Multichain-MWPoW: A $p/2$ Adversary Power Resistant Blockchain Sharding Approach to a Decentralised Autonomous Organisation Architecture

Yibin Xu, Yangyu Huang, Jianhua Shao and George Theodorakopoulos

Cardiff University, United Kingdom

Abstract

Blockchain Sharding is a blockchain performance enhancement approach. By splitting a blockchain into several parallel-run committees (shards), it helps increase transaction throughput, reduce resources required, and increase reward expectation for participants. Recently, several flexible sharding methods that can tolerate up to $n/2$ Byzantine nodes ($n/2$ security level) have been proposed. However, these methods suffer from two main drawbacks. First, in a non-sharding blockchain, nodes can have different weight (power or stake) to create a consensus. So an adversary needs to control half of the overall weight of the system in order for a piece of faulty information to be accepted into the blockchain ($p/2$ security level). In blockchain sharding, all nodes carry the same weight. Thus, it is only under the assumption that the honest participants are creating as many nodes as they can that a $n/2$ security level blockchain sharding reaches the $p/2$ security level. Secondly, when some nodes leave the system, other nodes need to be reassigned, frequently, from shard to shard in order to maintain the security level of the system. In this paper, we present Multichain MWPoW, a $p/2$ security level blockchain sharding architecture that does not require honest participants to create multiple nodes and requires less node reassignment when some nodes leave the system. It combines the Multiple Winners Proof of Work consensus protocol (MWPoW) with the flexibility of $n/2$ blockchain sharding. Our experiments show that Multichain MWPoW outperforms existing blockchain sharding approaches in terms of security, transaction throughput and flexibility.

Keywords: Blockchain, Distributed Ledger, Blockchain Security, Blockchain Sharding, Blockchain Performance

1. Introduction

Different kinds of blockchain, e.g., Nakamoto Blockchain [1], Ethereum[2], have been proposed in the past ten years. While blockchains have been designed initially to handle cryptocurrencies, they have since shown promise for more sophisticated usage, such as powering Decentralised-Autonomous-Organisations (DAO) or Decentralised-Autonomous-Companies (DAC), where anonymous participants can carry out tasks together without centralised control. Various mechanisms have been proposed to ensure the integrity of such decentralised work as well as the incentives to the participants, blockchains still suffer from both security and performance problems which significantly limit their applicability in practice.

The fairness and decentralisation of blockchain-based systems are dependent on how participants reach public consensus. This is usually done by a strength competition known as *mining*. Participants synchronise the new transactions, verify and approve the first legit block that reaches the threshold strength in every time window. If they approve a block, they will compete to create a new block of the threshold strength on top of the block (the hash of the block is embedded to the text of the new block). Rewards are then given to the creator of the approved block as incentive. This procedure, however, overlooks the heterogeneous nature of devices used in a blockchain, causing a vicious circle between the reward rate deprivation and the arms race for stronger computation capability as well as broader network bandwidth. This vicious circle can ultimately result in a centralised system where some participants are always winners of competitions, while others leave the system. Besides, as blockchains seek to improve throughput, they may choose extending block size or employing mining pools. Extending block size may force less powerful devices to leave the system as they do not have the capacity needed to constantly download and verify large blocks from the network. This drives a blockchain system gradually towards a centralised one. A mining pool assembles less powerful

participants and uses them collectively to create blocks. But as participants in a mining pool do not know how their computation power is used, this raises concerns over system security.

Various approaches have been explored to solve the security and performance problems associated with blockchains. These approaches can be commonly categorised into off-chains [3, 4], lightweight blocks [5, 6, 7], weighted models [8, 9, 10], directed acyclic graphs [11, 12], and blockchain sharding [13, 14, 15, 16]. Among these, blockchain sharding is promising as it offers a good balance between security and performance. Blockchain sharding works by splitting a blockchain into several parallel-run committees (shards), thereby increasing transaction throughput, reducing computational capacity required, increasing reward expectation for participants, yet still maintaining the security level required. Recently, several flexible sharding methods that can tolerate up to $n/2$ Byzantine nodes ($n/2$ security level) have been proposed. However, these methods suffer from two main drawbacks. First, in a non-sharding blockchain, nodes can have different weight (power or stake) to create a consensus. So an adversary needs to control half of the overall weight of the system in order for a piece of faulty information to be accepted into the blockchain ($p/2$ security level). In blockchain sharding, all nodes carry the same weight. Thus, it is only under the assumption that the honest participants are creating as many nodes as they can that a $n/2$ security level blockchain sharding reaches the $p/2$ security level. Secondly, when some nodes leave the system, other nodes need to be reassigned, frequently, from shard to shard in order to maintain the security level of the system.

In this paper, we propose Multichain MWPoW, a $p/2$ security level blockchain sharding architecture that does not require honest participants to create multiple nodes and requires less node reassignment when some nodes leave the system. It combines the Multiple Winners Proof of Work consensus protocol (MWPoW) with the flexibility of $n/2$ blockchain sharding. Our experiments show that Multichain MWPoW outperforms existing blockchain sharding approaches in terms of security, transaction throughput and flexibility. In the rest of the paper, we first provide a brief review of some of the well-known non-sharding blockchain approaches in Section 2 to give some background to our work, and then describe the blockchain sharding idea in detail in Section 3 and 4. We will describe our Multichain MWPoW approach in detail in Section 6, and highlight the challenges associated with it in Section 7. Analysis of the Multichain MWPoW structure in terms of bandwidth demand is given in Section 8. We report our performance study in Section 9. Finally we draw conclusions in Section 10.

2. Non-Sharding Blockchain Approaches

Blockchains were initially proposed to deal with cryptocurrency transactions. Central to this technique is a Proof of Work (PoW) scheme which describes a system that is difficult to create but easy to verify. The most widely used PoW scheme, Hashcash [17], is based on SHA-256 and is part of Bitcoin (Nakamoto blockchain) used as a computation strength competition method. Since then different kinds of PoW alternatives have been proposed for blockchains [18, 19, 20].

