Dfyn V2

Draft V1.0

March 2023

Abstract

The most powerful lever to catalyze the adoption of any innovation has always been its ability to provide an unrivaled financial incentive. The case study of blockchain technology is no different. While early adopters loved the novelty of Bitcoin's whitepaper, the general public wasn't exactly lining up to get a piece of the action. Most of them stayed on the fence until Web 3.0 started offering decentralized financial tools that freed every user from their dependency on archaic, centrally-controlled entities like banks. The Decentralized Finance (DeFi) boom truly laid the foundation of what later became the hotbed of innovation and simultaneous global adoption. While multiple innovative solutions were responsible for ushering in a swarm of new users, one of the most dominant of them was the Decentralized Exchange, or DEX for short. While Bancor was the founding father of the space by innovating the Automated Market Maker (AMM) model to facilitate swaps between Ethereum-based tokens, Uniswap, with its constant product market maker model, emerged as a pioneer in this space. Over the years, as the DeFi space matured, so did the offerings. The one thing that is common across any successful project is that they keep iterating and perfecting their offerings by adapting to market needs. As the first DEX on Polygon, the team at Dfyn also started engaging in a similar process of identifying weak links in the existing DeFi ecosystem and building an improved product for the masses. After months of developing the solution in stealth mode, we are proud to bring Dfyn V2. Unlike the current crop of DEXs, most of which have a monolithic architecture, Dfyn V2 is powered by four core modules and over thirteen contracts, all interacting with each other to provide a decentralized trading platform for all kinds of users. In this whitepaper, we will take a deeper look into Dfyn V2's architecture and working, which includes a concentrated liquidity market maker, a first-of-its-kind on-chain limit order system, and a high-frequency RFQ engine, among other innovations.

Contents

1 Introduction **4**
- 1.1 Background . . . 4
- 1.2 A Brief History of Dfyn V1 . . . 4
  - 1.2.1 Dfyn V1 Features . . . 5
  - 1.2.2 Dfyn V1 Challenges . . . 5
- 1.3 Intro to Dfyn V2 . . . 5
- 1.4 Organization . . . 5

2 What is Dfyn V2? **5**

3 Dfyn V2 Core Modules **6**
- 3.1 AMM with Provision for Concentrated Liquidity . . . 6
  - 3.1.1 AMM Curve . . . 6
  - 3.1.2 Dfyn V2 Ticks . . . 8
  - 3.1.3 AMM Mathematics . . . 8
- 3.2 hRFQ Engine . . . 11
- 3.3 Dfyn Signal . . . 11
- 3.4 Limit Order Engine . . . 12
  - 3.4.1 What are On-chain Limit Orders? . . . 12
  - 3.4.2 Limit Order Ticks in Dfyn V2 . . . 12
  - 3.4.3 Limit Order Fee Model . . . 12

4 Architectural Components **13**
- 4.1 Smart Contracts . . . 13
  - 4.1.1 Core Contracts . . . 13
  - 4.1.2 Periphery Contracts . . . 14
  - 4.1.3 hRFQ Contracts . . . 14
- 4.2 APIs & SDKs . . . 14
  - 4.2.1 hRFQ API . . . 14
  - 4.2.2 Signal API . . . 14
  - 4.2.3 Signal SDK . . . 15
- 4.3 Trade Engine . . . 15

5 Dfyn V2 Features **15**
- 5.1 First-of-its-kind On-chain Limit Order System . . . 15
  - 5.1.1 Centralized Limit Order Book Exchanges (CLOBs) . . . 15
  - 5.1.2 Decentralized Limit Order Book Exchanges (DLOBs) . . . 15
  - 5.1.3 Dfyn's Concentrated Limit Market Maker (CLMM) . . . 16
- 5.2 Delegation Vaults . . . 16
- 5.3 Dynamic Fee Model . . . 16
- 5.4 Cross-chain Capabilities . . . 16

Glossary

basis point A unit of measure of percentages in traditional financial markets. Corresponds to 0.01% (1/100th of 1%).

DEX A DEX allows users to trade their digital currency in a non-custodial, peer-to-peer ecosystem where transactions happen directly between crypto traders.

flash loan A flash loan, also termed as a **one block borrow**, is a loan that allows an entity to borrow a certain amount of tokens, given that they pay it back before the end of the same transaction.

layer 1 Base layer blockchain architecture, eg. Bitcoin, Ethereum. In recent years, multiple layer 1 alternatives to Ethereum have come to light. Most of these blockchains have made certain changes to Ethereum's existing infrastructure, like introducing sharding or implementing a different consensus mechanism.

layer 2 Blockchain networks implemented as smart contracts on top of another blockchain (layer 1). Layer 2 networks do not make any changes to the underlying blockchain architecture. For example, layer 2 networks on Ethereum take advantage of Ethereum's decentralized security model while negating its scalability constraints by adding another layer of transactions on top of it. There are mainly three ways in which layer 2 scaling solutions are exercised - State Channels, Plasma, and Rollups. Popular examples of layer 2 blockchains include Polygon, Optimism, and Arbitrum.

limit order A trade wherein the user specifies constraints on the state of the market at which they want to execute the order.

rebase A mechanism that algorithmically adjusts the supply of tokens to maintain a constant level.

Web 3.0 Third generation of internet services built on top of decentralized data networks with the aim to make the internet more open and trustless.

Acronyms

AMM Automated Market Maker.

CEX Centralized Exchange.

CPMM Constant Product Market Maker.

