ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 1, 218-223 (2021)

www.astesj.com

Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ

ISSN: 2415-6698

Analysis of Real-time Blockchain Considering Service Level Agreement (SLA)

Minkyung Kim[1], Kangseok Kim[2,3], Jai-Hoon Kim[*,3]

[1]*Dasan College, Ajou University, Suwon, 16499, South Korea*

[2]*Department of Artificial Intelligence and Data Science, Ajou University, Suwon, 16499, South Korea*

[3]*Department of Cyber Security, Ajou University, Suwon, 16499, South Korea*

ARTICLE INFO

Article history:
Received: 15 October, 2020
Accepted: 21 December, 2020
Online: 15 January, 2021

Keywords:
Consensus Algorithm
Byzantine Fault Tolerance
Blockchain
Internet of Things
Decentralized Framework

ABSTRACT

The Blockchain technologies enable decentralized networking consisting of large number of nodes. To determine the shared states and failures of all nodes in a fully distributed peer-to-peer system, the appropriate consensus algorithm needs to be selected for each Internet of Things system. In this paper, a novel hierarchical voting-based byzantine fault tolerance (HBFT) consensus algorithm is proposed. The proposed HBFT algorithm utilizes a typical PBFT algorithm hierarchically to guarantee low latency. The message complexity of HBFT shows that our proposed algorithm has better scalability. We also mathematically calculate the optimal number of groups based on the total number of nodes to determine the ratio of allowable faulty nodes per group. In addition, we analyze the reliability of byzantine fault tolerance to compare the reliability of group case with the reliability of non-group case. Finally, we introduce the methods of real-time Blockchain considering the service level agreement (SLA). The real-time processing performance of transactions is analyzed for the service level agreement (SLA).

1. Introduction

A Blockchain is a peer-to-peer distributed ledger , in which the process of transaction verification and recording is continuously executed [1, 2]. Participant nodes constantly verify a set of time-stamped transactions at a given time using a built-in consensus algorithm. The Blockchain technology improves reliability in addition to availability by storing and sharing data in a distributed manner. The use of Blockchain in Internet of Things (IoT) enables secure, decentralized networking for IoT data privacy and security [3]. Therefore, an efficient consensus algorithm is crucial to verify transactions and adjust interactions among IoT devices [4]. IoT data are validated through the consensus mechanism and then recorded securely in a distributed ledger.

This paper introduces a novel hierarchical voting-based byzantine fault tolerance (HBFT) consensus algorithm. The proposed HBFT algorithm operates the group level's consensus rather than the node level's consensus. It is to overcome limited scalability and high latency in a typical practical byzantine fault tolerance (PBFT) algorithm. Thus, the overall exchanging messages are significantly reduced even if the large number of nodes are participated in consensus operation. Despite the exponential growth in the number of nodes in the network, the scalability of HBFT is ensured compared with PBFT. Also, we calculate the optimal number of groups depending on the total number of nodes for grouping because the reliability can be affected by the number of groups. If faulty nodes are well distributed into groups, the probability of reaching consensus is higher even with an increased number of faulty nodes. Therefore, two cases are calculated to understand the influence on the number of faults per group.

However, since the participant nodes mutually verify and store the data, considerable number of computing resources are required, and concurrently the processing time decreases [5-8]. The same method of participant nodes (all nodes or certain nodes allowed to participate in Blockchain) has been used to process transactions on the Blockchain [9]; nevertheless, it is possible to apply the service level agreement (SLA) on Blockchain by determining the number of participant nodes according to the user's requirements. Therefore, we propose methods to control the number of participant nodes adaptively considering user requirements and computing environments. The transaction processing speed and throughput can be improved by using adaptive control method. Figure 1 is the diagram of real-time

*Corresponding Author: Jai-Hoon Kim, Department of Cyber Security, Ajou University, Suwon, 16499, Korea, jaikim@ajou.ac.kr

Blockchain considering service level agreement(SLA). In this diagram, the reliability of a node with respect to time is periodically collected by the Blockchain monitor. It depends on the failure rate of a node determined intentional or unintentional faults. The Blockchain control also reads user requirements and Blockchain status. According to the user's request, a transaction is processed for generating a smart contract of Blockchain. The user can request reliability, maximum time and the cost for block generation. The reliability of a participant node can affect the reliability of the transaction processed.

