Fixed transaction ordering and admission in blockchains

Paulo Mendes da Silva
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Portugal
psilva@gsd.inesc-id.pt

Miguel Matos
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Portugal
miguel.matos@inesc-id.pt

João Barreto
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Portugal
joao.barreto@tecnico.ulisboa.pt

ABSTRACT

Blockchain cryptocurrencies such as Ethereum offer a secure and decentralized transaction system and have the potential to replace legacy financial transaction systems. Despite their potential, they suffer from transaction ordering and admission problems. These stem from having miners deciding the transaction execution order, as well as which transactions are admitted in the blockchain. Transaction censorship, transaction removal due to double-spending attacks and long transaction commit delays are some of the resulting problems. In this work in progress, we will propose a Fixed Transaction Ordering and Admission (FTOA) algorithm to mitigate these problems, using EpTO logical timestamps in a Byzantine setting. We focus on Ethereum, but the general ideas can be applied to other blockchains.

CCS CONCEPTS

• **Computer systems organization** → **Peer-to-peer architectures**; • **Networks** → *Network protocols*;

KEYWORDS

Distributed systems, blockchain, total order

1 INTRODUCTION

Blockchain is the technology underpinning cryptocurrencies like Bitcoin and Ethereum [6]. Blockchains have become increasingly important for a broad range of applications such as medical records and asset ownership [4]. A blockchain is a distributed ledger that securely stores a permanent and verifiable history of confirmed transactions in a sequence of blocks. It relies on a peer-to-peer network of computation nodes known as *miners* [11]. No central trusted party is required [6]. Miners periodically generate blocks, roughly every 10 minutes in Bitcoin [6] and every 12-15 seconds in Ethereum [11]. When a miner generates a block, it gets a reward. In order to generate a block, miners are required to solve a probabilistic computation puzzle called "proof-of-work" (PoW) [6]. The solution to this puzzle is stored in the block header and can be easily verified by other miners for consensus and block admission into the blockchain [6]. A transaction list is also included in each block. Miners decide the transaction execution order when creating a new block to be mined [11]. This work considers open permissionless blockchains only. These allow any node to participate, whereas permissioned blockchains restrict participating nodes [2].

Blockchain forks may occur when multiple distinct versions of the next block are successfully mined and propagated. Consensus on the set of transactions is temporarily broken and then re-established by using the longest chain, that is, the one with the highest number of blocks [4].

Ethereum is a popular and groundbreaking cryptocurrency, offering complex transactions using programs residing in the blockchain known as smart contracts [11]. Ethereum transactions may include coin transfer between accounts and also smart contract execution updating internal contract state. Miners get an additional variable reward for executing such transactions.

Problems. Several transaction ordering and admission problems have been identified in blockchains. We will focus on Ethereum, but the general ideas can be applied to other blockchains. Ethereum transactions are not guaranteed to be executed in the same order they are submitted, that is, later transactions can be executed before transactions submitted earlier. This means dependent transactions may have problems [9]. For example, when a later payment depends on an earlier payment, the miner that happens to mine the next block can decide to run the later payment first thus causing the earlier payment to fail due to insufficient funds.

Transaction order can even change after a transaction is seen in a block. This is the case when a double-spending attack is executed by replacing the block including the transaction with a new version of that block without that transaction. Furthermore, this action can go undetected since the old version of the block is not stored [9].

In the case of fork, a transaction included in a block of a losing forked branch can be discarded. Despite offering mechanisms to consider such a transaction again for inclusion in a block, Ethereum offers no guarantee that it will always be re-included in a future block, since miners may silently drop it. This means submitted transactions are not guaranteed to be executed. On the other hand, no miner can be sure that every other miner has dropped a given transaction, meaning there is no upper bound for the time a transaction can live in the system without being executed [11].