A block in a blockchain embeds the information of a specific period, and the blockchain is periodically updated by the participants competing to create new blocks and attaching them to the existing chains of blocks. In a blockchain, *Difficulty* is a measure of how difficult it is to generate a PoW:

$$Difficulty = \frac{difficulty\_target}{current\_target} \tag{1}$$

where *difficulty_target* is a 256-bit constant and *current_target* is any 256-bit number. When calculating the difficulty of a hash, the hash is used as the *current_target*. A blockchain network has a global block difficulty: valid blocks must have a hash below the *current_target*, and the hash is adjusted by changing the value of Nonce (a field in the block). The global difficulty is adjusted to limit the rate at which the network can generate a new block within an approximately fixed time interval. A blockchain also has a pre-set security requirement that honest people must take more than 50% of the total calculation power so that malicious participants cannot to create a longer fork branch of blocks when honest people are working on another. New participants can then determine the correct records by staying with the longest chain (or the *mainchain*), which is expected to be longer than the second-longest chain by at least some given length.

When powering a cryptocurrency using a blockchain, the participants only need to check whether the sender of a transaction has spent the fund or not in the blocks of the blockchain before they accept this transaction. As a

result, double-spending is prevented: no one should be able to send the same money to more than one receiver at the same time. However, when we use blockchains to power other decentralised applications, additional performance and security issues have been considered and various approaches have been proposed to address them.

2.1. Off-chain approach

"If a tree falls in the forest and no one is around to hear it, does it make a sound?" The quote questions the relevance of unobserved events – if nobody hears the tree fell, whether it has made a sound or not is of no consequence [3]. In a blockchain, if only two participants care about an everyday recurring transaction, it is not necessary for all other participants in the blockchain network to know about that transaction [3].

The Off-chain approach was created based on this philosophy. Nodes use Micropayment channels [21, 22] to establish a relationship between two parties to perpetually update balances, deferring what is broadcast to the blockchain in a single transaction, netting out the total balance between the two parties [21]. The Off-chain approach empowers nodes to transfer funding privately through micropayment channels, and has mechanisms to secure the interest of the other side of the channel if a node makes a violation of its previous off-chain statements [4].

The Off-chain approach is quick and efficient. However, if we view it from a financial perspective: a network of Bidirectional Payment Channels [23] (similar to a BGP system) would be needed when multi-parties are involved. Consequently, users with a large sum of money become banks, and the system is then tending to be financially centralised. Note that with the Off-chain, transactions are only broadcast when one party violates previous transactions. If we assume that such transaction broadcasting is not often needed, users must keep monitoring the blockchain and refund their funding when violations are made [3]. This prevents personal devices like mobile phones and desktops from using the off-chain blockchains directly because they might not be able to monitor a blockchain all day long. Also, it is not clear how to use the off-chain approach in non-financial applications.

2.2. Lightweight block approach

Because transactions are broadcast to a network, it is relatively safe to assume that nodes would have received the majority of the transactions before receiving a block. The transactions inside a lightweight block are replaced using tiny transaction hashes, and the relevant plain-text transactions are only shared when a node fails to decode a lightweight node. Graphene [5] is a blockchain protocol that makes a block contain over 2000 transactions with a size of only 2.1 Kbytes. Similar approaches are Xtreme Thinblocks [6] and Compact Blocks [7]. The lightweight block approach significantly extends block throughput. However, nodes with less computational power might not be able to hear the extensive information due to limited bandwidth or to verify the information and calculate PoW in time to catch up with the mainchain. Thus, it may cause a more severe arms race among nodes.

2.3. Weighted models

With blockchains based on weighted models, some criteria are used to weight nodes in a blockchain such that the duties of the nodes are differentiated by their weights. A lightweight node system is an example of the weighted model: a lightweight node does not store any block and is a client of full nodes. Full nodes are the nodes that synchronise all transactions and blocks. Lightweight nodes use Simple Payment Verification (SPV) inquires to request relevant previous transactions from the full nodes to verify a new transaction. A lightweight node only takes up to 4.2 MBytes per year, regardless of the total size of blockchain [8], but it cannot verify the next blocks and can be misled by the full nodes. Delegated Proof of Stake (DPoS) [9] is a model where people elect a fixed number of representatives and contribute their stakes to these representatives. These representatives then compete in the game of PoS [18]. DPoS can support massive throughput because the representative nodes usually have a superpower calculation ability, storage, and network bandwidth.

These models are now commonly used in many blockchain-powered *IoT* systems, where lightweight nodes are at the edge, and they contribute their stakes to DPoS to function the system. These models are using authoritarian/superior nodes, and they are potentially centralised. The system security depends on these representatives. Therefore, weighted models are not an appropriate approach for *DAO* and *DAC*, which is what blockchains originally proposed and designed to support.

2.4. Directed acyclic graph

IOTA[24] is a Directed Acyclic Graph (DAG) implementing a blockchain, and it eliminates the power centralisation problem caused by extensive PoW competition. It is not like an ordinary blockchain: it does not employ a mining mechanism, nor holds periodical competitions. Instead, all actions in IOTA are carried out asynchronously. However, IOTA is more vulnerable to the so-called 34% power attack [12], and still requires a rather long pending time to use a transaction as an INPUT from the new transactions. In addition, nodes are still required to download a great number of transactions to determine the reliability of subsequent transactions.

3. Blockchain sharding

Blockchain sharding is a blockchain performance enhancement approach that divides transactions and participants of a blockchain into multiple zones (referred to as shards). This increases blockchain throughput as participants and shards increase. Every transaction has a corresponding shard that deals with any future transactions based on it. By enabling multiple shards to process transactions in parallel, the throughput is increased without increasing the computational requirement of the nodes. Blockchain sharding has also been used to reduce storage requirement for nodes [25], which helps blockchains to be implemented on IoT devices that lack storage space. Financial models [26] can be built into blockchain sharding to link the digital labour and market behaviour to the changes in pay and service prices.

