DP-Hybrid: A Two-Layer Consensus Protocol for High Scalability in Permissioned Blockchain

Fulin Wen[1], Lei Yang[2]✉, Wei Cai[3], and Pan Zhou[4]

[1] School of Software Engineering, South China University of Technology, Guangzhou 510006, China, 201921043987@mail.scut.edu.cn
[2] School of Software Engineering, South China University of Technology, Guangzhou 510006, China, sely@scut.edu.cn
[3] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China, and
Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518172, China, caiwei@cuhk.edu.cn
[4] School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China, panzhou@hust.edu.cn

Abstract. The permissioned blockchain has attracted the attention of multiple industries like the supply chain due to its decentralization and data tamper resistance. In these industries applications, the permissioned blockchain maintained by multiple participants often has a large number of nodes. The PBFT consensus is commonly used in the permissioned blockchain, but it requires a large amount of message transmission to reach consensus, resulting in poor scalability. In this paper, we propose DP-Hybrid, a novel two-layer consensus protocol, to reduce the communication costs and improve scalability. Specifically, nodes use PBFT to establish K autonomous systems at the bottom layer, and then participate at the top layer with Constrained PoW consensus protocol. DP-Hybrid reduces the communication costs from PBFT's $O(N^2)$ to $O(N^2/K^2)$. The experiment results show that DP-Hybrid's throughput is always about 10 times that of PBFT when the number of nodes increases.

Keywords: Blockchain; Permissioned Blockchain; Consensus Protocol; Scalability; Throughput

1 Introduction

Blockchain utilizes special consensus protocols in a decentralized network to maintain data consistency between nodes and provide tamper-proof capability. The permissioned blockchain is a blockchain that provides node authentication. The Practical Byzantine Fault Tolerance (PBFT) [1] is widely used in permissioned blockchain. It achieves high throughput and low latency when the number of nodes N is small. However, when N rises, its performance drops rapidly because of the $O(N^2)$ communication costs required to reach consensus.

Optimizations of PBFT were proposed in [2–4]. Researchers also proposed new consensus to achieve higher performance. In [5,6], fault tolerance is sacrificed

for faster consensus speed. Hierarchical consensus protocols were proposed in [7–13], which reduce the number of nodes participating in consensus and sacrifice fault tolerance to improve scalability. [7–13] use a voting-based consensus at the high layer, and there are leader nodes which vote on behalf of ordinary nodes to reduce the communication costs, which inevitably reduces security. We believe that non-voting-based consensus, such as Proof of Work (PoW), is more suitable as a high-layer consensus protocol in hierarchical consensus protocols.

In this paper, we propose DP-Hybrid to achieve high throughput and scalability in permissioned blockchain. DP-Hybrid uses PBFT as the bottom-layer consensus protocol and Constrained PoW (CPoW) as the top-layer consensus protocol. In DP-Hybrid, nodes are divided into groups at the bottom layer. The nodes within the same group communicate with each other based on PBFT. All the nodes represent their own group to communicate with other participants' nodes based on CPoW at the top layer. The hierarchical structure reduces the communication costs and improves scalability. Meanwhile, DP-Hybrid's security is not sacrificed and it can be configured to meet requirements. In contrast to [7–13], there are no leader nodes to represent the ordinary nodes in DP-Hybrid and thus the Byzantine failure of leader nodes does not exist.

We conduct extensive experiments to evaluate DP-Hybrid's throughput. Results show that DP-Hybrid's throughput is about 10 times of PBFT in the experimental environment. As the number of nodes increases, throughput of both consensus protocols decreases, but DP-Hybrid's throughput is always about 10 times of PBFT. We summarize our contributions as follows:

- To the best of our knowledge, DP-Hybrid is the first consensus protocol that combines CPoW and PBFT to improve scalability of permissioned blockchain without sacrificing security.
- We evaluate DP-Hybrid's throughput using extensive experiments. The results show that DP-Hybrid's throughput is always higher than PBFT when the number of nodes increases.

The remainder of this paper is organized as follows. Section 2 introduces the preliminary knowledge of blockchain and consensus protocols. Section 3 describes our design of DP-Hybrid. Section 4 analyzes communication costs, security, liveness and latency of DP-Hybrid and comparative consensus protocols. Section 5 presents the experiment we conducted to evaluate DP-Hybrid. Section 6 discusses the related works. Section 7 concludes this paper.

