Starport

A Multi-Asset Limit Order Station

Designed, Specified, Implemented by:

Magnetar (@0xMAGNETAR)

Abstract: In this work, we present Starport: a generalization of the central limit order book to an arbitrary number of assets. Starport breaks the traditional pairwise asset paradigm and introduce limit orders that can be filled with multiple assets. In doing so, Starport unifies the liquidity fragmented across a set of assets under a single pool, increasing capital expressivity and efficiency while preserving the price control granted by traditional order books. With an internal router between orders with a common intermediary asset, Starport further deepens liquidity — providing takers with better prices over more fragmented exchanges, and makers with higher inventory turnover. This monolithic architecture may also significantly reduce the operational costs of trading (L1 fees), as a single transaction can post orders that otherwise may have required dozens of transactions.

Starport Labs

1 Motivation

With the establishment and growth of automated market makers (AMMs), including concentrated liquidity variants (CLAMMs), and highly optimized central limit order books (CLOBs), the flourishing world of decentralized finance (DeFi) on Solana shows much promise in the years to come. Together, these primitives provide solutions for trade among many types of assets. For example, CLAMMs and stable swap AMMs effectively mediate trade among groups of highly correlated assets such as liquid staking tokens or stablecoins, while CLOBs provide a powerful price discovery engine between more volatile asset pairs. Nevertheless, there are several problems that many of these DeFi primitives leave unsolved: liquidity fragmentation, and capital expressivity and efficiency. **Starport**, the world's first multi-asset central limit order station in all of finance, aims to ameliorate these issues.

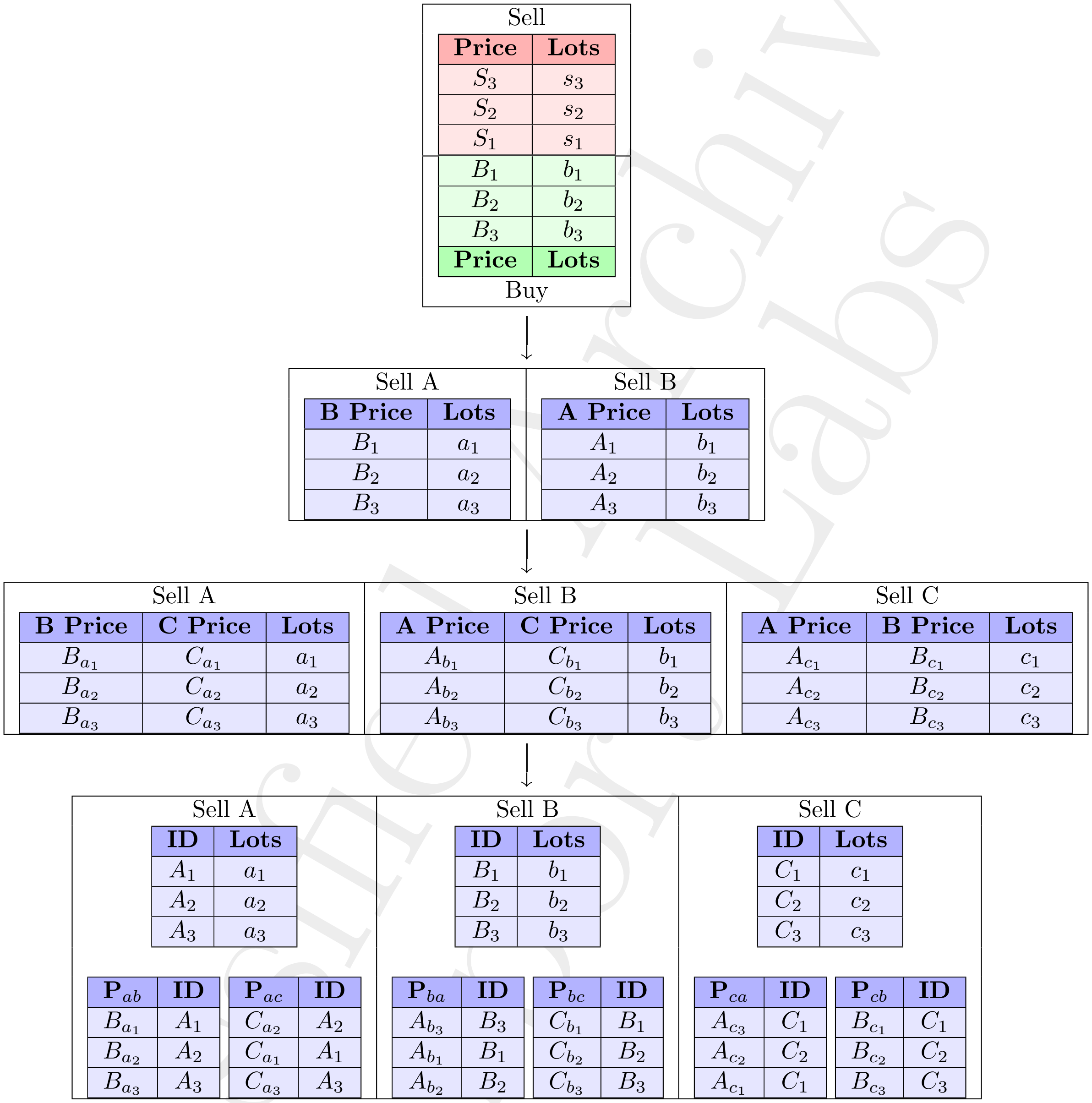
2 Starport: Liquidity Defragmentation, and Capital Expressivity and Efficiency

