Are the Gas Prices Oracle Reliable? A Case Study using the EthGasStation

Giuseppe Antonio Pierro
Dep. of Mathematics and Computer Science
University Of Cagliari
Cagliari, Italy
antonio.pierro@gmail.com

Henrique Rocha
Dep. of Mathematics and Computer Science
University of Antwerp
Antwerp, Belgium
henrique.rocha@gmail.com

Roberto Tonelli
Dep. of Mathematics and Computer Science
University Of Cagliari
Cagliari, Italy
antonio.pierro@gmail.com

Stéphane Ducasse
Inria Lille - Nord Europe
Villeneuve D'ascq, France
stephane.ducasse@inria.fr

***Abstract*—The Ethereum Blockchain is a distributed database that records all transactions and smart-contracts created on the platform. In Ethereum blockchain, the user needs to set a Gas price to get a transaction recorded. To have the transaction recorded, the Gas price has to be greater than or equal to the lowest Ethereum transaction fees. To help the users and smart contracts to set the right Gas price, the Gas Oracle categorizes the gas price into categories based on the interval of time the user might be willing to wait and for each of them suggests a gas price to set. The paper aims to verify the hypothesis that the predictions made by the EtherGasStation Oracle have a margin of error greater than the margin of error declared by it (2%). We collected data in two-months time from the EthGasStation Oracle which predict the Gas Price every time that 100 blocks are added to the Ethereum Blockchain. In the same time frame, two-months, we also collected over 10 million transactions from a Transaction Pool. By cross-checking the data collected by the Transaction Pool and the Gas Oracle, the study revealed that the Gas Oracle fails more often than it advertises.**

***Index Terms*—Ethereum, Gas, transaction fees, Gas price categories**

I. Introduction

In Ethereum [1], users need to pay a fee in a special resource called Gas when creating transactions [4], [7], [11]. Gas is like fuel for computational instructions executed in the blockchain. There are mainly three reasons for the Gas fees concept: (i) to make the users pay for the computation costs (e.g., energy, CPU) required to create and approve their transactions; (ii) to limit blockchain resource use; and (iii) to avoid issues of intentional or non-intentional network abuse (e.g., DoS attacks, infinite loops).

Transactions occurring in the network are verified by special nodes named "miners". In Ethereum, verifying a transaction means checking the sender and the content of the transaction. Miners generate a new block of transactions and then add such a block to the network. Currently, miners need to solve a mathematical puzzle (called "Proof of Work") to create a new Ethereum block. Miners receive the Gas transaction fees converted into cryptocurrency as a reward for adding a new block to the blockchain [3], [4], [7].

The Gas price value is set by the user who chooses how much to pay to execute the transaction. If the value set by the user is too low for the Ethereum miners, the transaction risks to never be included in the blockchain. On the other hand, if the transaction price is very high, the blockchain miners will be prone to include it in the Ethereum Blockchain, but the user will allegedly waste money.

In this paper, we analyze the data of one popular oracle to predict the Gas price, along with the Ethereum transactions' and blocks' data. More specifically, we use in this study the *EthGasStation* oracle, as its API is public and can be used by any other oracle or user. The Ethereum transactions' variables considered in the study are:

- the interval of time elapsed between the time when the transaction was first seen in the Transaction Pool and the time in which the transaction was added to the Ethereum Blockchain;
- the Gas price, i.e., the amount of Ether the user is willing to pay for every unit of Gas, which is measured in "GWei".

Oracle data is useful to predict the Gas price a user should pay to influence miners to verify a transaction (and consequently, add such transaction to a block). To help users decide on the price to pay to submit a transaction, Gas Oracle proposes the following four price categories: safe low, standard, fast, and fastest. These categories define the Gas price required to have a transaction confirmed within the next 100, 20, 5, and 2 blocks, respectively.

First, the results show that EthGasStation gives the Gas price prediction with a higher margin of error compared to what it claims (2%). The margin of error ranges from a minimum of 4% for the "fastest" category to a maximum of 28% for the "fast" category. Second, we argue that by performing the Poisson regression more frequently, the margin of error can,

in theory, be decreased to the declared mark of 2% for the "fastest" category. Finally, the results suggest that two of the Gas Oracle categories are not frequently used in practice: fast and average categories. It is indeed reasonable to expect that, to save money, single users or companies could set different requirements in terms of interval time to add a transaction, that is not provided by all the default categories.

