Power-of-Collaboration: A Sustainable Resilient Ledger Built Democratically

Junchao Chen, Suyash Gupta[†], Sajjad Rahnama, Mohammad Sadoghi
Moka Blox LLC
Exploratory Systems Lab
University of California, Davis
[†]University of California, Berkeley

Abstract

*The recent surge in blockchain applications has accelerated the research in designing efficient decentralized currencies. Building a decentralized economy on the traditional byzantine fault-tolerant (*BFT*) protocols or the Proof-of-Work (*PoW*) consensus protocol is inadequate as the immutability of the ledger created by the former is at the mercy of the long-term safe-keeping of private keys of participants, while the latter yields an extremely inefficient and environmentally unsustainable consensus. To ameliorate this situation, we envision the design of our* HybridChain *architecture, which offers the best of both worlds. Our* HybridChain *design runs a traditional* BFT *protocol to commit client transactions and employs our novel Power-of-Collaboration (*PoC*) protocol to notarize the* BFT *chain. Unlike* PoW*, our* PoC *protocol advocates for participants to work together collaboratively instead of competing (often selfishly), which results in a safe, high-throughput, and resource-efficient consensus design.*

1 Introduction

The past decade has observed a surge in the design and deployment of decentralized systems. A key reason for this surge is the growing desire in the society to have self-governing democratic financial systems that are not under the control of a privileged set of entities. A central control often translates to a forced trust model with limited provision to support transparency and accountability. The adoption of Blockchain, for example, is a by-product of the ability to break away from the forced-central control in a trust-worthy fashion [8]. The emerging blockchain platforms facilitate a reliable execution of any digital contracts (i.e., transactions) in a decentralized manner despite the existence of malicious actors. At the core of any blockchain platform is a Byzantine fault-tolerant (BFT) consensus protocol and a tamper-proof replicated ledger [2, 8, 24]. The BFT protocol helps to achieve *consensus* on the order of incoming client requests among all the replicas, while the ledger logs this agreement.

Traditional BFT protocols expect a *permissioned* system where the identities of all the replicas (i.e., participants) are known prior to any consensus as they rely on having a verifiable voting right in a democratic setting. These protocols rely on a *communication-oriented* consensus model, where all the participants exchange endorsements across multiple rounds before they can reach a decision [1, 3, 4, 7, 9, 10, 11, 17, 19, 41, 45]. In

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

these protocols, a system of $\mathbf{n}$ replicas can reach a common decision if at most $\mathbf{f}$ of them are malicious, such that $\mathbf{n} \geq 3\mathbf{f} + 1$. The $\mathbf{n}$ parties are said to reach a decision when at least a majority of honest parties agrees to that decision. This decision is logged by requiring all the agreeing parties to *sign* the decision. Hence, the reached decision is considered *tamper-proof* because it has support of a majority of honest participants.

Despite being around for more than two decades, traditional BFT protocols did not see any major practical applications until the introduction of blockchain technology. We attribute *two* key factors for this lack of adoption. (i) To ensure that the malicious participants do not spawn multiple identities, these BFT protocols need an authority (i.e., *a forced trust gateway*) to verify and register every participant to verify every vote [6]; some participants may find this intrusive if they do not want to reveal their personal information. (ii) To overwrite the ledger, malicious participants just require access to the private keys of honest participants. In a sense, the proof of the validity of the ledger is not self-contained, and it operates on the assumption that the private-keys are kept safe externally indefinitely.

To resolve these challenges, initial blockchain platforms such as Bitcoin [16] and Ethereum [22] offer a *permissionless* model of consensus. These systems employ the *Proof-of-Work* (POW) protocol [16, 22], which follows a *computation-oriented* consensus model and requires all the participants to compete with each other and try to solve a complex puzzle. Whichever participant solves the puzzle first gets to add a new entry (*block*) to the ledger. As a result, POW protocol eliminates the three challenges seen by traditional BFT protocols. (i) Malicious participants can spawn multiple identities, but what actually matters is the available compute power. (ii) Each block includes the hash of the previous block; overwriting the ledger requires recomputing all the blocks making it computationally infeasible. (iii) Since reaching the consensus is based on presenting the proof of work that is embedded on the ledger (i.e., self-contained), there is no longer any need for external safe-keeping of private keys to sign endorsements.

These properties offered by POW protocol help blockchain platforms to design a *decentralized economy*, where any person can participate in the consensus process, and the economy has a self-generating currency to monetize its participants. Monetizing the participants is necessary as the POW protocol expects the participants to spend their resources to solve a complex puzzle. Clients of the Bitcoin platform, create transactions that exchange Bitcoins and send them to the participants (miners) in the POW protocol. These miners check if the transaction is valid; the client has sufficient Bitcoins to transfer. If the transaction is valid, they run POW protocol to include this transaction in the ledger. The winning miner of POW gets a portion of the client's Bitcoin as *fees*, while the mining process (POW) mints new tokens to fund the economy. This new token is transferred to the winning miner's account and is recorded as a transaction in the block.

The key challenge with platforms like Bitcoin is their *practicality*. These platforms have abysmally low throughput in the order of 10 transactions per second in part due to inadequate choice of small block sizes. Furthermore, as more miners join the network, the complexity of the puzzle has to be increased. For example, the complexity of the current Bitcoin puzzle is so high that the miners work in large groups to have any positive probability of creating the next winning block [8]. Moreover, as miners are competing with each other, it leads to massive wastage of computational resources (energy) as only the winning miner's efforts are recorded and rewarded. This results in an unsustainable ecosystem [5, 21].

