David vs. Goliath in the TON Blockchain: A Technical Comparison of DeDust and STON.fi

Ramil Amirov

January 2025

Abstract

This paper presents a technical comparison of two decentralized exchanges (DEXs) on the TON blockchain: DeDust and STON.fi. Through an analysis of architecture, tokenomics, user experience, and security, we aim to provide a comprehensive evaluation of these platforms.

The Introduction includes an overview of the TON blockchain's asynchronous design and its implications for token and smart contract interactions. This foundational context is necessary for understanding the technical challenges and design decisions of DEXs operating in the TON ecosystem. By focusing on user-centric metrics such as average swap times, fees, and APY, this research seeks to highlight the practical differences between the two protocols, providing valuable insights for both users and developers.

1 Introduction or TON architecture

You cannot compare DEXs on the TON network without first understanding the nuances of the blockchain. I will do my best to explain this briefly, but if any questions remain, I recommend looking deeper into these topic before continuing with this article.

The main difference between TON and EVM-based blockchains is that TON is an asynchronous blockchain. To illustrate this, let's compare the flow of transactions in Ethereum and TON. First, let's define a "transaction" in this context as one or more method calls across one or more smart contracts.

In Ethereum, once a transaction starts executing, no other transactions can execute at the same time. Essentially, transactions form a queue, where each one must finish before the next begins. Thus, if our transaction calls two different contracts, those calls will be executed one after the other, without interleaving calls from other transactions in between. This approach is straightforward and

has certain advantages, but it does not scale well.

To address this limitation, TON was designed as an asynchronous blockchain. There is no strict transaction queue and no fixed order of execution. Returning to our previous example, in TON, any state changes could occur in between the two contract calls in the same transaction. This approach adds new complexities and potential pitfalls, but it solves the scalability issue by allowing many transactions to run in parallel.

First, it's important to understand the concept of asynchronicity in TON. When the TON network experiences heavy load, it splits into two parallel chains. All smart contracts from the original chain are then distributed across these two new chains. If one of those chains becomes overloaded, it will split again, and so on. Once the load decreases, these chains merge back together into fewer chains.

Now, let's recall what a token is in an EVM-based blockchain. Simply put, a token is usually just a single contract holding balances for all users and containing transfer methods. In synchronous blockchains like Ethereum, this is perfectly fine. Even if many users are transferring the same token, they simply call the transfer method on that contract, and everything continues to function as expected.

In TON, however, if we tried to implement a token as a single contract with all the balances and transfer logic, we would put a tremendous load on that one address. The TON blockchain would detect this load and attempt to split. But since all the calls related to that contract would go to the same shardchain (the new chain created in the splitting process), the load would remain concentrated in that shardchain. Repeated splitting would not alleviate the situation, leading that shardchain to either slow down significantly or even become unable to produce blocks efficiently. As a result, all token transfers would become extremely slow.

2 Architecture

First, let's explore the architecture of the STON.fi protocol. Its structure is similar to that of many familiar decentralized exchanges (DEXs) built on Ethereum Virtual Machine (EVM) blockchains. In this setup, the DEX is composed of a liquidity pool manager (router) and the liquidity pools themselves. All trades within the DEX are routed through this manager, which, given the low load on the blockchain, allows users to perform swaps quickly and efficiently.

But as I mentioned in the *Introduction*, having a single contract that processes a large volume of messages is a bad idea in TON. STON.fi v1 experienced precisely this issue. In the second version of their protocol, STON.fi introduced multiple routers across different shards. When a new liquidity pool is created,

the protocol selects a router in such a way that the router, its jetton wallets, and the pool itself all reside within the same shard. This optimization allows swaps to execute faster since multiple transaction steps can be completed within a single block.[2]

However, this design comes at the cost of increased complexity for users and developers. STON.fi's architecture dynamically distributes pools and routers across shards. While this makes certain performance optimizations possible, it also introduces additional considerations in terms of usability and liquidity fragmentation.