Starport ameliorates the aforementioned issues by breaking the traditional pairwise asset paradigm inherent to order books, allowing one-to-all trades between the assets present in a Starport. Accomplishing this requires introducing a new type of limit order, and abandoning the "book" for new data structures for order accounting. We cover the latter first, since it makes understanding the former more intuitive.

2.1 Limit Order Catalogs

To understand the generalization of the order book, we must first reinterpret limit order books. A limit order book can be understood as a set of two lists of orders sorted by price[1]. Every order book makes the distinction between the base and quote asset, and specifies a common tick size, base lot size, and quote lot size. This can be reinterpreted as two distinct **Catalogs** that simply list one posted asset (the asset being sold), and the price per lot for the asset that they are willing to accept. With this reinterpretation, it is easy to understand how one can add a third catalog and expand each of the catalogs to allow prices to be optionally specified for any of the other assets associated with the other catalogs. Figure 1 summarizes this train of thought and shows a sample extension to three assets.

[1] And submission time or any other prioritization attribute.



Sell

Price	Lots
S_3	s_3
S_2	s_2
S_1	s_1
B_1	b_1
B_2	b_2
B_3	b_3
Price	Lots

Buy

Sell A

B Price	Lots
B_1	a_1
B_2	a_2
B_3	a_3

Sell B

A Price	Lots
A_1	b_1
A_2	b_2
A_3	b_3

Sell A

B Price	C Price	Lots
B_{a_1}	C_{a_1}	a_1
B_{a_2}	C_{a_2}	a_2
B_{a_3}	C_{a_3}	a_3

Sell B

A Price	C Price	Lots
A_{b_1}	C_{b_1}	b_1
A_{b_2}	C_{b_2}	b_2
A_{b_3}	C_{b_3}	b_3

Sell C

A Price	B Price	Lots
A_{c_1}	B_{c_1}	c_1
A_{c_2}	B_{c_2}	c_2
A_{c_3}	B_{c_3}	c_3

Sell A

ID	Lots
A_1	a_1
A_2	a_2
A_3	a_3

$\mathbf{P}_{ab}$	ID
B_{a_1}	A_1
B_{a_2}	A_2
B_{a_3}	A_3

$\mathbf{P}_{ac}$	ID
C_{a_2}	A_2
C_{a_1}	A_1
C_{a_3}	A_3

Sell B

ID	Lots
B_1	b_1
B_2	b_2
B_3	b_3

$\mathbf{P}_{ba}$	ID
A_{b_3}	B_3
A_{b_1}	B_1
A_{b_2}	B_2

$\mathbf{P}_{bc}$	ID
C_{b_1}	B_1
C_{b_2}	B_2
C_{b_3}	B_3

Sell C

ID	Lots
C_1	c_1
C_2	c_2
C_3	c_3

$\mathbf{P}_{ca}$	ID
A_{c_3}	C_1
A_{c_2}	C_2
A_{c_1}	C_1

$\mathbf{P}_{cb}$	ID
B_{c_1}	C_1
B_{c_2}	C_2
B_{c_3}	C_3

Figure 1: Visualization of the generalization of the order book. (First) Traditional buy and sell side. (Second) An almost-equivalent formulation: a Starport with two assets. Note that it is not exactly equivalent because Catalog A specifies a lot size for asset A, while Catalog B specifies a lot size for asset B. (Third) Without loss of generality, an example extending the formulation to three assets. At this stage, there is no unique total ordering for the posted orders across all price columns. (Fourth) Restructuring tables to be indexed by each column price. P_{ij} is short for price of asset i in terms of j. Some sample sorting is shown. e.g. with price $B_{a_1} < B_{a_2}$ but $C_{a_2} < C_{a_1}$.

2.2 Starport Order

