applied sciences

MDPI

Article

PnV: An Efficient Parallel Consensus Protocol Integrating Proof and Voting

Han Wang [1,2], Hui Li [1,2,*], Ping Fan [3], Jian Kang [3], Selwyn Deng [3] and Xiang Zhu [3]

[1] School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen 518055, China; wanghan2017@pku.edu.cn
[2] Pengcheng Laboratory, Shenzhen 518000, China
[3] China United Network Communication Group Co., Ltd., Beijing 100140, China; pingfan@chinaunicom.cn (P.F.); 18679401189@wo.com.cn (J.K.); ho-selwyndeng@chinaunicom.cn (S.D.); zhuxiang@chinaunicom.cn (X.Z.)
* Correspondence: lih64@pkusz.edu.cn

Abstract: Consensus protocols, as crucial components of blockchain technology, play a vital role in ensuring data consistency among distributed nodes. However, the existing voting-based and proof-based consensus protocols encounter scalability issues within the blockchain system. Moreover, most consensus protocols are serialized, which further limits their scalability potential. To address this limitation, parallelization methods have been employed in both types of consensus protocols. Surprisingly, however, novel fusion consensus protocols demonstrate superior scalability compared with these two types but lack the utilization of parallelization techniques. In this paper, we present PnV, an efficient parallel fusion protocol integrating proof-based and voting-based consensus features. It enhances the data structure, consensus process, transaction allocation, and timeout handling mechanisms to enable concurrent block generation by multiple nodes within a consensus round. Experimental results demonstrate that PnV exhibits superior efficiency, excellent scalability, and acceptable delay compared with Proof of Vote (PoV) and BFT-SMART. Moreover, at the system level, the performance of the PnV-based blockchain system optimally surpasses that of the FISCO BCOS platform. Our proposed protocol contributes to advancing blockchain technology by providing a more efficient and practical solution for achieving decentralized consensus in distributed systems.

Keywords: consortium blockchain; consensus protocol; parallelization

Citation: Wang, H.; Li, H.; Fan, P.; Kang, J.; Deng, S.; Zhu, X. PnV: An Efficient Parallel Consensus Protocol Integrating Proof and Voting. *Appl. Sci.* **2024**, *14*, 3510. https://doi.org/10.3390/app14083510

Academic Editors: George Drosatos, Konstantinos Rantos and Konstantinos Demertzis

Received: 26 February 2024
Revised: 9 April 2024
Accepted: 17 April 2024
Published: 22 April 2024

1. Introduction

Blockchain, also referred to as distributed ledger technology, emerged from the cryptocurrency system known as Bitcoin [1]. The ledger is collectively maintained by multiple parties and enables consistent, immutable, and traceable data storage. In recent years, owing to the rapid advancement of blockchain technology, its application has extended to various domains, such as finance [2], education [3], healthcare [4], and logistics [5], among others.

Blockchains achieve state consistency among distributed nodes through consensus protocols, which can be traced back to the Byzantine generals problem [6]. Consensus protocols in blockchain address Byzantine failures in two ways, where one is based on voting, exemplified by Practical Byzantine Fault Tolerance (PBFT) [7] and BFT-SMART [8]. Voting-based consensus protocols rely on message interaction among nodes, ensuring that each consensus round generates a single block without forks, thus demonstrating strong consistency. The other approach is based on proof mechanisms such as Proof of Work (PoW) [1] and Proof of Stake (PoS) [9]. Notably, these proof-based consensus protocols possess an inherent potential for generating divergent blocks for individual nodes when considering identical preceding blocks [10]. However, these conflicting blocks are subsequently reconciled by using specific resolution techniques to ensure final consistency [11].

As mentioned earlier, the performance bottlenecks of consensus protocols based on voting and proof are primarily attributed to the limitations in communication resources and computational capabilities of nodes, respectively. Consequently, each consensus type possesses its own merits regarding support for node numbers and efficiency. This drives the emergence of integrating both consensus types to leverage their respective advantages and achieve extensive scalability. A notable example is Proof of Vote (PoV) [12,13], which is a robust consensus protocol with high efficiency and strong security features. By combining voting-based and proof-based characteristics through a delegated mechanism, PoV establishes committee members as the central cluster responsible for monitoring and verifying block production via voting. Independence is ensured by an elected bookkeeper team responsible for block packaging. Only bookkeeper candidates recommended by committee members and who meet minimum participation and performance standards are eligible. These standards include depositing funds and subjecting their work to ongoing supervision and evaluation. This approach aims to select bookkeepers who demonstrate honesty and reliability over multiple election cycles, thereby guaranteeing the safety and reliability of team work. In addition, by leveraging implicit two-phase commit, PoV significantly reduces communication complexity while ensuring fairness within the committee. Other fusion consensus protocols include HotStuff [14] and Votes-as-a-Proof (VaaP) [15].

The proposal of fusion protocols effectively overcomes performance bottlenecks related to communication resources and computing power. However, the maintenance of a single-chain structure presents an additional challenge to blockchain scalability, as it necessitates the serialization of all concurrent blocks. To alleviate this limitation, an intuitive approach involves parallelization methods instead of solely relying on a single chain for achieving superior concurrency [16]. This idea has been implemented in various proof-based consensus protocols, including parallel-chain structures [17,18] and Directed Acyclic Graph (DAG)-based structures [19,20]. Additionally, DAG-based consensus protocols such as DAG-Rider [21], Tusk [22], and Bull-Shark [23] have been proposed as viable scalable voting-based solutions.

Unfortunately, while fusion consensus protocols have demonstrated greater scalability compared with other consensus types [24], the incorporation of parallelization approaches into them has not been thoroughly investigated.

Consequently, this paper aims to present PnV, an efficient parallel fusion protocol integrating proof-based and voting-based consensus features. It enhances the data structure, consensus process, transaction allocation, and timeout handling mechanisms to enable concurrent block generation by multiple nodes within a consensus round. Here, the letter P carries a dual meaning, i.e., parallelization and proof, whereas V denotes voting. In our previously published conference version [25], PnV was referred to as PPoV, a parallelization improvement of the PoV protocol. To enhance its practical functionality, we additionally incorporate transaction allocation and timeout handling mechanisms in this paper. Furthermore, we implement the proposed PnV protocol by using Golang [26] and conduct both theoretical and experimental analyses for comprehensive evaluation.

The main contributions of PnV encompass a novel data structure called block group, an innovative consensus process featuring parallel bookkeeping. Other crucial mechanisms, including transaction allocation and timeout handling, further optimize the protocol. These advancements empower the blockchain to facilitate simultaneous block generation by multiple nodes within a consensus round, thereby significantly enhancing throughput with an acceptable level of delay. Experimental results demonstrate that PnV achieves peak throughput ranging from approximately five times higher than PoV and BFT-SMART in the best-case scenario. In comparison with FISCO BCOS, a prominent enterprise-level consortium blockchain platform, the PnV-based blockchain system exhibits an impressive four-fold performance enhancement.

The remainder of the paper is organized as follows: Section 2 provides an overview of related work. In Section 3, we establish a preliminary definition for both the node and data model. The PnV consensus protocol is presented in detail in Section 4, followed

by theoretical and experimental analyses in Sections 5 and 6, respectively. Finally, our comprehensive conclusion is drawn in Section 7.

2. Related Work

This section provides a survey of relevant prior art regarding blockchain consensus protocols. Specifically, Section 2.1 discusses existing blockchain consensus protocols that integrate voting-based and proof-based approaches, highlighting their scalability limitations. Section 2.2 then examines existing solutions proposed to enhance blockchain scalability, including parallelization methods.

2.1. Blockchain Consensus Integrating Voting and Proof

Blockchain consensus protocols can be categorized into voting-based and proof-based protocols, and their fusion, as shown in Table 1. Strictly speaking, the first and third types represent advancements over Byzantine Fault Tolerant (BFT) protocols in traditional distributed theory, although the latter incorporates certain unique features of proof-based consensus specific to blockchain systems.

Table 1. Categorization of consensus protocols.

Consensus Type	Description	Protocols
Voting-based	Rely on message interaction and require each consensus round to produce a single block without forks. Demonstrate strong consistency.	PBFT, BFT-SMART, Algorand, DAG-Rider, Tusk, and Bull-Shark
Proof-based	Possess inherent potential for generating divergent blocks but reconcile conflicts through resolution techniques to ensure eventual consistency.	PoW, PoS, Monoxide, OHIE, Conflux, and Phantom
Fusion	Integrate voting-based and proof-based characteristics to leverage advantages of both while addressing individual limitations.	PoV, HotStuff, VaaP, and PnV (PPoV)

PBFT [7], as the most renowned BFT protocol, distinguishes between primary and replica nodes, effectively reducing the algorithmic complexity from exponential to polynomial level. The only primary node confirms and receives client requests, while replicas confirm the primary node's and each other's information to ensure consistency. This advancement made BFT protocols practically applicable. BFT-SMART [8] shares similarities with PBFT but offers enhanced reliability, modularity, and flexible programming interfaces. Due to its separation of tokens and high-speed verification capabilities, PBFT and its variants have sparked extensive discussions within communities. The consensus delay achieved is at a second-level magnitude, meeting real-time requirements while ensuring high-throughput performance. However, it should be noted that the communication cost scales with $O(n^2)$, implying that as the number of nodes (n) increases, there will be a rapid growth in data volume leading to increased network burden. To support large-scale networks, Algorand [27] employs a delegated mechanism wherein only a subset of nodes can participate in a Byzantine agreement protocol called BA$\star$ [28,29] for a transaction set during a consensus round. Specifically, it utilizes a verifiable random function to enable users to privately and demonstrably ascertain their successful election for the subsequent consensus round.

In order to enhance blockchain scalability, researchers have proposed incorporating proof-based consensus into voting-based consensus. The first fusion consensus protocol,

known as PoV [12], adopts a delegated approach by assigning voting rights and bookkeeping rights to voters and bookkeepers, respectively. Blocks are produced by bookkeepers and determined based on the proofs derived from voting results among voters. PoV exhibits linear communication complexity with $O(3n)$. HotStuff [14] extends PBFT's two phases to three phases while utilizing threshold signatures to reduce proof data and enable pipelined consensus process for improved throughput. Additionally, Sync HotStuff [30] presents a simplified and intuitive synchronous version featuring a two-phase vote. However, in scenarios involving multiple bookkeepers, these fusion protocols still face challenges where non-leader bookkeepers remain idle, thereby impacting the complete resolution of scalability issues.

2.2. Blockchain Scalability Solution

Currently, the utilization of a single-chain structure in most consensus protocols results in a sequential block generation process, thereby constraining the throughput of blockchain systems. To address this scalability issue and alleviate the limitations imposed by serialized writing on a single chain, parallelization emerges as an effective approach.

It is a widely accepted notion to enable nodes to concurrently extend parallel chains in order to enhance the throughput of the blockchain. In terms of proof-based consensus, OHIE [18] serves as an example for parallel-chain schemes. It adopts a k-chain structure comprising k parallel instances, each implementing a PoW consensus protocol. To establish the serialization of blocks across these k chains, an additional tuple $(rank, next\_rank)$ is included within each block. The blocks are then sorted based on ascending rank values. Monoxide [17] divides workload encompassing communication, computing, and storage into multiple independent and parallel consensus zones, with one blockchain dedicated to each zone. On the other hand, in voting-based consensus mechanisms like VaaP [15], a parallel chain operates on every node, aiming at improving scalability.

The scalability solution for blockchain extends beyond parallel chains to include a DAG. Unlike parallel-chain protocols that have multiple independent genesis blocks during initialization, DAG-based protocols only have one genesis block. Here, a genesis block means the initial founding block of a chain. These consensus protocols transform the ledger from a single chain into a DAG structure, enabling high concurrency support [31]. An example of proof-based consensus is GHOST [32], which treats the candidate block set and its reference relationship as a tree and selects the maximum weight subtree as the final main chain. Building upon GHOST, Conflux [19] enhances throughput and transaction confirmation speed by incorporating block references and explicit sequencing rules in its design. In the context of voting-based consensus, DAG-Rider [21] is an asynchronous BFT protocol that comprises two layers: a round-based structured DAG for reliable message dissemination and a zero-overhead consensus protocol that enables nodes to independently determine a total order of messages to commit without incurring additional communication overhead. Tusk [22] represents an enhanced version of DAG-Rider, which reduces block generation round delays under normal circumstances. The synchronous variant of Bullshark [23], built on top of Tusk, further minimizes latency while being easier to implement. Meanwhile, the asynchronous edition of Bullshark incorporates fallback voting mechanisms that endow it with comparable low-latency performance as its synchronous counterpart in regular scenarios.

However, regardless of the parallelization method employed in a consensus protocol, it often exhibits high latency due to the waiting time before a block is committed.

3. Node and Data Model

PnV is a consensus protocol specifically designed for consortium blockchains. This protocol effectively classifies network nodes into five distinct roles, thereby segregating bookkeeping rights from voting rights in an efficient manner. More precisely, designated bookkeepers possess exclusive privileges to generate blocks while ensuring that each

block undergoes meticulous verification by voters before its ultimate commitment in the blockchain.

3.1. Role Model

PnV defines five distinct roles for nodes within the blockchain network: voter, bookkeeper, bookkeeper candidate, leader, and user. Let N_v represent the number of voters, N_b denote the number of bookkeepers, N_{bc} signify the number of bookkeeper candidates, N_u indicate the number of users, and N_{all} represent the total count of nodes. With exception of the genesis block group, each subsequent block group is generated by N_b bookkeepers. The term during which a node acts as a bookkeeper is referred to as T_b. At the conclusion of a term, the next team of bookkeepers is elected among the pool of bookkeeper candidates. Figure 1 illustrates in detail the specific transition relationship between node roles. All transitions occur through special transactions at any time, which are processed by a three-phase consensus (Section 4), similar to normal transactions such as account creation and money transfer. The responsibilities of each role are described below.

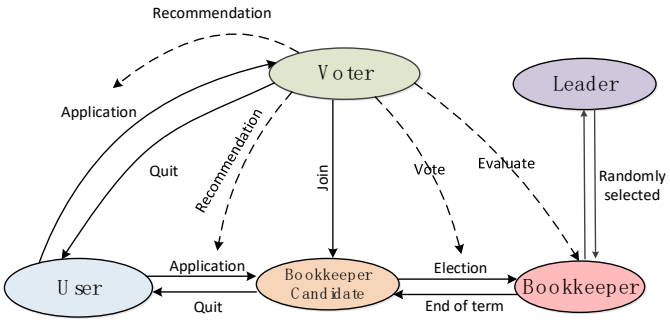

Figure 1. Role transition relationships.

Voter: Voters bear the responsibility of network supervision and are required to meticulously verify each block to ensure the accuracy of the transactions contained within. During a consensus round, each voter casts their vote in favor or against the blocks under consideration. Only blocks that receive more than two-thirds approval can be deemed legitimate. Furthermore, voters are entrusted with the crucial task of electing bookkeepers by thoroughly evaluating their performance scores, thereby determining the next set of bookkeepers once the current term concludes.

Bookkeeper: Bookkeepers are entrusted with the responsibility of conducting bookkeeping activities within the blockchain system. Similarly to voters, there exist multiple bookkeepers in the network. The blocks generated collectively by all bookkeepers during a consensus round constitute a block group. To ensure continuity and efficiency, the bookkeeper team operates under a tenure system whereby they are required to produce a number of block groups within their designated term. Specifically, each bookkeeper gathers relevant transactions from the network and stores them locally in their pool. Subsequently, when a consensus round commences, every bookkeeper assembles a set of transactions into a block and disseminates it across the network for verification purposes.

Bookkeeper candidate: Bookkeeper candidates are not directly involved in the consensus process, but they possess the necessary qualifications to be elected as new bookkeepers by voters. In order to ensure an adequate number of bookkeepers, it is imperative that the number of bookkeeper candidates exceeds or equals that of the existing bookkeepers. Any node has the right to apply for the role of bookkeeper candidate, and such candidates also have the freedom to voluntarily relinquish their roles.

Leader: During the consensus process, a single leader is selected from bookkeepers to oversee the collection and tallying of block votes generated by voters. Typically, if all blocks receive more than two-thirds of the votes, the leader generates a block group header and disseminate it across the network. At the conclusion of each consensus round, a new leader is randomly chosen based on information contained within the block group header. This approach ensures fair and impartial leadership selection while maintaining transparency throughout the voting process.

User: A newly added node initially lacks the authority to actively engage in any consensus process. However, it can passively receive data from other nodes and publish transactions. This type of node primarily operates as a user within the system, with its joining or exiting having no significant impact on the overall network dynamics.

3.2. Data Structure

In classic blockchains, a single block serves as the consensus and storage unit within each consensus round. These blocks are interconnected through hash values to form a chain, with each block being assigned a unique height number. To facilitate parallel data consensus and storage, we introduce the concept of a block group as the unit, as depicted in Figure 2. Each consensus round generates a distinct block group comprising multiple blocks, along with a corresponding block group header. The identification of each individual block within a block group is based on the block group height (i) and block number (j). According to the distinct transaction functions within block groups, we categorize them into three types: genesis block group, ordinary block group, and special block group.

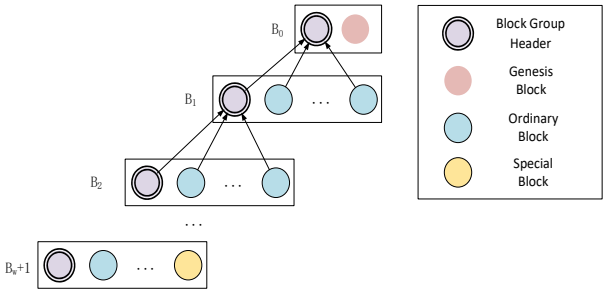

Figure 2. Block group composition.

Genesis block group: Similar to the genesis block in classic blockchains, the genesis block group serves as the initial block group responsible for ensuring the uniformity of the initial variables across all nodes. The genesis block group comprises a single block containing a transaction that typically records the public keys of bookkeepers, bookkeeper candidates, voters, and the leader in the initial state while also establishing system parameters.

Ordinary block group: An ordinary block group comprises a fixed number of blocks containing ordinary transactions, which is equivalent to the number of bookkeepers. Each block is generated by a specific bookkeeper, and its block number corresponds to the respective bookkeeper's number. In other words, an ordinary block group of height i can be represented as in Equation (1), where H_i is the block group header and B_{ij} refers to an individual block with number $j \in \{0, 1, 2, \ldots, N_b - 1\}$, with N_b being the total number of bookkeepers.

$$BG_i = \{H_i, B_{i0}, \ldots, B_{iN_b-1}\}. \quad (1)$$

Special block group: To ensure the robust functioning of the blockchain and facilitate voter oversight, it is imperative for voters to actively participate in electing the subsequent bookkeepers upon the end of the term. The voting result will be securely recorded within a dedicated transaction embedded in the final block group of that term, referred to as a special block group. In other words, the special block group $BG_i = \{H_i, B_{i0}\}$ with height i encompasses a block group header H_i and a block B_{i0}.

In particular, although there is no direct connection among blocks within a given block group, these groups are interconnected in an organized structure through hash values to ensure overall serialization. Apart from including the hash value of the previous block group header in the newly generated one, each block also necessitates recording the hash value of its preceding block group header in its own block header. Furthermore, within the block group header, it is essential to include the leader's number for the subsequent block group. It can be obtained randomly by concatenating the hash values of all valid blocks in the voting results and subsequently applying the modulus with respect to bookkeepers' count.

4. PnV Consensus Protocol

This section provides a comprehensive elucidation of the three-phase process entailed in generating append-only block groups within the PnV consensus protocol. With the exception of the genesis block group, all subsequent block groups necessitate consensus from both voters and bookkeepers. To enhance consensus speed, N_b bookkeepers engage in parallel block generation during a consensus round, ultimately amalgamating into a block group.

4.1. Three-Phase Consensus Process

The consensus process of a block group can be divided into three phases: propose, vote, and commit, as shown in Figure 3.

1. Propose Phase

Each bookkeeper selects a limited number of transactions from their memory pools based on the transaction allocation mechanism (Section 4.2) and generates a block. Subsequently, bookkeepers disseminate their respective blocks to the network. All nodes within the network, including users, will receive and store these blocks. The detailed pseudocode of the propose phase is in Algorithm 1.

Algorithm 1: Propose phase.

```
if do_propose = true do
    h←getBlockChainHeight() + 1;
    gen_block_group_num←h%(Bw + 1);
    if gen_block_group_num = 0 and leader do
        VoteForNextBookkeepers();
    end if
    block←GenerateBlock();
    Msg_Block←GenerateBlockMsg(block);
    for node in node_list do
        SendMessage(node, Msg_Block);
    end for
    do_propose = false;
end if
return
```

2. Vote Phase

When a voter receives blocks published by bookkeepers, it conducts verification to ensure the accuracy of transaction contents and key parameters such as block group height, block number, and signature. Based on the verification results, the voter generates its vote, *VoteTicket*, for the blocks and transmits it to the leader. The detailed pseudocode of the vote phase is in Algorithm 2.

Algorithm 2: Vote phase.

```
if do_vote = true do
    received_block_num←getLen(received_block_set);
    Nw←getBookkeeperAmount();
    if received_block_num< Nw and Nepoch = 0 do
        return
    end if
    Initial Vote_Ticket to a list with Nw {null, 0} elements;
    for block in received_block_set do
        block_num←getBlockNum(block);
        header←getHeader(block);
        hash←getHash(block);
        opnion←VerifiedBlock(block);
        Vote_Ticket[block_num]←Pair(hash,opinion);
    end for
    Msg_Vote←getVoteMsg(Vote_Ticket);
    node←getLeaderNode();
    sendMessage(node, Msg_Vote);
    do_vote = false;
end if
return
```

3. Commit Phase

The leader gathers the votes cast by all voters and stores them in $VoteList =< VoteTicket_1, VoteTicket_2 \ldots >$. Once all votes are received or a timeout (Section 4.3) occurs, a voting statistical rule is employed to generate the vote result, denoted as $VoteResult$. Subsequently, the vote list and result are encapsulated within the block group header before being disseminated across the network. The detailed pseudocode of the commit phase is in Algorithm 3.

Algorithm 3: Commit phase.

```
if do_commit = true do
    Can_Generated_Result, Result_List←VoteListStatistic(Vote_List);
    if Can_Generated_Result = false do
        return
    end if
    block_group_header←GenerateBlockGroupHeader(Vote_List, Result_List);
    Msg_Block_Group_Header←getBlockGroupHeaderMsg(block_group_header);
    for node in node_list do
        sendMessage(node, Msg_Block_Group_Header);
    end for
    do_commit =false;
end if
return
```

Specifically, the voting statistical rule is as follows: The leader generates a final voting result of length N_b, which corresponds to the number of bookkeepers. This result includes the hash value of each numbered block and its corresponding approval, objection, or no-opinion votes denoted by 1, -1, and 0, respectively.

As malicious nodes may distribute different blocks to various voters, it is possible for a specific block number to have distinct hash values in each vote. Assume that the vote for block p consists of m diverse hash values, which are sequentially numbered as $\{1, 2, 3, \ldots, m\}$. To represent the combination of the i-th hash value and its corresponding count of votes for approval, objection, and no opinion, we employ a triple $(hash_i, a_i, b_i)$.

Before a timeout occurs, a voter is limited to voting either in favor of or against a block, and there is no option for abstaining. Therefore, if any of the following conditions is met with respect to block p, a vote outcome will be generated for it:

Condition 1. *If* $\max_i b_i > N_v/2$, *block p will be marked as* -1 *to indicate objection.*

Condition 2. *If* $\max_i a_i > N_v/2$, *block p will be marked as 1, indicating approval for generating the block corresponding to the hash value with the number* $\arg\max_i a_i$.

Condition 3. *If the aforementioned two conditions are not met, block p will be marked as 0, indicating an inability to generate a valid block.*

Condition 4. *If a timeout occurs, the leader must gather votes exceeding* $2N_v/3$ *for each block within the block group and generate statistical results. In this scenario, a block is deemed legal only if it possesses a hash value that receives more than* $N_v/3$ *approval votes.*

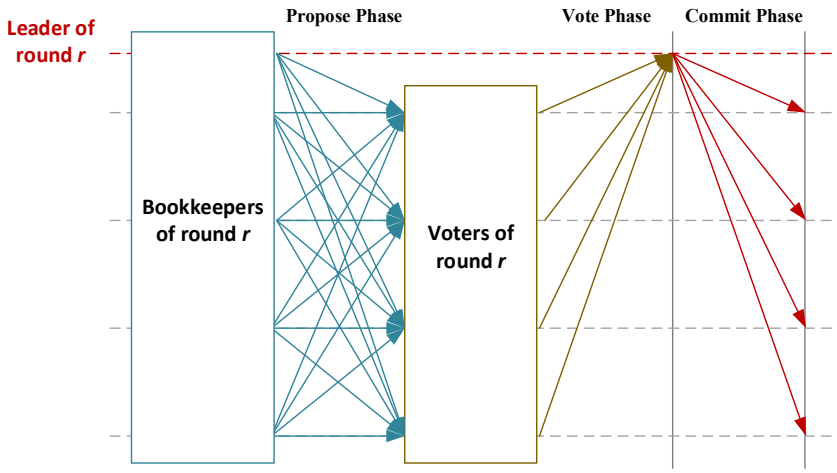

Figure 3. PnV consensus process in three phases.

4.2. Transaction Allocation Mechanism

In a blockchain network, nodes typically broadcast their transactions to multiple bookkeepers in order to increase the likelihood of their transactions being included in a block. Consequently, there tends to be a significant number of identical transactions present in the pools of different bookkeepers. In our proposed PnV protocol, multiple bookkeepers simultaneously generate blocks. However, if these transactions are not filtered properly, it is possible for the same transaction to be stored in multiple blocks within the same block group.

To address this issue, we introduce a transaction allocation mechanism wherein each transaction T_x is assigned an identifier $j (j = 0, 1, 2, \ldots, N_b - 1)$. The transaction identifier j is computed as

$$j = Hash(digest(T_x)) mod(N_b). \quad (2)$$

Subsequently, only transactions whose identifiers as calculated in Equation (2) match a bookkeeper's identifier are allowed to be included in blocks generated by that bookkeeper. This ensures that there is no intersection between the transactions contained within different blocks.

In addition to mitigating the issue of duplicate transaction storage, this can significantly enhance efficiency by constraining transaction packaging. For any user, as all bookkeepers concurrently generate blocks, there is no necessity to transmit the transaction to every bookkeeper for expediting transaction processing. Furthermore, even if a transaction is dispatched to another bookkeeper, it will be disregarded.

4.3. Timeout Handling Mechanism

Due to the inherent volatility of real networks and potential machine malfunctions, the blockchain system may experience stagnation or timeouts. To address this issue, we

propose a timeout handling mechanism. In order to comprehensively analyze the impact on the consensus process, we first focus on examining timeout failures related to three key node roles: voter, bookkeeper, and leader.

1. Voter failure: The determination of the legality or illegality of blocks generated by bookkeepers relies on the voting process conducted by voters. In case of voter failure, both positive and negative votes for a block may fall below the threshold, resulting in an inability to generate statistical results. Consequently, the leader will persistently await the fulfillment of statistical conditions, while other nodes will also experience prolonged waiting periods for the block group header.
2. Bookkeeper failure: Given that voters are required to receive all blocks from every bookkeeper prior to voting, any failure on the part of a bookkeeper will result in prolonged waiting times for voters as they await the arrival of these blocks.
3. Leader failure: In the event of a leader's failure, all nodes will enter a state of waiting for the block group header.

To address the failure of these nodes, we propose a timeout handling mechanism. Assuming that an epoch represents the smallest unit of a consensus round, denoted by t_0 for its start time and T_{cut} for its duration, N_{epoch} signifies the number of epochs elapsed from time t_0 to the current time, t:

$$N_{epoch} = (t - t_0)\%T_{cut}. \tag{3}$$

In a consensus round, the termination conditions for achieving consensus are as follows:

Condition 1. *Receive the block group header.*

Condition 2. *Receive all legal blocks passed by voting.*

If both conditions cannot be met during an epoch, the consensus process will surpass the time limit and require timeout handling, which can be classified into two distinct cases.

Case 1. *The legal block group header has been received. However, an insufficient number of legal blocks have been collected.*

In Case 1, the generation of the block group header signifies the completion of consensus. The insufficiency in receiving legal blocks could potentially be attributed to network failure. Consequently, it becomes imperative to solicit the missing blocks from alternative nodes.

Case 2. *The legal block group header has not been received.*

In Case 2, the absence of block group header generation may be attributed to a bookkeeper or leader failure. In such circumstances, it is necessary to proceed with the following steps:

Step 1. All nodes initiate a request to obtain the blockchain height from all voters and maintain an updated record of each voter's online status. In the event that there exists a voter whose blockchain height surpasses that of the local blockchain, a request is made to acquire block groups from said voter while simultaneously ceasing timeout handling. Conversely, if no such voter with a higher blockchain height is found, proceed with Steps 2–4.

Step 2. Promptly execute the voting operation by all voters. In case a block has been received, proceed with the conventional voting process. Otherwise, assign a value of 0 (indicating no opinion), and set the hash value as nil. Subsequently, transmit the vote to the leader of the subsequent epoch.

Step 3. Upon receiving more than $2N_v/3$ votes, the leader proceeds to tally the votes for each block within the block group, thereby generating a statistical result. At this juncture, if a block accumulates over $N_v/3$ votes for any given hash value, it is deemed a valid block and assigned a vote result of 1; otherwise, it is classified as an invalid block with a corresponding vote result of -1.

Step 4. If $N_{epoch} > 1$, indicating multiple occurrences of timeout, it becomes imperative to replace the leader. Assuming the bookkeeper number prior to this timeout to be denoted by n, the subsequent leader's number shall be determined by $(n+1)\%\ N_b$.

These steps ensure the eventual generation of a legal block group in a consensus round. Consequently, the timeout handling mechanism serves as a crucial supplement and enhancement to the PnV consensus protocol, thereby enhancing robustness and enabling normal blockchain operation even under abnormal conditions.

5. Theoretical Analysis

5.1. Resistance to Conflict Transactions

Let us consider a conflict transaction problem in the context of parallel and distributed block generation by multiple bookkeepers in PnV. In this scenario, if there exist conflict transactions within pools of bookkeepers, it raises the question of whether these conflict transactions will be stored in separate blocks within the final block group.

Let us suppose that there are two bookkeepers participating in a consensus round, publishing blocks A and $A\prime$, respectively, containing conflicting transactions. As per the requirement for voters to verify all blocks within this block group height, they can only vote for one of the blocks while voting against the other. Consequently, if block A receives more than half of the votes as a valid block, it follows that block $A\prime$ will receive less than half of the votes and thus be deemed invalid. Henceforth, it is impossible for two conflict transactions to coexist within the same block group.

5.2. Performance Analysis

In terms of efficiency, PnV optimizes block generation through the implementation of parallel processing, where multiple bookkeepers assume responsibility for producing blocks simultaneously. This allows transaction processing to occur while awaiting the block group header, effectively utilizing idle computing resources within each node and enhancing transaction packaging speed. Consequently, this protocol significantly improves overall throughput.

On the other hand, in terms of cost, the issue of transaction competition can be fundamentally resolved due to each bookkeeper solely accepting and packaging a portion of transactions. Consequently, each transaction only necessitates transmission to a single bookkeeper, thereby diminishing superfluous communication while also reducing network bandwidth consumption and node memory resource utilization.

6. Research Method and Experiment

In this section, we implement the proposed PnV protocol in a consortium blockchain environment by using Golang [26]. We conducted our experiments on four ZTE R8500 G4 servers. Each server was equipped with 128 GB of memory and an Intel(R) Xeon(R) Gold 6248R Processor running at a speed of 2.3 GHz $\times$ 96. In order to eliminate any potential impact from read–write disk operations, each node did not store the block group in the database. Other parameter settings can be found in Table 2.

Table 2. Configuration of parameters.

Parameter	Value
Transaction size	40 Bytes
Block header size	692 Bytes
Vote header size	400 Bytes
Vote size of a block	100 Bytes
Vote result header size	170 Bytes
Vote result size of a block	400 Bytes
Message header size	266 Bytes
Bandwidth	10 Gbps

6.1. Consensus Throughput and Network Load

First, we set the number of voters to 4 ($N_v = 4$) to examine the influence of varying the number of bookkeepers (N_b) and the number of transactions per block (K). By manipulating both variables, we record the corresponding values for consensus throughput and network load in each scenario.

The results depicted in Figure 4 demonstrate that the throughput of PnV exhibits an initial increase followed by a subsequent decrease as the number of bookkeepers increases. Moreover, the throughput experiences varying degrees of enhancement with an escalation in the number of transactions per block. The peak throughput reaches over 350,000 transactions per second (tx/s). In terms of network load, it is inevitable for communication data between nodes to augment and consequently lead to an increase in network load as the number of nodes increases. This can be attributed to the fact that both transmission time and processing time escalate alongside an increase in transaction volume. When there are only a few nodes, the rate at which time consumption increases is lower than that of transactions, resulting in an upward trend for throughput. Conversely, when there is a large number of nodes, the pace at which time consumption grows surpasses that of transactions, leading to a downward trend for throughput.

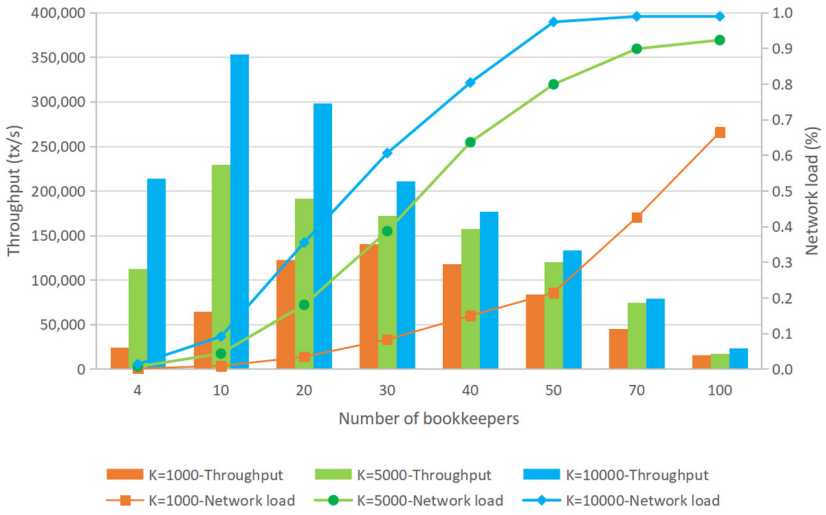

Figure 4. Throughput and network load for PnV consensus protocol ($N_v = 4$).

6.2. Comparison from an Algorithmic Perspective

In this section, we present a comparative analysis of the throughput and delay performance among PnV, PoV, and BFT-SMART [8]. We specifically chose these two protocols, as PoV represents the first fusion consensus approach, while BFT-SMART serves as a prominent example of voting-based consensus. To ensure comparability with BFT-SMART, both PnV and PoV were configured with an identical number of voters to bookkeepers ($N_v = N_b = n$). The maximum number of transactions per block (K) was set to 10,000.

The experimental results are illustrated in Figure 5, showcasing the trend of PnV's throughput as initially increasing and then decreasing as the number of nodes increases. In contrast, both PoV and BFT-SMART exhibit a gradual decline in throughput. Notably, PnV achieves its peak throughput at $n = 20$, which is approximately five times higher than that of PoV and BFT-SMART. Furthermore, even in the worst-case scenario, PnV's throughput remains about twice as high as that of PoV and BFT-SMART. In terms of delay performance, PnV demonstrates relatively favorable results when fewer nodes are involved. Consequently, compared with PoV and BFT-SMART, PnV exhibits superior efficiency, scalability, and acceptable delay in scenarios with a moderate number of nodes.

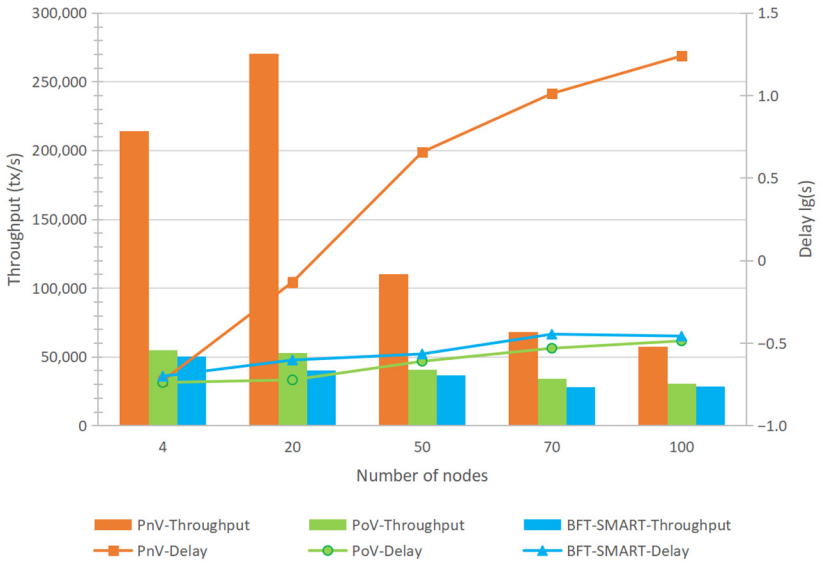

Figure 5. Throughput and delay for PnV, PoV, and BFT-SMART ($K = 10,000$).

6.3. Comparison from a Systemic Perspective

In this section, we realize a PnV-based blockchain system that encompasses two essential functionalities: account creation and money transfer. To evaluate the effectiveness of our system, we conduct a comparative analysis with FISCO BCOS [33], an enterprise-level financial consortium blockchain platform. Moreover, to ensure optimal performance, we configure each block to accommodate 10,000 transactions.

From Figure 6, it is evident that the throughput of both the PnV-based blockchain system and FISCO BCOS experience a decline with an increase in the number of nodes. Notably, when $n = 4$, both systems exhibit optimal performance; however, PnV outperforms FISCO BCOS by approximately four times.

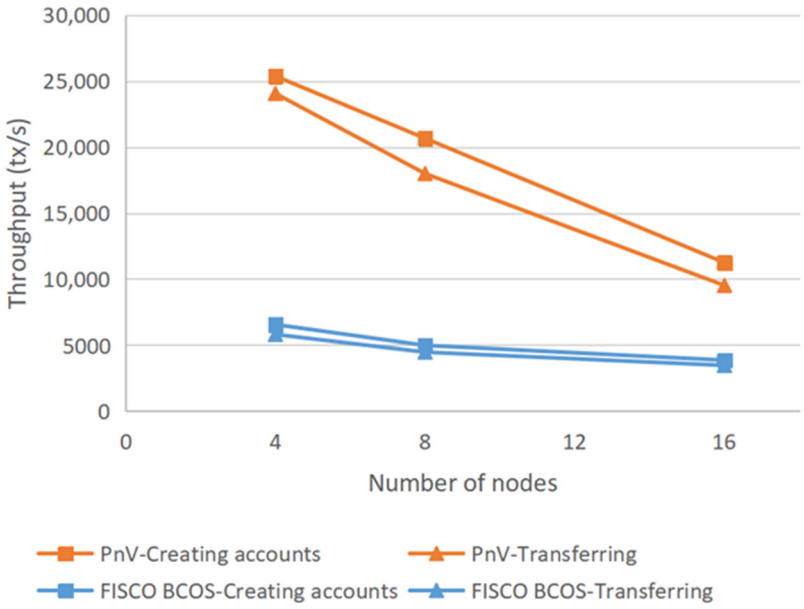

Figure 6. Throughput for PnV-based blockchain system and FISCO BCOS ($K = 10,000$).

7. Conclusions

In this paper, we propose the PnV consensus protocol, which facilitates concurrent block generation by multiple bookkeepers within a consensus round. Moreover, we adopt

the block group as the fundamental unit of consensus. Blocks lacking logical sequence relationships are stored in the block group, with their hash value being stored in the block group header for constructing a serialized, chained structure. Our transaction allocation mechanism prevents duplicate transactions from being stored within one block group, while our timeout handling mechanism ensures termination of consensus.

Compared with other voting-based and fusion consensus protocols, PnV exhibits superior performance in efficiency and scalability due to its parallel block generation and optimized resource utilization. As demonstrated via our experiments, PnV achieves throughput exceeding 350,000 txs/s, significantly outperforming serial alternatives.

While PnV inherits security, traceability, and assured inviolability properties from consortium blockchain techniques such as cryptography, hashing, and Byzantine Fault Tolerance, specialized designs are not proposed in this paper for those aspects. Going forward, future work will involve developing customized solutions within PnV to advance metrics like robustness, anonymity, and verifiability. Comprehensive testing in real-world networks will also help validate PnV's performance for diverse use cases.

Author Contributions: Conceptualization, H.W.; Methodology, H.W.; Software, P.F.; Validation, X.Z.; Formal analysis, H.W.; Investigation, H.W.; Resources, H.L., P.F., J.K., S.D. and X.Z.; Data curation, J.K.; Writing—original draft, H.W.; Writing—review & editing, H.L.; Visualization, X.Z.; Supervision, S.D.; Project administration, H.L.; Funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Keystone Research and Development Program of China, grant number 2017YFB0803204; Foshan Innovation Team, grant number 2018IT100082; Basic Research Enhancement Program of China, grant number 2021-JCJQ-JJ-0483; China Environment for Network Innovation, grant number GJFGW [2020]386, SZFGW [2019]261; Guangdong Province Research and Development Key Program, grant number 2019B010137001; Guangdong Province Basic Research, grant number 2022A1515010836; Shenzhen Research Programs, grants number JCYJ20220531093206015, JCYJ20210324122013036, and JCYJ20190808155607340; Shenzhen Fundamental Research Program, grant number GXWD20201231165807007-20200807164903001; ZTE Funding, grant number 2019ZTE03-01; and Huawei Funding, grant number TC20201222002.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

Conflicts of Interest: Authors Ping Fan, Jian Kang, Selwyn Deng and Xiang Zhu were employed by the company China United Network Communication Group Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *21260*, 1–9. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 9 April 2024). [CrossRef]
2. Ariffin, N.; Ismail, A.Z. The design and implementation of trade finance application based on hyperledger fabric permissioned blockchain platform. In Proceedings of the 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 5–6 December 2019; pp. 488–493.
3. Liyuan, L.; Meng, H.; Yiyun, Z.; Reza, P. E^2 c-chain: A two-stage incentive education employment and skill certification blockchain. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 140–147.
4. Guo, H.; Li, W.; Nejad, M.; Shen, C.C. Access control for electronic health records with hybrid blockchain-edge architecture. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 44–51.
5. Fu, Y.; Zhu, J. Operation mechanisms for intelligent logistics system: A blockchain perspective. *IEEE Access* **2019**, *7*, 144202–144213. [CrossRef]
6. Lamport, L.; Shostak, R.; Pease, M. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]
7. Castro, M.; Liskov, B. Practical byzantine fault tolerance. In Proceedings of the OSDI'99: Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999; pp. 173–186.

8. Bessani, A.; Sousa, J.; Alchieri, E.E.P. State machine replication for the masses with bft-smart. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 23–26 June 2014; pp. 355–362.
9. King, S.; Nadal, S. Ppcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Self-Published Paper, August 2012. Available online: https://bitcoin.peryaudo.org/vendor/peercoin-paper.pdf (accessed on 9 April 2024).
10. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. Modeling the impact of network connectivity on consensus security of proof-of-work blockchain. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 1648–1657.
11. Zhang, C.; Wu, C.; Wang, X. Overview of blockchain consensus mechanism. In Proceedings of the 2020 2nd International Conference on Big Data Engineering, Shanghai, China, 29–31 May 2020; pp. 7–12.
12. Li, K.; Li, H.; Hou, H.; Li, K.; Chen, Y. Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain. In Proceedings of the 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Bangkok, Thailand, 18–20 December 2017; pp. 466–473.
13. Li, K.; Li, H.; Wang, H.; An, H.; Lu, P.; Yi, P.; Zhu, F. Pov: An efficient voting-based consensus algorithm for consortium blockchains. *Front. Blockchain* **2020**, *3*, 11. Available online: https://www.frontiersin.org/article/10.3389/fbloc.2020.00011 (accessed on 9 April 2024). [CrossRef]
14. Yin, M.; Malkhi, D.; Reiter, M.K.; 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.
15. Fu, X.; Wang, H.; Shi, P. Votes-as-a-proof (vaap): Permissioned blockchain consensus protocol made simple. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4964–4973. [CrossRef]
16. Xu, J.; Wang, C.; Jia, X. A survey of blockchain consensus protocols. *ACM Comput. Surv.* **2023**, *55*, 278. [CrossRef]
17. Wang, J.; Wang, H. Monoxide: Scale out blockchains with asynchronous consensus zones. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, USA, 26–28 February 2019; pp. 95–112.
18. Yu, H.; Nikolic, I.; Hou, R.; Saxena, P. Ohie: Blockchain scaling made simple. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 90–105.
19. Li, C.; Li, P.; Zhou, D.; Yang, Z.; Wu, M.; Yang, G.; Xu, W.; Long, F.; Yao, A.C.-C. A decentralized blockchain with high throughput and fast confirmation. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20), Online, 15–17 July 2020; pp. 515–528.
20. Li, X.; Zheng, Y.; Xia, K.; Sun, T.; Beyler, J. Phantom: An efficient privacy protocol using zk-snarks based on smart contracts. *Cryptol. ePrint Arch.* **2020**, *2020*, 156.
21. Keidar, I.; Kokoris-Kogias, E.; Naor, O.; Spiegelman, A. All you need is dag. In Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, 26–30 July 2021; pp. 165–175.
22. Danezis, G.; Kokoris-Kogias, L.; Sonnino, A.; Spiegelman, A. Narwhal and tusk: A dag-based mempool and efficient bft consensus. In Proceedings of the Seventeenth European Conference on Computer Systems, Rennes, France, 5–8 April 2022; pp. 34–50.
23. Spiegelman, A.; Giridharan, N.; Sonnino, A.; Kokoris-Kogias, L. Bullshark: Dag bft protocols made practical. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; pp. 2705–2718.
24. Xie, M.; Liu, J.; Chen, S.; Lin, M. A survey on blockchain consensus mechanism: Research overview, current advances and future directions. *Int. J. Intell. Comput. Cybern.* **2023**, *16*, 314–340. [CrossRef]
25. Bai, Y.; Zhi, Y.; Li, H.; Wang, H.; Lu, P.; Ma, C. On parallel mechanism of consortium blockchain: Take pov as an example. In Proceedings of the 2021 the 3rd International Conference on Blockchain Technology, Shanghai, China, 26–28 March 2021; pp. 147–154.
26. Min-Group/pnv-Blockchain. Available online: https://github.com/MIN-Group/pnv-blockchain (accessed on 9 April 2024).
27. Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017; pp. 51–68.
28. Chen, J.; Micali, S. Algorand. *arXiv* **2016**, arXiv:1607.01341.
29. Micali, S. Fast and furious byzantine agreement. In Proceedings of the Innovations in Theoretical Computer Science (ITCS) Conference, Berkeley, CA, USA, 9–11 January 2017.
30. Abraham, I.; Malkhi, D.; Nayak, K.; Ren, L.; Yin, M. Sync hotstuff: Simple and practical synchronous state machine replication. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 106–118.
31. Pervez, H.; Muneeb, M.; Irfan, M.U.; Haq, I.U. A comparative analysis of dag-based blockchain architectures. In Proceedings of the 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, 19–21 December 2018; pp. 27–34.

32. Sompolinsky, Y.; Zohar, A. *Secure High-Rate Transaction Processing in Bitcoin*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 507–527.
33. Li, H.; Chen, Y.; Shi, X.; Bai, X.; Mo, N.; Li, W.; Guo, R.; Wang, Z.; Sun, Y. Fisco-bcos: An enterprise-grade permissioned blockchain system with high-performance. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 12–17 November 2023; pp. 1–17.