We observe these challenges in the designs of existing BFT protocols and blockchain platforms and envision a HYBRIDCHAIN system that learns from these models and eliminates their key challenges. Essentially, we aim to establish a new research agenda; a new field of hybrid consensus protocols that depart from competitive consensus to a collaborative consensus that is both resilient and sustainable. Our HYBRIDCHAIN architecture takes a step in this direction by running two consensuses on each client transaction while ensuring there is no increase in the latency observed by the client. Each client request is first ordered through a state-of-the-art BFT consensus protocol (*commitment*), subsequently, this ordered request is engraved into the ledger through the POW-style consensus (*settlement*). Specifically, HYBRIDCHAIN causes no increase in commitment latency while improving the settlement latency observed by existing protocols. Ordering the client transaction through a BFT consensus protocol first allows our HYBRIDCHAIN system to guarantee the following benefits: (i) clients receive

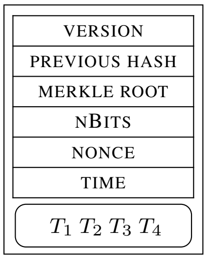

Figure 1: A Bitcoin-style block containing a header and a body with transactions (T_1, T_2, T_3 and T_4).

low-latency responses, and (ii) PoW participants no longer need to compete, resulting in a high-throughput sustainable chain. As a result, instead of employing the PoW for consensus, we design a novel protocol that allows miners to collaborate. We refer to this paradigm as *Power-of-Collaboration* (PoC).

Our PoC protocol splits the complex puzzle into disjoint slices and requires each miner to work on a distinct slice. This slice distribution significantly reduces the resource wastage and provides each honest miner with a reward for each new transaction added to the ledger. As each ledger entry is added collaboratively, any malicious entity that wishes to overwrite the ledger needs to match the computational power of all the existing miners making it practically impossible. These features of our HybridChain system make it lucrative; its design is the bedrock for a secure and efficient decentralized economy.

2 Preliminaries

We adopt the standard communication and failure model adopted by most BFT protocols [7, 3, 10]. We consider a service $\mathcal{S}$ of the form $\mathcal{S} = \{\mathcal{R}, \mathcal{M}, \mathcal{C}\}$. The set $\mathcal{R}$ consists of $\mathbf{n}_{\mathcal{R}}$ replicas of which at most $\mathbf{f}_{\mathcal{R}}$ can behave arbitrarily. The remaining $\mathbf{n}_{\mathcal{R}} - \mathbf{f}_{\mathcal{R}}$ are honest: they will follow the protocol and remain live. Similarly, the set $\mathcal{M}$ consists of $\mathbf{n}_{\mathcal{M}}$ miners of which at most $\mathbf{f}_{\mathcal{M}}$ can act maliciously. We assign each *anonymous miner* and replica a unique identifier, which can be obtained by a call to the function $\mathsf{id}()$. The range of these identifiers are $[0, |\mathcal{R}|]$ for replicas and $[0, |\mathcal{M}|]$ for miners. We further consider the existence of a finite set of clients $\mathcal{C}$ of which arbitrarily many can be malicious.

We assume *authenticated communication*: replicas employ standard cryptographic primitives such as MAC and digital signatures (DS) to sign messages. We denote a message m signed by a replica R using DS as $\langle m \rangle_{\mathbf{R}}$. We permit malicious replicas to impersonate each other but no entity can impersonate an honest replica. We assume that like Bitcoin miners are identifier through their public-keys (i.e., anonymous), while the identities of replicas is known before consensus (i.e., known verified identity). We employ a *collision-resistant* hash function $\texttt{hash}(\cdot)$ to map an arbitrary value v to a constant-sized digest $\texttt{hash}(v)$ [15]. Each replica only accepts a message if it is ***well-formed***.

We adopt the same partial synchrony model adopted in most consensus systems: safety is guaranteed in an asynchronous environment where messages can get lost, delayed, or duplicated. Liveness, however, is only guaranteed during the periods of synchrony [3, 7, 9, 10, 45].

Safety. If two honest replicas R1 and R2 order a transaction T at sequence numbers k and k', then $k = k'$.

Liveness. If a client sends a transaction T, then it will eventually receive a response for T.

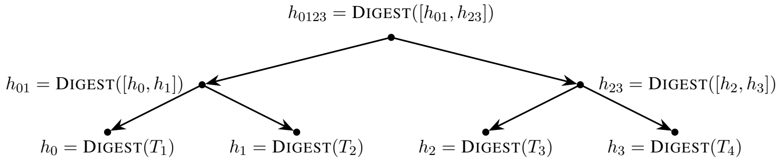
$h_{0123} = \textsc{Digest}([h_{01}, h_{23}])$
$h_{01} = \textsc{Digest}([h_0, h_1])$
$h_{23} = \textsc{Digest}([h_2, h_3])$
$h_0 = \textsc{Digest}(T_1)$
$h_1 = \textsc{Digest}(T_2)$
$h_2 = \textsc{Digest}(T_3)$
$h_3 = \textsc{Digest}(T_4)$

Figure 2: A Merkle tree over four transactions (T_1, T_2, T_3 and T_4) stored at the leaf nodes of the tree.

