Deterministic Transaction Fees

An improved fee model for blockchain cryptocurrencies.

Document v0.8.0
Revision 20190105

Jordan Mack
jordan.mack@brilliantnotion.com
www.brilliantnotion.com

Abstract

Deterministic Transaction Fees (DTF) is a fee model for blockchain based cryptocurrencies designed to alleviate many of the inherent design limitations present in the fee model which was pioneered by Bitcoin, and currently used by most cryptocurrencies today.

The Bitcoin model limits the size of the blocks and relies on market based competition to prioritize transactions for inclusion in the block. This results in an unpredictable fee schedule, and unpredictable confirmation time. This new model introduces a more reliable way of creating fees that remain market based, but are more predictable in both fee schedule and confirmation time.

Table of Contents

[Abstract](#)

[Table of Contents](#)

[Motivation](#)

[Assumptions](#)

[Goals](#)

[Specification](#)
- [Summary](#)
- [Timeframes](#)
- [Buckets](#)
- [Fee Schedule](#)
- [Timeframe Locking](#)

[Considerations](#)
- [Simultaneous Transaction Broadcast and Block Discovery](#)
- [Inclusion of Underpaid Transactions](#)

Motivation

The fee model which was pioneered by Bitcoin, and currently used by most cryptocurrencies today, is not practical in actual use. There exists only two true states for the block in relation to fees: Not full and full.

When block space is consistently under utilized, there is no incentive to pay any fee higher than the absolute minimum since the mempool will always be fully processed in the next block. The minimum fee is 0 on many chains.

When block space is over utilized, fees grow erratically and exponentially at users compete for the available space. The end result is devastating to the user experience. Fees may far exceed the amount of the transaction, rendering it completely useless for small payments. Confirmation times become largely unpredictable unless the user is among the highest fees paid.

Differences in fee estimators are problematic since an estimation could end up being too low or too high. The user could end up accidentally paying far too much for a transaction, or far too low, resulting in transactions that may not process for days, weeks, or ever.

Artificial block size limits are not sufficient for real world usage, and are damaging to the overall ecosystem. The end motivation of paying higher fees is always to improve confirmation time. Therefore, the fee model should be based on confirmation time directly, not on block size.

Assumptions

The future of blockchains is to incorporate some form of partitioning, sharding, or pruning. This will reduce the disk space requirements per node to a level that is negligible. Therefore, disk space utilization does not need to be a consideration for this model.

Goals

- Fees should never be set to specific fixed amounts. The fees must remain market based.
- The fee schedule must be clear. Users should not have to estimate. Fee amounts should be exact and deterministic.
- Confirmation time should be predictable. Users should not have to guess on when or if their transaction will process.
- Transactions which have paid the proper fees should never go into a limbo state when confirmation time is completely unpredictable.

Specification

Summary

The fee schedule for the next block is determined by analyzing the transactions of recently confirmed blocks. The fees paid (per byte) in the confirmed block are analyzed, and the fee schedule is determined based on the averages found.

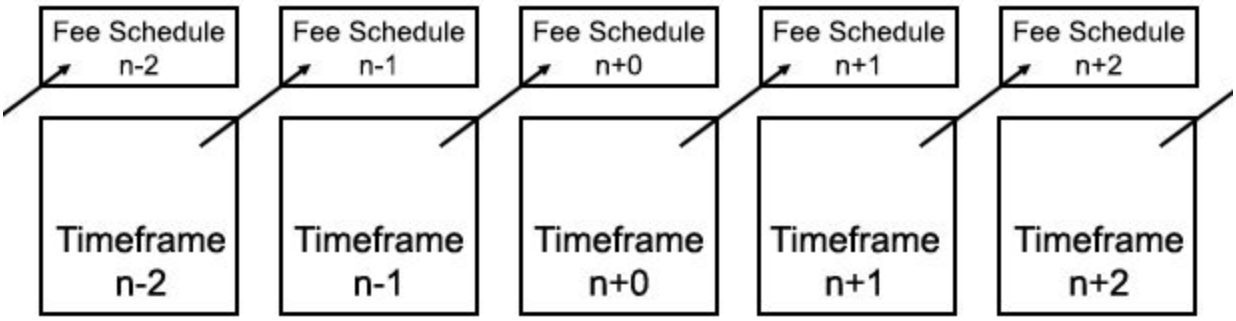
The calculated fee schedule will include exact rates which correspond to priority buckets. The top rate bucket will correspond to the next block (+1). The second highest rate bucket will correspond with the block after next (+2), and so on.

When the proper fee is paid by the user, their transaction is scheduled for inclusion into the blockchain in accordance with the bucket they paid for. If the fee paid is too low, the transaction is rejected by the mempool.

Timeframes

Absolute time does not exist in a blockchain. A timeframe is defined as a range of blocks that is greater or equal to one. For the examples in this document, a timeframe will typically be defined as one block. However, this can optionally be increased in the implementation to reduce pricing volatility.

