Coinspermia: A Cryptocurrency Unchained

Tom Portegys
Ernst & Young LLP
901 104th Ave NE, Bellevue, WA, USA 98004
tom.portegys@ey.com

***Abstract*—The latency and throughput of blockchain-based cyrptocurrencies is a major concern for their suitability as mainstream currencies and as transaction processors in general. The prevalent proof-of-work scheme, exemplified by Bitcoin, is a deliberately laborious effort: the time and energy required to mine blocks makes the blockchain virtually immutable and assists in the consensus-reaching process. *Coinspermia* (coin=money + spermia=seed) is a different approach: transactions are concurrently seeded throughout a network of peer nodes to an extent sufficient to achieve a high reliability of essential currency operations, including the fast transfer of coins from an owner to a recipient, and the prevention of double spending. A number of Bitcoin features are retained in Coinspermia, including transaction input-outputs and cryptographic addresses and signing, but no special proof-of-work is required to commit transactions. Instead, a client can be assured of an operation completion when a quorum of network nodes acknowledge the operation, which can occur before a transaction operation finishes propagating through the network. Simulation substantiates improved latency and throughput.**

Keywords—Cryptocurrency; Bitcoin; peer-to-peer network; distributed transactions; quorum system; blockchain

I. INTRODUCTION

To spend a Bitcoin [1], the owner is required to verify that the coin exists and belongs to the owner. This entails producing a pointer to a precedent transaction that pays the coin to the owner. This coin must also not exist as the input to another transaction, as this constitutes a double spending condition. These transactions are to be found in the blockchain, a data structure replicated over the Bitcoin peer-to-peer (P2P) network nodes.

In typical blockchain-based cryptocurrencies [2], blockchain growth is a deliberately laborious effort called mining. The time and energy involved in mining a block is required to make the chain virtually immutable and to assist in reaching a consensus about the contents of the blockchain among peer nodes. This mining effort, known as proof-of-work, is rewarded in Bitcoin with a fee. An unfortunate consequence of mining is the relatively slow throughput committing transactions into the blockchain, which is in the realm of 7 transactions per second. The transaction latency is also reported to average over 40 minutes [3]. These speeds are a serious drawback for a large-scale transaction processing applications [4].

BigchainDB [5] is an effort to marry blockchain with conventional database transaction processing with the goal of achieving transaction throughput rates comparable to the latter while retaining the durability of blockchain.

Iota [6] aims not only to improve throughput by harnessing transaction originators to validate other transactions, but also to reduce the transaction processing fees that make conventional blockchains less attractive for micro-transactions. Micro-transactions are expected to be a staple of the Internet of Things (IoT). Iota does this by forgoing blockchains entirely. Instead of using a sequential chain and separating the network into users and validators, it uses a directed acyclic graph architecture call a Tangle [7], which makes users and validators one and the same.

Kadena [8] improves performance by committing transactions through a majority vote from participating nodes, bypassing mining altogether. Kadena is an example of a private blockchain with a permissioned user model that many in the blockchain community frown upon, as these blockchains are administered by centralized agencies.

This cryptocurrency model, Coinspermia (coin=money + spermia=seed), is based on a P2P network of nodes, as in Bitcoin. Coinspermia's coins are called pipcoins (Fig. 1). Transactions are also replicated in ledgers among the nodes. A chained transaction crypto-scheme is used that is similar to Bitcoin. Transactions are also standalone objects in a directly addressable space of identifiers. Transactions are "committed" by achieving a consensus among a quorum of peer nodes.

One of the primary aims of Coinspermia is to analyze and simulate possible transaction latency and throughput performance improvements. It is therefore in this initial incarnation a stripped-down system that simply serves the purpose of transferring coins from owners to recipients.

Fig. 1. Coinspermia logo.

II. DESCRIPTION

A. Basic Method

Coinspermia is composed of a decentralized P2P network of randomly connected nodes. The connection randomness and density are crucial factors that will be discussed below.

Transactions record the process of transferring coins from one owner to another. Appendix 1 presents the format of a transaction. Similar to Bitcoin, transactions have one or more input and output components that specify coins and addresses.