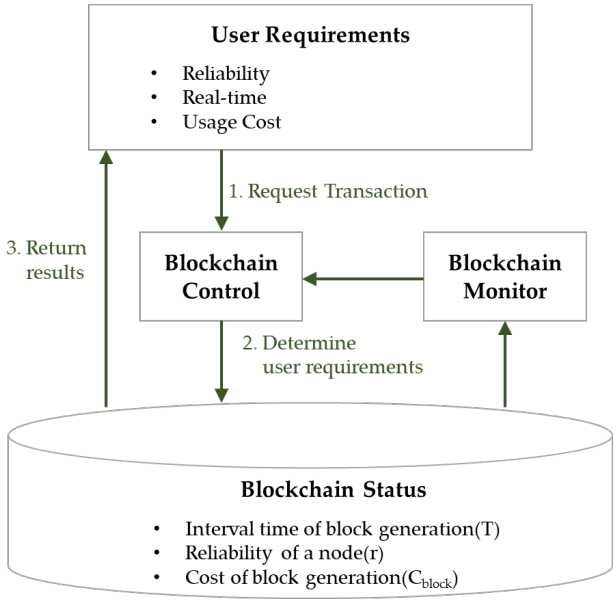

Figure 1: Schematic of Real-time Blockchain considering Service Level Agreement(SLA)

The remainder of this paper is organized as follows. Section 2 overviews relevant background about various consensus algorithms. The details of HBFT algorithm are presented in Section 3. Here, how the HBFT algorithm differs from previous consensus mechanisms has been described. In Section 4, we analyze the reliability of byzantine fault tolerance, and present the algorithm of real-time block generation in Section 5. Finally, Section 6 concludes the contributions with future research perspectives and additional improvements.

2. Related Work

There are various consensus algorithms adopted by the popular Blockchain-based platforms such as *Bitcoin*, *Ethereum* and *Hyperledger Fabric*. In [10-11], the different advantages and disadvantages of each existing consensus algorithm are introduced. Bitcoin utilizes Proof of Work (PoW), which the high computational power is required to solve a cryptographic puzzle for mining operation. Proof of Stake (PoS) is designed to overcome the disadvantages of PoW algorithms. However, PoS suffers from a problem called Nothing at Stake because each participant can vote to both blocks. Also, Delegated Proof of Stake (DPoS) achieves consensus by delegates (called block producers) elected by nodes instead of participation by all nodes. Practical Byzantine Fault Tolerance (PBFT), implemented in *Hyperledger Fabric*, has a primary node to manage the other nodes. Even if some untrusted nodes participate in the consensus operation, it is considered an agreement if more than 2⁄3 of the results are the same. Also, a variant of byzantine fault tolerance consensus algorithm called Delegated Byzantine Fault Tolerance (DBFT) [12, 13] is introduced for scalable participation. The randomly elected delegates (called bookkeeping nodes) of each group can only participate in the operation of the consensus process. Furthermore, PBFT is an permissioned protocol in which the identities of all nodes are exposed to the network [14]. Therefore, PBFT can be applied in permissioned platforms that enable autonomous, commercial, and financial application services in IoT environments. It is also suitable for applications that do not require tokens and incentives. That is why we developed a novel and efficient consensus algorithm based on PBFT for decentralized IoT.

3. Hierarchical voting-based Byzantine Fault Tolerance (HBFT) Consensus Algorithm to reduce System Resources

In this section, a novel hierarchical voting-based byzantine fault tolerance (HBFT) consensus algorithm is proposed. Also, we analyze the message complexity of the proposed novel consensus algorithm. Additionally, the best case and worst case are calculated to identify the influence on the number of faulty nodes per group in our proposed algorithm.

