Transaction lifecycle in Monad

Transaction submission

The lifecycle of a transaction starts with a user preparing a signed transaction and submitting it to an RPC node.

Transactions are typically prepared by an application frontend, then presented to the user's wallet for signing. Most wallets make aneth_estimateGas RPC call to populate the gaslimit for this transaction, although the user can also override this in their wallet. The user is also typically asked to choose a gasprice for the transaction, which is a number of NativeTokens per unit of gas.

After the user approves the signing in their wallet, the signed transaction is submitted to an RPC node using theeth_sendTransaction oreth_sendRawTransaction API call.

Of note, in Monad, the gas limit is the sum of the carriage gas and the execution gas. Carriage gas is a constant.

Mempool propagation

The RPC node forwards the pending transaction to other Monad nodes that are participating in consensus. The set of pending transactions is colloquially referred to as the 'mempool'. SeeMempool for additional details on mempool behavior.

For spam prevention reasons, nodes add the transaction to their mempool only if there is sufficient gas in the reserve balance .

Block inclusion

MonadBFT uses a rotating leader mechanism to produce blocks. Each round, a leader assembles a block from the list of pending transactions. After choosing a transaction to be included in the block, the leader decrements the reserve balance with the carriage cost.

Blocks are propagated through the network as discussed inMonadBFT . The quorum certificate (QC) for this block is propagated in the subsequent round of consensus (i.e. is sent out by the next leader). After seeing the QC, voting nodes send each other votes; when a node sees 2/3 of the stake weight vote yes, it finalizes that block.

Once the block is finalized, the transaction has officially "occurred" in the history of the blockchain. Since its order is determined, its truth value (i.e., whether it succeeds or fails, and what the outcome is immediately after that execution) is determined.

Local execution

As soon as a node finalizes a block, it begins executing the transactions from that block. For efficiency reasons, transactions are executedoptimistically in parallel , but it is as if the transactions were executed serially, since results are always committed in the original order.

Querying the outcome

The user can query the result of the transaction by callingeth_getTransactionByHash oreth_getTransactionReceipt on any RPC node. The RPC node will return as soon as execution completes locally on the node.

Previous MonadDb Next Other details Last updated5 months ago