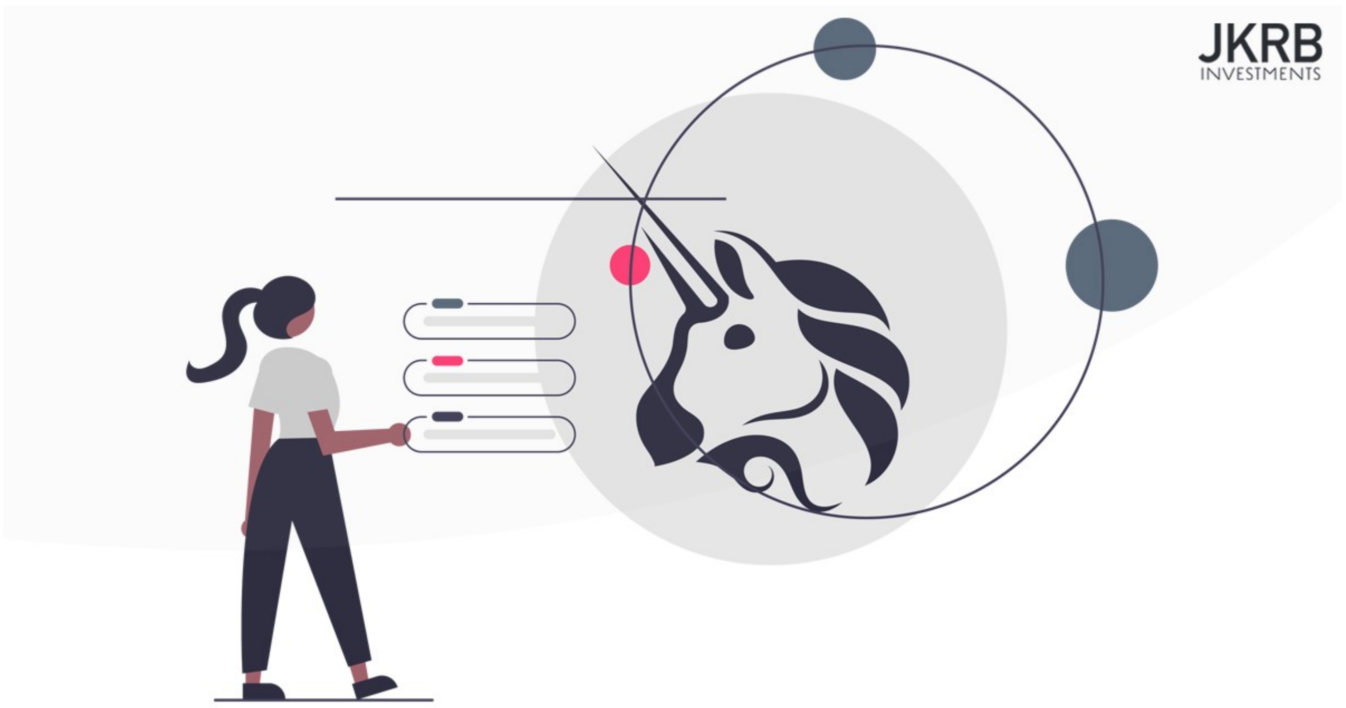
Uniswap V2: Everything New with the Decentralised Exchange

A comprehensive look at Ethereum's Uniswap V2 Protocol

Ross Bulat Aug 11, 2020 · 16 min read ★



Uniswap V2 is the next iteration of the on-chain exchange

Uniswap is an on-chain liquidity protocol on the Ethereum blockchain that enables trustless token swaps, meaning all transactions are executed from smart contracts without the need for an intermediary or trusted party. This decentralised approach to

Uniswap launched its second iteration on the Ethereum Mainnet on the 19th May 2020 to coincide with the core contract v1.0.1 release, after testing on the Rinkeby testnet for some time prior to this. The protocol has been used on the Mainnet for over 2 months at the time of writing, and thus far, no major breakages or issues have occurred pertaining to the integrity of the smart contracts.

If you are interested in holding and/or trading Cryptocurrency, Ethereum development, fin-tech, or finance trends in general, Uniswap should be on your radar. It has experienced some rapid growth in the 3rd quarter of 2020 to coincide with the booming DeFi space. Uniswap exists as a standalone service, but can also be integrated into third party wallets or Dapps. In addition to this, other smart contracts rely on Uniswap as a foundation to their services.

This piece will introduce the reader to the enhancements launched in conjunction with Uniswap V2 with an unbiased approach, covering the major features in detail.

Uniswap is also proving to be an integral piece in the DeFi ecosystem. To understand its place in DeFi and the major DeFi protocols in general, refer to my piece: ***DeFi: Exploring the Ecosystem around Yield Farming****.*

The foundation Uniswap V1 provided

Uniswap V1 laid the foundation of on-chain token swaps and decentralised liquidity pools that provided users rewards for providing liquidity, and charged small fees to make swaps.

