Multicast Implementation in Permissioned Interoperable Blockchain

Hans Tananda
School of Computer Science and Engineering

Prof. Lam, Kwok Yan
School of Computer Science and Engineering

Assoc. Prof. Wang Huaxiong
School of Physical & Mathematical Sciences

Abstract - This project is performed to implement a permissioned interoperable blockchain, that is aimed to enhance the privacy and performance of the currently available blockchain, so that it can be suitable for government and business sector. The blockchain is implemented using UDP multicast network to ensure the privacy of the transaction. In addition, it uses a voting mechanism to verify the transactions on the network, which is observed to be much faster than the traditional Proof of Work (POW) method.

Based on the current experiment using 3 nodes and a notary node, the time needed to verify each transaction is 25 milliseconds on average. Moreover, each node only used less than 10 MB of memory, hence it is very likely that the application would be able to be deployed into small IOT devices.

Keywords - Permissioned Blockchain, Privacy, Performance

1 INTRODUCTION

1.1 DEFINITION

Based on the Bitcoin Wiki page [1], blockchain is a database of transactions that is shared by nodes participating in the network running the same protocol. Every node contains a full copy of every transaction in the blockchain, thus one can find out how much value to any addresses at any point in history. The currency used in these transactions is called cryptocurrency.

If we look at Ethereum instead, it is said as not just a [2] cryptocurrency, but it can also be considered as decentralized computing resource, where developers can build decentralized applications (called "Dapps") that can gain the benefits of cryptocurrency and blockchain. Those applications are run in the system as "smart contracts".

If we look further into another project named Hyperledger, which is a project from Linux Foundation, it sees blockchain as a framework to create other applications that leverage technologies such as consensus and membership services with a modular architecture [3].

In this project, Blockchain is considered as a list of records, called blocks, which are linked using cryptography.

1.2 CHARACTERISTICS

Blockchain is characterized by the fact that every node participating in the consensus will store all transactions history. Therefore, if any party want to alter any recorded transaction, it will actually have to alter it in every node participating in the consensus. This makes the immutable characteristic of blockchain [4], [5].

Since all nodes in the network participate in the verification, it is also considered as decentralized, which can help in protecting the blockchain itself from attack and collusion [6].

Looking further, we can define blockchain as two different types, permissioned and permissionless blockchain [6]. In permissionless blockchain, each transaction generated is distributed to the network, typically via P2P, then it will be included in a block alongside with other transaction. Every node will receive a copy of the transaction and take part in the consensus protocol to update the blockchain.

On the other hand, in permissioned blockchain, only authenticated and authorized participants can take part in the consensus protocol to update the blockchain. It will also only allow nodes to join the network once identity and role are defined.

There are also many kinds of consensus algorithms used to verify the transaction itself. However, it can be classified mainly as two different types, which are "Proof-of-Work (PoW)" and "Proof-of-Stake (PoS)". Proof-of-Work is also known as mining, where there are some nodes running cryptographic algorithms and competing to get the verification hash based on the requirements given [4]. Currently, PoW is used to verify more than 90% of the total market capitalization in cryptocurrency markets [7].

Meanwhile, Proof-of-Stake is using the coins owned by the nodes themselves to be used in the verification mechanism [8].

2 BACKGROUND

The current designs of most blockchain have a major privacy risk as each node will receive copies of every single transaction due to the peer to peer broadcasting, thus anyone can see all the information in it [4].

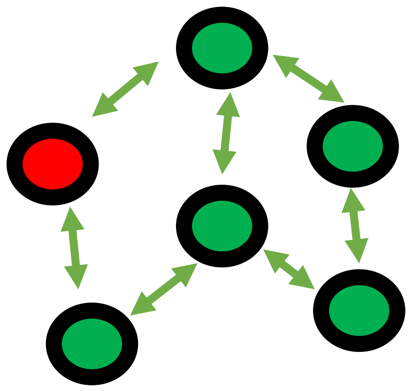
Fig. 1: Typical blockchain implementation

In the ideal scenario, only nodes trusted by the users should receive any information or take part in the consensus protocol to update their internal blockchain.

