A Simulation Approach for Studying Behavior and Quality of Blockchain Networks

Bozhi Wang[1(✉)], Shiping Chen[1,2], Lina Yao[1], Bin Liu[1,2], Xiwei Xu[1,2], and Liming Zhu[1,2]

[1] University of New South Wales, Sydney, NSW 2062, Australia
bozhi.wang@student.unsw.edu.au
[2] CSIRO Data61, Sydney, NSW 2110, Australia

Abstract. Blockchain, as the fundamental technology of Bitcoin, tends to be an-other technology revaluation of the Internet, due to its unique security, trustworthiness and reliability. However, due to the massive deployment and high running cost, it is hard for researchers to study the dynamic behavior of a large and alive Blockchain network, e.g., Bitcoin. In this paper, a proposal on a simulation approach to study Blockchain protocols in sizable Blockchain networks is made. First, several key requirements for software tools to simulate Blockchain are identified. Secondly, the focus is on the selection and definition of key performance metrics to quantify Quality of Blockchain (QoB). We aim to demonstrate the proposed idea by using an existing simulation tool to duplicate a simplified Blockchain Proof of Work (PoW) protocol with different parameters and observations. Our case study shows that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes and protocols.

Keywords: Blockchain · Proof of Work (PoW) · Discrete-event simulation Performance metric · Quality of Blockchain (QoB)

1 Introduction

For the past few years, people can do almost everything on the Internet. However, most of the online transactions from person to person, no matter email, money, music or anything else, rely on big intermediaries, such as email services, banks and telecommunications operators. There are growing problems with such centralized systems. For example, our privacy is under infringement; the cost of sending money overseas turns out to be too high and takes days. Things started to change in 2008, when Satoshi Noakmoto published his paper, "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. The key technology of Bitcoin, called Blockchain, shows its strength in the future Internet to enable decentralization as a new paradigm for large-scale distributed systems. While TCP/IP is the communicating protocol between computers, Blockchain is its trust mechanism and their cooperation protocol. Blockchain offers some unique capabilities/features, such as decentralized, reliability and immutability, which enabled people to establish trust and do transactions without relying on a trusted third party. Till now, Blockchain is starting to show its potentials in many interesting domains, such as transnational payment, distributed storage, food provenance and etc.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 18–31, 2018.
https://doi.org/10.1007/978-3-319-94478-4_2

Before Blockchain could be widely adopted in other areas, it still has a number of issues to be solved. First, the current mining (PoW- Proof of Work) mechanism of Blockchain are wasting a large amount of computing power on computation, which is necessary for PoW but meaningless and costly. Second, every node in the network has a complete ledger. As time passing by, the ledger will become larger and larger. And when a miner verifies a transaction, it should track all the historical transactions recorded on the blockchain. The performance issue is getting worse and worse. Then, though Blockchain is an anonymous system, all the transactions are publicly available, which may cause privacy issue. Lastly, in the current Bitcoin network, a common security strategy is to wait six blocks to confirm the transactions in the last block, which last around one hour, which significantly limits the adoption and applications of the current Blockchain technologies. With these problems, many ideas are proposed yet still under developing, such as IOTA [19], EOS.

On the other hand, most current blockchain systems are driven by electronic money. For example, the open source Etherum, once creating a smart contract, it costs some ETH, the cryptocurrency on Ethereum blockchain. When conducting some tests and/or updating, it is also very expensive and has side-effects and/or interruptions on the main blockchains. So we need an approach to testing new ideas in different scopes and sizes of blockchain networks with little costs and impacts on the existing real blockchain networks. Simulation is a good way out. It doesn't cost much, and could find out any tiny change in the system.

In this paper, we propose a simulation approach to study the large-scale Blockchain networks. First, we identify key requirements for the simulation software for implementing PoW protocols. Second, we collect and define a number of performance metrics to quantify the Quality of Blockchain (QoB). Then, we demonstrate the proposed idea using a simple simulation tool to duplicate a simplified Bitcoin PoW protocol using different configurations. Our case study shows that it is possible and practical to study a large-scale Blockchain network using simulation software.

The rest of this paper is organized as follows. Section 2 gives an overview of the PoW protocol in Bitcoin and defines some metrics for QoB. In Sect. 3, we identify the key criteria for functionality and capabilities required by Blockchain simulators. Section 4 shows a simple Blockchain simulator by leveraging SimPy. Section 5 presents some related work. We conclude the paper in Sect. 6.

2 Overview of Blockchain

2.1 Blockchain

Essentially, Blockchain is a peer-to-peer distributed ledger database, which consists of connected data blocks. The connection pointer between blocks is the Header Hash processed by cryptographic hash function that protects the transactions in every block, as well as the connected blocks. Thus, any of the historical transaction cannot be changed without invalidating a chain of Header Hash.