The schematic of the Starport limit order is summarized in Figure 2. Instead of having a base asset, quote asset, and a flag to distinguish between buys and sells, the posted asset specified assumes the equivalent role of the base asset. The order type T can be specified as Limit, Post, Immediate, Fill-Or-Kill. Limit and Post orders are sell orders, while Immediate and Fill-Or-Kill orders are buy orders. Limit orders have the usual definition, allowing partial fills for selling the posted asset when the order is posted, which remains valid until expiration. Post orders are identical to Limit orders, except that they are aborted if there is a partial fill when the order is posted (if the order crosses any of the pairs). Immediate orders are orders to buy lots of posted asset using any of the ask assets at the limit price specified at the time of execution – partials fills are allowed. Fill-Or-Kill orders are identical to Immediate orders, except that partial fills are not allowed.

Base Lots Posted	Price of Base in Quote Lots	Side
N	P_1	S

Posted Asset	Assets A Lots Posted	B Price	C Price	D Price	Type
A	N	P_1	P_2	P_3	T

Figure 2: Top (grey): A traditional limit order: all orders are in terms of the base asset. Bottom (blue): An order for a Starport with 4 assets. Any strict subset of the P_i can be null, which limits the assets that can fill the order. (Not shown: common attributes such as expiration time.)

2.2.1 Additional Order Types

Two additional independent order types are available on a Starport: **Omnibus** and **Alexandria**. One is geared toward makers and the other toward takers.

Omnibus: On traditional order books, market makers often submit sets of pairs of orders on each side of the book. These pairs typically share an expiration time. An Omnibus order allows a market maker to similarly post orders across all of the catalogs on the Starport. Essentially, it merges a set of Limit/Post orders into a more compact instruction, specifying only one expiration time and order type. Additionally, on a system like Solana, it eliminates all redundant account, instruction, and other validations that would be done repeatedly when posting several individual orders.

Alexandria: Alexandria orders are one-to-one asset orders that are internally routed through orders containing a common intermediary asset. That is, a user attempting to go from some source asset to some destination asset can be routed through the purchase and sale of some other asset on the Starport if orders to sell and buy the intermediary asset exist. This type of order is only useful on a Starport with more than two assets, as there are no intermediary assets to route through with only two assets. By invoking orders that would not usually be filled, offering Alexandria orders deepens liquidity and reduces price impact for users while simultaneously increasing volume for maker orders on less popular routes. See Figure 3 for a visualization.