However, the first sharding consensus protocol Elastico [13] has four weaknesses. First, after every iteration, all shards need to be rebuilt, and node identities need to be reset. Second, because it demands a significant amount of time to fill up all the shards by solving enough PoWs, the latency grows linearly with the increase of network size. Third, an adversary may calculate PoW in advance so that he or she can mislead the process of assigning nodes to shards. Fourth, as a shard of small size (around 100 members) is needed to restrict the running of Practical Byzantine Fault Tolerance (PBFT) [27] in each shard, it increases failure probability. This implies that the protocol is insecure in practice because the failure probability can be over 0.97 after six iterations [14]. Furthermore, even though Elastico requires each participant only to verify a subset of transactions, they must still synchronise every block from every shard.

Some improvements have been suggested to overcome these weaknesses. RSCoin [15] is a sharding protocol designed to scale up centrally-banked cryptocurrencies. It is an approach that attempts to transparentise today's banking systems by combining a distributed network with a centralised monetary supply. However, this blockchain protocol relies on a trusted source and it is not Byzantine fault-tolerant because each shard is executed on a two-phase commit protocol. OmniLedger [14] solved the problems suffered by the Elastico protocal, but it can only tolerate up to $n/4$ adversary nodes where n is the total number of nodes in a blockchain. RapidChain [16] increased this tolerance to $n/3$, but every shard must use a fixed size instead of a Binominal (or approximately, Poisson) random variable. As such, the shard size can still be a significant issue that will limit the improvement on blockchain throughput.

Xu and Huang [28] have recently proposed a new approach that takes blockchain sharding to a $n/2$ security level or can tolerate up to $n/2$ adversary nodes. This approach classifies nodes into different classes and maintains an equal number of nodes of different classes in every shard. This approach does not only increase security level, but also substantially shrinks shard size. However, such a blockchain can grind to a halt by an adversary with less than $n/2$ of nodes, that is, the blockchain stops to generate new blocks but the record in the blockchain is still correct. There is an extension to this approach [29], which dynamically alters the number of classes as well as the size of shards to bound the probability of global halting by an adversary and recovers the system from halting eventually. However, the number of shards allowed in this extension can be reduced drastically, which can un-stabilise throughput. Also for both solutions, every time new nodes are added or existing nodes leave the blockchain, a global node membership adjustment will take place. These drawbacks increase the frequency in data synchronisation, and uncertainty in system stability and throughput.

In the rest of this section, we explain the concept of blockchain sharding and the security models associated with a number of existing blockchain sharding approaches in detail.

3.1. Blockchain sharding hypothesis

Following the Off-chain philosophy, it is not necessary for everyone to hear every tree falling to maintain the fairness of a system. The fact that a tree has fallen and the correct time of its fall has been recognised by most people

around the tree assumes that these people have not colluded. Collusion is hard to take place when a sufficient number of people are assigned randomly to and evenly distributed in sub-areas of the forest, and if we relocate people from time to time to prevent accumulation of potential collusion in any particular sub-area. Blockchain sharding follows from this hypothesis. As long as the random and distributed assignment of nodes is secure and follows the principle of proportionality, taking control of a sub-area (shard) would require a similar effort as taking control of the whole forest (the system). Figure 1 shows an illustration of this hypothesis.

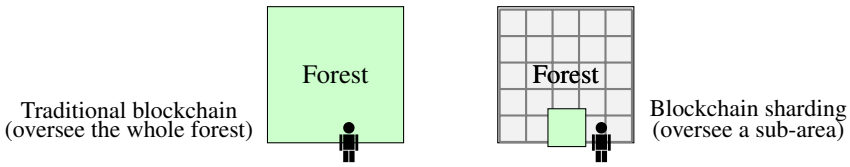


Figure 1: The philosophy of blockchain sharding

Note that this proposal is secure only when (1) people assigned to a sub-area of the forest have the right to record information about this sub-area; (2) no one cannot control or predict the sub-area into which it will be assigned; (3) the assignment follows a globally recognised rule, not by the power of some specific group of superior people; (4) people are periodically reassigned; and (5) qualified people can be assigned and the qualification period is not shorter than the time that one can continuously stay in a sub-area. This means that there will be no benefit for anyone to quit a sub-area and start over again, in case the miner assigned to a shard does not want to stay anymore. When the security threshold is maintained for the whole blockchain, we have more than half of the total population (or half of the total calculation power) being honest people. They do not need to hear every falling tree by themselves. They would only need to check the falling time of a tree in the sub-area that is of interest to them. If every sub-area does this, then people do not need to have a super hearing power, especially when the forest is dense. Instead, they only need to focus on monitoring the sub-area they are assigned to and split/merge sub-areas when they become dense or sparse.

3.2. Security models and levels

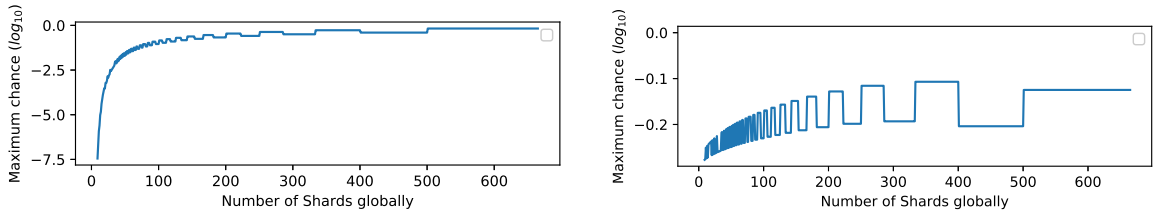
While blockchain sharding is a simple and appealing concept, there are some substantial security challenges to be addressed. First, how do we distribute people to sub-areas in a decentralised and unpredictable way? Second, how can people determine if a record of a sub-area is made by people assigned to that area? Third, without monitoring what happens in a sub-area, how can an outsider know if the majority in that sub-area supports a record or not? Fourth, to make a collusion hard to happen, how large the population in a sub-area must be and how many sub-areas the forest must have? In this section we discuss the security models and issues surrounding the blockchain sharding approach.