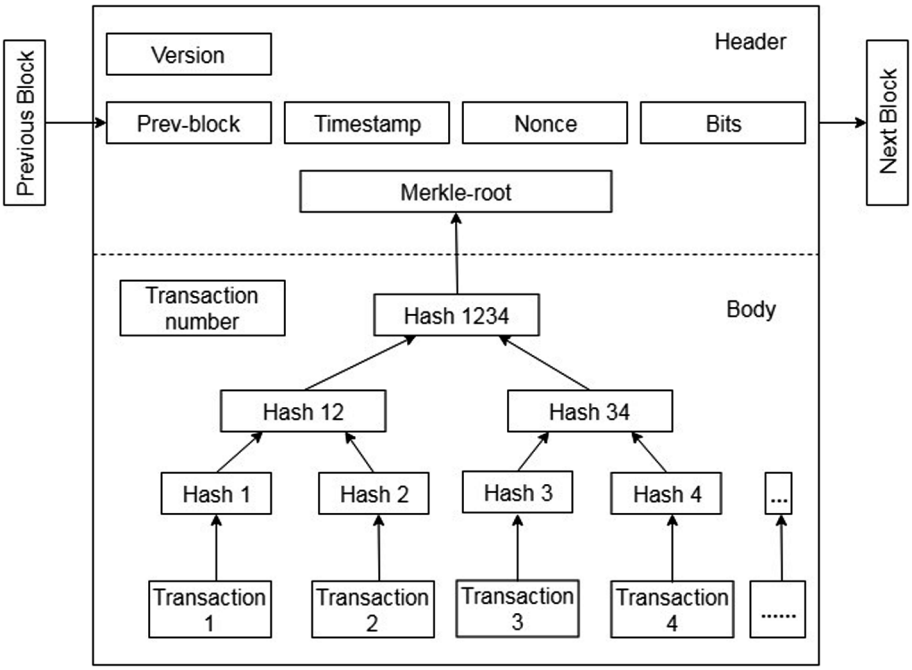

Fig. 1. The internal structure of block

In Bitcoin system, one block is created about every ten minutes. Transactions happening in the Bitcoin system are saved in the blocks. A block contains a Header and a Body, as shows in Fig. 1. The Header contains metadata of the block, such as Version, Prev-block pointing to the previous block, Timestamp, Nonce, Bits, Merkle-root etc. The body mainly includes the details of the transactions in the structure of Merkle Tree. These transactions form a publicly global ledger in Blockchain system, where the transactions recorded in the ledger could be queried by anyone who can access Internet. Every transaction is signed by a digital signature of the sender to ensure they are un-forged and not duplicated. The Merkle Tree with all the transactions has a unique Merkle-root calculated by the Hash procedure, which is recorded in the Header.

The Block is created through mining process, which is an exhaustive random number algorithm. During this process, the miners package the hash value of the previous block and all the transactions have happened in the latest ten minutes, and find a value for Nonce to calculate a hash value with 256 bits. The mining process is aimed to find the value of Nonce that makes the hash value meets some requirements, such as having a certain number of zeros in the first bits. The miner, who successfully find such a value, gets the right to write the new block to the blockchain by broadcasting the new block to the Bitcoin other nodes to verify (Fig. 2).

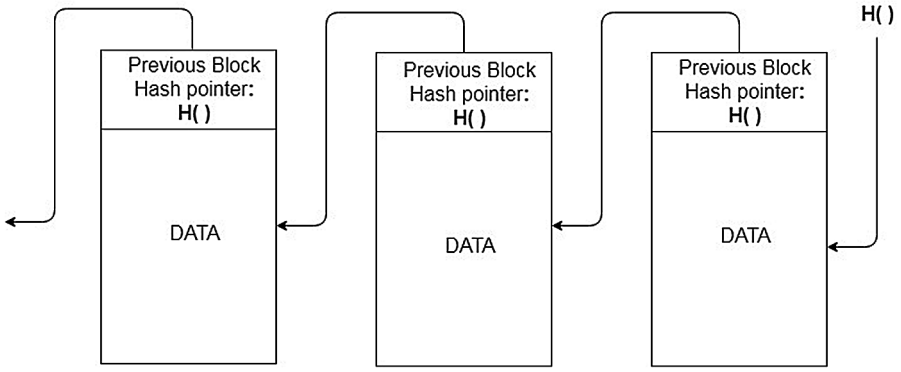

Fig. 2. The structure of blockchain

The system has a competition mechanism for miners to compete for the right of writing, which calls Proof of work (describe in the next part). During the process, the more computing power a miner spends, the larger possibility the miner can get the right. If the new block is successfully added into the blockchain, the miner will get some reward. Also, it is possible for two different miners to find new blocks almost at the same time. The two blocks might be verified and accepted by a subset of the Bitcoin network, which form a fork. In such a case, the other miners need to choose the fork which with larger number of blocks (implying heavier work).