DApp Decentralized Application.

DeFi Decentralized Finance.

DEX Decentralized Exchange.

EVM Ethereum Virtual Machine.

hRFQ high-frequency Request for Quote.

NFT Non Fungible Token.

RFQ Request for Quote.

TVL Total Value Locked.

1 Introduction

1.1 Background

The history of DeFi is one of innovation, growth, and potential. Its journey from being considered a niche concept within the community of early adopters to becoming an independent, thriving ecosystem that is actively disrupting traditional financial systems is one to behold.

DeFi can trace its origins back to the launch of Ethereum in 2015. Ethereum introduced Ethereum Virtual Machine (EVM), a novel concept that allowed developers to write and deploy self-executing computer programs (called smart contracts) [1, 2]. The onset of Ethereum changed how people think about digital money, global payments, and applications - smart contracts allowed for the creation of Decentralized Applications (DApps), which could be deployed on the Ethereum blockchain and operate without involving a central authority.

One of the earliest and most successful DApps built on Ethereum was a DEX. This tool allowed users to trade cryptocurrency assets directly with each other without the need for a centralized entity. The reason why DEXs initiated mass adoptions is that they disrupted the monopoly of existing centralized solutions. They stood against the custodial, non-transparent, and self-serving model of CEXs and provided a fully functional peer-to-peer offering where traders could swap/buy/sell their assets directly with each other. Therefore, when we analyze the incidence point of the first bull run, it becomes abundantly clear that DEXs played a pivotal role in the DeFi summer.

However, this progression was not without its fair share of flaws. With vast amounts of capital flowing into the ecosystem, the prominence of every Web 3.0 project was starting to get determined by the lowest common denominator: TVL, i.e., total value locked. DEXs with a high TVL were assumed to be well-established and dependable. An optic like that automatically attracted droves of traders and drove up their trading volumes. To attain the same level of prominence, instead of building a technical architecture that focused on maximizing the utility of existing liquidity, every DEX's primary focus shifted to acquiring liquidity. Since the only way to attract liquidity was to generate value for the liquidity providers, projects started shilling out rewards higher than the value generated from trades [3]. This paradigm created a quick race to the bottom for all stakeholders.

In the midst of this chaos, the principle of concentrated liquidity came to the fore. In a concentrated liquidity model, the liquidity is not distributed across a broad price range but across shorter price ranges. In doing so, liquidity providers construct individualized price curves that reflect their preferences. This innovation in AMMs has allowed DEXs to be more capital efficient and fulfill more orders with less liquidity.

With Dfyn V2, we have taken this foundational innovation as our inspiration and built a sophisticated solution for all classes of crypto traders in the space. Before we dive deep into the architecture we've built, let's walk down the memory lane of Dfyn V1 to understand the context behind its revamp.

1.2 A Brief History of Dfyn V1

With the foresight that up-and-coming layer 1s and layer 2s will play a pivotal role in DeFi, we launched Dfyn on Polygon in August 2020 with a vision to achieve an inclusive multi-chain DeFi ecosystem with a feature-rich, easy-to-use DEX at the heart of it.

Dfyn V1 is an exchange that implements the vanilla Constant Product Market Maker (CPMM) algorithm:

$$xy = k \tag{1}$$

As the name suggests, CPMM is an AMM algorithm that maintains a constant product of the pool of the assets it holds. This means that the total value of the assets in the pool remains constant, regardless of changes in the value of individual assets. Because they maintain a constant product, these AMMs can provide predictable returns for liquidity providers.

1.2.1 Dfyn V1 Features

As the first DEX on Polygon, Dfyn introduced and integrated several new features, including but not limited to the following:

- vDFYN vaults for the community to get fee-sharing from the protocol
- Build-your-own-farms (BYOF), a first-of-its-kind no-code farming product on the Polygon network
- Router Protocol's widget to offer cross-chain swaps within Dfyn
- Dfyn Fusion, a product designed to streamline the token conversion process for users by enabling them to convert multiple tokens to a single token in just one click

1.2.2 Dfyn V1 Challenges

One of the first things we did while going back to the drawing board for Dfyn V2 was to perform a comprehensive internal analysis and external benchmarking to identify features we needed to improve upon. We found that the conventional AMM model adopted by Dfyn was causing multiple challenges to scale operations economically with a seamless user experience. These challenges included:

1. **Expensive Liquidity Mining Campaigns:** We are all well aware that a high TVL guarantees less slippage in large orders. However, users will always park their liquidity where they will get more rewards. Hence, we had to shell out more dividends to attract large amounts of liquidity.
2. **Competitive Landscape:** In Dfyn V1, liquidity in any pool is distributed without bounds, allowing trades at any price from 0 to infinity. This approach can lead to inefficiencies in the AMM. The prices of an asset tend to fluctuate within a specific range, whether narrow or wide. Therefore, providing liquidity in a price range that is far from the current price or is unlikely to be reached does not make good use of capital and reduces the attractiveness of the Dfyn V1 as a DEX. As the market has matured, more DEXs with a concentrated liquidity model have come into the mix, making it increasingly difficult for Dfyn to offer competitive prices.

1.3 Intro to Dfyn V2

Constant innovation is the key to staying competitive in the market. Towards this end, we started building Dfyn V2 with the aim of enhancing our existing functionalities and introducing new cutting-edge solutions. Dfyn V2 has a significant focus on concentrated liquidity pools to benefit traders and liquidity providers alike. Besides the concentrated liquidity model, Dfyn V2 ships with a first-of-its-kind on-chain limit order system, a high-frequency RFQ engine, and a smart order router out of the box. These offerings are best-in-class features and either meet or beat any comparable DEX bar none.