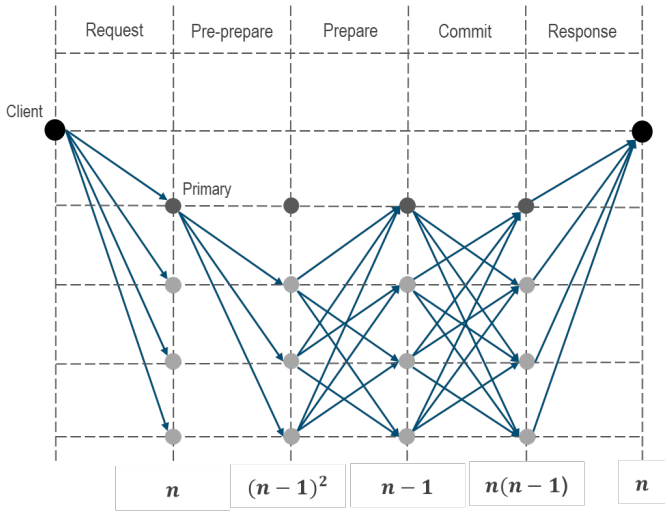

Figure 2: Flow Chart of PBFT

Practical Byzantine Fault Tolerance (PBFT) consensus algorithm has five processes for consensus processing [15-17]. Figure 2 shows the flow chart of PBFT. If the number of faulty nodes (f) is not more than $f = \lfloor (n-1)/3 \rfloor$, a consensus can be reached. Even though faulty nodes participate in the network, PBFT consensus algorithm can prevent up to $n/3$ of faulty nodes since more than $2n/3$ nodes must be reliable nodes. The overall message complexity is $2n^2$ based on the messages generated in each process. In case of the PBFT algorithm supported by *HyperLedger Fabric*, the reliability of the consensus is improved because every node participates in the consensus process through the interactions with each other. However, the message complexity

increases exponentially as the number of participating nodes increase. Therefore, if the number of nodes increases, the problem of scalability can be generated by high network traffic and latency.

3.1. Proposed HBTF Consensus Algorithm for Scalability

Based on our previous researches [18, 19], the operation of HBFT consensus algorithm, which utilizes a typical PBFT algorithm hierarchically, is re-designed effectively. Our proposed HBFT consensus algorithm has two phases to validate transactions. The flow chart of HBFT is shown in Figure 3.

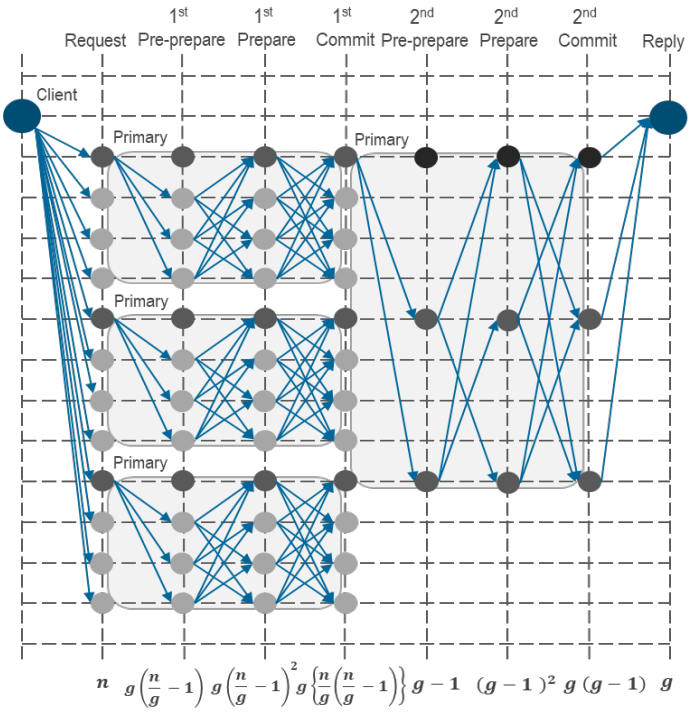

Figure 3: Flow Chart of HBFT

Firstly, the first phase start to request a verification about new transactions to all replica nodes. In the pre-prepare process, all nodes are randomly grouped into some groups and then, a primary node in each group is randomly selected to send a pre-prepare message to the other nodes. Each node compares a pre-prepare message with a request message. If the results match, a primary node responds a prepare message to the other nodes. In the commit process, each node reaches an agreement in the group depending on whether it received more than $2n/3g$ of prepare messages. Next, the second phase is executed in the same processes as the first phase to verify the voting results of each group. Finally, if more than $2g/3$ of prepare messages are received, the consensus is reached. Each primary node broadcasts the results to a client.

3.2. Message Complexity of HBFT Consensus Algorithm

The overall message complexity generated in each process of HBFT is $f(g) = 2g^2 - g + 2n^2/g - n$. The overall message complexity of HBFT is lower than that of PBFT by increasing the total number of participant nodes in Figure 4. Therefore, the processing overhead is significantly decreased. The lower latency and better scalability can be guaranteed because the overall overhead is reduced.