3 Background: Proof-of-Work Consensus

The *settlement* phase in our HYBRIDCHAIN architecture makes use of our novel POC protocol. The intuition behind POC's design stems from the POW protocol that helps create an immutable *chain of blocks*. The term immutable refers to the fact that each block appended to the chain requires miners to spend their resources. As a result, if an adversary attempts to over-write (or rollback) a part of the chain, it needs to create an alternate chain of all the desired blocks. However, to force all the honest miners to switch to this alternate chain, the adversary needs to compute this chain of blocks at a much faster rate than the original chain. This implies that the adversary needs to have much greater power than the honest miners. Prior works have illustrated that such an attack is hard to realize [12].

Prior to running the POW protocol, each miner $\textsc{m} \in \mathcal{M}$ needs to create a *block* of transactions. Although each miner $\textsc{m}$ has access to the certificate $\mathfrak{C}$, it needs to arrange the contents of this certificate in the format of a block. To explain the format of a block, we follow the popular blockchain platform, Bitcoin, where each block includes a header and body (refer to Figure 1) [8]. The header includes: (i) *version* for this block, (ii) *hash* of the previous block, (iii) *merkle root* of all transactions, (iv) *nBits*, which determines the difficulty of the puzzle, (v) *nonce*, the solution for puzzle, and (vi) *time* at which block is created once the nonce is found.

Computing Merkle root of all the transactions is trivial (refer to Figure 2) and requires a miner $\textsc{m}$ to compute a pairwise hash from the leaf to the root. This Merkle root helps to verify if a transaction was included to create the Merkle tree. However, the main challenge for a miner is to determine the nonce. In existing POW-based platforms, to solve the complex puzzle, each miner needs to calculate the *hash of the block* such that it contains a specific number of leading zero bits. For this purpose, the miners have to find a *nonce* value that yields the specific hash. This essentially makes the POW protocol like a *race* where all the miners are competing against each other to find the nonce. Whichever miner finds a valid nonce first, it gets the chance to propose the next block to be added to the chain. Hence, coming up with the correct number of leading zero bits in the hash is important as it sets the difficulty of the puzzle which simplicity controls the average time taken for each winning miner to propose a block. Once a miner finds the valid nonce, it can fill all the entries in the header, and it broadcasts the new block to all the other miners.

Notice that the ledger is essentially a chain of blocks (refer to Figure 3). In POW, it is possible that multiple miners propose the next block with valid nonces at approximately the same period of time. In such a case, the protocol states that each miner would only accept the first block it receives. This could lead to temporary branches or *forks*, all of which have the same previous hash. However, this condition resolves as time goes by because the protocol also expects the honest miners to stick to the *longest chain*—the one with the largest number of blocks. Eventually, all the shorter forks are discarded, and only the longest chain survives.

Incentives. Considering discarded forks and resources spent in the search for a valid nonce, what motivates a miner to participate in POW consensus? The answer is incentives. POW-based systems like Bitcoin reward the winning miner of the block present in the longest chain. These rewards help to offset the mining costs and maintain a sufficient number of honest miners. This requires a few more entries in the block header, which provide information such as the miner's account address and the reward amount.

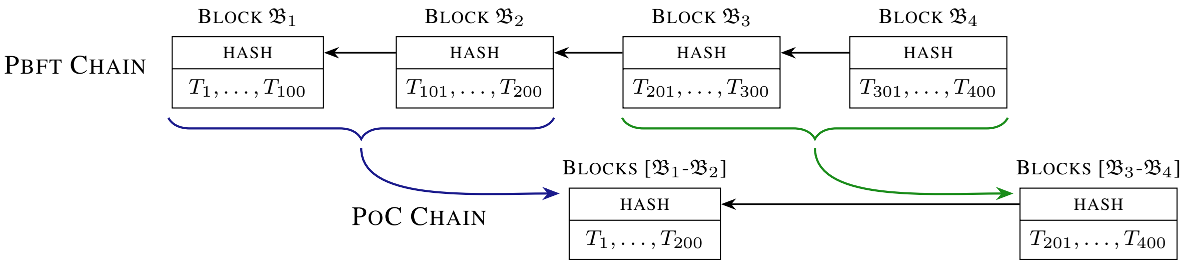
BLOCK $\mathfrak{B}_1$
HASH
$T_1, \ldots, T_{100}$
BLOCK $\mathfrak{B}_2$
HASH
$T_{101}, \ldots, T_{200}$
BLOCK $\mathfrak{B}_3$
HASH
$T_{201}, \ldots, T_{300}$
BLOCK $\mathfrak{B}_4$
HASH
$T_{301}, \ldots, T_{400}$
PBFT CHAIN
POC CHAIN
BLOCKS [$\mathfrak{B}_1$-$\mathfrak{B}_2$]
HASH
$T_1, \ldots, T_{200}$
BLOCKS [$\mathfrak{B}_3$-$\mathfrak{B}_4$]
HASH
$T_{201}, \ldots, T_{400}$

Figure 3: A schematic representation of a blockchain or ledger in the HYBRIDCHAIN architecture that consists of PBFT chain (i.e., *layer 1*) and POC chain (i.e., *layer 2*) that shows committed and settled transactions $T_1, \ldots, T_{400}$ on the PBFT and POC chains, respectively. The i^{th} block holds a *hash value* hash_{i-1} that identifies the preceding block and Block $\mathfrak{B}_1$ is the genesis block that does not have the preceding block.