1.4 Organization

In the sections that follow, we will first get a brief understanding of Dfyn V2, followed by a deep dive into Dfyn V2's core modules including its improved AMM model, hRFQ engine, limit order engine, and smart order router. Post that, we'll take a closer look at Dfyn V2's architectural components. In drawing the paper to a close, we will briefly discuss Dfyn V2's exclusive features.

2 What is Dfyn V2?

Dfyn is a fully decentralized multichain protocol that relies on a constant product market-maker algorithm instead of centralized order books to enable token swaps. Anyone can use the exchange to swap tokens or to earn fees by supplying liquidity to various liquidity pools. In comparison to

the previous version, Dfyn V2 is powered by a more capital-efficient AMM, which breeds a win-win situation for all the parties involved:

(a) the traders can complete their trades with minimal slippage,

(b) the liquidity providers can earn higher fees on the same amount of liquidity and,

(c) the AMM can support the increased trade volumes with a far lesser capital requirement.

Besides the upgraded AMM protocol, Dfyn V2 will ship with a host of new features (discussed in section 5).

3 Dfyn V2 Core Modules

3.1 AMM with Provision for Concentrated Liquidity

The concentrated liquidity pool model is one of the most powerful innovations in the domain of DEXs. In a conventional AMM model, all users provide liquidity in a $(0, \infty)$ price range. However, in a concentrated liquidity enabled AMM DEX, all users can pick the range in which they want to provide liquidity. This allows users to narrow down the liquidity provision range, which amplifies their liquidity within that range - meaning traders experience lesser price impact when the tokens are trading in that range, and liquidity providers accrue more fees. In other words, this model improves the capital efficiency of the system by allowing for more liquidity in a narrow price range. This also affords liquidity providers greater flexibility; they can configure their liquidity's price range based on the volatility of the liquidity pool to which they are contributing.

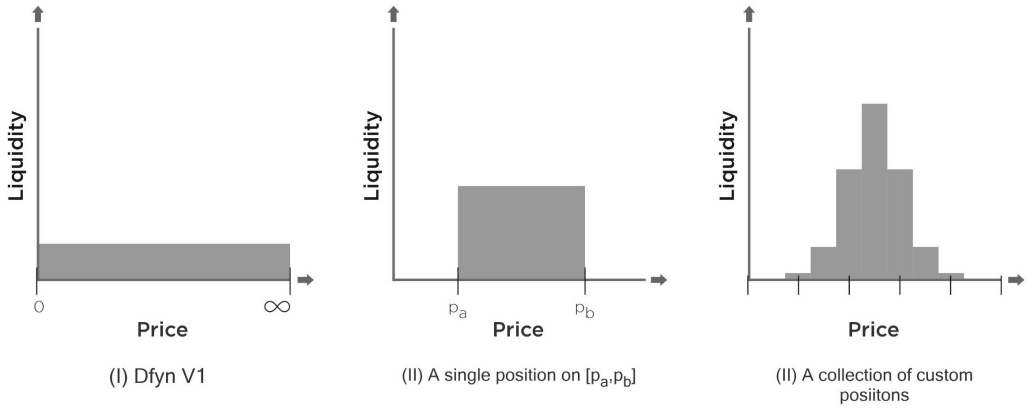

Figure 1: Dfyn V2 liquidity spread across various price ranges

An easier way to understand this model would be by thinking of a Dfyn V2 pair as a combination of multiple small Dfyn V1 pairs. In other words, the main difference between Dfyn V1 and V2 is that, in V2, the entire price range from 0 to infinity is split into shorter price ranges, and each of these shorter price ranges has finite reserves. However, what's crucial to understand is that within that shorter price range, Dfyn V2 works exactly like Dfyn V1.

3.1.1 AMM Curve

Let's try to visualize the Dfyn V2 curve - since we don't want the curve to be infinite, we cut it at points P_a and P_b and say that these are the boundaries of the curve. Moreover, we shift the curve so that its boundaries lay on the axes. The resulting curve is given in figure 2. Buying or selling tokens moves the price along this curve.

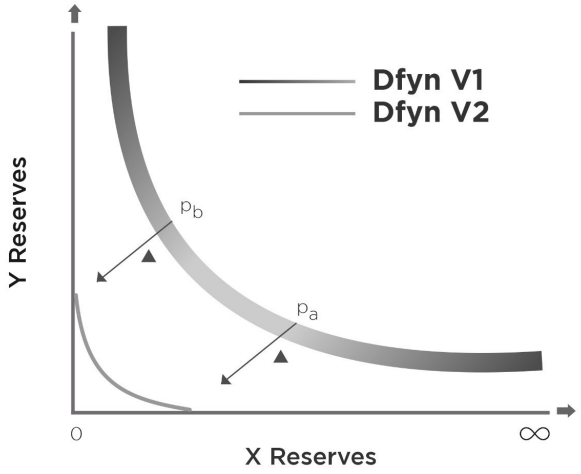

Figure 2: Liquidity in the USDC/ETH pool in Uniswap V3

Let's assume that the start price is at the middle of the curve. To get to point P_a, we need to buy all the available Y and maximize X in the range; to get to point P_b, we need to buy all the available X and maximize Y in the range. When the price moves to either of these points, the pool becomes depleted, i.e., one of the token reserves will be exhausted, and trading against this pair in this price range won't be possible. So what will happen when the liquidity pool at the current price range gets depleted during a trade? The liquidity pool will slip into the following price range. If the subsequent price range doesn't exist, the trade ends up being fulfilled partially.

