b
UNIVERSITÄT
BERN

Optimistic rollups: Offchain Labs Arbitrum

Cost differences analysed

Seminar report

Marius Asadauskas

from

Bern, Switzerland

Faculty of Science, University of Bern

23. December 2021

Prof. Christian Cachin
Cryptology and Data Security Group
Institute of Computer Science
University of Bern, Switzerland

Abstract

The recent growth in popularity surrounding blockchain-based technologies has brought many new users into the world of cryptocurrencies. While the increase in user base is mostly regarded as positive, it has also brought many of the issues related to distributed systems to light. One of these issues being the limited throughput that various cryptocurrencies suffer from. This effect is especially noticeable in Ethereum, where the transaction costs have increased drastically as a result.

To mitigate this issue, Layer 2 solutions have been proposed. These would offload and process some of the transaction data offchain and only post the most important information on chain. The most prominent of these layer 2 solutions is Arbitrum [1] with their protocol being based on optimistic rollups.

In the following report, we will analyse the cost differences between transactions happening on the layer 1 Ethereum blockchain and transactions happening on layer 2 via Arbitrum as to see whether the use of Arbitrum is practical for average users.

Contents

Chapter 1

Introduction

1.1 Transaction Fees

Transaction fees exist inside cryptocurrencies to act as an incentive for miners to add a transaction to a block. This incentive is necessary since before adding a transaction to a block, you must first perform validity checks, such as making sure that no double spending is happening. You must also perform other computations, such as computing the Merkel root of the transaction and updating the Merkel tree. Without an incentive to do so, a miner could instead focus all their resources on winning the Proof of Work race and claiming the reward.

Furthermore, the idea of transaction fees is nothing new. It was first proposed in the Bitcoin Witepaper [2] by Satoshi Nakamoto and is still in use today. Problems only start to arise once the throughput limit of the network has been reached. Once that happens, users stop paying for the computational costs and instead pay a much higher fee to get a spot in the block. As a result, transaction costs rise in tandem with the popularity of a cryptocurrency.

1.2 Layers in Blockchains

In the blockchain ecosystem, Layer 1 is the term used when describing the underlying blockchain architecture. The term Layer 2 is used when talking about protocols that run on top of Layer 1 and do not require modifications to the first layer. An example of a Layer 1 protocol would be the Ethereum blockchain. An example of a layer 2 protocol would be Arbitrum, which runs on top of the Ethereum blockchain. You can see this illustrated below in fig. 1.1.

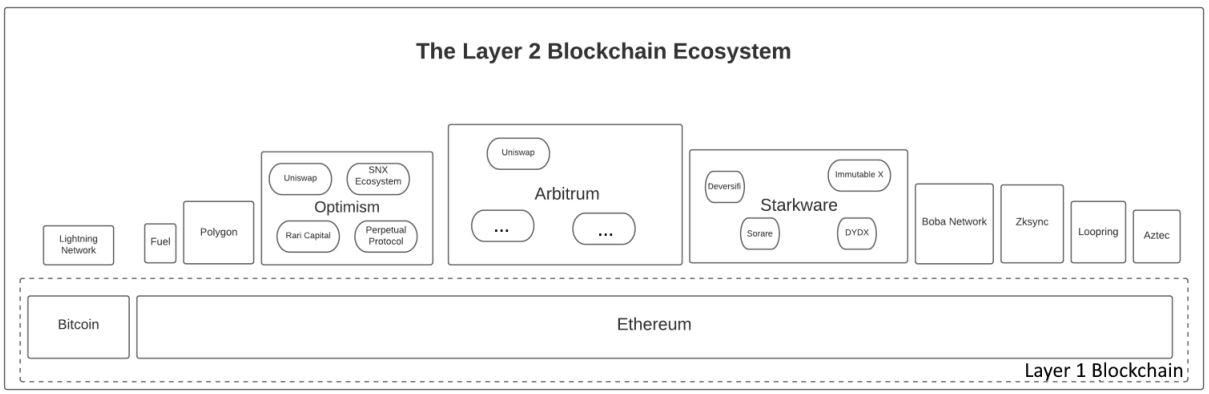

Figure 1.1. Layer differentiation

1.3 Optimistic Rollups

We have already mentioned Arbitrum and how it is a possible Layer 2 solution to solve Ethereum's scaling and throughput problem, but we have yet to mention how it functions. Arbitrum uses a protocol that is based on optimistic rollups. The name "optimistic rollups" is made up of two parts.

