Transaction Pricing

Every transaction using LayerZero has four main cost elements, one for each component that enables cross-chain messaging:

1. an initial source blockchain transaction
2. the fee paid to the OApp's configuredSecurity Stack
3. the configuredExecutor
4. fee for executing the message on the destination chain
5. the message execution cost of the transaction on the destination blockchain itself

Below are common best practices for accurately pricing and handling your transactions in LayerZero:

Message Execution Options

Because the source chain has no concept of the destination chain's state, you must specify the amount of gas you anticipate will be necessary for executing yourlzReceive orlzCompose method on the destination smart contract.

LayerZero provides robustMessage Execution Options , which allow users to provide detailed instructions regarding the gas amount andmsg.value the Executor pays for message delivery on the destination chain:

bytes

memory options = OptionsBuilder . newOptions ( ) . addExecutorLzReceiveOption (200000 ,

0) . addExecutorLzComposeOption (0 ,

1000000 ,

0) . addExecutorLzComposeOption (1 ,

500000 ,

0) ;

Gas Price Volatility

Gas prices can be highly volatile or have higher opcode costs on certain EVM-compatible chains. To mitigate the risk of transactions stalling due to fluctuating gas prices, it is advisable to test destination gas costs and incorporate a buffer by allocating additional gas upfront.

This strategy is essential to prevent transactions from getting "stuck" in the messaging channel due toOUT-OF-GAS issues.

For a more accurate estimation of gas requirements, a_quote method is available which calculates the transaction gas cost based on the four components above.

SeeEstimating Gas Fees to learn more.

Handling Errors

Transactions in LayerZero may occasionally encounter delays in transit from the source chain to the destination chain. Common causes for these delays include:

- Failure to initiate a valid transaction from the source chain into the LayerZero protocol.
- Insufficient gas payments made by the user.
- Transaction reverts on the destination chain, either due to in-contract or configuration issues.

LayerZero Scan offers a comprehensive tool for users to track their transactions. It provides detailed insights into where transactions may encounter delays, serving as a starting point for debugging.

Users can find detailed guidelines and support for debugging and recovering stalled transactions inDebugging Messages .

Real World Example

Lets assume user wants to send a message from Chain A to Chain B.

The price in Chain A native token is the quote for the gas to be consumed on Chain B.

If user receives a quote for a 200,000 gas units then it was obtained by multiplying the gas by the gas price on the destination chain. It also takes into account dollar prices of the source and destination native tokens.

Source native token quote is calculated using following formula:

GAS × DESTINATION_GAS_PRICE × DESTINATION_NATIVE_TOKEN_PRICE SOURCE_NATIVE_TOKEN_PRICE
\text{GAS} \times \text{DESTINATION_GAS_PRICE} \times \frac{\text{DESTINATION_NATIVE_TOKEN_PRICE}}
{\text{SOURCE_NATIVE_TOKEN_PRICE}} GAS

× DESTINATION_GAS_PRICE

× SOURCE_NATIVE_TOKEN_PRICE

DESTINATION_NATIVE_TOKEN_PRICE

We can use real world chains to demonstrate it. Lets assume Chain A is "Astar" and Chain B is "Astar zkEVM".

We end up with:200000 * (4000000000 / 10**18) * 3500 / 0.15 = 18.7 ASTR quote "Astar" uses token "ASTR" as a native token, and "Astar zkEVM" uses "ETH" as its native token. Other assumptions are:

1. ASTR = ~0.15
2. ETH = ~3500
3. Destination gas price = 4 Gwei Edit this page

Previous Estimating Gas Fees Next OApp Config