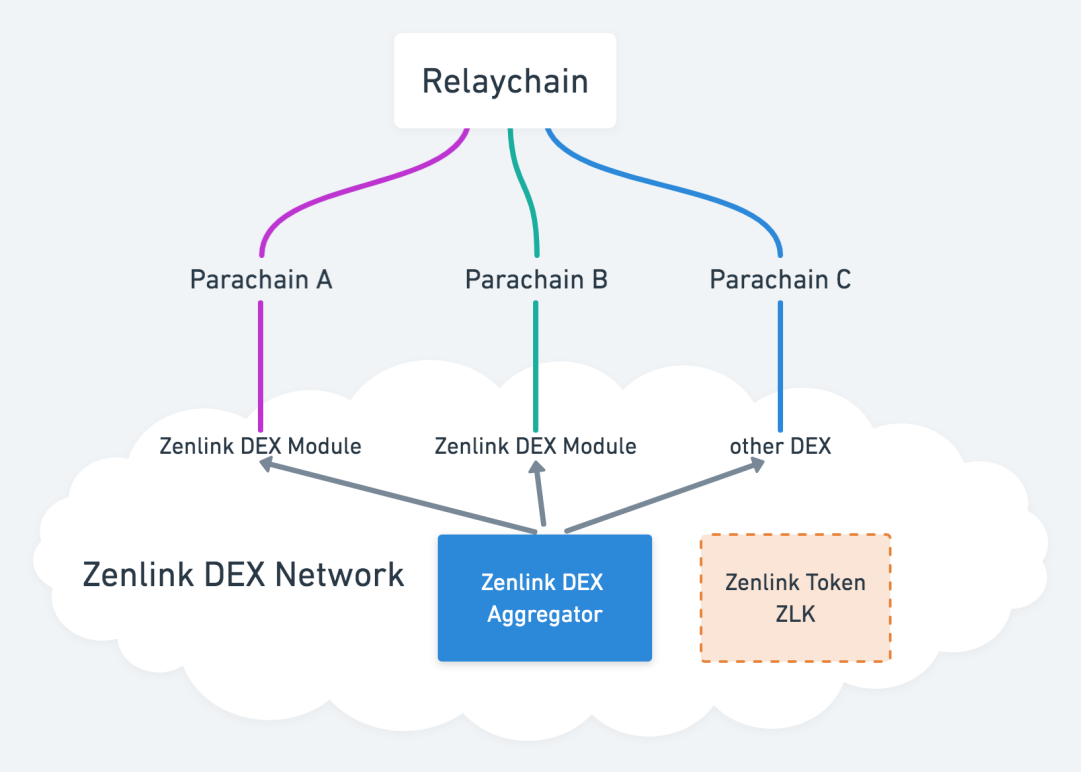
The whole planning of Zenlink

At first, Zenlink DEX Module will be implemented based on Zenlink DEX Protocol. Parachain can integrate with it quickly, so that be able to get the DEX function, even share the liquidity with other DEX on other parachains.In order to complete the Zenlink ecosystem, we would like to build a front-end website application, Zenlink DEX Dapp.

Besides, By the simple and smooth Zenlink DEX Aggregator that connects with most of all DEX on Polkadot network, users will do one-click trade with multiple DEX on low slippage seamlessly .

Furthermore, In order to provides a fair and transparent governance mechanism and reasonable value capture means to motivate ecosystem users to participate in the development of the network for a long time, Zenlink will issue its native token ZLK on the hight performance Polkadot network.

In general, in the early stage, by defining the trading protocol and realizing the trading module, we set up the first exchange application and exchange aggregator application on a parachain and issued the native token ZLK that takes into account both governance and incentive functions. Through the above stages, a complete Zenlink DEX network is formed.



In the medium to long term plan, we will work with other parachains projects to deploy the dex modules to more parachains, and we will also connect Zenlink DEX Aggregator with other Polkadot DEX applications to provide more trading pairs and greater liquidity.

Zenlink DEX Protocol

It is a top-level general decentralized trading protocol based on the Polkadot network. Its characteristics are:

- **Unified universal interface standard.** The advantage is that the contracts can be replaced with new ones at any time, so that the system can be upgraded and customized at any time. The application system can be easily extended to a wider network environment. The language-independent characteristics make all programmers fully available to write modules, as long as you can achieve this set of standards, you can access to the Zenlink DEX network, and so on.
- **Cross-chain interconnection.** In the protocol interface design, the communication characteristics of each parachain are abstracted to form an interoperable communication mechanism. Cooperate with the interoperability of the Polkadot network itself to achieve the message transmission between the parachains at the module layer.