2 Preliminary

2.1 Consensus Protocol

In a distributed system, consensus protocols are required to maintain consistent data across nodes. Blockchain is a special distributed system, whose data can only be added but not deleted or modified. Therefore, consensus protocols in the blockchain mainly describe the rules or processes of adding new data.

Table 1. Comparison of Different Consensus Protocols

	Type	Scalability	Throughput	Latency	Node authorization	Fault tolerance
PoW	PCP	good	low	high	no	-
PoS	PCP	good	low	high	no	-
PBFT	DCP	bad	high	low	yes	$\lfloor \frac{N-1}{3} \rfloor$

There are two types of consensus protocols in the blockchain, i.e., Deterministic Consensus Protocol (DCP) and Probabilistic Consensus Protocol (PCP). The outcome of DCP is irreversible if consensus is reached between the nodes. Conversely, the outcome of PCP consensus is reversible, but will gradually strengthen over time, making the outcome a definitive result. We compare common consensus protocols in terms of type, latency, etc. The details are listed in Table 1.

PCP has better scalability, where nodes can freely enter or exit the blockchain. In contrast, node changes in DCP require modifications of the remaining nodes. DCP has higher throughput and lower latency than PCP but performance drop rapidly as the number of nodes increases. PCP can be used in public blockchain without node authorization and it's fault tolerance is independent of the number of nodes, but related to node's read-world assets. Conversely, DCP can only be used in permissioned blockchain with node authorization to resist the Sybil Attack. DCP can tolerate a fixed proportion of faulty nodes.

2.2 PBFT Consensus Protocol

In the PBFT, N nodes are numbered from 0 to N-1. The PBFT can tolerate $F = \lfloor \frac{N-1}{3} \rfloor$ faulty nodes. Nodes move through a succession of configurations called views. Views are numbered consecutively starting from 0. In each view v, the node i with $i = v\%N$ acts as the *primary node* and the others act as *backup nodes*. The *primary node* accepts clients' requests and initiates PBFT to reach consensus. Each node sends the result to the client after reaching consensus. The client waits for F+1 replies with the same result from different nodes.

If the client does not receive enough replies before timeout, it broadcasts the request to *backup nodes*. If the request has been processed, *backup nodes* resend the result. Otherwise, *backup nodes* initiate a *PBFT view change* to generate a new *primary node*. Each *backup node* broadcasts a *view change* message to the others. if a node receives $2\times F$ same *view change* messages, it updates its v=v+1. After that, nodes reprocess the client's request.

The protocol steps in the normal case of no *primary node* faults are as follows:

Step 1: Request. Client sends a request r to the *primary node*.

Step 2: Pre-Prepare (PP). The *primary node* broadcasts a pre-prepare message containing r to *backup nodes*.

Step 3: Prepare (P). Each *backup node* broadcasts a prepare message to the other nodes to confirm r's contents.

Step 4: Commit (C). Each node broadcasts a commit message to the other nodes to confirm the execution of r.

Step 5: Reply. All the nodes execute r and reply to the client.

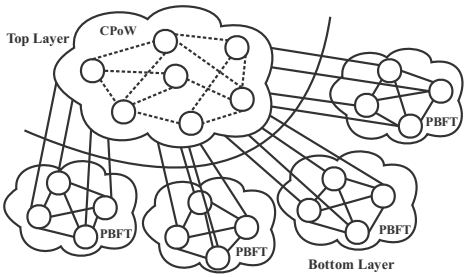

Fig. 1. Network model

According to the above process, the communication costs required to reach consensus in PBFT are $O(N^2)$. As N increases, the number of messages increases rapidly, which results in decreased throughput and increased latency. For above reasons, we believe that PBFT is not suitable for industries scenarios where there are a large number of nodes. We propose DP-Hybrid, which combines CPoW and PBFT to reduce communication costs and imporves scalability.

3 Design of DP-Hybrid

3.1 DP-Hybrid Structure

As shown in Fig. 1, DP-Hybrid is divided into two layers, i.e., the bottom layer and the top layer. We assume that K participants jointly maintain a DP-Hybrid blockchain. The participants could be companies which join the blockchain for data sharing. There exists total N nodes in the blockchain and each participant has N/K nodes. The nodes within the same participant are called *internal nodes* and the other participants' nodes are called *external nodes*. The *internal nodes* of a participant forms a group at the bottom layer and communicate with each other based on PBFT. Meanwhile, all the nodes can represent their own groups to communicate with *external nodes* based on the CPoW at the top layer. The details of the consensus process will be discussed in section 3.2.