3.1 POW Challenges

The key issue with running the POW consensus is that it leads to massive waste in energy and efforts: (1) Forks of the longest chain are subsequently discarded, which is a loss of resources for some miners who did find a valid nonce but did not receive any rewards. (2) Rational miners may acquire more resources to improve their chances of proposing the next block, but this leads to increasing the difficulty of hash computation to ensure fairness. (3) As the number of miners increases, the frequency of a single miner winning rewards decreases. (4) Several miners may work together in groups to find the valid nonce to increase their probability of winning rewards. This behavior significantly decreases the probability for a lone miner to propose the next block. (5) Malicious miners may attempt to perform attacks like selfish mining and double-spending, which can rollback client transactions and invalidate the rewards earned by honest miners [8].

These issues are so prevalent in blockchain systems like Bitcoin that, at present, almost every miner is trying to join some existing group. In these groups, miners *pool* their resources to find a valid nonce and propose the next block [18]. Every pool has its own participation rules and distributes rewards according to its policies. Despite this, the difficulty of hash computation is periodically increased or decreased in accordance with the average time to find a nonce by a pool of miners.

To address these challenges, in this paper, we aim to initiate a new avenue of research centered around hybrid consensus and collaborative mining. In particular, as a first step, we propose our novel *Power-of-Collaboration* (POC) protocol that re-imagines the POW consensus.

4 HYBRIDCHAIN Architecture

Our HYBRIDCHAIN runs two distinct consensus protocols in parallel. Specifically, it requires the replicas in set $\mathcal{R}$ to commit each client transaction through a BFT consensus protocol (commitment phase), following which the miners in set $\mathcal{M}$ run our POC consensus (settlement protocol). As all the BFT protocols follow the consensus dictated by PBFT [3], we use PBFT as the representative protocol for the ensuing discussions. *To summarize:* each client sends its request to the replicas running the PBFT protocol. Once these replicas commit this request, they forward it to the miners. These miners collaboratively run the POC consensus protocol, post which they add a block to the ledger. Next, we discuss each of these steps in detail.

4.1 Client Request and Transaction Ordering

PBFT follows the primary-backup model where one replica is designated as the *primary* while other replicas act as backups. Each consensus is led by the primary replica of the current *view*. In the case the primary is malicious, *view-change* takes place to replace the primary. We use Figure 4 to illustrate the three phases of PBFT.

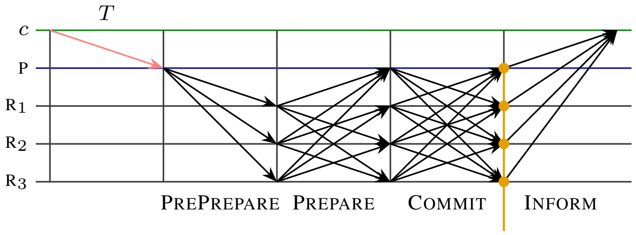

Figure 4: A schematic representation of the normal-case of the PBFT protocol with $\mathbf{n}_{\mathcal{R}} = 4$ and $\mathbf{f}_{\mathcal{R}} = 1$.

Client Request. A client c that wants to process a transaction T in our HYBRIDCHAIN architecture creates a request $\langle T \rangle_c$ and sends it to the replica designated as the primary of the view v. The client c uses DS to sign this message and adds a monotonically increasing timestamp to this message.

Pre-prepare. When the primary P replica receives a well-formed client request $m := \langle T \rangle_c$, it assigns m a sequence number k and creates and sends a PREPREPARE message to all the replicas. This PREPREPARE message also includes a digest $\texttt{hash}(m)$ of m, which is used in future communication to save space. During this phase, it is sufficient for the primary to sign the messages using MAC. When a replica $\text{R} \in \mathcal{R}$ receives a well-formed PREPREPARE message from the primary P of view v, it agrees to support the order k for m if it has not agreed to order another request at sequence number k. The replica R shows its support by broadcasting a PREPARE message.

Prepare. When a node R receives identical PREPARE messages from $2\mathbf{f}_{\mathcal{R}} + 1$ distinct replicas (can include its own message to reach the count), it marks the request m as *prepared* and broadcasts a COMMIT message. In HYBRIDCHAIN, we require each replica R to use DS to sign the COMMIT message.

Commit. When R receives identical COMMIT messages from $2\mathbf{f}_{\mathcal{R}} + 1$ replicas, it marks m as *committed*. If R has executed all requests with sequence number less than k, it executes m and sends a RESPONSE message to the client, which includes the result of execution r. The client c marks $\langle T \rangle_c$ as processed when it receives identical RESPONSE messages from at least $\mathbf{f}_{\mathcal{R}} + 1$ replicas.

Chain Communication. Post consensus on m, each replica R creates a certificate $\mathfrak{C}$, which includes: (i) the client request m, (ii) COMMIT messages for m from $2\mathbf{f}_{\mathcal{R}} + 1$ replicas, and (iii) the result r. Next, the replicas may follow the *cluster-sending* protocol [13] or delayed replication protocol [14] to communicate with the miners in set $\mathcal{M}$.[1] The cluster-sending protocol guarantees the delivery of at least one message between the two clusters, given that less than one-third of members of each cluster are malicious. To do so, each member from the sending cluster sends a message to a distinct member in the receiving member. In our case, we need the certificate $\mathfrak{C}$ to be sent to at least $2\mathbf{f}_{\mathcal{M}} + 1$ miners: replica R1 sends $\mathfrak{C}$ to miner M_1; replica R2 sends $\mathfrak{C}$ to miner M_2, and so on.

