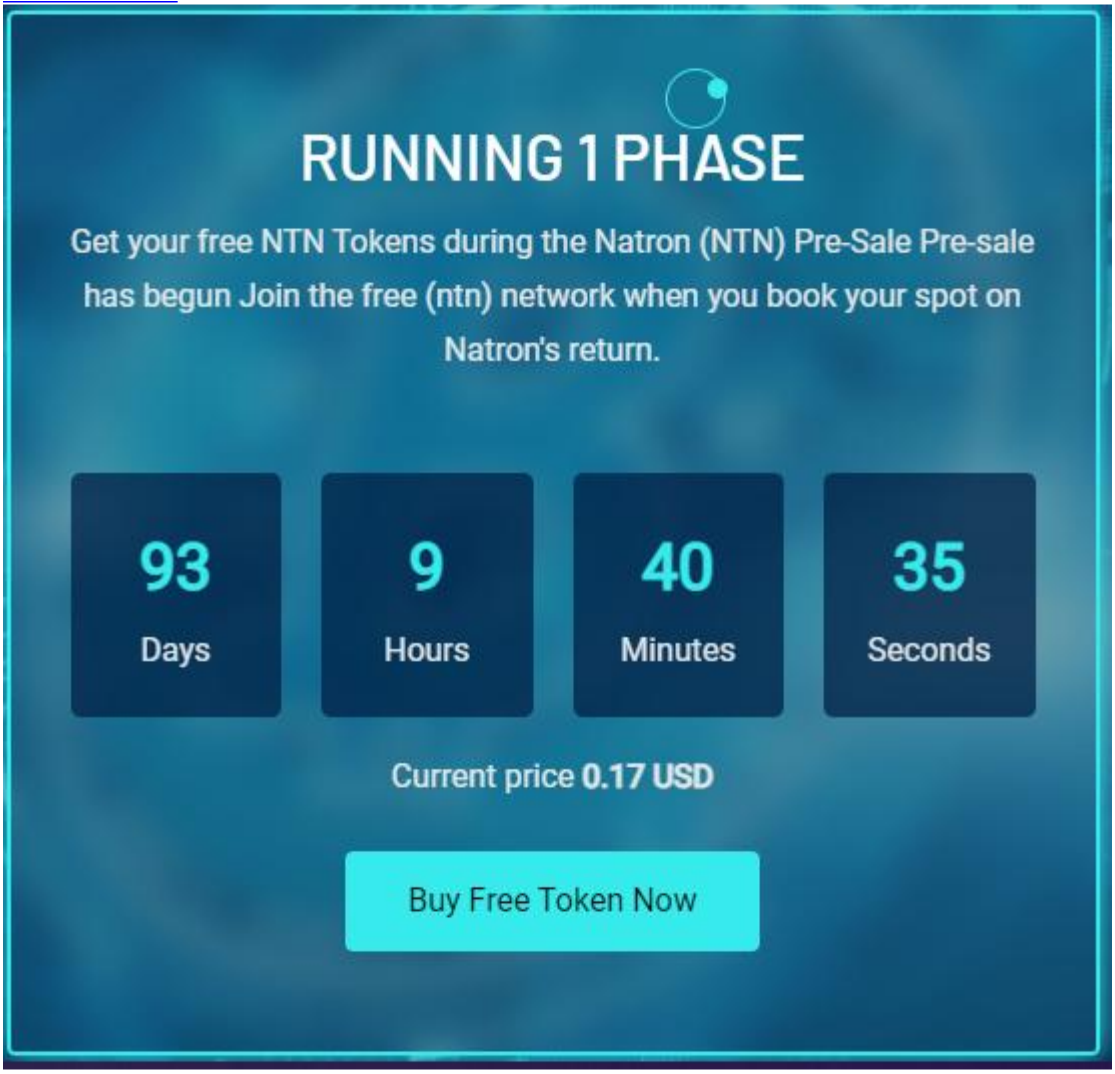
Natron DEX v1:
A Parallelizable DEX for a EUTxO-Based Blockchain

Marvin Bertin[1], Lars Bru¨njes[1], and Hud Wahab[2] [1]Natron Labs, {info@natron.in}

Abstract. Natron DEX is an order-book-based concentrated liquidity DEX designed to take advantage of benefits provided by EUTxO smart contracts, such as security, determinism, parallelism, scalability and composability. Natron DEX offers powerful features like Smart Swaps, enabling programmable and composable orders, and concentrated liquidity positions, which provide higher capital efficiency and higher yield opportunities. The Smart Liquidity Vault is introduced as a powerful secondary protocol built on top of the Natron DEX, providing yield optimization opportunities from AI-powered algorithmic trading strategies.

Natron Labs is hiring! Please send your resume to info@natron.in Natron Yield website & documentation



RUNNING 1 PHASE

Get your free NTN Tokens during the Natron (NTN) Pre-Sale Pre-sale has begun Join the free (ntn) network when you book your spot on Natron's return.

93	9	40	35
Days	Hours	Minutes	Seconds

Current price **0.17 USD**

Contents

1Introduction

Natron (NTN) DEX is decentralized exchange (DEX) designed specifically to take advantage of Cardano's Ex- tended Unspent Transaction Output (EUTxO)-based smart contract ledger.

The innovation brought by the EUTxO system enables Turing complete smart contracts absent on the original UTxO Bitcoin model. EUTxO-based smart contracts differ substantially from account-based protocols currently dominating the DApp space [1]. Therefore, novel protocol design patterns must be developed for the EUTxO model to take full advantage of the underlying data structure.

From first principles, Natron DEX is a complete redesign of the typical Automated Market Maker (AMM) DEX protocol. It is a new generation of DEX leveraging EUTxO architecture's benefits such as security, determinism, parallelism, scalability and composability.

The Natron DEX offers powerful features such as Smart Swaps enabling programmable and composable orders, and concentrated liquidity positions for higher capital efficiency and yield opportunities.

The DEX's improved order expressiveness and liquidity flexibility allow for powerful secondary protocols to be built on top of it. Natron's Smart Liquidity Vault is an example of one such protocol. By leveraging Natron DEX's advanced functionalities, yield optimization can be achieved with algorithmic trading strategies.

This paper is an overview of key design components behind the Natron DEX. It covers the following concepts: EUTxO smart-contracts, decentralized exchanges, concentrated liquidity, smart swaps, smart order router, and smart liquidity vaults.

Account-based Settlement Layer

A settlement layer is a distributed ledger or database that tracks all transactions past and present into the chainstate. The chainstate is a transparent and up-to-date record of all token ownerships across all participants at a given block. The chainstate is stored in each network's node and synchronized continuously through a decentralized consensus algorithm.

Account-based Model

The account-based model is simple; an account is used to hold the total token balance owned by a private key or a smart contract (Fig. 1). The account tracks the global state of the user's digital assets, where credit is added to the balance, and debit is debited. This represents how traditional bank accounts operate and by extension, many blockchains such as Ethereum.

The simplicity of the accounting model provides a great deal of flexibility especially when defining complex execution logic via smart contracts. The success of the accounting model is behind the explosion of DApps and DeFi protocols, aiding to fuel innovation in the smart contract sphere. However, excessive flexibility comes at the expense of execution guarantees, resulting in system vulnerabilities and attacks [2,3,4,5].

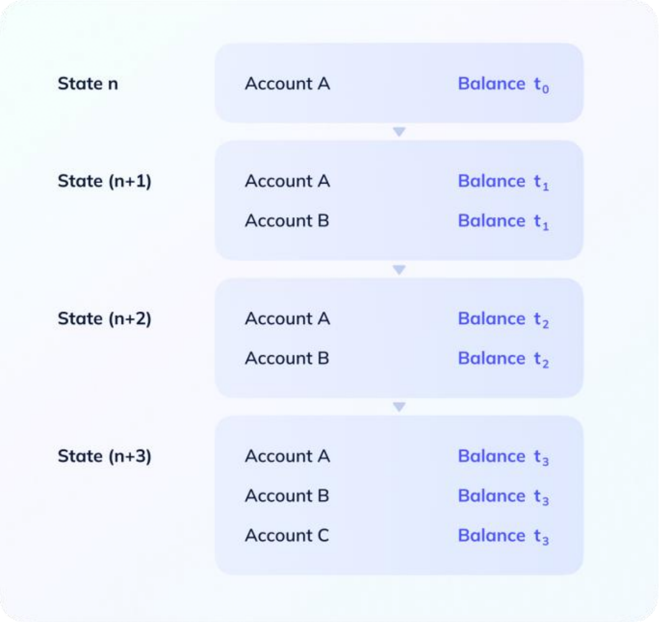


Fig. 1: State representations in an account-based model

Limitations of the Account-based Smart Contracts

Despite the popularity of the accounting model in smart contract-based blockchains, a key limitation is indeterminism. Unlike Bitcoin's UTxO model, account-based blockchains, namely Ethereum, are indeter- ministic. This means the effect a transaction has on the ledger is unpredictable because of the inherent mutability of user accounts. As such, the behavior of a smart contract is dependent on the global state and thus, the time of processing. Consequently, the lack of transaction behavior predictability leads to a number of issues:

Overpaying for transactions - Ethereum transaction fees are in part governed by a mechanism that is equivalent to a first-price auction. In this model, every transaction submits a bid; if accepted, they pay the exact bid amount. This is an inefficient bidding system that also incentivizes users to overpay for transactions. That being said, Ethereum recently introduced EIP 1559 [6], a proposal specifically designed to improve price predictability. Although a step in the right direction, it doesn't eliminate fee indeterminism.

Failed transactions - Submitted transactions are not guaranteed to succeed, however the fee will be charged regardless of the final status. A failed transaction can be caused by non-competitive gas prices (under bidding) that won't be prioritized. Another reason could be due to a programming bug in the contract itself that results in a transaction output "failure" state [7].

Network congestion - Since the ledger state is both mutable and global, transactions depend on the entire state of the network. As a result, network nodes must update frequently and transactions can only be executed by a single serial process. If one DApp experiences a spike in traffic, the whole network will be congested, subsequently affecting all DApps interacting with the ledger. That being said, network congestion can also be an issue on a UTxO blockchain.

