# Zenlink Whitepaper v0.7

## 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 a 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 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, the 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 Relaychain.
**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.

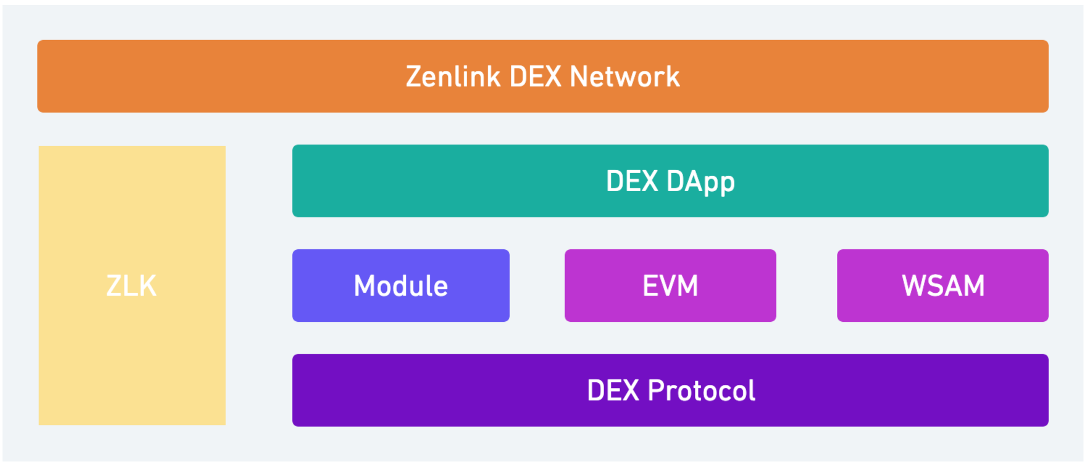
Based on the prediction of the further growth of the DEX ecosystem in the future and the rapid development of the public blockchain technology, we propose a Polkadot network-based, high-liquidity, upgradeable, cross-chain DEX network, Zenlink.

## Overall

Zenlink is committed to building a new generation of cross-chain DEX networks. 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 the long-term network development.

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

1. Zenlink DEX Protocol: The top-level unified general DEX protocol that includes the following three implementations:
    a. 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 get the DEX function, even share the liquidity with other DEX on other parachains.
    b. EVM Contract: it is a contractual deployment mode adopted in order to be compatible with the operation of the Ethereum EVM, and it is also a transition scheme adopted in the initial stage of the Polkadot network. The Zenlink DEX EVM version implements all the functions of the protocol layer, complements the perfect testing process, and can also be deployed to the parachain of EVM contracts at the first time, which greatly expands the applicability of the platform.
    c. WASM Contract: is the original contract implementation of Polkadot, and it is also the main contract deployment mode of Polkadot network in the future. The Zenlink DEX WASM version has the ability to deploy to WASM contract platform in the first place, and the applicability of the platform has been greatly improved.
2. Zenlink DEX DApp: A simple and user-friendly aggregator 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 instances on low slippage.
3. Zenlink Token: The native token of Zenlink DEX Protocol which can be used to distribute liquidity benefits, governance, etc