Instead of a live order book, token exchange rates are calculated using what is termed the “constant product formula”, that will briefly be revisited further down. It essentially provides a mechanism to keep a token’s value balanced relative to the token pair in question.

The underlying token pairs (E.g. `DAI` to `ETH`, or `DAI` to `USDC`) each have separate liquidity pools associated with them, where users can contribute to the liquidity by depositing either one of the tokens in the corresponding pool. Liquidity providers are then rewarded a share of a 0.3% fee whenever a transaction is made — this share is based on the ratio of their pooled tokens relative to the entire pool supply.

how exchange rates are calculated, how liquidity providers can earn commissions leveraging liquidity pools, and the advantages & disadvantages of using the platform. Check out that introductory piece here: ***Uniswap: Understanding the Decentralised Ethereum Exchange*****.**

What’s new in Uniswap V2

Uniswap V2 brings a range of upgrades and enhancements to the protocol that build upon the swapping and liquidity mechanisms the first iteration introduced. The major changes include:

- **ERC20 to ERC20 token swaps**, where ETH is no longer required to be an intermediary token to facilitate the swap process. This was also known as “ETH bridging”. Removing this requirement cuts the transaction count in half and saves on gas fees. This also allows Dapps to effectively find “routes” from one token to another in the event that there is not a pool set up for a direct token swap. This piece will visit these changes in more detail further down.
- **Price oracle** functionality that allow time-weighted average pricing based on token pair prices at each block. We’ll visit the mechanics behind this mechanism in detail.
- **Flash swapping**, or being able to “borrow” tokens from a Uniswap pool, make some arbitrary transaction with external services, and *pay back* your originally borrowed funds, *all in one transaction*. The transaction is atomic, meaning it is reverted in full if at any stage the transaction fails. The obvious use case for such a feature is to execute arbitrage trades leveraging a liquidity pool, but there are other use cases that provide benefits such as cutting gas fees for performing particular DeFi actions, such as closing a Maker Vault.
- **Support of non-standard ERC20 tokens** by treating a `void` return type of `transfer()` and `transferFrom()` as a successful transfer. This may seem trivial, but major tokens such as `USDT` (Tether) and `BNB` (Binance Coin) do exactly the above on their transfer methods. Including widely adopted tokens that did not completely adhere to the ERC20 standard has strengthened Uniswap’s proposition to be the leading on-chain exchange in terms of usage.
- Additional utility methods to prevent overflow if more tokens are transferred than Solidity is capable of supporting.

what was outlined. This issue makes a token non-standard. Renaming or removing required methods also makes tokens non-standard, but this is less practiced and would require major workarounds if protocols like Uniswap wanted to support such deviations from the finalised spec.

Uniswap V2 introduces a toggle-able 0.05% protocol charge that eats into the standard 0.3% fee, but it is currently turned off — and can only be turned on with a decentralised governance mechanism. We'll cover more about that further down.

Uniswap V2 rolled out after extensive testing and auditing

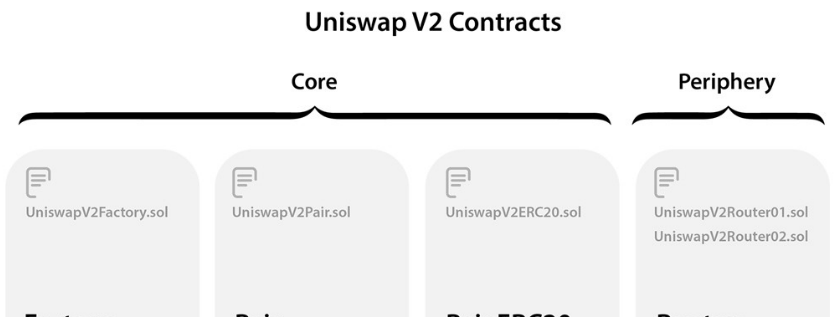
It has been well documented that the immutability of smart contracts pose a risk of catastrophic token loss / freezes if there are bugs present in them. If smart contracts are poorly written and contain errors that result in locking up funds, then there is no way to revert those transactions (unless a hard fork is performed on the entire blockchain — something that is very unlikely to happen given the size of Ethereum and how many parties now rely on it).

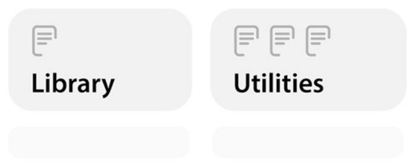
The Uniswap team, to their credit, followed industry standard practices to ensure that the risk of bugs and logic errors in their contracts are kept to a minimum.

The move from Uniswap V1 to V2 saw the contracts re-written in Solidity, from Vyper in the first version. This successfully overcame the limitations of Vyper and allowed the Uniswap developers to leverage the newer opcodes that the latest Solidity versions have rolled out, further optimising the contract execution in terms of CPU resources used (and therefore gas fees).