Adversarial behavior - Indeterminism also provides opportunities for adversarial agents to abuse network vulnerabilities. For instance, front-running attacks, where a Miner Extractable Value (MEV) searcher bot scans transactions in the Mempool to preemptively set a higher bid fee, profiting at the expense of regular users.

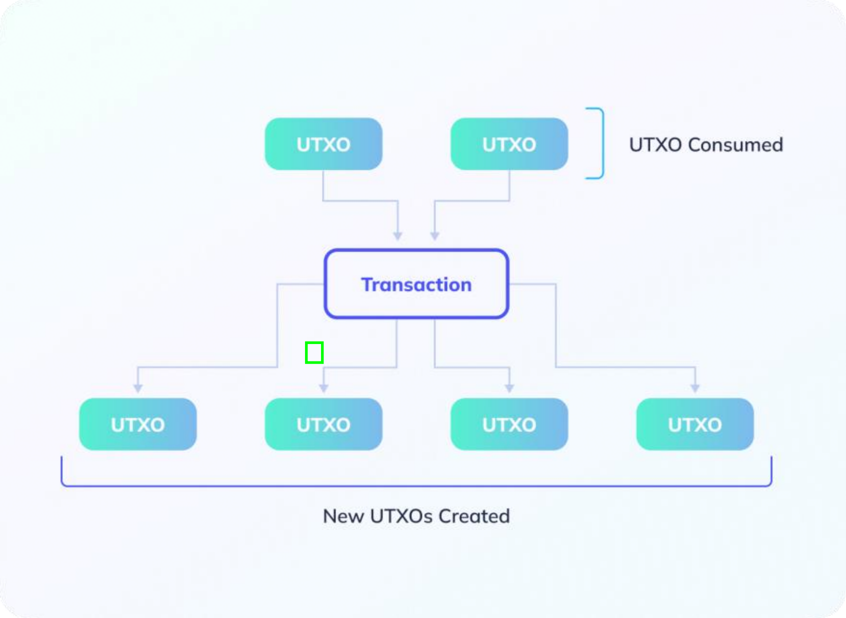


Fig. 2: Transactions in a UTxO model

In conclusion, the account-based model's flexibility comes at a steep cost and introduces issues solved in part by Bitcoin's UTxO model. However, the base UTxO model lacks the expressiveness required for a Turing Complete smart contract ecosystem. The EUTxO accounting model used by Cardano and Ergo aims at solving just that. EUTxOs provide the programmability of smart contracts of account-based models while also inheriting the determinism of UTxOs.

EUTXO-Based Settlement Layer

Unspent Transaction Output (UTxO) is a ledger accounting model used by Bitcoin and Cardano. However, Bitcoin's UTxO model does not provide the necessary programmable expressiveness required to support a Turing Complete execution language. For that reason, Cardano has developed the Extended Unspent Transaction Output (EUTxO) model [8] which upgrades the base settlement layer to support multi-assets [8] and smart contracts. This novel addition allows Cardano to take advantage of the original Bitcoin design while injecting the power of smart contracts introduced by Ethereum.

UTxO Model

A basic UTxO contains two types of information: a value amount of a native token such as ADA and an address such as a public key specifying the owner of the assets contained in the UTxO. Therefore, the assets associated with a digital wallet are stored in each block as multiple UTxOs fragments as opposed to a single wallet account.
As shown in Fig. 2, a transaction can be thought of as a function f (x1, x2, ..., xn) = (y1, y2..., yn) that consumes UTxOs (input x) and produces new UTxOs (output y). UTxOs are immutable and transactions must conserve value (total input value = total output value+transaction fee). These constraints prevent fundamental flaws like double-spending. Any subsequent transactions can then consume these outputs and generate new ones. In that sense, the aggregation of all transactions per block defines the state transition of ownership from one state of the blockchain (block) to the next. This is what we mean by the ledger.

The UTxO protocol maps perfectly onto a State Machine representation; it is therefore an appropriate and more intuitive mental model to use when describing such a system. This will become more apparent in the following sections.

Cardano's EUTxO Model

As the name suggests, the EUTxO model extends on Bitcoin's UTxO model and it does so in two distinct ways:

The concept of the UTxO address is generalized to contain a script rather than just a wallet address. The script defines a validator function that contains an arbitrary set of 'checks' in the form of an executable program. The validator simply inspects that a transaction's checks all pass and that the input UTxOs exist.

Additional data can be appended to both the output and input side of a transaction; datum on the output side can e.g. store the state of the UTxO, redeemer on the input side that has to match the datum to consume them, and the context that defines further conditions regarding e.g. time, fees. Thus, the scripts can be expressed beyond cryptographic signatures, by providing state information and state transitions. This is nothing more than a State Machine. Moreover, the EUTxO protocol may require more than one state machine for certain state transitions to converge [9]. Overall, val- idators can process more complex transactions while maintaining the composability and determinism provided by the UTxO paradigm.

V alidator(datum, redeemer, context) = True

EUTxO-based Smart Contracts

The UTxO model has a fundamentally different data structure from the account-based model. As such, UTxO-based smart contract design patterns also differ substantially from most existing smart-contract blockchains. As a result, DApps on account-based blockchains do not translate directly to Cardano since they need to be completely redesigned to match the underlying data representation. This is where the difficulty lies in bridging Ethereum protocols onto Cardano.

In spite of the large barrier to entry for new developers, the EUTxO model offers a number of benefits:

Security - UTxOs are immutable and can only be consumed once, simplifying transaction validation. Moreover, transactions are inherently stateless as they only track state transitions, and validators are inherently local as the validation process only depends on the transaction itself, as well as its inputs and outputs. The separation of logic description from runtime interpretation is at the heart of functional programming. Such declarative paradigms tend to be more concise, more predictable, and easier to test when compared to imperative programming [10]. The interested reader can refer to the next subsection and the references therein for our brief commentary on functional programming in the industry.

Determinism - How a transaction behaves and impacts the on-chain state can be predicted locally i.e. transactions are validated off-chain in the user's wallet before interacting with on-chain states (Fig. 3). This level of determinism translates to a predictable transaction cost and ledger state change. As such, a validated transaction is guaranteed to succeed and the transaction costs are fixed. However, a submitted transaction could still be rejected, meaning it is not applied to the ledger state, but no fees are paid. This can happen if an expected input has been consumed by another transaction.

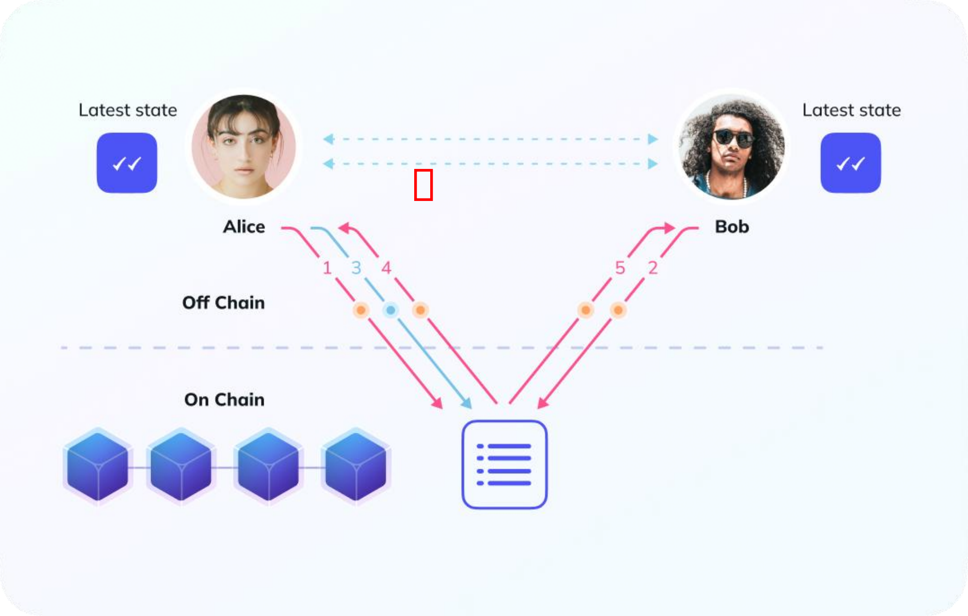


Fig. 3: Interactions with off- and on-chain states

Parallelism - Due to the local dependency of validators and the inherent state fragmentation pro- vided by UTxOs, parallel processing of transactions is both trivial and natural. However, if two transactions try to consume the same input, then processing one will automatically make the other get rejected. This is essentially the UTxO concurrency issue as it is often referred to. Therefore design patterns that trade concurrency for parallelism can benefit from a significant increase in throughput while keeping contract execution time the same [11].

Scalability - Transactions on an account-based model are processed sequentially, which is the equivalent to only being capable of executing a program on a single core of a computer [12]. An underappreciated fact about the UTxO model is that the transaction's computation happens off-chain before it is validated on-chain, without degrading any of the security and decentralization of the base layer (Fig. 4). In other words, the settlement layer is simply a verification model that validates the off-chain computation. This precise separation of responsibilities makes modularity and composability a core primitive of Cardano. As a result various novel scaling solutions can easily be "bolted" onto the blockchain, such as off-chain routing bots and isomorphic state channels, Hydra [13,14].

Composability - Smart contract code on account-based blockchains tends to become monolithic and complex as new features get added to old scripts, bloating the software and negatively impacting interoperability between smart contracts. Conversely, UTxO's determinism combined with Haskell's functional programming prevent side effects caused by shared states, making smart contracts composing much easier. UTxO's composability is achieved in two ways: 1) writing single-purpose programs (smart contracts) that do one thing well; 2) writing programs that interoperate together, reducing code duplication, and fostering script reuse/sharing across the network. Similar to Linux's command line, composability is achieved by combining well-defined programs in a virtually unlimited number of ways, enabling a large number of complex behaviors from a small number of simpler primitives.