Due to the unpredictability that phenomena like the above-mentioned ones cause, blockchain clients are typically conservative and only consider a transaction as *committed* after receiving a sequence of consecutive blocks mined after the first block that included it. The length of such a sequence of blocks is usually 12 blocks in Ethereum. As a consequence, most Ethereum transactions usually take more than 3 minutes to commit, after submission [11].

Miners can join a mining pool expecting sustained shared rewards upon solving PoW computation sub-puzzles distributed by the pool operator. Around 80% of the mining power in Ethereum resides in only six mining pools. Since mining pools can select which transactions to include in the blockchain, this means mining pools can potentially collude to control the network with a 51% attack and censor transactions [6].

Contribution. This work intends to make the following contribution: provide fixed transaction ordering and admission to blockchains using EpTO logical timestamps in a Byzantine setting.

The main advantages of this work will be guaranteeing the same transaction order for all miners in the Ethereum network and ensuring a valid submitted transaction cannot be discarded. As a consequence, all valid submitted transactions will be guaranteed to be executed and mining pools will not be able to censor them. Furthermore, transaction commit delay will also be reduced since transactions showing up in a mined block will be guaranteed not to be discarded.

Existing Ethereum transaction dissemination mechanisms are insufficient since they do not provide any transaction ordering or admittance guarantees. This also applies to manually attempting to submit the same transaction to multiple entry points, such as mining pools, which would also rely on these mechanisms. The current Ethereum blockchain consensus already provides a total order for transactions, which may explain why there has been no attempt to provide a new total order mechanism for enforcing miners to mine blocks with the same transactions in the same fixed order. On the other hand, timestamps are often perceived as vulnerable to manipulation attacks [10] which may explain why to the best of our knowledge they have never been used in Ethereum transactions.

We anticipate a challenging implementation. It will require changing the Ethereum dissemination mechanisms, making them more adequate for lower overall transaction commit times, at the the same time as using logical timestamps in a Byzantine setting and guaranteeing they are resistant to attacks, so as to make transaction order and admittance fairer.

2 WORK IN PROGRESS DESCRIPTION

The intuition behind this work in progress is to develop a Fixed Transaction Ordering and Admission (FTOA) algorithm for Ethereum, so as to prevent miners from deciding how to order transactions and which to include. In order to achieve this goal, the EpTO (Epidemic Total Order) algorithm will be adapted to Ethereum and its Byzantine setting by adjusting parameters (e.g. *fanout*) and adding logical timestamps to Ethereum transactions. The existing Ethereum Byzantine consensus setting will be considered for this work. It can be described as PoW-based. This means consensus can be reached as long as an adversary controls less than half of the network computing power. In such a setting, a 51% attack is possible, thereby allowing the attacker to arbitrarily manipulate the blockchain, for instance, in order to exclude transactions or modify their order [5]. The FTOA implementation using EpTO will likely not be trivial, since similar gossip based consensus systems are known to be challenging to implement [7], but our previous experience makes us confident it will be successful.

Logical timestamps will be used to order transactions. This will allow detecting missing and spurious transactions, as well as ensuring dependent transactions will be executed in the expected order. For this to happen, transactions will be disseminated and aged until there is a high probability they have been delivered to all nodes, before being considered for inclusion in a block. As a consequence, the system will converge to having all miners mining the same transactions in the same order. We do not expect any significant performance impact by adding logical timestamps to transactions and we do not anticipate any scalability issues.

FTOA will offer protection against against Denial-of-Service (DoS) attacks using malicious timestamps by restricting timestamp validity to a time window for aging all transactions for the current block. Transactions arriving outside the current time window will be discarded. Transaction timestamps will be signed be the issuing Ethereum address using PKI cryptography, thereby preventing tampering by an attacker. Miners will have no advantage in mining blocks with missing transactions since other miners will know the correct transaction set and will discard them.

