A Coordination Technique for Improving Scalability of Byzantine Fault-Tolerant Consensus

Jungwon Seo 1,*, Deokyoon Ko 2, Suntae Kim 3,4,* and Sooyong Park 4,*

1 Department of Computer Science & Engineering, Sogang University, 915 Ricci Hall, 35, Baekbeom-ro, Mapo-gu, Seoul 04107, Korea; jungwon@sogang.ac.kr
2 Noncelab. Inc., 802 Seoul Blockchain Center, 78, Mapo-daero, Mapo-gu, Seoul 04168, Korea; dykoh@noncelab.com
3 Department of Software Engineering, CAIIT, Jeonbuk National University, 567 Baekje-daero, Deokjin-gu, Jeonju-si, Jeollabuk-do 54896, Korea
4 Department of Computer Science & Engineering, Sogang University, 915A Ricci Hall, 35, Baekbeom-ro, Mapo-gu, Seoul 04107, Korea
* Correspondence: stkim@jbnu.ac.kr (S.K.); sypark@sogang.ac.kr (S.P.); Tel.: +82-63-270-4788 (S.K.)

Received: 15 September 2020; Accepted: 24 October 2020; Published: 28 October 2020

Abstract: Among various consensus algorithms, the Byzantine Fault Tolerance (BFT)-based consensus algorithms are broadly used for private blockchain. However, as BFT-based consensus algorithms are structured for all participants to take part in a consensus process, a scalability issue becomes more noticeable. In this approach, we introduce a consensus coordinator to execute a conditionally BFT-based consensus algorithm by classifying transactions. Transactions are divided into equal and unequal transactions. Moreover, unequal transactions are divided again and classified as common and trouble transactions. After that, a consensus algorithm is only executed for trouble transactions, and BFT-based consensus algorithms can achieve scalability. For evaluating our approach, we carried out three experiments in response to three research questions. By applying our approach to PBFT, we obtained 4.75 times better performance than using only PBFT. In the other experiment, we applied our approach to IBFT of Hyperledger Besu, and our result shows a 61.81% performance improvement. In all experiments depending on the change of the number of blockchain nodes, we obtained the better performance than original BFT-based consensus algorithms; thus, we can conclude that our approach improved the scalability of original BFT-based consensus algorithms. We also showed a correlation between performance and trouble transactions associated with transaction issue intervals and the number of blockchain nodes.

Keywords: consensus coordination; consensus algorithm; byzantine fault tolerance; PBFT; IBFT; hyperledger besu

1. Introduction

Recently, blockchain is considered one of the core technologies that enable us to create a transparent world with guaranteeing the integrity and transparency of data [1]. Significantly, researchers have tried to apply the private blockchain technologies that facilitate data sharing among permissioned participants to diverse areas such as business [2,3], and computer science [4–6]. However, due to the scalability issue, the application of technologies is limited [7–10]. Although there are diverse factors associated with scalability such as network bandwidth and cryptographic algorithms, a consensus algorithm is a key factor that significantly influences the issue, ensuring that participants keep maintaining the same data in distributed environments [11,12].

Among the consensus algorithms for private blockchains, the Byzantine Fault Tolerance (BFT)-based consensus algorithms are broadly used. PBFT (Practical Byzantine Fault Tolerance) [13,14] is a popular and
representative example. In the BFT-based consensus algorithm, the number of network communications among participants explodes with the increase in the number of participants. This is because all participants in the BFT-based consensus algorithm should be involved to complete the process for each transaction, and the process is even composed of diverse steps. This characteristic of the BFT-based consensus algorithm may raise the performance and scalability issue of the algorithm, as the more participants are newly involved, the slower the completion of the consensus process is [15].

There have been several previous studies on improving scalability of BFT-based consensus algorithms (see [16–23]). Some works tried to build sub-groups of blockchain nodes on a regular basis and reduced the number of network communications by executing two-step execution of the BFT-consensus algorithm: executing consensus inside sub-groups and conducting consensus between representatives of each sub-group. While this approach increased the PBFT algorithm’s scalability, it has a shortcoming that a new node cannot join any groups until new groups have been made. Other works also tried to optimize the number of communications by modifying the PBFT protocol with introducing a collector role or removing faulty nodes during consensus processes. Similarly, some works tried to reduce the number of prime node elections for minimizing the node election overhead. However, it is hard to expect the outstanding improvement of scalability through these approaches. In addition to these, other approach tried to deploy a new hardware-based BFT algorithm execution environment, but it has an apparent weakness that all nodes should prepare the specific hardware environments in advance.

To address the above issues, we propose a coordination technique for scalable Byzantine fault-tolerant consensus algorithms. The key idea is to introduce a Consensus Coordinator that controls conditional execution of a BFT-based consensus algorithm after classifying transactions of all nodes with respect to their equality. Our approach runs regularly associated with the block generation time interval, consisting of four steps. First, it starts with electing a prime node among all blockchain nodes for executing a BFT-based consensus algorithm and communicates with the consensus coordinator. Then, the consensus coordinator collects transactions from a transaction pool of each node. In the third step, the coordinator classifies transactions based on their equality and decides for executing a consensus algorithm. In the case that all transactions are equal, the coordinator lets the prime node execute block generation without executing a consensus algorithm, which is the completion of the synchronization in a blockchain network. When some transactions are not equal, the coordinator divides transactions into common and trouble transactions and requests the prime node to execute a BFT-based consensus algorithm only for trouble transactions. The prime node notifies agreed transactions to the coordinator. Finally, the coordinator sorts all common and agreed transactions by time order and requests the prime node to generate a new block containing all of processed transactions.

For the evaluation of our approach, we conducted three experiments for answering three research questions. We measured performance of the PBFT algorithm with and without our approach. In an experiment, the PBFT equipped with our approach obtained an average of 4.75 times better performance than only using the PBFT. In addition, we applied our approach to Hyperledger Besu using the IBFT (Istanbul Byzantine Fault Tolerance) consensus algorithm and showed a 61.81% performance improvement, compared to using only IBFT. We also presented correlation of performance and trouble transactions associated with transaction issue interval and the number of blockchain nodes. The contributions of our approach are summarized as follows:

- We propose a novel coordination technique to improve the scalability and performance of consensus algorithms, which is applicable to diverse BFT-based consensus algorithms.
- Our approach was implemented and applied to PBFT and Hyperledger Besu consensus algorithm, opened as an open source project for public access.
- We performed three experiments in response to three research questions and showed the feasibility of our approach.
The remainder of this paper is organized as follows. Section 2 presents BFT-based consensus algorithms as background and related work for improving the scalability of BFT-based consensus algorithms. Section 3 proposes our coordination technique and explains four steps for achieving the scalable BFT-based consensus algorithm in detail. Section 4 presents the evaluation of our approach by responding to our three research questions. Section 5 concludes our paper and discusses future works.

2. Background and Related Work

This section presents BFT-based consensus algorithms and their characteristics as background and introduces some previous works regarding how to improve BFT-based consensus algorithms' scalability.