2.2 Proof of Work Protocol in Bitcoin Blockchain

The distributed system consensus algorithm which appeared in 1980s is the foundation of Blockchain Consensus. There are several Byzantine Fault Tolerance protocols which used in different Blockchain platforms, such as PBFT (Practical Byzantine Fault Tolerance), Raft, PoW (Proof of Work), PoS (Proof of Stake), DPoS (Delegated Proof of State). Bitcoin Blockchain uses PoW (Fig. 3).

The main work of PoW is to do a lot of SHA256 hash calculations during the mining process. The process to reach consensus is discussed as below.

1. A user initiates and signs a new transaction, broadcasts to the whole network asking for accounting.
2. The miner, which receives the transaction, puts it in the Mempool.
3. For every round of computation, the miner add all the transactions in the Mempool to create the body of a new block, and works on PoW procedure, to find out one proof of work with enough difficulty.
4. The miner who wins the competition broadcasts the new block to the Bitcoin network.
5. Other miners who receive the new block verify all the transactions in the block to ensure they are all valid and new.
6. Once the new block is accepted by the whole network, it is added at the end of the blockchain.

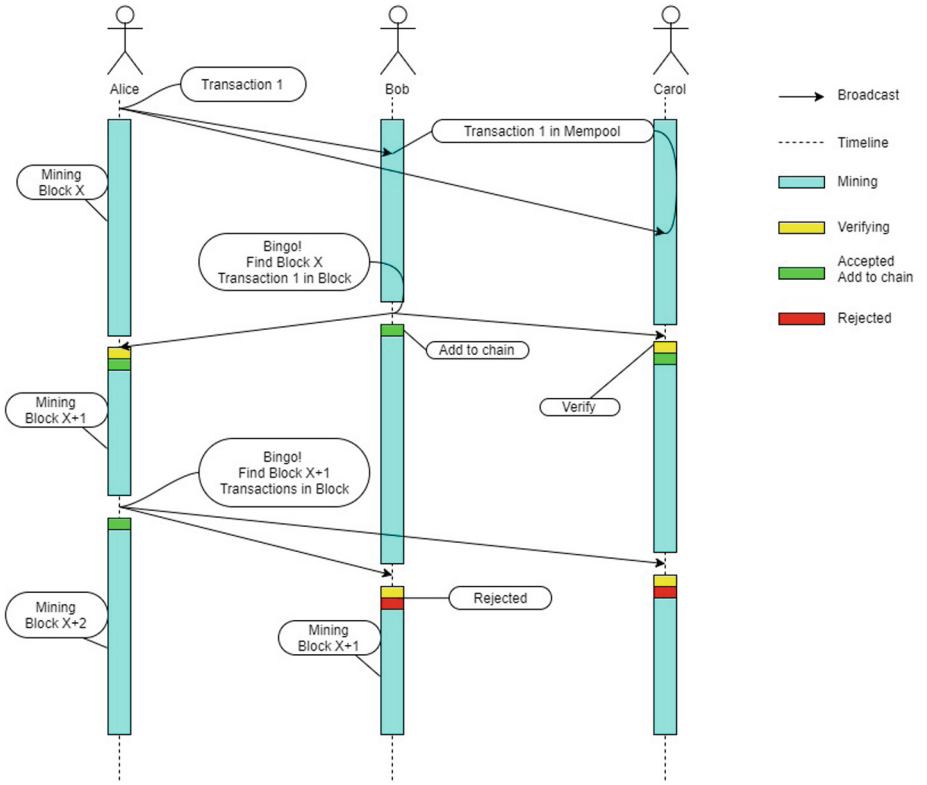

Fig. 3. Proof of work procedure

Through the mining process discussed above, the transactions from different users are written in the blockchain, which is hosted on every single node within the network. So we can get a distributed and high reliable and consistent global ledger.

2.3 Parameters and QoS Metrics for a Blockchain Network

In this section, we collect and define some metrics for QoS of Blockchain (Fig. 4). We can define and simulate a Blockchain Network using these metrics. We ignored both orphaned blocks and network delay in our simulator. It will be fixed in the future research.

TBN (Total Block Number). The number of blocks that have been mined in a specific time period
ABS (Average Block Size). The average block size in MB.
BCT (Block Commit time). The average time needed to commit a block to the main chain since being created.
TPB (Transactions per Block). The average number of transactions per block
ATS (Average Transaction Size). The average transaction size in Byte.