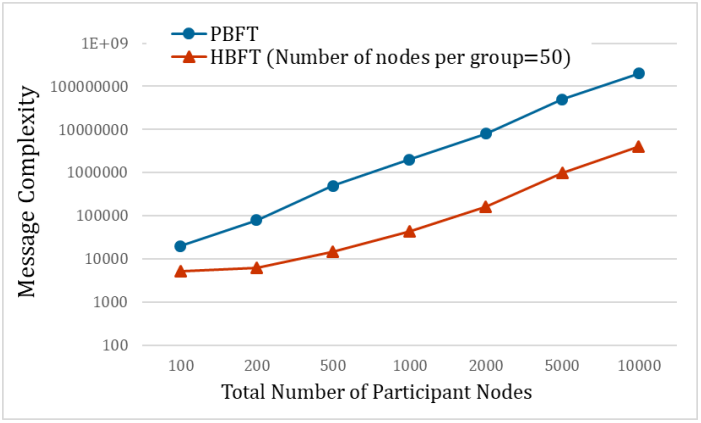

Figure 4: Overall Message Complexity Comparison between PBFT and HBFT

Additionally, we try to figure out the impact on the message complexity by increasing the number of groups in Figure 5. If the total number of nodes is 1000, the message complexity depends on the number of groups. The message complexity increases as the number of groups is too small or too large. Since the number of groups can affect the reliability of the agreement, the optimal number of groups according to total number of participant nodes needs to be calculated for facilitating efficient grouping. The optimal number of groups based on total number of nodes is calculated as follows. Eq. (1) is an equation for the overall message complexity of HBFT. By differentiating Eq. (1), we can achieve the number of groups with minimal message complexity. The optimal number of groups is given by Eq. (2). If the total number of nodes is 1000, the calculation result of Eq. (2) is about 79.45 as shown in Figure 5.

$$f(g) = 2g^2 - g + \frac{2n^2}{g} - n \tag{1}$$

$$\rightarrow \quad f'(g) = 4g - 1 - \frac{2n^2}{g^2} = 0$$

$$g = \frac{1}{12}\left(1 + \sqrt[3]{1 + 432n^2 - \sqrt{432n^2(432n^2 + 2)}} + \sqrt[3]{1 + 432n^2 + \sqrt{432n^2(432n^2 + 2)}}\right) \tag{2}$$

3.3. Fault Tolerance of HBFT Consensus Algorithm

In our proposed HBFT algorithm, consensus can be achieved to add a new block when the number of groups is more than two-thirds the total number of groups. Two cases are calculated to determine the influence on the number of faulty nodes per group. In the first case, the maximum number of allowable faults to reach consensus is calculated in (a). Additionally, a case where the consensus cannot be reached with the minimum number of faulty nodes is calculated in (b). When it is detected as a faulty node, a primary node is changed in the typical PBFT algorithm. Therefore, all primary nodes are assumed to be non-faulty nodes, similar to the PBFT algorithm. Additionally, we assume that the number of total groups is more than two, and the number of nodes per group

is at least four in our proposed HBFT algorithm for normal operation.

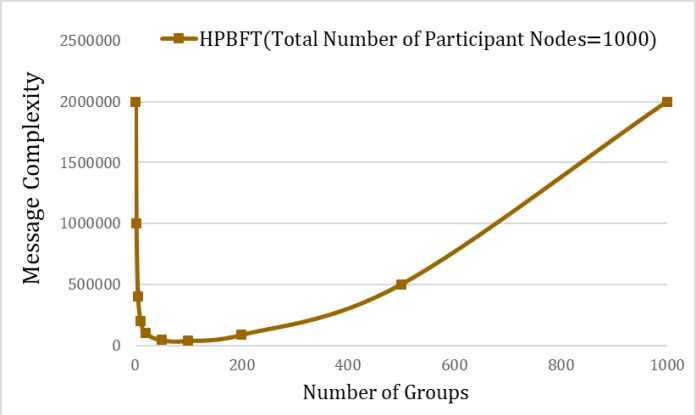

Figure 5: Message Complexity of HBFT depending on the Number of Groups

3.3.1. *Best case: The maximum number of allowable faulty nodes in order to reach a consensus*

One-third of total number of groups consists of faulty nodes and cannot reach a group consensus. The other two-thirds have only one-third faulty nodes in each group, leading to a group agreement. Consequently, the best case is that the number of faulty nodes is high; however, a consensus can be reached since faulty nodes are well distributed into groups. Eq. (3) shows the equation of the best case.

$$\left(\left\lfloor\frac{g}{3}\right\rfloor \times \left(\frac{n}{g}-1\right)\right) + \left(\left\lceil\frac{2g}{3}\right\rceil \times \left\lfloor\frac{n}{3g}\right\rfloor\right) \quad (3)$$

3.3.2. *Worst case: A consensus cannot be reached with the minimum number of faulty nodes*