When a miner $\text{M} \in \mathcal{M}$ receives a certificate $\mathfrak{C}$ from a replica $\text{R} \in \mathcal{R}$, it broadcasts this certificate to all the other miners. As at least $2\mathbf{f}_{\mathcal{M}} + 1$ miners receive certificates, there is a guarantee that at least one honest miner will broadcast the certificate. As a result, each miner will have access to the certificate $\mathfrak{C}$, and it can proceed with the POC computation.

4.2 Collaborative Mining

Post PBFT consensus, our HYBRIDCHAIN system runs the POC protocol on the agreed transaction to securely bind it to the ledger. POC requires all the miners to *collaborate* and work together to compute the required hash. To do so, POC divides the POW hash computation into $\mathbf{n}_{\mathcal{M}}$ disjoint subproblems and requires each miner to work on a distinct predetermined subproblem.

[1] We can model chain communication as either push- or pull-based model using existing peer-to-peer communication primitives.

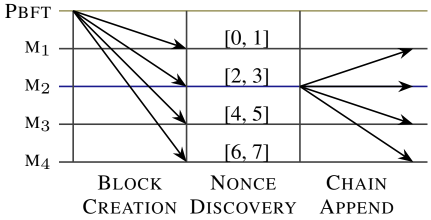

Figure 5: A schematic representation of the PoC protocol with $\mathcal{M} = \{\text{M}_1, \text{M}_2, \text{M}_3, \text{M}_4\}$. The total solution space $\mathcal{S}$ is $[0, 7]$ and is divided into four slices ($[0, 1], [2, 3], [4, 5], [6, 7]$). Miners receive transactions from the PBFT replicas. Post creating a block, each Miner M_i, $i \in [1, 4]$ tries to discover the nonce in its slice $\mathcal{S}_i$. Assume the valid nonce is 2, then once M_2 discovers the nonce, it broadcasts the same to other miners.

Let, $\mathcal{S}$ represent the solution space; all the random numbers that a miner has to try to find a valid nonce. Without the loss of generality, let us divide $\mathcal{S}$ into $\mathbf{n}_{\mathcal{M}}$ equal slices, $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_{\mathbf{n}_{\mathcal{M}}}\}$, such that

$$\mathcal{S}_1 \cap \mathcal{S}_2 \cap \cdots \cap \mathcal{S}_{\mathbf{n}_{\mathcal{M}}} = \varnothing \tag{1}$$

$$\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{\mathbf{n}_{\mathcal{M}}} = \mathcal{S} \tag{2}$$

Our PoC protocol assigns slice $\mathcal{S}_1$ to miner M_1, $\mathcal{S}_2$ to M_2, and $\mathcal{S}_i$ to M_i, $i \in [1, \mathbf{n}_{\mathcal{M}}]$. The key assumption is that if a miner takes time τ to find a valid nonce on the solution space $\mathcal{S}$, then if all the miners are honest and follow the PoC protocol, the time required to find the nonce should be of the order $\mathcal{O}(\frac{\tau}{\mathbf{n}_{\mathcal{M}}})$.

Further, PoC protocol ensures that each honest miner gets rewarded for their efforts; rewards are distributed among miners. If $\Diamond$ is the reward for a miner to find a valid nonce in PoW protocol, then in our PoC protocol, each i^{th} miner M_i receives a reward $\Diamond_i$ proportional to the size of its slice $\mathcal{S}_i$.

$$\Diamond_i = \frac{|\mathcal{S}_i|}{|\mathcal{S}|} * \Diamond \tag{3}$$

In the rest of this paper, for simplicity, we assume that all the slices have the same size.

4.2.1 PoC Protocol

Our PoC protocol works in rounds, and within each round each miner tries to find if a valid nonce exists in its slice of the block. In the rest of this section, we assume that the solution space $\mathcal{S}$ can be deterministically divided into $\mathbf{n}_{\mathcal{M}}$ disjoint equal slices by each miner. For example, in Figure 5, the solution space $\mathcal{S} = [0, 7]$ is divided into $\mathbf{n}_{\mathcal{M}} = 4$ slices; the slices are: $\mathcal{S}_1 = [0, 1]$, $\mathcal{S}_2 = [2, 3]$, $\mathcal{S}_3 = [4, 5]$, and $\mathcal{S}_4 = [6, 7]$. Designing optimal slice distribution schemes is an interesting research avenue, which we consider outside the scope of this work.

Certificate Dissemination. The PoC protocol starts when a miner M receives a certificate $\mathfrak{C}$ from a replica. The miner M checks if $\mathfrak{C}$ is well-formed and $\mathfrak{C}$ includes signatures from $2\mathbf{f}_{\mathcal{R}} + 1$ replicas; a proof that these replicas agreed to sequence this batch of transactions at a sequence number k. If this is the case, M broadcasts this certificate to other miners. Note: although while explaining PBFT we considered consensus on a single transaction, it can be trivially extended to a batch of transactions. This batching optimization is employed by all the existing BFT protocols to increase their throughputs [3, 7, 9].

Block Creation. When a miner M has the nonce for the block ordered at sequence number $k - 1$, it initiates the creation of a block at sequence k. It does so by generating a Merkle root of all the transactions in k^{th} batch and a new block header. As each miner M knows there are a total of $\mathbf{n}_{\mathcal{M}}$ miners, it creates $\mathbf{n}_{\mathcal{M}}$ slices and assigns itself the i^{th} slice $\mathcal{S}_i$ in round 0, where $i = \mathsf{id}(\text{M})$.