TCT (Transaction Confirmation Time). The average time for a transaction to be accepted into a mined block.
TPD (Transactions per day). The number of daily confirmed transactions.
MS (Mempool Size). The aggregate number of transactions waiting to be confirmed.

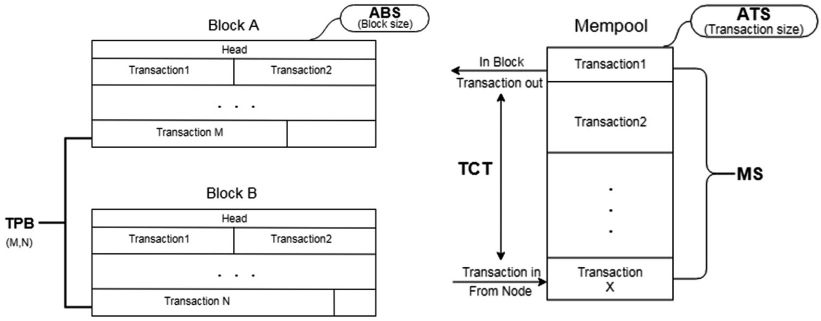

Fig. 4. QoS metrics for a blockchain network

We use these metrics to measure the quality of the Blockchain system. TBN directly related to simulation time. BCT is approximately equals to 6 times of mining time. Block Size includes the header and some transactions, and is limited by 1000000B (defined by Bitcoin source code). ABS is an indicator that shows whether the block is filled. One transaction needs to use the storage 4 times the simple message. After the update 'Segregated Witness' on 8/24/2017 [20], the extra 3 times does not calculate in Block size, which expand TPB. Transaction size range and transactions per seconds are the settings we want to change to measure the performance, which turns to be ATS and TPD after statistics. TCT shows how long one transaction is accepted, MS shows the waiting list of transactions.

In the system sight, we want the block to be filled (ABS as large as possible, TPB as many as possible), so we can transport more information at the same time. In client's view, TCT is the most important QoB, i.e. the less TCT is, the more effective the Blockchain network.

3 Requirements for Blockchain Simulation

In order to realize a Blockchain simulator, there are several requirements we should consider, as shown below:

- **Timing simulation:** The Blockchain's scale rises with time. Mining time, transaction time and network delay affect the performance of the system a lot.
- **Broadcasting:** In reality, we use multicast to realize IP communication. In the simulation, it turns out to be an ideal situation. So we need to broadcast all the transactions.

- **Event-driven:** A miner can change its status once some event happens during mining. The simulator should be event-driven to fit this pattern.
- **Message Processing:** Once a miner finds a new block, the miner broadcasts its block, at the same time other nodes receive the new block (which packaged as a message) and make its own response. So the software should support message processing.
- **Concurrency:** The Blockchain network is comprised of many doing different but similar things (mining, verifying, receiving or sending messages) at the same time. Concurrency is required as well.

4 Case Study – Simulate Blockchain Using SimPy

In this section, we demonstrate how to simulate a simple blockchain PoW by leveraging SimPy's capabilities and using the criteria above.

SimPy is a bare simulation API implementation written in Python. In SimPy, the basic simulation entities are processes. These processes can execute in parallel and may exchange Python objects among each other. Most processes include an infinite loop in which the main actions of the process are performed. The simulator is written by SimPy3.0.10 under python 3.6.0.

Computer parameter:

OS: Ubuntu 14.04 64 bit

CPU: Inter(R) Core(TM) i7-7700 CPU@ 3.60 GHz

Memory: 16G

Consider the PoW procedure we have mentioned in Sect. 2.2, our simulator is working with three processes:

1. Producing transactions. As showed in Fig. 5, we create a transaction with its size and creating time randomly, then set it in the Mempool.

```
def transaction_generator(env):
    """A process which randomly generates transactions"""

    while True:
        yield env.timeout(random.randint(TRANSAC_TIME[0], TRANSAC_TIME[1]))
        size = random.randint(BLOCK_SIZE[0], BLOCK_SIZE[1])
        global tranNO
        tranNO += 1
        transactions_list.append({'size': size, 'creatTime': env.now, 'inBlockTime': 0})
```