1.3.1 Rollups

The rollup half of optimistic rollups refers to the fact that only a rollup of the transaction data gets posted on Layer 1. For example, if you create a smart contract on Arbitrum, you only have to post its hash, a hash of its state, and some other parameters on Layer 1. Doing so takes up much less space on the Ethereum blockchain, and having a 256-bit cryptographic hash of a smart contract saved on the Ethereum blockchain ensures that the smart contract becomes immutable. The illustration below in fig. 1.2 shows a simplified version of how rollups work.

1.3.2 Optimistic

The optimistic half of optimistic rollups refers to how the rollups get posted on the Ethereum blockchain. Any manager of a smart contract can assert that a state change has happened. Arbitrum is optimistic in the sense that if nobody disputes this claim, the hash will be updated. However, if another user challenges the claim within the given time frame of around a week, the protocol will go into the bisection protocol. The bisection protocol takes a computation and splits it into two equal halves until a single false instruction is found. Once the single false instruction is found it can be sent to Layer 1 for evaluation to determine whether the claim was valid or not.

This way in the optimistic scenario nothing ever has to be computed on chain and in the worst scenario only a single instruction must be computed on chain.

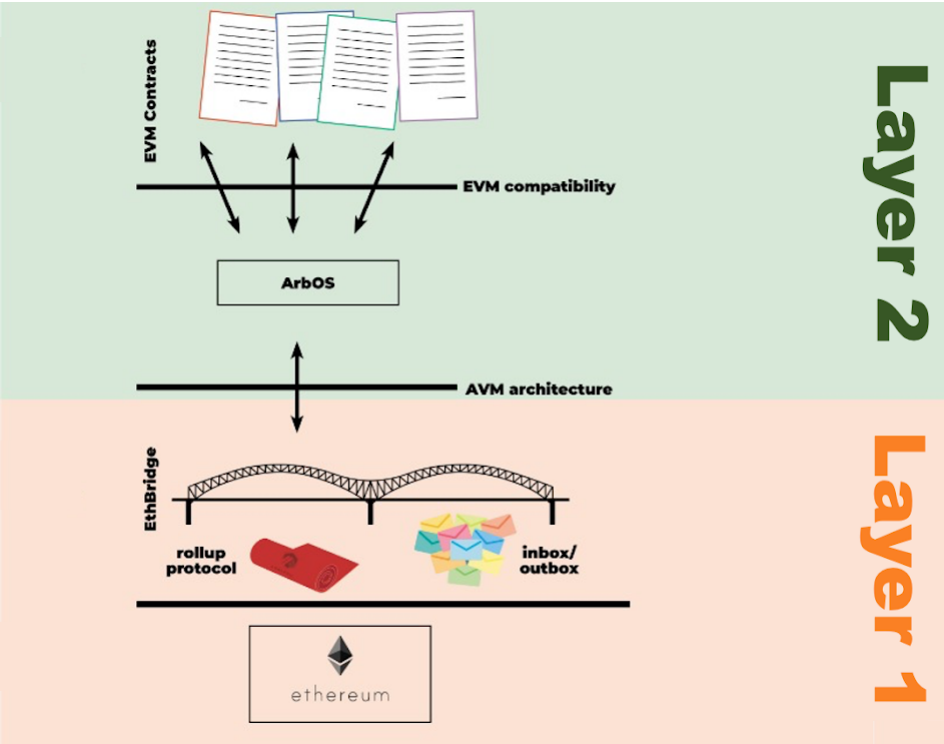

Figure 1.2. Optimistic Rollups

Chapter 2

Background

2.1 Arbitrum White Paper

The Arbitrum White Paper [3] was the first paper to propose optimistic rollups back in 2018. Over the following years, many of the details surrounding Arbitrum's implementation have changed and been refined. For example, in 2018 the main focus of Arbitrum was to increase privacy, scalability, and computational power of smart contracts. Nowadays, the word "privacy" is nowhere to be found on Arbitrum's website or in its protocol description. Enhanced computational power still remains a priority, but the main focus has shifted to scalability and increasing throughput. The main idea of relying on optimistic rollups to achieve this goal, however remains unchanged.

2.2 Arbitrum Developers' Guide

The Arbitrum developers' guide [4] is the main source of information when it comes to Arbitrum today. It contains instructions on how to deploy smart contracts on Arbitrum. Information on how transaction fees are calculated. Descriptions of what smart contracts are running on Ethereum to make Arbitrum possible. Most, if not all, information related to Arbitrum today can be found there.

2.3 Arbiscan