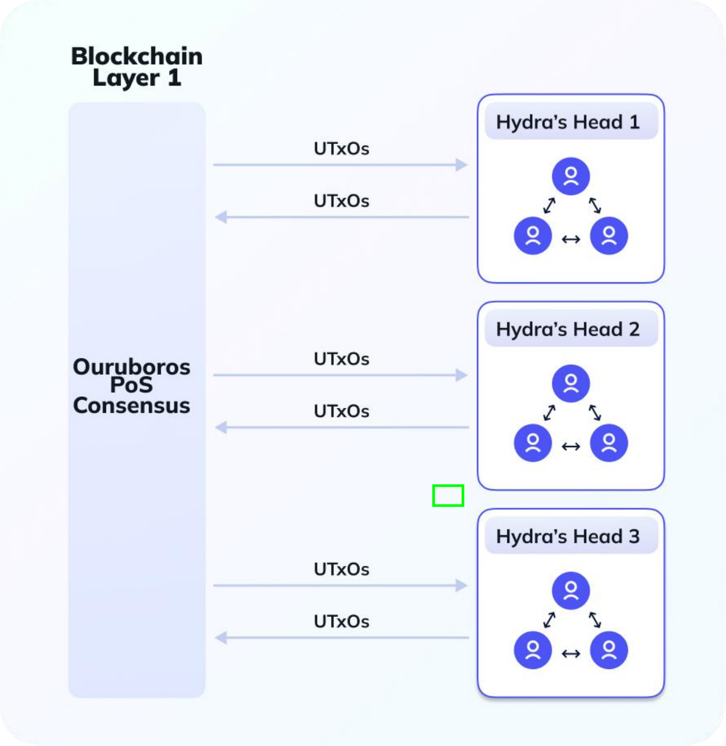


Fig. 4: Hydra: Layer 2 scalability solution
Functional Programming in the Industry

At the heart of the EUTxO paradigm is functional programming. To have a glimpse of what EUTxO-based DeFi could be, we briefly outline the adoption of functional languages in the real world [15]. For example, in the social media sector, WhatsApp uses Erlang to connect two million users per messaging server; Facebook relies on Haskell logic for active and automatic spam detection in daily news feeds. This illustrates the benefits of achievable scalability and reliability of functional programming. Functional programming is also widely adopted in the financial sector, credits not least to Peyton-Jones et al 's seminal work on modeling financial contracts in Haskell [16]. Functional programs accelerated evaluations of derivatives bringing advantages to large institutions such as Credit Suisse and Jane Street Capital, the latter being one of the world's largest market makers trading more than $17 trillion worth of securities in 2020 [17]. No one would risk losing such large sums of money, unless the underlying code is without bugs. We expect the benefits of functional programming to similarly supercharge scalability and reliability in DeFi.

Overall, well designed primitives, especially at the ledger level, with built-in guarantees at the core of the protocol can have compounding effects on the rest of the ecosystem. Strong fundamentals not only provide robustness and reliability, but also the building blocks for a flexible and composable system. A system in which expressiveness and complexity emerge from the interaction and combination of simpler components. It is particularly powerful for the development of DApps that can be infinitely extended like a castle of "money legos".
Consequently, there will come a point where protocols specifically designed for a UTxO architecture won't be replicable on an account-base model e.g. Ethereum. This is the competitive advantage Cardano can leverage to develop novel features unique in the crypto space.

2Related work

Decentralized Exchange (DEX)

Decentralized exchanges (DEX) are a type of cryptocurrency exchange protocol enabled by smart contracts. They allow for the execution of secure and direct peer-to-peer cryptocurrency transactions. In this scenario, the intermediary is replaced by a distributed ledger, the blockchain. There are two main types of exchange architectures: Automated Market Makers (AMMs) and order books. AMMs are simpler to implement and are the de facto design choice for account-based ledgers like Ethereum. However, the following sections will describe why an order book architecture is far more suitable for a UTxO-based ledger.

Automated Market Maker (AMM)

The first AMM-based DEX was implemented by Uniswap on the Ethereum blockchain [18]. It is a protocol that incorporates an autonomous trading mechanism and relies on a mathematical formula to price assets, eliminating the need for an intermediary (Fig. 5). Such protocols are permissionless, trustless and decentralized. They allow users to concurrently act as market takers (swap a digital asset for another) and market makers (provide liquidity to a DEX in exchange for trading fees).

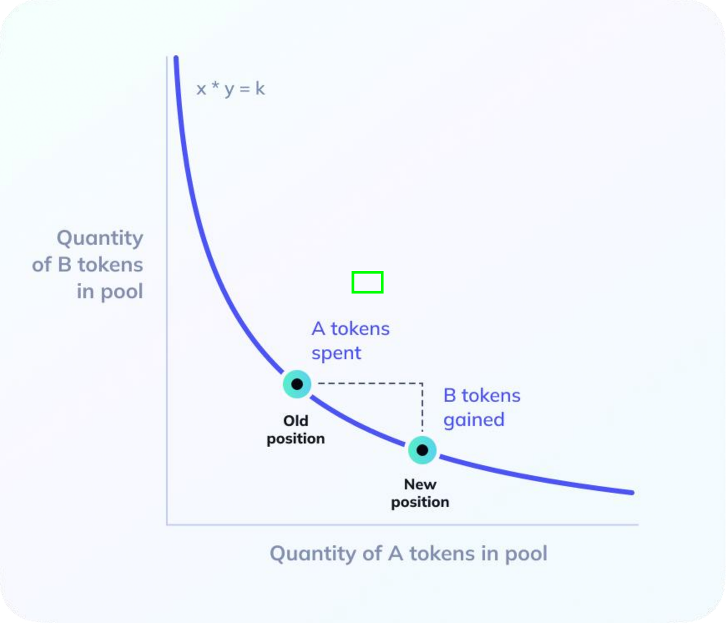


Fig. 5: Liquidity curves using the constant formula market maker

There are different pricing formulas but the most commonly used is the Constant Formula Market Maker (CFMM) [19]. It has the desired property that price response can be easily computed no matter how large the order size or how tiny the liquidity pool is. Their simplicity and robustness have made this design most favorable among the DEXs. However, they possess significant limitations that are often overlooked. First, capital allocation is highly inefficient compared to traditional markets. The DEX's Total Value Locked (TVL) is equally distributed across the entire price range (0, ∞), implying that the price of an apple is equally likely to be $1 or $10,000. This assumption makes the CFMM unrealistic and doesn't reflect actual market conditions. The main drawbacks of such capital inefficiencies are borne by the DEX’s liquidity provider since their revenues earned from trading fees are diluted. At the same time, their liquidity is subject to impermanent loss.

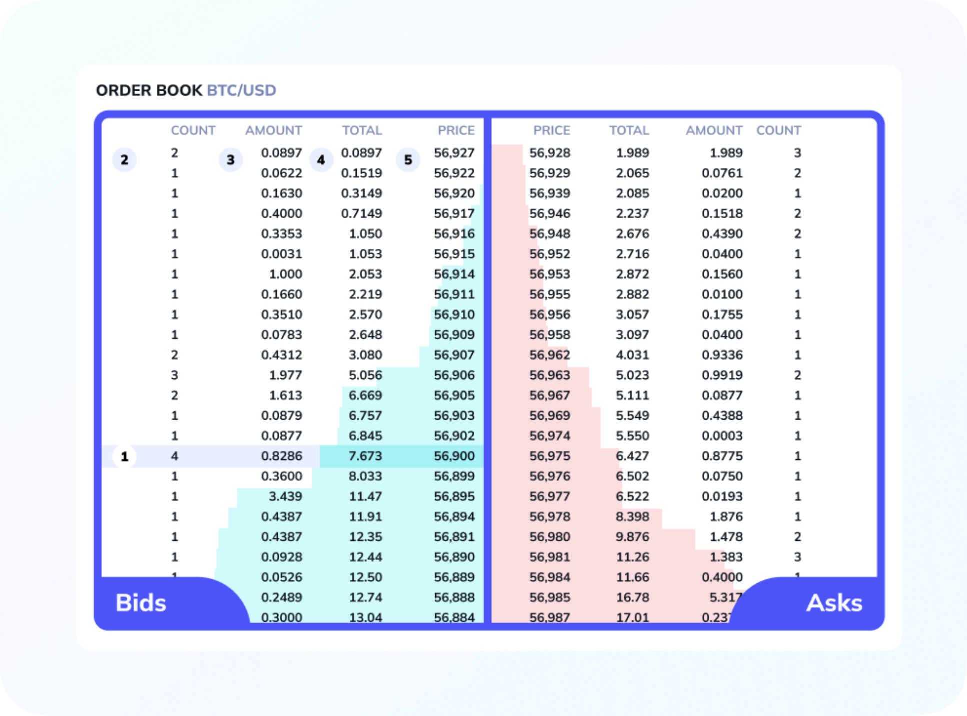
Impermanent loss is the perceived loss in total asset value from liquidity deposited into a pool, when compared to just holding the assets. It is due to a change in pool asset ratios (i.e. price) between a deposit and a withdrawal. This puts liquidity providers at a real disadvantage in spite of the revenue gained from trading fees. Therefore, in the absence of liquidity mining incentives AMM pools with high impermanent loss will see their liquidity drained overtime.

Order Book

The order book design, although rarely used in DEXs, is the go-to architecture employed by large exchanges in traditional finance, for example Robinhood and Coinbase. An order book is a registry of buy orders (bids) and sell orders (asks) at each price point (Fig. 6). In such a system, market makers help stabilize the price by filling trader’s orders and tightening the bid-ask spread.

Fig. 6: Order book exchange

Order book exchanges are capital efficient because price is maintained locally, where the bids meet the asks. The price is not a global state (like in an AMM), but an emergent dynamic property of the order book, allowing for an arbitrary liquidity curve as opposed to a deterministic one. Moreover, rich pricing and volume information is made visible to traders, making informed trading decisions easier by understanding market demand and supply. On an AMM, only market orders are possible (trades at the current price), whereas order book exchanges work with limit orders. Limit order is an order to buy or sell at a specified price, which remains in effect until executed or canceled by the user. In practice, it allows for more sophisticated trading strategies commonly used in traditional finance and traders can name their price.