We define notations as follows. P_i is the i^{th} participant. N_i is the number of nodes in P_i. $F_i = \lfloor \frac{N_i - 1}{3} \rfloor$ is the tolerable number of faulty nodes in P_i. n_i^j is the j^{th} node in P_i. Every node knows all P_i, N_i and F_i $(i = 1 \dots K)$. They also know which participant each node belongs to. We list the notations in Table 2.

In the initialization, the *internal nodes* within P_i establish connections with each other and keep *external nodes'* communication addresses. After that, a *PBFT view change* occurs in P_i to generate a *primary node* and then each *internal node* in P_i broadcasts a message to the *external nodes*. The message is named by External View Changed (ext-VC) message, which includes the information of the new *primary node*. If a *external node* receives $2 \times F_i + 1$ same ext-VC messages from P_i, it updates P_i's *primary node* information. With the ext-VC message, *external nodes* can identify P_i's *primary node*.

After finishing initialization, if a new node joins P_i or an existing node exits P_i, P_i's PBFT network system needs to be reconfigured. After that, except for

Table 2. Notations in This Paper

Notation	Meaning	Notation	Meaning
N	The total number of nodes	$\langle m \rangle_i^j$	Message m signed by n_i^j
K	The total number of participants	$\langle m \rangle$	Unsigned message m
P_i	The i^{th} participant	v_i	The PBFT view number of P_i
N_i	The number of nodes in P_i	Tx	The transaction from client
F_i	The tolerable number of faulty nodes in P_i	B_a	The a^{th} block in the CPoW blockchain
n_i^j	The j^{th} node in P_i	l_i	The latest ID of Tx in P_i

the new node or the exited node, each *internal node* in P_i broadcasts a message to the *external nodes*. The message is named by External Node Changed (ext-NC) message, which includes the information of the new node or the exited node. If a *external node* receives $2\times F_i+1$ same ext-NC messages from P_i, it updates N_i, F_i and P_i's nodes information. With the ext-NC message, the *external node* can verify the new node's messages or ignore the exited node's messages.

In the operation of the DP-Hybrid blockchain, the client sends a transaction Tx to the *primary node* of a random P_i. The nodes in P_i reach local consensus on Tx via PBFT, and the local consensus is broadcast to other participants. Each participant collects local consensus from other participants. Meanwhile, all the participants compete to package both internal and external local consensus into CPoW blocks. To make a CPoW block valid, they need to find a nonce that makes the block's hash value less than a threshold. The valid CPoW blocks and transactions within the blocks are considered as global consensus.

3.2 Consensus Process

The following descriptions focus on the nodes of one participant P_i with N_i and F_i, while the nodes of the remaining participants are collectively referred to as *external nodes*. For simplicity, we use n_i^0 as the *primary node* and n_i^j (j=1,2... N_i-1) as *backup nodes* in P_i to describe the consensus process.

At the bottom layer, each *internal node* acts as a PBFT state machine with an initial bottom state (*b_state*) of *Pre-Prepare* and performs state transitions as shown in Fig. 2. When a client sends a transaction to the *primary node*, the *primary node* triggers bottom state transitions. In each state, each *internal node* broadcasts confirmation message and waits for $2 \times F_i$ identical messages from other *internal nodes*. In the *Commit* state, the transaction is transformed into local consensus, and waiting to be packed into CPoW blocks.

The details of the bottom state transitions of the *primary node* and *backup nodes* are given in Algorithm 1 and Algorithm 2. We denote the message m signed by n_i^j as $\langle m \rangle_i^j$, the unsigned one as $\langle m \rangle$, the view number of P_i as v_i, the transaction as Tx, the assigned ID of Tx as l_i and the digest of Tx as d.