Failure probability

Assuming that there exists some method that can solve all or some of the challenges, we would like to know how likely its node assignment would cause a collusion to happen in a shard, or its failure probability. Given n nodes, the probability of having no less than $X(X > m/2)$ adversary nodes in a shard when randomly picking a shard of size m (the number of nodes inside the shard) can be calculated by the cumulative hypergeometric distribution function without replacement. Let X denote the random variable corresponding to the number of adversary nodes in a shard. The failure probability for one shard is

$$\Pr[X > \lfloor m/2 \rfloor] = \sum_{X=\lfloor m/2 \rfloor+1}^{m} \frac{\binom{t}{X}\binom{n-t}{m-X}}{\binom{n}{m}} \tag{2}$$

where t is the number of adversary nodes in the system. Figure 2 shows the maximum probability to fail with $n = 2000, t = n/3, t = n/2$ and $m = n/s$ where s is the number of shards. As can be seen from the result, the system has a very high failure chance when the adversary takes $n/2$ of nodes. This is the main reason why most blockchain sharding approaches can only withstand up to $n/3$ of nodes being adversary and only a few shards can exist.



$t = n/3 = 666$ $t = n/2 = 1000$

Figure 2: The chance to fail when $n = 2000$ and $m = n/s$ where s is the number of shards;

$n/2$ security level blockchain sharding

In this section, we introduce a sharding approach that can withstand $n/2$ of nodes being adversary in a system of n nodes. Let there be m classes of nodes and s number of shards. Every shard must have one and only one node of each class, so every shard has m nodes from the m classes. A consensus of a statement in a shard is reached when at least a pre-defined T number of nodes in this shard agree on this statement, typically $T > m/2$. The system is separated into a working zone and a pending zone, such that the nodes inside the working zone are placed into the shards and can mine (verify transactions, propose and approve blocks), while the nodes in the pending zone will wait to be assigned into the working zone. When new nodes join the system, they choose a class and are placed into the pending zone. Every class of nodes in the pending zone forms a queue and the nodes are queued in order of time at which they choose a class. Let $wq(i, j)$ represent node j of class i in the pending zone, and let $lwq(i)$ represent the number of nodes of class i in the pending zone. Every time when $\min_{1\leq i\leq m} lwq(i) \geq Q$ where Q is a pre-defined number, then the first Q nodes of every class are added to the working zone, and all the nodes in the working zone are reassigned to shards.

Now assume that an adversary control A_i nodes in class i in the working zone, and without loss of generality the adversary puts all the controlled nodes into classes $i = 1, \ldots, T$ of the m classes. Note that as the adversary has no more than $n/2$ of nodes, he or she does not have enough nodes to fill up all the spots in T classes. Then, the probability for the adversary to secure a manipulated consensus inside a shard is

$$Pr[T] = \prod_{i=1}^{T} \frac{A_i}{s} \tag{3}$$

where T is the number of nodes the adversary must take in a shard to manipulate the consensus. Table 1 shows a possible node assignment schedule table for 10 shards run in parallel with a shard size of 5 (5 people in different classes), where A refers to an adversary node and H an honest one.

Table 1: Class based node assignment

Class \ Shard	0	1	2	3	4
Class 1	A	A	A	A	A
Class 2	H	A	H	A	H
Class 3	A	H	A	H	A
Class 4	H	A	H	H	A
Class 5	H	H	H	H	A

To derive maximised $Pr[T]$, we want $\prod_{i=1}^{T} A_i$ maximised because s is the same. Let the adversary control a total of t nodes, then $t = \sum_{i=1}^{m} A_i$. To maximise $\prod_{i=1}^{T} A_i$, we consider

$$A_i = \lceil (t/T) \rceil, \quad i \in [1, t \bmod T] \tag{4}$$

$$A_i = \lfloor (t/T) \rfloor, \quad i \in (t \bmod T, T] \tag{5}$$

This represents the maximised scenario because given any positive integer Z,

$$Z \times Z > (Z-1) \times (Z+1) = Z \times Z - 1. \tag{6}$$

Therefore, the maximum chance for the adversary to succeed on an attack is to place the nodes into T classes as equally as possible. Thus,

$$Pr[T]_{max} \approx (\frac{t}{T \times s})^T \tag{7}$$

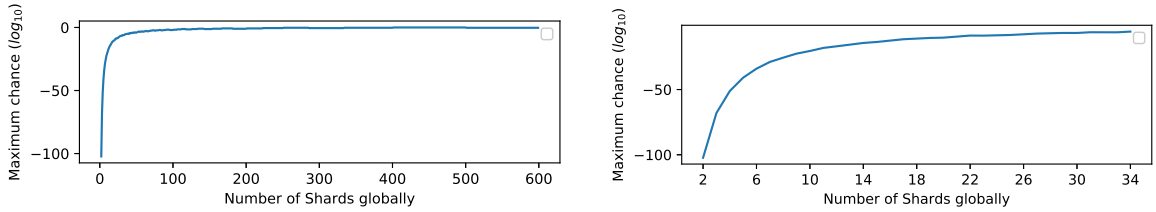
If $T = m$ (all the people in a shard reach the same decision), then

$$Pr[T = m]_{max} \approx (\frac{t}{s \times m})^m \tag{8}$$

Now if the adversary controls $t = \frac{s \times m}{2}$ nodes (half of the overall population), then the failure probability is

$$Pr[T = m]_{max} \approx (\frac{1}{2})^m \tag{9}$$

So to make the system function securely, we want $T \approx [m/2]$ while meeting the security threshold (e.g. 10^{-6} failure chance). Figure 3 shows the maximum failure chance with different s,$n = s \times m = 2000$, $T = 0.7 \times m$ and $t = 1000$ ($1/2$ fraction of the overall population).