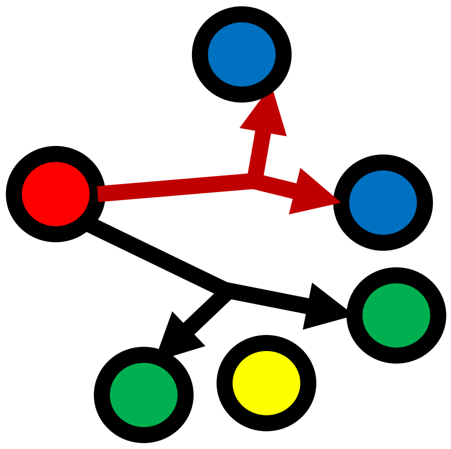
Fig. 2: Current implementation of multicasting

Aside from that, most blockchain implementations currently use PoW consensus, which can take hours to verify a transaction [9].

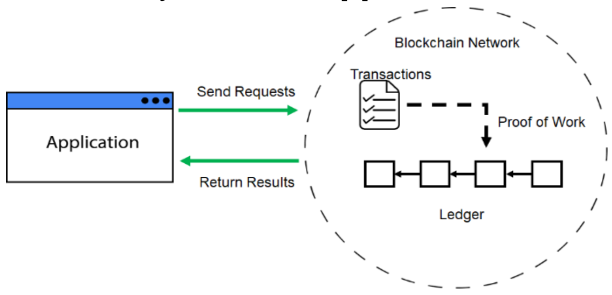

Fig. 3: Typical public blockchain protocol

Moreover, it consumes a lot of and even specialized hardware to run the algorithm [7] [9].

Thus, this experiment will utilize one of the various consensus protocols available, which have been tested to be able to perform a decent verification throughput with significantly less power than typical PoW protocols [9] [10].

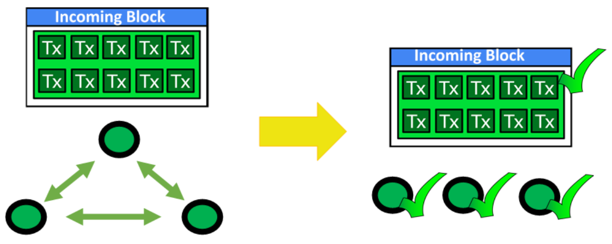

Fig. 4: Consensus protocol illustration

3 METHODOLOGY

3.1 GENERAL IMPLEMENTATION

Most of the implementations are based on simplified bitcoin implementation by Jeiwan [11]. Therefore, the addresses used for the transactions in the blockchain uses the same implementations in bitcoin, which is using the Elliptic Curve Digital Signature Algorithm (ECDSA) which uses elliptic curve cryptography. Furthermore, it is encoded in Base58 to make it more human-readable, thus it will look something like “1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa”.

The main changes introduced from the original implementation is the underlying network protocol used in the blockchain as well as the consensus mechanism used in transaction verification.

3.2 PACKET SENDING

Since the project aims to protect the privacy and business transaction data of the users, it utilizes a private interoperable blockchain fabric based on multicasting network protocol. This allows users to select nodes they trust and generate a private blockchain using those nodes.

This is possible since IP Multicasting sends the network packet only once and reaches every destination without unnecessary duplication in the network [8]. Therefore, nodes that are not supposed to receive the packet will not even know whether such a packet exists.

Each command sent from one node to other nodes will generate a packet sent over the network. Each packet consists of 2 sections, namely command and data. The command length is 12 bytes. Meanwhile, the size of the data is currently arbitrary. However, the size of the packet is currently limited to 8192 bytes.

3.3 NODES SPECIFICATION

Each node will have its own unique identifier so that they can be easily identified during the consensus protocol. It can be done using a public key from asymmetric cryptography, thus node authenticity can be checked as well. However, in the implementation of this experiment, ID must be generated manually and then saved in the environment variable of each node.

There are two types of node, the normal node and the notary node. The normal nodes are the ones that create the transactions and take part in the transaction verification protocol.

Meanwhile, the notary node is the mediator of those nodes. It tallies the verification results from each node and sends it back once the voting is finished. Thus, it does not take part in the transaction voting itself.

3.4 TRANSACTION GENERATION AND VERIFICATION

In the current implementation, any node can generate a transaction via the command line. There are two types of transactions, which are coinbase transaction and transfer transaction. Coinbase transaction is a transaction where a new amount of currency is introduced to the server.

After a transaction is generated, other nodes within the same multicast group will receive the transaction. Upon receiving the transaction, it will be stored in the memory, then those nodes will send a multicast message to request for voting.

When a notary node receives a voting request, it will transmit a message to all nodes in the multicast group.