This study is therefore relevant from a users' perspective for at least two reasons:

- It shows the EthGasStation oracle is less reliable than advertised. Therefore, users or companies employing Ethereum Blockchain technology should be more careful to trust oracles' recommendations.
- It suggests that two of the four categories proposed by the Gas Oracle may not meet the needs of Ethereum users. Therefore other categories could be created better suit the users' requirements. For example, a company might be interested to record a transaction with a maximum delay of 24 hours. Such a category, not considered by the Oracle, might help the company to save money in transaction fees.

The rest of the paper is organized as follows. Section II described the Transaction Pool and the Gas Oracle investigated in the paper. Section III presents the experimental design and methodology used to collect and analyze the data. Section IV discusses the results of the case study using EthGasStation Oracle. Section V describes the threats for the validity of the study. Section VI presents the related work. Finally, Section VII presents the conclusions and outlines future work ideas.

II. Background

In this section, we provide background information needed to better understand our study.

A. Block

The blockchain is an ordered list of blocks, where each block is identified by its cryptographic hash and a progressive number named "height" [1]. Each block refers to the block preceding it, resulting in a chain of blocks. Each block consists of a set of transactions. Once a block is created and attached to the blockchain, the transactions in the block cannot be changed or reverted. This is to ensure the integrity of the transactions and to prevent the double-spending problem [9].

The Block Time is defined as the interval of time the blockchain takes to mine a block. The time interval is not constant and changes every time a block is added to the blockchain. In Ethereum Blockchain the Block Time is expected to be between 10 to 19 seconds (with an average of 15 seconds). A standard unit of measurement for time in the blockchain is not the second but the block number.

B. Transaction Pool

Each node in the Ethereum network has a virtual place named "Transaction Pool", where transactions enter when they are received from the network or submitted locally. The Transaction Pool contains all currently known pending/unconfirmed transactions. They exit the Transaction Pool when they are included in the Ethereum Blockchain. The miners separate processable transactions, which can be added to a block, and future transactions, which can be wait to be added. Transactions move between those two states over time as they are received and processed.

Each node maintains its own Transaction Pool. When a node receives a new valid block, it removes all the transactions contained in the block from its Transaction Pool as well as the transactions which attempt to double spend the same output. A double spend is a potential flaw in a digital cash scheme in which the same single digital token can be spent more than once. The node can decide different policies: for example, if the Transaction Pool size gets too close to the RAM capacity, the node can set up a minimal fee threshold. Transactions with Gas price lower than the threshold are immediately removed from the Transaction Pool and only new transactions with a Gas price high enough are allowed to enter the Transaction Pool.

C. Gas Oracle

An Oracle is a software that finds and analyses data concerning real-world facts. Based on the data, it computes an estimate, extracting relevant information to predict future data trends. Examples of real-world facts are commodities and goods prices, flight or train delays. In the Ethereum Blockchain, the information provided by an oracle can be used by Smart Contracts the participants have agreed on, to execute the transactions.

In the context of this paper, Gas Oracle is an oracle that analyses blockchain data to predict the best Gas price to pay for a transaction to be approved within a certain number of blocks. We analyze data from one specific Gas Oracle, EthGasStation.

This oracle claims that all predicted values are estimations based on the current network condition and should be used as an indication. More specifically, EthGasStation employs a Poisson Regression [5] on Ethereum data to estimate the Gas prices accepted by the miners. However, it only computes and updates its predictions every 100 blocks (approximately 1,500 seconds or 25 minutes). Therefore, the estimation made by the oracles every 100 blocks may not reflect the most current status of the network. During this interval of time, some data regarding the blockchain network such as the number of miners, the number of transactions and the Gas price attached to the transactions might indeed suddenly change, thus having an impact on the value of the minimum transaction fee accepted by the miners to include a transaction in a block. All data regarding the blockchain network are publicly available, also in the form of a timeline chart confirming our hypothesis.[1]

[1] https://etherscan.io/charts

TABLE I: RESTful Services list

Resource name	RESTful API service URI
EthGasStation	https://ethgasstation.info/json/ethgasAPI.json
Block	https://api.blockcypher.com/v1/eth/main/blocks/0
Unconfirmed Transactions	https://api.blockcypher.com/v1/eth/main/txs

III. Experimental Design

We planned to test the reliability of a Gas Oracle by looking over real data from the Ethereum blockchain. The research method consists of four phases: (a) Retrieving Data, (b) Cleaning Data, (c) Modelling Data, and (d) Analyzing Data.