*Zenlink Ecosystem*

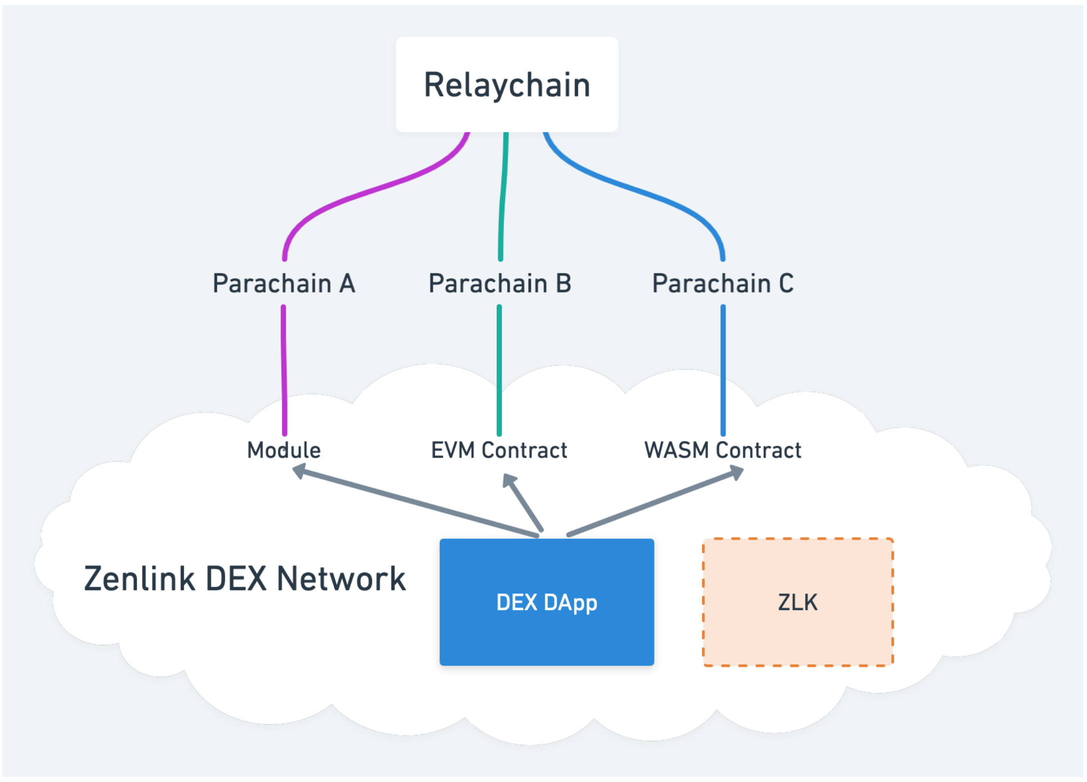
## The Whole Planning of Zenlink

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. Followed by Module, EVM Contract and WASM Contract will be implemented.

Besides, In order to complete the Zenlink ecosystem, we would like to build a front-end website application, Zenlink DEX Dapp. By the simple and smooth Zenlink DEX DApp that connects with most of all DEX on the Polkadot network, users will do one-click trade with multiple DEX on low slippage seamlessly.

Furthermore, In order to provide 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 different trading implementations, we set up the first exchange and 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.



*Zenlink Architechture*

In the medium to long term plan, we will work with other parachains projects to deploy the dex implementations to more parachains, and we will also connect Zenlink DEX DApp 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 modules 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 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.

### Architecture

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.

#### 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

Set `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. the 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.

### create_exchange

initializing trading pair.

```
pub fn create_pair(
    origin,
    token_0: AssetId,
    token_1: AssetId,
)-> DispatchResult
```

Parameters:

- origin: Trading account
- token_0: asset ID
- token_1: asset ID

Description:

- Token_0 and Token_1 represent the two assets that make up the trading pair (also known as the liquidity pool).
- (token_0, token_1) and (token_1, token_0) is the same trading pair.

### Add Liquidity

```
pub fn add_liquidity(
   origin,
   token_0: AssetId,
   token_1: AssetId,
   amount_0_desired : T::TokenBalance,
   amount_1_desired : T::TokenBalance,
   amount_0_min : T::TokenBalance,
   amount_1_min : T::TokenBalance,
   target_parachain: ParaId,
   deadline: T::BlockNumber,
)->DispatchResult
```

Parameters:

- origin: Trading account
- token_0: The asset ID that makes up the trading pair
- token_1: The asset ID that makes up the trading pair
- amount_0_desired: The token_0 amount you want to deposit to the liquidity pool.
- amount_1_desired: The token_1 amount you want to deposit to the liquidity pool.
- amount_0_min: The minimum token_0 amount you expect to deposit to the liquidity pool.
- amount_1_min: The minimum token_1 amount you expect to deposit to the liquidity pool.
- target_parachain: The parachain ID of the liquidity pool.
- deadline: The block deadline of this transaction.

Description:

Suppose in the following scenario:

- Alice has ABC(the native asset of Parachain200) and XYZ(the native asset of Parachain300).
- Alice wants to deposit liquidity to the ABC/XYZ liquidity pool.
- ABC is represented on Parachain300 as ABC'.
- The liquidity pool (trade pair ABC'/XYZ) is located on Parachain300.