Pallet Implementation

The ZenLink DEX Protocol defines several modules including Assets, Dex. Here is the basic function description, using the ABC/DOT trading pair as an example.

Module Assets

Assets is the fundamental module of ZenLink Dex Protocol. Assets looks like ERC20. Users can manage liquidity and token with it.

Functions

- issue: issue a new ERC20 token

```
fn issue(origin, #[compact] total: T::TokenBalance, asset_info: AssetInfo)
Parameters:
`total`: initial total supply.
`asset_info`: the asset info contains `name`, `symbol`, `decimals`.
```

- transfer: transfer token from owner to receiver.

```
fn transfer(origin, #[compact] id: T::AssetId, receiver: <T::Lookup as
StaticLookup>::Source, #[compact] amount: T::TokenBalance)
Parameters:
`target`: the receiver of the asset.
`amount`: the amount of the asset to transfer.
```

- approve: sets `amount` as the allowance of `spender` over the caller's tokens with token `id`. Returns a boolean value indicating whether the operation succeeded.

```
fn approve(origin, #[compact] id: T::AssetId, spender: <T::Lookup as
StaticLookup>::Source,  #[compact] amount: T::TokenBalance)
Parameters:
`spender`: the spender account.
`amount`: the amount of allowance.
```

- transfer_from: moves amount tokens from sender to recipient using the allowance mechanism. amount is then deducted from the caller's allowance. Returns a boolean value indicating whether the operation succeeded.

```
fn transfer_from(origin, #[compact] id: T::AssetId, from: <T::Lookup as StaticLookup>::Source, target: <T::Lookup as StaticLookup>::Source, #[compact] amount: T::TokenBalance)
Parameters:
`id`: the asset id.
`from`: the source of the asset to be transferred.
`target`: the receiver of the asset to be transferred.
`amount`: the amount of asset to be transferred.
```

Module DEX

Dex is the core module of ZenLink Dex Protocol. It implements the following functions:

- Initializing token trading pair.
- Token swap.
- Adding/extracting liquidity.
- Defining the liquidity constant function used throughout the protocol.

Functions

- create_exchange: initializing trading pair.

```
fn create_exchange(origin, token_id: T::AssetId) -> dispatch::DispatchResult
Parameters:
`token_id`: The exist asset's id.
```

- Token swap: swap dot to token. User specify the amount of the token to buy.

```
pub fn dot_to_token_output(origin, swap_handler: SwapHandlerOf<T>, tokens_bought: TokenBalance<T>, max_dot: BalanceOf<T>, deadline: T::BlockNumber, recipient: T::AccountId,) -> dispatch::DispatchResult
Parameters:
`swap_handler`: The wrapper of exchangeId and assetId to access.
`tokens_bought`: The amount of the token to buy.
`max_dot`: The maximum dot expected to be sold.
`deadline`: Time after which this transaction can no longer be executed.
`recipient`: Receiver of the bought token.
```

- add_liquidity/remove_liquidity: adding/extracting liquidity.

```
fn add_liquidity(origin, swap_handler: SwapHandlerOf<T>, dot_amount: BalanceOf<T>, min_liquidity: TokenBalance<T>, max_token: TokenBalance<T>, deadline: T::BlockNumber) -> dispatch::DispatchResult
Parameters:
`swap_handler`: The wrapper of exchangeId and assetId to transaction.
```

```
`dot_amount`: Amount of dot to deposit.
`min_liquidity`: Min amount of exchange shares(ZLK) to create.
`max_token`: Max amount of token to input.
`deadline`: Time after which this transaction can no longer be executed.
fn remove_liquidity(origin, swap_handler: SwapHandlerOf<T>, amount:
TokenBalance<T>, min_dot: BalanceOf<T>, min_token: TokenBalance<T>,
deadline: T::BlockNumber,) -> dispatch::DispatchResult
Parameters:
`swap_handler`: The wrapper of exchangeId and assetId to transaction.
`amount`: Liquidity amount to remove
`min_dot`: Minimum dot to withdraw.
`min_token`: Minimum token to withdraw.
`deadline`: Time after which this transaction can no longer be executed.
```