2.1 Background on EpTO

EpTO is an Epidemic Total Order algorithm with probabilistic agreement. Its intuition is making events available quickly at all nodes with high probability. It guarantees processes eventually agree on total order of events with high probability. The probability of holes in an event sequence can be made arbitrarily small. Processes do not necessarily know when a hole occurs and delayed events may either be dropped or tagged as "out-of-order" when their delivery would result in an order violation. EpTO uses a logical clock, timestamps and time to live for aging events before delivering them in timestamp order to applications [8].

A balls-and-bins approach is used for dissemination. Processes are modeled as bins and events as balls. The algorithm then tries to find how many balls have to be thrown so that a bin receives at least one ball with arbitrarily high probability. Events are disseminated in a small number of rounds that increases logarithmically with the number of processes. The round duration can be set to the latency of the well-behaving nodes for guaranteeing the well-behaving part of the network will satisfy the probabilistic agreement. EpTO is fully decentralized and therefore does not require central coordination for processes. Process churn and message loss can be mitigated increasing the *fanout* parameter, which defines how many balls are sent by each process in each round and is logarithmic in the number of processes [8].

2.2 FTOA algorithm overview

The Fixed Transaction Ordering and Admission (FTOA) algorithm for Ethereum will work as follows. Logical timestamps will be associated to transactions. Ethereum nodes will have a local clock for generating transaction timestamps. When a transaction is generated at a given node, it will receive the local clock value as timestamp, in a new field. The local clock will then be incremented by one unit. Transactions will be disseminated to all nodes using EpTO. Upon receiving a transaction, nodes may update their local clocks according to the received transaction timestamp. If transaction timestamps are received in order with no holes (that is, with no missing transactions and no missing timestamp values), nodes update their local clock to the received timestamp value. On the other hand, if the received timestamp originates a hole or if there is still a hole in the transaction timestamp list after adding it to the list, the local clock is not updated. In this case, the local clock is only updated again when a new sequence of transactions with no holes results from the new timestamp being added to the list and it is assigned the value of the highest timestamp of that sequence. This

will prevent spurious timestamp attacks that would result in nodes generating transactions with arbitrarily high timestamps that could consequently be delayed for a long time.

Miners will order transactions according to their logical timestamps. Transactions will have to wait in a staging list before being considered for inclusion in a block in order to guarantee high delivery probability to all nodes and low missing transaction probability. This waiting period can be defined as aging. The aging time encompasses the sum of the durations of all EpTO dissemination rounds required for ensuring high delivery probability to all nodes. The number of dissemination rounds and the time between dissemination rounds are EpTO parameters. After aging, aged transactions will be included in a block to be mined in a fixed logical timestamp order. Miners will attempt to mine full blocks, that is, blocks with the maximum possible number of transactions, a parameter dynamically agreed upon by the Ethereum network. Miners will order distinct transactions with the same logical timestamp using a fair criterion based on the previous block number and on the Ethereum accounts that issued them.

All the above leads miners to mine the same sequence of transactions for a given block. When a block is successfully mined, it will be disseminated to other miners. In the unlikely case a block with an incomplete transaction set is received by a miner that has received all transactions, it will be discarded due to its missing transactions. When 51% of network is well-behaving and all nodes have received all transactions with high probability, no blocks with missing transactions are expected to be accepted into the blockchain since miners will discard them and will converge to adopting blocks containing the complete transaction sets.

Miners will stop mining their current block and start mining a new one with a more completed transaction set when previously unknown transactions are received. The new transaction set will be the union of the old transaction set with the new received transactions. Transactions can be received during individual transaction dissemination or within a transaction list of a disseminated mined block.

As an example, by setting the EpTO round duration to the average Ethereum 120 ms latency value [3], having blocks carrying 95 transactions, considering a 12 second block mining time and a 10 second block propagation time, we can estimate a transaction commit time around 40 seconds. This is substantially lower than the usual 3 minutes commit time and includes aging all 95 transactions included in the block and the time required to fill the block up with those transactions, considering an observed Ethereum average generation of 11.5 transactions per second.

