2020 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2020, Virtual, November 16-19, 2020

Monitoring Private Blockchain Performance on Non-mining Nodes

Xuan Chen, Kien Nguyen, and Hiroo Sekiya

Graduate School of Science and Engineering, Chiba University
1-33, Yayoi-cho, Inage-ku, Chiba-shi, Chiba, Japan
Email: chenxuan, nguyen@chiba-u.jp, sekiya@faculty.chiba-u.jp

Abstract—A private blockchain is potentially used to maintain IoT systems, such as a smart home. In such a blockchain, the mining task (i.e., computation for generating blocks) is too heavy for resource-limited IoT devices. Thus, a powerful node with sufficient resources will serve as a miner for the whole blockchain network. Meanwhile, the other nodes operate without the mining function (i.e., non-mining nodes). Hence, understanding the performance of non-mining IoT nodes is essential for the private blockchain. In this paper, we aim to monitor the performance of the real devices (i.e., Raspberry Pi), which form a blockchain using the popular private Ethereum. We use on-device tools to track disk space usage, memory usage, CPU utilization, and network usage on non-mining nodes. The collected results within two hours show that it is feasible to monitor the nodes' performance.

1. Introduction

A blockchain is composed of a series of blocks, in which each block includes a block body containing transactions and a block header containing the previous block's cryptographic hash. When generating a new block, the nodes conduct a proof-of-work (PoW) algorithm, that aims to search an answer to a hard-to-solve, easy-to-verify mathematical problem. The generated block with the answer in the header is propagated to all other nodes. The nodes verify and execute transactions in the block body. A legal block will be successfully verified and attached to the blockchain on all nodes.

Ethereum is one of the most popular open-source blockchain platforms [1]. One of the distinguishing features of Ethereum is allowing a node uploading of automatically executed codes known as smart contracts. Ethereum supports both the public and private blockchain networks [2]. The former network enables participants to join without any limitations. Meanwhile, the latter usually requests participants to get permission before entering the blockchain. The private Ethereum has found its feasibility in several IoT applications [3].

The Internet of Things (IoT) include devices such as sensors or actuators, which are transferring data without human interaction. However, the traditional IoT systems rely on a centralized model, in which a central entity is in charge of data transmission and system management. When the number of IoT devices increases, the centralized model faces many problems, including scalability and security [4]. The blockchain technologies can be combined with IoT to loosen those problematic issues [5]. However, most of the current blockchains are computation heavy in solving the PoW on resource-limited IoT devices. One of the solutions for that is to introduce a powerful central miner [6], which achieves the complicated computation work. The other IoT devices, which are parts of the blockchain network, operate other non-mining functions. In [7], [8], the authors build a smart home architecture that contains several local private blockchain networks. In each private network, a node is elected as a central miner that can communicate with other networks' miners. In [9], the authors present another Ethereum based smart home system, that integrates a central miner and several non-mining nodes. The miner is installed on a powerful computer, managing data delivery, and storage.

Since the blockchain-IoT applications are still in an early stage, understanding their performance is one of the most critical issues. In [10], the authors proposed a log-based real-time framework to monitor Ethereum, but the focus is on the public networks. The work in [11] introduces an evaluation framework for analyzing private blockchain performance. In [12], the authors extend [11]'s evaluation with variations workloads. Both the works, however, have measured the performance metrics on the mining nodes. In [13], the authors have characterized latency in the private Ethereum network. Different from them, we aim to explore the feasibility of monitoring the non-mining nodes in private blockchains.

In this paper, we present a method to monitor the private blockchain system with the operating system's internal commands. We first build a private Ethereum blockchain network and deploy a smart contract. The non-mining nodes run the Ubuntu Mate operating system. The Web3.js library [14] is used to interact with the smart contract. We use a monitoring script that captures the log information of disk space usage, memory usage, CPU utilization, and network throughput. Then we edit *crontab* [15] on the Linux systems to execute the script every minute. The monitored results reveal the feasibility of the method with the performance values of non-mining nodes.

The rest of the paper is organized as follows. Section 2 describes our monitoring methodology. In Section 3, we present the monitoring results on a real blockchain IoT-based network. Finally, Section 4 concludes the paper.