If the number of faulty nodes in each group exceeds one-third, each group agreement fails. The worst case is in which the number of groups that do not attain group agreement exceeds one-third. As a result, even though the number of faulty nodes occupies a relatively small number, the consensus cannot be reached according to the ratio of faulty nodes per group. An equation of this worst case is described as follows.

$$\left(\left\lfloor\frac{g}{3}\right\rfloor + 1\right) \times \left(\left\lfloor\frac{n}{3g}\right\rfloor + 1\right) \quad (4)$$

Both equations are calculated to assess whether a consensus can be achieved according to the ratio of faulty nodes per group. The influence of the ratio of faulty nodes per group is shown in Figure 6. From these calculations, the number of allowable faulty nodes is up to approximately $5n/9 - g/3$ in the best case. In the worst case as already discussed, a consensus cannot be reached according to the ratio of faulty nodes per group. In both calculations, whether a consensus is reached or not is affected by the ratio of faulty nodes per group. Although the number of faulty nodes participating in consensus is high, a consensus can be reached depending on how all faulty nodes are well-distributed into groups as described in the best case. Finding the optimal ratio of faulty nodes per group to reach consensus in HBFT algorithm will be an important research area. In addition, it is required that the verification regarding the faulty nodes is executed at the beginning of consensus process. As a result, the number of allowable faulty nodes can be low compared to PBFT algorithm depending on the ratio of faulty nodes per group in the worst case. On the contrary, the number of allowable faulty nodes can be more than that in the PFBT algorithm depending on the ratio of faulty nodes per group in the best case.

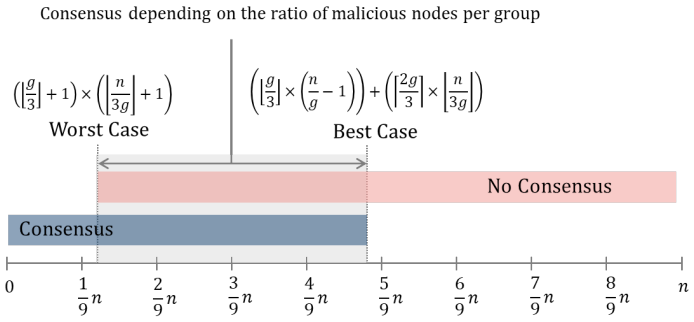

Figure 6: Influence on the Ratio of Faulty Nodes Per Group

4. Analysis of Reliability

We analyze the reliability of Blockchain operation in this section. It leads to the analyzing of the probability of normal operation in unit time. Assuming that the reliability of each participant node is equal to *r*, the calculations below are defined when *n* nodes participate in the processing a transaction. The reliability of the fail-stop faults can be calculated by the probability that at least one node is operating normally. A fail-stop means the occurrence of an unintentionally generated failure. It is assumed that a fail-stop at a node happens when a node is faulty.

$$1-(1-r)^n \quad (5)$$

In the case of the reliability for consensus, even if there are the fail-stop faults, the consensus can be reached if more than half of the number of nodes agree. Subsequently, the reliability of the byzantine faults can be calculated by the probability that less than half of faulty nodes are included on Blockchain as follows.

$$R(n,r) = 1 - \sum_{k=ceil(\frac{n}{2})}^{n} {}_nC_k(1-r)^k r^{n-k} =$$

$$\sum_{k=0}^{ceil\left(\frac{n}{2}-1\right)} {}_nC_k(1-r)^k r^{n-k} \quad (6)$$

In the case of the reliability for consensus in byzantine faults, if less than one-third of faulty nodes are in the operation, the consensus is reached. The calculation is shown in Eq. (7)

$$R(n,r) = 1 - \sum_{k=ceil(\frac{2n}{3})}^{n} {}_nC_k(1-r)^k r^{n-k} =$$

$$\sum_{k=0}^{ceil\left(\frac{2n}{3}-1\right)} {}_nC_k(1-r)^k r^{n-k} \quad (7)$$

In addition, the reliability of the byzantine faults in a hierarchical structure as in the proposed HBFT consensus algorithm can be defined as *R*(*g*, *R*(*n*/*g*, *r*)). It is assumed that the total number of nodes is n, and the number of nodes per a group is *n*/*g*. Additionally, when all nodes are divided into several groups for consensus, the number of groups is g. Therefore, the reliability of group is calculated as *R*(*n*/*g*, *r*) based on the reliability of a node. In case of the reliability for consensus, even if there are the

byzantine faults, this same definition is applied. If the more than two-third the number of normal nodes are in the consensus operation, a consensus is reached to add a new block.