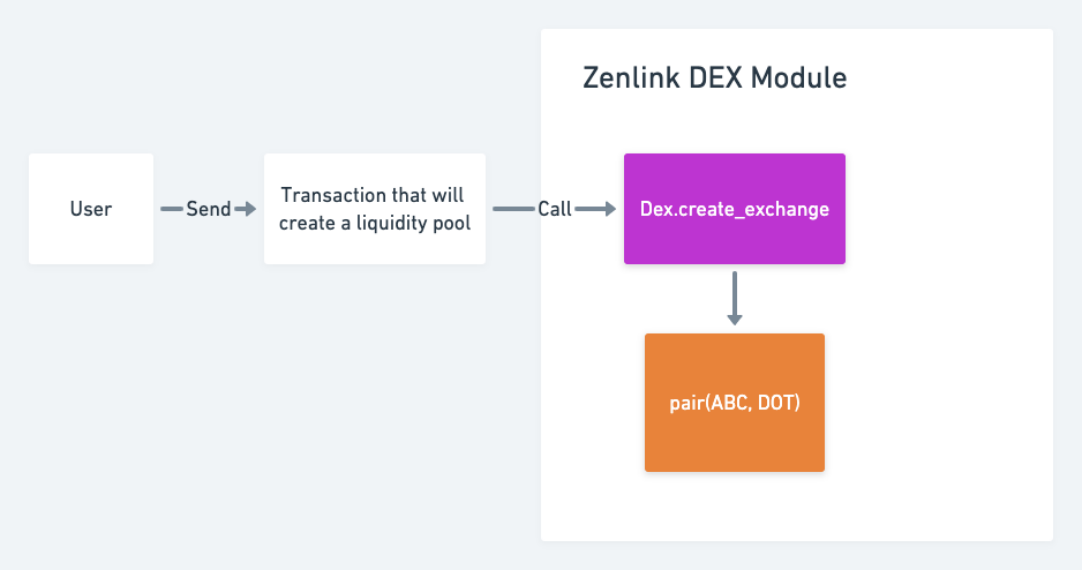
- get_output_price/get_input_price: Defining the liquidity constant function used throughout the protocol and calculating the return balance exactly.

```
fn get_output_price(output_amount: TokenBalance<T>, input_reserve:
TokenBalance<T>, output_reserve: TokenBalance<T>,) -> TokenBalance<T>
Parameters:
`output_amount`: Amount of Dot or Token the user want to bought.
`input_reserve`: Amount of Dot or Tokens (input type) in exchange
reserves.
`output_reserve`: Amount of Dot or Tokens (output type) in exchange
reserves
fn get_input_price(input_amount: TokenBalance<T>, input_reserve:
TokenBalance<T>, output_reserve: TokenBalance<T>,) -> TokenBalance<T>
Parameters:
`input_amount`: Amount of Dot or Token the user want to sold.
`input_reserve`: Amount of Dot or Tokens (input type) in exchange reserves
`output_reserve`: Amount of Dot or Tokens (output type) in exchange reser
```

The Class Diagram

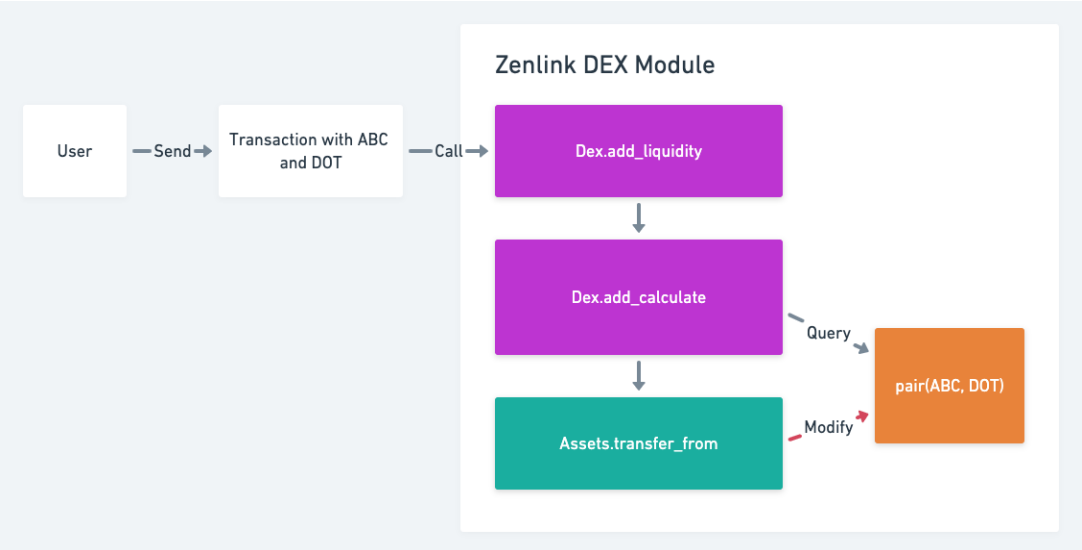
Based on some user scenarios, we will outline how modules in the protocol are called to each other.