After reaching local consensus, n_i^0 broadcasts $\langle P_i, Tx, l_i \rangle_i^0$ to *external nodes*. This allows Tx to be packed into CPoW block by other participants. Only the local consensus containing valid Tx, unused l_i and signature of P_i's *primary node* is valid and accepted. The *primary node* may send transactions that have not

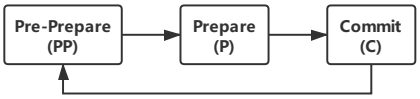

Fig. 2. State transitions of nodes at the bottom layer

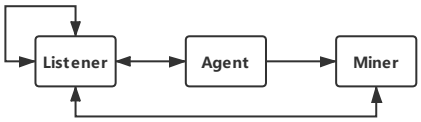

Fig. 3. State transitions of nodes at the top layer

Algorithm 1 PBFT for primary node at the Bottom Layer

$l_i \leftarrow 1$, $b\_state \leftarrow PP$
for l_i=1,2,3... **do**
  if receive Tx from client & $b\_state = PP$ **then**
    broadcast $\langle\langle PP, v_i, l_i, d\rangle_i^0, Tx\rangle$ to *backup nodes*
    $b\_state \leftarrow P$
    wait for $2 \times F_i$ prepare messages with same v_i, l_i and d from different nodes
  end if
  if meet the above conditions & $b\_state = P$ **then**
    broadcast $\langle C, v_i, l_i, d, 0\rangle_i^0$ to *backup nodes*
    $b\_state \leftarrow C$
    wait for $2 \times F_i$ commit messages with same v_i, l_i and d from different nodes
  end if
  if meet the above conditions & $b\_state = C$ **then**
    save $\langle P_i, l_i, Tx\rangle$ as local consensus
    reply to the client that Tx has been transformed into local consensus
    broadcast $\langle P_i, l_i, Tx\rangle_i^0$ to *external nodes*
    $l_i \leftarrow l_i$+1, $b\_state \leftarrow PP$
  end if
end for

Algorithm 2 PBFT for backup node at the Bottom Layer

$l_i \leftarrow 1$, $b\_state \leftarrow PP$
for l_i=1,2,3... **do**
  if receive $\langle\langle PP, v_i, l_i, d\rangle_i^0, Tx\rangle$ from *primary node* & $b\_state = PP$ **then**
    broadcast $\langle \mathrm{P}, v_i, l_i, d, j\rangle_i^j$ to the *primary node* and other *backup nodes*
    $b\_state \leftarrow P$
    wait for $2 \times F_i$ prepare messages with same v_i, l_i and d from different nodes
  end if
  if meet the above conditions & $b\_state = P$ **then**
    broadcast $\langle C, v_i, l_i, d, j\rangle_i^j$ to the *primary node* and other *backup nodes*
    $b\_state \leftarrow C$
    wait for $2 \times F_i$ commit messages with same v_i, l_i and d from different nodes
  end if
  if meet the above conditions & $b\_state = C$ **then**
    save $\langle P_i, l_i, Tx\rangle$ as local consensus
    reply to the client that Tx has been transformed into local consensus
    $l_i \leftarrow l_i$+1, $b\_state \leftarrow PP$
  end if
end for

reached local consensus to *external nodes*, but this does not have benefit. Because invalid transactions will not be accepted anyway while valid transactions have more opportunities to be packed into CPoW block after reaching local consensus.

While running PBFT to deal with transactions from client, each node communicates with *external nodes* according to the top-layer CPoW consensus protocol. We define the two constraints of CPoW:

- Confirmation Number (CN): If there are not less than CN blocks following the a^{th} block B_a, the transactions in B_a become global consensus.
- Maximum number of Blocks in CN (MBC): A maximum of MBC blocks from the same participant are allowed in CN consecutive blocks.

These constraints reduce the competition of computing resources and make security not entirely dependent on the distribution of computing resources. Due to the MBC constraint, not all the nodes in a participant perform the Write Operation, which wastes computing resources. Therefore, we define three top states (t_state) of the nodes when participating at the top layer as follows:

- Listener: Only performs the Read Operation.
- Miner: Performs both Read and Write Operations.
- Agent: Determines whether to perform the Write Operation and broadcasts signed message containing the set S of miners' ID if needed.

