Estimating Transaction Costs on OP Mainnet

In this tutorial, you'll learn how to use theOptimism SDK(opens in a new tab) to estimate the cost of a transaction on OP Mainnet. You'll learn how to estimate theexecution gas fee and theL1 data fee independently. You'll also learn how to estimate the total cost of the transaction all at once.

Check out the full explainer onOP Mainnet transaction fees for more information on how OP Mainnet charges fees under the hood.

Dependencies

- node(opens in a new tab)
- pnpm(opens in a new tab)

Create a Demo Project

You're going to use the Optimism SDK for this tutorial. Since the Optimism SDK is aNode.js(opens in a new tab) library, you'll need to create a Node.js project to use it.

Make a Project Folder

mkdir

op-sample-project cd

op-sample-project

Initialize the Project

pnpm

init

Install the Optimism SDK

pnpm

add

@eth-optimism/sdk

Install ethers.js

pnpm

add

ethers@^5 Want to create a new wallet for this tutorial? If you havecast (opens in a new tab) installed you can runcast wallet new in your terminal to create a new wallet and get the private key.

Get ETH on OP Sepolia

This tutorial explains how estimate transaction costs on OP Sepolia. You will need to get some ETH on OP Sepolia in order to run the code in this tutorial.

You can use theSuperchain Faucet(opens in a new tab) to get ETH on OP Sepolia.

Add a Private Key to Your Environment

You need a private key in order to sign transactions. Set your private key as an environment variable with theexport command. Make sure this private key corresponds to an address that has ETH on OP Sepolia.

export TUTORIAL_PRIVATE_KEY = 0 x...

Start the Node REPL

You're going to use the Node REPL to interact with the Optimism SDK. To start the Node REPL run the following command in your terminal:

node This will bring up a Node REPL prompt that allows you to run javascript code.

Import Dependencies

You need to import some dependencies into your Node REPL session.

Import the Optimism SDK

const

optimism

=

require ("@eth-optimism/sdk")

Import ethers.js

const

ethers

=

require ("ethers")

Set Session Variables

You'll need a few variables throughout this tutorial. Let's set those up now.

Load your private key

const

privateKey

=

process . env . TUTORIAL_PRIVATE_KEY

Create the RPC provider

Here you're creating a standard Ethers RPC provider and wrapping it as anL2Provider , a class provided by the Optimism SDK. This will add a few extra functions to the provider object that you'll use later in this tutorial.

const

provider

=

optimism .asL2Provider (new

ethers . providers .StaticJsonRpcProvider ("https://sepolia.optimism.io"))

Create the wallet instance

const

wallet

=

new

ethers .Wallet (privateKey , provider)

Estimate Transaction Costs

You're now going to use the Optimism SDK to estimate the cost of a transaction on OP Mainnet. Here you'll estimate the cost of a simple transaction that sends a small amount of ETH from your address to the address0x1000000000000000000000000000000000000000 .

Create the unsigned transaction

Ethers makes it easy to create unsigned transactions so you can estimate the cost of a transaction before you a user to sign it. Here you'll create an unsigned transaction that sends a small amount of ETH from your address to the address0x1000000000000000000000000000000000000000 . You can also create unsigned transactions that interact with contracts usingContract.populateTransaction (opens in a new tab) .

const

tx

=

await

wallet .populateTransaction ({ to :

'0x1000000000000000000000000000000000000000' , value :

ethers . utils .parseEther ('0.00069420') , gasPrice :

await

provider .getGasPrice () , })

Estimate the execution gas fee

You can estimate the execution gas fee the same way you'd estimate the gas fee for any transaction on Ethereum. Simply multiply the gas limit by the effective gas price.

const

gasLimit

=

tx .gasLimit const

gasPrice

=

tx .maxFeePerGas const

l2CostEstimate

=

gasLimit .mul (gasPrice) console .log (ethers . utils .formatEther (l2CostEstimate))

Estimate the L1 data fee

You can estimate the L1 data fee with theestimateL1GasCost (opens in a new tab) function. Under the hood, this function is estimating the amount of Ethereum gas required to publish this transaction on Ethereum and multiplying it by the current Ethereum gas price (as tracked by the L2). This function returns the current cost estimate in wei.

const

l1CostEstimate

=

await

provider .estimateL1GasCost (tx) console .log (ethers . utils .formatEther (l1CostEstimate))

Estimate the total cost

Once you've individually estimated the execution gas fee and the L1 data fee, you can sum these two values together to get the total cost of the transaction.

const

totalSum

=

l2CostEstimate .add (l1CostEstimate) console .log (ethers . utils .formatEther (totalSum))

Send the transaction

Now that you've estimated the total cost of the transaction, go ahead and send it to the network. This will make it possible to see the actual cost of the transaction to compare to your estimate.

const

res

=

await

wallet .sendTransaction (tx) const

receipt

=

await

res .wait () console .log (receipt .transactionHash)

Check the actual execution gas fee

Once you get back the transaction receipt, check the actual execution gas fee.

const

l2CostActual

=

receipt . gasUsed .mul (receipt .effectiveGasPrice) console .log (ethers . utils .formatEther (l2CostActual))

Check the actual L1 data fee

You can also check the actual L1 data fee.

const

l1CostActual

=

receipt .l1Fee console .log (ethers . utils .formatEther (l1CostActual))

Check the actual total cost

Sum these two together to get the actual total cost of the transaction.

const

totalActual

=

l2CostActual .add (l1CostActual) console .log (ethers . utils .formatEther (totalActual))

Check the difference

Finally, check the difference between the estimated total cost and the actual total cost. This will give you a sense of how accurate your estimate was. Estimates will never be entirely accurate, but they should be close!

const

difference

=

totalActual .sub (totalSum) .abs () console .log (ethers . utils .formatEther (difference))

Tracing Deposits and Withdrawals Triggering OP Mainnet Transactions From Ethereum