Creating liquidity pool



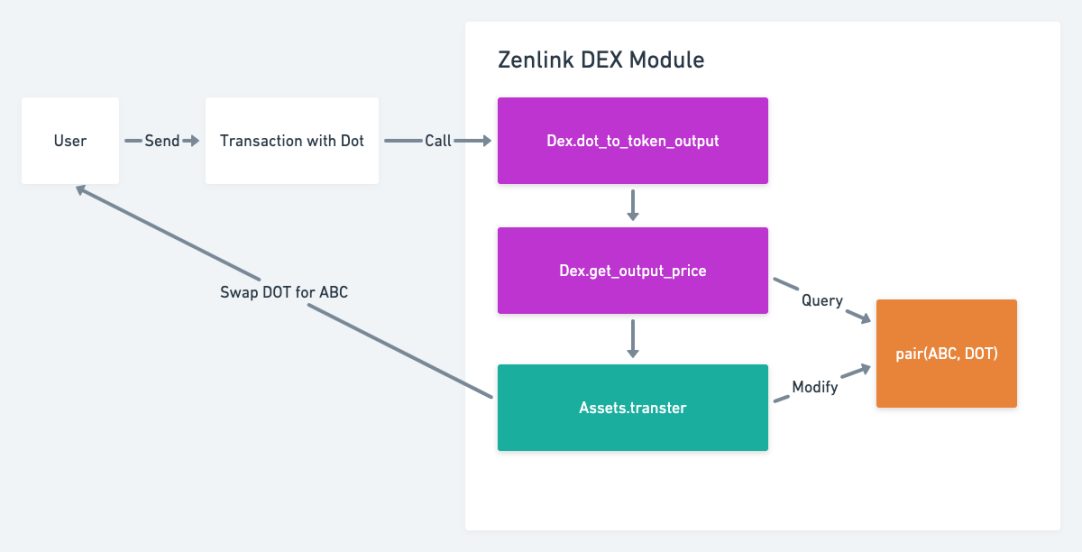
When user want to create a liquidity pool, the transaction would call the function create_exchange which exposes the public interfaces externally and belongs to contract factory. Then, a pair instance will be new.

Add liquidity



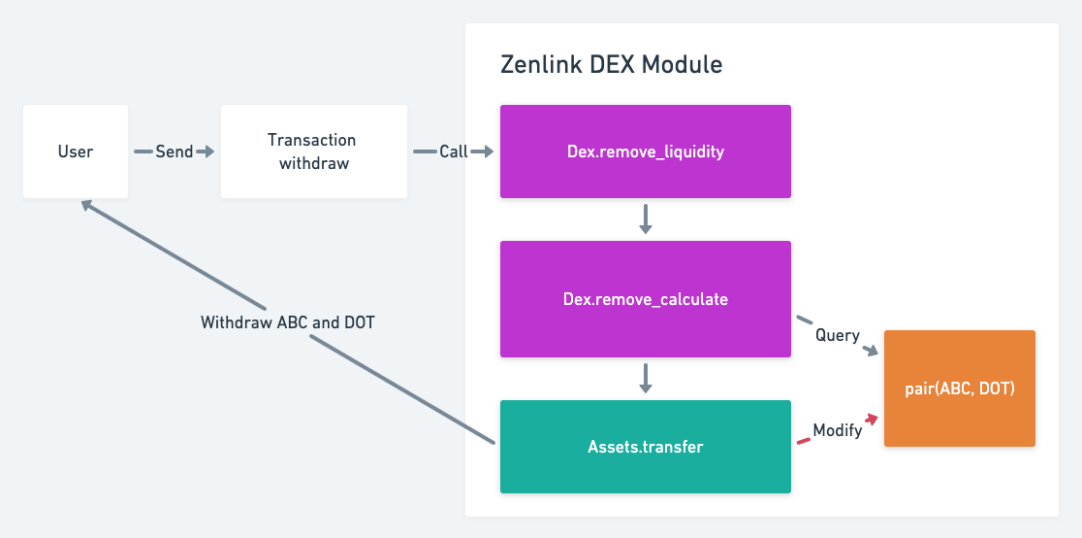
The function add_liquidity of contract exchange is exposed to public. The function add_alculate would give the correct amount of ABC and DOT which is required by the contant fucntion in the protocol. Then, the asset status would be changed.