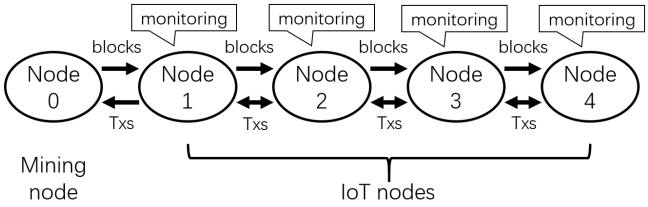

Figure 1: Private blockchain structure in experiments

Table 1: Details of RPI nodes

Processor	4x Cortex-A53 1.4 GHz
Memory	1 GB
Storage	16 GB MicroSDHC
OS	Ubuntu Mate 18.04
Ethereum client	Geth 1.9.14-stable

2. Methodology

Ethereum blockchains have different transmission protocols, such as Ethereum Wire Protocol (ETH) [16] and Light Ethereum Subprotocol (LES) [17]. The nodes following ETH can be in the *full* or *fast* mode. In the former, the nodes will download all previous blocks and execute transactions to generate the state tree, which is a mapping between accounts' addresses and account states (e.g., account balance). In the latter, the nodes download blocks and the state tree instead of re-building it. On the contrary, in the *light* mode, nodes only download block headers on demand. Moreover, they neither mine blocks nor verify transactions since they lack the state tree. To maximize nodes' functionality and ensure the security, we set all nodes in the *full* mode.

To mine blocks, a device needs enough memory space to generate directed acyclic graphs (DAG), which helps Ethereum blockchain nodes search for the PoW answer. Also, a sufficient CPU is required to mine a block within an acceptable time. Since the IoT device is resource-limited, a laptop is introduced for the tasks in our private Ethereum blockchain.

We monitor the private Ethereum blockchain, which has a linear structure, as shown in Fig. 1. In the figure, node 0 is the laptop serving as the miner. Other nodes (e.g., nodes 1 to 4) are resource-limited IoT devices. In this blockchain network, transactions (Txs) are sent from IoT nodes and synchronized on every node. The mining node executes PoW to generate blocks, which are propagated hop by hop. On each node, we enable monitoring tools, which are combined into a script. The monitoring script contains the following items.

- Monitoring disk space usage: Since Ethereum stores old blocks in a disk, hence it is worthy of investigating the disk usage. We use the Linux tool named `df` [18] to get the values of disk space usage.
- Monitoring memory usage: Ethereum stores recent blocks and pending transactions in an IoT node's memory. Therefore, the memory space relates to the capacity of achieving transactions. The monitoring values of memory usage can be obtained with the `free` [19] tool.
- Monitoring CPU utilization: Besides the miner, non-mining Ethereum nodes need to spend CPU resources with the formation of its transactions and verification of transactions and blocks from the others. Understanding the CPU utilization is, therefore, essential, especially on the IoT devices. We use the `iostat` [20] tool to get the values of the average CPU utilization.
- Monitoring network throughput: Ethereum nodes communicate with each other continuously to exchange states, propagate transactions and blocks through its underlying network. Therefore, network throughput plays an important role in understanding the whole blockchain performance. We log and read the accumulated bytes of data transmitted and received from the `/proc/net/dev` file on the IoT devices' operating system.

The monitoring script is integrated with the cron (i.e., in *crontab*), a built-in Linux utility. The utility is a time-based scheduler that executes the script periodically. This monitoring method has a negligible impact on the running blockchain system comparing with the RPC-based way. Additionally, the script can be extended with other parameters such as I/O speed, load average, and CPU temperature.

3. Monitoring Private Ethereum

3.1. Experiment Setup

In this work, we use the Raspberry Pi 3 Model B+ (RPI) IoT devices as the non-mining nodes. The details of PRI nodes are shown in Table 1. Geth [21], which is the official implementation of Ethereum nodes in Golang, is previously deployed on all nodes. We first initialize Geth on each device in the *full* mode with a custom genesis file, which defines a low difficulty to have a quick mining process. Second, we implement a smart contract, which can write a string to the blockchain and read the current string. Third, we use the Web3.js library on each RPI node to interact with the smart contract. We write a script to send transactions to write information to the smart contract every second. The miner collects transactions and generates blocks through the mining process. Finally, we modify the *crontab* on each device to execute the monitoring script every minute. With those preparations, we have monitored the private Ethereum blockchain for two hours. The results are presented as follows.