Uniswap V2 relies on multiple smart contracts ranging from the Factory, Router (V2), Pair and Pair ERC20 contracts, along with a Library contract for utilities.

The major contracts that get Uniswap working are as follows:





The structure of smart contracts that make Uniswap V2 work.

Router2 is a more optimised version of Router1, but they both perform the same tasks. Uniswap advise that all developers now switch to using the Router2 contract.

The total re-write of Uniswap warranted third party audits from respectable development studios as well as extensive testing on a testnet to mimic real-world usage. This happened — with Consensys Diligence being one of the parties contributing their own comprehensive report of Uniswap V2 source code. The full audit can be read here.

It appears that Uniswap V2 was extensively audited and reviewed prior to the Mainnet release. Carrying out this critical task of due-diligence on the source code has *most likely* ensured that there will not be any major bugs found in the future, although Uniswap do have a Bug Bounty Program if any major flaws are discovered.

Security is an ongoing concern in the blockchain space, and understandably so given the amount of value being traded in the field. Attacks on underlying blockchain mechanics that pertain to consensus, block creation and transaction validation are the focus of a lot of research in academia, but smart contract integrity is arguably just as important in the case of Ethereum where locked-up tokens rely on the integrity of the smart contract logic to keep them secure and reachable.

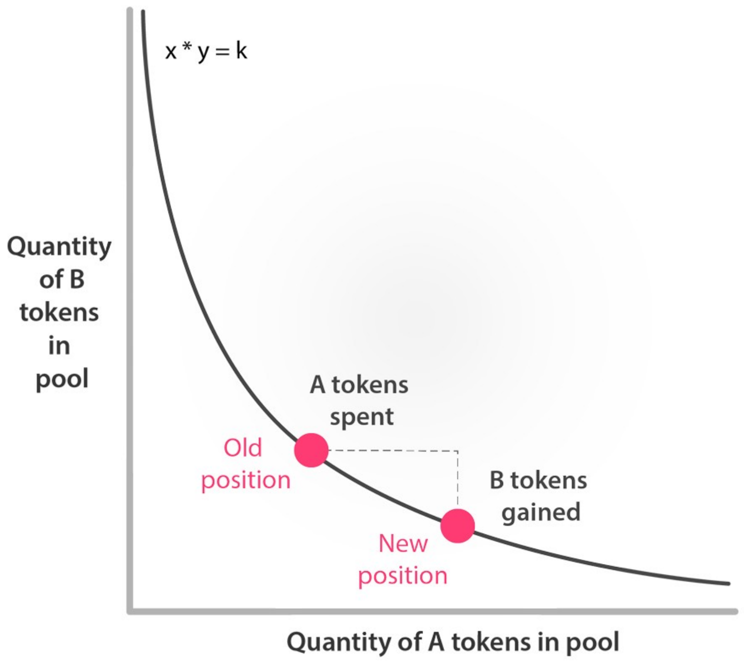
Now let's dive into some of the mechanisms of Uniswap V2 and further understand the protocol's capabilities.

. . .

How Uniswap V2 Works

Each pair of tokens has its own pool initialised via the Factory contract, and initial deposits are made to the pool in order to provide liquidity.

product formula, token values in a particular pair are calculated based on supply and demand, where the value moves along a curve of the formula:



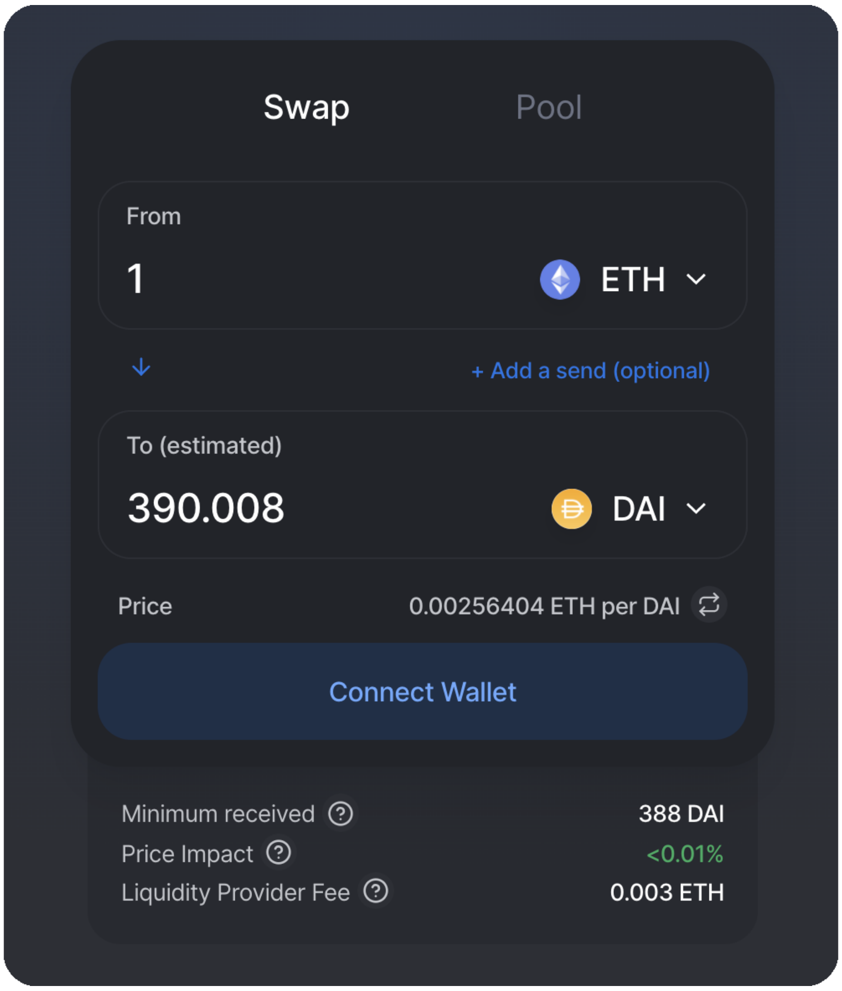
The constant product formula that determines exchange rates on Uniswap V2.

Exchange rates are automated based on the simple formula: `x * y = k`. The corresponding curve represents all possible token values, and each token pair will have their own curve that will regulate the current state of their exchange rates.

If `B Token` is in huge demand and quantities dry up then the price will increase exponentially, and move up the left side of the curve as a result. If on the other hand `B Token` has ample supply and no demand relative to `A Token`, the price will level out on the right side of the curve. Note that this supply-demand balance is based relative to `A Token`, so the same `B Token` paired with different `A Tokens` will represent separate quantities and price ratios.

As a result of these mechanics, Uniswap (and other decentralised exchanges relying on the constant product formula) rely on arbitrage trades to keep the value of the token in line with the rest of the market. Essentially, these protocols still require an external trading ecosystem to keep token values in check. Exchange rates of each token pair will constantly be updated to match the market prices — and this provides a huge opportunity for traders.

for other apps to provide their own interfaces for fetching token pairs and their exchange rates. Exchange rates are applied to native `ETH` in addition to ERC20 tokens:



The exchange rate being applied to the ETH | DAI trading pair within the Uniswap V2 UI.

Now let’s segue into the types of swaps that can be carried out on Uniswap V2. The expanded swap options now available are due to the fact that `ETH` is not used as an intermediary token, also known as “ETH bridging”, as it was in Uniswap V1.

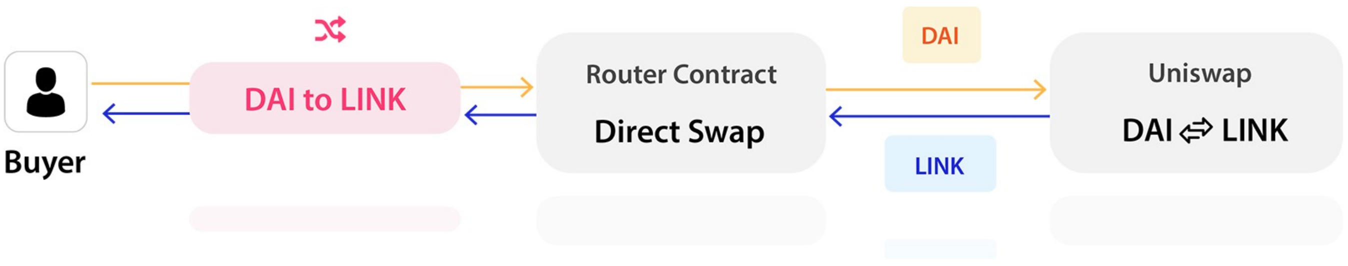
. . .

Uniswap V2 Swap Options

As mentioned above, in Uniswap VI `ETH` was used in every token swap. In a swap from `A Token` to `B Token`, `A Token` firstly had to be converted into `ETH`, and that ETH then converted into `B Token`. This resulted in double transaction fees and double the gas fees as a result.

efficient methods of carrying out trades that have been made available in V2. The omission of ETH as an intermediary token is one of such optimisations.