Exception:

- If an add-liquidity transaction is made on Parachain200, please ensure enough Parachain300 assets(XYZ). Otherwise, the transaction must fail, and the Parachain200 assets(ABC) exist on Parachain300 in the form of ABC'.
- If an add-liquidity transaction is made on Parachain300, please ensure enough Parachain200 assets on Parachain300(ABC'). If you don't have ABC', even if you have enough ABC on Parachain200, it will be a failed transaction.

### Remove Liquidity

```
pub fn remove_liquidity(
   origin,
   token_0: AssetId,
   token_1: AssetId,
   liquidity: T::TokenBalance,
   amount_token_0_min : T::TokenBalance,
   amount_token_1_min : T::TokenBalance,
   to: <T::Lookup as StaticLookup>::Source,
   deadline: T::BlockNumber,
)->DispatchResult
```

Parameters:

- origin: Trading account.
- token_0: The asset ID that makes up the trading pair.
- token_1: The asset ID that makes up the trading pair.
- liquidity: The amount of liquidity you can hold up.
- amount_token_0_min: The minimum token_0 amount you expect to be obtained after liquidity extraction.
- amount_token_0_min: The minimum token_1 amount you expect to be obtained after liquidity extraction.
- to: The recipient account.
- deadline: The block deadline of this transaction.

### Swap token_0 with the exact amount for token_1

```
pub fn swap_exact_tokens_for_tokens(
   origin,
   amount_in: T::TokenBalance,
   amount_out_min: T::TokenBalance,
   path: Vec<AssetId>,
   to: <T::Lookup as StaticLookup>::Source,
   target_parachain: ParaId,
   deadline: T::BlockNumber,
)->DispatchResult
```

Parameters:

- origin: Trading account.
- amount_in: The exact token_0 amount you want to send.
- amount_out_min: The minimum token_1 amount you expect to get.
- path: The transaction path.
- to: Recipient address.
- target_parachain: The parachain id of the liquidity pool.
- deadline: The block deadline of this transaction.

Description:

- The path is represented as an array of asset ids.
- The first element represents the asset being sent, and the last element represents the target asset.
- [A, B]: Exchange A for B in the A/B liquidity pool.
- [B, A]: Exchange B for A in the A/B liquidity pool.
- [A, B, C]: Exchange A for B in the A/B liquidity pool, then exchange B for C in the B/C liquidity pool.

Additionally,

- There is no limit to the length of the path.
- The path is obtained from the API server. Since we currently have only one pool, the path length is fixed at 2.

#### Swap token_0 for token_1 with the exact amount

```
pub fn swap_tokens_for_exact_tokens(
    origin,
    amount_out: T::TokenBalance,
    amount_in_max: T::TokenBalance,
    path: Vec<AssetId>,
    to: <T::Lookup as StaticLookup>::Source,
    deadline: T::BlockNumber,
    target_parachain: ParaId,
)->DispatchResult
```

Parameters:

- origin: Trading account.
- amount_out: The exact token_1 amount you want to get.
- amount_in_min: The minimum token_0 amount you expect to send.
- path: The transaction path.
- to: Recipient address.
- target_parachain: The parachain id of the liquidity pool.

- deadline: The block deadline of this transaction.

Description:

- The path is represented as an array of asset ids.
- The first element represents the asset being sent, and the last element represents the target asset.
- [A, B]: Exchange A for B in the A/B liquidity pool.
- [B, A]: Exchange B for A in the A/B liquidity pool.
- [A, B, C]: Exchange A for B in the A/B liquidity pool, then exchange B for C in the B/C liquidity pool.

Additionally,

- There is no limit to the length of the path.
- The path is obtained from the API server. Since we currently have only one pool, the path length is fixed at 2.

### Transfer mapped assets in a parachain

```
pub fn transfer(
    origin,
    asset_id: AssetId,
    target: <T::Lookup as StaticLookup>::Source,
    amount: T::TokenBalance
)->DispatchResult
```

Parameters:

- origin: Trading account
- asset_id: Asset id
- target：Recipient address
- amount：Asset amount

Description:

- This interface can only transfer assets that are mapped by other parachains to an account in a chain
- such as transfer ABC' on Parachain300 to an account on Parachain300, rather than an account on Parach200.