As a case study, consider the USDC/ETH pool in Uniswap V3 -

Figure 3: Liquidity in the USDC/ETH pool in Uniswap V3

The screenshot in figure 3 shows the breakdown of the USDC/ETH liquidity pair in Uniswap V3 across different price ranges. You can see a lot of liquidity around the current price range, but the further we go from there, the less liquidity there is. This is because liquidity providers strive to have higher efficiency of their capital. Also, the whole range is not infinite; its upper boundary is shown in the image.

3.1.2 Dfyn V2 Ticks

3.1.2.1 What is a Tick?

In financial markets, a tick refers to the slightest possible change in the price of a financial instrument. For instance, if a particular asset has a tick size of $0.01, its price can only be altered in increments of $0.01 and cannot fluctuate by an amount smaller than that. The tick size for different securities may vary depending on various factors, such as the liquidity of the security and the minimum price movement that the market can accommodate.

3.1.2.2 Tick Representation in Dfyn V2

Dfyn V2 allows for more precise trading by dividing the price range $[0, \infty]$ into granular ticks, similar to an order book exchange.

- The system defines the price range corresponding to each tick rather than relying on user input.
- Trades within a tick still follow the pricing function of the CPMM, but the equation is updated every time the price crosses a tick's price range.
- Orders can be executed at any price within the tick's price range. In the case of limit orders, we have **limit order ticks** (discussed in section 3.4.2).
- For every new position, we insert two elements into the linked list based on the range's start and end prices. To keep the list manageable, we only allow the prices to be represented as powers of 1.0001. For example, our range could start at tick(0) with a price of $1.0001^0 = 1.0000$ and end at tick(23028), which corresponds to a price of approximately $1.0001^{23028} = 10.0010$. We can use this approach to cover the entire $(0, \infty)$ price range using only 24-bit integer values.

To represent the linked list, we create a mapping of 24-bit integers to `Tick` structures, where each tick holds pointers to the previous tick and the next tick, the amount of liquidity within the tick's price range, as well as other variables to track fees and time spent within a price range.

```
struct Tick {
        int24 nextTickToCross; // based on the boolean zeroForOne
        uint160 secondsGrowthGlobal; // used to calculate yield
        uint256 currentLiquidity; // liquidity is stored against the upper tick
        uint256 feeGrowthGlobalA; // fee counter
        uint256 feeGrowthGlobalB;
        bool zeroForOne; // token 0 to token 1 or vice versa
        uint24 tickSpacing;
    }
```

3.1.3 AMM Mathematics

Dfyn V2 uses the same mathematical formulas as Dfyn V1, but they have been enhanced and augmented to provide better results.

3.1.3.1 Liquidity Math

To facilitate transitioning between price ranges, streamline liquidity management, and avoid rounding errors, Dfyn V2 represents L (a measure of liquidity in a pool) as the square root of the product of x and y.

$$L = \sqrt{xy} \tag{2}$$

Here x and y represent the token reserves in a pool. Basically, L is the geometric mean of the token reserves in a pool. Using the CPMM equation (given in equation 1), we can derive:

$$\begin{aligned} L &= \sqrt{k} \\ \implies L^2 &= k \end{aligned} \tag{3}$$

Table 1: Notations Used

Notation	Meaning
k	Amount of liquidity in a pool
L	Square root of the amount of liquidity in a pool
x	Token 0 reserves in a pool
y	Token 1 reserves in a pool
P_{xy}	Price of token 0 in terms of token 1
P_{yx}	Price of token 1 in terms of token 0
i	Tick index

3.1.3.2 Price Math

If P_{xy} denotes the price of token 0 in terms of token 1, then:

$$P_{xy} = \frac{y}{x} \tag{4}$$

Since token prices in a pool are reciprocals of each other, we can use only one of them in calculations (and by convention, Dfyn V2 uses P_{xy}). The price of token 1 in terms of token 0 can be calculated as follows:

$$P_{yx} = \frac{1}{P_{xy}} = \frac{1}{y/x} = \frac{x}{y} \tag{5}$$

Let's understand the price math with the help of an example. Assume that we have 2 ETH (token 0) and 3000 USDC (token 1) in an ETH/USDC pool. The price of ETH in terms of USDC can be calculated as $3000/2 = 1500$. Even though the price calculation is done via these formulas, we use square root denotation for the price throughout our contracts as we have done in the case of liquidity mathematics. Taking the square root of both the sides in equation 4, we get:

$$\sqrt{P_{xy}} = \sqrt{\frac{y}{x}} \tag{6}$$

Why do we use square root denotation?

1. Calculating square roots in a contract is not precise and can cause rounding errors. Thus, it's easier to store the square root values as they are and calculate squared powers instead of square roots (we do not store x and y in the contracts; we only store L and $\sqrt{P_{xy}}$).
2. Square root numbers are significantly smaller, allowing us to accommodate larger values for all variables.