Removing ETH as a utility token has enabled direct swaps of token pairs, as illustrated in the following diagram:



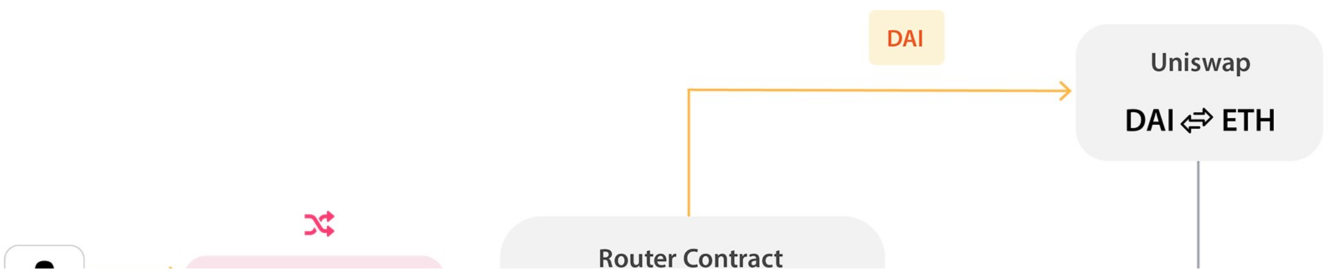
A direct swap between two ERC20 tokens.

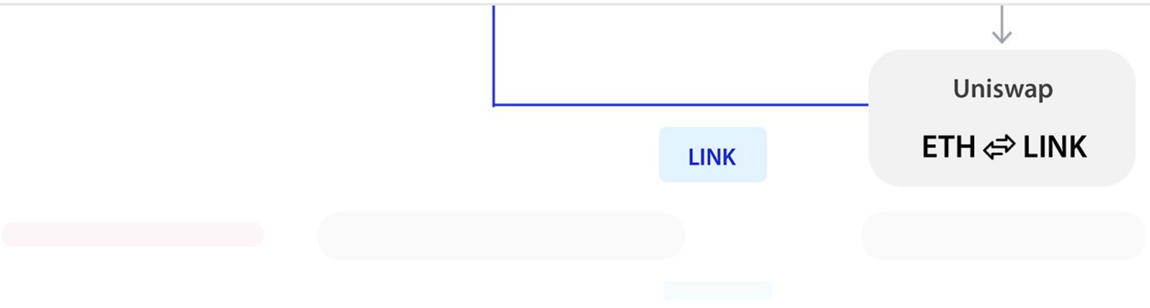
This seems reasonable and intuitive, and was perhaps the way V1 should have worked. The `swapExactTokensForTokens` and `swapTokensForExactTokens` methods on the Router contract can be called to make such a transaction.

The `Exact` terminology used in these method names represent the token you wish to trade to. In a trade from `DAI` to `ETH` where you require a particular amount of `ETH` in return, you would use the `swapTokensForExactTokens`. On the other hand, if you wanted to trade an exact amount of `DAI` for the corresponding `ETH` value, you would use `swapExactTokensForTokens`. This convention is used throughout Uniswap V2's smart contracts.

In addition to the direct swaps, users still have the option to swap between two tokens with `ETH` as the intermediary token. This becomes useful when there exists no pool for the input and output token, but does exist a pool between `ETH` and the two tokens.

The resulting swap-flow is the following in the case `DAI` and `LINK` are being exchanged through `ETH`:



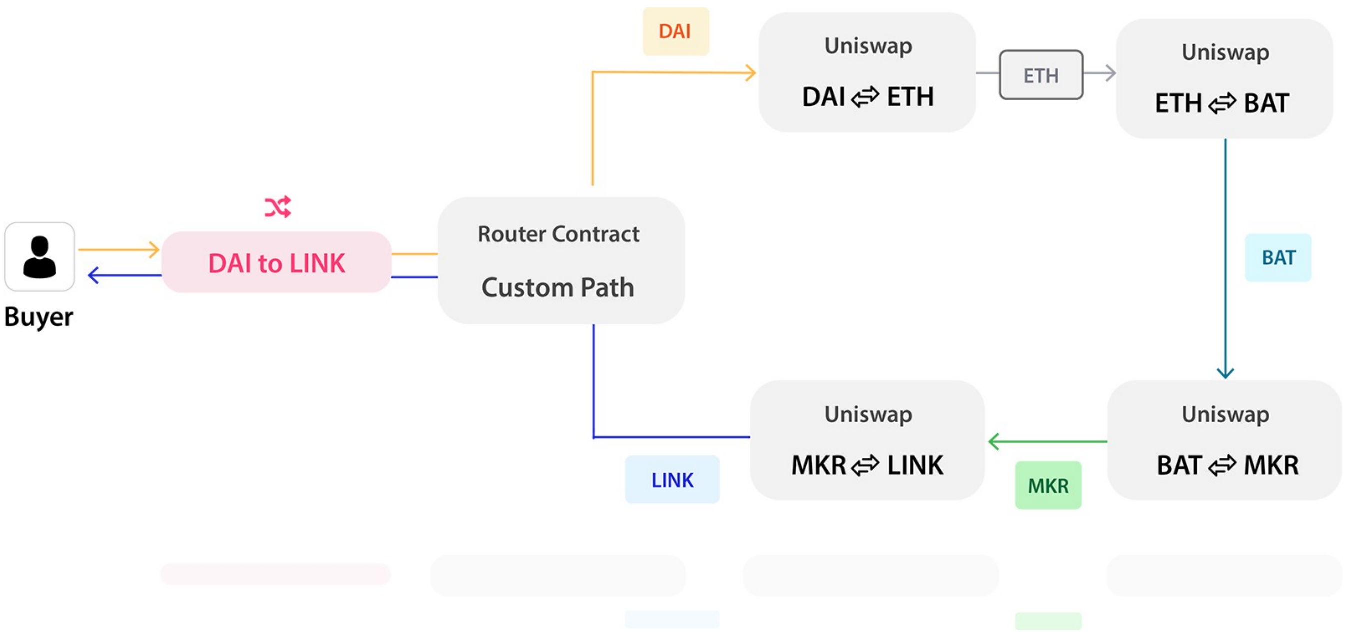


An indirect swap where ETH is the intermediary token (akin to Uniswap V1).

The corresponding Solidity methods that allow such transactions are `swapExactETHForTokens` and `swapETHForExactTokens`, along with `swapTokensForExactETH` and `swapExactTokensForETH`.

The final means of swapping tokens is one that routes a swap to multiple ERC20 tokens, or "arbitrary pairs of ERC20 tokens" as the white-paper terms it, before arriving at your desired output token. Of course, the native ETH token could be present in an arbitrary pair too.

Consider the following illustration that moves `DAI` value through a range of tokens before arriving at the desired `LINK` output token:



Routing funds through a range of tokens to exchange a token with no direct pair.

This method is useful if there are no pools between your direct tokens, and has been made possible with the omission of ETH-bridging that was covered earlier.

end applications must code the functionality manually and make multiple transaction calls to Uniswap instead.

This use case probably won't be used a lot — after passing value through 1 or 2 intermediary tokens, gas fees would make additional swaps uneconomic. In addition, there may be cheaper token swaps available on centralised exchanges. With all this said, such a capability could become more widely used once Ethereum 2.0's scaling features and more efficient Proof of Stake consensus are bought to the Mainnet, making multi-transaction swaps a more viable prospect.

. . .

Price Oracle

Uniswap's price oracle mechanism allows developers to calculate an average token price based on that token's price movement over a number of blocks, that also represent a period of time via their timestamps.

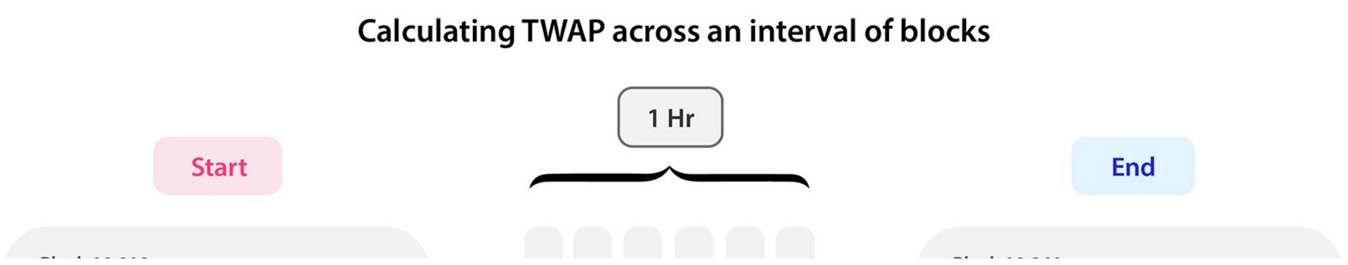
This period of time accumulated could be the last hour, 24 hours, or more.

Although Uniswap provides token prices, it does not store any historical values on-chain. Instead, it is the Dapp developer's responsibility to accumulate prices over a period of time to work out the average price over that time.

These prices are termed "Time Weighted Average Prices", or TWAPS.

The idea is to calculate the average price over a period of blocks by dividing the cumulative price (the price of the token at each block of the duration) by the timestamp duration (the end-of-duration timestamp minus the start-of-duration timestamp).

The following illustration summarises this calculation:



$$\text{TWAP} = \frac{\text{Price Cumulative}_{\text{End}} - \text{Price Cumulative}_{\text{Start}}}{\text{Timestamp}_{\text{End}} - \text{Timestamp}_{\text{Start}}} = \frac{49{,}291 - 11{,}300}{1{,}583{,}535{,}828 - 1{,}583{,}532{,}228} = 8.6$$

A TWAP is calculated by accumulating prices over a duration of blocks over the timestamp duration.

TWAPs are reliable and reflect the value of a token (not forgetting, based on a particular token pair) over time. This approach provides protection from flash crashes or wild price movements, activity that is not uncommon in Cryptocurrency trading. Providing a time-weighted price reflects a more accurate representation of the token in the event there is volatility in the market.