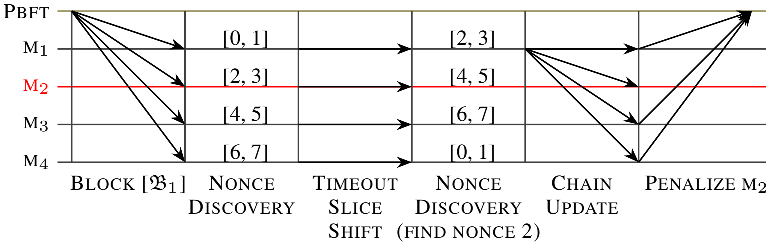

Figure 6: An illustration of the slice shifting procedure. Here, we assume 2 is the valid nonce of the block, and the miner M$_2$ is malicious. Hence, M$_2$ does not broadcast the block to other miners, which triggers slice shifting procedure. Post slice shifting, M$_1$ discovers the nonce and broadcasts to other miners.

Nonce Discovery. We assume that each miner M knows the characteristics of the expected hash (the number of leading zeroes). The miner uses this information to go over all the possible nonces in its slice range to find a valid nonce. Once a miner M computes the correct hash, it has access to a valid nonce. The miner M uses this information to complete the block header and forwards the block to all the miners.

Chain Append. When a miner receives a block from another miner, it first validates the nonce. If the nonce is valid, the miner appends this block to its local blockchain and assumes the PoC protocol for the corresponding batch as complete. Notice that if all the miners are well-behaving, then our PoC protocol requires only one round to find the valid nonce as the nonce is present in one of the slices. Post discovering the nonce, each miner starts working on the next block to be added to the chain.

4.2.2 Slice Shifting Protocol

In our HybridChain system, each miner receives certificates from the PBFT replicas. These certificates include client transactions that have been ordered by at least $2\mathbf{f}_{\mathcal{R}} + 1$ replicas. Our HybridChain architecture uses the PoC protocol to add these transactions to the ledger in the order defined by PBFT replicas. As a result, a malicious miner has limited attack opportunities; if a malicious miner finds a valid nonce in its slice, it can avoid forwarding this information to the honest miners. If such is the case, despite searching over its slice, each honest miner would not find any possible solution and would not be able to make progress.

To resolve this attack, our PoC protocol requires each miner to set a *timer* δ. Each miner M starts a timer δ when it receives a certificate from the PBFT replicas. M stops δ if it discovers the valid nonce or it receives a valid block from another miner. If M's timer δ expires, and it does not have access to the valid nonce, it initiates the *slice shifting* protocol. Once the slice shifting is endorsed by the majority of miners, then each miner searches for the nonce in the next slice. Specifically, if prior to slice shifting a miner M was working on the i^{th} slice $\mathcal{S}_i$, post slice shifting M will work on $((i+1) \mod \mathbf{n}_{\mathcal{M}})^{th}$ slice, $\mathcal{S}_{(i+1)}$. As each miner already has access to all the slices, this switch does not require any additional communication.

The key intuition behind the slice shifting procedure is that even if a malicious miner decides to hide the nonce, post switch, it will be discovered by another miner. However, it is possible that up to $\mathbf{f}_{\mathcal{M}}$ consecutive miners may be malicious. As a result, the honest miners will discover the valid nonce after $\mathbf{f}_{\mathcal{M}}$ shifts.[2]

4.2.3 Reward and Penalty Economy

Frequent slice shifting due to malicious miners will be detrimental to the performance of our PoC protocol; it forces honest miners to do more work and wastes their resources. Moreover, why would any rational miner want

[2] The search space can be salted deterministically upon shifting to expand the search space and guard against rare cases in which the original problem may have no solution irrespective of minors' behavior.

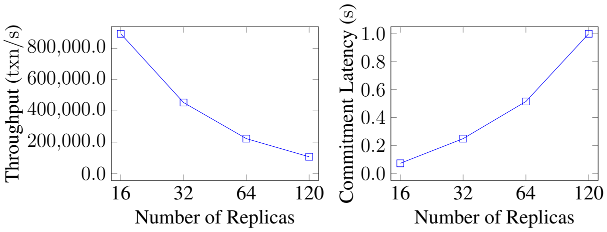

Figure 7: Evaluating peak throughput and commitment latency attained by PBFT consensus in HybridChain.

to join the PoC network and invest its computational resources? To make PoC protocol fruitful, we incentivize all the honest miners for their efforts; all miners are assumed honest until proven guilty.

First, like existing blockchain systems, such as Bitcoin and Ethereum, one of the aims of our HybridChain system is to establish a decentralized economy. To do so, like Bitcoin, in PoC, when a miner discovers a nonce and broadcasts the valid block to other miners, we assume the creation of a *new token*. For brevity, we skip diving into the crypto-economics of the token generation and disbursement, and refer to the existing literature on the same [16, 22]. However, the key goal is that this token is equally divided among the honest miners. Further, like Bitcoin and Ethereum, we expect each client to pay some fees for getting its transaction processed by our HybridChain system. This fees is also equally divided among all the miners working on the current block.