3.1.3.3 Relationship between Price and Liquidity

The geometric mean of the token reserves in a pool, L, is directly proportional to the change in the price of token 1 in terms of token 0, P_{yx} multiplied by the change in token 1 reserves.

$$L = \Delta\sqrt{P_{yx}} * \Delta y = \Delta\sqrt{P_{xy}} * \Delta x \tag{7}$$

To prove this, we'll use equations 2 and 5 in the aforementioned equation:

$$\sqrt{xy} = \Delta \frac{1}{\sqrt{P_{xy}}} * \Delta y \tag{8}$$

Expanding the Δ, we get:

$$\sqrt{xy} = \frac{1}{\sqrt{P_{xy,1}} - \sqrt{P_{xy,0}}} * (y_1 - y_0) \implies \sqrt{xy} * \left(\sqrt{P_{xy,1}} - \sqrt{P_{xy,0}}\right) = y_1 - y_0 \tag{9}$$

Using equation 6 in equation 9, we get:

$$\sqrt{xy} * \left(\sqrt{\frac{y_1}{x_1}} - \sqrt{\frac{y_0}{x_0}}\right) = y_1 - y_0 \tag{10}$$

We know that:

$$L = \sqrt{xy} = \sqrt{x_1 y_1} = \sqrt{x_0 y_0} \tag{11}$$

Using equation 11 in equation 10, we get:

$$\begin{aligned} \left(\sqrt{x_1 y_1}\sqrt{\frac{y_1}{x_1}} - \sqrt{x_0 y_0}\sqrt{\frac{y_0}{x_0}}\right) = y_1 - y_0 &\implies \left(\sqrt{y_1}\right)^2 - \left(\sqrt{y_0}\right)^2 = y_1 - y_0 \\ &\implies y_1 - y_0 = y_1 - y_0 \end{aligned} \tag{12}$$

Since the LHS = RHS, we can confidently say that the equation 7 holds true. Using this equation, we can directly calculate the output amount without calculating the actual prices. We can rearrange the equation 7 to get the following formula:

$$\Delta y = \frac{L}{\Delta \sqrt{P_{xy}}} = L \Delta \sqrt{P_{yx}} \tag{13}$$

Similarly, Δx can be calculated as:

$$\Delta x = L \Delta \sqrt{P_{xy}} \tag{14}$$

By storing L and $\sqrt{P_{xy}}$, we eliminate the need to store and update the pool reserves (x and y). Also, we don't need to calculate $\sqrt{P_{xy}}$ every time the pool reserves change because we can always find $\Delta\sqrt{P_{xy}}$ and its reciprocal, $\sqrt{P_{yx}}$.

3.1.3.4 Tick Math

We have already established in section 3.1 that the infinite price range of Dfyn V1 is split into shorter price ranges in Dfyn V2. Furthermore, as discussed in section 3.1.2.2, Dfyn V2 uses ticks to track the coordinates of the upper and lower boundaries of these price ranges. In other words, the entire price range in Dfyn V2 liquidity pools is demarcated by evenly distributed discrete ticks. Each tick has an index i and corresponds to a certain price:

$$p(i) = 1.0001^i \tag{15}$$

Taking powers of 1.0001 has a desirable property - the difference between two adjacent ticks is 0.01% or 1 basis point. Since we only store $\sqrt{P}$ in our contracts (instead of P), the formula given in equation 15 becomes:

$$\sqrt{p(i)} = (\sqrt{1.0001})^i = (1.0001)^{\frac{1}{2}i} \tag{16}$$

As we iterate over different indices, we get the following values:

$$\begin{aligned} \sqrt{p(0)} &= (1.0001)^0 = 1 \\ \sqrt{p(1)} &= (1.0001)^{\frac{1}{2}} \approx 1.00005 \\ \sqrt{p(-1)} &= (1.0001)^{-\frac{1}{2}} \approx 0.99995 \end{aligned} \tag{17}$$

Tick indices are integers that can be both positive and negative. Dfyn V2 stores $\sqrt{P}$ as a fixed point Q64.96 number - a rational number that uses 64 bits for the integer part and 96 bits for the fractional part. Thus, the range of $\sqrt{p(i)}$ values lies within:

$$\sqrt{p(i)} \in \left[(2)^{-64}, 2^{64}\right] \tag{18}$$

Using equation 16 in this equation, we get:

$$\begin{aligned} (1.0001)^{\frac{1}{2}i} &\in \left[(2)^{-64}, 2^{64}\right] \\ \implies (2)^{-64} &\leq (1.0001)^{\frac{1}{2}i} \leq 2^{64} \end{aligned} \tag{19}$$

To solve for i, we'll have to use the logarithmic function. After taking log_2, equation 19 becomes:

$$-64 \leq \log_2(1.0001)^{\frac{1}{2}i} \leq 64 \tag{20}$$

Using the log power rule, we get:

$$\begin{aligned} -64 \leq \frac{i}{2}\log_2(1.0001) \leq 64 \implies -128 \leq i\log_2(1.0001) \leq 128 \\ \implies -128 \leq 0.00014426 * i \leq 128 \end{aligned} \tag{21}$$

Since i can only take integer values:

$$\begin{aligned} -88272 \leq i \leq 88272 \\ \implies i \in [-88272, 88272] \end{aligned} \tag{22}$$

3.2 hRFQ Engine

One key concept in financial exchanges is the prevalence of professional market makers in making sure everything works as it should. Mutual coincidence of wants between buyer and seller is effectively bridged by market makers and they are appropriately rewarded for their efforts in a well-functioning exchange ecosystem.