Assuming that the number of nodes is 400 and less than half the faulty nodes are included, the reliability based on that of each node is shown in Figure 7. Furthermore, if the number of groups is 20 with the total number of nodes, the reliability of hierarchical byzantine fault tolerance by grouping is shown in Figure 7. When the consensus operation executes hierarchically in the group level, the reliability of group case is similar or lower than the reliability of non-group case. Although the difference in the case of reliability of hierarchical byzantine fault tolerance is lower than in the case by the node's reliability, the message complexity is much better described in the Section 3.2; so it can be said that it is good to be grouped in the proposed algorithm. When the number of faulty nodes is less than one-third of the total nodes, the consensus operates hierarchically according to the reliability of each node.

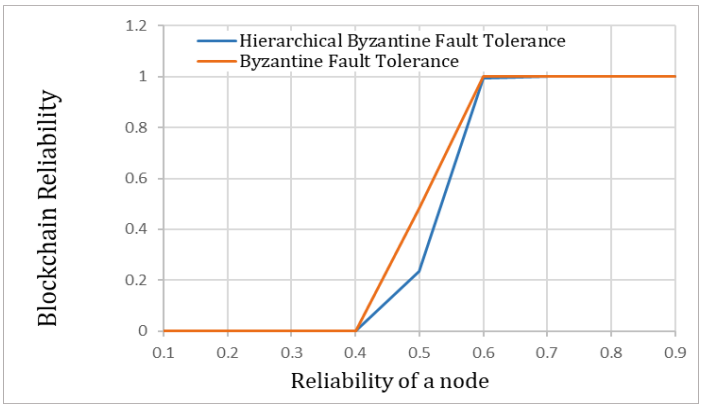

Figure 7: Reliability in the Byzantine Fault

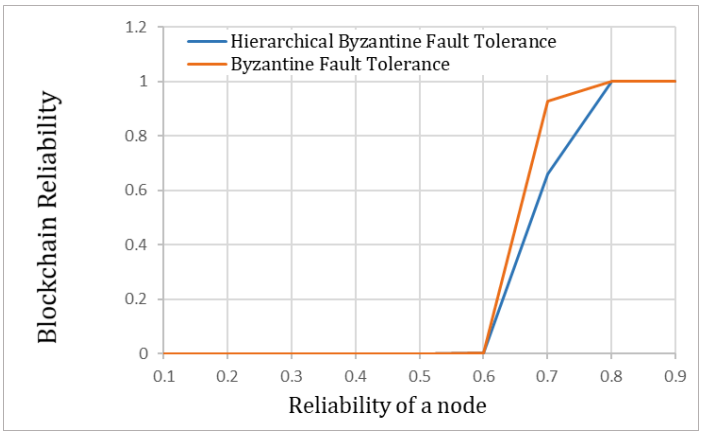

Figure 8: Reliability for Reaching Consensus in the Byzantine Fault

Figure 8 shows that the reliability of the consensus in the case of group level and in the case of non-group level consensus. Although the range with a little low reliability exists in the reliability of hierarchical byzantine fault tolerance for reaching consensus, there is no notable difference in the reliability of the overall ranges, similar to Figure 7. Even if the reliability of each node is different, or an incorrect answer is generated by an intentional failure or byzantine faults, the reliability can be calculated. In addition, even if the participant nodes have different conditions such as reliability, the possibility of cyber infringement, computing power, and data collection, it is possible to apply the proposed method of using Blockchain resources minimally according to various Blockchain configuration conditions and the user's requirements.

5. Real-time Performance Analysis using Block Generation Algorithm

Although the Blockchain has various advantages such as reliability, security, traceability and transparency, the excessive computing resources and communication bandwidths required to reach consensus and maintain consistency between duplicated ledgers are disadvantageous. Additionally, it is necessary to consider the reduction of the deadline meet ratio according to the laxity time for consensus in computing environments required real-time transaction processing. In this paper, the real-time processing performance of transactions is analyzed according to the redundancy of nodes. There are the applicable methods for maximizing real-time performance and satisfying user's requirements. The first method is real-time block generation algorithms. The second method is to use off-chain for reliable real-time performance although the reliability is decreased. A method to obtain the priority for block generation exists, as it pays more gas costs. Subsequently, the delay time is controlled by adjusting the number of nodes participating in consensus. In this section, we propose real-time block generation algorithm to adjust block generation according to the laxity time of the transaction. The two algorithms of block generation are compared in Figure 9 and Figure 10.