Also as in Bitcoin, a particular source of coins must be entirely consumed, which constitutes a "rolling" account in a transaction chain.

Transactions are collected in a ledger (see Appendix 2 format) that is fully replicated throughout the network. To economize memory and boost speed, only transaction outputs that contain unspent coins are kept in fast memory (see unspent transaction outputs). Full transactions are committed to an archive for audit and restoral purposes.

To execute a transaction, the inputs must match to precedent unspent transaction outputs that are found in the ledger using universally unique identifiers (UUIDs) as keys. As in Bitcoin, the precedent output address must validate the signature of the input. The input and output coin quantities must also be equal. If the transaction is valid, the matching precedent outputs are removed from the unspent store and the new outputs are stored.

B. *Transaction Processing*

A client sends a transaction to a random initial network node. The initial node selects a quorum of other network nodes (see, quorum selection method below), and forwards the transaction the quorum with the expectation of discovering either matching precedent transactions or the valid absence of them. To achieve reliable performance, the probability of encountering such conditions must be high. The quorum uses an algorithm such as *Raft*, *Paxos*, or *ZooKeeper* to reach consensus about the consistency of transactions.

Once the quorum is formed, transaction processing is as follows:

1) Check for other pending transactions in the quorum that affect the transaction. These could be a double spending transaction or a transaction that is transferring coins to the current transaction. In this case, suspend the transaction for a random period before repeating the check. This will allow one or more of the pending transactions to commit or abort.

2) When there are no conflicting pending transactions, check if the transaction is supported by precedent committed transactions. This will be the case when coins are properly owned by the transaction. If this is the unanimous condition, then commit the transaction to the entire network. If it is unanimously unsupported, then abort the transaction. If there is not unanimity, then record the current state of the precedent transactions and wait a specific time for precedent transactions to completely propagate through the network.

3) After the wait, check again for pending transactions as in Step 1. If no pending conflict, check if the inconsistent committed state has been resolved. If so, commit or abort the transaction. If the state is still inconsistent, but in a different way from the state recorded in Step 2, assume another transaction is propagating and go to Step 2. If the previous state is the same, then the network is not consistent. The quorum must then decide by consensus whether the transaction is valid or not. If valid, the transaction is committed to the entire network, overriding conflicting ledgers.

C. *Quorum Intersection Probability*

The network connection distribution must be sufficient to guarantee that transactions propagate to all the nodes. The network must also constitute a quorum system [9], meaning it ensures a high probability of a quorum intersection for dependent transactions.

Fig. 2 depicts the quorum for a "blue" transaction that requires a match to a transaction in the nodes immediately updated by a precedent "green" transaction. For example, a blue client may want to spend a coin obtained from a green client. In this case, the red node is a directly connected peer involved in both the blue and green quorums. Once found, the blue transaction evaluation will reflect its presence and potentially force the blue quorum to wait for the green transaction to propagate to the blue quorum nodes. In the case of a double spend, the red node will not contain the precedent transaction to support the double spending blue transaction.

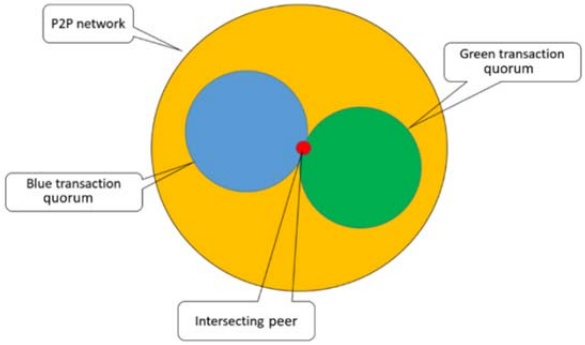

Fig. 2. Intersecting transaction quorums.

Eventually, the transactions propagate through the entire network. However, to achieve low latency, a client is notified of a transaction completion when its quorum acknowledges the operation. For this to work reliably, the probability of finding an intersecting node must be very high.