It is also worth noting a significant drawback in STON.fi's architecture: not only do all DEX-related transactions pass through a single router contract, but each transaction typically goes through it at least twice. This design choice creates a "bottleneck" situation, potentially impacting network efficiency and scalability.

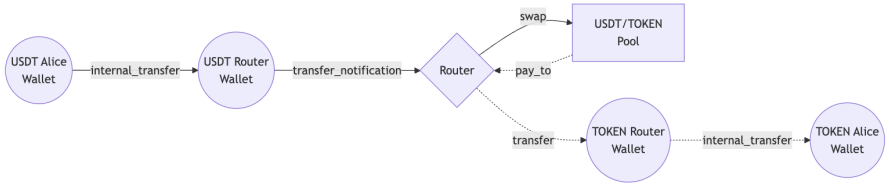

Figure 1: STON.fi swap flow

The main question is: how does one determine which DEX instance a given jetton should be assigned to? STON.fi solves this issue by simulating the creation of a pool on their backend, selecting the best option from those they manage to calculate.[2]

On the other hand, the architecture of DeDust is fundamentally different. Unlike STON.fi, it does not rely on a single entry point, and overall, its DEX logic aligns more closely with how the TON blockchain is designed to operate. DeDust has created separate vaults for each type of asset (including the native TON currency, which it supports directly, without wrapping). In contrast, STON.fi does not natively handle TON and instead wraps it into a pTON token. This is likely not the optimal solution, as it introduces issues with transaction parsing and increases transaction costs.

Consequently, a user only interacts with these vaults, either by depositing tokens or withdrawing them. With this approach, the architecture is more resilient for an asynchronous blockchain. It is also worth mentioning that during any user transaction, interaction with any vault occurs only once, which is more efficient than STON.fi's approach (where the router processes 2 or more mes-

sages per transaction). This is a better solution, as it reduces the pressure on the shardchain, resulting in improved scalability and performance.

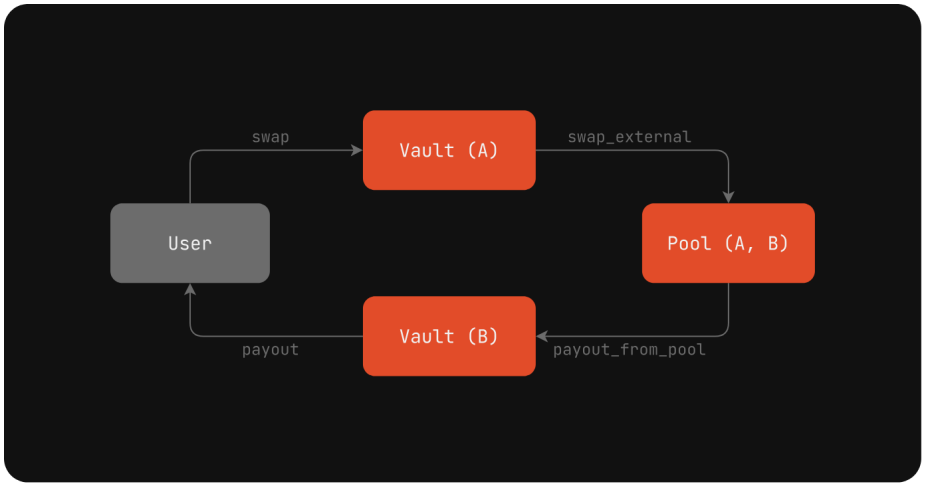

Figure 2: DeDust swap flow

However, an observant reader may notice a potential issue. The TON blockchain frequently encounters challenges during airdrop events, where a large number of users simultaneously try to sell their tokens and send numerous transactions to the DEX. In such cases, the vault of the token being sold becomes overloaded, resulting in transaction queues. This, in turn, slows down swaps for that token and, in the worst-case scenario, completely overwhelms the shard, throwing the blockchain into chaos.

Currently, this problem has been mitigated at the blockchain level by introducing a dispatch queue mechanism.[16] This update reduces the impact of a single contract on the shard, preventing such bottlenecks, and improving overall stability.

