Gas and Fees

There are two parties a user pays when submitting a tx:

- the poster, if reimbursable, for L1 resources such as the L1 calldata needed to post the tx
- the network fee account for L2 resources, which include the computation, storage, and other burdens L2 nodes must bare to service the tx

The L1 component is the product of the transaction's estimated contribution to its batch's size — computed using Brotli on the transaction by itself — and the L2's view of the L1 data price, a value which dynamically adjusts over time to ensure the batch-poster is ultimately fairly compensated. For details, seeL1 Pricing .

The L2 component consists of the traditional fees Geth would pay to stakers in a vanilla L1 chain, such as the computation and storage charges applying the state transition function entails. ArbOS charges additional fees for executing its L2-specificprecompiles , whose fees are dynamically priced according to the specific resources used while executing the call.

Gas Price Floor

The L2 gas price on a given Arbitrum chain has a set floor, which can be queried viaArbGasInfo 'sgetMinimumGasPrice method (currently 0.1 gwei on Arbitrum One and 0.01 gwei on Nova).

Estimating Gas

Calling an Arbitrum Node'seth_estimateGas RPC gives a value sufficient to cover the full transaction fee at the given L2 gas price; i.e., the value returned frometh_estimateGas multiplied by the L2 gas price tells you how much total Ether is required for the transaction to succeed. Note that this means that for a given operation, the value returned byeth_estimateGas will change over time (as the L1 calldata price fluctuates.) (See2-D fees andHow to estimate gas in Arbitrumfor more.)

Tips in L2

The sequencer prioritizes transactions on a first-come first-served basis. Because tips do not make sense in this model, they are ignored. Arbitrum users always just pay the basefee regardless of the tip they choose.

Gas Estimating Retryables

When a transaction schedules another, the subsequent transaction's executionwill be included when estimating gas via the node's RPC. A transaction's gas estimate, then, can only be found if all the transactions succeed at a given gas limit. This is especially important when working with retryables and scheduling redeem attempts.

Because a call toredeem donates all of the call's gas, doing multiple requires limiting the amount of gas provided to each subcall. Otherwise the first will take all of the gas and force the second to necessarily fail irrespective of the estimation's gas limit.

Gas estimation for Retryable submissions is possible via theNodeInterface and similarly requires the auto-redeem attempt to succeed. Edit this page Last updatedonMar 7, 2024 Previous Inside AnyTrust Next L1 Pricing