Consider the probability of finding at least one of K items in N nodes with S samples:

$$P = 1 - \left(\frac{\binom{(N-K)}{S}}{\binom{N}{S}} \right) \quad (1)$$

where:

N: number of nodes

K: number of nodes containing search item (precedent green transaction)

S: number of random samples (current blue transaction)

P: probability of finding item.

Fig. 3 shows how densely connected nodes must be in the network to achieve 99%+ success of finding a matching transaction. It can be seen that as the network grows, the density of connections also grows, but at a significantly lower rate. See Abraham and Malkhi [9] for a deeper analysis. For this reason, the conjecture is that a network that is sufficiently connected might support a much lower latency and higher throughput than a Bitcoin-like network.

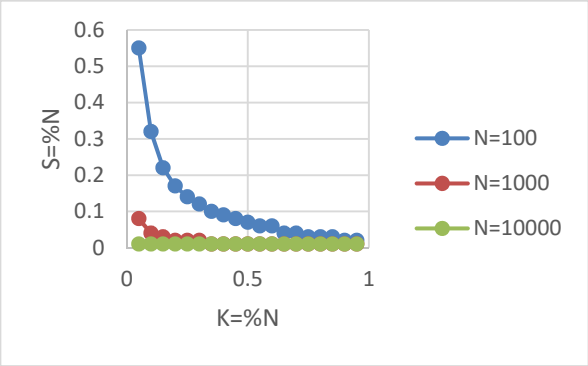

Fig. 3. 99% success: K and S as % of N.

The topology of the network is also of importance since the existence of clusters could lower the probability of finding an intersecting node. This might call for a distributed process to inventory and reshuffle connections between nodes, which could also serve to thwart network tampering.

Once the desired connectivity is determined, the density of the network is given by the graph theory definition: |C| / (|N| * (|N| − 1)), where C is the number of connections and N is the number of nodes.

D. *Quorum selection*

Any P2P network must accommodate the presence of faulty and dishonest nodes. And at some degree of compromise, consensus becomes impossible in any network [10]. In addition to the quorum intersection probability described above, the selection of a quorum must sample the nodes sufficiently to reduce the probability of selecting a set dominated by dishonest or faulty nodes. This can be expressed as a confidence interval:

$$p \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{n}} \qquad (2)$$

where:

p = probability of honest node

n = quorum size

α = desired confidence

$z_{1-\alpha/2}$ = "z value" for desired level of confidence

A simple method for selecting a quorum based on the existence of a set of known nodes is as follows:

1) Let the node initially receiving the transaction select a seed for a commonly accepted random number generator. This might be from a global time source or some random process.

2) Randomly select a quorum from the set of network nodes.

3) Send the transaction to the nodes and execute the consensus process.

4) If the transaction is valid, propagate it and the quorum information to the remainder of the network. Each node can then verify whether the quorum was correctly selected.

E. *Serial Transaction Delivery*

By forwarding transactions serially, it is guaranteed that at a node will be validly updated. For example, suppose node N1 has neighbors N2 and N3 who forward transactions A->B, B->C to it concurrently. The first A->B transaction will be accepted, regardless of origin. The next A->B will be dropped since it will be discovered as a duplicate. The same will occur with the B->C, regardless of who is first, N2 or N3. The point is that there could be many interleaving's that are valid. When a transaction is dropped, it also is not forwarded, preventing transmission cycles.

III. SIMULATION

An implementation has been developed that is derived from a file-sharing package that supports broadcasting files into a P2P network [11]. Its GUI can be seen in Fig. 4 and the code is available at https://bitbucket.org/portnoid/coinspermia.

Simulation data was collected using a network of 100 nodes each randomly connected to 20 other nodes. Care was taken to ensure the network was fully connected. The total number of connections was thus 2000 and the probability of intersecting immediate transactions was over 99%.

The simulated times to transmit a transaction to a connected node and to process a transaction within a node were set to 1. This was chosen to approximate a millisecond unit of time. The quantity of clients and the mean transaction origination time were independently varied. Specific transaction originations were drawn from a random distribution centered on the mean.