Token swap



When the user want to trade with a liquidity pool, the transaction with DOT would call the function dot_to_token_output which exposes the public interfaces externally. Then, the function output_price would be in charge of the assets status estimation and modify. After that, the function transfer will send ABC to the user.

Remove liquidity



The function remove_liquidity of contract exchange is exposed to public. The function remove_calculate would give the correct amount of ABC and DOT which can be withdrawn from the liquidity pool. Then, the asset status would be changed, so that the user would receive ABC and DOT.

Smart Contract Implementation

This is other implementation of ZenLink DEX Protocol using ink! pallet as smart contract. It defines several contracts including Factory, Exchange.

Contract Factory

The factory contract can be used to create exchange contracts for any ERC20 token that does not already have one. It also functions as a registry of ERC20 tokens that

have been added to the system, and the exchange with which they are associated.

Interfaces

- initialize_factory: `set the Exchange contract wasm hashcode on the chain.

```
pub fn initialize_factory(&mut self,exchange_template_address : Hash)
Parameters:
  `exchange_template_address`: Exchange wasm hashcode on the chain
```

- create_exchange: create trading pair

```
pub fn create_exchange(&mut self, erc20_token_account : AccountId,
token_ammount : Balance)
Parameters:
`erc20_token_account`: The erc20 token account.
`token_ammount`: Ammount tokens transfer from erc20_token_address to the
account of exchange contract.
```

- get_exchange: get exchange account by token from the trading pair.

```
pub fn get_exchange(&self, token : AccountId)-> AccountId
Parameters:
`token`: A token account in a trading pair.
```

- get_token: Get Token account by token from the trading pair.

```
pub fn get_token(&self, exchange_account : AccountId)-> AccountId
Parameters:
`exchange_account`: a Exchange account in a trading pair.
```

Contract Exchange

Exchange is the core contract of ZenLink Dex Protocol. It implements the following interfaces:
Initializing token trading pair.

- Token swap.
- Adding/extracting liquidity.
- Defining the liquidity constant function used throughout the protocol.

Interfaces

- new : constructor of the Exchange contract

```
pub fn new(token_account_id: AccountId, factory_account_id : AccountId,
deployer : AccountId,token_ammount : Balance) -> Self
Parameters:
`token_account_id`: AccountId of a Erc20 token which trade on this
contract.
```

```
`factory_account_id`: AccountId of the Factory which instantiate this contract.
`deployer`: Account deploy this contract and provide initial liquidity.
`token_ammount`: Ammount of token the deployer will transfer from token_account_id to this contract account
```

- Token swap: swap dot to token. User specify the amount of the token to buy.

```
pub fn dot_to_token_transfer_output(&mut self, tokens_bought : Balance, deadline : Timestamp, recipient: AccountId) ->Balance
Parameters:
`tokens_bought`: The amount of the token to buy.
`deadline`: Time after which this transaction can no longer be executed.
`recipient`: Receiver of the bought token.
```

- add_liquidity/remove_liquidity: adding/extracting liquidity.

```
pub fn add_liquidity(&mut self, min_liquidity: u128, max_tokens: u128, deadline: Timestamp) ->Balance
Parameters:
`min_liquidity`: Min amount of exchange shares(ZLK) to create.
`max_token`: Max amount of token to input.
`deadline`: Time after which this transaction can no longer be executed.
pub fn remove_liquidity(&mut self, ammount : Balance,min_dot : Balance, min_token : Balance, deadline : Timestamp ) ->(Balance, Balance)
Parameters:
`amount`: Liquidity amount to remove
`min_dot`: Minimum dot to withdraw.
`min_token`: Minimum token to withdraw.
`deadline`: Time after which this transaction can no longer be executed.
```