To disburse transaction fees and tokens among the miners, there are two possible approaches: (i) Like Bitcoin, each miner includes $\mathbf{n}_{\mathcal{M}}$ transactions that assigns an equal fraction of the reward to every other miner's public-key (account). These transactions can be deterministically created by each miner prior to mining and are included while creating the Merkle root. However, this will create unnecessary book-keeping, increasing the size of blocks. (ii) We assume that the genesis block of the ledger records the information about the founding miners and their respective initial slices. Further, miners can redistribute, resale, and divide their slices to other miners (similar to buying and selling of stocks), and any such transactions must be stored on the ledger before becoming effective. Assume that when a miner purchases a slice, sufficient tokens are reserved to enable penalization of misbehaving miners, which results in slashing their reserve funds similar to Proof-of-Stake designs [8]. Given all this information, when a block is formed by the PoC miners, we add the incentives to the accounts of the respective miners; miners can validate if they received incentives or not.

This rewarding process of PoC is similar to strategies adopted by *mining pools* in systems like Bitcoin. Most importantly, in PoC, the agreement on what to be included in the next block is strictly determined by PBFT chain, not miners. This substantially simplifies the design of PoC by making it deterministic, eliminating any lottery-based or leader-less consensus challenges that traditional PoW must cope with. Furthermore, we present the novel idea of slice shifting for the cases when no miner in a round broadcasts a valid nonce. As slice shifting requires each miner to work on the next slice to find the nonce, it is expensive. We mitigate the need for slice shifting by heavily *penalizing* malicious miners. Specifically, we require each miner to count the number of *shifts* it took to find a valid nonce and to identify the miner who failed to find the valid nonce. Further, as the order of all initial slice assignments is known to all the miners, so each miner can trivially determine which miner was responsible for previous shifts. Notice that any misbehaving miner will be discovered and penalized as it diminishes the returns for other honest miners.

5 Proof-of-Concept Evaluation

We now present an initial evaluation of our vision of HybridChain architecture, which we implement in the open-sourced ResilientDB fabric [7, 10, 9, 17].[3]

[3] For our proof-of-concept of HybridChain architecture, we employ an experimental version of ResilientDB with the codename of *NexRes* (Next Generation ResilientDB); this is an architectural rewrite of the ResilientDB 3.0 (the latest stable version).

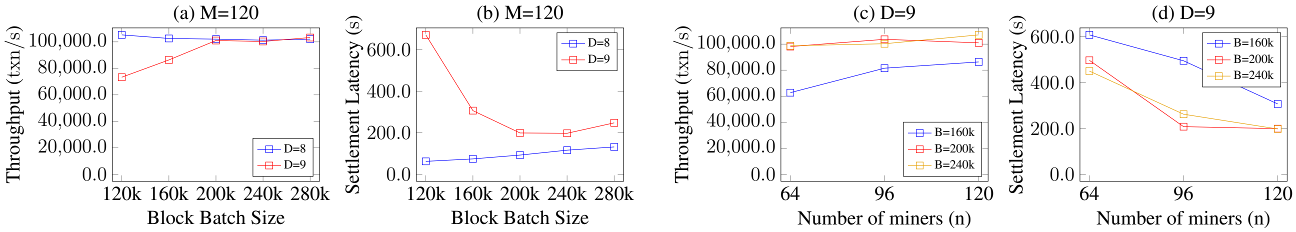

Figure 8: Evaluating PoC throughput and average settlement latency with different difficulty (D), miners (M) and block batch size (B).

Experimental Setup. We use Oracle Cloud Infrastructure's VM.Standard2.8 architecture to deploy miners and replicas (16 cores, 8.2 Gbps bandwidth, 120 GB Memory). In our experiments, we generate 400 million client requests of size 64 B each, while client response has size 17 B. We average results over three runs. We require clients to sign their messages using ED25519 while replicas use CMAC.

Batching. We employ the standard practice of batching client transactions, which we refer to *transaction batching*, to optimize the PBFT consensus. Additionally, during the PoC consensus, each miner aggregates multiple batches from PBFT consensus, ' which we refer to *block batching*, prior to mining.

PBFT Scaling. In Figure 7, we gauge the peak throughput and *commitment latency* incurred by the PBFT consensus of our HybridChain architecture. This is an important metric as it informs the rate at which we can reply to the clients. For this experiment, we increase the number of replicas from 16 to 120 and require the primary to process a batch of 100 transactions per consensus; transaction batch size is set to 100. As expected, on increasing the number of replicas, there is a drop in peak throughput and an increase in incurred latency, which remains in the subsecond range. This phenomenon occurs because, at each setting, we are approximately hitting the network bandwidth; as the number of replicas increases, more messages are communicated per consensus. In summary, the peak throughput reaches well over 800 k transactions/second at 16 replicas while sustaining over 100 k transactions/second even when scaling to as many as 120 replicas.

PoC Scaling. Next, we study the impact of our PoC consensus protocol on the HybridChain architecture. We deploy 120 replicas for running PBFT consensus. For the PoC setting, we set the solution space parameter to 42 nonce bits and split it into equal disjoint slices based on the number of miners. Notice that the difficulty of each problem is the number of leading zeros in the hash. In Figure 8, we present our results; here D refers to difficulty (with the default of $D = 9$), M refers to the number of miners (with the default of $M = 120$), and B refers to the number of batches in a block. As stated earlier, for every B blocks produced by PBFT a single block is notarized and minted by PoC.