Arbiscan is a website [5] that allows its users to explore the Arbitrum blockchain without having to download the chain locally. This interface makes gathering data from the Arbitrum chain much easier and was the main tool we used when calculating the average transaction costs in Arbitrum. The website is controlled by Arbitrum and was developed together with the help of the makers of Etherscan [6].

Chapter 3

Analysing the cost differences

In the following section we will be analysing the cost differences between Arbitrum and Ethereum from both a theoretical and a practical point of view.

3.1 Theoretical costs

For finding out the theoretical cost differences between Arbitrum and Ethereum, our best reference was the Arbitrum developer guide [4]. This contained detailed explanations of how Arbitrum calculates the cost of each transaction.

Similar to Ethereum, Arbitrum uses a measure called ArbGas. The idea behind ArbGas is identical to that of Ethereum. Since nodes have to verify a transaction, you pay them in ArbGas. With the main difference being that ArbGas costs much less than Ethereum Gas and has a higher global limit. This way, you pay less for transactions and can perform larger computations. However, Translating ArbGas into Ether is not as simple. Firstly, the Arbitrum network must calculate an estimate of the Ethereum gas fee. We define this as

$$P := \text{Expected L1 Gas Price.}$$

Afterwards, the transaction fee gets calculated via four different aspects:

- L2 *tx*: A base fee for each L2 transaction, to cover the cost of servicing a transaction

  $$\text{L2 } tx = \frac{CP}{B}.$$

  With C being the L1 cost to submit a Batch to L1 and B being the batch size.

- L1 *calldata*: A fee per units of L1 calldata directly attributable to the transaction (each non-zero byte of calldata is 16 units, and each zero byte of calldata is 4 units)

  $$\text{L1 } calldata = P \cdot \text{calldata units.}$$

- *computation*: A fee per unit of ArbGas used by the transaction.

  $$computation = \text{ArbGas} \cdot \text{Base Price}$$

  With the base price being determined by the congestion of the Arbitrum network. If the network is under heavy load, the base price gets multiplied by $9/8$. If the network is under low load, the base price gets multiplied by $7/8$. Furthermore, the base price cannot go below $P/10'000$.

- *storage*: A fee per location of EVM contract storage, based on the net increase in EVM storage due to the transaction.

  $$storage = 2000 \cdot P \cdot \text{storage}$$

You must pay 2'000 times the estimated L1 gas price per 256-Bit word of storage allocated. Considering that in Ethereum you pay 20'000 gas per 256-bit word this implies that in Arbitrum you pay 10% the expected Ethereum storage price.

