Article ethereum

Gas and Payment - Ethereum Yellow Paper Walkthrough (3/7)

Lucas Saldanha
04 Mar 2019 • 8 min read

Another day, another Ethereum Yellow Paper blog post! In this post, we will learn more about Gas and Payments in Ethereum. We will also brush over the economics behind Ethereum and why fees are so important in the Ethereum ecosystem.

After reading this post, we will understand why transaction costs are called *gas* and what the difference is between gasPrice and gasLimit. We will also understand what miner nodes are and what strategies they use to select transactions. This post refers to section 5 of the Ethereum Yellow Paper.

This is the third post in the series Ethereum Yellow Paper Walkthrough. The goal of this series is to demystify the concepts in the paper and make it accessible to a broader audience. If you missed the previous posts, here they are!

- The blockchain paradigm - Ethereum Yellow Paper Walkthrough (1/7)
- Merkle Tree and Ethereum Objects (World State, Transaction, Block, etc.) - Ethereum Yellow Paper Walkthrough (2/7)

(DISCLAIMER: this post is based on the Byzantium version of the Yellow Paper, version 12779ac from 27th November 2018)

Introduction

When I first started learning about Ethereum and what it was capable of, one of my first questions was around transaction fees. I remember thinking: **“Why do I need to pay to use a Dapp if I can use similar services for free?”**. It didn’t take long until I realised how wrong I was.

Computing costs money, and I’m not talking about how much you paid for your laptop or tablet. I’m talking about the costs for hosting a service, storing data, and processing information. Nowadays, **we are so used to free services that we forget one key aspect: someone is paying the bills**. If you are using an email service like Gmail or Yahoo, your email provider is paying for all their servers, data storage and the infrastructure costs of keeping the service up and running. If you have a Facebook or Instagram account, you must know that all the computing power to process your pictures uploads and store them safely in the cloud is not free. The only difference is that someone (that is not you) is paying the bills.

Now think about the Ethereum network as one big computer. You can use it to perform computations and to read and write data to its storage. Different to the computer that you have at home, **the Ethereum computer is shared among everyone**. Anyone with an Ethereum account can transfer Ether, deploy smart contracts and interact with the platform. However, due to its distributed nature, the question about who pays the bills is a bit harder to answer.

Gas

The **Ethereum solution for who pays the bill is Gas**. This is the unit in which all computation in Ethereum is priced. Do you want to transfer Ether between two accounts? Ok, this will cost you some gas. Do you want to deploy a smart contract that stores your phone contacts? No problem! You just need to pay the gas fees.

The analogy with a car and fuel is inevitable. If you own a car, and you need to drive it from point A to point B, you need an amount of fuel. In the same way, if you have some operations that you want to execute in the Ethereum EVM, you need gas. With your car, the further you drive, the more fuel you need. In Ethereum, **the more you compute, the more gas you need**.

How much gas you need for each operation is specified in Appendix G of the Yellow Paper. The values might seem arbitrary but there is reasoning behind them. Basically, the cost in gas of an operation is a representation of the computational cost of performing that operation (measured in time) and the amount of permanent storage required by that operation (when writing to the storage). If you want to review the formula used to calculate the gas cost of each operation, you can check the Ethereum 1.0 gas cost spreadsheet. I don't know if it is up to date with the latest implementations, but it should give you an idea of the reasoning behind the gas costs.

Another aspect of charging the user for their actions in the network is to prevent abuses. If you are paying for every operation you execute, you'll do your best to implement your code in the most efficient way. The gas cost also prevents bad actors from flooding the system with useless operations (unless they are willing to spend a lot of money to execute useless code).

gasPrice and gasLimit

Now that we understand what gas is, it is time to understand how much it costs. To understand that, let's go back the car and fuel analogy.

If your car has a 50 liter tank, how much do you pay to completely fill the tank? The answer depends on the price per litter in the pump, right? It is the same with Ethereum and gas! If you have a transaction that needs 10 gas to execute, **the price you pay to execute that transaction depends on the price per unit of gas**.

So how do you know what the price of a unit of gas is? The misleading answer would be: you can pay as much as you want! Technically, this answer is not wrong, but we need a bit more context to understand how exactly the price is set. So let's start!

If you read my previous post in the series, you might remember that a transaction has, among other fields, a **gasPrice** and **gasLimit**.

The gasPrice is the value that the transaction sender is willing to pay per gas unit. That means that the transaction sender is capable of choosing how much he wants to pay per unit of gas. If our transaction needs 10 gas and we are willing to pay 3 Wei per unit of gas, our transaction cost would be 30 Wei in total (I'm not using real values, I just want you to understand the basis of the calculation).

The gasLimit is the maximum gas that the transaction sender is willing to spend executing that transaction. Sometimes, when executing a transaction, you might not know exactly how much it is going to cost. Imagine a scenario where you have a smart contract with a bug, an infinity loop. Without a gasLimit, it would be possible to consume the whole balance of the sender account. The gasLimit is a safety mechanism to prevent someone from using all their Ether due to a bug or an estimation error.

Another interesting aspect of the gasLimit is that it can be seen as a prepaid gas amount. When validating a transaction, the node multiplies the gasPrice by the gasLimit to calculate the intrinsic cost of the transaction. If the intrinsic cost is higher than the balance of the sender account, the transaction is considered invalid. After the transaction has been processed, any unused gas is refunded to the sender account. However, if your transaction runs out gas during execution, there is no refund. That is why usually the transaction sender sets the gasLimit a higher than the estimated amount of gas.