$s \in [2, 600], t = 1000 = n/2$ $s \in [2, 34], t = 1000 = n/2$

Figure 3: The chance to fail with different s when $n = 2000$ and $m = n/s$ where s is the number of shards;

As can be seen from the result, when there are 10 shards and $n/2$ people being adversary, the failure chance is below 10^{-20}, which significantly outperforms the previous sharding approach at 10^{-6} (see Figure 2) when there are 10 shards and only $n/3$ nodes being adversary. If we maintain failure chance at 10^{-6} with $T = 0.7 \times m$, on the other hand, then the $n/2$ approach can have 33 shards running in parallel, which is significantly better than the $n/3$ solution.

Global halting problem

With the $n/2$ approach introduced above, an adversary will not be able to manipulate a consensus when he or she does not control at least T nodes inside a shard. However, it can still halt a consensus to be reached on a statement when it has $m - T + 1$ nodes in a shard. As a result, the verdict on this statement cannot be made until the nodes are reassigned. So the whole system can halt when there are $s \times (m - T + 1)$ adversary nodes and all of these nodes are in the same $m - T + 1$ classes. In this case, it is guaranteed that the adversary will have $m - T + 1$ nodes inside every shard. Table 2 shows an example of a system halting. With $m = 5$ and $T = 4$, the adversary takes $m - T + 1$ nodes in every Shard. Note that the halting problem cannot be eased by adding more nodes, as the shard which in charge of the membership issues will also stop to function.

Table 2: A halting scenerio

Class \ Shard	0	1	2	3	4
Class 1	A	A	A	A	A
Class 2	A	A	A	A	A
Class 3	H	H	H	H	H
Class 4	H	H	H	H	H
Class 5	H	H	H	H	H

Flexible $n/2$ security level blockchain sharding

A flexible $n/2$ security level blockchain sharding approach [29] has been proposed to solve the problem of global halting. In this approach, every node is given a colour from the colour spectrum, and every shard categories its nodes by grouping them to their closest *base colours*. If there are m categories inside a shard, then there are m base colours, which together represent the colour spectrum as a whole. Figure 4 shows an example of base colour. The system may change the number of categories (by combining/splitting shards) to maintain the security threshold. When a global halting occurs, we can increase the categorisation number globally (which will decrease the number of shards) to overcome the halting problem. Because the adversary does not have more resources than the honest people globally, in the worse case, the halting problem can be solved by reducing the number of shards to one. Once a system halting scenario is resolved, the system can then begin to split shards again.

We now consider the failure probability of this flexible $n/2$ security level sharding approach. Assume that an adversary has $t = \frac{n}{2} - 1$ nodes. The chance for the adversary to take control of a shard in a system of n nodes, m base_colours and s shards ($s = n/m$) is:

$$\Pr[T]_{Max} = \left(\frac{t}{T \times s}\right)^T = \left(\frac{t/T}{n/m}\right)^T \approx \left(\frac{m}{2 \times T}\right)^T \tag{10}$$

So when the adversary takes $T > 0.5m$ colour categories of nodes inside a shard, it controls that Shard. Thus, T can be adjusted with m while maintaining a fixed threshold failure chance. As can be seen from Figure 5, when m is over 800, T/m is very close to 0.5, i.e. the adversary needs to take approximately $n/2$ of nodes to break the system.

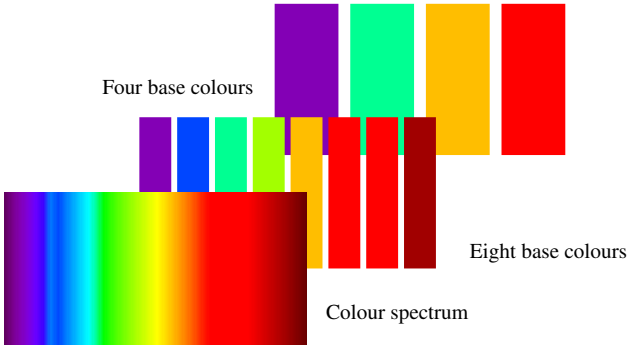


Figure 4: The colours spectrum and *base colours*

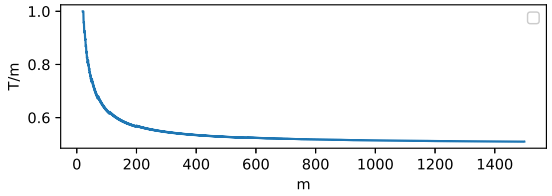


Figure 5: T/m for maintaining a 10^{-6} failure chance with different m

4. $p/2$ security level blockchain sharding

Blockchain sharding approaches use plurality voting instead of resource/strength competition (mining) to generate consensus in every shard. That is, they trust a statement voted by most people, rather than by most strengths/influences, inside a shard. Thus, while an adversary in classical blockchains needs more than 50% of overall strength to replace the mainchain with a new chain (referred to as $p/2$ security level), an adversary in blockchain sharding would need to take more than $n/3$ or $n/2$ of the total number of nodes to force faulty information into a blockchain (referred to as $n/3$ or $n/2$ security level).