5.1. Algorithm of block generation at regular intervals (Conventional Blockchain)

Assuming that the time interval of block generation is T, the transaction arrival time is t, and the laxity time for block generation is *D*. When each block is generated at regular intervals, the deadline meet ratio, the average response time and the cost for block generation can be calculated as follows. A block is generated when a transaction arrives in the period *D*. Therefore, the deadline meet ratio is calculated as *D/T*. In Figure 9, if the period of *D* is greater than the period of *T*, a block is generated at any time within the period of block generation. The average response time is defined as *D/2*. The period *D* refers to the laxity time to wait for block generation. The cost for block generation in period of T is calculated as C_{block}/T. The graph of (a) algorithm for the cost is shown in Figure 10.

5.2. Algorithm of block generation by the shortest deadline of the arrived transactions (Real-time Blockchain)

If the transaction processing request arrives, and other transactions arrive at *1/f* intervals, all blocks about the arrived transactions within the laxity time of the first requested block generation are generated. Therefore, all transactions in the laxity time are processed based on the laxity time of the initially arrived transaction. It is assumed that the arrival frequency of transaction is *f*, and a new transaction arrives at equal intervals. Then, the arrival time of a new transaction is *1/f* (= *P*), and the block generation time of the new transaction can be calculated as *1/f* (= *P*) + *D*. All transactions are always processed when the transactions arrive in the laxity time. Therefore, the deadline meet

ratio of (b) algorithm is 1 as shown in Figure 9. The average response time for block generation is calculated as $(P + D)/2$. The graph in Figure 10 shows the cost for block generation of (b) algorithm calculated as $C_{block}/P + D$.

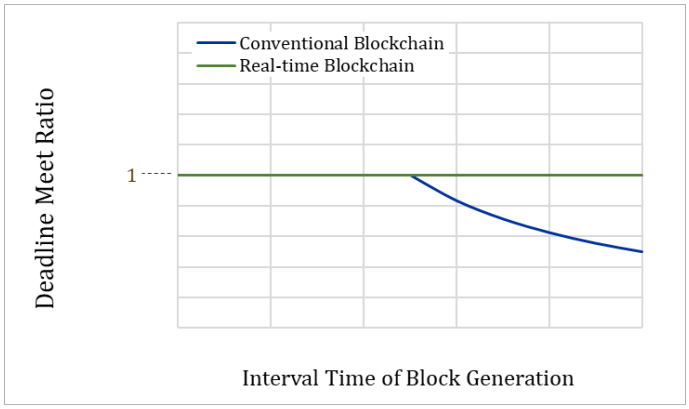

Figure 9: Deadline Meet Ratio

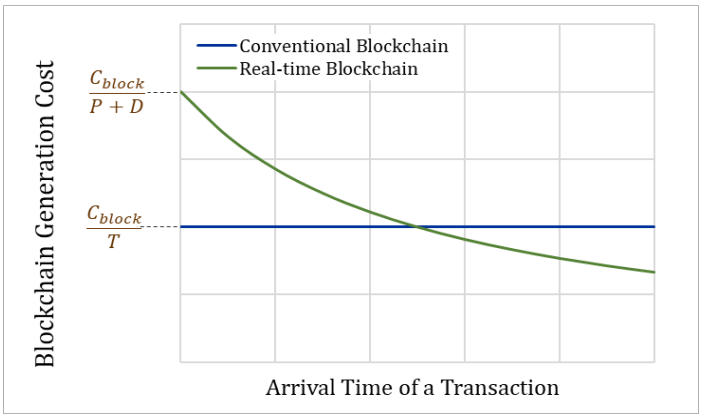

Figure 10: Blockchain Generation Cost

6. Conclusion

We propose a novel HBFT consensus algorithm required for transaction generation to reduce the processing throughput and increase scalability in a typical PBFT algorithm. Since our proposed algorithm utilize PBFT algorithm hierarchically, the number of exchanging messages in the consensus processing is significantly reduced. It means that the HBFT consensus algorithm ensures the scalability even though large numbers of nodes are involved in the consensus process. Furthermore, we calculate the optimal number of groups to identify the influence on the overall message complexity depending on the number of groups. In addition, both the best and worst cases are calculated to determine the ratio of allowable faulty nodes per group. Additionally, the reliability of byzantine fault tolerance is analyzed to compare the reliability of group case with the reliability of non-group case. Finally, it is necessary to consider the reduction of the deadline meet ratio according to the laxity time for consensus in computing environments required real-time transaction processing. To compare with the conventional Blockchain algorithm to consider the service level agreement (SLA), the real-time processing performance of transactions is analyzed according to the redundancy of nodes. In the future work, we plan to demonstrate HBFT algorithm through simulations. Furthermore, we would also implement Blockchain-based application services to effectively manage the decentralized IoT data.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1D1A1B07040573).