To this end, Dfyn's hRFQ module is a gas-efficient model that bypasses all the complexities of an AMM and connects traders directly to a set of professional market makers. Once users submit their request for a trade, the hRFQ engine requests quotes from multiple market makers and displays the best quote to the user. The user then has the option to accept or reject the trade. This module is complementary to our AMM module in the sense that when using Dfyn with the hRFQ functionality enabled, the user's experience is unaffected.

Dfyn's hRFQ module is highly capital efficient as it does not require any idle capital from the market makers. In fact, due to the low fees involved, this mechanism incentivizes market makers to compete and provide the best prices to the user requesting quotes. And since the verification of market makers' signatures and execution of orders happens on-chain, this model ensures that the decentralized aspects of security and anonymity of a DEX are maintained while getting closer to the prices provided on centralized exchanges.

3.3 Dfyn Signal

Signal is a multi-chain compatible smart trade-finding engine capable of finding the most optimal trade for the user by comparing quotes across different token paths across various protocols. The steps involved in Signal's workflow are as follows:

Step 1) Fetch quotes from Dfyn's hRFQ engine as well as quotes from Dfyn V1 & V2 AMMs (across different pools);

Step 2) Compute the best split, i.e., the split that guarantees the best price and minimum slippage;

Step 3) Show it to the user on the UI (if called via the Dfyn UI) or abstract it in the form of transaction call data (if called via the Signal API/SDK).

By default, all the trades on Dfyn's UI go through Dfyn Signal.

3.4 Limit Order Engine

One of the most significant modules in Dfyn V2 is its decentralized limit order framework that will allow users to place on-chain limit order that can be executed without any external triggers.

3.4.1 What are On-chain Limit Orders?

A trade order placed on a blockchain that specifies the maximum or minimum price at which a user is willing to buy or sell a particular asset is termed an on-chain limit order. The major element that differentiates Dfyn's on-chain limit order framework from its other off-chain counterparts is that it allows users to trade against each other in a trustless and transparent manner. In other words, Dfyn's on-chain limit order engine does not require any centralized intermediary to match buyers and sellers. A detailed comparison of Dfyn's limit order engine with existing limit order frameworks is given in section 5.1.

3.4.2 Limit Order Ticks in Dfyn V2

Conventional tick logic in Uniswap V3 limits the ability to provide liquidity on a single tick [4]. However, in Dfyn V2, we have introduced limit order ticks that allow for highly concentrated liquidity on a single tick resulting in enhanced price precision. This enables users to add limit order liquidity at a precise price point, similar to how order book exchanges operate. Let's understand this with the help of an example. Assume that Alice wants to set a limit order against the ETH-USDC pool wherein she wants to sell 1 ETH when the price of ETH in terms of USDC reaches 2000 (assume the current price of ETH to be 1500 USDC). To do so, in a conventional concentrated liquidity model, Alice would have had to provide liquidity in a range, however small. However, in Dfyn V2, Alice does not need to provide liquidity between any price range but can instead put her entire ETH liquidity on the tick corresponding to the nearest price point (in this case, tick index $i = 76013$ will correspond to a price of 2000.035). Since the limit order liquidity in Dfyn V2 acts as an independent entity, another benefit of placing this limit order via Dfyn is that it won't be rolled back after it is executed, i.e., Alice's ETH will be permanently converted to USDC when it reaches the desired target. In existing models, however, Alice's liquidity would have been restored back into a mix of USDC and ETH as soon as the price of ETH came below 2000 USDC.

3.4.3 Limit Order Fee Model

As mentioned above, Dfyn allows users to add limit order liquidity on a precise price point instead of a range. Since this liquidity is treated differently from the AMM liquidity, it follows a different fee model. Whenever a trade is executed on Dfyn V2, it uses the concentrated liquidity within two ticks. However, as soon as the liquidity within the two ticks is exhausted, it will utilize the limit order liquidity deposited on the terminal tick before going into the following range.

For the part of the trade that is fulfilled via the AMM liquidity, the trader has to pay the standard AMM fee associated with that pool, the majority of which goes to the liquidity providers. For the remaining part of the trade, i.e., the part fulfilled via the limit order liquidity, the trader does not have to pay the AMM fee but has to pay a separate limit order fee which goes to the protocol. This fee will be minimal in comparison to the AMM fee and can be configured independently for each liquidity pool. Since the protocol has a direct source of income from the limit order liquidity, it can have lax fee requirements on the limit order creators. This model benefits the whole ecosystem in the following way:

(a) the traders need to pay a lower fee to get their trades executed;

(b) limit order creators can get their orders fulfilled at zero or low cost (depending on the fee for placing a limit order against a particular pool);

(c) the protocol has an additional revenue stream.

4 Architectural Components

Now that we have an in-depth understanding of what Dfyn V2 offers, let us dive into the Dfyn V2 architecture.

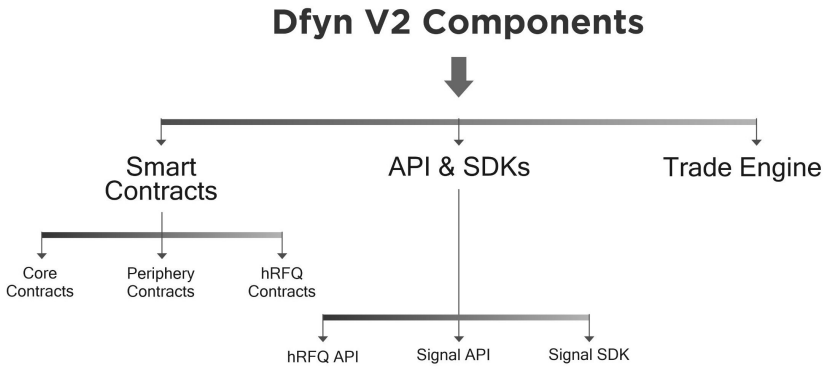

Figure 4: Dfyn V2 Architectural Components

4.1 Smart Contracts

Like any other decentralized application, smart contracts form the backbone of the Dfyn platform. Depending on their functionalities, we have classified the Dfyn V2 smart contracts into three categories - (1) Core contracts, (2) Periphery contracts, and (3) hRFQ contracts.