Now that we know what these two parameters are, you might

be wondering **why the transaction sender is the one that chooses the price to pay per unit of gas**. If you try going to your nearest gas station and saying to the cashier: “I want to pay 5 cents per liter of fuel”, the cashier will either laugh at you (if he is nice) or call the cops (if he is sane). To understand how this works, we need to understand what a Miner node does and what mining fees are.

Miners

The Miner nodes in Ethereum are the nodes that create blocks in the chain. A block is a data structure that contains a set of transactions. When creating a block, the miner will select some transactions from its pool of pending transactions (transactions waiting to be included in the chain) and start mining the block.

I don’t want to discuss the details about the mining algorithm used in Ethereum (maybe in a future post). The important thing to know is that mining is an expensive process. Therefore, if Miners didn’t get anything in return for mining, no one would do it.

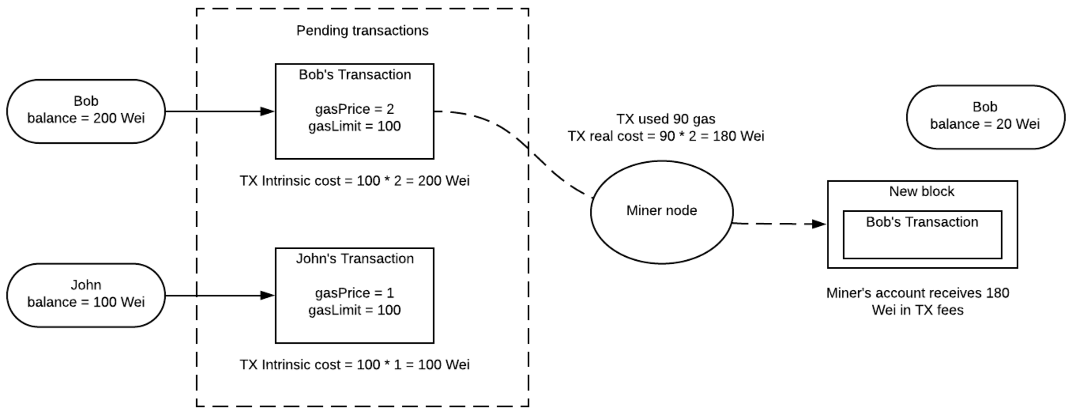
In Ethereum, when a miner mines a new block, it receives the fees from all transactions included in this block. Therefore, **the higher the gasPrice in the transactions, the higher the fees that the miner receives will be.** The miner also receives a fixed reward per block and a reward for including uncles in the block, but we won't discuss them in this post. If you want to

know more about the different rewards the miner can receive, check this page.

Let’s use an simple scenario here to illustrate. Bob’s account has a balance of 200 Wei and John’s accounts have a balance of 100 Wei and they both want to send a transaction that needs 90 gas to execute.

Bob creates the transaction with gasLimit = 100 and gasPrice = 2. Unfortunately, John only has 100 Wei, he can’t set the gasLimit to 200 because that would make the transaction intrinsic cost higher than his current balance. John creates the transaction with gasLimit = 100 and gasPrice = 1.

When it is time to pick a transaction to include in the next block, the miner node is likely to choose the transaction that will reward him more fees. In our example, Bob has set a gasPrice twice as high as John's gasPrice. Since both transactions have the same gas cost, the miner will receive twice as much Wei as a reward if it chooses Bob's transaction.



Miner selecting the transaction with highest gasPrice

This mechanism of charging the transaction sender and rewarding the miner creates a self-regulated economy. The senders are always trying to minimise fees and the miners always trying to maximize their reward. When sending a transaction, you can set a higher gasPrice to make mining this transaction more interesting to miners, resulting in the transaction being mined faster.

Some miners even have a minimum gasPrice, meaning they ignore any transactions with a gasPrice lower than what they want.

When sending a transaction, it can be hard to know what is the minimum gasPrice at that moment. There are some tools that scan the network and the average gasPrice used in recent transactions to help with choosing a fair gasPrice that is likely to be accepted by miners.

Conclusion

In this post we learnt how **Ethereum transactions need gas to run**, just like our cars. I hope this post opened your eyes to the cost of computing and why we need to pay to use the Ethereum computer.

We also discussed what **gasPrice** and **gasLimit** are, and why it is so important to understand what they are used for. We saw how the **gasLimit is used to protect the user from wasting his Ether** because of a bug in a smart contract or estimation errors.

Last but not least, we took a look at the economics behind transaction fees and how the **Miners choose transactions to maximise their return in fees**. Now we know that we can **adjust the gasPrice to make a transaction more attractive to miners** and, consequently, have it mined faster.

In my next post, I'd like to discuss the Transaction Execution model (Section 6 of the Yellow Paper). This is, by far, the most complex part of the Yellow Paper (believe me, they even said that in the Yellow Paper!), and I might break it into more than one post. Stay tuned for more!

Have you enjoyed this series so far? I'd love to hear your thoughts. As always, please leave a comment if you find anything wrong or if there is something that I can improve. I'm always open to feedback. :)

See you in the next one!

References

- Ethereum Yellow Paper
- Ethereum 1.0 gas cost spreadsheet
- Understanding gas in Ethereum
- Guide to Ethereum: What is Gas, Gas Limit and Gas Price?
- Ethereum Wiki - Mining Rewards
- ETH Gas Station

Topic ethereum blockchain Share

Show Comments

Transaction Execution -...

In this post, we will look at how the Ethereum platform...

05 Nov 2019

Merkle Tree and Ethereu...

Hi everyone! This is another post in our series exploring the...

11 Dec 2018