An $n/3$ security level is always less secure than a $p/2$ level, and an $n/2$ security level is generally less secure than a $p/2$ level. $n/2$ and $p/2$ security levels offer equal security only when all participants in an $n/2$ level sharding scheme create as many nodes in the system as they can. This is because in an $n/2$ or $n/3$ sharding, a node only needs to have certain (threshold) strength to join the system, yet a participant may have more strength than a node requires, therefore in order to fully present all their strengths, they need to create multiple nodes within the system. However, there is a cost for a participant to maintain many nodes, as their nodes can be assigned to different shards, requiring increased workload in sychronising and processing data. Due to this cost, an honest participant, especially those using

a device with a limited computational power, is likely to create just one or a small number of nodes, leaving room for an adversary to create many nodes to gain control over the system,

In this section, we propose a $p/2$ blockchain sharding approach that (1) requires less frequent data resynchronisation and membership adjustment in comparison to [28, 29]; (2) overcomes the halting problem [28] before a complete halting occurs; (3) causes less loss in transaction throughput than [29] when recovering from halting; and (4) lifts an $n/2$ Byzantine node resistant blockchain into a $p/2$ Adversary (Byzantine) power resistant level. In the following we describe this approach in detail.

4.1. Model description and failure chance analysis

Suppose that every node has a different strength when voting a consensus. In PoW-based systems, this strength represents the calculation power that a participant has, whereas in PoS-based systems, it represents the amount of stock that a participant has. In this paper, we refer to strength as calculation power. Our model has three main components.

1. **Node classification**. Line up all the nodes into a list $L = \{L_0, \ldots, L_n\}$ in order of their strengths and n is the number of nodes in the system (excluding pending nodes). Let CP_x represent the strength of L_x and Sg be a pre-defined number of groups for nodes such that every group $i \in [0, Sg)$ has a lower strength boundary $bl(i)$ than others in the CP list.

$$bl(i) = CP_{\lfloor \frac{n}{Sg} \times i \rfloor}, i \in [0, Sg) \tag{11}$$

   Every shard must have at least one node from every group.
2. **Block evaluation**. We assume that when an adversary proposed a block containing faulty information, the honest nodes would not vote for it. Let AP be the overall strength of the adversary nodes, the chance for a block in shard j being controlled by the adversary is

$$\Pr(j) = \prod_{i=0}^{i<Sg} \frac{\binom{AP/tt}{NgS(i,j)}}{\binom{n/Sg}{NgS(i,j)}} \tag{12}$$

   where $NgS(i, j)$ is the number of nodes in group i which are currently located in shard j and have voted for the block; $DG(i) = 1$ if at least one node from group i in shard j voted for this block, otherwise $DG(i) = 0$. Let

$$tt = \sum_{i=0}^{i<Sg} DG(i) \times bl(i) \tag{13}$$

   In order to make the system maintain a $p/2$ security level, we consider

$$AP = \frac{\sum_{x=0}^{x<n} CP_x}{2} \tag{14}$$

   Formula 12 brings an overestimated result because we assume every node in any group i has the same strength ($bl(i)$). Figure 6 gives an illustration of Equation 12. We can accept this block safely if (1) more than half of the strength in shard j has voted for it (the majority principle), and (2) the chance for the block to be a wrong one is lower than the security threshold. Nodes can still mine on blocks that are insecure, but transactions in them would only be accepted when the blocks or the branches stemmed from them reached the security threshold.
3. **Shard merge**. From time to time, some shards may merge. Shard j will be merged with another when:
   (a) $Max(Pr(j)) > Th$, where Th is a security threshold (e,g. 10^{-6}). $Max(Pr(j)) = Pr(j)$, when for every $i \in [0, Sg), DG(i) = 1$. It is obvious that shard j should be merged with others when all the nodes inside shard j have voted for a block but the chance for this block to be a wrong one (proposed by the adversary) is still larger than the security threshold.
   (b) When at least five continuous blocks in the mainchain of shard j have not reached the security threshold. In this case, we say a **local halt** has occurred.
   (c) When there is no node from a group currently located in shard j.

   To make a local halt difficult, we want every possible AP/tt to be smaller. Thus, when adding new nodes, the system should prioritise those shards whose strength is close to the average strength of nodes. They should post penalty or delay adding nodes that would raise AP/tt. Restrictions should also be placed to avoid an extremely unbalanced power distribution inside the system.

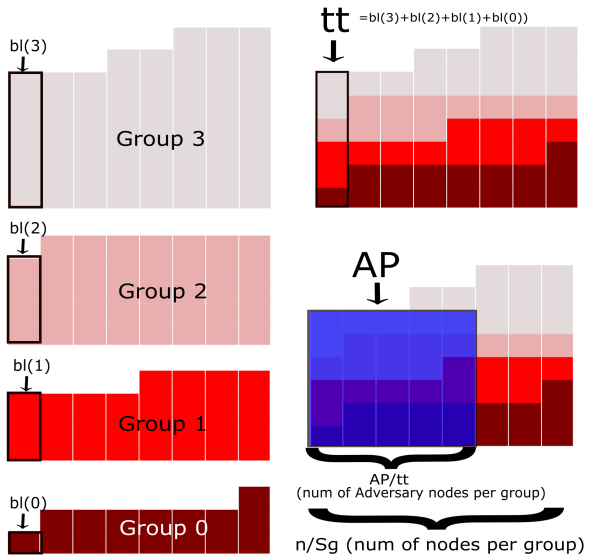


Figure 6: An explanation of formula 12. $Sg = 4$. The heights of the bars represent the strengths of the nodes. If a block has supports from every group, then $tt = \sum_{i=0}^{3} bl(i)$. The adversary would create AP/tt number of nodes in every group – the best strategy for the adversary is to place an equal number of nodes into different groups for the reason given in Section 3.2.