A. Retrieving Data

In this phase, we collect the data by making requests to various HTTP RESTful API services at different times. Figure 1 describes the periodic polling used to get new information from the HTTP RESTful API services. The flow collects data as follows:

- a request is sent to the server every 15 seconds;
- if the request is successful, the server responds to the client request sending a payload in JSON format;
- if the request is not successful, the client will not record any data for that time frame. During the data retrieving operation, an average of 1 request out of 20,160 requests was unsuccessful, i.e. 0.0049% of the requests failed.

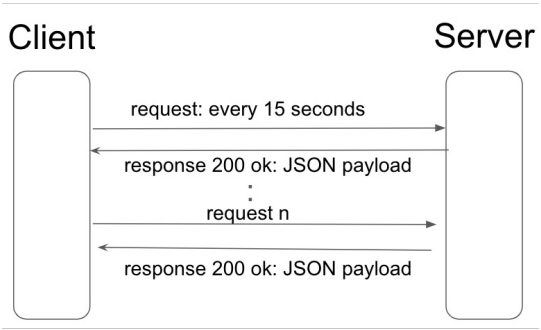

Fig. 1: Regular Polling every 15 seconds

Table I shows the URI of the RESTful API services used to fetch the Gas Oracle data, the Blocks data and the Unconfirmed Transactions data, i.e., the latest transactions that have not been included in any block.

The data have been stored as files in JSON format in the file system of the server where the analyses are performed.

```
{
    "fastest":116.0,
    "fast":100.0,
    "safeLow":17.0,
    "average":60.0,
    "block_time":13.24,
    "blockNum":8937688
}
```

Fig. 2: JSON payload extracted from EthGasStation RESTFul API Services

Figure 2 shows an example of the Gas Oracles payload formatted in JSON format.

The key value pairs shown in code 2 represent respectively the Gas to pay to have the transactions confirmed within: 2 blocks (fastest), 5 blocks (fast), 20 blocks (average), and 100 blocks (safe low).

B. Cleaning Data

In this phase, we perform a control of the data quality. The data retrieved have been checked in compliance to the API documentation; Example of data not in compliance to the API documentation are:

- string value where a numeric value was instead expected;
- numeric value where a string value was instead expected;
- numeric value which is not in the expected range;
- date value which is not in the expected time frame;
- missing value;
- missing key/value pairs;
- numeric value with different units of measurement.

We rejected the data, falling in one of the categories listed above, except for the latest category where the values have been recalculated according to the expected measurement unit.

As an example, a numeric value which is not in the expected range, is a negative block's height value, a date value in the future, or a negative value of the waiting time for transactions to be added to the Ethereum Blockchain. According to Kanda and Shudo [6], a negative value of the waiting time variable may suggest a transaction propagation delay among different nodes. This means that different nodes in the blockchain could see the transaction at different instants of time.

During the cleaning phase, 0.83% of data has been rejected, distributed as follows: 770K out of 11M transactions (0.75%), 182 out of 345K blocks (0.05%) and 112 out of 345K Oracle's predictions of the Gas price (0.03%).

C. Modelling Data

1) Validation Condition: In this step, we define the condition to assess the correctness of the Oracles' Gas price prediction. Oracles usually make the prediction based on the history of the mined blocks data, such as the lowest Gas price accepted by the miner to add the transaction to the block.

Suppose that during the time interval when the i-th block, B_i, is mined:

- op (Oracle Price) is the price predicted by the Gas Oracle to have the transaction included at the most within j blocks;
- $B = \{B_{i+1}, ..., B_{i+j}\}$ is the set of j blocks mined in the blockchain following the i-th block;
- $T = \{tx_1, tx_2, ..., tx_n\}$ is the set of unconfirmed transactions, i.e. that have not been included in any blocks, with Gas price respectively of $tp_1, tp_2, , tp_n$, that there were in a Transaction Pool when the i-th block was mined.

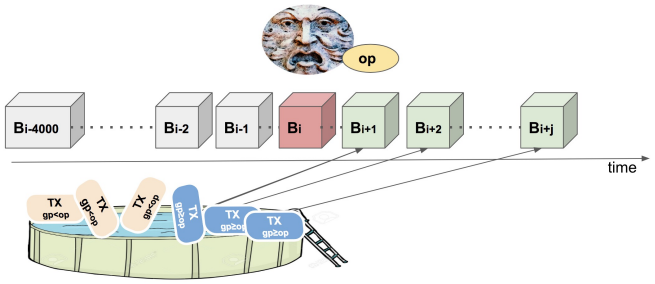

Fig. 3: Gas Oracle prediction based on block history. The transaction having a Gas price higher or equal to op (the yellow circle) are displayed in white text on a darker background. The transaction having a Gas price lower than op are displayed in black text on a lighter background.