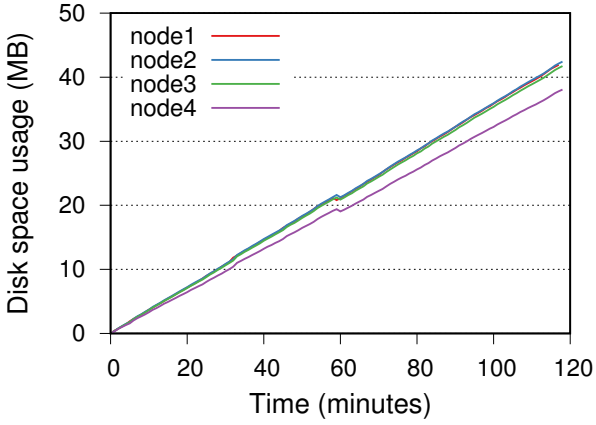

Figure 2: Disk space increment on non-mining RPI nodes

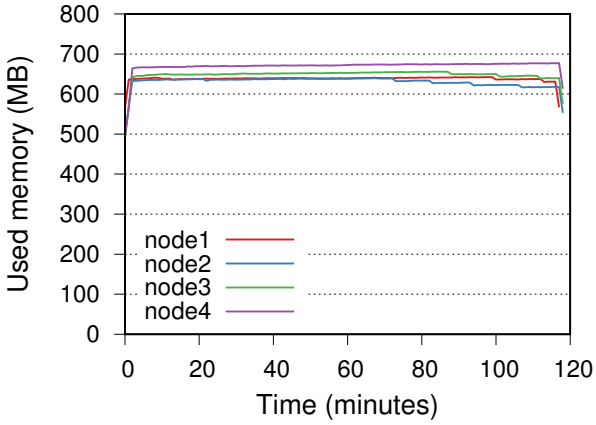

Figure 3: Memory occupation on non-mining RPI nodes

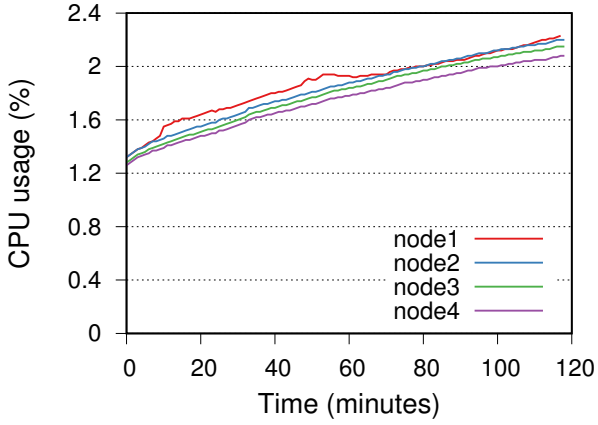

Figure 4: CPU utilization on non-mining RPI nodes

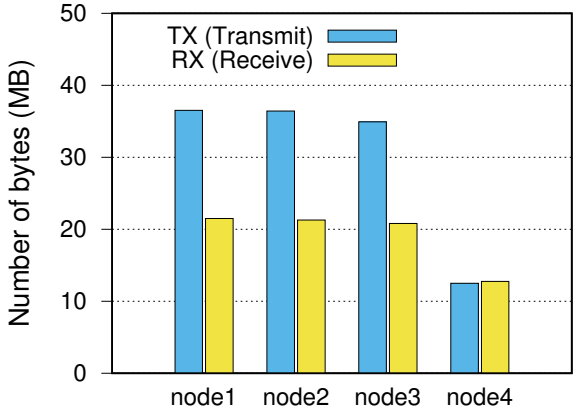

Figure 5: TX and RX bytes on each RPI after two hours

3.2. Result

3.2.1. Disk space usage

The increment of used disk space on each node is shown in Fig. 2. In 120 minutes, the mining node generates 647 blocks, which contain about 28800 transactions. Most of the blocks are stored in disk space. The size of a block depends on the number of transactions inside. An empty block takes 537 Bytes space, and each transaction takes about 200 Bytes (various on different types of transactions.) Moreover, the disk usage will not decrease since each node keeps a full copy of the entire blockchain.

3.2.2. Memory usage

In our experiments, all the RPI nodes consume 256 KB swap space, which is negligible. Thus, we present the memory variation on each non-mining node in Fig. 3. It indicates RPI nodes need approximately 650 MB memory. The memory is consumed by unverified transactions and child processes created by Geth, and remain stable at the front half of the figure. In the back half, memory usages drop because some transactions are timeout and removed. Besides, the memory usage rises and drops at both ends of the line, which is the start and close of Geth processes.