4.2. Our approach

In our approach, chains are like shards in the previous $n/2$ approaches but shards can dynamically merge or split depending on the workload. Nodes are ranked according to their strength, and they are divided into groups, as introduced in Section 4.1. Restrictions are placed to secure every chain having at least one node from every group, and there are mechanisms to restore the restrictions when they are broken. More specifically, We require that,

1. Taking any $2/3$ fraction of nodes out of the system, the sum of their strength must be equal to or larger than $1/2$ fraction of overall strength in the system. Nodes will be kept in the pending status if adding them to the system would compromise this requirement.
2. Every shard can assign nodes to other shards, there is no particular chains to deal with membership issues (like the committee shard in [28, 29]).
3. Nodes are only required to synchronise the blocks of the chain they are assigned to and the block headers of all chains.
4. The nodes are relocated from chain to chain on a periodical basis (e.g. every Ti iterations of the *mining game* after the node participated, and Ti is a pre-defined parameter). We do not adjust the nodes globally when adding new nodes.
5. When new miners are joining in the system, we ask them to present Ti times of strength that they intend to use per iteration of the mining game. In this way, it requires the same effort between remaining in a chain for Ti iteration of mining and qualifying as a new miner.
6. A miner does not need to be re-qualified if it is re-assigned to other chains after Ti iterations of competition.
7. In every round of a mining game, the chains exchange their local group boundaries of their nodes. They do so by recording that information into the block header, which is synchronised by everyone. By viewing all block headers, $bl(i \in [0, Sg))$ of Formula 11 can be derived.

We use an edited MWPoW [19] (will be discussed in session 5), a decentralised mining pool like blockchain protocol as the protocol to run every chain. The design of MWPoW that a miner needs to register strength before participating in the mining game can secure the power distribution in a multichain scenario. It also helps divide nodes into different groups automatically. By using the edited MWPoW, nodes of other chains can determine if a block of a chain is created by the population of that chain and if the majority supports that block.

Figure 7 and Table 3 show a comparison among the IOTA, blockchain, Multichain MWPoW and other blockchain sharding approaches. Full nodes in the IOTA store every transaction ever sent to the network. Full nodes in the blockchain store every block in the mainchain. Nodes in other sharding approaches keep transactions in their areas as well as node membership information from a particular shard (the committee or the court office).

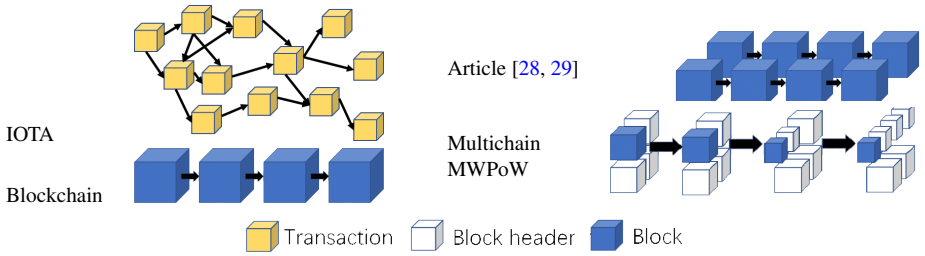


Figure 7: Local storage comparison

Table 3: Comparison of approaches

Name	Transactions stored *	Byzantine fault tolerance level	Storage refreshing time#	Transaction pending time (minutes)$	Maximum No. of shards^
Nakamoto	Tx	$p/2$	None	30 (3 block confirmation)	1
Article [28]	Tx/s	$n/2$, global halt may occur.	$T_{add} + T_{drop}$	10 (Plurality voting) &	33
Article [29]	Tx/s	$n/2$, global halt recoverable.	$T_{add} + T_{drop}$+ restriction restore	10 (Plurality voting) &	33
IOTA	Tx	$p/2$	None	Unstable @	None
RapidChain	Tx/s	$n/3$	T_{add}/Avg_{add}	Plurality voting and unstable %	10
Multichain MWPoW	Tx/s	$p/2$, no global halt	Every Ti iteration and restriction restore	Around 12.5 (0.25 block interval)~	33

* Tx is the number of overall transactions, and s is the number of shards.

\# T_{add}/T_{drop} refer to every time when there are nodes added or dropped. Avg_{add} refers to the average number of nodes added each time.

$ The block interval is 10 minutes, and the transactions must pend before the block reaches its acceptance criteria.

@ IOTA requires an accumulation of later transactions pointed to a transaction in order to accept that transaction. Thus, the speed to accept a transaction eventually is largely dependent on how active the network is.

% As it needs approval from all the Input committees, the time to accept a transaction is unstable.

& Plurality voting systems require nodes to vote within a pre-defined time window (block interval), and the transaction is confirmed after one block interval if a consensus is reached.

~ Nodes in Multichain MWPoW need to submit four shares (PoWs) per iteration. Thus, when a node reaches Acceptance Difficulty, a support rate can be derived similarly to that of Plurality voting. Sometimes the support rate of a block is not enough to make a block to be accepted eventually, and we need to wait until new blocks to be created on top of it. On average, a block can be accepted in 1/4 block interval after it has been announced (reaches Acceptance Difficulty).

^ In a 2000-node system and the system is secure from global halting. For Multichain MWPoW, there are 33% of power that is adversary power. For others, the adversary nodes are taken 33% of the overall node population.

Note that with Rapidchain [16], whenever a fixed number of nodes is added to a shard, the same number of existing nodes need to be re-assigned to other shards. Nodes then need to synchronise data from the new shard once they are there. In [28, 29], when nodes are added or dropped, nodes assignment needs to be adjusted to meet the categorisation requirement. Also, local halting is resolved by a global membership re-arrangement. When global halting occurs, the approach given in [29] will also need to re-assign nodes while cutting the colour categorisation. In our proposed approach, the adjustment is more stable. We re-assign nodes to other shards after every Ti iterations of a mining game,

where *Ti* is a pre-defined global parameter. A shard would merge with another to overcome local halting, and there is no need to make global membership adjustment.

To avoid the double-spending problem, in our approach, transactions are governed by different chains: new transactions can only be conducted on the governing chain of the INPUT transaction. If a user wants to transfer a transaction to another chain, they need to conduct a cross-chain operation. Furthermore, our approach does not have levels of committees, a feature inherited from the fast confirmation property of MWPoW [19], and as such it generally outperforms other strength-based blockchains in terms of transactions per second.