ORDER BOOK BTC/USD

COUNT	AMOUNT	TOTAL	PRICE	PRICE	TOTAL	AMOUNT	COUNT
2	0.0897	0.0897	56,927	56,928	1.989	1.989	3
1	0.0622	0.1519	56,922	56,929	2.065	0.0761	2
1	0.1630	0.3149	56,920	56,939	2.085	0.0200	1
1	0.4000	0.7149	56,917	56,946	2.237	0.1518	2
1	0.3353	1.050	56,916	56,948	2.676	0.4390	2
1	0.0031	1.053	56,915	56,952	2.716	0.0400	1
1	1.000	2.053	56,914	56,953	2.872	0.1560	1
1	0.1660	2.219	56,911	56,955	2.882	0.0100	1
1	0.3510	2.570	56,910	56,956	3.057	0.1755	1
1	0.0783	2.648	56,909	56,958	3.097	0.0400	1
2	0.4312	3.080	56,907	56,962	4.031	0.9336	1
3	1.977	5.056	56,906	56,963	5.023	0.9919	2
2	1.613	6.669	56,905	56,967	5.111	0.0877	1
1	0.0879	6.757	56,903	56,969	5.549	0.4388	1
1	0.0877	6.845	56,902	56,974	5.550	0.0003	1
4	0.8286	7.673	56,900	56,975	6.427	0.8775	1
1	0.3600	8.033	56,899	56,976	6.502	0.0750	1
1	3.439	11.47	56,895	56,977	6.522	0.0193	1
1	0.4387	11.91	56,894	56,978	8.398	1.876	1
1	0.4387	12.35	56,891	56,980	9.876	1.478	2
1	0.0928	12.44	56,890	56,981	11.26	1.383	3
1	0.0526	12.50	56,889	56,984	11.66	0.4000	1
	0.2489	12.74	56,888	56,985	16.78	5.317	
	0.3000	13.04	56,884	56,987	17.01	0.237	

Why are order books not widespread in DEXs? Historically, AMM-based protocols have been easier to implement on account-based blockchains to mitigate transaction costs, settlement speeds, and complexity. In an order book exchange, liquidity is inherently distributed across bids and asks. Matching orders involve complex operations that require highly scalable matching engines, making them both technically and economically infeasible in an account-based paradigm.

UTxO-based DEX

AMM DEXs came out of the account-based mindset spearheaded by Ethereum. On Cardano, DEXs will be fundamentally different as they will need to take a UTxO-first approach to be successful. In a UTxO model, state fragmentation and parallel processing are base features of the ledger and coincidentally a natural design pattern for order books. The order book design has been battle-tested in traditional finance and has demonstrated to be both a powerful and efficient way to run large scale exchanges. UTxO-based smart contracts may be the right abstraction the industry needs to make such exchanges feasible on a blockchain.

Concentrated Liquidity

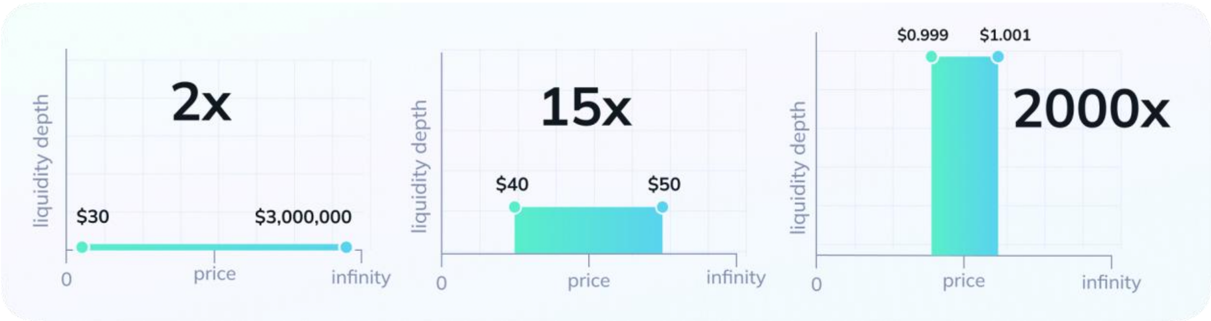
Concentrated liquidity allows Liquidity Providers (LPs) to allocate their capital to a DEX at selected price ranges. It is in contrast to regular liquidity pools, where liquidity is distributed uniformly across the entire price range. As mentioned in the AMM section, this results in high capital inefficiencies and impermanent loss. Concentrated liquidity pools attempt at remediating AMM’s limitations.

Uniswap V3

In Uniswap’s v3 whitepaper [20], the concept of concentrated liquidity pools is introduced, in which LPs can allocate their capital at specific price ranges called range orders. When asset prices exit the range order bounds, no liquidity is provided and consequently no fees are collected. This unlocks tremendous capital efficiency gains for liquidity providers (market makers) who can manually adjust their exposure based on their trading strategies (see Fig. 7). As a result, LPs can earn more rewards with less capital investment and incur less impermanent loss.

Fig. 7: The effect of concentrated liquidity

Instead of a fixed liquidity curve defined by the CFMM formula, the Uniswap V3 DEX allows the market sentiment to decide on the liquidity distribution in a decentralized way. This is, in fact, what naturally occurs in an order book system and the development of the concentrated liquidity AMM attempts at replicating these desirable properties.



Liquidity providers (LPs) can now create as many liquidity positions as they see fit. However, although more flexible than static AMMs, a concentrated liquidity AMM still isn't as flexible and efficient as an order book exchange. A drawback of concentrated liquidity pools is that they require active and frequent manual "rebalancing" of a LP's position to remain within the optimal price range and maximize rewards. For blockchains with high transaction fees, for example Ethereum, frequent rebalancing may not even be economical as rewards get canceled out by fees paid for the liquidity rebalance. Nevertheless, unless you are a day trader with technical understanding of the market, inefficient allocation of liquidity can still occur in a concentrated liquidity pool.

For these reasons, Natron (NTN) believes it is necessary to augment a concentrated liquidity DEX by overlaying a secondary protocol that can algorithmically automate asset allocation to maximize users' APYs, while minimizing risk exposure. Natron (NTN)'s Smart Liquidity Management solution will be an AI-powered protocol able to execute various trading strategies based on the user's predefined risk tolerance, expected returns and timeframe. We will dive deeper into the Smart Liquidity Vault protocol in a later section.

UTxO-based Liquidity

In practice, Uniswap V3 is implemented by fragmenting the liquidity pool into thousands of individual price ticks (price increments). Each price tick accumulates liquidity independently, allowing for an arbitrary liquidity curve (Fig. 8). Range orders are minted as NFTs, represented as a LP's active position. Positions are closed and rewards cashed out by simply burning the NFT.

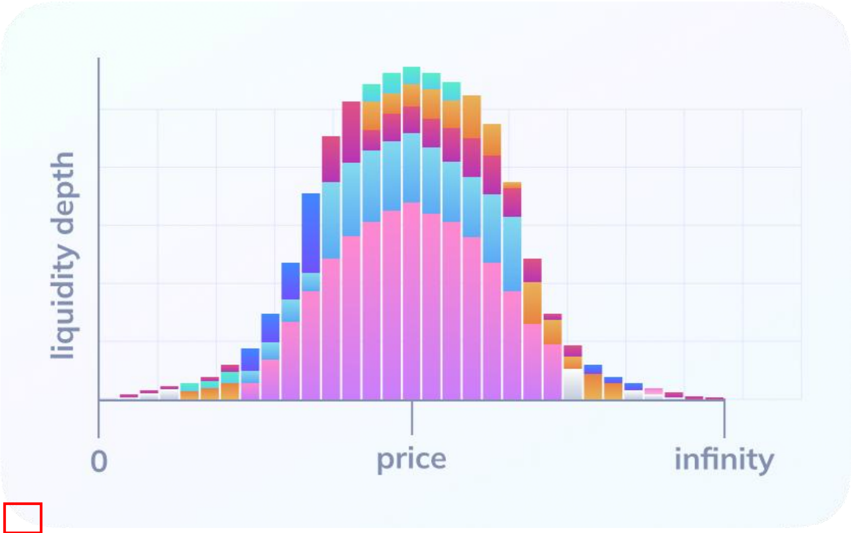


Fig. 8: Liquidity aggregated across price ticks

Upon closer look, the fragmentation of a liquidity pool maps nicely to the UTxO paradigm. Instead of tracking the pool liquidity in one UTxO, as it would be the case for an AMM, the price ticks dividing the liquidity can each be tracked by a separate UTxO. As a result, concentrated liquidity doesn't just make the DEX more capital efficient, it also maps well to the UTxO paradigm (more detail in Section 3.2).

3Methodology

Smart Swaps

Natron (NTN) DEX’s core primitive is the Smart Swap, which enables many of the DEX’s novel functionalities in a computationally efficient and elegant way. Smart Swaps are buy or sell orders that automatically execute based on some programmable logic. Smart Swaps are to the DEX what smart contracts are to the ledger. They allow for the automated execution of a trade triggered by a verifiable set of conditions. This enables users to submit orders that do not get filled instantly at the market price but instead ‘wait’ on the ledger until their desired conditions are met to execute the swap.

Smart Swaps are not unique to Natron (NTN). For example, a limit order is a type of smart swap in which a trade is triggered once a price is available. Such order types are prevalent in traditional, centralized exchanges e.g. Coinbase, Robinhood, etc. However, they are not common within DeFi, because the AMM model used by most DEXs makes such order types unfeasible.

DeFi protocols designed specifically for a UTxO model can provide new functionalities not feasible on other smart contract blockchains. Maladex, another well researched and designed Cardano DEX, has already introduced the idea of programmable swaps [21] and other DEXs are experimenting with new and creative architectures [22]. UTxO-based blockchains are now leading the charge in the research and development of the next generation of DEXs.

Smart Swaps don’t just limit themselves to limit orders. On the contrary, they allow for incredible flexibility and modularity with the potential to provide sophisticated trading functionalities not found in centralized exchanges.

Below are different types of Smart Swaps made available by Natron (NTN) DEX:

Market Order: An order to immediately execute a swap at the best possible market price. Since it has no trigger conditions, the execution price fluctuates with the market. It is equivalent to a swap on an AMM-based DEX.