- output_price/input_price: Defining the liquidity constant function used throughout the protocol and calculating the return balance exactly.

```
fn output_price(&self, output_ammount: Balance, input_reserve : Balance, output_reserve : Balance) -> Balance
Parameters:
`output_amount`: Amount of Dot or Token the user want to bought.
`input_reserve`: Amount of Dot or Tokens (input type) in exchange reserves.
`output_reserve`: Amount of Dot or Tokens (output type) in exchange reserves.
fn input_price(&self, input_ammount : Balance, input_reserve : Balance, output_reserve : Balance) -> Balance
Parameters:
`input_amount`: Amount of Dot or Token the user want to sold.
`input_reserve`: Amount of Dot or Tokens (input type) in exchange reserves
```

```
14 `output_reserve`: Amount of Dot or Tokens (output type) in exchange
reserves.
```

Zenlink DEX Contract

We implemented a general and stable Zenlink DEX Contract with ink! which is a Rust-based eDSL for writing Wasm smart contracts, according to the Zenlink Protocol standard. Its characteristics are:

- Are inherently safer to the network
- Have built in economic incentives against abuse.
- Have computational overhead to support graceful failures in logic.
- Have a lower bar to entry for development.
- Enable fast pace community interaction through a playground to write new logic.

Trading Paradigms

Trading paradigms can generally be divided into two types: the OrderBook model and the automated market maker model(AMM). AMM itself has a long pedigree, and now the blockchain Defi applications is making it even more accessible to the public. In today's Defi context, AMM is mostly "constant function market maker (CFMM)". Let's use the term AMM for the moment to follow the general convention. AMM (CFMM) is adopted by a lot of DEX, such as Uniswap, Balancer, Curve, etc. Its characteristics include:

- A trader trades with a pool of assets rather than with a particular counterparty.
- A specific mathematical formula is used to maintain the liquidity pool and provide a stable trading environment.
- There is no need for a matching system, relying instead on its own mathematical formula to automate the settlement.
- Support users to inject liquidity pool.

From what has been discussed above, Zenlink initially considered using the Constant Function Market Maker(CFMM) model to provide a stable and simple trading paradigm for the Polkadot ecosystem and cold-start liquidity sharing. In the later stage, as we gradually develop, we will consider converting to a Constant Mean Market Maker(CMMM) model. Zenlink will provides an n-dimensional automatic market maker for liquid mining. Users can provide up to n tokens to the liquidity pool, and can set the relative weight in the liquidity pool for each token, and automatically rebalance the user's portfolio according to price fluctuations.

CFMM

We'll start with a simple CFMM mode to build the system and the simplest one is nothing more than:

$$x * y = K$$

Consider a decentralized exchange that trades two tokens X and Y. Let x and y be the number of tokens X and Y, respectively. Then, keep the constant before and after the

exchange.

That is, when some one sell Δx tokens, he will get Δy tokens such that:

$$x * y = (x + \Delta x) * (y - \Delta y) = K$$

So that, the Δy should be

$$\Delta y = y - \frac{K}{x + \Delta x}$$

The price p should be

$$p = \frac{\Delta x}{\Delta y} = \frac{\Delta x(x + \Delta x)}{y(x + \Delta x) - K}$$

Therefore, the user only needs to provide Δx which is the amount of token X he want to sell, and by automatically calculating the program within the module, we can provide the price p and Δy which he will get ideally.

Liquidity Pool

Liquidity is essential to the creation and development of financial markets. AMM DEX typically has its own liquidity pool. The liquidity pool is essentially a pool of tokens set up in a Zenlink trading module or smart contract. The liquidity pool in the Zenlink trading module has the following characteristics:

- Users are free to create liquidity pools, which means they are free to add trade pairs to Zenlink DEX Network.
- Each trading pair liquidity pool depends on the establishment of each parachain, itself independent of each other.
- Transactions involving multiple liquidity pools can be matched by Zenlink DEX Aggregators.
- The liquidity pool is maintained by smart contract without human intervention.