### Transfer mapped/native assets to another parachain

```
pub fn transfer_to_parachain(
   origin,
   asset_id: AssetId,
   para_id: ParaId,
   account: T::AccountId,
   amount: T::TokenBalance
) -> DispatchResult
```

Parameters:

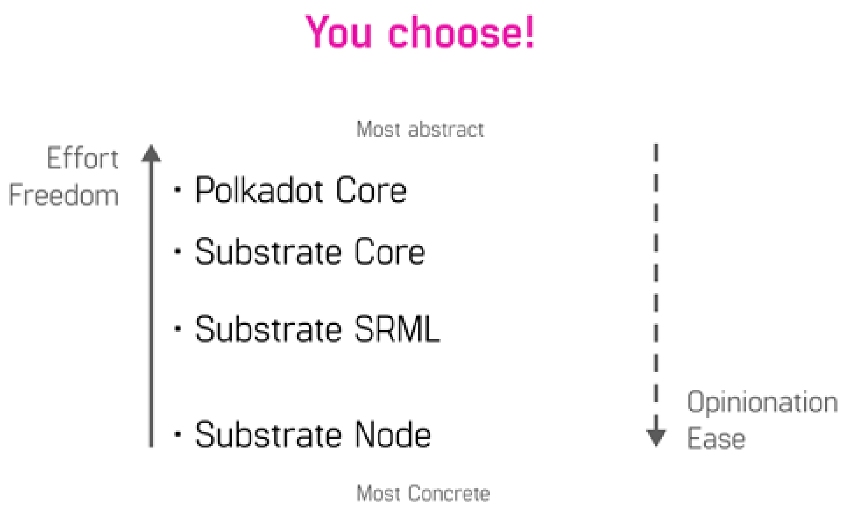
- origin: Trading account
- asset_id: Asset id
- para_id: Target chain id
- account：Recipient address
- amount：Asset amount

Description:

- This interface can transfer mapped/native assets to an account in another parachain.
- such as transfer ABC' or XYZ on Parachain300 to an account on Parachain200

## Zenlink DEX Module

Polkadot cleverly adopts a layered and clear architecture, allowing developers to develop their own blockchain quickly.



*Polkadot Architecture*

We implemented a general and stable Zenlink DEX Module on the Substrate Runtime Module Library (SRML) layer according to the Zenlink Protocol standard. Its characteristics are:

- **Plug, connect, and use it conveniently.** Parachains on the Polkadot network can quickly implement DEX on the parachains by introducing Zenlink DEX Module and can connect to a wider Zenlink DEX Network. While the tokens on the parachain inject liquidity into the Zenlink DEX Network, they also get a more free flow of values with other parachains and DOT.
- **Upgrade flexibly and freely.** Parachains only need to replace Zenlink DEX Module to upgrade to the latest version and experience more powerful functions.

## 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 are making it even more accessible to the public. In today’s Defi context, AMM is mostly a ”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 provide 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 someone sells $\Delta x$ tokens, he will get $\Delta y$ tokens such that:

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

So that, the $\Delta y$ 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 $\Delta x$ which is the amount of token X he wants to sell, and by automatically calculating the program within the module, we can provide the price p and $\Delta y$ 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 pairs 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 contracts without human intervention.