To conclude the analysis of architecture, I would like to reference a post by Anatoly Makosov[10], one of the core developers of the TON blockchain, written a month before the release of STON.fi's v2. Despite its timing, I believe the insights shared in this post remain relevant. In it, Makosov discusses the challenges faced by DEXs and concludes that DeDust's architecture is the superior approach. Additionally, it is worth noting that the DEX aggregator *swap.coffee* is currently building its own DEX, and its architecture appears to be more similar to DeDust's approach rather than STON.fi's.[21] This trend may influence the community's perception of which architectural model is best suited for TON's asynchronous environment.

3 Security

A significant drawback of DeDust is its unverified smart contracts and the lack of any publicly available code for the community. This naturally raises concerns among users. Especially considering that DeDust has no technical restrictions preventing them from updating their smart contracts and potentially withdrawing all user funds[1].

Another important point is that DeDust had only two developers[3], which increases the likelihood of errors compared to STON.fi, where the technical team is much larger and a significant amount of code reviews are conducted[4]. Users may feel more comfortable investing their funds in the more academic approach of STON.fi, as opposed to the fast-paced development of DeDust. However, the technical team of DeDust has recently grown, while, according to some sources, the DEX core of STON.fi was written by a single developer. This might raise concerns for some readers as well.

However, DeDust has passed two audits from prominent auditing companies, CertiK[5] and Zellic[1], which is obviously a positive aspect. Despite the unfinished status of the CertiK audit, the DeDust protocol team has already implemented all the recommended changes from the preliminary report[1].

Let's continue with STON.fi: it has a setup where only the router contract can be upgraded[7], which seems like a security advantage. However, user funds are still stored in the router contract[8], meaning there are no strict technical limitations preventing the STON.fi team from withdrawing those funds.

On the other hand, STON.fi gains points for having undergone three audits by CertiK[6] and for publicly sharing the code of the first version of its protocol[9]. Although the second version's code is currently neither verified nor published on GitHub, this still provides more transparency compared to the total lack of publicly available information from DeDust.

4 User Experience

Let's begin by comparing the conditions offered by each DEX. Below is a table summarizing the standard swap fees:

DEX	**Standard Swap Fees**
DeDust	0.25% $\rightarrow$ 0.2% to liquidity providers, 0.05% to DUST staking
STON.fi	0.3% $\rightarrow$ 0.2% to liquidity providers, 0.1% to protocol revenue

Table 1: Comparison of Swap Fees Between DeDust and STON.fi [17][11]

NB: Protocols occasionally adjust their fees in specific pools, for example,

at the request of token creators or under certain conditions. At the time of writing, DeDust had temporarily reduced fees in its TON/USDT pool to minimize the spread during a farming period.[1]

After analyzing the fees offered by the protocols, we can conclude that the situation is the same for liquidity providers, as both protocols allocate 0.2% of the fees to them. However, users are charged higher fees on STON.fi, which benefits liquidity providers but is clearly a disadvantage for users.

Let's now analyze some statistics. For example, we'll compare the tsTON/USDT pool on both DEXs to determine which one offers a better return for an average user. I created a dashboard on Dune[18] to visualize APY data, excluding any farming rewards or grants to calculate the pure, natural APY that liquidity providers (LPs) would earn solely from swap fees.