Figure 3 depicts the Ethereum Blockchain in the past (grey color $B_{i-4000}...B_{i-1}$), with the addition of the block mined in the present (red color B_i) and with future blocks (green color $B_{i+1}...B_{i+j}$). The Gas Oracle prediction is based on the block history of the last 4,000 blocks represented in grey color. The swimming pool represents the status of a Transaction Pool at the time when the Gas Oracle makes the prediction (red block B_i). All the transactions belonging to the set T that have a Gas price higher or equal to op (the yellow circle) are displayed in green color to the right of the B_i block. If the prediction of the Oracle were correct, all the transactions belonging to the set T having a Gas price higher or equals to op, would be mined in one of the following j blocks $\{B_{i+1}, ..., B_{i+j}\}$.

The condition is expressed by the following equation:

$$\forall tx_i \in T \wedge tx_i \geq op : tx_i \in \{B_{i+1}, ..., B_{i+j}\} \qquad (1)$$

Equation 1 is used to verify the prediction of the EthGasStation Oracle Gas price.

2) Data Modelling: In this step, data were collected and stored in a relational database, where each table represents the following items: blocks, transactions, and Oracles. Even though, for this study we only considered data from one Oracle, we are planning to expand this research to include other Oracles in the future.

Figure 4 shows the data contained in the database, the relationships between table fields and their types (ex. string, integer, boolean, enumerate). It is noteworthy that the wait time for a transaction is not stored in the database but calculated by the difference between confirmed (time) and received (time).

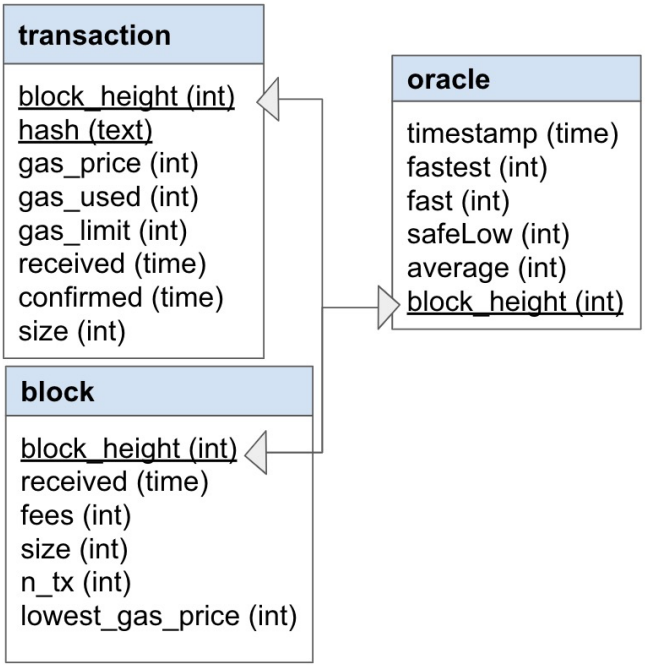

Fig. 4: Database schema.

D. Analyzing Data

In this phase, we analyse the data to:

1) view some aggregated statistical metrics such as percentile, mean, standard deviation, mode of the numerical data series;
2) find distribution of different variables such as: gas_prices and time a transaction needs to wait before being recorded in the blockchain;

The goal is to make a descriptive analysis of the data-sets concerning different items, as modeled in the Section III-C. All the data cover a period of time ranging from March 29, 2019 to May 28, 2019.

IV. Case Study: EthGasStation

In this section, we present our case study by analyzing oracle data from the EthGasStation. The data-set analysed in the paper is publicly available at our open-access repository.[2] The dataset is an SQLite database having three tables. The first table, named "transaction", contains more than 11 millions rows. The second table, named "block", contains around 345 thousand blocks. The last table, named "oracle", contains 345 thousand rows of Oracles' predictions for the Gas price of each category (fast, fastest, average and safe low). The dataset refers to a period of time of two months, ranging from March 29, 2019 to May 28, 2019.

A. Transactions Data Analysis

Figure 5 shows the box plot of the waiting time in seconds before a transaction is added to the Ethereum Blockchain. The

[2]http://doi.org/10.5281/zenodo.3584242

violin plot shows the presence of a time peak along the vertical axis at the value of 20 seconds, which means that, according to the data-set II analysed in the paper, most transactions wait from one to two blocks before being added to the Ethereum Blockchain.