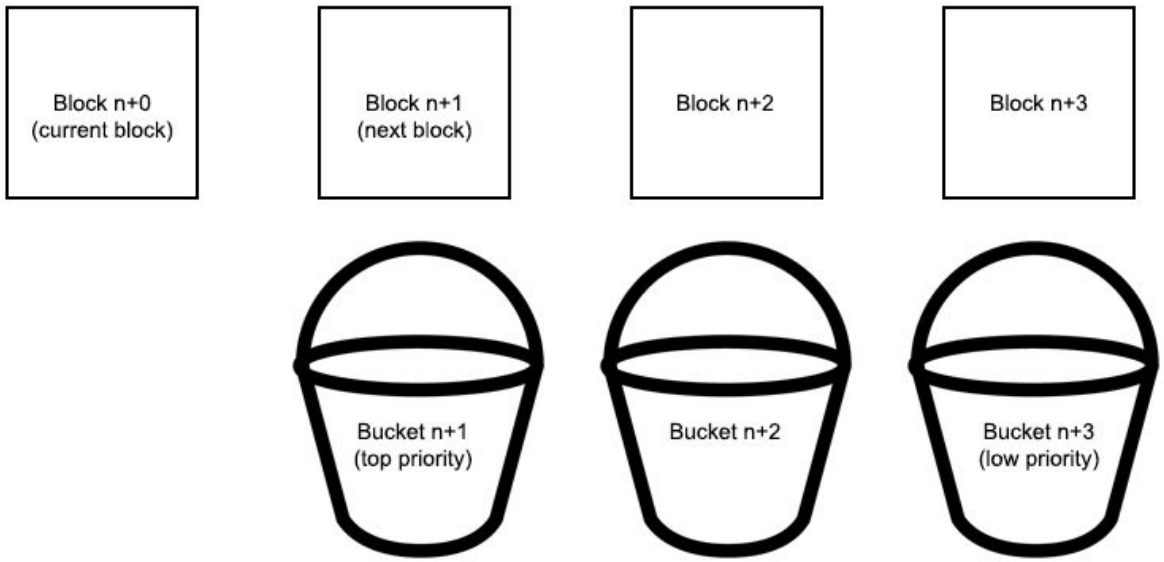
Timeframes are used for the purpose of determining subsequent fee schedules. The active fee schedule is always on a previous timeframe. In the diagram below, the current timeframe is “Timeframe n+0”. The fee schedule for inclusion in the next block is “Fee Schedule n+1”, and this is determined by the previous timeframe.



Buckets

A bucket is a grouping used to schedule pending transactions for inclusion in future blocks. A set of buckets exists for each timeframe, and each bucket represents a priority for inclusion. The highest priority bucket will be included in the next block (n+1). The second highest priority bucket will be included in the block after the next (n+2), and so on.

In the diagram below, buckets are aligned with the blocks the correspond to. The top priority bucket is “Bucket n+1”. This bucket would have the highest fee according to the fee schedule. Transactions that pay the required fee to be included in this bucket will be included in the next block “Block n+1”. If the user does not require the next block, then they can select a lower fee bucket that will be scheduled for inclusion in a later block.



The above example has three buckets, but a more realistic implementation would have a minimum of 6 buckets or more. Each bucket has a corresponding price for inclusion based on the fee schedule. If a transaction fails to pay the minimum fee for the lowest priority bucket, then it is rejected by the mempool.

Fee Schedule

The current fee schedule is based on the previous timeframe, which is, in turn, based on the blocks which have already been included in the blockchain. This means that fees are always exact and deterministic since they are based on the history of the blockchain.

The basis for the fee schedule is the average fee paid per byte in the previous timeframe. This ensures that the fees always adjust based on market conditions. The calculation of fees should

be modified to fit the specifics of the implementation, but in all cases, higher priority buckets should have a higher rate than lower priority tiers.

In the example below, the price per bucket is calculated using the function `ceil(a * f ^ d)` for buckets with a priority higher or equal to the average, and `ceil(a / f ^ d)` for buckets with a priority lower than average.

This assumes 9 bucket levels, and a timeframe average fee of 5 sats/byte, and a constant growth factor of 1.5. The actual pricing function may vary depending on the actual implementation.

Bucket Levels: 9

Timeframe Average: 5 sats/byte

Growth Factor: 1.5

Bucket n+1 (top priority)	ceil(5 * 1.5^4) = 26
Bucket n+2	ceil(5 * 1.5^3) = 17
Bucket n+3	ceil(5 * 1.5^2) = 12
Bucket n+4	ceil(5 * 1.5^1) = 8
Bucket n+5 (average)	ceil(5 / 1.5^0) = 5
Bucket n+6	ceil(5 / 1.5^1) = 4
Bucket n+7	ceil(5 / 1.5^2) = 3
Bucket n+8	ceil(5 / 1.5^3) = 2
Bucket n+9 (lowest priority)	ceil(5 / 1.5^4) = 1

Timeframe Locking

When a transaction is broadcasted it must include the current block hash. If a node receives a transaction broadcast that is timeframe locked with a block that does not match the current block, it should be rejected from the mempool. This ensures that transactions are locked to the current timeframe of the chain, and that stale transactions are rejected.

Considerations

Simultaneous Transaction Broadcast and Block Discovery

If a transaction is broadcast to the network at the same moment that a new block is discovered, it may be rejected by part or all of the network. Transaction broadcast propagation is a time-sensitive action, and therefore detection of a potential issue can be automatically flagged by the client if the two events occur in a short window. The client can automatically assess the situation by querying the mempool of connected nodes. If there is a high possibility the transaction was affected, it can be rectified by submitting a replace by fee transaction utilizing the same UTXO inputs.

Inclusion of Underpaid Transactions

Without a block size restriction, a non-cooperative miner could attempt to include all pending transactions, regardless of what fee was paid and bucket they should have been included into. This is avoided by using timeframe locking. If the transaction is locked to the incorrect timeframe, attempting to it in the wrong block would result in an invalid block which would be rejected by the consensus of the network.