If we add it all together we get

$$\text{Transaction Fee} = \text{L2 } tx + \text{L1 } calldata + computation + storage.$$

Furthermore, if we assume that the base price is the minimum value of $P/10'000$ we get

$$\text{Transaction Fee} = P(\frac{C}{B} + \text{calldata units} + \frac{\text{Arb. Gas}}{10'000} + 2000 \cdot \text{storage}).$$

3.1.1 Example: Simple transaction

According to the Ethereum yellow paper [7], for a simple transaction on Ethereum from one address to another you would have to pay 21'000 Gas. If we assume that the base fee was estimated correctly and no tip was given, we would get

$$\text{Eth. cost} = 21,000 \cdot (\text{Base Fee} + \text{tip}) = 21'000 \cdot P.$$

The same transaction on Arbitrum would cost us

$$\text{Arb. cost} = P(\frac{C}{B} + 0 + \frac{21'000}{10'000} + 2000 \cdot 0) = \frac{CP}{B} + 2.1P.$$

We can already see that the main cost comes from submitting the transaction to the L1 chain, while verifying the transaction would only cost us 2.1 Gas. If we take some reasonable values for B and C we would get $C = 1'200'000$, $B = 200$ and then the cost becomes

$$\text{Arb.cost} \approx 6'000P + 2.1P = 6002.1P$$

In theory a basic Arbitrum transaction could cost around $6002.1/21'000 \approx 0.286$, which is 28.6% of the price of an Ethereum transaction. However, this value depends heavily on $\frac{C}{B}$ and the percentage will only go down the more difficult a transaction becomes. Consider the following

$$\lim_{\text{Gas}\to\infty} \frac{P(6'000 + \text{Gas}/10'000)}{P \cdot \text{Gas}} = \lim_{\text{Gas}\to\infty} \frac{6000 + \text{Gas}/10'000}{\text{Gas}} = \lim_{\text{Gas}\to\infty} \frac{6000}{\text{Gas}} + \frac{1}{10'000} = \frac{1}{10'000}$$

This means that extremely computationally heavy transactions on Arbitrum could cost one 10'000th of the price they would on the Ethereum blockchain. In practice however, we can not let the ArbGas go to infinity since we would reach the ArbGas limit.

3.2 Practical costs

On august 31st, 2021, the Arbitrum Mainnet went online [8] and became accessible to the public. This development allowed us to not only theoretically analyse the cost differences but also analyse the practical cost differences between transactions happening on Arbitrum and transactions happening on Ethereum.

A motivation to do so was the fact that average transaction costs were readily available for Ethereum [9], while for Arbitrum there were no such services. Furthermore, Arbiscan, the main blockchain explorer for Arbitrum, is made with the help of Etherscan [6] and both applications are close to identical in terms of design and functionality. One of the key differences between the two sites is that Etherscan provides this information while Arbiscan does not. The exclusion of the average transaction costs seems intentional from the side of Arbitrum.

3.2.1 Data Gathering

Arbiscan provides an API for developers to gather and request data from the blockchain. This was very helpful since it meant that we did not have to store the blockchain ourselves. Using the provided API seemed like the most logical way of gathering data until we realized that the provided calls were for very specific requests that involved exact account addresses and not for the type of broad requests we needed to calculate average transaction costs.

We instead opted to get the data ourselves from the Arbiscan website directly via html extraction. For this, we employed the use of urllib3 [10], which would request and store the pages, and BeautifulSoup [11], which was responsible for extracting the information. Arbiscan keeps track of the past 500'000 transactions with 50 transactions per page. This amounts to around 2 weeks worth of transactions.

3.2.2 Average calculation

The 500'000 entries were stored on a total of 10'000 pages. Instead of processing all 10'000 pages and putting a large load on the Arbiscan servers we opted to process a limited amount of pages chosen from the entire set. However, taking pages at random was also not an option. This would only give us the average of 50 entries which would be prone to large variations. We would instead create clusters of multiple pages and distribute theses clusters evenly throughout the 10'000 pages. Using urllib we collected the data, parsed it, and then used the function **numpy.mean**(costs) to get the average of each page and the average of each cluster. The function **numpy.meadian**(costs), would be more robust to outliers but it would return the median value instead of an average value.