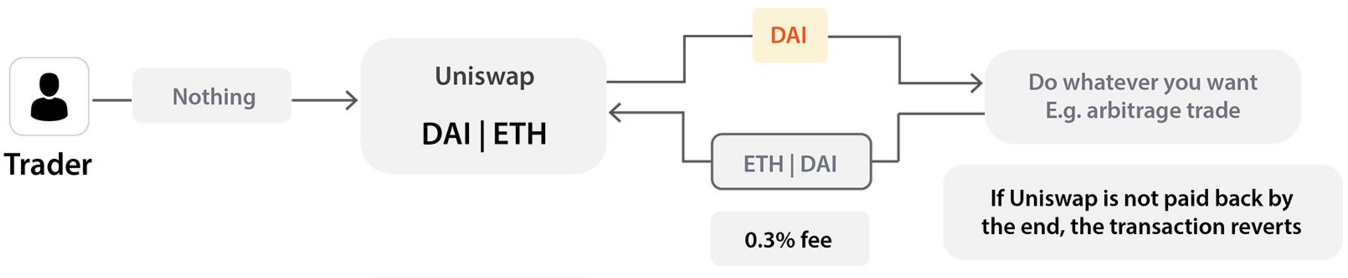
Developers are free to leverage price oracles if needed for their service, that they can retrieve via the Uniswap JavaScript SDK that will be visited further down.

. . .

Flash Swapping

Flash Swaps are transactions consisting of a multi-stage process of borrowing tokens from a Uniswap token pool, doing something with those tokens, and paying back those tokens — all in that single transaction. If any stage of the transaction fails, all state changes are reverted and tokens remain in their corresponding Uniswap pool.

Consider the following illustration showing the buyer can execute the flash-swap transaction with *no held tokens*. This is because that token value is guaranteed to return to the pool either by the transaction failing or the buyer paying back the tokens borrowed in the same transaction:



walks away with the profit made from the arbitrage trade every time.

Another use case is utilising a Uniswap pool to settle a Maker Vault, where you would pay back debt and withdraw your collateralised ETH (or other collateral token) from that vault to pay back the Uniswap pool. This entails less gas than using your own funds to achieve the same task.

Flash swapping is a relatively new capability of Uniswap, but we can expect more Dapps to integrate the feature in the near future.

In the use case of trading bots, flash swapping could also be leveraged to automatically perform arbitrage trading. The bot would not need funds in order to execute the trade, being only required to identify the arbitrage opportunities and execute the flash-swap transaction.

. . .

New 0.05% Protocol Fee and Governance

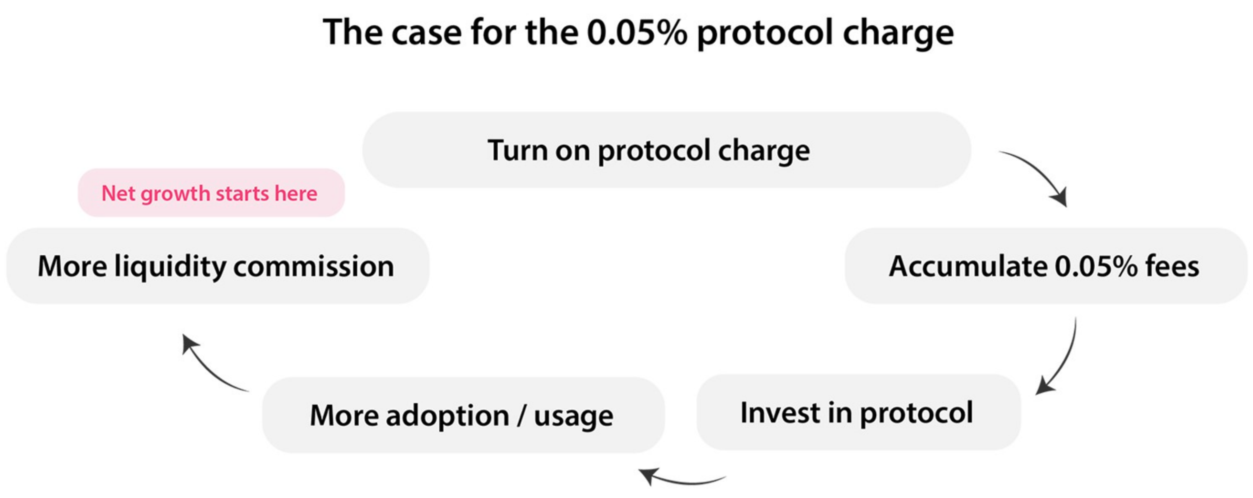
Uniswap V2 introduces an additional fee to the protocol in the form of a "protocol fee", that when turned on routes 0.05% of the transaction value to a Uniswap owned address. This 0.05% is deducted from the standard 0.3% fee on the exchange, so liquidity providers will be the ones losing out in the event this charge is turned on.