4.1.1 Core Contracts

Dfyn's core contracts are responsible for handling the various functions of the platform, such as facilitating transactions, managing liquidity, and enforcing the platform's rules. The three core contracts that act as the central point of control for the Dfyn platform are as follows:

1. **Liquidity Pool:** The `ConcentratedLiquidityPool` contract encapsulates the implementation of a liquidity pool in the Dfyn V2 protocol. It includes functions to mint (create/alter a mapping of the user's position on the contract), burn (remove/alter a user's position on the contract), and swap tokens within the pool while tracking the coordinates of upper and lower price boundaries using ticks. The contract also uses the `LimitOrderManager` to create and manage limit orders. It imports several libraries to assist with calculations and validations, as well as a number of interfaces for interacting with other contracts in the Dfyn protocol.

2. **Pool Deployer:** This contract is used by the `ConcentratedPoolFactory` (a periphery contract) to deploy new pools with the specified configurations.

3. **Vault:** In the interest of minimizing security vulnerabilities in Dfyn V2, we have introduced a dedicated vault contract to store and handle all the user funds rather than storing them in liquidity pool contracts. This contract maintains the balance for each user for each ERC20 token that they have deposited, and it uses a rebase mechanism to keep track of the total supply of each token. The contract also allows the owner to whitelist certain addresses, which allows those addresses to borrow tokens from this vault through a flash loan mechanism without paying a fee.

4.1.2 Periphery Contracts

Periphery contracts are high-level contracts responsible for handling core contracts and maintaining position-level data for concentrated pool liquidity and limit orders. Dfyn V2 comprises five major periphery contracts:

1. **Router:** As the name suggests, this contract acts as a router for transactions on the Dfyn V2 platform. For example, when users want to swap one token for another on the Dfyn V2 AMM, they have to interact with the `DfynRouter` contract. This contract automatically finds the best exchange rate for the desired swap and executes the trade on the user's behalf.

2. **Factory:** The `ConcentratedPoolFactory` contract acts as a factory for creating new pools on the Dfyn V2 platform. Furthermore, in Dfyn V2, we have introduced multi-factory compatibility, meaning that users can create multiple types of pools (customized pools) with the help of different factory types. This contract allows the owner to set the deployer address and update the swap fee for a given pool.

3. **Pool Helper:** The `ConcentratedPoolHelper` is a periphery contract for reading state information like tick state, price range, limit order range, among other things, from a concentrated liquidity pool without interacting with it directly.

4. **Pool Manager:** The `ConcentratedPoolManager` is responsible for handling liquidity in the concentrated liquidity pools. This contract mints Non Fungible Tokens (NFTs) to the liquidity providers as proof of their positions and maintains position-level data at all times. This contract has functions for minting and redeeming liquidity positions, collecting fees, calculating pool rewards, and updating the staked position.

5. **Limit Order Manager:** This contract handles limit orders in the concentrated liquidity pools — it has functions to create, claim and cancel limit orders. Additionally, the `LimitOrderManager` uses the `LimitOrderToken` contract to mint NFTs to limit order creators as proof of their limit orders and maintains position-level data at all times. This contract is used by the `ConcentratedLiquidityPool` interface to manage limit orders in a concentrated liquidity pool.

6. **Limit Order Token:** The contract has two functions - mint and burn. The mint function allows the contract's owner to mint new tokens by specifying the recipient and the token ID. The burn function allows the owner to burn existing tokens by specifying the token ID.

7. **Quoter:** The Quoter contracts are responsible for fetching quotes from various liquidity pools in Dfyn V1 & V2 AMMs.

4.1.3 hRFQ Contracts

As mentioned in section 3.2, Dfyn's high-frequency Request for Quote (hRFQ) engine is a no-slippage peer-to-peer trading technology between users and market makers. The hRFQ contracts are responsible for two things - (a) verifying the signature from the market makers and (b) handling the transfer of funds between the user and the market maker.

4.2 APIs & SDKs

4.2.1 hRFQ API

The hRFQ API is responsible for fetching the quote from the market makers with their signatures.

4.2.2 Signal API

As mentioned in section 3.3, Dfyn Signal is a smart order router that finds the most optimal path for a user's trade. The Signal API fetches the best trade for the users by comparing different token paths & protocols. This API powers Dfyn's UI.

4.2.3 Signal SDK

To simplify the process of querying Dfyn Signal and executing the quotes, we have abstracted it in our JavaScript SDK which can be integrated seamlessly by any JS-based project.

4.3 Trade Engine

The trade engine is responsible for executing trades for the user across different protocols with the quote data provided by Dfyn Signal.

5 Dfyn V2 Features

5.1 First-of-its-kind On-chain Limit Order System

One of the most striking features of Dfyn V2 is its on-chain limit order system. We have already covered the basics about our limit order engine in section 3.4. In this section, we'll compare Dfyn's limit order engine with existing frameworks in the market.