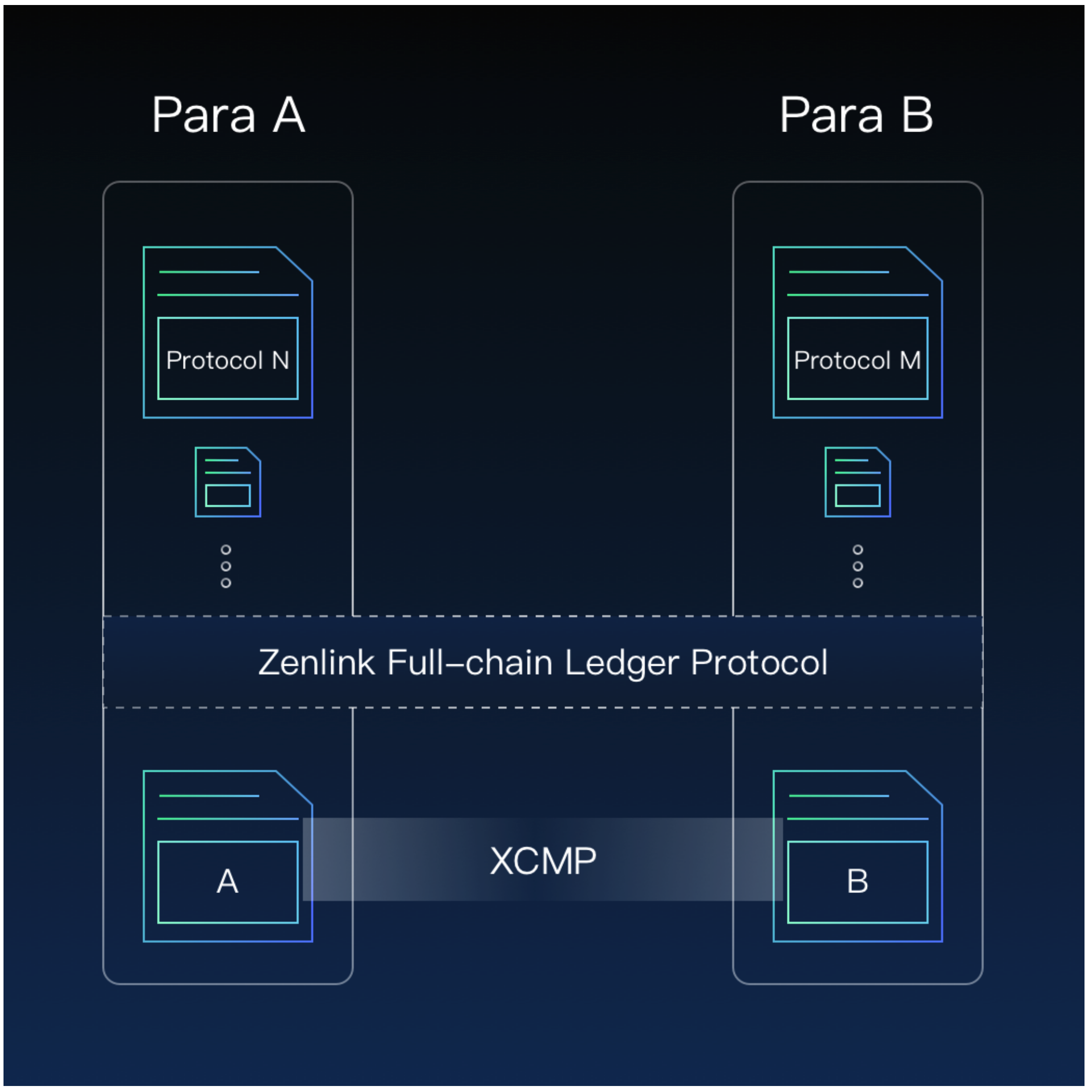
## Technical Solution of Zenlink DEX Module

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.