Coins were minted and deposited in client wallets and node ledgers to allow money transfer transactions. A transaction consisted of transferring a single coin from one client to another. When a client originated a transaction, a random recipient client and originating node were selected. Transactions were serially executed, meaning a client waited for an active transaction to complete before originating a new one.

The total number of transactions, mean transaction latency, and mean node transaction queue length were measured. A client was informed of a transaction's completion when the transaction's originating node and its immediate peers updated their ledgers. This defined the latency time. The transaction continued asynchronously to propagate to all the other nodes in the network. No attempt was made to model fraudulent transactions in this simulation, as only performance was of interest at this time.

Each setting of the independent variables was run 10 times under different random initial conditions. Each simulation run was for 900,000 time steps, or 15 simulated minutes. Tables 1, 2, and 3 show the results for 500, 1000, and 1500 clients, respectively. Overall, the latency remains less than a simulated second. However, overloading the network retards throughput significantly due to the work backlog in the nodes.

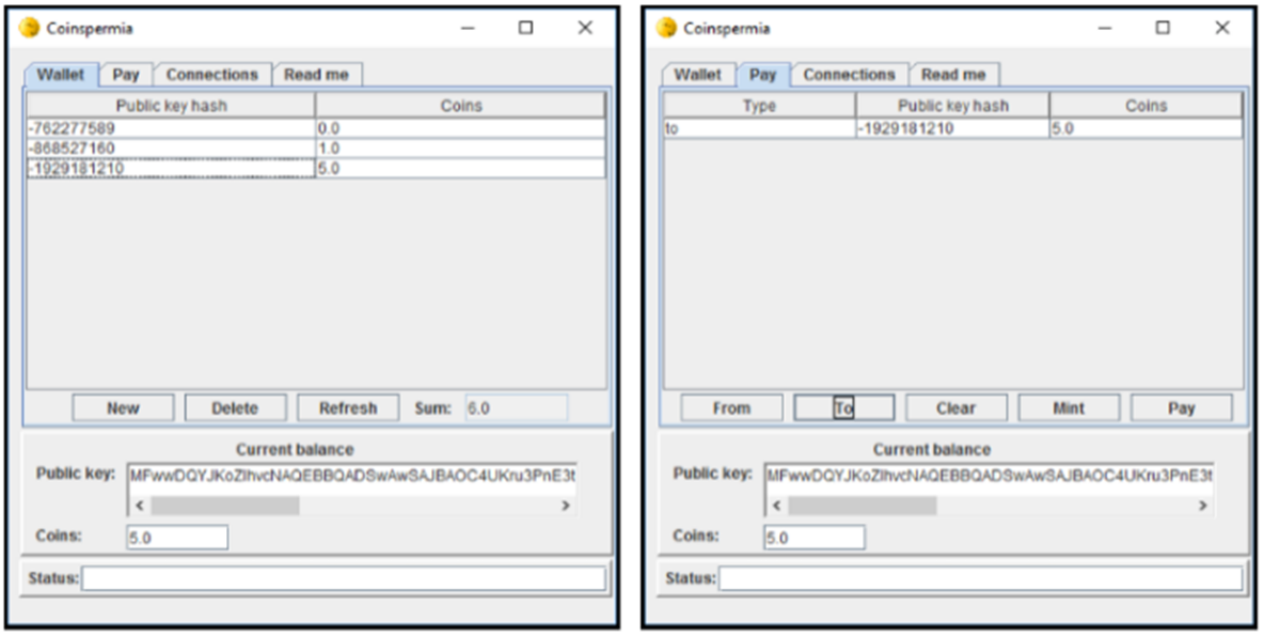

Fig. 4. Coinspermia GUI.