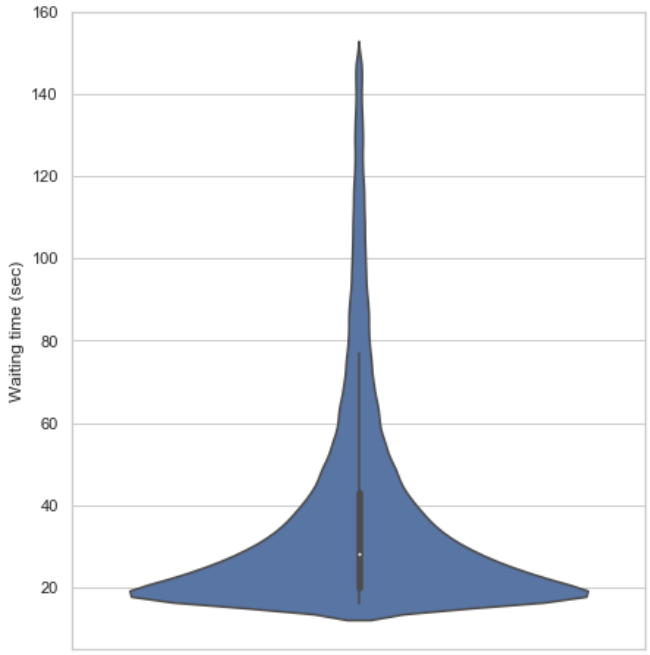

Fig. 5: Violin Plot (median, first and third percentiles, range) of the waiting time in seconds before a transaction is added to the Ethereum Blockchain.

Figure 6 shows two violin plots of the waiting time of the transactions for different Gas prices. Interestingly, the violin plots suggest that the Gas price attached to the transaction influences the interval of time the transaction needs to wait before being added to the Ethereum Blockchain. The violin plots also present the same peak at the value of 20 seconds regardless of the Gas price. The difference is the shape of the violin plot, which becomes larger at the decreasing of the Gas price. In addition, having the Gas price higher than 10 Gwei does not guarantee that the transaction is added to the blockchain within 1-2 blocks, but the probability is anyway higher when compared to the transactions having the Gas price lower than 10 GWei.

B. Gas Oracle Data Analysis

We analyzed the data of the Oracle EthGasStation. This Oracle can diversely predict the Gas price values to attach to transactions to have the transaction included at the most within n blocks.

Table III reports the mean, the standard deviation (SD), the minimum (min), the first quartile (25%), the median (50%), the third quartile (75%) and maximum (max) of the Gas price recommendation for the transaction to be included at most in two blocks according to EthGasStation.

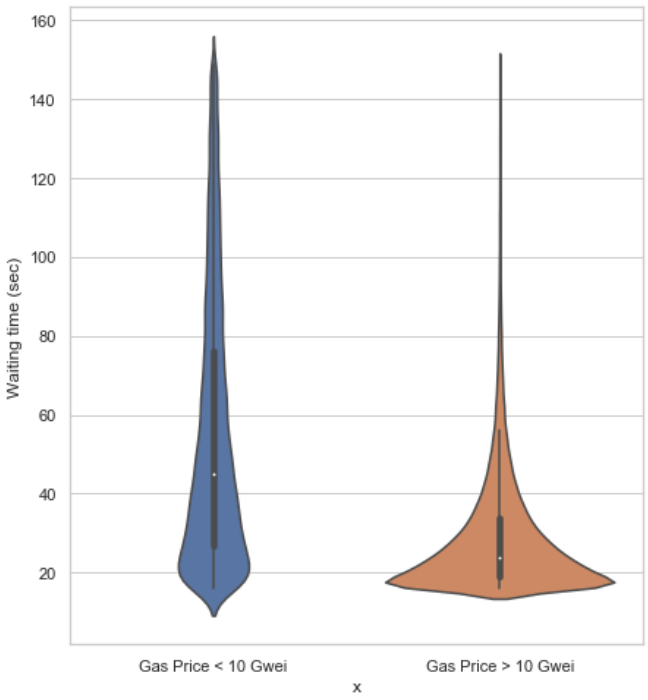

Fig. 6: Violin Plot of the waiting time in seconds before a transaction is added to the Ethereum Blockchain. The blue plot to the left refers to the transactions having a Gas price lower than 10 GWei. The orange plot to the right refers to the transactions having a Gas price higher than or equal to 10 GWei.