3.2.3 Plotting the data

Plotting the obtained average values in matplotlib gave us the average transaction fees over the course of the backlog. This can be seen below in fig. 3.1

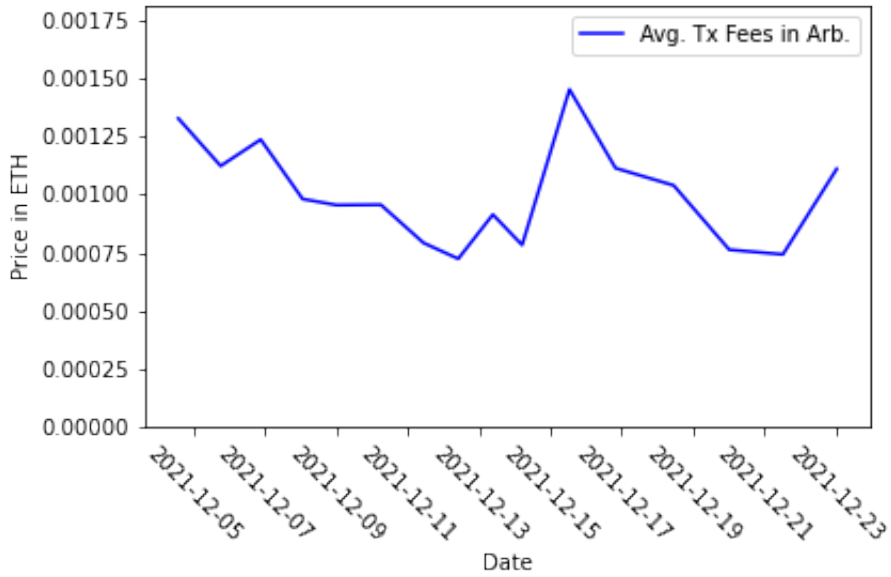

Figure 3.1. Average Transaction Fees in Arbitrum in ETH

In fig. 3.1 you can see the price of an average transaction in the Arbitrum network represented via the blue line. The y-axis shows us the average transaction costs in ETH and the x-axis shows us the respective dates on which these values were recorded. As we can see the backlog of 10'000 pages is enough to store 18 days worth of transactions. The average values of these transactions ranged from

0.0005 to 0.0015 ETH. We also see large variations between the data points. This could be due to the fact that the average is sensitive to outliers and we computed the average over 250 unfiltered entries.

After obtaining the average transaction fees we needed to obtain historic Ethereum prices, as this would allow us to plot the average transaction price in dollars. For this we used the python module cryptocompare. Multiplying the costs with the Ether prices at the given dates gave us the graph in fig. 3.2