Fig. 5. Transaction process

2. Producing blocks. As showed in Fig. 6, we randomize a node as the miner of the block, and then set it mining time randomly. Then calculate the number of transactions could be written into the block, package all these transaction details and then broadcast. Confirm the block before six blocks if it is the branch with biggest computing power, add it into Blockchain (As we didn't consider the situation of orphaned blocks, it is just a confirmation without comparing).

```
def message_generator(env, out_pipe):
    """A process which randomly generates messages."""

    while True:
        yield env.timeout(random.randint(MINGING_TIME[0], MINGING_TIME[1]))
        nodeID = random.randint(0, NUN_OF_NODES - 1)
        # print("nodeID = ", nodeID)
        global tranFin, tranNO
        eventID = 0
        event = events[eventID]
        blockID = current_blockID[nodeID]
        current_blockID[nodeID] += 1

        weight = 0
        length = 0
        while (tranFin < tranNO - 1) and (weight + transactions_list[tranFin + 1]['size'] < BLOCKSIZE):
            length += 1
            tranFin += 1
            weight += transactions_list[tranFin]['size']
            transactions_list[tranFin]['inBlockTime'] = env.now

        block = {'owner': nodeID, 'blockID': blockID, 'state': 0, 'blockSize': weight, 'transactionNO': length,
                 'creatTime': env.now, 'power': 0, 'commitTime': 0, 'note': 'null'}
```

Fig. 6. Block process

3. Every miner is a process (Fig. 7): once a miner receives a transaction or block, it verifies it. The process adds the block at the end of its own chain and confirm the block before six blocks.

```
def message_consumer(id, env, in_pipe):
    while True:
        myID = id
        my_current_block_id = current_blockID[myID]
        msg = yield in_pipe.get()

        senderID = msg[0]

        if senderID != myID:
            s = msg[2]
            block = json.loads(s)
            blockID = block["blockID"]
            key = str(blockID)
            if blockID >= current_blockID[myID]:
                current_blockID[myID] = blockID + 1
                blockchains_list[myID].update({key: block})
```

Fig. 7. Message process

In order to have a clear look at the mempool, we set a fourth process (Fig. 8) just for recording.

```
def transaction_mempool(env):
    while True:
        yield env.timeout(WAITING_TRAN_TIME)
        global tranFin, tranNO
        waiting_tran_num.append(tranNO - tranFin - 1)
```

Fig. 8. Mempool process

There are several parameters we can change during simulation (Table 1):

Table 1. Parameters in simulation

Parameters	Description	Default
SIM_TIME	The amount of time that simulation runs	24 h
NUN_OF_NODES	The number of nodes in simulation	512
MINGING_TIME	The rank of time that a block could be mined	[8, 10] min
BLOCKSIZE	The limited size of a single block	1000000B
TRANSAC_SIZE	The rank of size that a single transaction could be	[100, 2000] B

The time in the simulation is calculated by 0.1 s, as the transaction is 1 Byte.

The parameters above configure how the simulator runs. In the real Bitcoin network, most of these parameters are formulated or limited. We can change these parameters in order to find out how it affects the Bitcoin network. In this version of simulator, we ignored network delay and orphan blocks.

With this simulator, we can get some important information about the Blockchain in each node. Due to Blockchain's working method, every node gets a similar but not the same chain. Once we consider the network delay, the chain on different nodes may have a little different. In our case, the only different is commitTime caused by the block creator. One Blockchain in node 0 create with default settings shows below. The first diagram in Fig. 9 shows the first few blocks in the network which includes Master Block, the second diagram shows the latest blocks includes some Blocks haven't been committed yet.


```
{"owner": -1, "blockID": 0, "state": 1, "blockSize": 0, "transactionNO": 0,
 "creatTime": 0, "power": 0, "commitTime": 33166, "note": "Master block"}
{"owner": 84, "blockID": 1, "state": 1, "blockSize": 999483, "transactionNO": 960,
 "creatTime": 5901, "power": 5901, "commitTime": 38996, "note": "null"}
{"owner": 26, "blockID": 2, "state": 1, "blockSize": 998162, "transactionNO": 955,
 "creatTime": 11451, "power": 11451, "commitTime": 44008, "note": "null"}
{"owner": 161, "blockID": 3, "state": 1, "blockSize": 999801, "transactionNO": 978,
 "creatTime": 16791, "power": 16791, "commitTime": 49133, "note": "null"}
{"owner": 253, "blockID": 4, "state": 1, "blockSize": 998736, "transactionNO": 959,
 "creatTime": 21938, "power": 21938, "commitTime": 54455, "note": "null"}
{"owner": 379, "blockID": 5, "state": 1, "blockSize": 999108, "transactionNO": 941,
```

```
{"owner": 121, "blockID": 153, "state": 1, "blockSize": 999862, "transactionNO": 969,
 "creatTime": 824690, "power": 32049, "commitTime": 856996, "note": "null"}
{"owner": 304, "blockID": 154, "state": 1, "blockSize": 998800, "transactionNO": 958,
 "creatTime": 830524, "power": 32765, "commitTime": 862147, "note": "null"}
{"owner": 92, "blockID": 155, "state": 0, "blockSize": 999817, "transactionNO": 947,
 "creatTime": 835648, "power": 32631, "commitTime": 0, "note": "null"}
{"owner": 166, "blockID": 156, "state": 0, "blockSize": 998939, "transactionNO": 957,
 "creatTime": 840779, "power": 31774, "commitTime": 0, "note": "null"}
{"owner": 180, "blockID": 157, "state": 0, "blockSize": 998882, "transactionNO": 928,
 "creatTime": 845959, "power": 31650, "commitTime": 0, "note": "null"}
{"owner": 75, "blockID": 158, "state": 0, "blockSize": 999461, "transactionNO": 960,
 "creatTime": 851238, "power": 31586, "commitTime": 0, "note": "null"}
{"owner": 195, "blockID": 159, "state": 0, "blockSize": 999956, "transactionNO": 945,
 "creatTime": 856979, "power": 32289, "commitTime": 0, "note": "null"}
{"owner": 28, "blockID": 160, "state": 0, "blockSize": 999908, "transactionNO": 935,
 "creatTime": 862131, "power": 31607, "commitTime": 0, "note": "null"}
```