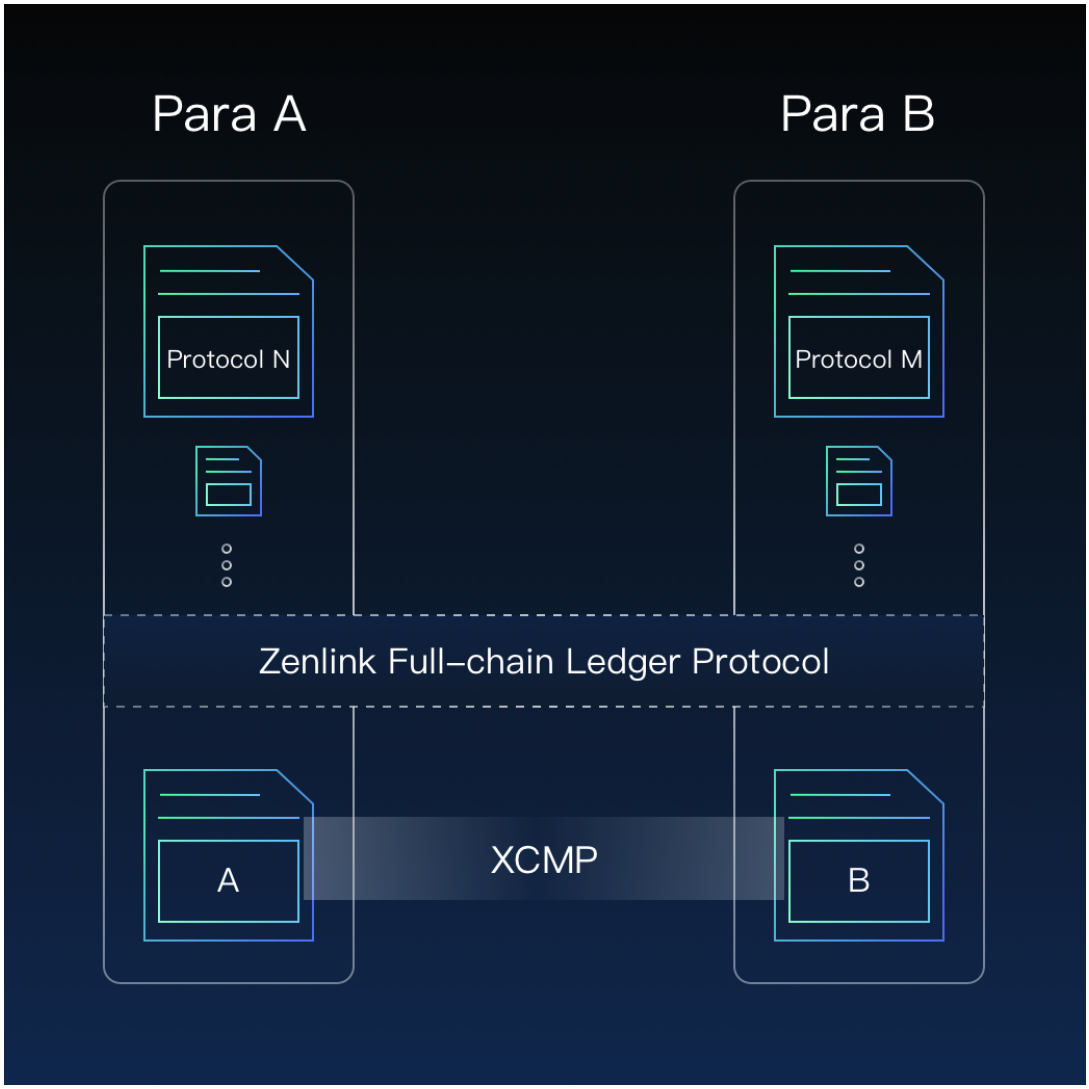
Technical Solution of Zenlink DEX Module

Parachains on Polkadot are essentially a collection of completely independent and freely programmable Runtime Modules. Compared with Ethereum's smart contract, parachains are more completely isolated, and the running calculations between parachains can be independently parallelized. This allows Polkadot's comprehensive TPS to increase by several levels, but it will also make the interaction between parachains more complicated.

The team led by Gavin is vigorously developing the communication protocol XCMP on the parachains, but even if the development of the XCMP protocol is completed, we still need to develop various application-related general protocols on the XCMP protocol. Zenlink DEX Protocol is to implement a protocol for parachains to interoperate and share the liquidity pool.