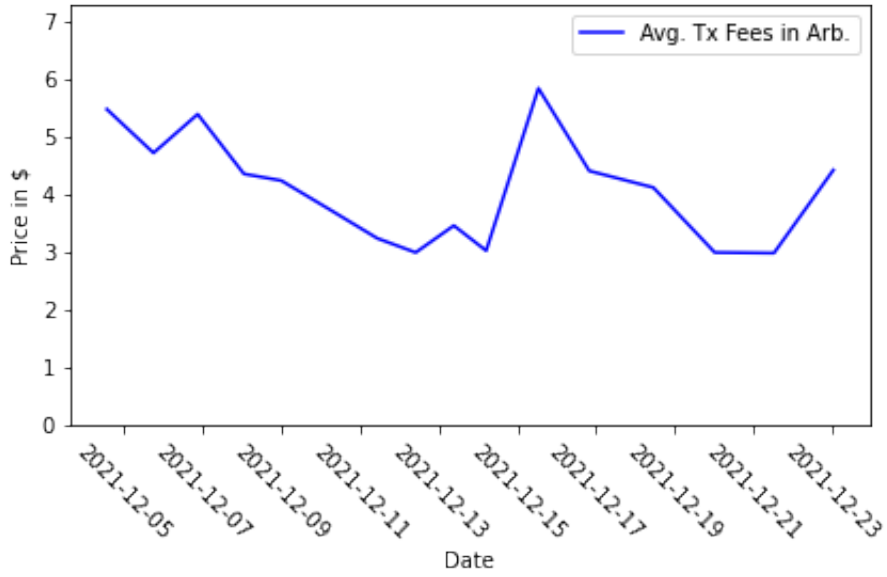

Figure 3.2. Average Transaction Fees in Arbitrum in USD

In fig. 3.2 We see a graph which is practically identical to the graph in fig. 3.1. The only difference being that the y-axis now represents the average transaction costs in USD instead of ETH. The x-axis continues to show us us the respective dates on which these values were recorded. We see that an average Arbitrum transaction ranges from 3-6 USD. We can also finally say that the expected transaction fee lies at around 4 USD. The strong variation in the graph remains even when multiplying the values with historic Ethereum prices. This is mostly due to the Ethereum prices not changing too drastically percentage-wise.

3.2.4 Comparison

To compare the average Arbitrum transaction fees with the average Ethereum transaction fees required us to first find a source of Ethereum transaction fees for given dates. For this we downloaded a csv file [12] which only included Layer 1 transactions in the average calculation. Some sources started including Layer 2 transactions into the average calculations, but this would impact our comparison unfairly in favour of Ethereum. Comparing the averages of the two gave us the following results seen in fig. 3.3.

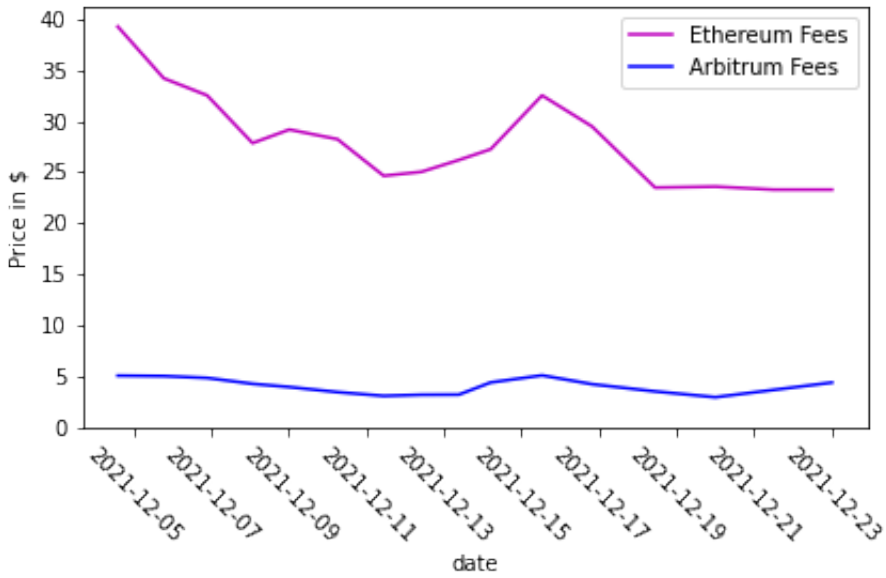

Figure 3.3. Arbitrum Ethereum Comparison

In fig. 3.3 We see the results of our data gathering. The purple line represents the average Ethereum transaction costs, while the blue line represents the average Arbitrum transaction costs. Furthermore, the y-axis represents the given cost translated to USD, while the x-axis represents the dates on which these values were recorded. We can see that the average Arbitrum transaction lies at around 4-5 USD while the average Ethereum transaction averages at about 30 USD. This means that Arbitrum transactions cost around 85% less compared to Ethereum transactions.

we can also see that the Arbitrum network is not immune to fluctuations in the Ethereum network. On the 5th of December and on the 16th of December Ethereum had its highest transaction costs and so did Arbitrum.