TABLE I. 500 CLIENTS

Origination mean	Transactions	Latency	Queue length
300000/5 minutes	1291	5.2	0
180000/3 minutes	2236	5.3	0
60000/1 minute	6918	6.1	0

TABLE II. 1000 CLIENTS

Origination mean	Transactions	Latency	Queue length
300000/5 minutes	2596	5.4	0
180000/3 minutes	4582	5.7	0
60000/1 minute	4584	150	7527

TABLE III. 1500 CLIENTS

Origination mean	Transactions	Latency	Queue length
300000/5 minutes	3909	5.6	2
180000/3 minutes	6827	6	18
60000/1 minute	5186	68	24475

IV. CONCLUSION

This design is streamlined for space and speed. It foregoes blockchains as a means of validation and completely relies on quorum consensus. It would conceivably be more suitable for the more numerous small transactions such as are executed during the normal course of living: buying food, paying bills, etc. If the transaction is for a large sum, the quorum size could be a function of the transaction size. There could be hybrid schemes as well for additional security that involves mining immutable blocks for large transaction amounts, for example, the purchase of a house or an exchange of funds for a large business contract.

REFERENCES

[1] Nakamoto, S. “Bitcoin: A Peer-to-Peer Electronic Cash System”. 2008.

[2] Coursera. “Bitcoin and Cryptocurrency Technologies”. https://www.coursera.org/learn/cryptocurrency 2014.

[3] Gilbert, D. “Bitcoin’s Big Problem: Transaction Delays Renew Blockchain Debate”. http://www.ibtimes.com/bitcoins-big-problem-transaction-delays-renew-blockchain-debate-2330143. 2016.

[4] Young, J. “R3 Appears to Admit Defeat, Stops Blockchain Development”. https://themerkle-com.cdn.ampproject.org/c/s/themerkle.com/r3-admits-defeat-stops-blockchain-development/amp/. 2017.

[5] McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., Granzotto, A. “BigchainDB: A Scalable Blockchain Database”. https://www.bigchaindb.com/whitepaper/. 2016.

[6] Varshney, N. “The Internet of Things, Blockchain-less Token IOTA Launched: Interview with Co-Founder”. https://cointelegraph.com/news/the-internet-of-things-blockchain-less-token-iota-launched-interview-with-co-founder. 2017.

[7] Popov, S. “The tangle: some aspects of a blockchainless cryptocurrency”. https://www.docdroid.net/xdARu5z/tangle-paper-rev.pdf.html. 2016.

[8] Samman, G. “Evolution of Kadena, the First Real Private Blockchain”. http://www.coindesk.com/evolution-kadena-first-real-private-blockchain/. 2016.

[9] Abraham I., Malkhi D. “Probabilistic Quorums for Dynamic Systems”. In: Fich F.E. (eds) Distributed Computing. DISC 2003. Lecture Notes in Computer Science, vol 2848. Springer, Berlin, Heidelberg. 2003.

[10] Lamport, L. “Byzantizing paxos by refinement”. In Proceedings of the 25th international conference on Distributed computing (DISC'11), David Peleg (Ed.). Springer-Verlag, Berlin, Heidelberg, 211-224. 2011.

[11] Portegys, T. E. Spores: a Push and Pull Peer-to-Peer File Sharing Approach. The 2004 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04) http://tom.portegys.com/research.html#spores. 2004.

APPENDIX 1 – TRANSACTION FORMAT

```
Transaction:
{
 # List of precedent transaction inputs.
 Inputs:
 [
  Input:
  {
   # ID of precedent transaction output.
   # This is used to access the ledger.
   Precedent transaction output ID: UUID
   # Signature of ID.
   ID signature: Signature
  }
  ...
 ]
 # List of transaction outputs.
 Outputs:
 [
  Output:
  {
   # Output address.
   Output address: Address
   # Quantity of coins to output.
   Quantity: Coin
   # Transaction output ID.
   Transaction output ID: ID
  }
  ...
 ]
}
```

APPENDIX 2 – LEDGER FORMAT

```
Ledger:
{
 # Unpaid transaction outputs.
 Unpaid transaction outputs:
 [
  Output:
  {
   (Same as Transaction Output).
  }
  ...
 ]
 # Transaction archive.
 Transaction archive
 [
  Transaction:
  {
   (Same as Transaction)
  }
  ...
 ]
}
```