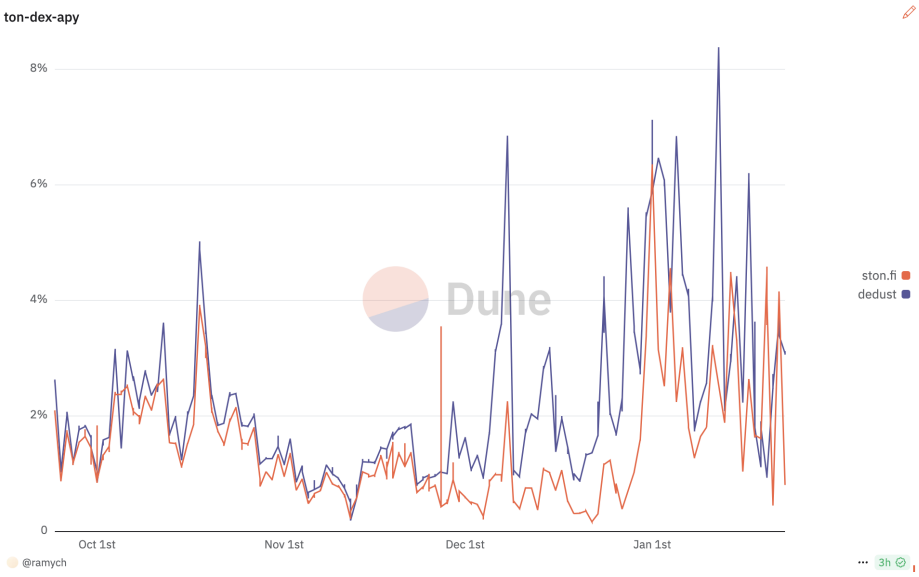

Figure 3: APY Comparison for the tsTON/USDT Pool on DeDust and STON.fi

As we can see from the chart, the APY on DeDust has been higher for the majority of the time.

4.1 Liquidity Incentive Mechanisms

Another key aspect of user experience is how each DEX incentivizes liquidity providers. STON.fi implements a farming mechanism directly through smart contracts, ensuring that rewards are automatically distributed without manual intervention.[2] DeDust, on the other hand, uses a system called boosts, where

rewards are distributed via a cumulative merkle tree.

DeDust's approach allows for permissionless boosts, whereas STON.fi cannot support this due to the storage limitations of its smart contracts.[2] In their case, reward pools are stored in a contract, and STON.fi wouldn't want people to spam the reward pools with worthless tokens, as this would clog up the contract's storage.[2]

4.2 Average Swap Time

Let's now examine the average swap time on these exchanges. How well do our theoretical assumptions from the architecture section align with reality?

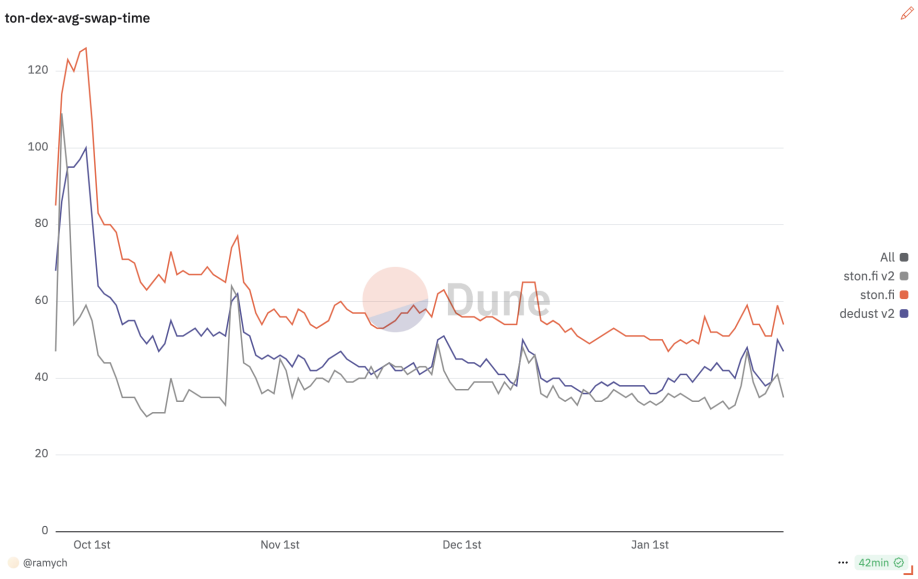

Figure 4: Comparison of Average Swap Time on DeDust and STON.fi

As we can observe on Figure 4, despite my concerns regarding the architecture of STON.fi's contracts, their optimizations in the second version have significantly improved transaction speeds. Thanks to their significant reduction in the number of cross-chain messages, swaps under normal conditions are carried out much faster than on DeDust.[2] Currently, DeDust processes swaps faster than the first version of STON.fi but slower than the second version.