Limit Order: An order to buy or sell a token at a specific price. As opposed to a market order, it does not execute immediately. Rather, it waits until a matching order (either a smart swap or liquidity provider) in the opposite direction becomes available. A limit order can be filled fully or partially based on the depth of market. Moreover, the order can be configured to expire after a certain time or be active until canceled.

Stop Order: A stop order will submit an order when a stop price has been met. A stop order functions like a limit order but with two critical differences. First, a limit order uses a price to designate the least acceptable amount for the transaction to take place; a stop order uses a price to merely trigger an actual order when the specified price has been traded. Second, a limit order is visible in the order book while a stop order is invisible until it is triggered. A stop order helps traders protect profits, limit losses, and initiate new positions.

Dynamic Order: An order where the trigger value is not price, but another data type such as an oracle data stream. More specifically, the trigger value can be a statistical metric or financial indicator commonly used in trading technical analysis tools. For example, a momentum indicator, such as the relative strength index (RSI) and the moving average convergence divergence (MACD). These indicators are typically used by traders to determine the strength or weakness of a stock’s price. With dynamic orders, trades can be triggered based on these metrics automatically, opening the door to more complex trading strategies.

Algorithmic Order: This type of order can combine all order types described above into a meta order. An algorithmic order can chain an arbitrary number of orders triggered by an arbitrary number of conditions spread out across time. This allows entire algorithmic trading strategies to be defined and executable within a single Smart Swap. For example, if a trader is bullish on the long term price of an asset, but believes short-term dips are likely, they may adopt a “buy the dip” strategy. On the other hand, if the investor believes a token pair is intrinsically correlated (such as two stablecoins tracking the same asset), then a good strategy would be to trade the volatility. Algorithmic orders allow for abstraction and automation of advanced trading strategies within a smart contract.

Natron (NTN)’s Smart Swap is a flexible and composable protocol on which expressive and powerful orders can be defined. In fact, (NTN) second product, Smart Liquidity Management, leverages the algorithmic order type to develop advanced strategies designed and backtested by our in-house (NTN)Labs team. We will dive deeper into the algorithmic order and backtesting platform in a future paper.

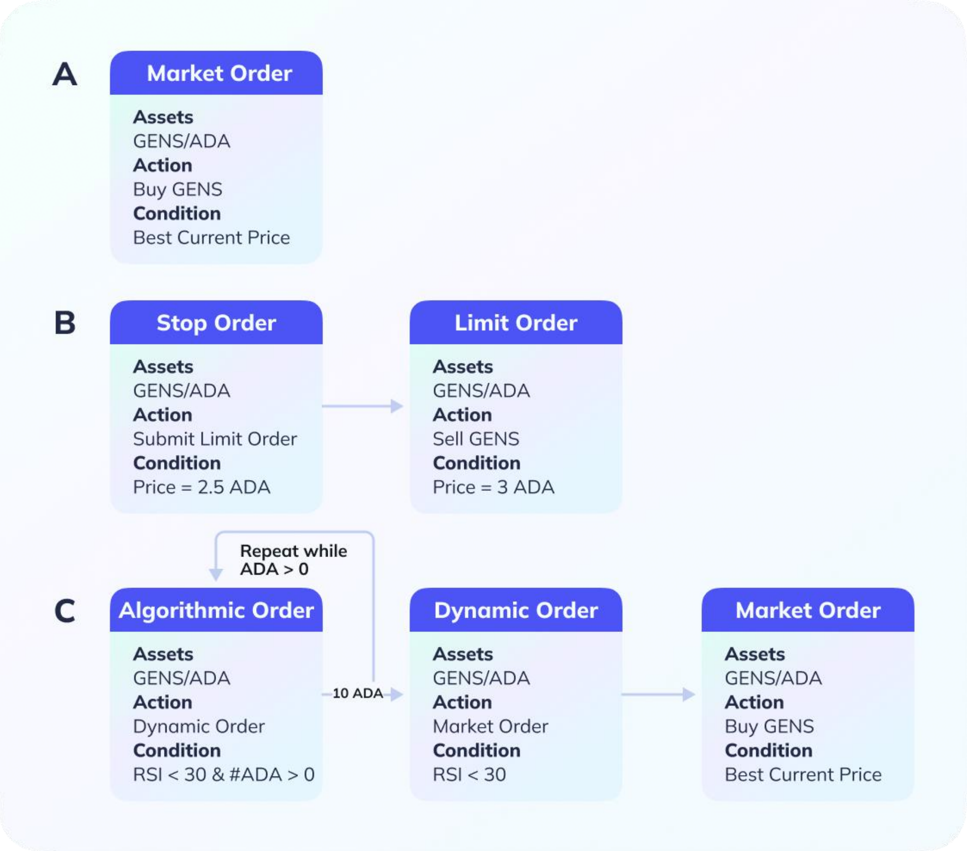
The Smart Swap protocol is simple by design and is defined by three attributes:

Assets: token pairs that will be traded with each other

Action: behavior of the swap when the order is executed

Conditions: set of conditions that must be met to trigger the action

Fig. 9: Examples of order types: (a) Market order, (b) Stop-Limit order: stop order triggers a sell limit order, (c) ”buy the dip” strategy: an algorithmic order transfers 10 ADA into a dynamic order, which submits a Buy market order when RSI (relative strength index) < 30 (signaling an undervalued asset)



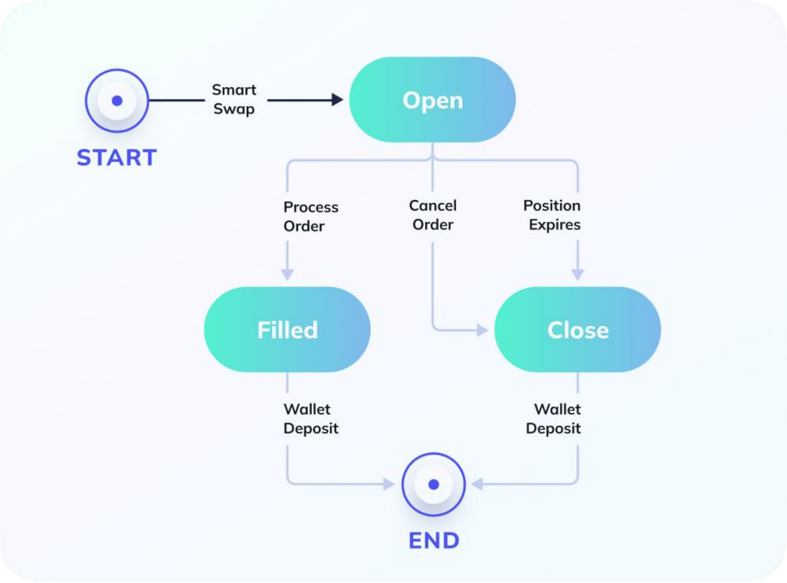
Therefore a limit order can be represented as a Smart Swap in which the Action is a buy/sell trans- action and the Condition, a price at which it must occur. Such a design provides a lot of flexibility and allows for complex order behaviors to emerge from the composability of order types. Figure 9 illustrates how different order types can be combined together to create different algorithmic strategies.

Smart Swap (Market Taker) State Machine

The EUTxO smart contract paradigm gives state machine diagrams the most intuitive representation of a protocol's functionalities (Figs. 10 and 12). For the market taker, the state machine diagram in Fig. 10 shows how a Smart Swap is defined in three states and five state transitions.

Fig. 10: State machines for the market taker
Smart Swap states:
Open: an order submitted to the DEX (EUTxO at the script address), but not yet processed
Filled: once an order's trigger conditions are met, routing bots execute the swap
Close: open orders not yet triggered can be canceled or expire after a set time
Smart Swap state transitions:
Smart Swap: submit a Smart Swap order to the DEX
Process Order: a routing bot matching the smart swap with a liquidity position and executing the transaction
Cancel Order: order canceled by owner before it is processed
Position Expires: order canceled itself after a predefined time
Wallet Deposit: transaction outputs return to owner wallets
Scalable Swaps

The appeal of such a design is in its relative simplicity and full parallelizability. The separation of order submission and order execution into distinct states allows users to submit DEX orders onto the blockchain in parallel without congesting the execution of other swaps. This is unlike traditional AMM DEXs, where all swaps are market orders that must be executed the moment they are submitted regardless of congestion or transaction fees. Additionally, Smart Swaps can be defined with minimum information (i.e. no pool state needed), resulting in lower fees.



Concentrated Liquidity Provider

A DEX that supports concentrated liquidity empowers liquidity providers (LPs) to optimize their liquidity allocation strategy. This enables LPs to potentially earn more yield, while making the DEX more performant and decentralized.

In a traditional exchange, a liquidity provider (or market maker) has the function to supply liquidity to a tradable asset and help stabilize price variation (volatility), reduce price slippage, and more generally facilitate an efficient price discovery. This is achieved by quoting both buy and sell orders of an asset with the goal of making a profit on the bid-ask spread.

In an AMM DEX, price is computationally derived by the constant product formula, whereas in an order book, price is defined by market forces of supply and demand. Therefore, LPs play a crucial role in providing the liquidity depth necessary to allow for an efficient marketplace. In other words, LPs supply the liquidity needed for the DEX to function, whereas the users represent the demand for said liquidity. It is the very interaction of traders, facilitated by the liquidity providers, that makes price discovery decentralized.

Let’s say we have a market for a trading pair A/B. A liquidity position would supply both sides of the trade at a specific price. Once a user opens a buy order for A at price X, a matching sell order for A will be executed between the two parties. The sell order can either come from another party’s limit order or a LP position available at that price. LP positions are very similar to limit orders except they are reversible orders. Once a LP position participates in a trade, the received liquidity stays in the same LP position to supply the other side of the trade.