2.3 FTOA attack mitigation

Ethereum transaction fees offer protection against Denial-of-service (DoS) attacks. An attacker might try to overwhelm the network by continuously issuing new transactions. However, such a sustained attack is considered to be too expensive [1] and can be estimated to cost around 250 USD per block full of transactions, as of December 2017. Therefore, as an example, issuing 1000 blocks of transactions would have an estimated cost of 250000 USD for the attacker. This is also valid for an attacker attempting to issue transactions with malicious timestamps. A transaction timestamp can be within or outside the interval defined by the earliest and latest transaction timestamps to be included in the current block. Timestamps in the past will be discarded. Transactions with timestamps in the future will be queued until all consecutive timestamps in between are received. In case an attack is attempted with timestamps within that interval, the corresponding transactions will be accepted and ordered with the remaining valid transactions, in case the aging time of the oldest transaction with an equal or higher timestamp has not yet elapsed. In case that aging time has elapsed, transactions with lower or equal timestamp are discarded, since they should already have been received with high probability. The only short-term advantage for the attacker will be having transactions executed slightly before other valid transactions during this aging time. As it happens today with Ethereum, no long-term advantage can be foreseen from flooding the network with transactions as it would be too expensive.

Furthermore, an Ethereum address will only be able to provide transactions with monotonic increasing timestamps. This means an attacker wishing to overload the network with transactions with a repeated timestamp, for instance, would have to use different Ethereum addresses for each transaction. An attacker will also not be able to tamper with the transaction timestamps, since timestamps will be signed by the Ethereum address issuing the transaction using PKI cryptography. Thus, an attacker will only be able to produce new transactions. On the other hand, miners will have no advantage in mining blocks with missing transactions since other miners will know with high probability there are additional transactions and will discard those blocks.

2.4 Status and future directions

This work is at an early stage. The algorithm definition is still in progress and there is no implementation yet. As short term directions, we anticipate to finish the algorithm definition, in particular on how to adjust EpTO to the Ethereum Byzantine setting. Then we expect to implement a prototype to validate the algorithm and finally to implement it in Ethereum. In the long term, we will evaluate the solution suitability for other blockchains such as Bitcoin.

ACKNOWLEDGMENTS

This work is supported by the Portuguese National Science Foundation under Grant No. SFRH/BD/130017/2017.

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A survey of attacks on Ethereum smart contracts (SoK). In *International Conference on Principles of Security and Trust*. Springer, 164–186.

[2] Christian Cachin. 2016. Architecture of the Hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

[3] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. 2018. Decentralization in Bitcoin and Ethereum Networks. *arXiv preprint arXiv:1801.03998* (2018).

[4] Lucianna Kiffer, Dave Levin, and Alan Mislove. 2017. Stick a fork in it: Analyzing the Ethereum network partition. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 94–100.

[5] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. 2017. A survey on the security of blockchain systems. *Future Generation Computer Systems* (2017).

[6] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. 2017. SMART POOL: Practical Decentralized Pooled Mining. *IACR Cryptology ePrint Archive* 2017 (2017), 19.

[7] Francisco Maia, Miguel Matos, José Pereira, and Rui Oliveira. 2011. Worldwide consensus. In *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 257–269.

[8] Miguel Matos, Hugues Mercier, Pascal Felber, Rui Oliveira, and José Pereira. 2015. EpTO: An epidemic total order algorithm for large-scale distributed systems. In *Proceedings of the 16th Annual Middleware Conference*. ACM, 100–111.

[9] Christopher Natoli and Vincent Gramoli. 2016. The blockchain anomaly. In *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*. IEEE, 310–317.

[10] Marko Vukolić. 2015. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*. Springer, 112–125.

[11] Ingo Weber, Vincent Gramoli, Alex Ponomarev, Mark Staples, Ralph Holz, An Binh Tran, and Paul Rimba. 2017. On availability for blockchain-based systems. In *Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on*. IEEE, 64–73.