However, my earlier statements about STON.fi's performance issues under high blockchain load are somewhat validated. While there have not been any extreme loads since the launch of STON.fi v2, even during periods of increased transaction volume, we can see noticeable slowdowns in swap times on STON.fi v2. However, the number of swaps on DeDust and STON.fi differ. The STON.fi team claims that during periods of a sharp increase in swap times, the number of swaps on DeDust was ten times lower[2]. To verify this information, I have compiled some analytics on Dune, where interested readers can examine the issue in more detail.[18]

Additionally, an interested reader can explore various statistics provided by TON Foundation[19]. I chose not to include them in this article, as they do not reflect the direct impact on users, unlike the graphs presented here.

5 Tokenomics

5.1 DeDust (DUST Token)

Formerly known as SCALE, the DUST token uses a fairly standard revenue-sharing model: 20% of all exchange fees are distributed to stakers of the token[11]. This model has proven effective in other DeFi ecosystems over time. At the moment, there is no DAO or governance structure in place, although the DeDust team claims that new mechanics are on the horizon.[1]

It is also worth noting the initial allocation of DUST. DeDust does not have external investors, which often reduces the risk of large sell-offs that can depress token prices. However, the team currently controls 69% of the remaining supply (considering that 20.54% of the tokens have been burned), which raises concerns about centralization and the potential impact on market dynamics.[11]

However, potential investors should keep in mind that a revenue-sharing model may attract the attention of the SEC. The Uniswap DAO has been considering implementing a protocol fee for several years now, but due to the risk that its token could be classified as a security, they have not yet moved forward. Moreover, a recent attempts to activate the protocol fee once again drew the SEC's attention.[23]

5.2 STON.fi (STON, GEMSTON, and ARKENSTON Tokens)

STON.fi takes a different approach by issuing three tokens: STON, GEMSTON, and ARKENSTON. STON can be staked to earn two other tokens:

- **ARKENSTON** – NFT, which represents STON stake, which provides voting power in the STON.fi DAO.
- **GEMSTON** – currently without a direct utility, though the STON.fi team hopes the community will develop use cases over time.

This creates a multi-token ecosystem where STON serves as the primary staking asset, and ARKENSTON governs protocol decisions. Meanwhile, GEMSTON remains a flexible token that may take on additional functions in the future.[12] It is also worth noting that since STON.fi was launched, there have been no governance proposals or votes using the STON.fi DAO. As a result, the practical application of the entire STON.fi token system remains questionable, given the lack of real-world usage. However, the STON.fi team asserts that they are following their whitepaper, albeit with some delay, so both a DAO and token utility are planned.[2]

6 Future

6.1 DeDust

Looking ahead, DeDust appears to be moving toward a new version of its DEX that draws inspiration from Uniswap v4 (an industry standard protocol in crypto). Recently, Nick (the founder of DeDust) announced DeDust X[3], which will allow users to add their own code to DEX contracts, reminiscent of the "hooks" concept introduced in Uniswap v4.

6.2 STON.fi

STON.fi, on the other hand, is taking a different path. They are currently developing OMNISTON[14], which aims to integrate on-chain RFQ resolvers[15] with a traditional AMM DEX. This approach is intended to offer the best possible price to end users. At the time of writing, a beta version of the protocol is already available for anyone to try.[20] However, industry professionals have raised questions about how effectively STON.fi will be able to implement this feature in practice[13].

Additionally, STON.fi has placed a strong focus on building a cross-chain swap service[12], though there is currently no information regarding when this functionality will go into production.