Lastly, if we plot one minus the average Arbitrum costs divided by the average Ethereum costs we get the following graph seen in fig. 3.4

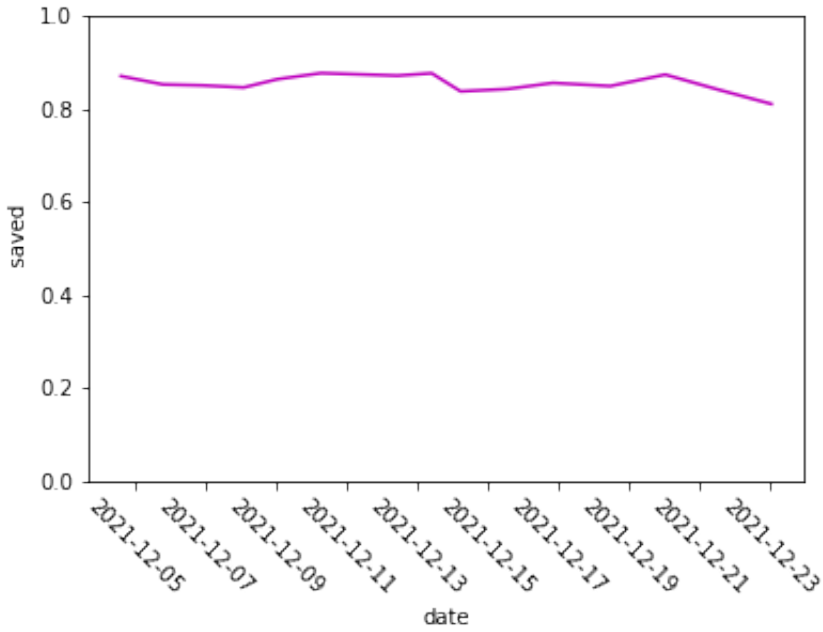

Figure 3.4. Arbitrum Cost Saved

The purple line represents the calculated ratio between Arbitrum and Ethereum transactions. The y-axis contains values between 0 and 1. The x-axis continues to represent the dates on which these values were recorded. We can see that executing a transaction on the Arbitrum network would save us on average more than 80% of the transaction costs.

Chapter 4

Conclusion

This report set out to give a clear comparison and provide concrete numbers between transactions on Layer 1, which happen on the Ethereum blockchain, and transactions on Layer 2, which happen on Arbitrum. For this, we first had to extract data from the Arbitrum blockchain and calculate the average transaction fees ourselves, since the Arbitrum blockchain explorer refused to provide this data.

When comparing both averages directly with each other, we can see that the average cost per transaction on the Arbitrum network is much lower than that of an average transaction on the Ethereum network. These results fit rather well with our theoretical estimate. There, we calculated that an Arbitrum transaction would cost less than 70% the amount that an equivalent Ethereum transaction would. Our results say that Arbitrum transactions cost 80% less. Overall the Practical implementation of Arbitrum coincides with the Theoretical implementation in terms of transaction fees.

The only questionable thing remains that Arbiscan does not display the average transaction fees, although this data is readily available to them. It highlights the strong centralization of Arbitrum and the power they hold over the network. Restricting the flow of information and instead only showing extremely favourable information is not in the spirit of decentralized networks.

Bibliography

[1] "Layer 2 rankings," https://l2beat.com/, accessed: 2021-12-20 at 12:00.

[2] S. Nakamoto *et al.*, "Bitcoin," *A peer-to-peer electronic cash system*, 2008.

[3] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner

[4] O. L. D. Center, "Arbitrum developer quickstart," https://developer.offchainlabs.com/docs/developer_quickstart, 2021.

[5] arbiscan, "Arbitrum charts, statistics," https://arbiscan.io/charts.

[6] Arbitrum, "Arbiscan is now live," https://twitter.com/arbitrum/status/1432112087489257482, 2021.

[7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[8] O. Labs, "Mainnet for everyone," https://offchain.medium.com/mainnet-for-everyone-27ce0f67c85e, 2021.

[9] etherscan, "Average transaction fee chart," https://etherscan.io/chart/avg-txfee-usd.

[10] A. Petrov, "urllib3," https://pypi.org/project/urllib3/.

[11] "Beautiful soup documentation," https://beautiful-soup-4.readthedocs.io/en/latest/.

[12] Messari. Average transaction fees. https://messari.io/asset/ethereum/chart/txn-fee-avg. Accessed: 2021-12-22 at 20:00.