A Scalable Multi-layer PBFT Consensus for Blockchain

Wenyu Li, Chenglin Feng, Lei Zhang, *Senior Member, IEEE*, Hao Xu, Bin Cao, *Member, IEEE* and Muhammad Ali Imran, *Senior Member, IEEE*

Abstract—Practical Byzantine Fault Tolerance (PBFT) consensus mechanism shows a great potential to break the performance bottleneck of the Proof-of-Work (PoW) based blockchain systems, which typically support only dozens of transactions per second and require minutes to hours for transaction confirmation. However, due to frequent inter-node communications, PBFT mechanism has a poor node scalability and thus it is typically adopted in small networks. To enable PBFT in large systems such as massive Internet of Things (IoT) ecosystems and blockchain, in this paper, a scalable multi-layer PBFT based consensus mechanism is proposed by hierarchically grouping nodes into different layers and limiting the communication within the group. We first propose an optimal double-layer PBFT and show that the communication complexity is significantly reduced. Specifically, we prove that when the nodes are evenly distributed within the sub-groups in the second layer, the communication complexity is minimized. The security threshold is analyzed based on faulty probability determined (FPD) and faulty number determined (FND) models respectively. We also provide a practical protocol for the proposed double-layer PBFT system. Finally, the results are extended to arbitrary-layer PBFT systems with communication complexity and security analysis. Simulation results verify the effectiveness of the analytical results.

Index Terms—PBFT, communication complexity, node scalability, consensus mechanism, blockchain.

1 INTRODUCTION

The consensus mechanism/algorithm, which orders transactions and guarantees the integrity with consistency of the blockchain across geographically distributed nodes, is of importance to blockchains, which is the backbone of groundbreaking decentralized ledger technology and cryptocurrency. It provides secure, accountable, and immutable and low-cost solutions. Thus, it has shown great potential in various sectors such as financial services, energy trading, supply chain management, Internet of Things (IoT), etc [1].

The consensus algorithms largely determine the performance of distributed system especially for blockchains, such as transaction throughput, latency, node scalability, security level, etc. Depending on application scenarios and performance requirements, different consensus algorithms can be considered. In the case of a permission-less public blockchain, nodes are allowed to join or leave the network without permission and authentication; therefore proof based algorithms such as Proof-of-Work (PoW) [2], Proof-of-Stake (PoS) [3], and their variants are commonly used in many public blockchain applications, where the cryptocurrency such as Bitcoin is the most well-known one. Proof based algorithms are designed with excellent node scalability performance through nodes competition, which is essential to deal with the double-spending problem. However, they could be very resource demanding. For instance, recently published estimates of bitcoin's electricity consumption are wide-ranging, on the order of 20-80 TWh annually, or about $0.1\% - 0.3\%$ of global electricity consumption [4]. Also, these consensus mechanisms have other limitations, such as long transaction confirmation latency and low throughput. For instance, the Transaction Per Second (TPS) is generally limited to 7 in Bitcoin and about 15 in Ethereum, while the transaction confirmation delay is typically as considerable as 10 minutes in Bitcoin and 15 seconds in Ethereum [5]. It is worth pointing out the computational requirement of Proof-based consensus varies from one to another, the notable examples of non-computing Proof-based consensus is InterPlanetary File System (IPFS), a distributed file system uses the concept of proof-of-space/ space-time [6]. Though, Proof-based consensus has been mostly seen in the applications of public blockchain, it has a limited generic distributed system and blockchain coverage, as it is still incrementally resource demanding, and the search of voting-based consensus for blockchain and new generation distributed system is imminent.

Unlike the public blockchain, the private and consortium blockchains prefer to adopt lighter consensus protocols such as PBFT, Paxos [7] and Raft [8] [9] to reduce the amount of computational power and improve the transaction throughput [10]. They have been widely used in general distributed systems for data synchronization, meanwhile, their property is also critically important to the application scenarios of the blockchain-enabled IoT ecosystem, which is typically composed of low cost and low power devices. Though some private chain suitable consensus only enables the Crash Fault Tolerance (CFT) [8], as it does not protect the integrity of transactions from malicious attacks, but they are acceptable for private blockchain where the nodes are trusted.

1.1 PBFT applied to blockchain

To protect distributed systems from malicious users, Practical Byzantine Fault Tolerance (PBFT) was proposed in [11] as an improved and practical protocol based on original Byzantine Fault Tolerance (BFT) [12] [9]. Comparing to the Proof based consensus such as PoW, where the security threshold is 51%, i.e., absolute secure transaction can be achieved if the malicious user(s) occupies no more than half of the overall resource, PBFT requires the number of malicious users under 33% of total participants to ensure the system immune from the malicious attacks [11].

PBFT is favoured for private and consortium chains, thanks to the lower complexity and low energy consumption, which is particularly important for wireless IoT applications [13]. A promising

W. Li, C. Feng, L. Zhang (Corresponding author), H. Xu, M. Imran are with James Watt School of Engineering, University of Glasgow, United Kingdom (e-mail:{2357476L, 2357707F}@student.gla.ac.uk; {Lei.Zhang, Muhammad.Imran}@glasgow.ac.uk; H.Xu.2@research.gla.ac.uk). B. Cao is with Beijing University of Posts and Telecommunications of China (email: caobin65@163.com). *The work was supported in part by the UK EPSRC under grant number EP/S02476X/1.*

advancement of PBFT can be found in Hyperledger development [14], part of Hyperledger business blockchain frameworks, which has been adopted by tech giants like IBM or Wall Street Fintech, such as J.P. Morgan [15].

1.2 Motivations

Though PBFT has shown good performance in terms of latency, resource requirement and nodes complexity, the node scalability, which is a metric to measure how well the network reflects the capacity of the system to handle the increasing number of nodes, is a bottleneck of PBFT since it relies on heavy inter-node communications. Thus, from the communication complexity perspective, the PBFT based blockchain hardly scales up to 100 nodes [16].

Variant PBFT-based consensuses have been proposed to solve the problem of poor scalability. For example, PBFT with short-lived signature [17], minimizes the consensus time for signature verifying. Another significant evolution path is sharding, where shards have their own consensus group; hence the transactions can be confirmed within a short time because of the smaller size network [18] [19]. However, the trade-off includes increased communication costs and lowered security levels. For instance, every shard keeps its own data, which is not shared with other shards. In the case of users requesting contracts on several shards, inter-node communications go up quickly. Meanwhile, putting segments of data into different repositories without proper redundancy and recoverability is risky. Losing control of any individual shard will interrupt the blockchain completely, either causing untraceable and irreversible records to future records or forking the chain from the breaking point [20].

The communication complexity issues in the PBFT system is due to the exhaustive peer to peer communication among the nodes. To reduce the cost of communication, intuitively, one can construct a hierarchical multi-layer PBFT by refraining the communication within their layers or sub-groups. The sub-consensus can be performed per group, and the overall consensus can be defined as exceeds the number of groups achieved the sub-consensus. This multi-layer PBFT system model is initially proposed in [21], providing a brief analysis of communication complexity based on a particular case. However, there are still many challenges to be addressed before this idea can be applied in a real system. Firstly, the complexity analysis derived in the [21] cannot be applied to generic situations as it is developed under the premise that the number of nodes in each sub-group is equal. To better represent the practical situation, detailed analysis should be provided considering various node allocation scenarios. Also, the security analysis in [21] is derived under the hypothesis that faulty nodes only exist in the bottom layer, which does not apply to real situations where faulty nodes are randomly distributed into all layers. Therefore, new security analysis should also be provided to verify the reliability of multi-layer PBFT system. Finally, to ensure liveness and safety of the network, a new complete protocol is also needed.

1.3 Contributions and organizations

In this paper, we first propose a double-layer PBFT system, as shown in Fig. 1, where we have m replicas in the first layer and n sub-layer replicas in the second layer, which give a total number of $1 + m + mn$ nodes in the network. Then we extend this system model to a more general X-layer case ($X > 2, X \in \mathbb{I}$) to analyze and compare communication complexity in multi-layer systems. The security analysis shows that the security performance

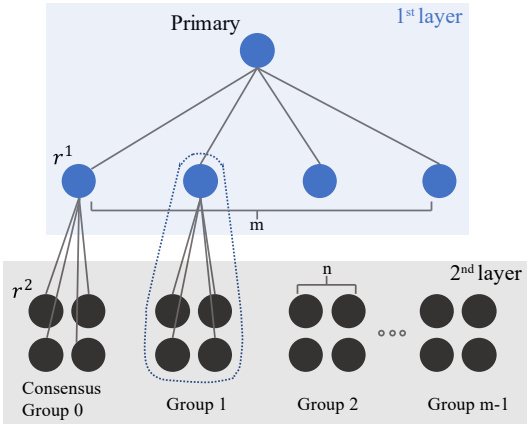

Fig. 1. Topology of the proposed double-layer PBFT system. (Note that we give double-layer system as an example here, the idea proposed in this paper can be extended to arbitrary-layer PBFT.)

is improved over time, and the consensus algorithm accommodates at most $\lfloor \frac{n}{3} \rfloor \times \lfloor \frac{m}{2} \rfloor$ faulty nodes to ensure the absolute safety of the system under malicious attacks. Meanwhile, only $m^2 + mn^2$ inter-node messages, instead of $O(Z^2)$ messages in a traditional single-layer PBFT network consensus, are required.

However, it is inevitable that the scalability is improved at the cost of a longer delay as the consensus-reaching process goes through more than one layer. As a result, the proposed multi-layer PBFT would not be appropriate for systems such as some financial scenarios in consortium blockchain which requires both node scalability and low latency transaction. On the contrary, in other applications such as blockchain-enabled scenarios with massive IoT devices connection, where scalability is a bottleneck but not latency, multi-layer PBFT provides a solution with greatly improved node scalability while protecting the system from malicious attacks [22] [23] [24]. For instance, blockchain can be used to play an important role for tracking and managing all elements involved in supply chain from raw materials to the final products to ensure the quality of the manufacturing [25]. Given the fact that there are many player within the system, traditional single layer PBFT based blockchain is hard to scale up. With multi-layer PBFT, the suppliers could be divided into groups and layers and consensus can be reached among them to provide reliable information for different parties and ensure the efficiency of manufacturing.

Note that, in such hierarchical scenarios, the consensus is reached in a network with biased rights in different layers. Indeed, we have seen the most consensus models in blockchain have treated their nodes with the equality, however, it is also common to find examples of inequality in real-life for upper level scalability and availability; for example, the practice of liquid democracy, has shown a good influence on social stability and performing the best interests of represented members, in which case, the participants are still in active power after the upper-tier delegate is elected. Another common example of master/slave (leader/follower) model of data storage has produced promising results regarding the availability and I/O performance. Moreover, a lack of flexibility and liquid can be added with the proposed dynamic multi-layer design. The multi-layer PBFT and the advanced system together form a rolling, robust and flexible consensus for not only digital

distributed systems, but also prompts real-life impacts. From this point of view, the proposed PBFT consensus serves as a viable solution to the current society to a trade-off between an efficient but low secure, centralized architecture and a highly secure but low efficient distributed one.

To summarize, this paper makes the following contributions

- This paper first introduces a novel double-layer PBFT model. This model is scalable since it reduces the inter-node communications to $C \approx 1.9Z^{\frac{4}{3}}$, comparing to the traditional PBFT system of $O(Z^2)$.
- Second, the analytical security performance of the proposed system is derived. It proves that under certain conditions, the maximum number of faulty nodes can increase from $\lfloor \frac{m}{3} \rfloor$ to $\lfloor \frac{n}{3} \rfloor \times \lfloor \frac{m}{2} \rfloor$.
- Third, a new double-layer PBFT protocol is introduced, based on which consensus can be reached among nodes in different layers.
- Finally, a general X-layer PBFT model is proposed, which is proven to have the minimum communication complexity reduced to linear $C = \frac{16Z-16}{3}$ when the network depth is maximized to X_{max}. Additionally, the security threshold is derived.

The rest of the paper is organized as follows. Section 2 reviews the related work. In Section 3, the double-layer PBFT model is proposed. Then, the communication complexity is analyzed and compared to the original PBFT. In Section 4, the security threshold is derived based on the double-layer system. A general X-layer system is proposed and analyzed in Section 5. Section 6 proposes the protocols for double-layer PBFT system and Section 7 concludes the paper.

2 RELATED WORK

Variant consensus algorithms have been designed for permissioned blockchain to provide safety under trustless environment with different performances [26]. Practical Byzantine Fault Tolerance (PBFT) [11] is one of the most popular State Machine Replication (SMR) technology, which provides $\frac{1}{3}$ optimal tolerance under malicious attacks with low latency. In recent years, with the growing interest in blockchain, many new protocols based on SMR are proposed. In *The Next 700 BFT Protocols*, the author present a method to simplify the designing of new protocols by introducing Abortable Byzantine faulT toleRant stAte maChine replicaTion (Abstract) as a new way to illustrated BFT [27]. Results show that the proposed protocol by using Abstract provides a better performance in terms of both latency and throughput.

Apart from the application in permissioned blockchain, the concept of quorum certificate is also borrowed in Ethereum Casper to provide safety for Ethereum 2.0 [28]. It is an extra mechanism on top of PoS and serves as a finality gadget. The Casper does not generate the block but determines the sequence of blocks on chain. The designated replicas (validators in Ethereum) votes for a parent-children relationship for two collections of blocks and form a quorum certificate granted by more than $\frac{2}{3}$ of replicas. Considering that Ethereum 2.0 updates rapidly, it is difficult to draw a conclusion on the application of shards and Casper. The sharding may sacrifice safety and increase communication complexity, and the actual implementation of Casper in Ethereum 2.0 also affects it.

One problem PBFT protocols may encountered is the difficulty to implement. BFT-SMART is so far the most popular open-source library for BFT-SMR application based on Java, filling the gap between the literature work and practical implementation [29]. Evaluation shows that the throughput of BFT-SMART reaches more than 80,000 TPS. Also, although it is simpler than other BFT implementing systems, it still contains 13.5K lines of Java code, which is much more complicated than the implementing systems for Paxos [26].

Another bottleneck of PBFT is its scalability. As mentioned in the introduction, the communication complexity limits the performance of protocols. Former researchers have also proposed various solutions. The HotStuff leverages threshold signature to reduce communication complexity [30]. In each phase, the primary broadcasts messages and each replica responses a valid message with a partial signature. The primary collects them responses with partial signature more than $\frac{2}{3}$ replicas and combine into a digital signature. This signature is broadcasted by replica again, which serves as a quorum certificate. Unlike HotStuff, the multi-layer PBFT proposed improves node scalability by reorganizing network structure, where the threshold signature can also be applied to further reduce the communication complexity. Moreover, it is noticeable that as the network scales up, the primary in HotStuff has to collect and broadcasts an increasing number of messages and combines more partial signatures. This workload can be barked down by implementing our tree-like structure. Also, the pace of HotStuff is affected by the primary since it waits for the aggregation of partial signature to advance.

3 COMMUNICATION COMPLEXITY OF DOUBLE-LAYER PBFT

3.1 Original single-layer PBFT

Before introducing the system model of double-layer PBFT, the protocol and communication complexity of original single-layer PBFT [11] are briefly analyzed in this subsection. Fig. 2 shows the protocol diagram of the original single-layer PBFT. As an example, we assume there is one primary node and three state machine replicas. The consensus is triggered by a client sending a request to the nodes' header (Replica 0). Then consensus will be operated among the nodes, and if an agreement is reached among the replicas, the new record will be committed to the blockchain, vice versa. The whole consensus process includes three stages $pre-prepare$, $prepare$ and $commit$ as shown in Fig. 2. On receiving the request from the client, the primary node (i.e., Replica 0) broadcasts a $pre-prepare$ message to the other nodes. In $prepare$ and $commit$ stages, all replicas send messages to check the validity of received messages. In each stage, a minimum number of consistent messages are required for stepping into the subsequent stage.

The consensus is technically reached when the commit phase is successful among the majority of non-faulty nodes. Specifically, the client must receive at least $f+1$ replies (f denotes the number of faulty nodes in the group) from the nodes to validate the final consensus with a total number of $3f+1$ replicas. This ensures that at least one non-faulty replica replies to this operation. In the case of the client fails to collect $f+1$ replies, the client may resend the request to primary for retry. Upon receiving the same request again, if the consensus is already reached on the commit phase, replicas just resend the final stage messages. If the consensus is not reached, the network goes over the protocol again.

From Fig. 2, we can see that PBFT is a communication demanding protocol. Given the total node number Z, the original single-layer PBFT requires $O(Z^2)$ times of inter-node communications to reach consensus. Obviously, the system is not scalable since the complexity burden is non-affordable when Z is large (i.e., thousands).

In the next, a scalable multi-layer PBFT system and protocol is proposed to reduce the communication complexity. The performance analysis and protocol design of a double-layer system will be introduced first, and then the arbitrary-layer system will be proposed in Section 5.

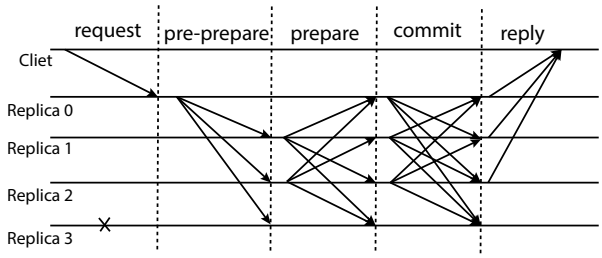

Fig. 2. Single-layer PBFT consensus processing [11].

3.2 Protocol overview of double-layer PBFT

In the double-layer PBFT protocol, scalability is improved by recursively inserting the PBFT consensus algorithm between commit and reply phases as the sub-layer algorithm. The higher layer forms a certificate, which proves that the consensus was reached. With this certificate, a node in the first layer initiates a PBFT consensus reaching process among second layer nodes.

As illustrated in Fig. 1, there are m replicas and a primary node in the first layer. Each replica in the first layer forms a consensus group with n sub-layer replicas in the second layer. The primary invokes a new operation by multicasting $pre-prepare$ messages to replicas in the first layer with information about this operation. The replica accepts valid $pre-prepare$ requests and step into prepare phase by multicasting $prepare$ messages within the first layer. If one replica receives no less than $2f$ identical $prepare$ messages from other first-layer replicas, that operation is prepared on this replica. A prepared replica then multicasts $commit$ messages among first-layer replicas.

Similarly, this operation is committed after collecting $2f$ identical $commit$ messages. At this point, this particularly first-layer replica is considered as the primary node in its consensus group and starts consensus reaching by multicasting new $preprepare$ messages to its sub-layer nodes. For instance, in Fig. 1, a committed node r^1 sends $pre-prepare$ messages to node r^2 and other replicas in $ConsensusGroup0$. Second-layer replicas repeat the process mentioned above until the commit phase. Finally, all committed replicas in the second layer send reply messages to primaries, and primaries reply to the client. This protocol will be stated in detail in Section 6.

3.3 Communication complexity analysis of double-layer PBFT

The double-layer PBFT model is proposed in Fig. 1, where the first-layer leader controls m replicas, each of which serves as a primary node of the n sub-layer replicas in the second layer. Therefore, there are $1+m+mn$ nodes in the system. Note that here we assume each sub-group in the second layer has the same number of nodes, and a generic case will be discussed in *Proposition 11*. Based on this, the communication complexity can be calculated as the following proposition.

Proposition 1. *For a double-layer PBFT system with m replicas in the first layer and n sub-layer replicas in each sub-group, the communication complexity C to reach consensus is*

$$C = (m+1)^2 + m(n+1)^2. \tag{1}$$

Proof is derived in Appendix.

In the next, we aim to find the optimal setup of a double-layer system with given Z nodes to provide the lowest communication cost. When the overall number of nodes Z is given to form a double-layer PBFT, we can assign either larger groups (i.e., smaller m and larger n) or a larger number of groups (i.e., larger m and smaller n). *Proposition 9* gives the best grouping algorithm in terms of minimizing the communication complexity.

Proposition 2. *For a double-layer system containing Z nodes in total, the minimum communication complexity can be achieved when n equals to the nearest integer to the real positive root of following equation:*

$$n^3 + 3n^2 + n = 2Z - 1, \quad (3 \le n \le \frac{Z-4}{3}). \tag{2}$$

Proof is derived in Appendix.

Note that, *Proposition 9* is based on the assumption that the system is full, in other words, the number of sub-layer replicas in each sub-group is equal.

Proposition 9 provides with a pragmatic method to design a double-layer PBFT system with minimum C. In the next, we try to find a direct function of C vs. Z so that the communication complexity of any double-layer system can be estimated analytically.

Proposition 3. *When m and n are fixed by optimal allocation and the system is full, the relationship between communication complexity C and total node number Z can be written as*

$$C \approx 1.9Z^{\frac{4}{3}}. \tag{3}$$

Proof is derived in Appendix.

Fig. 3 compares the analytical and estimated results from *Proposition 9* and *Proposition 10* respectively and validates the availability of equation (27).

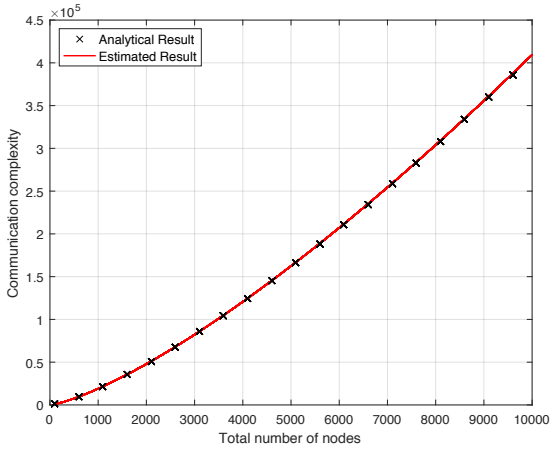

Fig. 3. Comparison of analytical and estimated results of communication complexity for double-layer PBFT.

From Fig. 3 we can see that compared with original single-layer PBFT, where C is quadratic of Z, the communication complexity is greatly reduced in double-layer PBFT systems. For example, the communication complexity of a system with 1000 nodes is reduced by two orders of magnitude, from 10^6 to 2×10^4. Unfortunately, the communication complexity of the double-layer PBFT system is still not linear against the node number; instead, it is of $\frac{4}{3}$ order.

In the following proposition, we focus on the setup of the systems that are not full. In other words, the number of sub-layer replicas in each sub-group may be different.

Proposition 4. *If the double-layer system is not full, the communication complexity reaches a smaller value when the vacancies are equally distributed into the sub-groups. The minimum value can be reached by distributing vacancies to the minimum number of sub-groups.*

Proof is derived in Appendix.

Through double-layer PBFT, complexity can be significantly lowered compared to the original single-layer PBFT. However, the proposed system may cause a longer delay. In addition, with such a topology, the security performance should be analytically investigated to guide the actual system deployment.

4 SECURITY ANALYSIS

4.1 Security threshold analysis

In the double-layer PBFT, nodes in both layers participate in the consensus reaching process. The first layer is a classic PBFT model that tolerates no more than $\lfloor \frac{m}{3} \rfloor$ faulty nodes based on the conclusion $Z >= 3f + 1$ [11]. In the second layer, as there are m PBFT consensus groups, we need to analyze the threshold of consensus-reached sub-groups required to ensure the security and Liveness of the whole system. During the consensus-reaching process, as the leader of each sub-group directly send $post - reply$ to the client, each consensus sub-group is regarded as a whole. While any individual node in PBFT systems can be divided into three categories, including consensus reached, not reached and faulty node. A consensus network may only be in two situations: consensus reached and not reached. In other words, a consensus sub-group would also be in two situations, either consensus reached or not. In this case, a system tolerates at most $\lfloor \frac{m}{2} \rfloor$ failed groups to reach consensus, i.e., the security threshold of consensus-reached sub-groups is $\lfloor \frac{m}{2} \rfloor$.

To facilitate distributed systems and blockchain in different environments, we analyze the success rate in two models under malicious attacks. The faulty probability determined (FPD) model is used when the probability of every single faulty node is fixed, and the faulty number determined (FND) model is used when the number of faulty nodes in the system is fixed. In these two models, we are given different initial conditions to analyze the security performance of the system. More specifically, we assume the faulty nodes in the FPD model are independent with each other, and they have the same faulty probability. Conversely, in the FND model, the probability of whether one node is faulty depends on other nodes because the total number of faulty nodes is fixed. In addition, these two models have different application scenarios. The FND model, which is more similar with the traditional PBFT, is suitable for small systems where the number of faulty nodes can be easily estimated. However, it is more appropriate to use FPD model to evaluate the performance of large systems where node failure is estimated by probability. For example, in manufacture, we are often given the reject rate of one product instead of the specific number. Finally, though FPD and FND models have many differences, the security performances of them approach the same as the system scaling up, which will be shown in Section 4.2.

TABLE 1
Frequently used notations

Notation	Definition
P_P	consensus success rate in FPD model
P_f	faulty probability of nodes
$P(A)$	probability of Event A
$P(B\|A)$	probability of Event B under the condition of Event A happening
P_N	consensus success rate in FND model
K	total number of faulty nodes
P_A	consensus success rate in advanced model
Z_X	total number of nodes in X-layer system
C_X	communication complexity of X-layer system
T_X	threshold to guarantee success in X-layer system

4.1.1 Faulty probability determined

Let's assume P_f is the faulty probability of each node. To find out the relationship between the success rate P_P and P_f, we shall first define two important conditions, under which consensus can be reached.

- no more than $\lfloor \frac{m}{3} \rfloor$ faulty nodes in the first layer (EVENT A)
- no more than $\lfloor \frac{m}{2} \rfloor$ groups fail in the second layer (EVENT B)

In addition, Event A and Event B are not independent. If one replica in the first layer is faulty, it will be impossible for the corresponding sub-group to reach consensus. Therefore, we have $P_P = P(A)\times P(B|A)$. Assume that there are i faulty nodes in the first layer and $0 \le i \le \lfloor \frac{m}{3} \rfloor$. According to the cumulative distribution function [31], we can get

$$P(A) = \sum\nolimits_{i=0}^{\lfloor \frac{m}{3} \rfloor} C_m^i \left(1 - P_f\right)^{(m-i)} P_f^i. \tag{4}$$

The value of $P(B|A)$ depends on the value of $P(A)$. Equation (4) indicates there are already i faulty nodes in the first layer. It means i out of m groups in the second layer share no chance to reach consensus as they have a faulty leader. Therefore, there can be at most $\lfloor \frac{m}{2} \rfloor$ failed groups in the second layer. We assume there are j groups, which do not have a faulty leader, fail to reach consensus. $0 \le j \le \lfloor \frac{m}{2} \rfloor - i$. We have

$$P(B|A) = \sum_{j=0}^{\lfloor \frac{m}{2} \rfloor - i} P_g^j \left(1 - P_g\right)^{(m-i-j)}. \tag{5}$$

P_g represents the probability of a group, with a non-faulty leader, failing to reach consensus. We assume there are g faulty nodes in one single group. To make this group fail, $\lfloor \frac{n}{3} \rfloor + 1 \le g \le n$ since PBFT group tolerates up to $\lfloor \frac{n}{3} \rfloor$ faulty nodes. Therefore, we have

$$P_g = \sum\nolimits_{g=\lfloor \frac{n}{3} \rfloor + 1}^{n} C_n^g P_f^g \left(1 - P_f\right)^{n-g}. \tag{6}$$

We can get the function of the system consensus success rate P_P against P_f as follows

$$P_P = \sum_{i=0}^{\lfloor \frac{m}{3} \rfloor} (C_m^i (1-P_f)^{(m-i)} P_f^i \sum_{j=0}^{\lfloor \frac{m}{2} \rfloor - i} P_g^j (1-P_g)^{(m-i-j)}). \tag{7}$$

To verify the closed-form expression derived, a simulation is performed based on random sampling in MATLAB. In the simulation process, we take the faulty probability P_f and node number m, n as the input and use a random array consisting $m+mn$ numbers to represent the status of nodes. Each number in this random array is either 1 or 0, representing faulty and non-faulty node respectively. On top of that, we set that each array element has a probability of P_f to be 1 (faulty), otherwise it is 0 (non-faulty). In this case, by counting the faulty nodes in each layer and sub-group and comparing the results with the thresholds, we can determine whether the consensus can be reached or not. Then, the above mentioned process is repeated over 10000 times and the simulation success rate for one value of P_f can be obtained by taking the ration of success times to the total repeating times. Finally, by increasing P_f from 0 to 1, we can easily get the simulation curve and compare it to the analytical result of the closed-form expression. The simulation design of the FND model is similar while the only difference is that FND takes the faulty node number instead of faulty probability as the input.

Note that, the purpose of the simulation is to examine the correctness of the derivations. Therefore, the complex peer-to-peer communication process is temporarily ignored in the simulation performed. However, we are also working on a system simulation, which takes every communication and view change into consideration, to further test the multi-layer PBFT system.

Fig. 4 shows clearly that the two curves match well, which verifies the effectiveness of the analytical result.

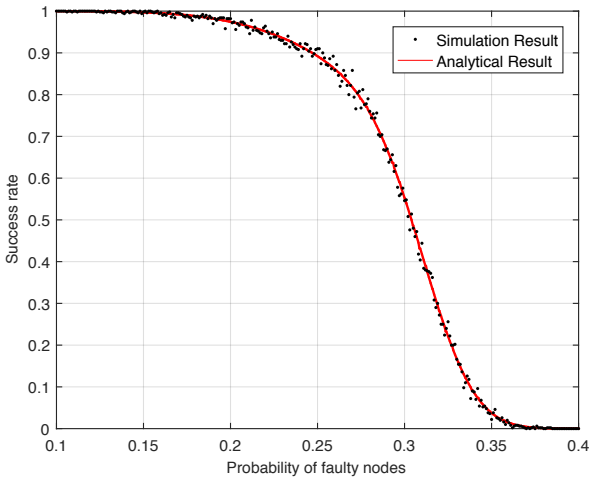

Fig. 4. Analytical and simulation results for success rate in FPD model. ($m = n = 30$)

4.1.2 *Faulty number determined*

In the FND model, we assume that there are K faulty nodes in the whole system and aim to find the relationship between K and the success rate P_N. Meanwhile, we can use the same assumption of event A and event B to calculate P_N, $P_N = P(A) \times P(B|A)$.

Unlike the FPD model, $P(A)$ and $P(B|A)$ are calculated using the hypergeometric model [32] since the FND model is based on the prerequisite of a fixed number of faulty nodes. Thus, we have

$$P(A) = \frac{1}{C_{m+mn}^K} \sum_{i=0}^{\lfloor \frac{m}{3} \rfloor} C_m^i C_{mn}^{K-i}. \tag{8}$$

Also, there should be at most $\lfloor \frac{m}{2} \rfloor - i$ failed sub-groups with non-faulty leaders, which means at most $\lfloor \frac{m}{2} \rfloor - i$ sub-groups have more than $\lfloor \frac{n}{3} \rfloor$ faulty nodes. However, in the FND model, the number of faulty nodes in each group affects situations in the other groups so that it will be extremely complicated to consider m groups together. Therefore, a simplified binomial distribution model on the group level is adopted, assuming every group has the same faulty probability of P_{g2}.

P_{g2} represents the probability of a group with a non-faulty leader failing in the second layer. It can be calculated as follows

$$P(B|A) \approx \sum_{j=0}^{\lfloor \frac{m}{2} \rfloor - i} P_{g2}^j (1-P_{g2})^{(m-i-j)}, \tag{9}$$

$$P_{g2} = \sum_{g=\lfloor \frac{n}{3} \rfloor + 1}^{n} \frac{C_n^g C_{mn-n-1}^{K-g}}{C_{m+mn-1}^K}. \tag{10}$$

Then we can get the probability P_N against K as

$$P_N = \frac{1}{C_{m+mn}^K} \sum_{i=0}^{\lfloor \frac{m}{3} \rfloor} (C_m^i C_{mn}^{K-i} \sum_{j=0}^{(\lfloor \frac{m}{2} \rfloor - i)} P_{g2}^j (1-P_{g2})^{(m-i-j)}). \tag{11}$$

In equation (11), $\sum_{i=0}^{\lfloor \frac{m}{3} \rfloor} C_m^i C_{mn}^{K-i}$ requires $K - i > 0$ since the combinatorics of a combination number must be positive. When $K - i > 0$, i.e., $K < \lfloor \frac{m}{3} \rfloor$, the success rate can be simply calculated as

$$P_N = \frac{1}{C_{m+mn}^K} \sum_{i=0}^{K} C_m^i C_{mn}^{K-i}. \tag{12}$$

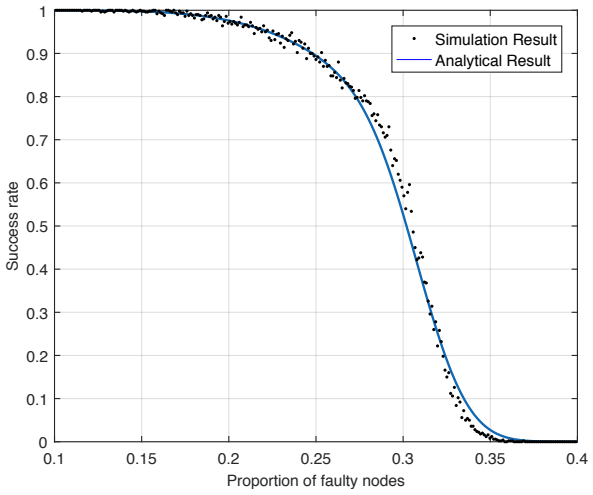

Fig. 5. Analytical and simulation curves for success rate in the FND model. ($m = n = 30$)

The curves in Fig. 5 show that, in the high success region where ($P_N > 0.85$), the analytical curve matches the simulation curve. When the success rate of the system is lower, there is a slight difference between our calculation and simulation results. The rationale behind this is that a part of the model (group level

in the second layer) is simplified from hypergeometric distribution to the binomial distribution by using P_{g2} as a failure rate for every group in equation (10). However, the error is negligible and will approach zero as the total number of the nodes in the system increases since the hypergeometric distribution approaches the binomial distribution in large systems [32].

4.2 Fault tolerance evaluation

Comparing the FPD model and the FND model with different network sizes, Fig. 6 shows that the curves of the two models gradually concur as m and n increase.

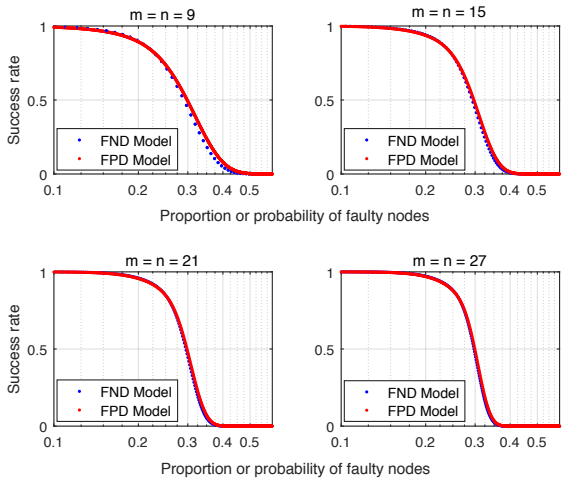

Fig. 6. FPD and FND models' analytical results with different m and n.

From Fig. 6, it can also be observed that as m and n get larger, the slope of the success rate curve approaches infinity around $x = \frac{1}{3}$ in both models. If we increase m and n to 500, the curve shows a more obvious trend to step around $P_f = \frac{1}{3}$, as in Fig. 7. The rationale behind this is that for a scaled-up system, faulty nodes are approximately evenly distributed. In this case, the proportion of faulty nodes in the whole system and that in each sub-group are very close. In other words, with the proportion of faulty nodes in the whole system reaching around $\frac{1}{3}$, the corresponding proportion within each consensus group also approximates $\frac{1}{3}$, which is the security threshold in original single-layer PBFT. Therefore, the success rate steps to zero when the proportion of faulty nodes or the faulty probability exceeds $\frac{1}{3}$. Therefore, we have the following proposition.

Proposition 5. *The system tolerates a larger proportion of faulty nodes when scaling up. The fault tolerance of two-layer PBFT converges to $\frac{1}{3}$ when Z goes to infinite.*

Fig. 6 only compares the situations where m and n are multiples of 3. We choose these special circumstances because when the m and n are multiples of 3, the system provides better security performance compared with others. This can be explained by considering the threshold of faulty nodes within each group. For example, two PBFT groups with $3f + 1$ and $3f + 3$ nodes can both tolerate f faulty nodes at most based on the conclusion in [11]. However, the ratio of maximum faulty nodes to the total nodes in the first group is bigger than that in the second one. Therefore, assigning m and n to be multiple of three makes the ratio reaches its maximum. Bringing these groups together, the whole system also shows better security performance. Moreover,

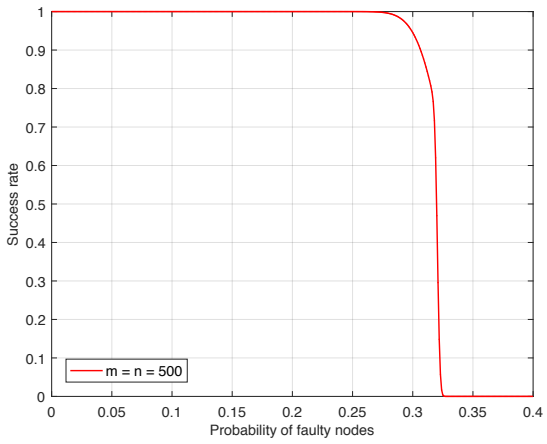

Fig. 7. Analytical result of FPD model with $m = n = 500$.

when m is a multiple of 6, the security performance is even better since we require at least half of the sub-groups to reach consensus. This means two systems with $m = 18$ and $m = 19$, for example, both tolerate 9 failed sub-groups at most. In this way, systems with an even integer m hold a larger ratio of faulty nodes. Based on these, we have the following Remark.

Remark. *By assigning m and n to be multiples of 6 and 3 respectively, the fault tolerance performance can be improved.*

4.3 Advanced system and security optimization

In this subsection, the advanced system is proposed as an ideal and ultimate situation where the nodes in the two layers are classified. We assume that nodes in the first layer are always non-faulty, while nodes in the second layer have a probability of P_f to be faulty. Note that, this is built under the assumption that, by implementing view change whenever vulnerability is detected in the first layer, after a long period, the nodes left in the first layer are the ones with higher reliability. These nodes show more stable performance and remain faithful for a certain period of time. The consensus success rate of the advanced system can be calculated as

$$P_A = \sum_{j=0}^{\lfloor \frac{m}{2} \rfloor} P_{g3}^{j} \left(1 - P_{g3}\right)^{(m-j)}, \tag{13}$$

$$P_{g3} = \sum_{g=\lfloor \frac{n}{3} \rfloor + 1}^{n} C_n^g P_f^g \left(1 - P_f\right)^{n-g}, \tag{14}$$

where P_f represents the faulty probability of second-layer nodes and P_A represents the consensus success rate of the advanced system. Comparing the analytical result with the simulation results in Fig. 8, we can see that the two curves concur. This proves the analytical result is valid.

In the practical systems, it is always worth to know what the determined security threshold is, i.e., to achieve 100% success rate, what is the maximum faulty nodes. Unlike the single-layer PBFT, the double-layer PBFT can be vulnerable if the faulty nodes are randomly distributed. This is because that the first layer is made up of one PBFT group consisting of a limited number of nodes, and the system has no chance to reach consensus if more than one-third of them are faulty nodes. In this case, as long as

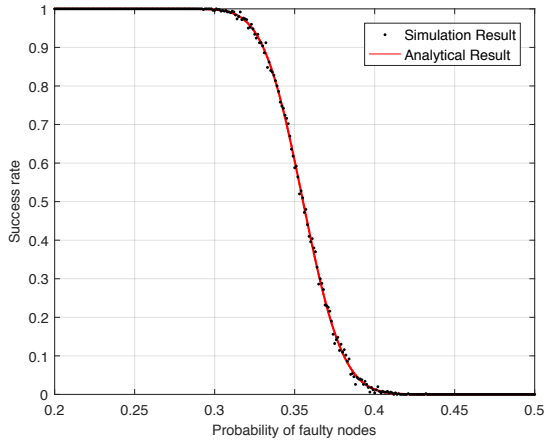

Fig. 8. Analytical and simulation curves for success rate in advanced system. ($m = n = 30$)

there are more $\lfloor \frac{m}{3} \rfloor$ faulty nodes in the system, some specific faulty node distributions will cause the system to fail. In other words, the first-layer nodes have more weights than the second for the final decision. However, as we mentioned in the introduction, this topology is very useful in decentralized systems to balance efficiency and security performance.

Fortunately, the determined security threshold can be improved over the operation time. In the advanced system where there is no faulty node in the first layer, the success rate stays at 100% until the number of faulty nodes increases to $\lfloor \frac{n}{3} \rfloor \times \lfloor \frac{m}{2} \rfloor$. Note that the intermediate states exist but not presented. Therefore, we have the following proposition.

Proposition 6. *To achieve 100% success rate, the maximum number of faulty nodes tolerated increases from $\lfloor \frac{m}{3} \rfloor$ to $\lfloor \frac{n}{3} \rfloor \times \lfloor \frac{m}{2} \rfloor$ if the advanced system can be achieved.*

However, this does not indicate that the consensus rate falls immediately to 0 when this threshold is exceeded. As shown in Fig. 8, the curve does not show a rapid decline until the probability of faulty nodes reaches around 0.3. To guarantees 100% success of consensus, number of faulty nodes can not be more than 1/6 of overall nodes. However, in practical deployments, especially with wireless communication uncertainty in large systems, it is very costly (and even impossible) to achieve a 100% success of consensus. Thus, fault tolerance of the double-layer system depends on the reliability requirement in different scenarios. For example, in 5G where the reliability requirement is lowered to 99.999% [33], the statistical fault tolerance is much higher than $\frac{1}{6}$ according to Fig. 8.

5 X-LAYER PBFT SYSTEM

5.1 Communication complexity analysis of X-layer system

X-layer PBFT system represents a general situation where nodes in a network are allocated to more than 2 layers. The consensus algorithm of i-th ($i \leq X$) layer is inserted between the commit and reply phase in $i-1$-th layer so that communication complexity can be further lowered. Suppose in an X-layer system, where every layer is full, the number of nodes in each subgroup in the i-th of X-layer is $m_i + 1$ ($m_0 = 1$). The total number of nodes Z_X in this X-layer system is

$$Z_X = 1 + \sum_{a=1}^{X} \left(\prod_{i=1}^{a} m_i \right). \tag{15}$$

The communication complexity C_X of this X-layer PBFT system is

$$C_X = \sum_{i=1}^{X} m_{i-2} m_{i-1} (m_i + 1)^2, \tag{16}$$

where m_{-1} is defined to be 1.

In this case, the minimum communication complexity of X-layer system can also be transformed into a typical optimization problem, which aims to solve the minimum value of equation (16) under the restriction of equation (15). The method is similar to *Proposition 9*, which is omitted here. Fig. 9 compares the communication complexity of systems with the same number of nodes but different network depth. It shows that lower communication complexity can be obtained by dividing a system into more layers.

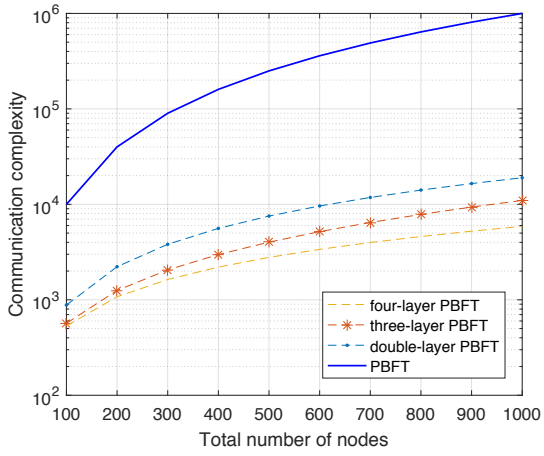

Fig. 9. Communication complexity of PBFT and multi-layer systems.

Therefore, we infer that the minimum communication complexity C_{Xmin} will be reached if a system is divided into maximum network depth X_{max}, i.e., by allocating the minimum number of 3 replicas into every sub-group. In other words, the Xth layer contains 3^X nodes in total. For given Z, X_{max} can be expressed as

$$X_{max} = \lfloor \log_3 (2Z + 1) \rfloor - 1. \tag{17}$$

To analyze the communication complexity of this limiting case, we suppose that Z is an integer that satisfies $Z = 1 + 3 + 3^2 + ... + 3^{X_{max}}$. This is to say, there are exactly 3 replicas in each sub-group in each layer. In this case, C_{Xmin} can be calculated as

$$C_{Xmin} = \sum_{i=1}^{X_{max}} 3^{i-1} (3 + 1)^2 \tag{18}$$

$$= 16 \times \frac{1 - 3^{X_{max}}}{1 - 3} \tag{19}$$

$$= \frac{16Z - 16}{3}. \tag{20}$$

So far, we have analyzed the communication complexity of multi-layer PBFT when the network depth is maximized. During this

process, we make an assumption about the number of replicas in each sub-group to explore the what the communication complexity would approach when X continues to increase. The analytical result shows the communication complexity of the X_{max}-layer system is lowered to $\frac{16Z-16}{3}$, a linear relationship with the node number Z. Therefore, we have the following proposition.

Proposition 7. *The communication complexity is further lowered by increasing network depth and is in the range of* $1.9Z^{\frac{4}{3}} \geq C \geq \frac{16Z-16}{3}$. *In other words, multi-layer PBFT system provides better node scalability compared with the original PBFT.*

However, It is worth mentioning that other performances such as security and latency serve as a trade-off when reducing the communication complexity. Detailed analysis is provide in Section 5.2 and Section 5.3 as follows.

5.2 Security analysis of X-layer system

Like the double-layer system, the security performance of the X-layer PBFT system can also be optimized over operation time by a proper protocol. In this case, the threshold T_X to maintain the success rate at 100% is

$$T_X = \lfloor \frac{m_X}{3} \rfloor \times \lfloor \frac{\prod_{i=1}^{X-1} m_i}{2} \rfloor. \tag{21}$$

From equation (21), it can be seen that $T_X < \frac{1}{6}$ and it drops as X increases. In other words, by increasing the network depth, the security performance is weakened while the communication complexity is improved. This is because increasing layer number leads to fewer node number in each layer, which makes the system more vulnerable to randomly distributed faulty nodes. In the practical scenarios, this trade-off should be considered when designing multi-layer PBFT systems.

5.3 Latency analysis

Another trade-off when reducing the communication complexity is latency, which will be analyzed in this subsection. In the proposed multi-layer PBFT, we construct a hierarchical structure to limit the peer-to-peer communication within each group or layer. Meanwhile, it is inevitable that the system confirmation delay is prolonged since the consensus reaching process is carried out in each layer successively.

In fact, the confirmation delay would keep increase with increasing network depth X. Assuming that each layer takes average t_{avg} seconds to reach consensus and propagate to the next layer, the average propagation delay t_{apd} holds a linear relationship with the network depth X, i.e., $t_{apd} = Xt_{avg}$. Fortunately, if the protocol is used in the Internet, we can use parallel routes and distribute different information through different paths. In this way, though the consensus-reaching process still goes through different layers, the delay within groups is reduced, contributing to a lowered overall latency [34].

Conclusively, compared with original PBFT, the multi-layer PBFT sacrifices certain system delay while providing low complexity. Even though, compared with other consensus algorithms such as PoW that has good scalability, the latency of multi-layer PBFT is significantly lower. Therefore, the proposed system can be regarded as a trade-off between system delay and scalability of the existing protocols. Table 2 is provided to compare the performances of different protocols and the proposed multi-layer PBFT. As such, Considering the different demands of practical scenarios, different consensus protocols could be adopted to provide optimal performance accordingly.

6 THE PROTOCOL

In this section, we propose a practical protocol dedicated to the double-layer PBFT, where replicas in the first layer are denoted as r_i^1 (superscript 1 for layer number, subscript i for replica index). r_i^1 act as leaders for corresponding second-layer replicas (r_i^2). One leading r_i^1 and its corresponding r_i^2s, all together, form a consensus group, resulting in a tree-like topology structure.

When each group reaches consensus, group members reply to their leader instead of the client. Then the leader collects replies and sends it to the client on behalf of that consensus group. The client accepts the results only agreed by more than half consensus groups. The protocol overview is illustrated in Fig. 10. Meanwhile, there is a group configuration GP that describes the allocation group members and their leader. It will be updated when the network structure is changed. In brief, the new protocol inserts successive $pre-prepare$, $prepare$, and $commit$ phases before starting the $commit$ phase in the upper layer.

6.1 Consensus flow

6.1.1 The client

A client c sends a request message $[o, t, c]_{request}$ to primary. This request invokes an operation o with timestamp t. Timestamps are ordered by time, so the stamp of later operation contains higher values. The request is sent to the replica. The identity of the replica is extracted from the view number contained in replies from previous operations. On receiving the request, primary multicasts messages using the protocol stated below.

All group leaders reply results to the client directly. The $poset-reply$ has the form $[o, t, c, i, r, rc^1, \nu, GP]_{poset-reply}$ where ν is the current view number, i is the replica number, r is the result of the execution, rc is the reply certificate and GP describes the replicas allocation.

Assuming there are a number of m replicas in the first layer and n in the second, and the number of faulty replicas in a consensus group is f^g (to distinguish from f). If leaders have received $f^g + 1$ matching valid replies from the same consensus group, this group is said to have reached consensus. The network reaches consensus when more than half of the groups have the same replies. The client only accepts results replied from group leaders when at least half of them are consistent.

6.1.2 First-layer protocol

In the first layer, the primary and m replicas form a consensus group. When primary p receives a request $M = [o, t, c]_{request}$ from client, it authenticates the request and client's identity. Then primary assigns a sequence number α to M. After that, the primary steps into $pre-prepare$ phase by multicasting $[M, d, \alpha, \nu]_{pre-prepare^1}$ where d is the digest of M (superscript is to distinguish $pre-prepare^1$ in the first layer from $pre-prepare^2$ in the second layer). Primary multicasts messages only among r^1. Thus, only r^1 reacts on $pre-prepare^1$. The propagation is restricted since the protocol runs layer by layer.

For $pre-prepare$, $prepare$ and $commit$ messages, a replica r_i^1 accepts the one with the same view ν; the authenticity is then verified; α is between watermark h and H. The watermark is introduced to ensure a weak synchronization and defined in Section 6.4.

With conditions above, a replica i in the first layer accepts a $pre-prepare^1$ message from primary only when there is none different request with the same view ν and sequence number α is accepted. Then it multicasts $[d, \alpha, i, \nu]_{prepare^1}$ messages to all r_i^1 in the first layer. It records both $pre-prepare^1$ and $prepare^1$

TABLE 2
Performance comparisons of the proposed and state of the art consensuses

	Byzantine fault tolerance	Security	Latency	Communication complexity	Scalability
Original PBFT [11]	Yes	$\frac{1}{3}$	Low	$O(Z^2)$	Low
RAFT [8]	No	$\frac{1}{2}$	Low	$O(Z)$	High
Hotstuff [30]	Yes	$\frac{1}{3}$	Medium	$O(Z)$	High
Double-layer PBFT (proposed)	Yes	equation (7) equation (11)	Medium	$1.9Z^{\frac{4}{3}}$	Medium
Multi-layer PBFT (proposed)	Yes	equation (21)	High (Increases with network depth)	$1.9Z^{\frac{4}{3}} \geq C \geq \frac{16Z-16}{3}$ (decreases with network depth)	High

messages to its log. During the $prepare$ phase, each r_i^1 replica collects $2f$ messages with matching sequence number α, view ν, and request M. With the received $pre-prepare^1$ messages, they form $prepared-certificate^1$, which indicates a particular r_i^1 replica has prepared the request.

For prepared r_i^1, it multicasts $[d, \alpha, i, \nu]_{commit^1}$ and waits for more than $2f+1$ matching $commit^1$ messages with the same view, sequence and digest from different r^1 replicas. Received messages form $commit-certificate^1$ (cc^1) and this request is said to be committed on replica r_i^1. Then the replica pauses the execution and initiates another round of protocol in the second layer as described in Section 6.1.3.

The committed replicas send $[o, t, i, r, \nu]_{reply^1}$ to their group leader, i.e., the primary in the first layer. The primary confirms that this group has reached consensus by checking more than half of group members reply consistent $reply^1$ messages, including itself. The group leader collects $reply^1$ and forms a reply-certificate rc^1. After that, this primary replies to the client with $[o, t, c, i, r, rc^1, \nu, GP]_{poset-reply}$. Notice that it is not necessary for primary to reply to the client on behalf of consensus, but we require primary to do so to keep the algorithm the same on all replicas in case a massive deployment.

6.1.3 Second-layer protocol

A committed r_i^1 multicasts new $pre-prepare$ message to r^2 within the same consensus group, where another round of PBFT protocol is implemented. All group members reply to the leader in a similar manner in Section 6.1.2 when the request is committed again. For a replica r_p^1 which acts as primary, it multicasts a similar $[M, d, \alpha, \nu, cc^1]_{pre-prepare^2}$ to r_i^2 replicas in same consensus group, where cc^1 is the $commit-certificate^1$. The ν, α, and M are inherited from the previous process. A replica r_i^2 in consensus group will accept the request if the condition mentioned in the first-layer protocol is satisfied, in addition to the presence of cc^1.

On receiving valid $pre-prepare^2$ message, the pre-prepared r_i^2 multicasts $[d, \alpha, i, \nu]_{prepare^2}$ messages to all r_i^2 in same consensus group. It adds both $pre-prepare^2$ and $prepare^2$ messages to its log. In the $prepare^2$ phase, each r^2 replica collects $2f$ messages with matching sequence number α, view ν, and request M. With received $pre-prepare^2$ message, it forms a quorum prepared $certificate^2$, which indicates that this r_i^2 replica has prepared the request.

Then the prepared replicas r_i^2 and their r_i^1 leader multicast $[d, \alpha, i, \nu]_{commit^2}$ and collect $2f+1$ matching $commit^2$ messages with the same view, sequence and digest form different r_i^2 replicas. These $commit^2$ form $commit-certificate^2$, and this request is said to be committed. Replica then executes the message which has been committed. After the execution, all group members reply to the result to their group leader, and the leader replies to the client in a similar manner in Section 6.1.2. The pseudocode for protocol are described in algorithms 1, 2 and 3.

Algorithm 1 Primary Normal-case Pseudocode

```
while valid request^1 received=True do
  if client identity authenticated=True then
    m←n.
    multicasts pre−prepare^1 to r^1.
  end if
end while
while valid prepare^1 received=True do
  if number of valid prepare^1 > 2f then
    forms prepared−certificate^1.
    multicasts commit^1 to r^1.
  end if
end while
while valid commit^1 received=True do
  if number of valid commit^1 > 2f then
    forms commit−certificate^1.
  end if
end while
while valid reply^1 received=True do
  if number of valid reply^1 >half of members then
    forms rc^1.
    reply client with post−reply^1.
  end if
end while
```

6.2 Faulty primary elimination

6.2.1 Faulty primary detection

The most commonly applied condition for initiating a view-change is by detecting whether the primary is responding, i.e., the replicas keep a timer which will be reset each time a new request is received. However, a faulty primary that assigns different $pre-prepare$ to different replicas will not trigger time-out. Thus, we present a possible mechanism without a timer to detect faulty primary nodes that multicast random messages during $prepare$ phase. Since one replica may skip several operations when the

Algorithm 2 r_i^1 Normal-case Pseudocode

while valid $pre-prepare^1$ received=True **do**
  multicasts $prepare^1$ to p r^1.
end while
while valid $prepare^1$ received=True **do**
  if number of valid $prepare^1 > 2f$ **then**
    forms $prepared-certificate^1$.
    multicasts $commit^1$ to r^1.
  end if
end while
while valid $commit^1$ received=True **do**
  if number of valid $commit^1 > 2f$ **then**
    forms $commit-certificate^1$.
    multicasts $pre-prepare^2$ to subordinate r^2.
  end if
end while
while valid $prepare^2$ received=True **do**
  if number of valid $prepare^2 > 2f$ **then**
    forms $prepared-certificate^2$.
    multicasts $commit^2$ to r^2.
  end if
end while
while valid $commit^2$ received=True **do**
  if number of valid $commit^2 > 2f$ **then**
    forms $commit-certificate^2$.
    reply primary with $reply^1$.
  end if
end while
while valid $reply^2$ received=True **do**
  if number of valid $reply^2$ >half of members **then**
    forms rc^2.
    reply client with $post-reply^2$.
  end if
end while

Algorithm 3 r_i^2 Normal-case Pseudocode

while valid $pre-prepare^2$ received=True **do**
  multicasts $prepare^2$ to r^1 r^2 in same consensus group.
end while
while valid $prepare^2$ received=True **do**
  if number of valid $prepare^2 > 2f$ **then**
    forms $prepared-certificate^2$.
    multicasts $commit^2$ to r^2 in same consensus group.
  end if
end while
while valid $commit^2$ received=True **do**
  if number of valid $commit^2 > 2f$ **then**
    Forms a quorum $commit-certificate^2$.
    Send $reply^2$ to group leader.
  end if
end while

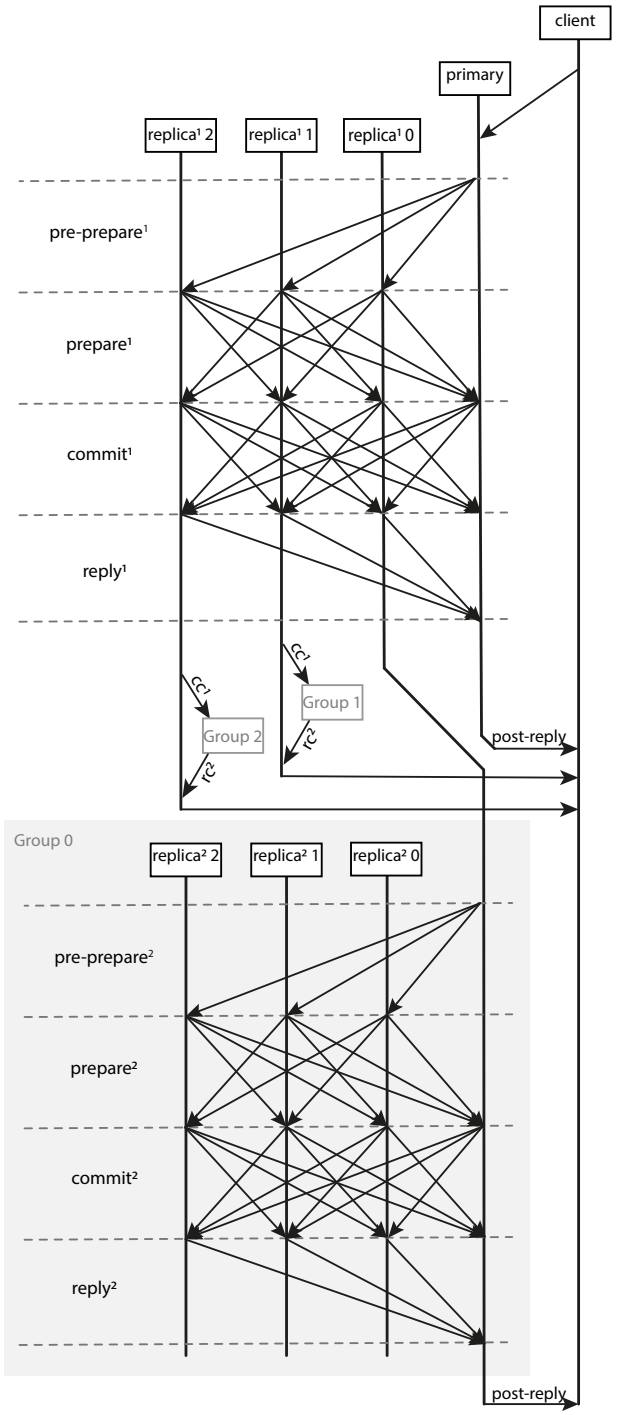

Fig. 10. Implementation flow chart for double-layer PBFT model.

connection is lost, we want the detection to be independent without the prerequisite of total order and continuous sequence number α, i.e., replicas accept discrete α as long as it is consistently increasing/decreasing. To facilitate understanding of the mechanism, we first assume that the primary is not faulty. Since there are at most f faulty replicas, the collected $2f+1$ messages contain f faulty messages in the worst situation. And the rest $f+1$ messages must be obtained from non-faulty replicas; thus, those $f+1$ messages are identical. Those faulty messages (or the digest of the messages) could be unmatched with each other. Or, the faulty messages are identical, but those faulty messages are different from those

matching messages among non-faulty nodes. For simplicity, we say that in the latter case, there are two versions of messages in $2f + 1$ collections, one kept among non-faulty replicas, and another version is kept among faulty replicas.

When primary is non-faulty and each replica collects $2f + 1$ messages, the number of different versions n_v falls in the interval 2 and $f + 1$. That is because, in the worst case mentioned above, each faulty replica multicasts different versions of messages, resulting in f different versions among faulty replicas. Besides, there is an additional version kept by $f + 1$ non-faulty replicas. At the presence of faulty primary that multicasts random messages, the messages in $pre-prepare$ that received by non-faulty replicas differ. In other words, n_v exceeds the upper bound $f + 1$, and the primary is detected to be faulty. This condition, along with time out, triggers the $view - change$ phase.

6.2.2 View change

In conventional PBFT [11], replicas invoke view change in prepare phase. Changes are made to adapt to our multi-layer model. As shown in Fig. 1, a replica is the primary of its sub-layer replicas. Thus, in this protocol, replicas in a specific layer detect their faulty primary in the upper layer and invoke cross-layer view change. Each replica, which suspects the primary to be faulty, multicasts a $view - change$ messages with the stable checkpoint to new primary (the new primary may be determined by election mechanism based on current view number). The new primary decides whether to lunch $new - view$.

Suppose a replica is in layer L as a member for group K, and notice the group leader is the group member of group in layer $L - 1$. If it suspects the group leader is faulty for not responding or deliver $pre - prepare$ messages with invalid sequence number, it stops accepting requests and starts view change that moves the view of this group from ν into $\nu + 1$ by multicasting $[\nu + 1, \alpha, C, P, i]_{view-change}$, where α is the sequence number for the latest checkpoint for this replica and C is the $2f$ matching certificate for this checkpoint. P is the collection of $prepare - certificate$ ($2f + 1$ matching prepare requests) for each $pre - prepare$ request that higher than α. i is the identification of the sender.

If the new primary p in new view $\nu + 1$ has received $2f - 1$ valid $view - change$ messages from other group members. It multicasts $[\nu + 1, GP, \gamma, O]_{new-view}$, where the γ is a set of $2f$ matching $viewChange - certificates$, and O is the set of $pre - prepare$ messages than need to be multicasted for they are failed to reach consensus in last view ν. The sequence of $pre - prepare$ in O is ranging from the latest checkpoint know to the new leader and the latest α in P. GP is the update of itself. describes how p allocate its group member in layer $L - 1$ to the rest members in group K. Member replicas accept and execute a valid new view. Those mentioned above are similar to original view-change protocol.

However, taking over the leader of group K in layer L implies that this replica becomes the member of a group in layer $L - 1$ (for instance, group J). Thus, the primary p multicasts $[join, \nu, \gamma, i]_{join}$ to group J after the $view - change$. The ν is view number for group J since it should remain unchanged if there is no view change in J. p extracts view in J from the $commit - certificate$ that passed by the original group leader from group K. Matching ν and γ prove the validity. Then this replica directly commits the current operation since it must have been committed in layer $L - 1$, otherwise the consensus protocol will not be executed in layer L. This $join$ message informs the member in group J that there is a change of replica and group J update local GP.

Also, a member in layer L is the leader in layer $L + 1$. To reallocate its former members (since p is no longer their leader), it multicasts $[v+1, G, \gamma, i, r]_{redistribute}$ to replica r where G is the new designated group in layer $L - 1$ for replica r. The replica r then redirects itself to group G and update local GP. The replicas governed by group K in layer deeper than $L - 1$ are reaching consensus in seam sequence since they are (indirectly) lead by group K. The effect of view change is limited.

To reduce the extra communication complexity and latency introduced by this modified protocol, the $redistribute$ messages are embedded into GP in $new - view$ message. Then it is extracted by current members of group K and pass to their former group in layer $L + 1$ by piggybacking in $pre - prepare$. Those replicas in layer $L + 1$ redirect themselves before responding to new $pre - prepare$. Thus, only the join message attributes to extra communication expenditure which, in the two layer case, the size m of layer 1.

To facilitate the assessment of view change complexity of our double-layer scenario, suppose that the number of replicas in the first layer is m, and for each group in the second layer is n (as depicted in Fig. 1). Note that the view change can be triggered from either the first layer or the second layer. First, we consider the case that the view change is triggered in the first layer. As described in the protocol, the first part of our view change process runs within this layer (which is similar to original PBFT). For this part, the complexity is $O(m^3)$. As one of the members in the first layer is selected as the new primary, the extra complexity is introduced by multicasting the $redistribute$ messages to its group in the second layer. Since the certificate is generated in the first layer, the complexity of this part is $O(m^2 n)$. The total complexity is $O(m^3 + m^2 n)$.

Then we consider that a view change is triggered in one of the groups in the second layer. Similarly, the complexity is $O(n^3)$ for the first part. The extra complexity is introduced by multicasting the $join$ messages to the first layer. However, the certificate is now generated in one of the groups in the second layer. For this reason, the additional complexity is $O(n^2 m)$. Then the total complexity is $O(n^3 + n^2 m)$.

For comparison, the complexities are $O((m + mn)^3)$ for the original PBFT and $O(m + mn)$ for HotStuff when considering the same total number of replicas. It can be seen that in both first layer or second layer triggered cases, the complexity of proposed double layer PBFT is not as good as HotStuff, however, it reduces the complexity in large scale when compared with original PBFT.

6.3 Operation synchronization

The precondition for entering the next phase does not require synchronization across all replicas. Due to the loss of connection or other reasons, one replica may skip some operations. Though the consensus is still reached, the sequence number α is not necessarily continuous.

But for clients who also access data from the network, operations are preferably synced on each replica. The asynchronous replica can be detected by the discrete sequence number and the reached watermark. A replica may extract the missing operations by inquiring them from other replicas in real-time or at a constant interval.

Once the replica decides to extract a missing operation from the rest of the network, it multicasts $[n\alpha, i, \nu]_{extract-request}$. If there are no fewer than $2f + 1$ valid $extract - reply$, the replica accepts the reply as its missing operation. For replica who

received $extract-request$, it replies $[O, \alpha, i, \nu]_{extract-reply}$ if the request is valid.

6.4 Garbage collection

The data recorded on replicas increase its size as the protocol run. To discard unnecessary operations that have already reached a consensus, we implement Castro's [11] garbage collection.

Replicas multicast $checkpoint$ messages with the sequence number of latest committed operations. Sequence number with $f+1$ $checkpoint$ (including its own) is seated as low watermark h. To prevent a replica who encounters with the transmission delay from going too far. A logic size L is set that the replica only executes operation between h and $H = L + h$.

6.5 Safety and liveness

The group-wide operation is inconsistence with the original PBFT. The network is weakly synchronised that the time t for a messaged been received after sending does not go infinity. The byzantine replica is assumed unable to subvert cryptography. The safety and liveness are retained within/across groups with modified view change protocol. Suppose there are no more than $\frac{1}{3}$ of the nodes in a group are faulty, and the network is weakly synchronised that the time t for a messaged been received after sending does not go infinity.

Since the protocol requires more than $\frac{2}{3}$ replicas to communities before advanced into next operation (to ensure the number of responses from non-faulty replicas is always greater than that from faulty), there is at least one non-faulty replica overlap for two consecutive operations. All non-faulty replicas agree on each other, then they agree on total order of operations, providing safety. Also, the bound of faulty replicas indicates the protocol always collects sufficient responses to proceed for liveness. The consensus in upper layer is the precondition to invoke protocol in sub-groups. Thus, safety is guaranteed across groups. The modified view change protocol replaces faulty group leader. Thus, liveness is guaranteed across groups. This consistency with PBFT in safety and liveness within and across groups leads to consistency for the whole network.

7 CONCLUSION

A scalable multi-layer PBFT mechanism is proposed to reduce the communication complexity of the original single-layered PBFT. This paper proves that the communication complexity of the proposed double-layer PBFT system is significantly reduced to a minimum of $C \approx 1.9Z^{\frac{4}{3}}$ at system's maximum optimized capacity. To reach the minimum communication complexity, the optimal values of m and n are proposed in Section 3.3. Moreover, the analytical results of the security threshold show that the success rate sinks significantly when the proportion of faulty nodes exceeds $\frac{1}{3}$ of the total. Also, the threshold which keeps success rate at 100% rises from $\lfloor \frac{m}{3} \rfloor$ to $\lfloor \frac{n}{3} \rfloor \times \lfloor \frac{m}{2} \rfloor$ in advanced model. These results show that the security performance of the double-layer system is largely determined by the first layer and is improved over operation time. Latency performance is also a trade-off. The confirmation delay increases with increasing network depth. Finally, we expand the double-layer system to the multi-layer. We have compared security performance and communication complexity between double-layer and multi-layer systems. Results show that communication complexity can be further lowered to a minimum of $\frac{16Z-16}{3}$ if the network depth is maximized to X_{max} at the expense of certain security performance degradation.

This paper provides guidance for multi-layer PBFT system design and performance analysis, which would serve as a foundation for future research. Meanwhile, there are also limitations to be improved. For example, a system model to differentiate nodes in the first and second layer could be proposed as a trade-off between FND/FPD and the advanced model. Also, experimental evaluation of *Proposition 7* would give a further insight into the performance tradeoffs to help with the system design in different application scenarios. Another potential topic is the deployment of multi-layer PBFT system. It is worth to mention that some scenarios such as in financial, are sensitive to both latency and scalability, thus more advanced research should be conducted to solve the issue.

APPENDIX A
PROOF OF PROPOSITIONS

Proposition 8. *For a double-layer PBFT system with m replicas in the first layer and n sub-layer replicas in each sub-group, the communication complexity C to reach consensus is*

$$C = (m+1)^2 + m(n+1)^2. \tag{22}$$

Proof. In the consensus reaching process, every node in the double-layer PBFT communicates within its group. As a result, this system can be regarded as $m+1$ separated PBFT groups. Among them, m groups have $n+1$ nodes and 1 group has $m+1$ nodes. Based on the conclusion that $O(Z^2)$ times of communications are needed for original single-layer PBFT with Z nodes, the communication complexity of double-layer PBFT is $(m+1)^2 + m(n+1)^2$. □

Proposition 9. *For a double-layer system containing Z nodes in total, the minimum communication complexity can be achieved when n equals to the nearest integer to the real positive root of following equation:*

$$n^3 + 3n^2 + n = 2Z - 1, \ \ (3 \le n \le \frac{Z-4}{3}). \tag{23}$$

Proof. The relationship among Z, n and m is

$$Z = 1 + m + mn. \tag{24}$$

Let $Y = Z - 1$, we have

$$m = \frac{Y}{1+n}, \ \ (n \ge 3, m \ge 3). \tag{25}$$

$n \ge 3, m \ge 3$ because a PBFT group requires a minimum of 4 nodes. The communication complexity C can be expressed as

$$C = (\frac{Y}{1+n} + 1)^2 + Y(1+n). \tag{26}$$

As Y is a constant, C is a function of n. Also, The second-order derivative of equation (26) equals to $\frac{6Y^2}{(1+n)^4} + \frac{4Y}{(1+n)^3}$, which is always positive. The first-order derivative has a zero point. Therefore, the value of C decreases when n is small and increases afterwards. The minimum value of C can be reached at that zero point of the first-order derivative.

Simplifying the equations, the optimal solution of n to obtain the minimum communication complexity can be calculated by equation (23). It can be proven that equation (23) has three real roots, one of which is positive. Moreover, a solution which satisfies $3 \le n \le \frac{Y}{3} - 1$ exists when $Z \ge 29$. Therefore, the system has its minimum communication complexity if n is the nearest integer to the positive root of this cubic equation when $Z \ge 29$. Otherwise, $n = 3$ is the solution. □

Proposition 10. *When m and n are fixed by optimal allocation and the system is full, the relationship between communication complexity C and total node number Z can be written as*

$$C \approx 1.9Z^{\frac{4}{3}}. \tag{27}$$

Proof. When a system is optimally setup, both C and Z can be expressed by polynomials containing m and n. However, it is difficult to find out the relationship between C and Z directly. To solve this problem, intuitively, we first calculate the ratio of C to Z. Divide equation (22) by equation (24)

$$\frac{C}{Z} = \frac{(m+1)^2 + m(n+1)^2}{1+m+mn} \tag{28}$$

$$\approx \frac{mn^2 + m^2 + 2mn + 3m}{m+mn} \tag{29}$$

$$= \frac{n^2 + m + 2n + 3}{1+n} \tag{30}$$

$$\approx n + 2 + \frac{m}{n}, \tag{31}$$

where m and n are allocated according to *Proposition 9*. To simplify this ratio, we use equation (24) and equation (23) to substitute $\frac{m}{n}$ by an algebraic expression of n only. Therefore, $\frac{C}{Z}$ can be expressed as

$$\frac{C}{Z} \approx \frac{3n}{2} + 2. \tag{32}$$

In this calculating process, some constant terms are ignored. All of the approximations hold when $m \gg 3$ and $n \gg 3$.

So far, we know that Z is a cubic function of n and $C \approx (\frac{3n}{2} + 2)Z$, so the function of C vs Z can then be simplified as $C = kZ^{\frac{4}{3}}$, where k is a constant. By estimating k according to equation (23), the final relationship between C and Z can be written as equation (27). □

Proposition 11. *If the double-layer system is not full, the communication complexity reaches a smaller value when the vacancies are equally distributed into the sub-groups. The minimum value can be reached by distributing vacancies to the minimum number of sub-groups.*

Proof. Once n is determined according to *Proposition 9*, m is then solvable by equation (25). Possibly the result is not an integer. In this case, we round m up to the nearest integer and suppose that

$$Y + r = m(1+n), \tag{33}$$

where r is the smallest positive integer to satisfy equation (33). Note that, r also represents the vacancies to be distributed in the system which is not full. In this way, since n is determined, the problem of how to allocate the nodes into the sub-groups is transformed into how to allocate these vacancies in the second layer. To facilitate the subsequent derivation, we need to compare the value of r, $n+1$ and m first. Since $r = \lceil \frac{Y}{1+n} \rceil \times (1+n) - Y$, $r < 1 + n$. Also, by calculating the optimal solutions for m and n, we have $m > n + 1$ for large systems, i.e., systems with more than 100 nodes. Therefore, $r < n + 1 < m$.

To begin with, we calculate the communication complexity when the vacancies are equally distributed. Suppose that r vacancies are assigned into r_1 groups evenly ($r = kr_1$, $k \in \mathbb{I}$). Then $m - r_1$ groups with $n + 1$ nodes are generated, so are r_1 groups with $n + 1 - \frac{r}{r_1}$ nodes in the second layer. Hence, the communication complexity C_1 will be

$$C_1 = (m+1)^2 + (m - r_1)(n+1)^2 + r_1(n + 1 - \frac{r}{r_1})^2. \tag{34}$$

The difference between C_1 and C in equation (22) is

$$C - C_1 = r_1(n+1)^2 - r_1(n + 1 - \frac{r}{r_1})^2 \tag{35}$$

$$= 2(n+1)r - \frac{r^2}{r_1}. \tag{36}$$

From equations above, the difference between C and C_1 increases as r_1 increases. In order to obtain the minimum value of C_1, r_1 should be as small as possible under the condition of $n+1-\frac{r}{r_1} > 3$.

However, if the vacancies are not equally distributed. For example, when one of the r_1 groups contains $\frac{r}{r_1} + b$ ($b \in \mathbb{I}$) vacancies and another group contains $\frac{r}{r_1} - b$ vacancies. In this case, the total communication complexity of these two groups will be $(n + 1 - \frac{r}{r_1} - b)^2 + (n + 1 - \frac{r}{r_1} + b)^2$ rather than $2(n + 1 - \frac{r}{r_1})^2$ where vacancies are equally allocated. By calculating the difference, the communication complexity result is increased by $2b^2$ compared with the situation where vacancies are evenly distributed into sub-groups. This result also applies when there are more than two groups with a different number of vacancies. Therefore, assigning the vacancies into the sub-groups equally gains lower communication complexity. □

REFERENCES

[1] M. H. Miraz and M. Ali, "Applications of blockchain technology beyond cryptocurrency," *Annals of Emerging Technologies in Computing*, vol. 2, no. 1, 2018.

[2] S. Nakamoto, "White paper: Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[3] P. Vasin, "White paper: Blackcoin's proof-of-stake protocol v2," 2014. [Online]. Available: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf

[4] C. Bendiksen, S. Gibbons, and E. Lim, "The bitcoin mining network-trends, marginal creation cost, electricity consumption and sources," *CoinShares Research*, vol. 21, 2018.

[5] B. Cao, Y. Li, L. Zhang, L. Zhang, S. Mumtaz, Z. Zhou, and M. Peng, "When Internet of Things meets blockchain: Challenges in distributed consensus," *IEEE Network*, vol. 33, no. 6, pp. 133–139, Nov 2019.

[6] J. Benet, "IPFS - content addressed, versioned, p2p file system," *arXiv:1407.3561*, jul 2014.

[7] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.

[8] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," *USENIX*, pp. 305–320, 01 2014.

[9] H. Xu, L. Zhang, Y. Liu, and B. Cao, "RAFT based wireless blockchain networks in the presence of malicious jamming," *IEEE Wireless Communications Letters*, 2020.

[10] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*. Springer, 2015, pp. 112–125.

[11] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," *OSDI 1999: Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, pp. 173–186, 1999.

[12] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[13] O. Onireti, L. Zhang, and M. Imran, "On the viable area of wireless practical byzantine fault tolerance blockchain networks," *IEEE GLOBECOM*, 2019.

[14] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference*, pp. 30:1–30:15, 2018.

[15] J. Morgan, "White paper: Blockchain and the decentralization revolution." [Online]. Available: https://www.jpmorgan.com/jpmpdf/1320745566550.pdf

[16] H. Sukhwani, J. M. Martinez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network," *2017 IEEE 36th Symposium on Reliable Distributed Systems*, pp. 253–255, 2017.

[17] X. Fan, "Scalable practical byzantine fault tolerance with short-lived signature schemes," *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, pp. 245–256, 2018.

[18] M. Zamani, M. Movahedi, and M. Raykova, "White paper: RapidChain: Scaling blockchain via full sharding dfinity palo alto, ca," 2018. [Online]. Available: https://eprint.iacr.org/2018/460.pdf

[19] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, 2016.

[20] P. J. Sadalage and M. Fowler, *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2013.

[21] W. Lv, X. Zhou, and Z. Yuan, "Design of tree topology based byzantine fault tolerance system," *Journal of Communications*, vol. 38, no. Z2, pp. 143–150, 2017.

[22] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2017, pp. 173–178.

[23] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[24] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran, "Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5791–5802, 2019.

[25] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[26] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild (keynote talk)," in *31st International Symposium on Distributed Computing (DISC 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 91. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 1:1–1:16.

[27] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 363–376.

[28] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[29] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with BFT-SMART," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.

[30] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[31] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 2008, vol. 2.

[32] J. A. Rice, *Mathematical statistics and data analysis*. Cengage Learning, 2006.

[33] B. Chang, L. Zhang, L. Li, G. Zhao, and Z. Chen, "Optimizing resource allocation in URLLC for real-time wireless control systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8916–8927, 2019.

[34] G. Voron and V. Gramoli, "Dispel: byzantine SMR with distributed pipelining," *arXiv preprint arXiv:1912.10367*, 2019.