5.1.1 Centralized Limit Order Book Exchanges (CLOBs)

This methodology utilizes Web 2.0 services to maintain an order book where buy and sell orders can be matched via sorting algorithms in real-time.

One benefit of centralized limit order channels is low latency - they leverage order-matching engines to sort through a vast number of orders to match the right pair of trades and execute trades faster than any other method. However, while using centralized frameworks, users have to surrender the custody of their assets to the exchange. From the moment you initiate the trade till the time you withdraw the asset, as a trader, you lose possession of your assets.

5.1.2 Decentralized Limit Order Book Exchanges (DLOBs)

In DLOBs, the orders are stored either on or off the blockchain, but the trigger to execute the trades takes place off the chain. As the price inches closer to the target price of the limit order on the exchange, attempts are made by external entities to place the trade at that price. Hence, although christened as **limit orders**, such transactions resemble **market orders** because trade at the limit price is not executed via an automated mechanism of the exchange but through an external interjection.

One benefit of this model is that it allows the custody of the asset to stay with the user. Since there is no centralized repository to sort orders, there is no need to surrender assets to the exchange. However, this model is riddled with a variety of issues:

1. Although the trade occurs on-chain, orders are aggregated on off-chain nodes, due to which limit order data is shrouded from scrutiny.

2. Such orders act as traders that squeeze the liquidity instead of market makers that add to it. This is in contrast to Dfyn's CLMM model discussed below, where limit orders, by virtue of being part of the chain, add to the liquidity of the exchange.

3. This framework has a high dependency on third-party indexing servers to identify newly placed orders to be executed by bots.

4. Another major disadvantage of placing off-chain limit orders is the delay in triggers that can occur whilst executing the trade. The dynamism of the environment coupled with the inefficiency of the bots used to perform trades can lead to failure in realizing the limit order price.

5.1.3 Dfyn's Concentrated Limit Market Maker (CLMM)

Till now, we discussed two types of off-chain limit order models. Unlike those mechanisms, Dfyn's limit order engine does not have any nodes, bots, or order books to assist in placing orders. Instead, Dfyn utilizes a modified version of the concentrated market maker - the idea is to superimpose limit order liquidity on limit order ticks (refer to section 3.4.2).

There are several potential benefits to using Dfyn's on-chain limit order model:

- **Automation:** On-chain limit orders are executed automatically when the market price reaches the specified price, allowing seamless and efficient trading without needing a third-party intermediary.
- **Efficient Liquidity:** Limit orders placed on Dfyn don't take away from its liquidity but add to it.
- **Transparency:** Because on-chain limit orders are executed on a blockchain, they provide a transparent and verifiable record of all trades. This can help to build trust and confidence in the market.
- **Security:** Dfyn limit orders are facilitated by smart contracts, which provide a secure and tamper-proof way of executing trades. This reduces the risk of fraud and ensures that users retain control over their assets at all times.
- **Cost savings:** On-chain limit orders can help users to save on trading fees, as they do not require the services of a central authority to match buyers and sellers.

5.2 Delegation Vaults

In Dfyn V2, all funds committed for limit orders and AMM pools are deposited and managed in a separate vault contract. The mathematical operations for the addition/removal of liquidity, execution of trades & limit orders are handled by separate contracts. The vault is only used to deposit/withdraw the funds required to fulfill the operation in question. Since only a small part of the liquidity in the vault is necessary to complete the ongoing operations at any given time, the remaining funds can be used to provide flash loans or invested in blue chip lending/borrowing protocols like Aave [5]. By providing an additional yield on top of the fees collected from the AMM operations, delegation vaults will maximize the yield for the liquidity providers.

5.3 Dynamic Fee Model

In Dfyn V2, we have structured the fee module to allow for a dynamic fee model that automatically adjusts AMM fees in response to market conditions. For any given liquidity pool, this model is a function of (a) market volatility for the pair involved, (b) the amount of liquidity in the pool, and (c) trading volume against that pool. Numerous scenarios can emerge depending on these factors. However, Dfyn's dynamic fee model will ensure that, at any given time, the fees in a pool are favorable for both the traders and the liquidity providers, i.e., the fees will be set in a way that incentivizes traders to continue trading while encouraging liquidity providers to maintain their liquidity in the pool without worrying about the impermanent loss. The detailed mathematics behind our dynamic fee model will be released in the following months.

5.4 Cross-chain Capabilities

Unlike most DEXs, which are multi-chain at best, Dfyn utilizes Router Protocol's cross-chain asset transfer functionality to allow value transfer across chains. In other words, Dfyn acts as an entry and exit point to and from other chains and enables users to trade across chains in a single click.

References

[1] Ethereum Foundation. Ethereum Virtual Machine (EVM). https://ethereum.org/en/developers/docs/evm/, Jan 2023.

[2] Ethereum Foundation. Introduction to Smart Contracts. https://ethereum.org/en/developers/docs/smart-contracts/, Sep 2022.

[3] Vatsal Gupta and Ramani Ramachandran. Bridge Liquidity Paradox: It's Not Always Water Under the Bridge. https://beincrypto.com/bridge-liquidity-paradox-water-under-the-bridge/, Oct 2022.

[4] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core. https://uniswap.org/whitepaper-v3.pdf, Mar 2021.

[5] Aave - Open Source Liquidity Protocol. https://aave.com/. Accessed: 2023-01-12.