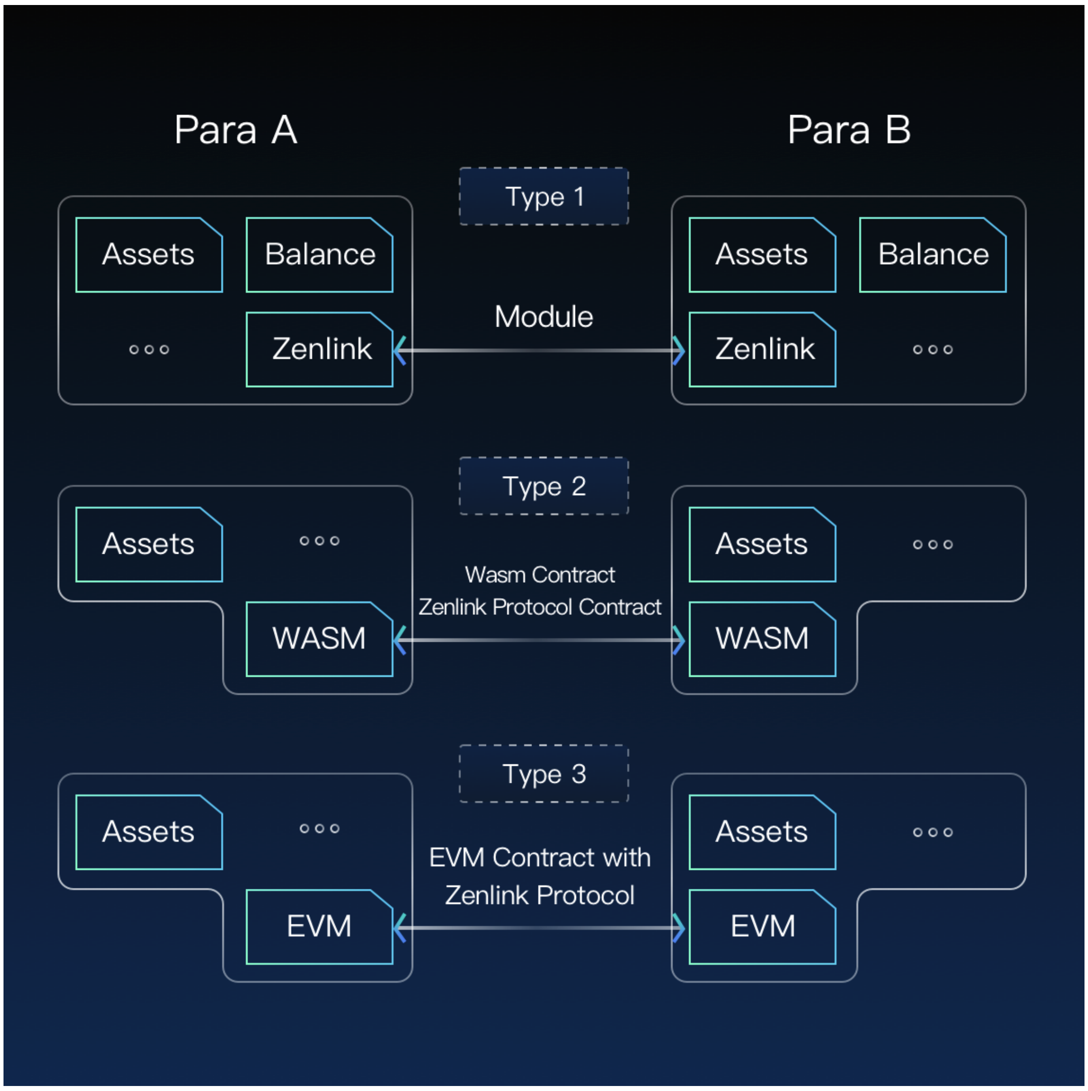
The cross-chain protocol on Polkadot can be compared to the traditional TCP/IP protocol. The XCMP protocol is similar to the data link layer on TCP/IP, which solves the transmission function of the indiscriminate perception protocol between each chain (route).

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 functionalities can be imported into parachains in the following three ways.

1. Integrated with parachains as a substrate module.
2. Deployed to the Wasm Contract module on parachains.
3. Deployed to the EVM Contract module on parachains.



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

## Zenlink DEX DApp

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 DApp is an abstract unified aggregator entry point to these different exchanges. It has the following functions:

- Compatible with a variety of interface protocols for decentralized exchanges.
- The aggregator automatically matches prices across multiple exchanges for existing trading pairs, 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 the 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 DApp, 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 Q4

- Build Zenlink DEX Prototype on a substrate testnet chain.
- Build Zenlink DEX Dapp for test.

### Milestone 2: 2021 Q1

- Implement Zenlink DEX Smart Route functionality.
- Launch the 1st Zenlink DEX DApp public beta test.
- Implement WASM Contract.

### Milestone 3: 2021 Q2

- Implement EVM Contract.
- Launch the 2nd Zenlink DEX DApp public beta test.
- Deploy Zenlink DEX Protocol to some Kusama parachains.
- Launch Zenlink DEX DApp on Kusama.
- Issue Zenlink Token.

### Longterm

- 2021 Q3, cooperate with more parachains, integrate Zenlink DEX Protocol, and provide more trading pairs and liquidity for Zenlink DEX Network.
- 2021 Q3, Deploy Zenlink DEX Protocol to some Polkadot parachains and launch Zenlink DEX DApp on Polkadot.
- 2021 Q4, 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.
7. 2021.06.15, v0.7 updated.