Fig. 11: Example of a Smart Order Router matching a Smart Swap with a liquidity position

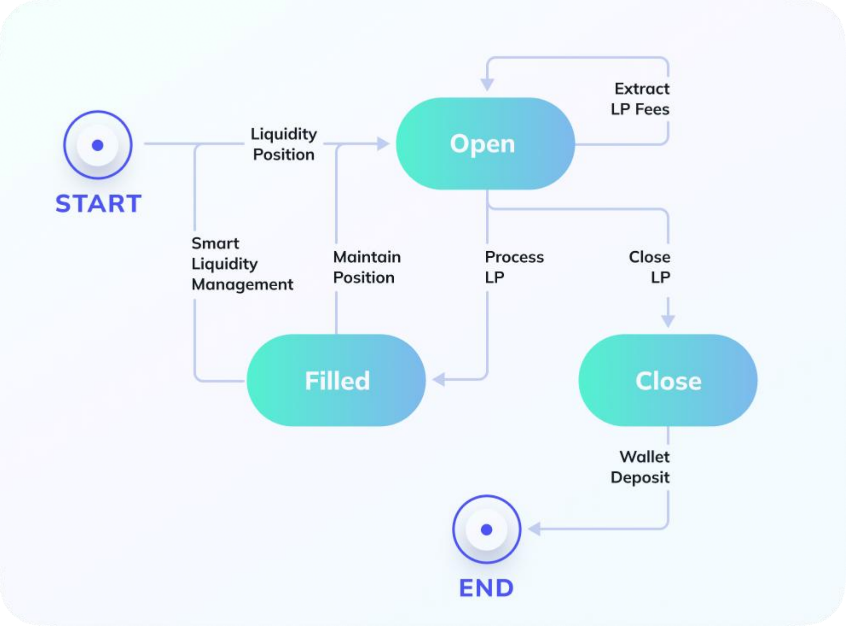
In Fig. 11, we have an example of a GENS/ADA pool. A user submits a Buy GENS limit order at price P3. However, no matching Sell GENS order exists on the DEX. Instead of remaining open, the order is matched with a GENS liquidity position, also available at price P3. The user wallet receives the GENS tokens and the LP receives ADA in return. For the LP, the ADA tokens are added back into its liquidity position at price P3, ready to accept trades in the opposite direction. In the next block, a buy and sell order gets submitted at P4. Since no ADA is available at P4, the sell order stays open. However, GENS liquidity is available at P4 and the buy order can be executed. Now ADA is available at P4 and the sell order can go through. Note that if the sell and buy orders at P4 were for the same amount, they could have been filled without using any of the pool’s liquidity.

How do Liquidity Providers (LPs) Earn Profits?

A LP earns a portion of the execution fee paid by the DEX user (Section 3.5). In addition, Natron (NTN) will give liquidity mining incentives to LPs that contribute liquidity to certain pools. Liquidity mining allows LPs to earn additional GENS protocol token used for governance such as voting on Natron (NTN) Improvement Proposals (GIP).

Liquidity Providers (Market Maker) State Machine
In Fig. 12, the state machine diagram for market makers depicts how liquidity positions are defined in three states and seven state transitions.

Fig. 12: State machines for the market maker
Liquidity Position states:
Open: a liquidity position submitted to a pool at a specific price range, ready to absorb trades. LP rewards are automatically reinvested, but can also be withdrawn without closing the position
Filled: once an order matches with a liquidity position, routing bots execute the swap. The leftover liquidity is either maintained in the current position or transferred in a more optimal price point determined by the Smart Liquidity Vault
Close: the full liquidity can be returned to the LP's wallet by closing the position Liquidity Position state transitions:
Liquidity Position: add concentrated liquidity to a pool
Process LP: a routing bot matching LP with an order and executing the transaction
Close LP: LP is closed by owner
Maintain Position: LP is reopened at the same price to accept trades in the opposite direction
Smart Liquidity Management: rebalance LP to a new price point based on strategy defined in the Smart Liquidity Vault
Update LP: add/remove liquidity or change price of position



Scalable Liquidity

In a UTxO architecture, centralizing the pool state into one UTxO (as in a traditional AMM) would constrain the DEX to one transaction per block. To achieve high throughput, the DEX must take advantage of the inherent parallelism properties made possible by UTxOs. For this reason, the pool state is divided along two dimensions (i.e price ticks and liquidity providers), enabling parallel processing of UTxOs within the same block (Fig. 13). In addition to increased throughput, fragmentation also minimizes memory requirements. Since each transaction is only dependent on a local state, this results in lower transaction sizes and thereby lower fees. Another desirable property of fragmented liquidity is the emergence of arbitrary liquidity curves. Instead of predefining a fixed curve like in an AMM, the liquidity distribution is dynamic, decentralized, and governed by market sentiment, resulting in a more efficient asset allocation.

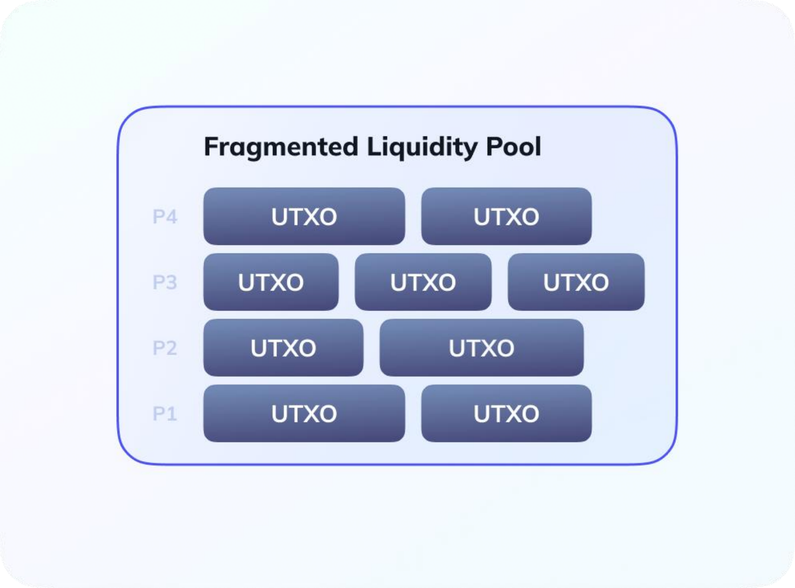


Fig. 13: Fragmented liquidity pool with UTxOs sitting at various price ticks P

Smart Order Router (SOR)

Orders on the Natron (NTN) DEX have two main phases (Fig. 14). First, orders are submitted to the Cardano ledger, then they are picked up and executed by Smart Order Routers (SOR). SORs are off-chain bots that execute a routing algorithm that scans the blockchain for open orders, matches them based on their trigger conditions and submits new transactions back to the ledger to perform the swap state transitions. Each Smart Swap encodes trigger conditions that must be fulfilled by the SOR to execute the swap.

SORs automate the process of handling orders and are designed to take the best available opportunity across a range of market conditions. They need to continuously scan and analyze the current state of the DEX (set of UTxOs defined by Smart Swaps and Liquidity Providers) and rely on a set of rules, configurations and algorithms to best execute a customer’s order, based on price and liquidity.

Order Selection Strategies

Order selection is a complex topic and an active area of research [23,24,25]. It is particularly challenging when designing strategies within a decentralized environment. For example, large orders may need to be

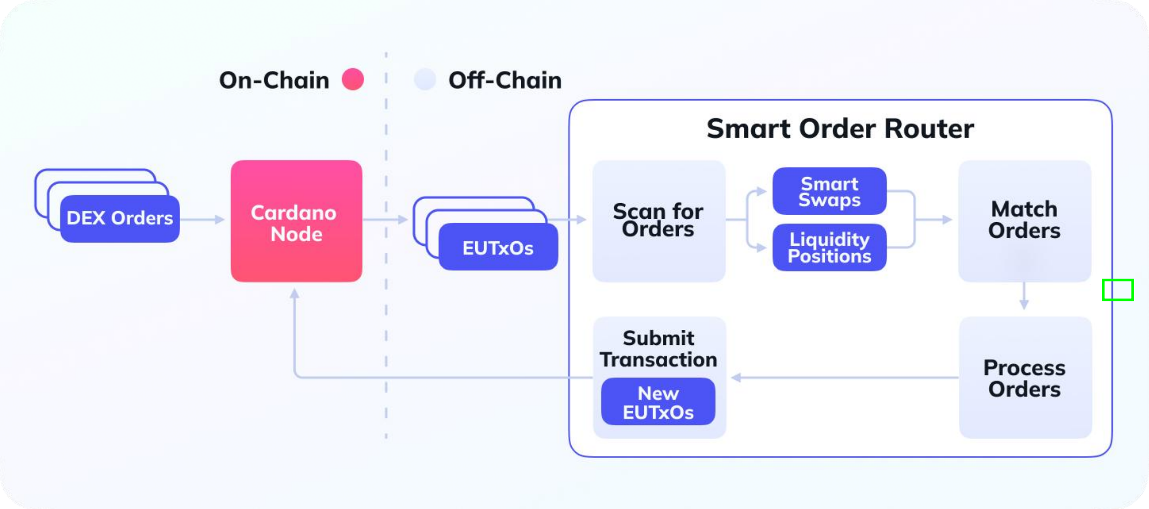


Fig. 14: Smart Order Router: off-chain bots responsible for matching and processing orders

matched across multiple orders to obtain sufficient liquidity. However, the more UTxOs are combined, the higher the likelihood that at least one UTxO will already have been spent by another SOR by the time the transaction is submitted (transaction sequences are mutually exclusive, creating a race condition). To reduce the number of UTxO needed to fill an order, one simple strategy is to search for the closest matching order value. However, this strategy tends to create a lot of ‘dust’, i.e. small unspent outputs that remain unused.

Alternatively, supporting partial orders through large order fragmentation could give more flexibility to SORs to optimize order selection. The issue here is that partial orders require more transactions and therefore incur more fees. Moreover, the generation of more transactions may worsen blockchain congestion (though L2 scaling solutions like Hydra will lessen this problem). All in all, order selection strategies need to be carefully crafted and fine-tuned to function effectively. It is an active area of research at Natron (NTN) Labs, and the team is taking a first principles approach to designing and validating strategies using simulations. As a starting point, Natron (NTN) is taking inspiration from IOHK’s coin selection algorithms used by the Cardano Wallet [26].

Decentralization

The code governing the Smart Order Router will be made open source for maximum transparency and accountability. SORs can be deployed by anyone willing to support the DEX and even customize the underlying matching algorithm for their own benefit. The goal is to foster an open and competitive environment, where the core DEX operation is in the hands of the community and market forces. However, as a first step, a restricted version of the DEX will be released with vetted SOR operators in order to ensure a stable DEX experience. As trading volume and liquidity grows, further decentralization will be introduced progressively.

