TL;DR

- Uniswap Labs has released a new protocol called UniswapX
- The X protocol's contracts are owned by the DAO's timelock contract and has a fee switch that is controlled by the DAO
- There are offchain components (APIs and user interfaces) that interact with the X protocol that are not owned or controlled by the DAO
- The X protocol is not an AMM but is complementary; it solves problems the AMM cannot
- The Foundation is soliciting input from community members

Introduction

The Uniswap community and in particular those of us who participate in UNI token governance have only ever had to consider our roles in relation to a set of contracts which enable the core AMM functionality. Specifically, Uniswap users have only ever been able to:

- Create liquidity pools
- Contribute liquidity to those pools
- Trade directly with those pools

Though its on-chain components are also governed by UNI token holders, the functionality UniswapX enables is a departure from these core functionalities. While we believe its feature set is complementary to the AMM and its development was driven by user needs that the core AMM cannot service, it is a distinctly new protocol.

At the Uniswap Foundation, we are still thinking about how X fits into our mandate. Clearly, it did not exist when the proposal that funded our organization was passed almost a year ago. Just as clearly (we believe), if it is successful, the core AMM benefits in a big way.

Similarly, the DAO has never had to govern anything other than the AMM. Should cross-chain deployments of X be declared canonical in the same way? Should the fee switch be turned on? Do you, as delegates, care about either of those questions?

This post will attempt to set the scene to allow the community to consider how X might be incorporated into the greater ecosystem. We will touch on the basic architecture of X including its on-chain components and the off-chain services required for its success as well as the functionality it enables and how that functionality works alongside Uniswap's core AMM.

We are still digesting UniswapX's design and features, as well as its implications for DAO governance and where it fits within the Foundation's scope of work. To that end, we would love to hear from the community. What do you

think about UniswapX? Feel free to start a discussion in the comments, or book a time to chat with me here.

UniswapX Technical Overview

Contract infrastructure

X the protocol is a collection of open source, immutable smart contracts. The protocol's core contracts are known as Reactors. At a high level, Reactors interpret trade orders and coordinate the distribution of tokens between counterparties according to rules encoded in those orders. While they are highly configurable contracts, each Reactor contract can only interpret one order type.

Reactors make use of signed orders, which necessitates a change in the transaction flow for Swappers. Instead of constructing a trade transaction and broadcasting it to the mempool themselves, Swappers instead just add their signature to a piece of offchain data that contains the terms of a trade that they would be willing to do; someone else executes it for them.

Reactor contracts can manage the distribution of fees on a trade-by-trade basis. Fees can be defined in two ways:

- The owner of the Reactor (currently the Uniswap Timelock) can configure a ProtocolFeeController

contract. That contract's configuration determines the criteria of trades that should be charged a fee, the address to which the fees are sent, and the rate (no greater than 5 basis points of the output token). A governance vote is required to enable the ProtocolFeeController

contract.

- Orders can be configured (most likely by user interfaces) to send fees of any amount of the output token to any address. Each order can have an arbitrary fee configuration (orders can have any number of outputs), and the swapper must sign the order for it to be valid.

Trade execution and counterparties

In the future there may be many different X Reactors, but there is currently a single implementation handling order flow on Ethereum mainnet. It uses a mechanism similar to a Dutch Auction (referred to in the documentation as a Dutch Order) to guarantee best execution for the swapper. You can read more about Dutch Auctions here.

The counterparties in a X trade are known as Swappers and Fillers. Both roles are entirely permissionless (and you can read about creating a Filler here).

Swappers add their signature to a piece of data known as an Order. An Order authorizes the exchange of token A for token B according to whatever rules the Reactor requires (starting price, reserve price, and time limit for a Dutch order).

Fillers ingest these orders programmatically and calculate the best price at which they can execute them. In practice, this likely involves surveying a number of on- and off-chain liquidity sources, including calculating the optimal route through Uniswap v2 and v3 pools (and eventually v4). When they've determined the best price they can provide based on the optimal combination of their available liquidity, a Filler is incentivized to submit the completed order as quickly as possible by calling a function on the Reactor; every other Filler in existence is trying to do the same. Once a valid order has been submitted, the Reactor executes it.

Supporting off-chain infrastructure

There are several important factors that will determine the success or failure of X that cannot be managed on-chain. We will discuss two of them here.