2.1. Background: BFT-Based Consensus Algorithms

BFT (Byzantine Fault Tolerance)-based consensus algorithms indicate a group of consensus algorithms for resolving the byzantine general problem regarding how to achieve a consensus of data in an environment where normal and malicious nodes are mixed [24]. The representative example is PBFT (Practical Byzantine Fault Tolerance) [13,14] in the Hyperledger Fabric 0.6 (Hyperledger Fabric later changed the PBFT into Raft since version 2.0 [25]), and diverse variations of PBFT such as Tendermint in Cosmos [26], Hotstuff in Libra [27], and IBFT in Hyperledger Besu [28] are broadly used. BFT-based consensus algorithms' characteristics are the fast finality of a transaction, which indicates that the transaction is immediately finalized once the transaction is issued by a client and validated by \( N = 3f + 1 \) participants. However, in other consensus algorithms such as PoW and PoS, a client should wait until their transaction is contained in a new block after issuing transactions. In Bitcoin, for example, it takes 1 h to finalize transactions theoretically. In the worse case, it often takes more time when a block is forked. Despite the fast finality of BFT-based consensus algorithms, it is inevitable to decrease performance and scalability when the number of nodes is increased. This is because all participants should join the consensus process, and two over three nodes should agree with transactions by communicating with each other in four steps: pre-prepare, prepare, commit, and reply (Figure 1). Thus, this mechanism always causes the scalability issue depending on the number of nodes [15].

![Figure 1. The consensus process of the PBFT consensus algorithm.](image)

2.2. Related Work

Many approaches have been suggested to improve the scalability of BFT-based consensus algorithms. Most of their methods tried to reduce the number of network communications. To control the number of nodes participating in the consensus protocol, Feng et al. suggested the SDMA (Scalable Dynamic Multi-Agent)-PBFT approach that reduces the number of participants [16]. The approach
builds sub-groups among the peers and elect an agent as a primary node in each sub-group. Then, it carries out the consensus process in sub-groups at first, and the second consensus process is performed only among the agents. While this approach increases the PBFT algorithm’s scalability by reducing communication paths from the established blockchain network, it has a shortcoming that a new node cannot join any previously established groups until new groups have been made.

Similar to Feng et al.’s approach, Luu et al. proposed SCP (Scalable Byzantine Consensus Protocol) by executing the first consensus algorithm within sub-groups and the second consensus algorithm among group leaders from a result of the first execution [17]. The approach builds sub-groups by generating a random group number based on their IP address, public key, and nonce, while Feng et al.’s approach builds sub-groups by making the spanning-tree from a root node. Although this research contributed to reducing communication paths, it still has a similar problem that new nodes cannot easily join a blockchain network as in the approach of Feng et al.

As another approach, some research tried to optimize the number of communications by introducing collector role or removing faulty nodes during consensus processes. Kotl et al. proposed a new BFT-based consensus protocol, named Zyzzyva, where the number of non-faulty nodes for PBFT adaptively changes from \( N = 3f + 1 \) into \( N = 2f + 1 \) when a faulty node is detected during a consensus process [18]. Gueta et al. suggested that SBFT (State-of-the-art Byzantine Fault Tolerant) [19] reduces the number of communication paths among nodes by collecting messages in a consensus process into two collector nodes and validates messages in limited places. Similar to Gueta et al.’s approaches, Jiang et al. suggested HSBFT (High Performance and Scalable Byzantine Fault Tolerance), which makes a prime node to play a collector role that collects all messages and validates them [20]. HSBFT has a prime node electing process based on a node stable table containing identity number, state, IP, and public key. Based on the table, HSBFT excludes unstable nodes and optimizes communication paths. Although three approaches reduce the number of participant nodes and communication paths, it is hard to expect an outstanding improvement regarding scalability or performance.

In addition, Lei et al. proposed RBFT (Reputation-based Byzantine Fault Tolerance) algorithm for reducing communication paths in a private blockchain [21]. Each blockchain node computes their reputation score based on evaluation of their behaviors (e.g., a good behavior for generating a new block) and the number of permitted votes of each node is different depending on the reputation score. Votes are used to decide pass of PBFT’s steps by checking if the number of votes is over a specific threshold. Although it can reduce the number of communications, only limited nodes can have an out-sized influence on the voting process.

Some research tries to minimize the number of prime node election processes for enhancing scalability. Gao et al. proposed a trust eigen-based PBFT consensus algorithm, which is called T-PBFT [22]. In the approach, they tried to minimize the number of the prime node elections based on each node’s trust evaluation. Before starting the PBFT consensus process, the proposed eigen trust model evaluates all nodes’ trust scores and makes a group called a primary group. Then, the consensus process is composed of two steps: (1) the consensus within the primary group; and (2) the consensus between the remaining nodes and the primary group. It can improve scalability by reducing change the proportion of the single primary node. However, its number of communications is the same as the PBFT, so it is hard to expect a distinct improvement of scalability.

A new hardware-based execution environment is also introduced for improving performance of the BFT-based consensus algorithm. Liu et al. proposed a hardware-based BFT algorithm execution environment, which is named Fast BFT [23]. All nodes use a hardware chip (e.g., Intel SGX) for using TEE (Trusted Execution Environment) to execute the consensus algorithm and the TEE supports public key operation (e.g., multi signatures) during the consensus process. They also suggested the new Fast BFT algorithm that reduces verification steps by collaborating with TEE. While they improved the BFT algorithm’s scalability, their assumptions that all nodes should be executed upon TEE must be their limitations.
3. A Coordination Technique for Scalable BFT Consensus

This section presents a coordination technique for achieving the scalability of BFT-based consensus algorithms. Our approach is composed of two parts: Our Coordination Technique and BFT-based Consensus Algorithm (Figure 2). The BFT-based Consensus Algorithm part located at the bottom of the figure indicates traditional BFT-based algorithms such as PBFT and IBFT. It is composed of one prime node that controls consensus process and other general nodes similar to general BFT-based blockchain platforms. Each node has BFT-Module controlling the consensus process and generating new blocks and Transaction Pool maintaining unconfirmed transactions. Thus, the BFT-Module accesses transactions of the transaction pool regularly and executes a BFT-based consensus algorithm. Once all nodes have been achieved, the consensus of transactions, the BFT-Module produces a new block after accumulating agreed transactions.

Our Coordination Technique part corresponds to the top of the figure, and it is also located to the part above the BFT-based Consensus Algorithm part. In the technique, we newly introduce the Consensus Coordinator that controls each node’s BFT-based consensus algorithms’ conditional execution depending on the equality of transactions. Our consensus coordination technique consists of four steps. (1) The prime node is elected among all participating nodes. (2) The coordinator collects all transactions that existed in the transaction pool of each node. (3) The coordinator checks the equality of transactions and classifies transactions into common and trouble transactions. For trouble transactions, the coordinator requests a prime node to execute a consensus algorithm and obtains agreed transactions. (4) The coordinator merges common and agreed transactions and requests the controller of all nodes to execute block generation with merged transactions. In the following subsections, the steps are described in more detail.

![Figure 2. Overview of the coordination technique for scalable BFT-based consensus.](image)

3.1. Step 1. Electing a Prime Node

The first step is to elect a prime node among all participant nodes. The elected prime node plays a role of interacting with a consensus coordinator. This election step runs each regular t time (we assumed that all nodes have the same time unit through a logical clock or a physical clock algorithm (e.g., [29–31])). Once all steps are completed, this prime node election step is carried out again. The algorithm of this step is shown in Algorithm 1.
Algorithm 1 [All Nodes] Electing Prime Node

1: random = Random(seed) % N(Node)
2: if random equals to Node; then
3:    $\text{sig}_{\text{prime}} = \text{Signature} (\text{Node}_i, \text{seed})sk_{\text{prime}}$
4:    notify ($\text{sig}_{\text{prime}}, \text{pk}_{\text{prime}}$) to CC
5: end if

E lecting the prime node starts with a seed which is a previous block hash value, and modulates the random number with the seed by total number of nodes $N(node)$ to get a prime node number. The hash value of a previous block must be the same throughout all nodes because they are already agreed in a previous round. In the case that the result $random$ from the random algorithm is equal to a unique number of each node $Node_i$ that is assigned before, a node is elected as a prime node. After signing its node number $Node_i$ and seed with its private key $sk_{\text{prime}}$, it notifies the consensus coordinator CC with the result of sign $\text{sig}_{\text{prime}}$ and its public key $pk_{\text{prime}}$.

3.2. Step 2. Collecting Transactions from Transaction Pool

Once the coordinator gets the prime node election notification from the prime node, it collects all transactions from each node’s transaction pool as the second step. This collection step is presented in Algorithm 2. The input parameters of this step are $\text{sig}_{\text{prime}}$ and $pk_{\text{prime}}$ from the prime node. Then, the coordinator checks elected prime node’s validity by generating a random number with the delivered seed and $Node_i$ (see Lines 3–5). If the generated random is equals to $Node_i$ received from the prime node, the consensus coordinator requests all nodes to send all transactions accumulated in the transaction pool of each node between the previous time $time_p$ to the current time $time_c$. When random is different from $Node_i$, the coordinator terminates all coordination steps of this round and waits for the next idle state.

Algorithm 2 [Coordinator] Collecting Transactions from TxPool

1: Inputs: $\text{sig}_{\text{prime}}, pk_{\text{prime}}$
2: Initialize: $\text{Txs} \leftarrow \{\}$
3: $Node_i, seed = \text{Signature}(\text{sig}_{\text{prime}}, pk_{\text{prime}})$
4: random = Random(seed) % N(Node)
5: if random equals to $Node_i$ then
6:    for $i=0, i \leq N(Node), i++$ do
7:        $\text{Txs}_i \leftarrow \text{request} Node_i$ to send $Tx_{(time_p \sim time_c)}$ of $\text{TxPool}_i$
8:    end for
9: else
10:    Terminate
11: end if

3.3. Step 3. Processing Equal/Unequal Transactions

Based on collected transactions from each node, this step decides the execution of the BFT-based consensus algorithm. Figure 3 shows processing steps for collected transactions. At first, the coordinator checks if all transactions collected from each node are equal to those from other nodes. Thus, when all transactions are equal, it executes the Handling Equal Transactions step, and then this third step is terminated. When transactions are not the same, it first classifies transactions into common and trouble transactions. The coordinator then requests the prime node to execute a consensus algorithm only for trouble transactions and gets agreed transactions. Finally, the coordinator merges and sorts common and agreed transactions in time order.
3.3.1. Step 3.1 Handling Equal Transactions

The coordinator performs this step if all transactions from each node are equal. Algorithm 3 shows detailed steps for handling equal transactions. To check transactions’ equality, a set of transactions from each node is converted into a hash function (see Lines 3–5) first, and their equality is compared. Due to the characteristic of the hash function, their hash values must be the same if all transactions are the same.

Algorithm 3 [Coordinator] Handling Equal Transactions

1: Inputs:
   \[ Txs = \{ Txs_0, Txs_1, ... Txs_N \} \]
2: Initialize:
   \[ TxsList \leftarrow \{ \} \]
3: for \( i=0, i \leq N(Node), i++ \) do
4: \[ TxsList_i \leftarrow \text{Hash}(Txs_i) \]
5: end for
6: if All hashes of TxsList are equal then //Handle Equal Transactions
7: \[ \text{isConfirmed}, sig'_{prime}, pk_{prime} \leftarrow \text{request Node}_{prime} \text{ to confirm } Txs_0 \]
8: if isConfirmed == true then
9: \[ \text{request} \text{ All Nodes to generate a new block with } Txs_0 \text{ and } sig'_{prime} \]
10: \[ time_p \leftarrow time_c \]
11: else
12: Terminate
13: end if
14: else //For Unequal Transactions
15: handle Unequal Transactions(Txs)
16: end if

If all transactions are the same, the coordinator requests the prime node Node_{Prime} to confirm transactions \( Txs_0 \) and the prime node responses to the confirmation of transactions. This confirmation step is necessary for mutual trust between the coordinator and the prime node on integrity of transactions from the coordinator. The response from the prime node includes isConfirmed, sig'_{prime}, and pk_{prime}. Among the responses, sig'_{prime} results from execution \( \text{Signature}(Txs_0)_{sk_{prime}} \) of the prime node (see Lines 7–8).
When the prime node confirms all equal transactions, the coordinator requests all nodes of blockchain network to generate a new block. Then, the controller receives the request and delegates the request to the BFT-Module to generate a new block. All nodes’ \( \text{time}_p \) is updated with the current \( \text{time}_c \) for designating the starting period for the next round (see Line 10). If the prime node does not confirm transactions, this round is terminated and \( \text{time}_p \) remains at the previous time. In addition, when some of the transactions are different, the coordinator performs the handle unequal transactions step presented in the next subsection.

### 3.3.2. Step 3.2 Handling Unequal Transactions

This step is executed by the coordinator when some of the transactions are not equal. This step is composed of three sub-steps: (1) classifying transactions; (2) executing a consensus algorithm; and (3) sorting all transactions.

**1) Classifying Transactions:** In this step, the coordinator classifies all transactions into common transactions and uncommon transactions throughout all transactions from each node. For uncommon transactions, we rename them as trouble transactions. Algorithm 4 shows the classification step. The output of the classification is stored in \( \text{List}_{\text{comm}} \) and \( \text{List}_{\text{tr}} \). The classification step is very intuitive. With the boolean flag \( \text{isCommon} \), it iterates all transactions of each node and checks if a transaction exists in their transaction lists (see Lines 3–18).

**Algorithm 4 [Coordinator] Classifying Transactions**

1. **Inputs:**
   \( Txs = \{ Txs_0, Txs_1, ... Txs_n \} \)
2. **Initialize:**
   \( \text{List}_{\text{comm}} \leftarrow \{ \} \), \( \text{List}_{\text{tr}} \leftarrow \{ \} \), \( \text{List}_{\text{agg}} \leftarrow \{ \} \)
3. for \( i=0, i \leq N(\text{Node}) \), \( i++ \) do
4.   for \( j=0, j \leq N(Txs_i) \), \( j++ \) do
5.     \( \text{isCommon} \leftarrow \text{true} \)
6.     for \( k=0, k \leq N(\text{Node}) \) & \( i \neq k, k++ \) do
7.       if \( \text{Tx}_j \) does not exist in \( Txs_k \) then
8.         \( \text{isCommon} \leftarrow \text{false} \)
9.       break
10. end if
11. end for
12. if \( \text{isCommon} == \text{true} \) then
13.   \( \text{List}_{\text{comm}} \leftarrow \text{Tx}_j \)
14. else
15.   \( \text{List}_{\text{tr}} \leftarrow \text{Tx}_j \)
16. end if
17. end for
18. \( \text{List}_{\text{agg}} \leftarrow \text{request} \text{Node}_{\text{prime}} \) to execute a Consensus Algorithm(\( \text{List}_{\text{tr}} \))

**2) Executing a Consensus Algorithm:** For trouble transactions, \( \text{List}_{\text{tr}} \) from the previous sub-step, the coordinator requests the prime node to execute a consensus algorithm and obtains a list of agreed transactions \( \text{List}_{\text{agg}} \) from the prime node, denoted on Line 19 Algorithm 4. It should be noted that any BFT-based consensus algorithms can be applied in this step. All transactions in \( \text{List}_{\text{tr}} \) cannot be contained in \( \text{List}_{\text{agg}} \), because some of the transactions might not be completed in the consensus algorithm. At that time, transactions not agreed upon are removed according to the BFT-based consensus algorithm.

**3) Sorting All Transactions:** Based on common transactions \( \text{List}_{\text{comm}} \) and agreed transactions \( \text{List}_{\text{agg}} \), the coordinator merges and sorts them in time order in this step. Then, the coordinator requests the prime
node to confirm merged transactions similar to the process of equal transactions (see Algorithm 5, Line 5). The $\text{sig}''$prime is produced by the prime node using $\text{Signature} (\text{SortedList}_{\text{cscn}})k_{\text{prime}}$. If transactions are confirmed, the coordinator requests all nodes to generate a new block with the $\text{SortedList}_{\text{cscn}}$, and all nodes' $time_p$ is updated by the current time $time_c$.

Algorithm 5 [Coordinator] Sorting the Merged Transactions

1: Inputs:
   List$_{\text{comm}}, \text{List}_{\text{agg}}$
2: Initialize:
   List$_{\text{cscn}} \leftarrow \{\}$, SortedList$_{\text{cscn}} \leftarrow \{\}$
3: List$_{\text{cscn}} \leftarrow (\text{List}_{\text{comm}} \cup \text{List}_{\text{agg}})$
4: SortedList$_{\text{cscn}} \leftarrow \text{sort} (\text{List}_{\text{cscn}})$
5: isCon$\text{firmed}, \text{sig}''$prime, $pk_{\text{prime}} \leftarrow \text{request Node}_{\text{prime}}$ to confirm SortedList$_{\text{cscn}}$
6: if $\text{isCon}f\text{irmed} == \text{true}$ then
7:   request All Nodes to generate a new block with SortedList$_{\text{cscn}}$ and sig$''$prime
8:   $time_p \leftarrow time_c$
9: else
10:   Terminate
11: end if

3.4. Step 4. Generating a New Block

The last step is to generate a new block that is relied on blockchain platforms. The coordinator does not intervene in this final step. Thus, all nodes create a new block with transferred transactions and a previous block’s hash. The controller, for requesting the BFT-Module to generate a new block and accessing transaction pools, is developed for each blockchain platform and our approach can apply to diverse BFT-based consensus algorithms.

4. Evaluation

This section describes our experiments’ results designed to evaluate our approach. For this evaluation, we established the three research questions below and carried out three experiments in response to research questions.

- RQ1: How much can the scalability of PBFT be increased through our proposed approach?
- RQ2: What is the correlation between the trouble transactions and performance?
- RQ3: How much can our approach improve the scalability of IBFT of Hyperledger Besu?

4.1. RQ1: How Much Can the Scalability of PBFT be Increased through our Proposed Approach?

This first research question is intended to figure out how much our suggested approach achieves our research aim which is to improve the scalability of the BFT-based consensus algorithm. We selected and implemented the PBFT consensus algorithm for this research question, which is the most popular BFT-based consensus algorithm. Then, we measured the performance of the PBFT equipped with and without our approach. Furthermore, we increased the number of nodes to figure out the scalability of our approach.

Experimental setting for RQ1. To respond to RQ1, we built the PBFT network (Our source code for implementing the PBFT network is available at https://github.com/jungwons/JwRalph_Seo/tree/master/lab/Agent_Consensus) based on the Castro and Liskov’s research [13,14]. Initially, we structured four nodes and issued transactions every 10 ms for 10,000,000 (ms), so that we transmitted one million transactions for 2 h 40 min. In addition, we set a block generation time to be 10 ms, which implies that each node generates a new block every 10 s with transactions in their transaction pools (by using the sensitivity analysis, we obtained 10 s as the best block generation time
for the best performance.). Then, we measured the total elapsed time until the consensus process of transactions is complete and computed an average elapsed time. We prepared 81 physical computers and deployed 80 PBFT nodes and one consensus coordinator into each computer. The hardware specification of each computer was Intel i5-3570 3.4 GHz CPU with 4GB RAM, and Windows 10 OS installed.

Experimental Result for RQ1. Figure 4 shows the result of the experiment. In the initial experiment with four nodes, the PBFT with the consensus coordinator expressed in \( CC + PBFT \) achieved 0.0328 s for each transaction on average, while PBFT without our approach denoted as \( PBFT \) showed 0.1237 s. The gap of the elapsed time of two approaches grew as the number of nodes increased. When the number of nodes reached 80, the elapsed time of PBFT and \( CC + PBFT \) became 1.1191 and 6.2212 s, respectively. Thus, the PBFT equipped with our approach obtained 3.77 times (=0.1237/0.0328) higher performance than PBFT with the initial four nodes. The performance gain was increased so that the PBFT with our approach achieved 5.56 (=6.2212/1.1191) times higher performance with 80 nodes. In all experiments throughout node changes, the performance of the PBFT with our approach increased an average of 4.75 times compared to the use of the PBFT alone. Therefore, we can recognize that our approach contributed to improving the performance of the PBFT consensus algorithm.

![Figure 4. Elapsed time comparison between PBFT and PBFT equipped with the Consensus Coordinator (CC).](image)

In addition to this, we observed that the elapsed time increase rate of the PBFT is bigger than that of \( CC + PBFT \). While the increase rate of the PBFT from 4 to 80 nodes is 50.29 times (=6.2212/0.1237), that of the \( CC + PBFT \) is 34.12 (=1.1191/0.0328). It implies that our approach contributes to improving PBFT consensus algorithm’s scalability depending on the increase of nodes compared to only using the PBFT. The reason for the increase of the elapsed time of PBFT is because all nodes in the PBFT algorithm should participate in consensus process and the number of communications. In addition, the consensus process should always be executed for all transactions (i.e., one million transactions). However, our approach checks the equality of transactions and executes the consensus process only for trouble transactions. Thus, depending on the proportion of trouble transactions associated with the number of the nodes, the elapsed time of \( CC + PBFT \) is increased but not as steeply as that of PBFT.
4.2. RQ2: What is the Correlation between the Trouble Transactions and Performance?

The second research question is for finding out how much our approach can contribute to the performance improvement of the BFT-based consensus algorithm. In the real-world, all blockchain nodes issue transactions, respectively, and it may rarely happen that all transactions in a transaction pool are equal. According to Donet and Pérez-Solà’s experiment [32], transaction propagation time in the Bitcoin network composed of 344 nodes took 35 min on average, which means that transactions in each node’s transaction pool are commonly different. In this research question, we build an environment where each node has many trouble transactions as in the real-world by controlling the interval of the transaction issue time and computed correlation between the elapsed time and the proportion of trouble transactions as $\tau$.

Experimental setting for RQ2. To simulate the environment, we started with the experimental setting for RQ1 with the same hardware specification and issued one million transactions by controlling interval of transaction issue time from every 15 to 1 ms. Then, we measured a total elapsed time and obtained an average elapsed time for each transaction by dividing the total elapsed time by the number of transactions, as shown in Table 1. In addition to this, we observed the proportion of the trouble transaction $\tau$ in the consensus coordinator to obtain the correlation between trouble transactions and the elapsed time.

Experimental results for RQ2. Table 1 shows the result of the experiment. In the table, Columns 15 ms, 10 ms, 5 ms, 2 ms and 1 ms denote the time interval of the transaction issue and we only selected some of the representative time intervals. We computed $\tau$ by averaging the proportion of trouble transactions in a transaction pool collected from each node for every $time_p \sim time_c$ period (i.e., $10$ s). Based on the $\tau$, we highlight cells with the same colors associated with its $\tau$ value (see Color Legend in the table). When the number of nodes is 16, and a transaction is issued every 15 ms, our approach obtained $0.099$ s for each of the average time interval (see the italic in the table). Depending on decreasing the transaction issue time interval, the average elapsed time and $\tau$ were increased. In addition, an increase in the number of nodes causes an increase in the average elapsed time and $\tau$.

| N(Node) | 15 ms | 10 ms | 5 ms | 2 ms | 1 ms |
|---------|-------|-------|------|------|------|
| 16      | 0.099 | 0.120 | 0.244| 0.330| 0.406|
| 28      | 0.186 | 0.253 | 0.316| 0.405| 0.521|
| 40      | 0.232 | 0.438 | 0.533| 0.753| 0.835|
| 52      | 0.462 | 0.813 | 1.100| 1.308| 1.672|
| 64      | 0.714 | 1.000 | 1.355| 1.915| 2.511|
| 76      | 0.713 | 1.805 | 2.106| 2.781| 3.155|

Color Legend $\tau = 0.1x$ $\tau = 0.2x$ $\tau = 0.3x$ $\tau = 0.4x$ $\tau > 0.5x$

From the result, we established the correlation between $\tau$ and the average elapsed time for a transaction $Avg.\ Elap.\ Time_{tx}$ based on the transaction issue time interval $tit$ and the number of nodes $N(Node)$ as Equation (1). The equation implies that the average elapsed time is $\delta$ times proportional to the proportion of the trouble transaction $\tau$. In addition, $\tau$ is inversely proportional to the transaction issue interval $tit$ and it has logarithmic relation with the number of nodes $N(Node)$. In the experiment, we obtained $\delta = 2.5$, $\alpha = 2$, and $\beta = 0.1$, which indicates that the proportion of the trouble transaction strongly affects the average elapsed time for each transaction.

$$\text{Avg.}\ Elap.\ Time_{tx} \approx \delta \ast \tau \approx \frac{1}{tit} + \beta \log(N(Node))$$ (1)
4.3. RQ3: How Much Can Our Approach Improve the Scalability of IBFT of Hyperledger Besu?

We established the third research question regarding the applicability of our approach to the real-world open-source blockchain framework using another BFT-based consensus algorithm. We selected Hyperledger Besu, a popular implementation of Ethereum client that supports public and private blockchain. It uses IBFT (Istanbul Byzantine Fault Tolerance) that enhances the performance by decreasing the number of nodes for transaction confirmation from \(3f + 1\) to \(2f + 1\) (IBFT https://github.com/ethereum/EIPs/issues/650). In the experiment for RQ3, we modified the Hyperledger Besu source code to communicate with our Controller for requesting new block generation and accessing transaction pool (our source code for implementing the BFT network is available at https://github.com/jungwonrs/JwRalph_Seo/tree/master/lab/besu_backup). Then, we performed an experiment similar to that of RQ1 to figure out how much our approach can improve the IBFT consensus protocol’s performance.

Experimental setting of RQ3. We modified Hyperledger Besu 1.5.1 (https://github.com/hyperledger/besu/tree/1.5.1) to communicate with our consensus coordinator. For the experiment, we transmitted random transactions to Hyperledger Besu on a regular basis during the designated period. Initially, the number of nodes was four and we gradually increased the number of nodes until it reached 40. Then, we measured the number of transactions contained in generated blocks to recognize its throughput. We set the block generation time of the Hyperledger Besu to 10 s, and our coordinator execution interval was also set into 10 s because this time showed the best performance. The experiment was also carried out every 5 min (=300 s and 30 block generations) for each node configuration.

In the experiment, the interval of the transaction transmission started from 10 ms, because a significant number of transactions was missed in Hyperledger Besu in the case of under 10-ms interval. We carried out this experiment with the transaction transmission interval of 5 ms from 10 to 40 ms. Our hardware specification as a blockchain node and consensus coordinator was Intel i7-8700, 3.2 GHz CPU with 24 GB RAM and Windows 10 OS. All nodes and the consensus coordinator were executed on one computer. Due to the hardware specification limitation, the maximum node number of Hyperledger Besu was set to 40. In addition, the gas limitation of Hyperledger Besu was removed to generate transactions continuously (we refer to the method shown on the official Besu website: https://besu.hyperledger.org/en/stable/HowTo/Configure/FreeGas/).

Experimental results of RQ3. Figure 5 shows the result of the experiment where its transmission interval is 10 ms with from 4 to 40 blockchain nodes. In the figure, results of only using the IBFT and IBFT equipped with our approach are expressed in IBFT and CC + IBFT, respectively. The y-axis indicates the number of transactions contained in the generated blocks for 5 min. For four blockchain nodes, the numbers of IBFT and CC + IBFT contained in 30 generated blocks were 24,173 and 26,152, respectively. CC + IBFT achieves 8.19% (=26,152 − 24,173)/24,173) performance improvement. The total number of transactions that can be issued every 10 ms for 5 min is 30,000, but 3848 (=30,000 − 26,152) and 5827 (=30,000 − 24,173) are missed due to the performance limitation of Hyperledger Besu and our approach. For the experiment with 40 blockchain nodes, IBFT processed 1563 transactions, while CC + IBFT processed 6607 transactions, indicating a 322.71% (=6607 − 1563)/1563) performance improvement. Thus, in all node configurations with 10 ms of transaction issue interval, the combination of IBFT and our consensus coordinator obtained 37.75% (=154,782 − 112,361)/11,2361) performance improvement on average.

Figure 6 shows the result of the experiment where the interval is 25 ms. As the transaction time interval is 25 ms, the total number of transactions that can be issued for 5 min is 12,000 (=300/0.025), which is the maximum number of transactions that can be contained in generated blocks. In four blockchain nodes, the numbers of IBFT and CC + IBFT are 11,900 and 12,000, respectively, in which most of the issued transactions are contained in generated blocks. Thus, the gap between the two numbers of transactions is not big. However, for 20 nodes, CC + IBFT achieved 53.25% (=11,255 − 7344)/7344) performance improvement. In addition, CC + IBFT obtained 344.81% (=7584 − 1705)/1705) performance improvement in the case of 40 blockchain nodes, compared to only
using the IBFT. In all node configurations, the combinational use of IBFT and our approach gained 61.81% ($\frac{(103,348 - 63,868)}{63,868}$) performance improvement on average. Thus, it is possible to conclude that our approach contributed to improving the performance of a specific blockchain node configuration and improving the IBFT consensus algorithm’s scalability because the loss of performance is smaller depending on the increase of the blockchain nodes. All datasets resulting from this experiment are presented in the AppendixA.

![Figure 5](image)

**Figure 5.** Comparing the amount of transactions between IBFT and CC + IBFT when the transaction generation interval is 10 ms.

While carrying out this experiment, we observed that the proportion of equal and unequal transactions in consensus coordinator, as shown in Table 2. As the coordinator execution interval is 10 s, our approach’s maximum number of execution is 30 for 5 min. Then, we counted the number of cases that carry out equal transactions, that is, the case of Step 3.1 Handling Equal Transactions and the number of unequal transaction cases that process Step 3.2 Handling Unequal Transactions. Each of them is expressed in Equal Txs and Unequal Txs in the table. In four nodes with 10 and 25 ms, all transactions of the transaction pool of all nodes are equal, which are 93.33% and 96.67% of 30 coordinator executions, respectively. However, the number of nodes is increased, the higher proportion of the case for equal transactions is decreased, and that of unequal transactions is increased. As a result, average equal and unequal transactions are 55.33% and 44.67% in the case of 10-ms transaction issue interval, while those of the 25-ms case are 72% and 28%, respectively. Thus, we recognized that our contribution to the performance and scalability improvement is positively associated with the proportion of equal transactions, as pointed by Equation (1).
Table 2. Result of monitoring equal and unequal transactions of consensus coordinator.

| N(Node) | 10 ms Equal Txs | Unequal Txs | 25 ms Equality | Inequality |
|---------|-----------------|-------------|---------------|------------|
| 4       | 28 93.33%      | 2 6.67%     | 29 96.67%     | 1 3.33%    |
| 8       | 27 90%         | 3 10%       | 30 100%       | 0 0%       |
| 12      | 25 83.33%      | 5 16.67%    | 29 96.67%     | 1 3.33%    |
| 16      | 20 66.67%      | 10 33.33%   | 26 86.67%     | 4 13.33%   |
| 20      | 19 63.33%      | 11 36.67%   | 23 76.67%     | 7 23.33%   |
| 24      | 16 53.33%      | 14 46.67%   | 20 66.67%     | 10 33.33%  |
| 28      | 14 46.67%      | 16 53.33%   | 16 53.33%     | 14 46.67%  |
| 32      | 10 33.33%      | 20 66.67%   | 17 96.67%     | 13 43.33%  |
| 36      | 4 13.33%       | 26 86.67%   | 14 46.67%     | 16 53.33%  |
| 40      | 3 10.00%       | 27 90.00%   | 12 40%        | 18 60%     |
| Average | 166 55.33%     | 134 44.67%  | 216 72%       | 84 28%     |

Figure 6. Comparing the amount of transactions between IBFT and CC + IBFT when the transaction generation interval is 25 ms.

4.4. Threats to Validity

Construct Validity. The results of RQ1, RQ2, and RQ3 may be influenced by hardware specification and the version of Hyperledger Besu. Although there are diverse factors related to performance, the experimental results, such as elapsed times and the number of transactions, can differ. However, we tried to carry out our experiments on the same hardware specification for control and experimental groups, so that the relative comparison of the result is considered desirable for our experiment. In addition, the result of the experiment for RQ3 can be different due to the version of Hyperledger Besu. We selected the 1.5.1 version of Hyperledger Besu in the experiment for RQ3, which was the most recent version at the time. However, the version upgrade is frequent so that the use of different versions would show different results.

Content Validity. In this paper, the concept of scalability is measured by the extent of the decrease in performance, depending on the increase in the number of nodes. Based on this definition, we keep measuring the performance gap, depending on node changes. Besides, we also defined that the term transaction issue indicates that a client issues one transaction, and the transaction is contained in a new block through block generation. However, the block generation interval was set to be every 10 s in
the experiments for RQ1 and RQ3, while the unit of the measure of the elapsed time must be 10 s, which are not the exact time. Due to this issue, we performed our experiment for 2 h 40 min, which is a long time enough to ignore the 10 s time gap for measuring the average elapsed time for processing a transaction. In the experiment for RQ3, we fixed the experiment time into 30 block generations (i.e., 5 min) to resolve the issue.

**Internal Validity.** Experiment results may be affected by different sets of the interval execution of the consensus coordinator (i.e., $time_p \sim time_c$) and the block generation time for PBFT and IBFT in Hyperledger Besu. To handle this issue, we performed the sensitivity analysis and observed that setting the execution time interval of the coordinator and block generation time of PBFT and IBFT to 10 ms showed the best performance. However, the performance may be different depending on the interval setting.

**External Validity.** In this paper, we claim that our approach is efficient for BFT-based consensus algorithms, and we applied our approach to two consensus algorithms: PBFT and IBFT. It is hard to claim that our approach applies to all BFT-based consensus algorithms. However, we selected the most popular BFT-based consensus algorithm, PBFT. Many consensus algorithms such as IBFT, Zyzzyva [18], SBFT [19], Hotstuff [33,34], and Tendermint [26] are derived from PBFT. Although we selected IBFT as a representative derivation of PBFT, we argue that our approach can be applied to other BFT-based consensus algorithms.

5. Conclusions

This paper proposes a coordination technique for improving the scalability of BFT-based consensus algorithms. The technique is composed of four steps: (1) electing a prime node; (2) collecting transactions from transaction pools; (3) processing equal and unequal transactions; and (4) generating a new block. Our key idea is to control a conditional execution of the consensus algorithm by dividing the transaction pool into equal and unequal transactions and secondly dividing common and trouble transactions among unequal transactions. The consensus algorithm is then executed only for trouble transactions, and the results are merged and finalized through sharing the transactions throughout all blockchain nodes.

Based on this approach, we carried out three experiments to respond to three research questions. As a result of the experiments, the use of PBFT equipped with our approach showed 4.75 times the performance improvement on average compared to using PBFT only. In addition, our approach contributed to improving the performance by a maximum of 61.81% of the performance, compared to the single-use of IBFT. In addition to this, we showed the correlation of performance and trouble transactions associated with the transaction issue interval and the number of blockchain nodes.

Although our approach showed the scalability improvement of BFT-based consensus algorithms, it has explicit limitations. First, the consensus coordinator is centralized so that it exposes the coordinator to the single point failure issue. Second, our approach does not address the recovery issue of the coordinator when a system has failed or restarted. Third, our approach should be tested in the real-world environment where diverse synchronization issues exist such as clock synchronizations throughout distributed nodes. For future work, we plan to carry out more research on distributing the centralized consensus coordinator and establishing recovery strategy from the system failure in the real-world environment.

**Author Contributions:** Conceptualization, J.S. and D.K.; Funding acquisition, S.P.; Investigation, J.S.; Methodology, S.K.; Supervision, S.P.; Writing—original draft, J.S., D.K. and S.P.; and Writing—review and editing, S.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2017-0-01628) supervised by the IITP (Institute for Information & communications Technology Promotion).

**Conflicts of Interest:** The authors declare no conflicts of interest.
## Appendix A. Raw Data of RQ3

| Node Count | 10 ms IBFT | CC + IBFT | 25 ms IBFT | CC + IBFT | Node Count | 10 ms IBFT | CC + IBFT | 25 ms IBFT | CC + IBFT |
|------------|------------|-----------|------------|-----------|------------|------------|-----------|------------|-----------|
| 1          | 846        | 880       | 684        | 401       | 1          | 766        | 766       | 623        | 421       |
| 2          | 783        | 916       | 259        | 476       | 2          | 566        | 566       | 155        | 421       |
| 3          | 588        | 876       | 382        | 400       | 3          | 685        | 685       | 320        | 401       |
| 4          | 767        | 916       | 388        | 401       | 4          | 587        | 776       | 309        | 376       |
| 5          | 692        | 916       | 391        | 476       | 5          | 684        | 684       | 339        | 401       |
| 6          | 817        | 916       | 387        | 401       | 6          | 678        | 780       | 335        | 376       |
| 7          | 745        | 916       | 391        | 401       | 7          | 661        | 701       | 339        | 391       |
| 8          | 803        | 926       | 392        | 401       | 8          | 601        | 756       | 345        | 401       |
| 9          | 809        | 936       | 392        | 401       | 9          | 648        | 766       | 340        | 397       |
| 10         | 803        | 926       | 392        | 376       | 10         | 655        | 756       | 348        | 401       |
| 11         | 809        | 916       | 392        | 376       | 11         | 623        | 757       | 341        | 397       |
| 12         | 833        | 916       | 392        | 476       | 12         | 659        | 659       | 344        | 401       |
| 13         | 815        | 916       | 392        | 401       | 13         | 612        | 759       | 340        | 401       |
| 14         | 876        | 939       | 391        | 376       | 14         | 633        | 777       | 352        | 397       |
| 15         | 787        | 941       | 393        | 376       | 15         | 657        | 757       | 348        | 401       |
|             | 4          |            |             |            | 8          | 627        | 775       | 352        | 401       |
| 17         | 875        | 875       | 391        | 401       | 17         | 652        | 787       | 349        | 397       |
| 18         | 840        | 840       | 392        | 376       | 18         | 649        | 777       | 348        | 401       |
| 19         | 803        | 803       | 392        | 397       | 19         | 662        | 662       | 353        | 401       |
| 20         | 873        | 873       | 393        | 376       | 20         | 615        | 764       | 349        | 397       |
| 21         | 795        | 795       | 392        | 401       | 21         | 679        | 679       | 349        | 401       |
| 22         | 874        | 874       | 394        | 376       | 22         | 640        | 776       | 352        | 401       |
| 23         | 761        | 761       | 392        | 401       | 23         | 636        | 765       | 350        | 411       |
| 24         | 851        | 851       | 393        | 376       | 24         | 659        | 659       | 351        | 401       |
| 25         | 834        | 834       | 392        | 401       | 25         | 637        | 779       | 353        | 401       |
| 26         | 805        | 805       | 391        | 376       | 26         | 641        | 840       | 351        | 401       |
| 27         | 832        | 832       | 392        | 401       | 27         | 652        | 797       | 351        | 401       |
| 28         | 802        | 802       | 392        | 401       | 28         | 654        | 665       | 350        | 401       |
| 29         | 833        | 833       | 393        | 401       | 29         | 647        | 776       | 349        | 401       |
| 30         | 825        | 825       | 392        | 401       | 30         | 653        | 817       | 312        | 401       |

| sum 24,173 | 26,152     | 11,900    | 12,000     | sum 19,418 | 22,263     | 10,397    | 12,000     |

| Node Count | 10 ms IBFT | CC + IBFT | 25 ms IBFT | CC + IBFT | Node Count | 10 ms IBFT | CC + IBFT | 25 ms IBFT | CC + IBFT |
|------------|------------|-----------|------------|-----------|------------|------------|-----------|------------|-----------|
| 1          | 530        | 656       | 504        | 391       | 1          | 532        | 176       | 261        | 387       |
| 2          | 586        | 664       | 292        | 401       | 2          | 532        | 756       | 261        | 387       |
| 3          | 596        | 656       | 296        | 401       | 3          | 476        | 616       | 261        | 401       |
| 4          | 576        | 624       | 296        | 401       | 4          | 477        | 626       | 276        | 387       |
| 5          | 556        | 673       | 276        | 401       | 5          | 571        | 627       | 276        | 387       |
| 6          | 576        | 676       | 296        | 401       | 6          | 571        | 627       | 277        | 387       |
| 7          | 546        | 677       | 276        | 401       | 7          | 571        | 627       | 277        | 411       |
| 8          | 576        | 656       | 276        | 401       | 8          | 476        | 633       | 277        | 387       |
| 9          | 546        | 656       | 296        | 396       | 9          | 476        | 626       | 277        | 387       |
| 10         | 576        | 656       | 276        | 401       | 10         | 532        | 573       | 261        | 387       |
| 11         | 576        | 677       | 276        | 401       | 11         | 489        | 626       | 287        | 377       |
| 12         | 577        | 678       | 286        | 391       | 12         | 477        | 676       | 277        | 387       |
| 13         | 576        | 656       | 276        | 401       | 13         | 456        | 626       | 261        | 387       |
| 14         | 576        | 674       | 276        | 401       | 14         | 497        | 176       | 261        | 387       |
| 15         | 578        | 663       | 292        | 401       | 15         | 556        | 327       | 261        | 387       |

| sum 16,917 | 19,808     | 8909      | 12,000     | sum 15,474 | 17,669     | 8005      | 11,690     |
| Node | Count |
|------|-------|
| IBFT | CC + IBFT | IBFT | CC + IBFT |
| 1 | 463 | 536 | 241 | 371 |
| 2 | 474 | 576 | 241 | 371 |
| 3 | 462 | 563 | 241 | 371 |
| 4 | 451 | 563 | 231 | 416 |
| 5 | 389 | 576 | 231 | 371 |
| 6 | 422 | 576 | 231 | 356 |
| 7 | 434 | 576 | 231 | 371 |
| 8 | 473 | 575 | 241 | 371 |
| 9 | 474 | 575 | 241 | 371 |
| 10 | 474 | 570 | 241 | 371 |
| 11 | 432 | 576 | 241 | 376 |
| 12 | 464 | 572 | 241 | 386 |
| 13 | 464 | 572 | 241 | 386 |
| 14 | 454 | 556 | 241 | 371 |
| 15 | 462 | 572 | 241 | 371 |
| 16 | 460 | 576 | 241 | 371 |
| 17 | 459 | 556 | 241 | 371 |
| 18 | 456 | 546 | 241 | 401 |
| 19 | 454 | 576 | 241 | 371 |
| 20 | 452 | 576 | 241 | 371 |
| 21 | 427 | 576 | 241 | 371 |
| 22 | 454 | 526 | 241 | 416 |
| 23 | 434 | 576 | 277 | 371 |
| 24 | 456 | 556 | 241 | 371 |
| 25 | 456 | 556 | 241 | 371 |
| 26 | 451 | 576 | 241 | 371 |
| 27 | 451 | 456 | 241 | 371 |
| 28 | 441 | 576 | 241 | 371 |
| 29 | 441 | 576 | 241 | 371 |
| 30 | 441 | 556 | 241 | 371 |

| Node | Count |
|------|-------|
| IBFT | CC + IBFT | IBFT | CC + IBFT |
| 5  | 397 | 280 | 319 |
| 1  | 416 | 136 | 326 |
| 2  | 416 | 136 | 331 |
| 3  | 260 | 256 | 331 |
| 5  | 216 | 166 | 326 |
| 6  | 452 | 168 | 326 |
| 7  | 274 | 167 | 296 |
| 8  | 416 | 136 | 296 |
| 9  | 330 | 136 | 326 |
| 10 | 409 | 130 | 326 |
| 11 | 416 | 130 | 326 |
| 12 | 377 | 136 | 301 |
| 13 | 363 | 136 | 299 |
| 14 | 416 | 280 | 309 |
| 15 | 397 | 141 | 326 |
| 16 | 416 | 63 | 326 |
| 17 | 224 | 256 | 326 |
| 18 | 237 | 256 | 309 |
| 19 | 416 | 84 | 309 |
| 20 | 374 | 177 | 314 |
| 21 | 374 | 220 | 326 |
| 22 | 416 | 141 | 326 |
| 23 | 416 | 141 | 326 |
| 24 | 416 | 141 | 326 |
| 25 | 416 | 141 | 326 |
| 26 | 416 | 179 | 338 |
| 27 | 416 | 136 | 343 |
| 28 | 401 | 176 | 326 |
| 29 | 416 | 134 | 326 |
| 30 | 375 | 161 | 326 |

| Node | Count |
|------|-------|
| IBFT | CC + IBFT | IBFT | CC + IBFT |
| 1  | 397 | 280 | 319 |
| 2  | 416 | 136 | 326 |
| 3  | 260 | 256 | 331 |
| 5  | 216 | 166 | 326 |
| 6  | 452 | 168 | 326 |
| 7  | 274 | 167 | 296 |
| 8  | 416 | 136 | 296 |
| 9  | 330 | 136 | 326 |
| 10 | 409 | 130 | 326 |
| 11 | 416 | 130 | 326 |
| 12 | 377 | 136 | 301 |
| 13 | 363 | 136 | 299 |
| 14 | 416 | 280 | 309 |
| 15 | 397 | 141 | 326 |
| 16 | 416 | 63 | 326 |
| 17 | 224 | 256 | 326 |
| 18 | 237 | 256 | 309 |
| 19 | 416 | 84 | 309 |
| 20 | 374 | 177 | 314 |
| 21 | 374 | 220 | 326 |
| 22 | 416 | 141 | 326 |
| 23 | 416 | 141 | 326 |
| 24 | 416 | 141 | 326 |
| 25 | 416 | 141 | 326 |
| 26 | 416 | 179 | 338 |
| 27 | 416 | 136 | 343 |
| 28 | 401 | 176 | 326 |
| 29 | 416 | 134 | 326 |
| 30 | 375 | 161 | 326 |

sum 13,515 16,903 7,344 11,255 sum 9,521 15,271 5,138 10,402
### References

1. Swan, M. *BlockChain BluePrint for a New Economy*, 1st ed.; O’Reilly Media: Sebastopol, CA, USA, 1978; ISBN 978-149-192-049-7.