Figure 7 shows the violin plots of the Gas price prediction according to the EthGasStation Oracle for each Gas price category: 1) fastest (in blue, leftmost plot), 2) fast (in orange, second plot), 3) average (in green, third plot), and 4) safe low (in red, rightmost plot). The violin plot shows that the most frequent value of Gas price for each category is as follows: 20 GWei for the fastest category, 5 GWei for both the fast and average category, and 3 GWei for the safeLow category.

Table IV reports the mean, the standard deviation (SD), the mode, the minimum (min), the first quartile (25%), the median (50%), the third quartile (75%) and maximum (max) of the Gas price recommendation for the transaction for each Gas price category.

The mode for the ”fast” and ”average” categories are the same. This indicates that most times the Gas price is the same for both categories, in spite of being different categories in terms of execution time. Therefore, our analysis suggests that these two categories should be merged.

It is not possible to be sure that the users who submit the transactions are following the Gas Oracles' recommendation. However, it is reasonable to assume that the users who set the Gas price equal to the Gas price suggested by the Oracle are indeed following the Oracle's recommendation. Even if they were not following the Oracle's recommendation, it is likely that the user agrees with the Gas price attached to the transaction and the waiting time. If the user disagrees with the

TABLE II: Statistical description of Transactions data

	Mean	St.D	Mode	Min	25%	50%	75%	Max
waiting_time (s)	44	82	25	0	25	29	38	1,499
gas_price (GWei)	32	443	50	0	10	20	50	313,734
gas_used	70,124	320,908	21,000	0	21,000	21,969	49,993	8,000,000
gas_limit	303,967	947,926	21,000	21,000	42,000	70,000	150,000	8,000,030
size (Byte)	191	499	–	83	112	114	174	31,791

TABLE III: Statistical description of EthGasStation prediction on the Gas price recommendation (in GWei) for the transaction to be included at most in two blocks.

Mean	St.D	Min	25%	50%	75%	Max
15.28	6.60	3.0	10.0	20.0	20.0	61.0

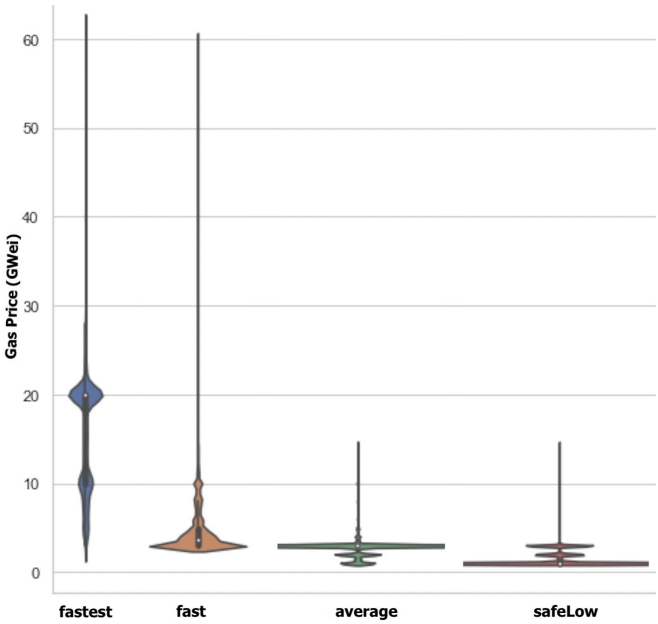

Fig. 7: Violin plot of the EthGasStation Oracle's Gas price categories

waiting time, s/he would change the Gas price to rely on the expected waiting time.

The analysis of the Gas price of the transactions in the Transaction Pool shows that 16% of the transactions have a price equal to the price suggested by the Gas Oracle. The percentage of transactions having the Gas price equal to the Gas price suggested by the Oracle is distributed among the four categories as follows: 1) 7% safe low, 2) 1% fast and average, and 3) 8% fastest.

Figure 8 presents the percentage of Gas price categories used in Ethereum Blockchain. The data analysis of the transactions waiting in the Transaction Pool to be added to the Ethereum blockchain suggests that the categories "fast" and "average" are not followed by most users probably because these categories do not respond to their need. On the other hand, the categories "fastest" and "safe low" are much more used in practice.