The protocol fee is currently turned off, and there have been no public announcements from Uniswap to suggest that it will be turned on in the near future.

This addition may seem like a questionable decision, but Uniswap developers have deemed it a necessary feature to support Uniswap's growth and sustainability for the long term. This is backed up by the fact that in the event they wish to turn on this fee, a governance mechanism is in place where the community would need to vote on whether to switch the mechanism on, or keep it off.

Although a community consensus will need to be reached for this feature to be activated, doing so would give the Uniswap organisation a huge financial and therefore competitive advantage as fees are routed to their wallets.

benefit liquidity providers. This theoretically makes up the loss initially introduced by the charge. This idea is summarised by the following illustration:



The positive outcome of turning on the 0.05% protocol charge

Not much detail has been published about the governance mechanism that'll determine whether the protocol charge is turned on, but one can assume an ERC20 token would be leveraged to submit votes, similar to Maker's governance and voting protocols.

To manage the protocol charge mechanism, the Uniswap Factory contract contains `feeTo` *and* `feeToSetter` *methods to set the protocol charge wallet address and the account able to set the* `feeTo` *value respectively.*

. . .

JavaScript SDK

This piece deliberately avoids becoming too technical, but it is worth mentioning the existence of the Uniswap JavaScript SDK. The open source library provides JavaScript APIs for NodeJS, JavaScript frameworks and any browser-based app to integrate with Uniswap.

This has already led to third parties integrating Uniswap directly into their products and services.

The JavaScript SDK documentation covers APIs for fetching token pair data and pair addresses, in addition to APIs for initiating trades and fetching pricing, that utilises the price oracle function discussed above.

Note that trades cannot be automatically executed, instead these APIs will *prepare* a trade, taking into consideration the mid market price and the minimum amount of output token you're willing to settle (also known as slippage).

Once the trade is configured the user must initiate and sign a transaction with wallet software such as MetaMask.

More articles dedicated to developing Dapps with the Uniswap JavaScript SDK will be added here when they are published.

A note on Token Lists

If you are wondering how the token list is decided on Uniswap.exchange, this is a manual process, and the Uniswap organisation make the call on which tokens to add to it. Even though a pool may exist on-chain for a particular ERC20, it is not a guarantee that it will appear in the list of tokens on the official Uniswap site.

If a token owner wishes to request their token be added to the default list of tokens on uniswap.exchange, a GitHub issue needs to be filed within the Default Token List repository.

There is a function on Uniswap.exchange that allows you to input an ERC20 token address in the search field for the token, that will successfully add it to your list if it exists. This allows the token to be used on the Dapp, but the token is not persisted to the list for others to use, and therefore does not add exposure to the token.

Developers can also create their own token lists to populate their own integration of Uniswap by following a simple JSON structure. More detail on doing so can be found on Uniswap's Token Lists repository on GitHub.

. . .

In Summary

weighted price oracle and flash swapping capabilities are the major enhancements, with under-the-hood improvements such as the move to Solidity-based smart contracts that leverage the latest opcodes for a more efficient execution.

Uniswap V2 remains a trustless, decentralised exchange that lives on the Ethereum blockchain that cannot be tampered with unless the fundamental Ethereum protocol is successfully attacked. This has not happened to date and will unlikely happen with the amount of miners and validators currently in operation.

The Medalla testnet of ETH 2 launched with 20,000 validators — I expect this figure to be much higher for the mainnet launch where validators will be incentivised with real value.

On the other hand, Uniswap V2 still remains a less than perfect implementation of a token exchange. Gas fees are currently the highest they've been in history (at the time of writing), making each Uniswap transaction expensive. There are no live order books with Uniswap, resulting in traders still relying on centralised exchanges for executing trading strategies. As mentioned earlier, centralised exchanges still play an irreplaceable role in balancing Uniswap exchange rates, that rely on arbitrage trades to stay synced with the mean market price.

Uniswap V2 is a positive move in the right direction, but there are still major challenges ahead if decentralised exchanges aspire to totally replace centralised exchanges. Saying this, the sentiment throughout the Cryptocurrency community and Ethereum in-particular is extremely positive — one can expect with a high degree of confidence that the DeFi space will not be slowing down in terms of technical capability and adoption in the short to mid term.

Useful Links

- Uniswap.exchange: The default exchange app that can also be used as a front-end reference implementation.
- Uniswap.org: The official Uniswap website homepage.
- Uniswap.Info: Uniswap usage statistics.
- Smart Contract Documentation: Dive deeper into the smart contract methods and how to integrate Uniswap into your app.

- Uniswap on GitHub: Explore the current state of code, issues, and progress beta releases, and technical discussion.

Ethereum Blockchain Software Engineering Programming Development