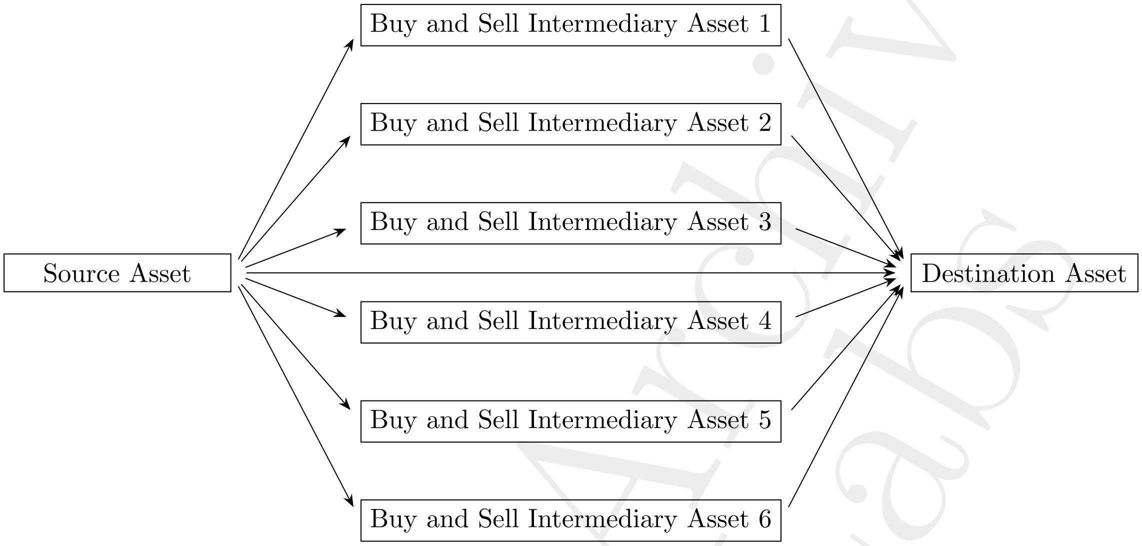

Buy and Sell Intermediary Asset 1
Buy and Sell Intermediary Asset 2
Buy and Sell Intermediary Asset 3
Source Asset
Destination Asset
Buy and Sell Intermediary Asset 4
Buy and Sell Intermediary Asset 5
Buy and Sell Intermediary Asset 6

Figure 3: An Alexandria order on a Starport with 8 assets. The direct route plus all routes including an intermediary asset are considered when filling the order.

3 Discussion

This generalization of the limit order has far-reaching ramifications. In regards to liquidity fragmentation, a smaller fraction of a market maker's portfolio is required to have open orders of equal size across the same pairs at any single point in time. In regards to capital expressivity, such an order is in the worst case only as expressive as a traditional limit order (when only one price P_i is not null). Inverting this statement implies that a traditional limit order suppresses expressivity — always forcing only one price P_i to be not null. In regards to capital efficiency, all else equal, an order with more than one non-null P_i can expect to be filled faster than an order with only one non-null P_i. This implies that market makers on a Starport can expect higher inventory turnover for the same capital [2]. This is further enhanced with the additional volume from the internal router used by the Alexandria order. Altogether, all else equal, users making markets for a set of assets on a Starport should expect higher returns compared to market making on the corresponding set of order books. If the Starport hosts a group of highly correlated assets[3], a market maker can expand from a single pair to the entire set and capture order flow across the other assets with little additional market risk[4].

Another benefit of a Starport containing many assets is the reduction in the operational costs of market making as it relates to transaction fees. We begin first with an analysis of transaction fees on Solana. Consider a high frequency trader that updates an order book every slot. One transaction every 400 ms would cost approximately 394 SOL per year. At a price of $100 per SOL

[2] Or equal inventory turnover with less capital, allowing them to make other markets instead.

[3] Examples include some quote asset such as USDC and one set of: liquid staking tokens, equal-expiry options with similar strike prices, associated in-game assets such as armor sets or potion sets.

[4] External risks such as vulnerable bridges or smart contracts that back one or several of the tokens or assets on the Starport are still present, e.g. depegging, drained reserves, etc.

that is almost $40,000 per year to update orders on the book. Consider a Starport with 8 assets containing 28 possible asset pairs. Making just five of these pairs independently would cost market makers $200,000 a year according to our previous calculation. However, on a Starport, orders can be posted on all 28 pairs (56 sides) or any subset thereof using an Omnibus order with just one transaction per update ($40,000 per year for one transaction per slot).

The idea of a multi-asset exchange is not new. Particularly, there exists constant product automated market makers with multiple assets. However, the constant product curve (or any other uninformed curve) makes them an ineffective option for liquidity providers outside of stable swaps. To our knowledge, there is not yet a CLAMM implementation for more than two assets. Starport Labs is currently researching this as a future product. It is not yet clear if there can exist an implementation which allows users to opt out of supplying liquidity on particular asset within the CLAMM, so such a product would serve different use cases than a Starport.