In Figures 8(a) and 8(b), we fix the number of miners to 120, which allows creating 120 equal slices, and increase the block size from 120 k to 280 k. We test at two difficulty levels: $D = 8$ and $D = 9$. Our results indicate that $D = 8$ is relatively easy, due to which each miner has to perform a smaller amount of work. As a result, any increase in batch size does not increase peak throughput. Hence, we test on $D = 9$ at which mining smaller batch size impacts the system throughput as miners have to participate in a larger number of consensus rounds. On further increasing the block size, we observe that the throughput hits the PBFT's peak as desired. Thus, notarizing the blocks by PoC no longer hinders the system throughput, it only prolongs the *settlement latency* as expected. When examining the end-to-end system throughput of HybridChain which includes both PBFT and PoC, we observe a sustained throughput of 105 k with a commitment latency of 1 second and the settlement latency of 198 seconds when the batch size is set to 200 k. *Note:* We could not test at difficulty beyond $D = 9$ as each nonce computation became prohibitively time- and resource-intensive given our available commodity hardware.

In Figures 8(c) and 8(d), we gauge the performance of PoC when there are 64 to 120 miners. For these experiments, we also test at three different block batch sizes. As expected, the degree of collaborative mining is directly proportional to the number of miners. As we increase the number of miners, there is an increase in peak throughput and a decrease in the settlement latency.

6 Conclusions

In this paper, we present the vision of our HYBRIDCHAIN system, which facilitates the creation of a safe and efficient decentralized economy. HYBRIDCHAIN achieves these guarantees by separating the life-cycle of a client transaction into two phases: commitment and settlement. In the commitment phase, HYBRIDCHAIN employs a traditional BFT protocol to order the client transactions. Post this, HYBRIDCHAIN requires a set of miners to run the settlement phase, where they notarize the ordered client transactions. Our HYBRIDCHAIN architecture does not impact the transaction latency observed by the client as each client receives the commitment response post BFT consensus. Further, to efficiently notarize transactions during the settlement phase, our HYBRIDCHAIN architecture introduces the notion of collaborative mining, where participants work together instead of competing with each other. These notarized transactions are written to a ledger and can be queried in the future.

7 Acknowledgments

This work was supported in part by (1) *Oracle Cloud Credits* and related resources provided by the Oracle for Research program and (2) the *NSF STTR* under *Award Number* 2112345 provided to *Moka Blox LLC*.

References

[1] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. *SharPer: Sharding Permissioned Blockchains Over Network Clusters*, page 76–88. Association for Computing Machinery, 2021.

[2] Mohammad Javad Amiri, Chenyuan Wu, Divyakant Agrawal, Amr El Abbadi, Boon Thau Loo, and Mohammad Sadoghi. The bedrock of BFT: A unified platform for BFT protocol design and implementation. *CoRR*, abs/2205.04534, 2022.

[3] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[4] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, pages 123–140. ACM, 2019.

[5] Alex de Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.

[6] John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[7] Suyash Gupta, Jelle Hellings, Sajjad Rahnama, and Mohammad Sadoghi. Proof-of-Execution: Reaching consensus through fault-tolerant speculation. In *Proceedings of the 24th International Conference on Extending Database Technology*, 2021.

[8] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. *Fault-Tolerant Distributed Transactions on Blockchain*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2021.

[9] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. RCC: resilient concurrent consensus for high-throughput secure transaction processing. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 1392–1403. IEEE, 2021.

[10] Suyash Gupta, Sajjad Rahnama, Jelle Hellings, and Mohammad Sadoghi. ResilientDB: Global scale resilient blockchain fabric. *Proc. VLDB Endow.*, 13(6):868–883, 2020.

[11] Suyash Gupta, Sajjad Rahnama, Shubham Pandey, Natacha Crooks, and Mohammad Sadoghi. Dissecting BFT consensus: In trusted components we trust! *CoRR*, abs/2202.01354, 2022.

[12] Suyash Gupta and Mohammad Sadoghi. Blockchain transaction processing. In *Encyclopedia of Big Data Technologies*, pages 1–11. Springer, 2019.

[13] Jelle Hellings and Mohammad Sadoghi. Brief announcement: The fault-tolerant cluster-sending problem. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019*, volume 146 of *LIPIcs*, pages 45:1–45:3, 2019.

[14] Jelle Hellings and Mohammad Sadoghi. Coordination-free byzantine replication with minimal communication costs. In *23rd International Conference on Database Theory, ICDT*, pages 17:1–17:20, 2020.

[15] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2nd edition, 2014.

[16] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[17] Sajjad Rahnama, Suyash Gupta, Rohan Sogani, Dhruv Krishnan, and Mohammad Sadoghi. RingBFT: Resilient Consensus over Sharded Ring Topology. In *Proceedings of the 25th International Conference on Extending Database Technology*, pages 2:298–2:311. OpenProceedings.org, 2022.

[18] Meni Rosenfeld. Analysis of Bitcoin pooled mining reward systems, 2011.

[19] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolic. State machine replication scalability made simple. In Yérom-David Bromberg, Anne-Marie Kermarrec, and Christos Kozyrakis, editors, *EuroSys '22: Seventeenth European Conference on Computer Systems*, pages 17–33. ACM, 2022.

[20] Florian Suri-Payer, Matthew Burke, Zheng Wang, Yunhao Zhang, Lorenzo Alvisi, and Natacha Crooks. Basil: Breaking up bft with acid (transactions). In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP '21, page 1–17. Association for Computing Machinery, 2021.

[21] Harald Vranken. Sustainability of bitcoin and blockchains. *Current Opinion in Environmental Sustainability*, 28:1–9, 2017.

[22] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. 2015.

[23] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 347–356. ACM, 2019.

[24] Kaiwen Zhang and Hans-Arno Jacobsen. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1337–1346, 2018.