Solana Fees in Theory and Practice

Introduction

Solana's fee structure is designed to maintain the network's performance while balancing non-uniform shocks in supply and demand. Fees on any blockchain serve the purpose of preventing spam and incentivizing validators. On Solana, some of these fees are dynamically adjusted based on network conditions, allowing the network to more accurately price demand at a given time.

Fees in Solana are a hot topic, with "local fee markets" that give some expressivity for Solana to price blockspace and specific accounts more accurately. The current implementation is far from perfect but does give loose guarantees on ordering on a per-account basis. While Solana is still in its early stages, more stake and activity in the network necessitates deeper discussion and analysis of the first and second-order effects of any in-protocol changes, such as fee model changes.

In this piece, we'll go over fees in theory as well as how they manifest on-chain. While some have criticized Solana for being centralized and having centralizing forces with its stake-weighted design of QoS and Turbine, there has been a clear discrepancy between how this manifests in reality, even over many years. Similarly, we aim to comprehensively analyze how fees manifest via on-chain behavior.

Fees in Theory

Solana's fee system consists of two components: the base fee and the priority fee. Broadly, each fee component ideally serves the following purpose:

- Base fees

: right to utilize the network's resources

- Priority fees

: determine the order in a leader's transaction queue

Base Fee

The base fee, currently set at 0.000005 SOL (5,000 lamports) per signature, forms the foundation of the transaction cost. This is a fee paid by an address in order for the right to use the network's resources. This is a single lump-sum payment paid upfront to the network regardless of the amount of actual resources used to execute the transaction (or if the transaction even executes). Solana transactions request a specified number of compute units (CUs) upfront, and if this number is exceeded, the transaction fails. This means developers currently have little to no financial incentives to minimize compute unit requests.

Priority Fees

Additionally, users can pay a priority fee to expedite their transactions for a higher likelihood of inclusion within a block. This is a non-deterministic guarantee for users to pay for priority. Efforts to improve transaction determinism are underway, with significant scheduler changes expected to land in the 1.18.

As a side note, vote transactions do not have an associated priority fee and are treated differently than standard transactions.

The incentive for validators to include transactions with priority fees exists outside the runtime. Leaders collect 50% of the priority fee for including the transaction within its block, with the other 50% being burned.

Today, most validators (80%+) run unmodified versions of the Solana Labs or Jito-Solana client. This means that these validators are outsourcing "block production" to the default scheduler (some people on Solana refer to "block ordering" as "block production" when it means something entirely different on Ethereum). Some teams have modified client code and implemented a more complex scheduler that allows for more control over ordering flow, enabling some to extract MEV by reordering or sandwiching transactions.

Priority Fee Indeterminism

The current implementation of the scheduler does not guarantee that transactions with higher priority fees will be included in a given block. Instead, it makes a loose guarantee that transactions with priority fees are more likely to be included within a given block. The current implementation of the scheduler enacts 4 execution cores (2 additional cores are reserved for vote transactions).

Each thread operates its own queue, prioritizing packets independently without knowledge of the packets being processed

by the other threads. Each thread continuously cycles from the start to the finish, attempting to lock and execute transactions. When a thread completes its current cycle, it will collect more packets and initiate the cycle again.

As a result, you could be working on a high-priority transaction at the top of one thread's queue while simultaneously, a different thread may be concluding its own queue by processing a transaction involving the same account.

The specific details of the current and future implementation of the scheduler will be explored in a separate piece. Understanding that priority fees only work on intra-thread

(within its own lane), not inter-thread

(between lanes), is enough to understand that the scheduler is far from perfect and exhibits "jitter".

Fees in Practice

Landing Transactions

While fees are a major factor in whether or not a transaction lands, they are not the only determining factor. For example, transactions may not land simply due to the loss of a UDP network packet. During periods of high network activity, validators might be inundated with more transactions than they can manage. While validators have the capability to pass on excess transactions through the tpu_forwards mechanism, they can only handle a finite volume of data, and each transaction can only be relayed a finite number of times (until the blockhash expires). Stake-weighted quality of service mitigates some of these issues for addresses with higher stake, offering reserved bandwidth and increasing the likelihood of transaction inclusion.

Two less commonly discussed reasons for transaction drop-off exist as well. The first involves discrepancies within an RPC pool. One segment of the RPC pool may race ahead of others, creating coordination problems. For instance, if a transaction's recentBlockhash is retrieved from a more updated segment and then submitted to a slower segment, the latter may not recognize the updated blockhash and, as a result, discard the transaction. Developers can spot such issues at the time of submission if they have preflight checks activated in the sendTransaction function.