Fig. 9. Blockchain information in node 0

We do some simulations with different settings, the result shows below.

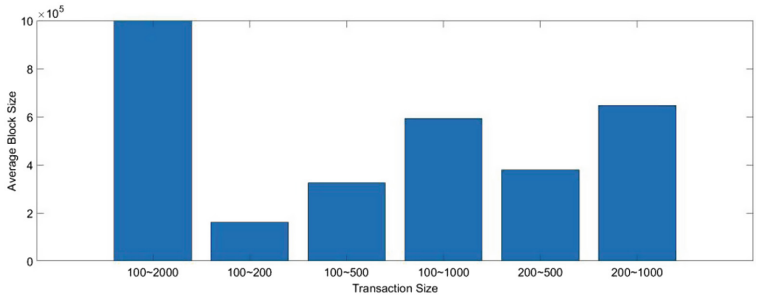

Fig. 10. ABS (change transaction size)

In Fig. 10, as the number of transactions per day and the block size keeps, only when the transaction size is from 100B to 2000B, transactions make the block full. When the transaction is small, more transactions could be recorded in one block. Once there are not enough transactions, the block won't be filled. In real system, mining a block cost a lot, so we want it contain as much information as possible, but not racing.

In Fig. 11(a), it shows details during one day that the mempool size keeps in a low level. The transactions do not need to wait for a long time to be set in a block. Also in Fig. 11(b) and (c), when the transaction per second changes, the mempool size may keep in a low level or go straight high. Once the mempool size goes high, the system will become more and more redundant. Just like in Fig. 13, the transaction commit time rises with time. In the real network, the transaction number depends on different events, which have peak and off-peak time. As well as the transaction size. In further research, we will test the network with huge transactions in a limited time and the ability the network could solve with the stacked Mempool.

In Fig. 12, the number of nodes has little impact on the Blockchain performance. During our simulation, the node could be up to 20000. Actually, the increase of node number will give the system more computing power, which may cause a shorter mining time. Once the mining time is out of range, the system will improve the difficulty of calculation, which finally keeps the mining time limited. In our simulation, the mining time is randomized is in a range. So the increase of node number just cause some pressure on Memory.

In Figs. 10 and 11(a), we change the same settings. But it shows good performance in Fig. 11(a) which turns to be clients' view, bad performance in Fig. 10 which shows the miners' view. It turns out to be different.

Our simulation can show that different parameters change the behavior of the whole network. In future work, we will learn more about how the parameters affect the behavior and how to find out a best setting which will show good performance in both clients' and miners' view.