7 Conclusion

This study highlights two contrasting approaches on the TON blockchain: DeDust's decentralized vault design, which aligns well with TON's asynchronous nature, and STON.fi's router-based system, which has seen significant optimization in its second version. DeDust addresses scalability by avoiding a single entry point, and it offers what appears to be a simpler and more transparent tokenomics model—20% of exchange fees are distributed to DUST stakers, and the project lacks external investors who might exert downward pressure on the token price. Meanwhile, STON.fi has introduced multiple tokens (STON, GEMSTON, and ARKENSTON) to power a DAO and broader ecosystem, yet

no governance proposals have been held so far, raising questions about the real-world utility of these tokens.

On the security side, both DEXs have unverified and upgradeable contracts, enabling their respective teams to withdraw user liquidity if they chose to. This is highly unusual for major DeFi projects on other blockchains, where code verification and trustless smart contracts are the norm. While DeDust and STON.fi have each undergone some level of auditing, the lack of fully verified contracts remains a concern.

From a user experience perspective, DeDust has often shown higher APY returns on comparable pools (e.g., tsTON/USDT) and simpler swap flows, whereas STON.fi's architecture, although improved in version two, may suffer performance bottlenecks under high load. Nonetheless, STON.fi's broader vision includes cross-chain functionality and OMNISTON, potentially appealing to those seeking advanced DeFi features.

In conclusion, neither DEX is without flaws. DeDust's architecture and tokenomics may be more appealing for users who value simplicity, while STON.fi's feature-rich ecosystem could attract those looking for a multi-token approach and potential cross-chain opportunities. This comparison underscores the importance of understanding TON's asynchronous design when evaluating DEX performance, as well as the critical need for greater transparency, full contract verification, and decentralization in the TON ecosystem.

References

[1] Personal conversation with Nick Nekilov, founder of DeDust.

[2] Personal conversation with Slavik Baranov, co-founder of STON.fi.

[3] Nick Nekilov's speech at The Gateway 2024. `https://www.youtube.com/watch?v=LcosaYmfQe8`

[4] Interview with the CMO of STON.fi. `https://youtu.be/YVCpaMqbi08?t=5389`

[5] CertiK audit status for DeDust. `https://www.certik.org/projects/dedust`

[6] CertiK audit status for STON.fi. `https://skynet.certik.com/projects/ston-fi`

[7] STON.fi documentation page about their architecture. `https://docs.ston.fi/docs/developer-section/architecture#router`

[8] The example of STON.fi router. https://tonviewer.com/EQCS4UEa5UaJLzOyyKieqQOQ2P9M-7kXpkO5HnP3Bv250cN3?section=tokens

[9] STON.fi v1 implementation. https://github.com/ston-fi/dex-core

[10] A Telegram post discussing DEX architecture issues by one of the TON core developers. https://t.me/anatolii_makosov/43

[11] DeDust documentation page about their token https://help.dedust.io/dedust/welcome-to-dedust.io/about-dust

[12] STON.fi whitepaper https://static.ston.fi/whitepaper.pdf

[13] Interview with the CTO of swap.coffee. https://youtu.be/14_s5cjMHAw?t=2891

[14] STON.fi documentation page about OMNISTON. https://docs.ston.fi/docs/developer-section/omniston

[15] Article about RFQ from the Bebop. https://medium.com/bebop-dex/wtf-is-rfq-on-chain-19560e00058b

[16] TEP description about dispatch queue https://github.com/ton-blockchain/TEPs/blob/master/text/0160-dispatch-queue.md

[17] STON.fi documentation page about their fees. https://docs.ston.fi/docs/user-section/fees

[18] My Dune dashboard https://dune.com/ramych/ton-dex-comparison

[19] TON Dex TVL & LP Dune dashboard by TON Foundation https://dune.com/ton_foundation/dex-tvl-stats

[20] OMNISTONE demo app https://omniston.ston.fi/

[21] swap.coffee documentation page about their DEX's concepts https://docs.swap.coffee/technical-guides/dex/concepts

[22] DeDust post about closure of built-in DeDust referral revenue sharing program https://t.me/dedust_dev/7

[23] Article about Uniswap's problems with its token and SEC https://bitpushnews.medium.com/uniswap-sued-by-sec-what-does-it-mean-for-the-future-of-defi-f4ccc5f4a76a