Another issue arises around temporary network forks. If a validator lags in processing its blocks, transactions could end up on a minority fork that does not become canonical. When a client references a recentBlockhash in their transaction that only exists on this minority fork, and then the network abandons that fork before processing the transaction, the transaction will be dropped because the blockhash can no longer be found.

Priority Fees

In practice, we see evidence that although priority fees are far from perfect, they are working on a macro scale. Transactions that include priority fees are more likely to be included in blocks, with transactions setting higher priority fees enjoying a greater likelihood for inclusion.

Based on Helius RPC data, we see that transactions with priority fees are more likely to land, and when they do, they land quicker across the board:

On January 21, there was a spike in average priority fees due to the mockJUP airdrop, gearing up for the actual JUP airdrop next week. While there were significant changes in demand for blockspace, there was relatively little change felt by actual users regarding the transaction land rate and time.

This UX is primarily supported by the Solana RPC methodgetRecentPrioritizationFees, allowing developers to accurately determine the priority fee to append to a transaction. The endpoint returns a list of priority fees over the last 150 blocks that were used to successfully land at least one transaction with the respective address and input parameters. This provides a snapshot of the minimum required value to set for priority fees and is relatively limited in its usefulness. Alternatively, Helius offers a Priority Fee API that does additional calculations to provide a better priority fee estimate.

While priority fees work somewhat as intended in theory, upcoming scheduler changes in 1.18 will add more determinism for transaction inclusion with scheduler improvements. This should reduce the amount of spam that lands on-chain as the dominant strategy no longer requires spamming the chain for transaction inclusion.

Base Fee

The base fee on Solana is definitively too low, with blocks saturating and not being dynamic, preventing the base fee from reaching a market-clearing price for blockspace. On Ethereum, the dynamic base fee is achieved via EIP-1559's controller mechanism, which looks at recent blocks and targets a 50% utilization rate.

Solana statically prices 5,000 lamports per signature (typically 1 signature per transaction). This means it is an ineffective fee as the base fee does not express any change in demand for blockspace and validator resource usage. This effectively externalizes the priority fee to be a market-based alternative for the priority fee, further congesting the network as validators process additional transactions that were likely never going to be included. Furthermore, the dominant strategy is submitting

a large number of transactions with minimal priority fees for inclusion. This imposes large externalities on the UX for all network participants.

Incentives

RPCs are incentivized to relay the correct information downstream to offer the highest rate of transaction inclusion for minimal costs. Integrating with top-staked validators enables RPCs to have a more accurate view of the current state of the network, as many of Solana's mechanisms are stake-weighted. A symbiotic relationship emerges where validators with significant stakes and integrated RPCs can enhance the efficiency and reliability of transaction processing, potentially creating a feedback loop that further entrenches the position of top-staked validators.

Additionally, RPCs – which are currently treated as zero-staked validators – will themselves become stake-weighted. RPCs themselves can seek to attract stake without partnering with a validator. It is not uncommon for applications themselves to run their own validators for more vertical integration, enabling additional control over the end-user experience and transaction/MEV supply chain.

Despite the economic incentive suggesting a trend toward centralizing stake, Solana has not seen large-scale agglomeration of capital for stake-weighted gains. This could be for a number of reasons:

- Individuals may view maximizing decentralization locally as the dominant strategy for long-term gains for the network, primarily driven by the culture and social layer of Solana.
- Participants on Solana have largely been retail and prosumers and are not as sensitive to returns as professional firms. As the activity level and absolute returns increase, this may incentivize adjustments to the participants' demographics and rate sensitivity.
- Individuals are not well-coordinated in the marketing of their differentiated products.

Conclusion

In this piece, we've described in detail the high-level theory of Solana's fee mechanism and how it impacts the network on-chain. Fees drive incentives, which have large externalities and affect the behavior of all participants on Solana.

Mechanisms, such as the base fee and priority fee in Solana, are not perfect in their current implementation. The base fee is unadjustable and not reflective of the current supply and demand equilibrium. This leads to issues such as network congestion and inefficient resource allocation. Priority fees exhibit a degree of indeterminism due to the current implementation of the scheduler. Future updates, such as the anticipated scheduler changes, promise to bring more determinism and efficiency to transaction processing, potentially reshaping the on-chain behavior we observe today.

New proposals are on the horizon, such as exponential fees for write lock accounts, which aim to price the cost for transactions more accurately by arbitrarily locking access to accounts. Additional discussions are being had around a dynamic base fee mechanism that more accurately prices access to state.

The interaction between fees, validators, and RPCs is a complex web of incentives. Validators and RPCs are incentivized, in theory, to integrate and increase their stake weight, potentially leading to concerns about centralization. However, in reality, Solana has managed to maintain a decentralized set of operators and stake, likely due to its community-driven governance, technical barriers, economic counter-incentives, and the optimization functions currently not primarily driven by returns.