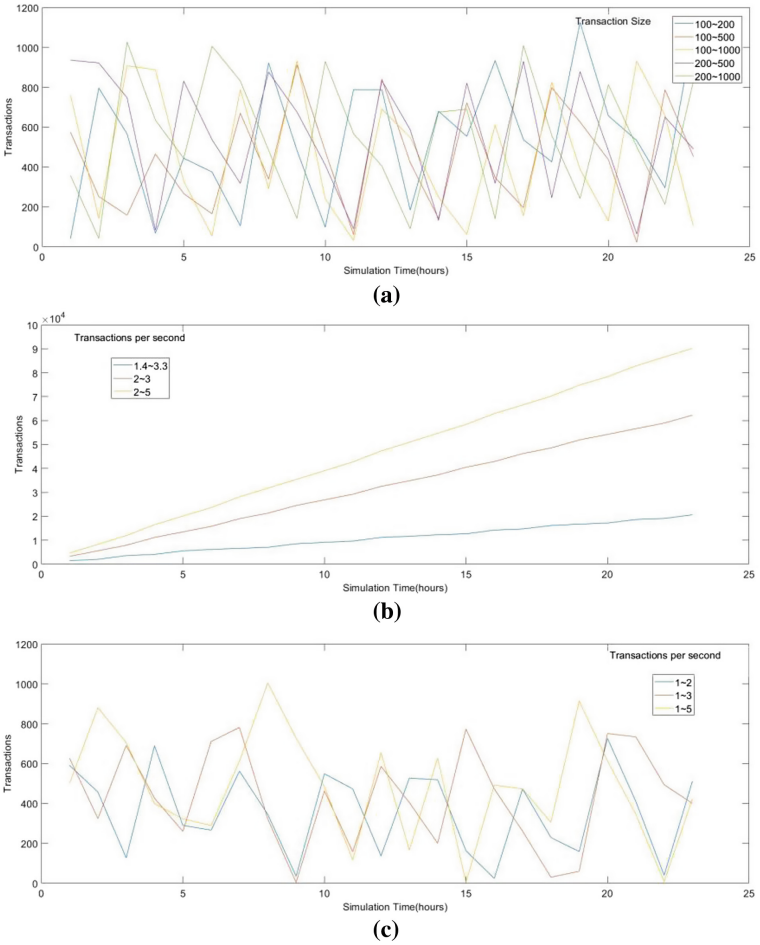

Fig. 11. **(a):** Mempool size (change transactions size) **(b):** Mempool size (change transaction per second) **(c):** Mempool size (change transactions per second)

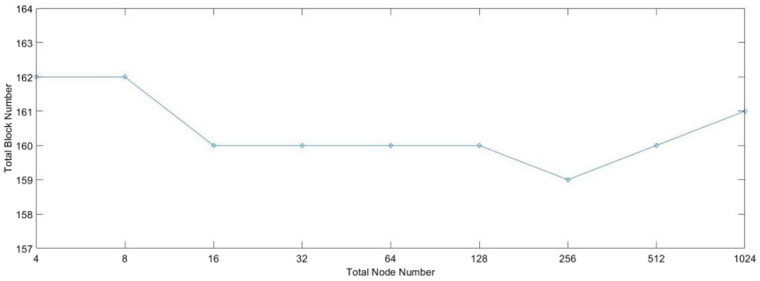

Fig. 12. Total block number (change total node number)

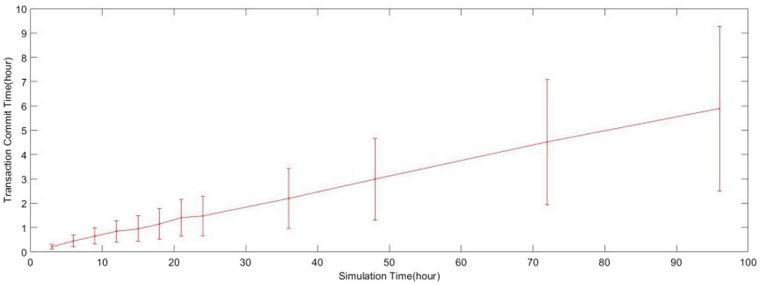

Fig. 13. transaction commit time

With these results, we can made the following observations:

- Our metrics are effective in QoS of Blockchain, we can get a clear view on the performance of the Blockchain.
- Once we jump the mining part and set the mining time in a range, the increase of node number won't influence our simulation till now. But it may cause higher possibility fir nodes to create an orphaned block in the future research.
- Changes in transaction size and transaction per second will show up in the Mempool size. With the rising of Mempool size, the users' experience will become worse because of the long transaction commit time.
- The transaction amount has peak and off-peak time in the real network, so we need a certain amount of Mempool size to keep the block filled and keep the system efficiently. Not too much, not rising all the time, but in a limited size.

5 Related Work

Rajitha et al. [2] used architectural performance modelling and the same incident management exemplar for this approach provided by Weber et al. [4] to measure the latency arising from the Blockchain-related factors, such as the configuration of the number of confirmation blocks and inter-block times. Their management system shows that predictions of median system level response time with a relative error mostly under 10%. Their approach could be used in the design of blockchain-based systems. Their model mostly regards as API, so they ignore the Blockchain mining network, node communication, or consensus algorithm. They modeled the resource and performance characteristics of a local node as a component. In our research, the consensus is one of the main points, which may highly influence the performance in no matter network view or user's view. But their modelling concepts are well-aligned with component-based development and support the re-use of constructed models and components.