Scalable SORs

As explained in an earlier section, off-chain computation is both practical and desirable in a UTxO model. Similar to Bitcoin, Cardano’s settlement layer is a verification model that validates off-chain computation. Therefore the distribution of off-chain logic can be used to scale order processing. At scale, the DEX’s operation is powered by multiple Smart Order Routers competing with each other for the best trades in order to maximize personal rewards. With the right set of incentives, competition can become a desirable force maintaining DEX performance. Natron (NTN) Labs takes an iterative approach combined with rigorous stress testing to produce fast order execution, efficient flow of liquidity and market-driven price discoverability.

How do Smart Order Routers Earn Profits?

SOR operators do not get a cut of the execution fee charged by the DEX. Instead, the SOR exclusively makes profits from the bid/ask spread caused by price volatility, low trading volume and other inefficiencies in the market. For example, a user wants to buy 1 GENS at a price of 10 ADA, but GENS liquidity is available for 9.5 ADA. In this case, the SOR can make a profit of 0.5 ADA per GENS token sold. SORs are essentially arbitrage bots that are rewarded for removing capital inefficiency in the market. When the prices of two assets fall out of line, the SORs will buy and sell until the relationship gets back in line. Therefore, introducing a competitive environment with many independent SORs will result in a tight, deep, and liquid order book.

Security

Operating a SOR is both a privileged position and potentially very lucrative one. Therefore, adversarial behaviors taking advantage of vulnerabilities are to be expected. Natron (NTN) Labs is carefully evaluating all possible attack vectors and manipulation strategies e.g. front-running, pool extractable value, trickle attacks, denial of service, etc. to produce a high performance, secure, and fair DEX experience. In future articles, Natron (NTN) Labs will explore DEX attack management and mitigation strategies.

Smart Liquidity Vault

A separate paper will be dedicated to the Natron (NTN) Optimizer protocol, describing in detail how the AI-powered algorithmic strategies are designed and optimized for each vault. Below is a brief introduction to this protocol.

In a concentrated DEX, creating liquidity positions becomes an active job for liquidity providers. LPs need to strategically position their liquidity at the optimal price to capture trade volume and maximize rewards. Moreover, liquidity positions need to be adjusted frequently to stay within the optimal price range where swaps are happening. This requires active monitoring and interaction with the DEX. For users that are looking for a more passive yield farming experience, Natron (NTN) created an active liquidity management protocol, called the Smart Liquidity Vault.

As shown in Fig. 15, vaults are liquidity pools managed by a trading bot executing a machine learning optimized algorithmic strategy. Different vaults execute different strategies based on the users' risk tolerance and outlook on the market. Vaults automate simple strategies like DCA (Dollar Cost Averaging), Smart Rebalance, or more advanced ones like modeling volatility and forecasting trend reversals. Each strategy is developed using a powerful backtesting ML platform leveraging historical data and blockchain data streams. The performance of each strategy will be methodologically tested and validated before deployment onto the Natron (NTN) platform.

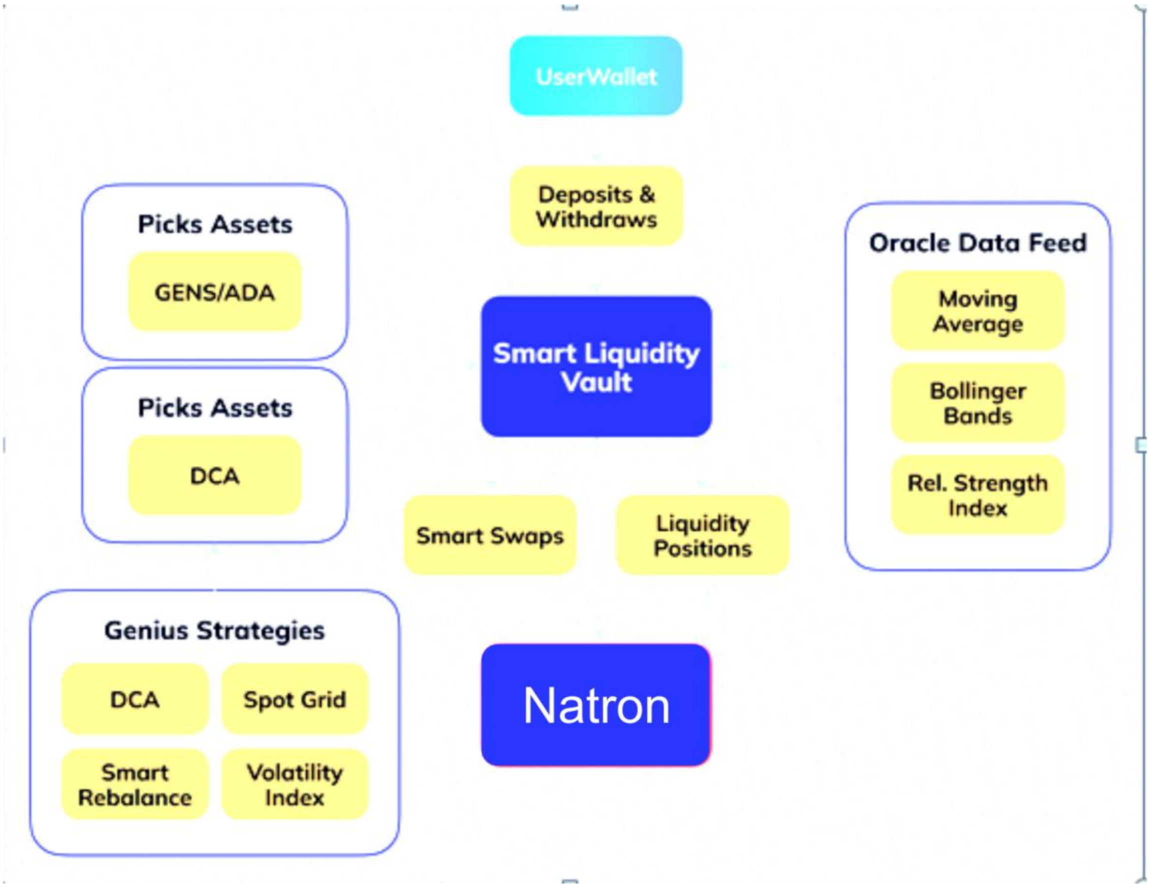


Fig. 15: Smart Liquidity Vault

Under the hood, Smart Liquidity Vaults leverage the Smart Swap abstraction and concentrated liquid- ity positioning functionality to execute their strategy. More specifically, the flexibility, expressiveness, and composability of dynamic order and algorithmic order types are the building blocks of these strategies. For these reasons, nothing stops a user from designing their own algorithmic strategies using Smart Swaps and competing with the Natron (NTN) Vaults. In fact we encourage it, and aim to support community-designed vaults in the future. Through a social trading leaderboard, any users will be able to learn and benefit from the community's best strategies. The vision is to empower the community to learn from the best and democratize yield optimization for all.

Natron (NTN) Platform

The Natron (NTN) platform ties together each of the components described above into an all-in-one DeFi ecosystem on Cardano. The platform is composed of two layers: the Natron (NTN) DEX and the Smart Liquidity Vault.

Natron DEX

The Natron (NTN) DEX is the core protocol that brings together the traders, liquidity providers and SORs to create the token swap marketplace. At its core, the DEX places orders onto the Cardano ledger and SORs match and processes them. Orders can either be Smart Swaps or concentrated liquidity positions.

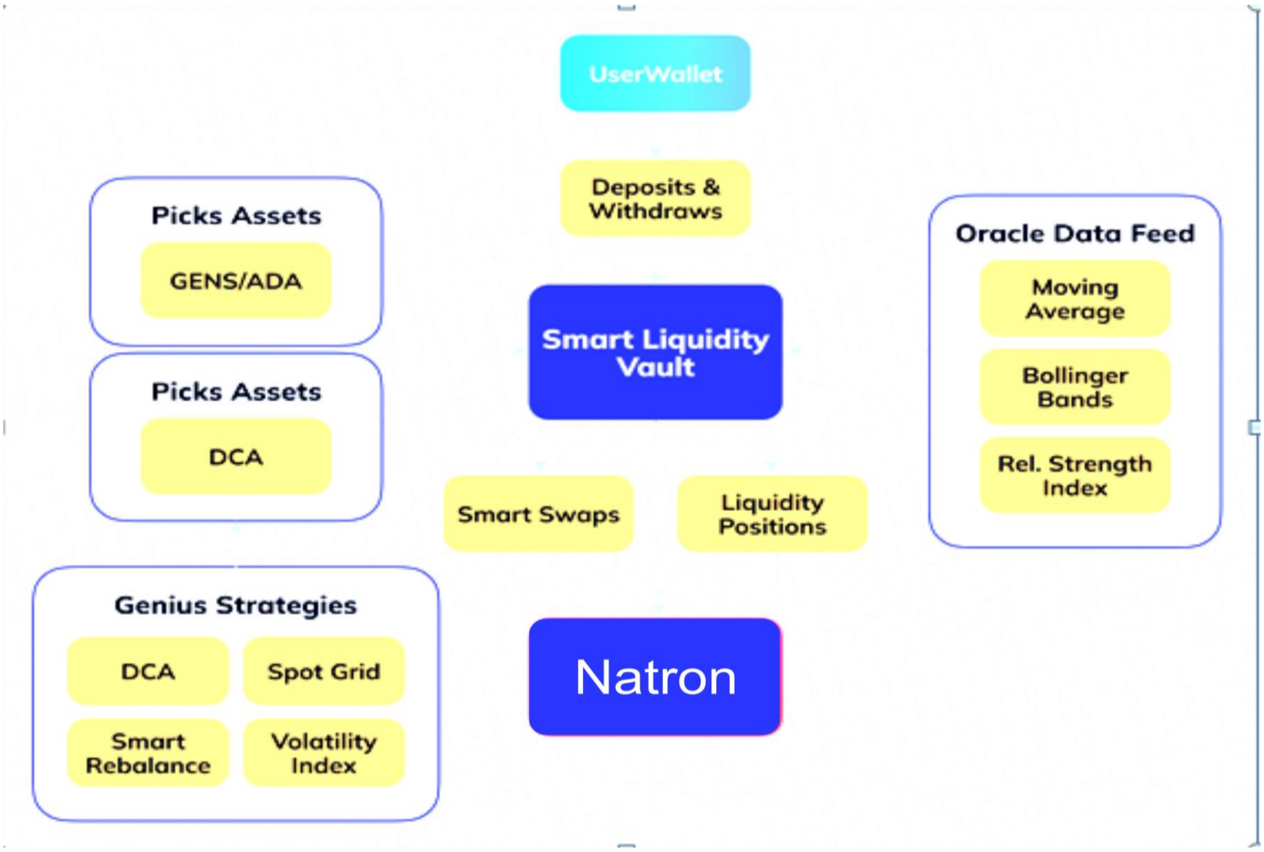


Fig. 16: Overview of the Natron (NTN) platform

Smart Liquidity Vault

The Natron (NTN) Optimizer sits on a level of abstraction above the DEX. It executes algorithmic trading strategies defined by each vault as a way to optimize a user’s yield farming opportunities.