Afterward, those nodes will verify the validity of the transaction by looking towards the UTXO list. Each node will send a message regarding the verification results, then the notary node will tally those results. The notary node will inform all the nodes in the multicast group regarding the tally result after more than half of the nodes in the network reaches the same conclusion (either accept/reject).

By doing this, it is actually possible for each node to have a different list of transactions in the blockchain and have multiple blockchains running on the same protocol.

3.5 OTHER IMPLEMENTATIONS

To simplify the implementation, instead of saving the blocks into files, currently, it is stored into a simple key-value database file. Furthermore, to simplify the testing, the number of transactions stored in a block is only 10 instead of 3,500 transactions, which is the theoretical limit of transactions stored in a bitcoin block.

The source code of the blockchain implementation is available at my GitHub repository: https://github.com/hanstananda/blockchain_ureca.

3.6 TESTING METHODOLOGY

All nodes are simulated within a computer due to the resource limitation given in the laboratory. Each node is run within a shell; thus 4 shell instances are created to simulate all of the nodes, consisting of 3 normal nodes and 1 notary nodes.

The computer specification used to test the application is as follows:

Memory: 16 GB

Processor: Intel i5-8600K

Storage: SSD 860 EVO M.2 250GB

Operating System: Ubuntu Linux 18.04

(Note: The platform should also work on Windows, but it is not tested)

To simulate the generation and verification of the transaction, 3 addresses are generated for each node.

Then, a script is used to initialize the blockchain and the database. After that, another script is used to simulate a transaction sending from one node to the other nodes.

4 RESULTS AND DISCUSSION

4.1 PERFORMANCE EVALUATION

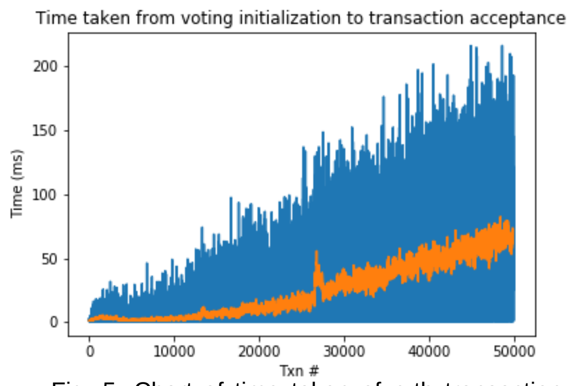

Fig. 5: Chart of time taken of n-th transaction from voting initialization to transaction acceptance

The blue line in the figure 5 denotes the time taken for the n-th transaction to be accepted by the blockchain. The yellow line is the moving average of the time taken. It can be seen here that as the number of transactions increases, the deviation of the time taken to accept a transaction increases linearly. This is because of the fact that there are some transactions that refer to some other transactions that are pretty old, thus more time is taken to retrieve them as the number of blocks increased.

Table 1: Statistics of transaction verification duration

Mean	24.7416895 ms
Standard Deviation	33.0989205386367 ms
Number of accepted transactions	50000

This is confirmed when the statistics of the transactions are generated. Among 50000 transactions, the average time taken to verify a transaction is 24 ms, yet with a standard deviation of 33 ms.

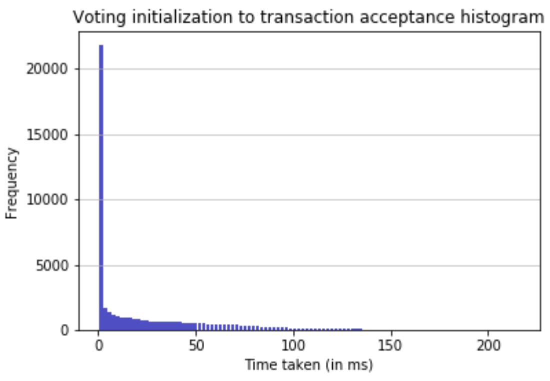

Fig. 6: Chart of voting initialization to transaction acceptance histogram

It is shown in the histogram above that most of the transactions are verified within 10 ms. However, there are some transactions that took more than 100 ms to verify.

The memory usage of the application itself is very light, only 10 MB each.

4.2 KNOWN LIMITATIONS

Due to the fact that the deployment of the multicast systems has a very high dependency on the hardware, especially routers, it may be challenging to be used in public network. However, since permissioned blockchain will most likely be used in a closed environment, it should still be possible to deploy the nodes for the blockchain.

Moreover, the performance of the blockchain will also depend on the network itself. Latency, traffic, increased number of nodes can degrade the performance in the system. These parameters were not specifically tested during the experiments due to resource limitations.