Gobel et al. [5] developed two Blockchain simulators, based on the DESMO-J simulation framework [7]. They studied the effect of communication delay in Bitcoin Blockchain under a 'selfish-mine' strategy. First, they use a simplified Markov model that tracks the contrasting states which includes a small amount of dishonest (selfish) miners to establish that the use of block-hiding strategies, such as selfish-mine, which

may cause the increase of orphan blocks. Then they use a spatial Poisson process model to study values of Eyal and Sirer's parameter γ, to find out the proportion an honest miner mine a block which previous block is mined by an honest miner. Finally, they use discrete-event simulation to study the behaviour of a network of Bitcoin miners, which includes selfish-mine action under a network with communication delay between miners. Their study found out that if dishonest miners are exist, the performance of both honest and dishonest miners will become worse, the system also can monitor the production of orphan blocks to find out the behavior of selfish-mining. We haven't mention selfish-miner yet, we could consider it in the further research.

Grevais et al. [6] provided a complete Bitcoin simulator written by NS2. They introduce a novel quantitative framework to analyze the security and performance implications of various consensus and network parameters of PoW blockchains. They find some method to fight against or limit double-spending and selfish mining under their framework, by changing the basic settings such as network propagation, different block sizes, block generation intervals, information propagation mechanism, and the impact of eclipse attacks. Under their framework, they can find a balance between performance and security in Blockchain Network. Compared with our simulator, their simulator is kind of complete but facing the problem on the number of nodes. All the simple node's location and its network delay should be defined individual. And it simulates the procedure of mining as well, which cost a lot of performance. When the node number rises, the simulator takes a bad respond.

Goswami [8] discuss the factors that make Block-chain largely non-scalable. They provide the simulator written by java. This research delves into the scalability issue of blockchains and provides a comparative analysis of several blockchain parameters with real time data. It delves into the factors that make block chains largely non-scalable. This is done by the simulation of blockchain. It then addresses the various mechanisms that can be employed to resolve this limitation through measuring the differences between the simulator and real time scenarios. Their simulator which simulated the PoW work without node communication, just finish the work in one client. It's effective but getting troubles in combination with real network.

6 Conclusion

In this paper, we collect and define a number of key performance metrics to quantify the Quality of Blockchain (QoB). We also use a simple simulation tool to simulate a simplified Blockchain Proof of Work (PoW) protocol within different arguments and our observations. The results shows its relation between the basic settings and the Quality of Blockchian. It shows that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes and protocols. The next step of our work is to make network delay which may cause orphan block into consideration, and then try some other Consensus such as DPoS, PBFT (Fabric) and Tangle (IOTA).

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
2. Rajitha, Y., Staples, M., Weber, I.: Predicting latency of blockchain-based systems using architectural modelling and simulation. In: 2017 IEEE International Conference on Software Architecture (ICSA 2017), pp. 253–256 (2017)
3. Weingartner, E., Vom Lehn, H., Wehrle, K.: A performance comparison of recent network simulators. In: 2009 IEEE International Conference on Communications (ICC 2009), pp. 1–5 (2009)
4. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
5. Göbel, J., et al.: Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. Perform. Eval. **04**, 23–41 (2016)
6. Gervais, A., et al.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16 (2016)
7. Page, B., Kreutzer, W.: Simulating discrete event systems with UML and JAVA. Environ. Sci. Pollut. Res. **13**(6), 441 (2006)
8. Goswami, S.: Scalability analysis of blockchains through blockchain simulation. University of Nevada, Las Vegas (2017)
9. Fairley, P.: Blockchain world-feeding the blockchain beast if Bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous. IEEE Spectr. **54**(10), 36–59 (2017)
10. Augot, D., Chabanne, H., Chenevier, T., George, W., Lambert, L.: A user-centric system for verified identities on the Bitcoin blockchain. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 390–407. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_22
11. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
12. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing, pp. 1–10 (2013)
13. Eyal, I., et al.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI, pp. 45–59 (2016)
14. Gervais, A., et al.: Tampering with the delivery of blocks and transactions in Bitcoin. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 692–705 (2015)
15. Kogias, E.K., et al.: Enhancing Bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium, pp. 279–296 (2016)
16. Kosba, A., et al.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy, pp. 839–858 (2016)
17. Pazmiño, J.E., Rodrigues, C.K.S.: Simply dividing a Bitcoin network node may reduce transaction verification time. SIJ Trans. Comput. Netw. Commun. Eng. **3**(2), 17–21 (2015)
18. The Ethereum community. Ethereum White Paper, July 2015. https://github.com/ethereum/wiki/wiki/WhitePaper
19. Popov, S.: The Tangle. IOTA White Paper (2017). https://iota.org/IOTA_Whitepaper.pdf
20. https://en.wikipedia.org/wiki/SegWit