Natron (NTN) DEX Fee Structure

The Natron (NTN) DEX does not charge a fee for placing an order except for the standard Cardano transaction fee. Therefore, it is almost free to submit a Smart Swap or place a liquidity position onto the Natron (NTN) DEX.
Order Fee (Smart Swap or Liquidity Position) = Cardano Transaction fee

A fee is only paid when the order gets executed by a Smart Order Router. The execution fee is charged as a percentage of the total order value and is divided into two components.
Execution Fee = % Total Order V alue = Natron (NTN) DEX Fee + LP Fee
where
Natron (NTN) DEX Fee: fees to the Natron (NTN) company, 20% of which is returned to GENS holders through the GENS staking program
LP Fee: fees to the liquidity provider
As mentioned in Section 3.3, the Smart Order Router operator does not get a cut of the execution fee, instead earns profits from the bid/ask spread. In other words, the DEX users do not pay for their orders to be matched and processed — the market's inefficiency does. This is an elegant solution to align incentives across the three parties while providing the cheapest possible price for the DEX users.
Note: the exact fee percentage breakdown is being researched. The document will be updated once finalized.

Smart Liquidity Vault Fee Structure

The Smart Liquidity Vault fee structure is centered around the vault fee. Smart Liquidity Vaults give users access to AI-powered algorithmic trading strategies. These vaults are actively maintained and optimized by the Natron (NTN) Labs team, charging a management fee combined with a performance fee.
V ault Fee = Management Fee + Performance Fee
where
Management Fee: fee charged as a percentage of the TVL (Total Value Locked) in the vault, 20% of which is returned to GENS stakers
Performance fee: fee charged as a percentage of the profits generated by the vault strategy, 20% of which is returned to GENS stakers
Note: the exact fee breakdown is being researched. The document will be updated once finalized.

GENS Staking Program

GENS token is only a utility token, therefore it is not used for any transaction payments. However GENS holders can choose to stake their Natron (NTN) in a Natron (NTN) pool to earn extra rewards through yield farming and other platform benefits.
GENS token's value centers around the Natron (NTN) Staking Program. The program is incentivized to create high demand for GENS by providing staking yields. APY generated by staking GENS comes from two sources, as outlined in Fig. 17.

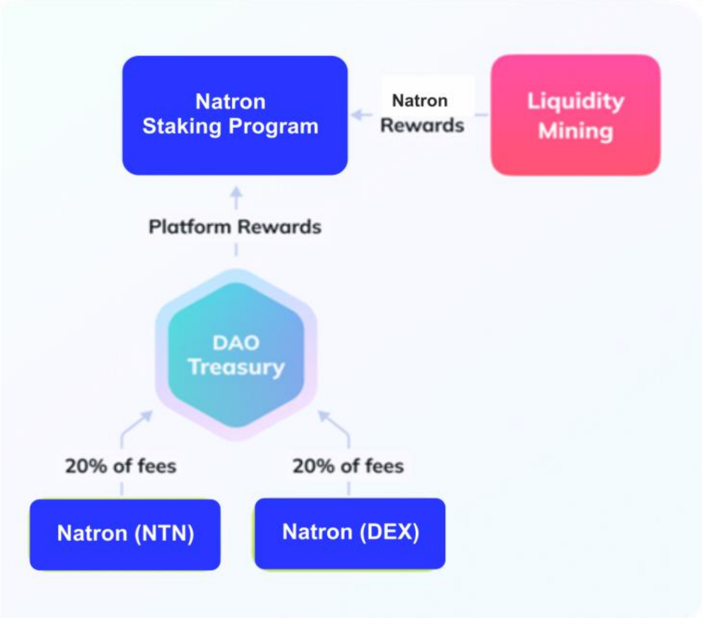


Fig. 17: GENS staking program: reward structure
Staking Rewards:
1.Redistribution of 20% of Natron (NTN) platform fees 2.Earn GENS through liquidity mining incentives
GENS staking isn’t just a source of yield, it also unlocks a number of platform features. Staking Benefits:
1.Access to advanced DEX trading features 2.Access to premium Smart Liquidity Vaults
3.Governance i.e. voting power in Natron (NTN) Improvement Proposals (GIP) 4.Unlock Natron (NTN) NFTs
Get access to Natron (NTN) X sales
Get access to Natron (NTN) Academy premium content
4Conclusion

This paper introduces the key concepts that differentiate Natron (NTN) DEX from a typical AMM DEX. It is an order-book-based concentrated liquidity DEX designed to take advantage of benefits provided by EUTxO smart contracts, such as security, determinism, parallelism, scalability and composability. Natron (NTN) DEX offers powerful features like Smart Swaps, which enables programmable and composable orders, and concentrated liquidity, which provides higher capital efficiency and higher yield opportunities.
Furthermore, the Smart Liquidity Vault is introduced as a powerful secondary protocol built on top of the Natron (NTN) DEX, providing yield optimization opportunities by automating algorithmic trading strategies. The overarching objective is to democratize DeFi for everyone by providing an all-in-one solution to yield optimization opportunities.
Natron (NTN) Labs is hiring! Please send your resume to info@natron.cin

Disclaimer

The content of this paper and the DEX protocol design are in draft form. 1Natron Labs reserves the rights to change, and make any corrections as it sees fit. 1Natron is taking an empirical and iterative approach to research and development to better understand limitations, evaluate tradeoffs, and fine-tune parameters in order to converge to an optimal solution. None of this content should be used to make any form of financial, tax, or legal decisions. This paper is for the benefit of the public and to foster open discussion and collaboration within the DeFi community.

References

5Glossary

arbitrage: simultaneous purchase and sale of the same asset in different markets to profit from small differences in the asset's listed price.

capital efficiency: the measure of how much capital is used to match orders when compared to how much capital is actually offered for trades at all possible prices in the pool

centralized exchange: traditional trading platforms that match buyers and sellers that functions essentially as online brokerage accounts

concurrency: the ability to structurally compose a program into units of independently executing computations. Functional programming is the most practical way to write concurrent programs that scale

concentrated liquidity: liquidity that is allocated within a custom price range, as opposed to being distributed uniformly along the price curve between 0 and infinity

composable orders: orders of different types that can be combined with each other, type-checked i.e. the constraints of types can be verified and enforced, and expressed as new composable orders

Extended Unspent Transaction Output (EUTxO): a novel system that extends Bitcoin's original UTxO paradigm that can be be used to design arbitrarily complex dApps due to its turing-completeness across transactions

functional programming: a programming style, where the main program is a function that is defined in terms of other functions, and the primary method of computation is the application of functions to arguments. It focuses on what is being computed rather than how it is being computed, similar to how we define mathematical functions

imperative programming: traditional programming style such as object-oriented programming e.g. solidity, Java, python, where computation is a sequence of transitions from states to states, as opposed to functional programming that has no implicit states. It is less declarative and often longer than functional counterparts, leading to more bugs

impermanent loss: a DeFi phenomenon where the rebalancing formula in AMMs creates divergence between asset price inside and outside of a liquidity pool

isomorphic: a concept in category theory, where two objects of a category have isomorphism between them i.e. they are the same for all practical purposes. This term describes the nature of Hydra head in Cardano's L2 solution, whereby off-chain transaction validation, including script execution, follows the same rules as on-chain semantics

limit order: an order type that specifies a price either above the current ask or below the current bid and awaits the movement of prices to become active. If the market is rising, the upward price movement triggers limit orders to sell; if the market is falling, the downward movement triggers limit orders to buy. Limit orders thus provide liquidity to the market.

market order: an order type that buys or sells at the market's prevailing price that typically has execution certainty

market maker/taker: The market maker places an order (to buy or sell at said price), while a taker accepts that placed order (to execute the buy or sell at the said price). Market makers provide liquidity and depth to markets to take profit from bid/ask spread or from predicted price entries during market volatility; market takers benefit from this liquidity and the ability to enter or exit positions quickly
momentum indicator: a technical analysis tool that shows trend direction and its magnitude i.e. when the price is moving upward or downward and how strongly

moving average convergence divergence (MACD): a popular and effective momentum indicator that composes two moving averages by subtracting the longer moving average from the shorter one

parallelism: the ability to execute multiple tasks at the same time. Functional programming is attractive for parallel tasks because they have powerful abstraction mechanisms and no side effects that eliminates unnecessary architectural dependencies for parallelization

price tick: A measure of the minimum upward or downward movement in the price of a security. In an order book, each price listed is a tick.

relative strength index (RSI): a popular momentum indicator to evaluate overbought or oversold conditions in the price of an asset
slippage: the difference between the expected and executed price of a trade
state channel: a L2 solution to scale up blockchains. The main idea is to achieve L1 security guaran- tees while limiting on-chain operations via off-chain interactions between users, reducing corresponding overhead

state machine: an abstract way of how computers and computations work. State machines have some internal state that can be changed in response to an external event. They are responsible for the safe transfer of UTxO sets that correspond to the final state of a Hydra head back to the blockchain
total value locked (TVL): the sum of all assets deposited or staked in a specific protocol