Since the results are tested locally, it does not simulate network delay which may occur in the real-world environment.

4.3 FUTURE PERFORMANCE OPTIMIZATION

It is possible to add some additional fields in the transaction, which will be stored in the blockchain. If we take this further, it can be used to implement smart contracts that can be executed by all nodes in the network.

Performance optimization can be further obtained by using blocks saving implementation used in typical public blockchains like bitcoin.

4.4 POTENTIAL APPLICATIONS

In supply chain and industry sector, this can be used to automate the transaction recording and verification with the linked and immutable nature of the ledger. Furthermore, we can verify the authenticity of each transaction. Additional business rules can also be encapsulated easily on top of the base blockchain itself.

In the banking and finance sector, it can help in payment and assets transferring without requiring any complex arrangements beforehand.

5 CONCLUSIONS

This paper assesses the performance of the consensus algorithm implemented in a permissioned blockchain that is using multicast instead of a peer-to-peer network to communicate from one node to other nodes. The results show that the transactions can be verified quickly, thus it is possible to have a significant throughput with such blockchain implementation. With several more tweaks in the database systems, it is possible to have better performance with less deviation in time required for transaction verification. Moreover, since the memory usage is considerably light, it is possible to run the nodes in on embedded systems.

This study has shown two important aspects. It provides data in the research of consensus algorithm for permissioned and private blockchains, thus will be useful for future optimization of the consensus algorithm itself. Aside from that, this paper has also shown the possibility of using a multicast network to be implemented in blockchain platforms. In addition, this experiment also shows the limitation of using such method in blockchain implementation.

In future work, I would like to evaluate other consensus algorithms and other performance optimizations that can be implemented to the blockchain protocols. Another direction that deserves further study is the use of other network protocols to be implemented in the blockchain platforms.

ACKNOWLEDGMENT

This work was done under the supervision of Assoc. Prof. Wang Huaxiong and Prof. Lam Kwok Yan.

Thank you for the support from my mentor Elmo Huang Xuyun who teaches me during my research experience.

This work was done in collaboration with URECA member Bedi Jannat who works in similar project as mine.

The blockchain platform was implemented based on the Bitcoin implementation in Golang created by Ivan Kuznetsov.

We wish to acknowledge the funding support for this project from Nanyang Technological University under the Undergraduate Research Experience on CAmpus (URECA) programme.

REFERENCES

[1] Bitcoin, “Bitcoin wiki,” Bitcoin, 28 October 2018. [Online]. Available: https://en.bitcoin.it/wiki/Block_chain. [Accessed 23 June 2019].

[2] Ethereum, "Ethereum," Ethereum, [Online]. Available: https://www.ethereum.org/beginners/. [Accessed 23 June 2019].

[3] Linux Foundation, "Hyperledger Fabric," Linux Foundation, [Online]. Available: https://www.hyperledger.org/projects/fabric. [Accessed 24 6 2019].

[4] C. HOLOTESCU, "Understanding Blockchain Opportunities and Challenges," in *The 14th International Scientific Conference eLearning and Software for Education Bucharest,*, Timisoara, 2018.

[5] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Bitcoin.

[6] D. Dob, "Permissioned vs Permissionless Blockchains: Understanding the Differences," Blockonomi, 17 July 2018. [Online]. Available: https://blockonomi.com/permissioned-vs-permissionless-blockchains/. [Accessed 24 6 2019].

[7] Gervais, Arthur, Karame, G. O., Wüst, Karl, Glykantzis, Vasileios, Ritzdorf, Hubert, Capkun and Srdjan, "On the Security and Performance of Proof of Work Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, 2016.

[8] Bentov, Iddo, Lee, Charles, Mizrahi, Alex, Rosenfeld and Meni, "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake," *IACR Cryptology ePrint Archive,* vol. 2014, p. 452, 2014.

[9] J. Kwon, "Tendermint: Consensus without Mining," Cornell, 2014.

[10] H. Yue, L. Yi, D. Xinghua, F. Li and C. Ping, "Performance Analysis of Consensus Algorithm," in *EEE Intelligent Vehicles Symposium (IV)*, Changshu, Suzhou, China, 2018.

[11] Jeiwan, "Blockchain Go Github Jeiwan," [Online]. Available: https://github.com/Jeiwan/blockchain_go. [Accessed 1 March 2019].

[12] Z. Beichuan, J. Sugih and Z. Lixia, "Host multicast: a framework for delivering multicast to end users," in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, 2002.