References

[1] O. Novo, "Blockchain Meets IoT: An architecture for Scalable Access Management in IoT," IEEE Internet of Things Journal, **5**, 1184-1195, 2018, doi: 10.1109/JIOT.2018.2812239.

[2] R. Casado-Vara, P. Chamoso, F. De la Prieta, J. Prietoa, J. M. Corchado, "Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management," Information Fusion, **49**, 227-239, 2019, doi: /10.1016/j.inffus.2018.12.007.

[3] P. Brody, V. Pureswaran, "Device democracy - Saving the future of the internet of Things," IBM institute for business value (2015) Available online: https://www.ibm.com/downloads/cas/Y5ONA8EV (accessed on 15 July 2020)

[4] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, M. Guizani, "A Blockchain-based Self-tallying Voting Scheme in Decentralized IoT," eprint arXiv:1902.03710, 2019.

[5] K. Košťál, P. Helebrandt, M. Belluš, M. Ries, I. Kotuliak, "Management and Monitoring of IoT Devices Using Blockchain," Sensors, **19**, 1-12, 2019, doi: /10.3390/s19040856.

[6] G. Sagirlar, B. Carminati, E. Ferrari, E. Ferrari, J. D. Sheehan, E. Ragnoli, "Hybrid-IoT: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-blockchains," 2018, doi: 10.1109/Cybermatics_2018.2018.00189.

[7] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, E. D. Sciascio, "Semantic Blockchain to Improve Scalability in the Internet of Things," Open Journal of Internet of Things (OJIOT), **3**, 46-61, 2017.

[8] R. Jayaraman, K. Salah, N. King, "Improving Opportunities in Healthcare Supply Chain Processes via the Internet of Things and Blockchain Technology," International Journal of Healthcare Information Systems and Informatics (IJHISI), **14**. 1-20, 2019, doi: 10.4018/IJHISI.2019040104.

[9] S. Gupta, M. Sadoghi, "Blockchain Transaction Processing," in Encyclopedia of Big Data Technologies, 1-11, 2019, doi:10.1007/978-3-319-63962-8_333-1.

[10] A. Panarello, N. Tapas, G. Merlino, F. Longo, A. Puliafito, "Blockchain and IoT Integration: A Systematic Survey," Sensors, **18**(8), 1-37, 2018, doi: 10.3390/s18082575

[11] A. Baliga, "Understanding blockchain consensus models," Persistent Systems Ltd. White paper, 2017, URL: https://www.persistent.com/wp-content/uploads/2018/02/wp-understanding-blockchain-consensus-models.pdf

[12] L. M. Bach, B. Mihaljevic, M. Zagar, "Comparative analysis of blockchain consensus algorithms," in Proceedings of 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 1545-1550, 2018.

[13] E. Zhang, "A Byzantine Fault Tolerance Algorithm for Blockchain," NEO White paper, 2018, URL: https://docs.neo.org/en-us/basic/consensus/whitepaper.html

[14] Y. Xiao, N. Zhang, W. Lou, Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," in IEEE Communications Surveys & Tutorials, **22**(2), 1432-1465, 2020, doi: 10.1109/COMST.2020.2969706.

[15] M. C.Igor, N. C.Vitor, P. A.Rodolfo, Y. Q.Wang, D.R.Brett, "Challenges of PBFT-Inspired Consensus for Blockchain and Enhancements over Neo dBFT," in future internet, **12**(8), 129, 2020

[16] M. Castro, B. Liskov, "Practical Byzantine Fault Tolerance," in Proceedings of the Third Symposium on Operating Systems Design and Implementation, 173-186, 1999.

[17] M. Castro, B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," ACM Transactions on Computer Systems, **20**(4), 398-461, 2002, doi: 10.1145/571637.571640.

[18] M. Kim, J-H. Kim, "Hierarchical Voting-based Byzantine Fault Tolerance Consensus Algorithm," in Proceedings of International Conference on APIC-IST 2020, 183-185, 2020

[19] M. Kim, J-H. Kim, "Decentralized Data Management Schemes for IoT Blockchain," in Ph.D. Dissertation, Department of Computer Engineering Graduate School of Ajou University, 2019