3.1 Drawbacks and Limitations

At Starport Labs, we aim to build the future while remaining transparent and honest. As part of this philosophy, we delineate some of the drawbacks and currently known limitations of Starport in this section and how we can ameliorate them.

3.1.1 Large Data Requirements

The current implementation of Starport does not allow for only a sparse set of routes within a Starport to be traded. A Starport with N assets contains $N(N-1)$ catalogs. Thus, with a large number of assets (roughly ≥ 8), the amount of data required on-chain can grow to be tens of megabytes. It is worth noting that an equivalent set of traditional order books covering the same routes will run into the same issue, but traditional order books do not require that all $\frac{N(N-1)}{2}$ trading routes be made available. Nevertheless, this can be ameliorated either with a sparse implementation that only allows a subset of possible routes to be traded, or by limiting the catalog capacity for larger Starports which is currently constant across all Starports.

It should be emphasized that the $\mathcal{O}(N^2)$ scaling behavior is not inherent to Starport itself, but rather is a consequence of the underlying combinatorics. Starport is not intending to solve or circumvent these combinatorics – there are legitimately inactive routes that are not worth serving or maintaining. With this in mind, the intended use case of a Starport is to host zero, one, or two quote tokens, and then many correlated or associated assets. Here are some examples of asset combinations that are reasonable to consider:

- **Liquid Staking Tokens** (8-10 assets): stablecoin, L1 token, and 6-8 liquid staking tokens.
- **Speculative or Memetic Tokens** (8-10 assets): stablecoin and 7-9 speculative tokens (e.g. dog tokens).
- **Categorical Tokens** (8-10 assets): stablecoin and 7-9 tokens from a particular category of protocols such as DePIN.
- **Stablecoins**: containing however many stablecoins are offered on the L1.

Choices like these open up trade routes between correlated or associated tokens without being subject to unbounded quadratic data (and compute) requirements.

3.1.2 Monolithic Design: Parallelization

Starport takes inspiration from the success of Solana's monolithic design. Solana offers a unified layer of atomically composable programs and state to deliver higher expressivity and efficiency to developers without external bridges. Correspondingly, Starport offers a unified layer of atomically composable liquidity to deliver higher expressivity and efficiency to makers without external routers. But, this has its trade-offs. In the initial implementation of Starport, write locks are taken across all catalogs within a Starport, even if trades are not routed through them. This is because it is necessary to check if the price specified for a pair crosses the other side of the pair; traditional order books similarly lock both the buy and sell side of a book when an order is posted on one side. This, however, reduces the capacity for parallelism across independent routes on a Starport, which is easily achievable with separate traditional order books (analog with sharded blockchains). This is a problem that is not inherent to Starport, but rather to routers that unify and deepen liquidity across different trade routes. Notably, the same problem arises when external router applications or aggregators unify different liquidity pools (AMMs, orderbooks) when dynamically checking routes at run time to deliver users with the best possible price. **It is the necessary cost of delivering the best market price available across relevant routes at the time of transaction execution**.

There is thus a tradeoff between optimizing execution price and parallelization. It goes without saying Starport Labs believes offering better prices and higher volume via unification is a prudent compromise over sharded designs[5]. Coincidentally, this problem is also ameliorated by the same solution presented in §3.1.1. That is, by having several Starports with associated assets instead of a single large Starport, write locks are only taken across catalogs associated with routes that are feasibly active.

4 Conclusion

We introduced a generalization of the central limit order book to an arbitrary number of assets. We discussed the necessary changes that need to be made to the limit orders and internal accounting to allow for one-to-all trades. We characterize the many benefits of this generalization, which can be summarized with:

1. Defragmentation of liquidity
2. Enhanced expressivity and efficiency of capital
3. A significant reduction in the operational costs associated with market making dozens of pairs between a handful of assets

We also discussed some of the drawbacks and limitations of this design, justified and ameliorated the trade-offs made, and briefly discussed our plans for future improvements.

[5] Designing a more sparse implementation that reduces the number of write locks is an active area of research internally. In particular, we are exploring the asynchronous, sparse implementation design space that strikes a balance between the composability of liquidity and instruction parallelism. Such an implementation may allow for one large monolithic Starport that only has a subset of possible routes active, and supports instruction parallelism across independent routes at the cost of introducing a crank and capping the number of ask assets in a limit order.