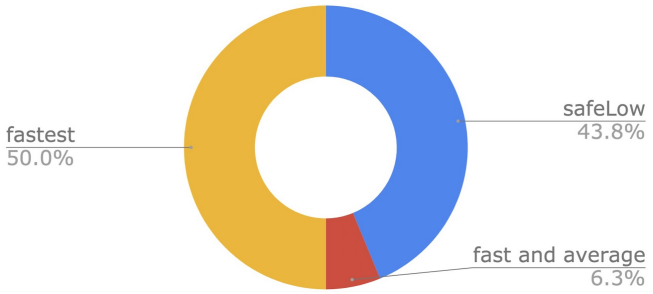

Fig. 8: Usage of Gas Oracles Categories.

We used Equation 1 to calculate the margin of error of the EthGasStation predictions. EthGasStation claims to have a 2% margin of error. Table V shows the percentage of error among the Gas price categories recommended by the Oracle. As we can see in Table V, the margin of error for every category is greater than 2%, and the "fast" category shows the greatest margin with 28% margin. Therefore, EthGasStation predictions may be less reliable than advertised.

C. Discussion: How to Improve the Margin of Error?

One of the reasons for the EthGasStation Oracle to have such a margin of error is because it performs its calculations (a Poisson Regression) to update the Gas predictions every 100 blocks (approximately 25 minutes). We argue that 100 blocks is not an appropriate interval of time for an Oracle to update its recommendations. Especially in the Ethereum blockchain, where the Gas prices can vary a lot within minutes.

In future related work, we aim to show that, by performing the Poisson Regression at more frequent intervals, it is possible to improve the accuracy of the prediction of the minimum Gas price to pay to have the transaction executed in a given time lapse.

V. Threats to Validity

External Validity. In this paper, we analyzed data Gas predictions from the EthGasStation Oracle. Since Gas is a concept unique to the Ehtereum blockchain platform, this study cannot be generalized to other blockchain platforms. We did not address this threat for two reasons: (i) this paper is an exploratory study on one specific Oracle; and (ii) analyzing other types of transaction fees for different blockchains is outside the scope of this research.

TABLE IV: Statistical description of EthGasStation Gas price categories (in GWei)

.

	Mean	**St.D**	**Mode**	**Min**	**25%**	**50%**	**75%**	**Max**
Fastest	15.33	6.60	20	3	10	20.0	20.0	61.0
Fast	4.58	2.42	3	3	3	3.6	5.0	60.0
Average	2.83	0.80	3	1	3	3.0	3.0	14.5
Safe Low	1.34	0.68	1	1	1	1.0	1.1	14.5

TABLE V: Percentage of error among the Gas prices recommended by EtherGasStation according to Equation 1.

	Margin of error			
	Fastest	Fast	Average	Safe Low
EthGasStation	4%	28%	7%	5%

Construct Validity. We could not find how the oracle originally calculates its margin of error. Therefore, the oracle may indeed have the margin of error it claims under its method. However, since such method for error calculation is not openly available, we need a properly defined one for comparison. Therefore, we defined the equation 1 to measure the margin of error of the Oracle's predictions.

VI. Related Work

Singh and Hafid [10] proposes a more fine-grained classification model that splits the confirmation time of transactions into eight classes: within 15 seconds, within 30 seconds, within 1 minute, within 2 minutes, within 5 minutes, within 10 minutes, within 15 minutes and within 30 minutes or longer. We know that on average, a transaction has to wait for two block confirmations ($\sim$30 seconds) before being added. However, in the cases where the model would predict that the transaction belongs to the "within 5 minutes" class, there is no way for the user to know if it would take 3 minutes, rather than 4 minutes or more. Hence, while the paper presents a model with good prediction accuracy, it considers confirmation time prediction as a simple classification problem. It can only provide a user with an approximation of time it would take for their transaction to be confirmed, which may or may not always be ideal. In addition, Singh and Hafid [10] compare the performance of two machine learning regression models (Multi-Layer Perceptron and Random Forest) and the more classical, statistical model (Poisson Regression) on the task of predicting the confirmation time for a transaction in Ethereum Blockchain. The authors suggest that machine learning regression models perform well and better than the already used statistical approach. However, due to the need for the model to be periodically retrained and the time taken by the model to learn new data, the two machine learning regression models are not the most viable solution for the confirmation of the time prediction, as the users may need to know the Oracles' response in a much shorter time interval.

Pierro and Rocha [8] investigated the factors that influence the Ethereum transaction fees and the possible resulting decision-making behaviour of Ethereum Blockchain users, miners included. They observed that the past history of the Oracle Gas price prediction is useful to predict the number of waiting transactions, even though the converse is not true. The results of the Pearson correlation test showed that they are instead inversely correlated: when the Oracle price increases, the number of waiting transactions in the Ethereum network decreases. It stands to reason that when the oracle suggests a high price to pay, the users wait to submit a transaction, thus decreasing the overall number of pending transactions in their memory pools.