The top state transitions are shown in Fig. 3. Algorithm 3 and Algorithm 4 describe the Write Operation and the Read Operation. The initial t_state of the nodes is $Listener$. We use the timestamp of the latest CPoW block, denoted as T, to generate the agent node and thus the agent node changes after receiving or generating new blocks. The n_i^j with $j = T\%N_i$ acts as the agent node and determines whether to perform the Write Operation according to the MBC constraint. If the MBC constraint is violated, the agent node broadcasts an empty set. Otherwise, the agent node generates a miner set S based on the incentive policy and broadcasts it to other *internal nodes*. The agent node soon changes its t_state based on S it generated. *Internal nodes* judge S based on the same incentive policy to change their t_state. If they find that S provides far smaller than the needed computing resources, they broadcast agent change message to

Algorithm 3 Write Operation at the Top Layer

```
1: function WriteOperation()
2:     Generate a block B containing the hash of the previous block, local consensus
   from each participant, etc.
3:     find a nonce that makes B's hash start with D zeros
4:     add B to the CPoW blockchain
5:     broadcast B to both internal and external nodes
6:     t_state ← Listener
7:     execute OnReceiveBlock(B)
8: end function
```

Algorithm 4 Read Operation at the Top Layer

```
1: function OnReceiveBlock(B)
2:     if B is valid then
3:         add B to the CPoW blockchain
4:         if t_state = Miner then
5:             stop WriteOperation()
6:         end if
7:         if ID j ≠ B.time%N_i then
8:             t_state ← Listener
9:         else
10:            t_state ← Agent
11:            if the MBC constraint is violated then
12:                generate miner set S
13:                broadcast S to other internal nodes
14:                execute OnReceiveMinerSet(S)
15:            else
16:                broadcast empty set to other internal nodes
17:                t_state ← Listener
18:            end if
19:        end if
20:    end if
21: end function
22: function OnReceiveMinerSet(S)
23:    if ID j ∈ S then
24:        t_state ← Miner
25:        execute WriteOperation()                    ▷ on new thread
26:    else
27:        t_state ← Listener
28:    end if
29: end function
```

replace the agent node. if a node receives $2 \times F_i$ same messages, it consider the n_i^q with $q = (T+1)\%N_i$ as the new agent node.

In the Algorithm 3, miner nodes find a *nonce* that makes B's hash start with D zeros. D is the difficulty factor of CPoW and it changes periodically. All the nodes in the network calculate the Average Block Interval (ABI) in a cycle and then adjust D based on the difference between the current ABI and the expected value. The expected value of ABI is a hyperparameter set before building the blockchain. ABI is negatively correlated with D, and the change in D always makes next cycle's ABI closer to the expected value. If the ABI is too high, the nodes reduce D in the next cycle, and the rest cases are similar.

Honest nodes only accept valid blocks from participants who do not violate the MBC constraint. A valid block should contain the hash of the previous block and a nonce that satisfies D and not contain any invalid transactions. When a valid block satisfies the CN constraint, the transactions in the block become global consensus and take effect.

3.3 Incentive Policy

There is no cryptocurrency in permissioned blockchain to motivate nodes to perform the Write Operation. To solve this problem, we limit the default amount of transactions from each participant contained in a CPoW block, denoted by Default Size (DS), and provide an additional amount of transactions to participant who generated blocks as a reward, denoted by Reward Size (RS). If P_i generated B_i, an additional RS transactions in each block from B_i to B_{i+CN-1} are allowed from P_i. Therefore, the Block Size (BS) can be expressed as $BS = DS \times K + RS \times CN$.

By setting the appropriate parameters, participants with large data volumes actively perform the Write Operation to obtain RS to meet the demand for writing data. Participants with less data volumes can meet the demand relying on DS without investing too much computing resources.

4 Theoretical Analysis and Comparison

4.1 Communication Costs

Communication costs are related to blockchain's performance and scalability. Lower communication costs can lead to better scalability and performance. We assume a total of K participants and N nodes, and each participant has N/K nodes in our proposed system. For a transaction from the client, the communication costs to reach consensus at the bottom-layer PBFT are $O(N^2/K^2)$. After that, the *primary node* broadcasts the local consensus to *external nodes*, leading to $O(N)$ communication costs. Broadcasting a CPoW block causes $O(N)$ communication costs, but it usually contains multiple transactions, so it can be ignored when considering only one transaction. According to the above analysis, the communication costs of DP-Hybrid are $O(N^2/K^2)$.

4.2 Security

Blockchains need to resist attacks that tamper with valid blocks or write invalid blocks to ensure security. 51% attacks [14] are commonly used attacks that damage the security of PoW-based blockchains. To launch 51% attacks, attackers have to generate a long enough blockchain to replace the original one. For example, an attacker's valid Tx_a was packed in B_a, and it takes effect after the generation of B_{a+CN}, which is the latest block now. If the attacker wants to destroy the record of Tx_a in B_a, he or she has to regenerate blocks B_a to B_{a+CN+1}, i.e., longer than the original one. In the other case, if an attacker wants to pack an invalid Tx_b in the latest CPoW B_b, he or she has to continuously generate blocks B_b to B_{b+CN+1}, because honest nodes refuse to follow B_b containing invalid Tx_b. However, due to the MBC constraint, it is impossible for any participant to continuously generate CN blocks.

To reflect the difficulty of attacks, we define the Minimum number of Attackers $MinA = CN/MBC$, which means that attackers need to control at least

$MinA$ participants to generate CN consecutive blocks. We define the Attack Tolerance Factor $ATF = MinA/K$. A larger ATF means that the blockchain can tolerate more participants collusion.

4.3 Liveness

Liveness means that the blockchain can handle valid transactions. When the number of normal nodes is not enough to continuously generate blocks and extend the blockchain, the blockchain loses its liveness.

Similarly, we define the Minimum number of Crashed nodes $MinC = K - MinA + 1$, meaning that when the nodes of $MinC$ participants are crashed, the remaining participants cannot continue to extend the blockchain. We define the Crash Tolerance Factor $CTF = MinC/K$. A larger CTF means that the blockchain can tolerate more participants' nodes simultaneously crashed.

4.4 Latency

Latency is the time elapsed between the client sending the transaction and the transaction taking effect. In DP-Hybrid, transactions are transformed into local consensus by PBFT, and then packed into CPoW block. When there are CN CPoW blocks generated following B_a, the transactions in B_a become global consensus and take effect. The time required to generate a block is not fixed, but in the long-term operation of the CPoW-based blockchain, ABI is stabilized within a range. We assume that a transaction was submitted at time period $(ABI, 2 \times ABI)$, transformed into global consensus at time point $3 \times ABI$ and took effect at time point $(CN + 3) \times ABI$. The latency is between $(CN + 3) \times ABI - ABI$ and $(CN + 3) \times ABI - 2 \times ABI$ and the average latency is $(CN + 1.5) \times ABI$.

4.5 Comparison

We compared communication costs, security, liveness, latency of DP-Hybrid, PBFT, PoW, committee-based (CB) and leadership-based (LB) consensus protocols. The details are listed in Table 3.

We give a comparative analysis according to the table. DP-Hybrid has lower communication costs than PBFT but usually higher than CB and LB. The typical communication costs of both CB and LB are $O(NC)$ or $O(C^2)$, where C is the number of committee or leader nodes and smaller than N. In terms of security, DP-Hybrid is configurable and can achieve higher security than PBFT. But for CB and LB, the security is sacrificed for lower communication costs. The fault tolerance of CB is $F = \lfloor \frac{C-1}{3} \rfloor$ and LB even has a smaller F, because the leader node directly represents the subsidiary nodes below it. The consensus protocols mentioned above have good liveness, but for some implementations of CB and LB, view changes are often triggered to ensure liveness, which may reduce the actual performance. The latency of DP-Hybrid is stable and higher

Table 3. Comparison of Different Consensus Protocols

	DP-Hybrid	PBFT	CB	LB
Communication costs	$O(N^2/K^2)$	$O(N^2)$	$O(NC)$	$O(C^2)$
Security	configurable	$F < \lfloor \frac{N-1}{3} \rfloor$	low	low
Liveness	configurable	high	high	high
Latency	high	low	low	low

than the remaining consensus protocols. The latency of PBFT, CB and LB is low, but increases rapidly as the number of nodes increases.

Latency is what DP-Hybrid sacrifices for other performance and scalability. We believe that sacrificing latency is better than sacrificing security.

5 Experiments

We have conducted experiments on throughput in permissioned blockchain. The experiments measure normal-case behavior without Byzantine failure to achieve the best performance. The experiments ran on one six-core twelve-thread desktop with a frequency of 3.2Ghz and 8GB RAM. We use Docker to simulate multiple nodes, and the nodes are connected through a physical wireless network. The client is deployed on another computer and sends transactions to the blockchain system over the same wireless network. The client sends transactions to the blockchain system at a sufficient rate, so it does not limit the throughput.

We developed a PBFT blockchain and developed the DP-Hybrid blockchain based on it. Signing and verification are omitted for simplicity. We conducted comparative experiments on the throughput of these two blockchains.

We define the throughput of the blockchain as the number of transactions that can be handled per second (Tx/s). We tested the throughput under a varying number of nodes, including N=4, 7, 10, 16, 28, 40, 52. In DP-Hybrid, K is 4, CN is 6, MBC is 2 and BS is 10240. The experiments for DP-Hybrid were conducted only at N=16, 28, 40, 52 and each participant has 4, 7, 10, 13 nodes.

The throughput of PoW-based blockchain is not a fixed value, because the interval between blocks is different. For simplicity, we simulate that a new CPoW block is generated every 15 second in DP-Hybrid, and thus the throughput is the number of transactions contained in the block divided by 15. As shown in Fig. 4, with an increase in N, the throughput of PBFT decreases rapidly because of the $O(N^2)$ communication costs. DP-Hybrid's throughput also decreases with the increase of N, but always has about 10 times the throughput of PBFT in the experimental environment.

The BS is a factor that affects DP-Hybrid's throughput. We test DP-Hybrid's throughput under a varying BS, including BS=10240, 7168, 5120, 4096, 3072. The remaining parameters are the same as the above experiments. As shown in Fig. 5, BS limits the maximum throughput of DP-Hybrid, which is reflected when the number of nodes is small. Larger BS allows larger CPoW blocks, but transferring these blocks consumes more network bandwidth. According to the

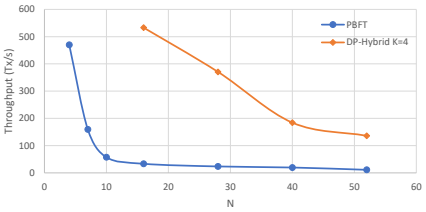

Fig. 4. Throughput of DP-Hybrid and PBFT under a varying N

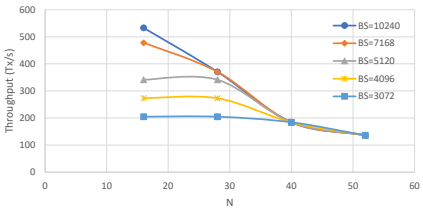

Fig. 5. Throughput of DP-Hybrid under a varying BS

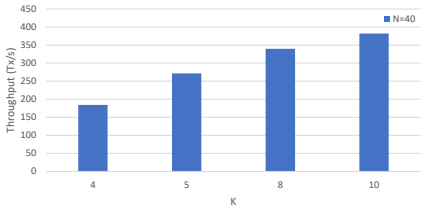

Fig. 6. Throughput of DP-Hybrid under a varying K

incentive policy discussed above, BS is determined by the DS and the RS, so the BS should be configured properly in conjunction with the incentive policy.

The number of participants is also a factor that affects DP-Hybrid's throughput. We test the throughput of different K in the case of N=40, including K=4, 5, 8, 10, and each participant has 10, 8, 5, 4 nodes. We keep the BS at 10240 by changing RS and DS, and the remaining parameters are the same as the above experiments.

Fig. 6 shows that a larger K results in higher throughput when N is fixed. This means that in industry applications, an increase in the number of participants does not degrade performance. Besides, more participants make the incentive policy and security configurations more flexible.

We conclude that DP-Hybrid has much higher throughput than PBFT when the number of nodes is large. With low reconfiguration overhead and configurable incentive policy and security, DP-Hybrid has better scalability for industry applications with many participants.

6 Related Work

As a core part of the blockchain, consensus protocols have been extensively studied. Researchers have proposed optimization on leader election [2], PBFT commit stage [3] and no view-change case [4] to speed up PBFT. The trade-off between the consensus speed and fault tolerance was discussed in [5, 6], which sacrifices fault tolerance for faster consensus speed. Researchers also proposed new consensus protocols to speed up consensus by providing higher voting rights for honest nodes [15, 16], but honest nodes may not remain honest in the future.

Another way to improve performance and scalability is to reduce the amount of message transmission. This is usually achieved through committee-based and leadership-based consensus protocols. Committee-based consensus protocols were proposed in [7–9]. Committee nodes are selected randomly [7, 8] or by a combination of the latest consensus results and the node's authentication information [9]. Committee nodes participate consensus on behalf of other nodes, and the ordinary nodes accept the consensus results obtained by majority committee nodes.

Leadership-based consensus protocols were proposed in [10–13]. In [10], every several nodes form a committee at the bottom layer and select a leader node to participant in the upper layer and then recursively build a hierarchical structure. Consensus protocol is run on the committees and the local consensus are uploaded to the upper level by the leader node and then recursively reached global consensus. In [11–13], consensus protocol only runs on the leader nodes and the result are passed to the lower-layer nodes by the leader node. In contrast to committee-based consensus protocols, lower-layer ordinary nodes directly accept the consensus results obtained by their unique leader node.

However, both committee-based and leadership-based consensus are hard to solve Byzantine faults of committee nodes or leader nodes. In contrast, DP-Hybrid uses PoW-based consensus instead of voting-based consensus as a high-layer consensus, which can imporve scalability without sacrificing security.

7 Conclusion

In this paper, we have studied the consensus protocols in permissioned blockchain. We found that the PBFT consensus protocol which is commonly used in permissioned blockchain leads to poor scalability and high reconfiguration overhead. To solve this problem, we proposed a two-layer consensus protocol called DP-Hybrid that combines PBFT and CPoW. DP-Hybrid reduces both communication costs and reconfiguration overhead and thus improves the scalability, while providing configurable incentive policy and security. We conducted experiments and results show that DP-Hybrid's throughput is always about 10 times that of PBFT when the number of nodes increases.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (No. 61972161 and No. 61902333), and in part by the Fundamental Research Funds for the Central Universities, China (No. 2018MS53).

References

1. M. Castro, B. Liskov, "Practical Byzantine Fault Tolerance", Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI), pp. 173-186, 1999.

2. J. Augustine, G. Pandurangan, P. Robinson, "Fast Byzantine Leader Election in Dynamic Networks," Distributed Computing: 29th International Symposium (DISC), pp. 276-291, 2015.
3. L. He, Z. Hou, "An Improvement of Consensus Fault Tolerant Algorithm Applied to Alliance Chain," 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), pp. 1-4, 2019.
4. Y. Jiang, S. Ding, "A High Performance Consensus Algorithm for Consortium Blockchain," 2018 IEEE 4th International Conference on Computer and Communications (ICCC), pp. 2379-2386, 2018.
5. N. Braud-Santoni, R. Guerraoui, F. Huc, "Fast Byzantine Agreement," In Proceedings of the Annual ACM Symposium on Principles of Distributed Computing, pp. 57-64, 2013.
6. M. M. Jalalzai, C. Busch, "Window Based BFT Blockchain Consensus," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 971-979, 2018.
7. A. Naif, B. Nirupama, "Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain," Proceedings of the first Workshop on Cryptocurrencies and Blockchains for Distributed Systems, (CryBlock'18), pp. 30-35, 2018.
8. M. M. Jalalzai, C. Busch, G. G. Richard, "Proteus: A Scalable BFT Consensus Protocol for Blockchains," 2019 IEEE International Conference on Blockchain (Blockchain), pp. 308-313, 2019.
9. Y. Meng, Z. Cao, D. Qu, "A Committee-Based Byzantine Consensus Protocol for Blockchain," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), pp. 1-6, 2018.
10. G. Chander, P. Deshpande, S. Chakraborty, "A Fault Resilient Consensus Protocol for Large Permissioned Blockchain Networks," 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 33-37, 2019.
11. J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, L. Li, "A Proof-of-Trust Consensus Protocol for Enhancing Accountability in Crowdsourcing Services," IEEE Transactions on Services Computing, vol. 12, no. 3, pp. 429-445, 2019.
12. C. Chen, J. Su, T. Kuo, K. Chen, "MSig-BFT: A Witness-Based Consensus Algorithm for Private Blockchains," 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), pp. 992-997, 2018.
13. K. Li, H. Li, H. Hou, K. Li, Y. Chen, "Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism & Consortium Blockchain," 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), pp. 466-473, 2017.
14. M. Conti, S. Kumar E, C. Lal, S. Ruj, "A Survey on Security and Privacy Issues of Bitcoin," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3416-3452, Fourthquarter 2018.
15. K. Lei, Q. Zhang, L. Xu, Z. Qi, "Reputation-Based Byzantine Fault-Tolerance for Consortium Blockchain," 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), pp. 604-611, 2018.
16. L. Bahri, S. Girdzijauskas, "When trust saves energy: A reference framework for proof of trust (PoT) blockchains," Companion Proceedings of the The Web Conference 2018 (WWW '18), pp. 1165-1169, 2018.