Order discovery

Fillers are incentivized to execute orders at the best price possible because if they don't, someone else will. However, a Filler that has exclusive access to some source of order flow does not face the same competition. If execution quality drops, Swappers will look elsewhere. It is therefore of paramount importance that as many Fillers as possible see every order and compete to fill it.

Orders are currently initiated by Swappers on the Uniswap Labs-ownedapp.uniswap.org. They are then passed to an API, also owned by Labs, where they are broadcast through a separate endpoint for Fillers to monitor, ingest, and execute.

Alternative UIs, such as GFX Labs' Oku, can coordinate with Labs and configure their front ends to pass orders to this same API. Those orders would then also be broadcasted to the Filler. Front ends, or the Swappers that use them, are responsible for parameterizing each order.

This API is currently the only solution for order discovery.

Order parameterization

The execution of a Dutch order requires that a Filler can provide a valid price for the trade between when the auction begins and when it ends. The parameters that determine whether a price is valid are set by the Swapper with help from the UI where they generated the order. A badly parameterized order will never be filled, so it is important that the UI provide guidance to the Swapper in setting these parameters (or abstract the process away entirely).

Uniswap Labs runs a proprietary RFQ service to help determine the Dutch orderparameters of the orders that Swappers broadcast from their front end to the Filler network. Other front ends may take similar approaches, or they could approach the problem in a different manner entirely. In general, the better-parameterized the orders that are being broadcast to the Filler network, the more volume will flow through X.

UniswapX's place in the ecosystem

Hayden has done a good job describing Uniswap Labs' rationale behind building X and we recommend watching his ethCC presentation or listening to his Bankless podcastinterview. At the Foundation, we have spent a lot of time digging into X and believe that there are two primary ways that it benefits the larger Uniswap ecosystem.

Complementary functionality to the core AMM

While AMMs have created a novel primitive that makes aggregating liquidity at scale possible, they do not provide the swapping ergonomics required for mass adoption. In other words, the design decisions that make Uniswap v3 an efficient place for Liquidity Providers to deploy capital cause some shortcomings in the Swapper's user experience. Specific

examples of these shortcomings include:

- A lack of visibility into execution price prior to receiving output tokens,
- Generally variable execution quality caused by MEV or other mid-block changes in liquidity conditions
- Variable and sometimes expensive gas costs for executed transactions
- Gas costs for reverted transactions
- Inability to access cross-chain liquidity

Implementing features that solve for any one of these problems within the AMM itself would require tradeoffs in the existing design. Those tradeoffs would almost certainly come at the cost of a degraded Liquidity Provider feature set. Instead of making sacrifices that might decrease the efficacy of the protocol for Liquidity Providers, X separates the concerns entirely. Effectively, X provides Swappers with customizable access to the liquidity provided by the core AMM's Liquidity Providers. The Dutch Order format that is currently live eliminates four of the failures noted above, and a pending cross-chain release handles the fifth.

A market-based approach to financing router research and development

Uniswap v4 presents a massively expanded design space for what a liquidity pool can look like. This expansion is exciting, but it comes at the cost of liquidity fragmentation - there can be dozens or hundreds of distinct pools per token pair, and each pool might make use of a hook that impacts the prices at which it will execute a swap. While the gas cost of routing through many pools to execute a single trade will drop dramatically because of the new singleton architecture, the computational overhead of determining that route has grown exponentially due to the increased complexity. Uniswap Labs' lead protocol developer Will Pote describes the problem well here.

X provides incentives, in the form of order flow, for any team in the world to build and maintain the best routing algorithm possible. Pote mentions in his thread that "…the skill ceiling for finding swap routes on V4 is significantly higher than V3" and this is undoubtedly true. A potential corollary is that the skill variance between routing teams could also be quite high. If X is processing a large volume of order flow, the profit potential for best-in-class router teams is very, very high. Creating a marketplace for routers, which is one of the stated goals of X, is one potential solution to the massively expanded complexity of the problem routers are trying to solve.

Open questions

While these benefits are compelling, there are open questions about X's impact on the current state of the Uniswap protocol. In particular, the Foundation is considering the impact X will have on the AMM's liquidity in fat tail pairs as well as those parties who provide it, and the potential aggregation of network effects to centralized order discovery services that are not owned by the DAO. We are interested in hearing feedback on the concerns, as well as any issues that we may not have considered. We encourage you to post in the comments here or open another topic entirely.