Chen et al. [2] identified seven gas costly patterns that are not optimized by the Solidity compiler. The authors analyze 4,240 contracts on three gas costly patterns. Their results show that over 80% of the contracts suffer from those costly patterns. The authors' work differs from ours because they focus on detecting possible waste of gas units, while in this paper, we focus on the gas price (and the Oracles predictions for such price).

Ducasse et al. [3] proposed an open-source platform for blockchain analysis called SmartAnvil. Although SmartAnvil is intended to be independent of a specific blockchain platform, their work focus on Ethereum blockchain and contracts written in the Solidity language. For that reason, the authors have plans to include Gas optimization and estimation on SmartAnvil in the future, and they argue the importance of Gas estimation for contract development and analysis.

VII. Conclusion

The present study evaluated the validity of the prediction the EthGasStation oracle makes on the Gas price to pay to have the transaction recorded in the Blockchain.

We analyzed the oracle's predictions and have assessed that it carries a higher margin of error than originally claimed. EthGasStation claims to have a 2% margin of error, while our analysis shows that margin to be at least twice as much. For instance, the "Fastest" category showed a 4% margin of error, while the "Fast" category showed 28%. We argued the reason for that higher margin of error is because it cannot take into account changes in the Ethereum Network in real-time. We also argued that by updating the Gas price recommendations at every new block (instead of every 25 minutes), the margin of error can be lowered considerably.

The data analysis also indicates that the categories "average" and "fast" are not very used in practice, with less than 1% of the transactions set the Gas price suggested by the Oracle in those categories.

As future work, we plan to analyze data on other Gas Oracles besides the EthGasStation, to see if our findings also occur in other oracles. We are also going to implement our oracle to assess the feasibility and reliability to update the Gas recommendations more frequently.

ACKNOWLEDGMENT

This work is supported by (a) the Fonds de la Recherche Scientifique-FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO) under EOS Project 30446992 SECO-ASSIST (b) Flanders Make vzw, the strategic research centre for the manufacturing industry.

REFERENCES

[1] BUTERIN, V. A next generation smart contract & decentralized application platform. *Ethereum White Paper* (2014), 1–36.

[2] CHEN, T., LI, X., LUO, X., AND ZHANG, X. Under-optimized smart contracts devour your money. In *Saner'17 - Early Research Achievements* (2017).

[3] DUCASSE, S., ROCHA, H., BRAGAGNOLO, S., DENKER, M., AND FRANCOMME, C. Smartanvil: Open-source tool suite for smart contract analysis. In *Blockchain and Web 3.0: Social, Economic, and Technological Challenges*, M. Ragnedda and G. Destefanis, Eds., Routledge Studies in Science, Technology and Society. Taylor & Francis, 2019, ch. 13.

[4] ETHEREUM FOUNDATION. Solidity documentation release 0.6.12, jan 2020.

[5] GREENE, W. Models for counts of events. In *Econometric Analysis*, 7th ed. Pearson Education, 2011, ch. 18, pp. 802–828.

[6] KANDA, R., AND SHUDO, K. Estimation of data propagation time on the bitcoin network. In *Proceedings of the Asian Internet Engineering Conference* (New York, NY, USA, 2019), AINTEC '19, ACM, pp. 47–52.

[7] LUU, L., CHU, D.-H., OLICKEL, H., SAXENA, P., AND HOBOR, A. Making smart contracts smarter. In *CCS'2016 (ACM Conference on Computer and Communications Security)* (2016).

[8] PIERRO, G. A., AND ROCHA, H. The influence factors on ethereum transaction fees. In *2nd International Workshop on Emerging Trends in Software Engineering for Blockchain* (Piscataway, NJ, USA, 2019), WETSEB '19, IEEE Press, pp. 24–31.

[9] PORRU, S., PINNA, A., MARCHESI, M., AND TONELLI, R. Blockchain-oriented software engineering: Challenges and new directions. In *Proceedings of the 39th International Conference on Software Engineering Companion* (2017), ICSE-C '17, IEEE Press, p. 169–171.

[10] SINGH, H. J., AND HAFID, A. S. Transaction confirmation time prediction in ethereum blockchain using machine learning, 2019. https://arxiv.org/pdf/1911.11592.

[11] WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper. Byzantium version 7e819ec* (Oct 2019), 1–39. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf.