Carriage Cost and Reserve Balance

Motivation

Deferring execution is incredibly powerful, as it allows execution and consensus to occur in parallel, massively expanding the time budget for execution.

An obvious objection is: now that consensus nodes don't have an up-to-date view of state, what prevents them from mistakenly including transactions from accounts that have spent all of their gas? This would create a Denial-of-Service vector.

To defend against this, Monad introduces a cost for a transaction to be carried over the network (the "carriage cost"), maintains a reserve balance for each account which is updated at consensus time, and charges carriage costs against the reserve balance.

Carriage cost

In Monad, there is a charge for having a transaction carried over the network in a block (the "carriage cost"). This is a separate charge from the cost of execution.

The carriage cost is necessary for spam prevention; the cost is minimal, but reflects the cost of utilizing network resources.

It is possible for a transaction to be included in consensus (and charged for carriage) but to have insufficient execution budget relative to the specified gas limit. In that case, at execution time, the transaction will fail but be charged gas up to the point of failure. Note that this is no different from Ethereum: transactions submitted with insufficient Ether in their account will use up Ether and fail. Charging gas up to the point of failure is necessary to prevent execution DOS vectors.

Reserve balance

For each address, nodes maintain two balances:

- areserve balance
- , used to pay for the carriage cost
- theexecution balance
- , used to pay for transaction execution
-

The carriage cost is charged to the reserve balance when the transaction is included in a block (consensus); it is deducted from the execution balance at execution time (double charge), and repaid to the reserve balance after the delay period ofD blocks (10 seconds) passes.

The reserve is effectively a budget for "in-flight" orders; it exists to ensure that only transactions that are paid for are included in blocks. You can think of the reserve balance as decrementing in realtime (i.e. as consensus occurs); although the node's view of the full state is lagged, the reserve balance always reflects up-to-date expenditures.

Target reserve balance

The target reserve balance is a per-account parameter. The default is anticipated to be a large multiple (200x) of the carriage cost, so that users can submit a large number of inflight orders without issue.

Users who anticipate sending a large number of inflight orders from the same EOA can alter the target reserve balance by interacting with an enshrined smart contract. Reserve balance alterations are treated as executions, i.e. they are only reflected in the reserve balance after the delay period passes.

Previous Deferred Execution Next Execution Last updated5 months ago

On this page * Motivation * Carriage cost * Reserve balance * Target reserve balance