3.2.3. CPU utilization

The CPU utilization rate of each node is shown in Fig. 4. The RPI nodes consume approximately 2% of the CPU because the verification of blocks and transactions don't require too much calculation. On the contrary, the mining process consumes almost all of the CPU on the laptop. The CPU utilization is slowly growing because `iostat` command calculates the average CPU time since the system was booted.

3.2.4. Network throughput

We first monitor the idle situation, in which the nodes are connected without mining or sending transactions. The nodes keep confirming the peer connections and checking the highest block. We then investigate the running situation, in which the miner starts mining, and all non-mining nodes keep submitting transactions one per second. Figure 5 shows the amount of transmitted and received data on the wireless interface of each RPI in the running situation.

Table 2: Throughput on RPI nodes (Bytes/s)

	Node 1	Node 2	Node 3	Node 4
Idle	232.63	219.65	222.94	122.78
Running	4837.23	4623.88	4759.25	2042.71

The first three nodes transmit more data than received because they have two peers to propagate, while node 4 only has one. We calculate the total throughput of both situations in Table 2. The throughput of node 4 in both situations is approximately half of the other nodes.

4. Conclusion

In this paper, we have shown the performance monitoring of non-mining nodes in an Ethereum blockchain network. Our built blockchain has four RPI nodes serving as non-mining nodes. We upload a smart contract to store string sent from RPI nodes and use the Web3.js library to interact with it. We then use Linux monitoring tools and execute the monitoring tasks in a script with *crontab*. We have collected the data of disk space usage, memory usage, CPU utilization, and network usage on the RPI nodes for 2 hours. The results show the method is feasible, and also present the performance of non-mining nodes. The monitoring script can be extended to include many other parameters.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 19K20251, 20H04174. Additionally, Kien Nguyen is supported by the Leading Initiative for Excellent Young Researchers (LEADER) program from MEXT, Japan.

References

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger." `https://ethereum.github.io/yellowpaper/paper.pdf`. (accessed: 2020-08-16).

[2] K. Wüst and A. Gervais, "Do you need a blockchain?," in *Proc. IEEE CVCBT*, pp. 45–54, 2018.

[3] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[4] J. Wurm, K. Hoang, and O. Arias, "Security analysis on consumer and industrial iot devices," in *Proc. ASP-DAC*, pp. 519–524, 2016.

[5] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[6] H. Rathore, A. Mohamed, and M. Guizani, "A survey of blockchain enabled cyber-physical systems," *Sensors*, vol. 20, no. 1, p. 282, 2020.

[7] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in internet of things: challenges and solutions," *arXiv*, 2016.

[8] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *Proc. IEEE PerCom workshops*, pp. 618–623, 2017.

[9] Y. N. Aung and T. Tantidham, "Review of ethereum: Smart home case study," in *Proc. INCIT*, pp. 1–4, 2017.

[10] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. IEEE/ACM ICSE-SEIP*, pp. 134–143, 2018.

[11] T. T. A. Dinh, J. Wang, G. Chen, and R. Liu, "Blockbench: A framework for analyzing private blockchains," in *Proc. ACM International Conference on Management of Data*, pp. 1085–1100, 2017.

[12] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proc. IEEE ICCCN*, pp. 1–6, 2017.

[13] X. Chen, K. Nguyen, and H. Sekiya, "Characterizing latency performance in private blockchain network," in *Proc. MONAMI*, 2020. (accepted).

[14] "web3-npm." `https://www.npmjs.com/package/web3`. (accessed: 2020-08-15).

[15] "crontab-linux man page." `https://linux.die.net/man/1/crontab`. (accessed: 2020-08-14).

[16] "Ethereum wire protocol (eth)." `https://github.com/ethereum/devp2p/blob/master/caps/eth.md`. (accessed: 2020-08-04).

[17] "Light ethereum subprotocol (les)." `https://github.com/ethereum/devp2p/blob/master/caps/les.md`. (accessed: 2020-08-04).

[18] "df-linux man page." `https://linux.die.net/man/1/df`. (accessed: 2020-08-14).

[19] "free-linux man page." `https://linux.die.net/man/1/free`. (accessed: 2020-08-14).

[20] "iostat-linux man page." `https://linux.die.net/man/1/iostat`. (accessed: 2020-08-14).

[21] "Go ethereum." `https://geth.ethereum.org/`. (accessed: 2020-08-14).