2. Cong, L.W.; He, Z. Blockchain Disruption and Smart Contracts. *Rev. Financ. Stud.* 2019, 32, 1754–1797, doi:10.1093/rfs/hhz007.

3. Catalini, C.; Gans, J.S. Some Simple Economics of The Blockchain. *Commun. ACM* 2019, 63, 80–90, doi:10.3386/w22952.

4. Banerjee, M.; Lee, J.H; Choo, K.K.R. A blockchain future for internet of things security: A position paper. *Digit. Commun. Netw.* 2018, 4, 149–160, doi:10.1016/j.dcan.2017.10.006.

5. Azaria, A.; Ebblaw, A.; Lippman, A. MedRec: Using blockchain for Medial Data Access and Permission Management. In Proceedings of the 2016 2nd International Conference on Open and Big Data(OBD), Vienna, Austria, 22–24 August 2016; pp.25–30.

6. Korpela, K.; Hallikas, J.; Dahlberg, T. Digital Supply Chain Transformation toward Blockchain Integration. In Proceedings of the 50th Hawaii International Conference on System Sciences, Hawaii, HI, USA, 4 January 2017; pp. 4182–4191.

7. Pelz-Sharpe. Available online: https://www.deep-analysis.net/wp-content/uploads/2019/08/DA-190812-Ent-Blockchain-forecast.pdf (accessed on 20 December 2019).

8. Kim, S.; Kwon, Y.; Cho, S. A Survey of Scalability Solution on Blockchain. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 17–19 October 2018; pp. 1204–1207.

9. Chauhan, A.; Malviya, O.P.; Verma, M.; Singh, T.M. Blockchain and Scalability. In Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 16–20 July 2018; pp. 122–128.

10. Scherer, M. Performance and Scalability of Blockchain Networks and Smart Contract. Available online: http://www.diva-portal.org/smash/record.jsf?pid=diva2:1111497 (accessed on 1 March 2020).

11. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X. Blockchain Challenges and Opportunities: A Survey. *Int. J. Web Grid Serv.* 2018, 14, 352–375, doi:10.1504/IJWGS.2018.095647.
12. Cachin, C.; Vukolic, M. Blockchains Consensus Protocols in the Wild. Available online: https://arxiv.org/abs/1707.01873 (accessed on 21 December 2019).
13. Castro, M.; Liskov, R. Practical Byzantine Fault Tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999, pp. 173–186.
14. Castro, M.; Liskov, R. Practical Byzantine Fault Tolerance and Proactive recovery. ACM Trans. Comput. Syst. 2002, 20, 398–461, doi:10.1145/571637.571640.
15. Sukhwani, H.; Martinez, J.M.; Chang, X.; Trivedi, K.S.; Rindos, A. Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric). In Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, China, 26–29 September 2017, pp. 253–255.
16. Feng, L.; Zhang, H.; Chen, Y.; Lou, L. Scalable Dynamic Multi-Agent Practical Byzantine Fault-Tolerant Consensus in Permissioned Blockchain. Appl. Sci. 2018, 8, 1919–1940, doi:10.3390/app8101919.
17. Luu, L.; Narayanan, V.; Bawoja, K.; Zheng, C.; Gilbert, S.; Saxena, P. SCP: A Computationally-Scalable Byzantine Consensus Protocol For Blockchains. Available online: https://eprint.iacr.org/2015/1168/20160823:024020 (accessed on 15 March 2020).
18. Kotla, R.; Alvisi, L.; Dahlin, M. Zyzzyva: Speculative Byzantine Fault Tolerance. ACM Trans. Comput. Syst. 2010, 27, 45–58, doi:10.1145/1658357.1658358.
19. Gueta, G.G.; Abraham, I.; Grossman, S. SBFT: A Scalable and Decentralized Trust Infrastructure. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, 24–27 June 2019; pp. 568–580.
20. Jiang, Y.; Lian Z. High Performance and Scalable Byzantine Fault Tolerance. In Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019, China, pp. 1195–1202.
21. Lei, K.; Zhang, Q.; Xu, L.; Qi, Z. Reputation-Based Byzantine Fault-Tolerance for Consortium Blockchain. In Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed System, Singapore, 11–13 December 2018, pp. 604–611.
22. Gao, S.; Yu, T.; Zhu, J.; Cai, W. T-PBFT: An Eigen Trust-based practical Byzantine fault tolerance consensus algorithm. China Commun. 2019, 16, doi: 10.23919/JCC.2019.12.008.
23. Liu, J.; Li, W.; Karama, G.O.; Asokan, N. Scalable Byzantine Consensus via Hardware-Assisted Secret Sharing. IEEE Trans. Comput. 2019, 68, doi:10.1109/TC.2018.2860009.
24. Lamport, L.; Shostak, R.; Pease, M. The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst. 1982, 4, 382–401, doi:10.1145/3335772.3335936.
25. Sousa, J; Bessani, A.; Vukolic, M. A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. In Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Luxembourg, 25–28 June 2018, pp. 51–58.
26. Kwon, J. Tendermint: Consensus without Mining. Available online: https://tendermint.com/docs/tendermint.pdf (accessed on 20 May 2020).
27. Amsden, Z.; Arora, R.; Bano, S.; Baudge, M.; Blackshear, S.; Bothra, A.; Cabrera, G.; Catalini, C.; Chalkias, K.; Cheng, E.; et al. The Libra Blockchain, Available online: https://developers.libra.org/docs/the-libra-blockchain-paper (accessed on 2 July 2020).
28. Hyperledger Besu 1.5 Performance Enhancement. Available online: https://www.hyperledger.org/category/hyperledger-besu (accessed on 1 September 2020).
29. Fan, K.; Sun, S.; Yan, Z.; Pan, Q.; Li, H.; Yang, Y. A blockchain-based clock synchronization Scheme in IoT. Future Gener. Comput. Syst. 2019, 101, doi:10.1016/j.future.2019.06.007.
30. Bertasi, P.; Bonazza, M.; Moretti, N.; Peserico, E. PariSync: Clock synchronization in P2P networks. In Proceedings of the 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Brescia, Italy, 12–16 October 2009, pp. 1–6.
31. Iwanicki, K.; van Steen, M.; Voulgaris, S. Gossip-Based Clock Synchronization for Large Decentralized Systems. Self-Manag. Netw. Serv. Syst. Serv. 2006, 3996, doi: 10.1007/11767886_3.
32. Donet Donet, J.A.; Pérez-Solà, C. The Bitcoin P2P Network. In Proceedings of the Financial Cryptography and Data Security FC2014, Christ Church, Barbados, 7 March 2014; pp. 87–102.
33. Yin, M.; Malikhi, D.; Reiter, M.K.; K.Reiter, M.; Gueta, G.G; Abraham. I. HotStuff:BFT Consensus with Linearity and Responsiveness. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, Toronto, ON, Canada, 29 July–2 August 2019; pp.347–356.
34. Yin, M.; Malkhi, D.; Reiter, M.K.; Gueta, G.G; Abraham, I. HotStuff:BFT Consensus in the Lens of Blockchain. Available online: https://arxiv.org/abs/1803.05069 (accessed on 30 May 2020).

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).