The cross-chain protocol on Polkadot can be compared to the traditional TCP/IP protocol. The XCMP protocol is similar to the data link layer on TCP/IP, which solves the transmission function of the indiscriminate perception protocol between each chain (route).

Zenlink DEX Protocol is similar to the application layer on TCP/IP. As long as each parachain implements the Zenlink DEX Protocol, it can share all the liquidity pool of parachains on Polkadot.



Zenlink DEX functions can be imported into parachains in the following three ways.

1. Integrated with parachains by the substrate module.
2. Deployed to the smart contract module on parachains by Wasm Contract.
3. Deployed to the smart contract module on parachains by EVM Contract.

Zenlink will specifically implement the above three ways to facilitate the parachains one-click integration of Zenlink DEX Module.

Zenlink DEX Aggregator

It can be seen that over a long period of time, decentralized exchange (DEX) and centralized exchange (CEX) will coexist and complement each other. And if we look at the Polkadot network, we will certainly see the existence of many different types of decentralized exchanges in the future. Zenlink DEX Aggregator is an abstract unified entry point to these different exchanges. It has the following functions:

- Compatible with a variety of interface protocols for decentralized exchanges.
- For existing trading pairs, the aggregator automatically matches prices across multiple exchanges, provides a trading path with the lowest slippage point, and eventually matches trades across multiple exchanges.
- For trading pairs that are not yet supported, the aggregator performs a path search across multiple exchanges to finalize the transaction.
- Provide a simple, unified application interface for end users. The users can make a one-click exchange within the application without having to worry about the logic behind it.

Tokenomics

Based on Zenlink technical architecture and ecosystem planning, the main application scenarios of the native token ZLK are as follows.

Liquidity mining

For users or pools who provide liquidity to the network, we would release the corresponding ZLK token to the liquidity providers in a non-linear function according to their amount of assets and duration of deposit. The larger the amount and the longer the duration of the liquidity, there will be additional encouragement, namely The concept of "coins per day" will be introduced.

The on-chain governance of the trading network

ZLK token will be deeply involved in the trading network constructed by the entire protocol, such as token listing, liquidity access, access to other DEX slots, protocol upgrades, etc.

Obtaining network revenue

The revenue generated by the network, such as trading fees, slash, etc., will be partially or fully returned to ZLK token holders, and the weight of the return is related to the holding time, and the concept of "coins per day" will also be introduced.

Zenlink DEX Network

Zenlink DEX Network is an abstract decentralized trading network ecology. It is directed by Zenlink DEX Protocol as the top-level Protocol, which is composed of Zenlink DEX Module on each parachain or other exchange applications to form the low-level trading nodes, and all trading nodes are linked by Zenlink DEZ Aggregator, so as to provide richer trading pairs and stronger liquidity. Zenlink Token (ZLK) is used to achieve the goal of orderly development and community governance.

With the development of the Polkadot network and Defi, Zenlink DEX Network will even introduce more types of products such as lending services, oracles, and financial derivatives to achieve the goal of evolution and growth.

Roadmap

Milestone 1

- 2020.11, Build Zenlink DEX Prototype on a substrate testnet chain.
- 2020.12, Deploy Zenlink DEX to Kusama and publish Zenlink DEX Dapp for test.

Milestone 2

- 2021.01, Build Zenlink Aggregator Prototype.
- 2021.02, Deploy Zenlink Aggregator to Kusama and integrated with Zenlink DEX.
- 2021.03, Full test on Kusama.

Milestone 3

- 2021.04, Deploy Zenlink DEX and Aggregator to Polkadot sync with the publish of parachains.

Longterm

- 2021 Q1, cooperate with more parachains, integrate Zenlink DEX Module, and provide more trading pairs and liquidity for Zenlink DEX Network.
- 2021 Q3, Zenlink DEX Aggregator access more Polkadot DEX applications.

- ...

Summary

What we envision in the future is such a scenario: more and more projects will be built on Polkadot. Parachains with various businesses will need to interact and communicate with each other. The huge liquidity of digital assets constitutes an important link in the entire digital world. Zenlink committed to becoming important support behind these bonds, allowing the value of the entire network to flow freely.

Version History

1. 2020.08.17, v0.1 created.
2. 2020.08.30, v0.2 updated.
3. 2020.09.13, v0.3 updated.
4. 2020.09.21, v0.4 updated.
5. 2020.11.03, v0.5 updated.
6. 2020.12.01, v0.6 updated.