In summary, our proposed Multichain MWPoW approach has the following novelties:

- **Increased Byzantine Resiliency**. To the best of our knowledge, Multichain MWPoW is the first sharding approach that can withstand up to 50% of Adversary (Byzantine) power without assuming honest nodes having to hold as many nodes as possible. There is less chance for a global halting to occur, as it cannot be deliberately planned like the scenario we discussed in Section 3.2. If a global halting does happen by accident, it can be resolved just like how we deal with a local halting.

- **More Flexibility**. A chain (shard) can be split and merged base on its data flow. Every chain can carry a different number of participants. In contrast, the number of shards in RapidChain [16] and in [28] are fixed. In [29], this number can change, but nodes still need to be equally divided into shards. When they lose a node in a shard, they need to cancel that shard, pushing nodes back to the pending zone and reorganise a new shard from the pending zone. Our approach is therefore more efficient.

- **Increased Transaction per Second**. Less time is spent on halting, and an attack is hard to be conducted. Multichain MWPoW allows fewer number of nodes per chain, and the chains are more stable. More shards can process transactions in parallel for the same security threshold compared to others.

- **Faster Transaction Confirmation**. There is no level of election network in Multichain MWPoW. A transaction is confirmed when the governing chain has confirmed it. There is only one governing chain per transaction.

5. Multiple Winners Proof of Work protocol

Multiple Winners Proof of Work (MWPoW) protocol [19] is proposed to shorten the transaction pending time, improve the reward rate for individual miners, and ease the centralisation problem of blockchain. In the following we first introduce some definitions, then describe how the protocol works.

5.1. Definitions

- **Calculation Power Claim.** A miner's calculation power is defined as the hash difficulty one can achieve in a fixed time window, or what a miner intends to reach in every episode of a mining game.

$$CP = CP_0 + CP_1 + \cdots + CP_{N-1} \tag{15}$$

  where CP is the overall calculation power claimed by registered participants, N the number of registered participants in the network, and $CP_i, 0 \leq i \leq N-1$ is the calculation power claim of registered participant i.

- **New Join.** New Join is a data set which records the calculation power claim of a participant and a wallet address of this participant (the wallet address is used for receiving remuneration). There is a HashPrevBlock field in the New Join, which records the hash of the latest block in the mainchain. There is a Nonce field in the New Join, which is used for adjusting the hash of the New Join. For a New Join to be valid, its hash must meet at least the calculation power claim indicated in the New Join.

- **Try Range.** Try range TR is a number interval of Nonce in the block header.

$$TR_i = \left[\sum_{k=0}^{i-1} Tt_k, \sum_{k=0}^{i} Tt_k\right), Tt_{i\in N} = \frac{CP_i}{CP} * 2^{256} \tag{16}$$

  where N is the number of registered participants in the network. Miner $i \in N$ mines on TR_i.

- **Acceptance Difficulty.** The first block which reaches Acceptance Difficulty in an episode of mining should be placed in the mainchain. Acceptance Difficulty is adjusted based on how much time has been consumed for the winner block to achieve the Acceptance Difficulty.

$$AD_x = \frac{BI * AD_{x-1}}{Timestamp_{x-1} - Timestamp_{x-2}} \tag{17}$$

where AD_x is the Acceptance Difficulty at block height X; BI is the pre-defined block interval, and $Timestamp_x$ is the time when block X is created.

- **Entrance Difficulty.** A block is broadcast to the network when it reaches Entrance Difficulty. Entrance Difficulty of a new episode is adjusted based on how many blocks reach Entrance Difficulty in the previous round of the mining game.

$$ED_x = min(\frac{NE_{x-1}}{DN} * ED_{x-1}, \frac{AD_x}{2}) \tag{18}$$

where ED_x is the Entrance Difficulty at block height X, NE_{x-1} the number of blocks reach Entrance difficulty at block height $X - 1$, and DN the ideal number of NE, which is set 1.

- **Share.** Share is a container of Nonce when broadcasting. The Nonce inside a Share, sent by a miner, must make the hash of the block fulfill at least 25% of this miner's calculation power claim.

- **Countable Share.** If a miner has sent at least two Shares for a block, the difficulties of these Shares will be count towards the Support Rate of this block, and the miner will be able to receive remuneration for announcing this block if this block wins the game later.

- **Share Difficulty Cap.** The maximum sum of difficulties of Countable Shares sent by a miner X in a round of a game is CP_X (its calculation power claim). If it sends more, the sum is capped at CP_X.

- **Reward.**

$$R_{i \in N^R} = \frac{SD_i}{SD_{\{X\}}} * R_{\{X\}} \tag{19}$$

N^R represents the miners who have contributed Countable Shares for announcing block X; $R_{\{X\}}$ is the overall reward assigned from the system for the block in block height X; Shares of block X are embedded in block $X + 1$; $SD_{\{X\}}$ is the total difficulty of the Countable Shares embedded in block $X + 1$; SD_i is the difficulty of the Countable Shares miner i has contributed, and $R_{i \in N^R}$ is the amount of remuneration given to miner i as a Coinbase transaction in block $X + 1$.

- **Valid Block.** A miner determines a block as a valid one when the transactions, new joins, and shares in this block are correct, and more than 90% of the shares and new joins must be previously known to the miner.

- **Support Rate.** The Support rate of a block is defined as the ratio between the sum of the difficulties of the Countable Shares for the branches stemming from this block and the sum of difficulties of all Countable Shares of all the branches in the blockchain from the block height of this block.

$$SR_X = \frac{\sum_{i=X}^{XL} SD_{\{i\}}}{\sum_{i=0}^{k} \sum_{j=i}^{iL} SD_{\{j\}}} \tag{20}$$

SR_X is the support rate of block X; XL is the latest block on top of the blockchain branch stem from block X; k is the number of all the branches; iL refers to the latest block on top of a specific branch; $SD_{\{X\}}$ is the total difficulty of the Countable Shares for block X.

- **Statement Rate:** $STR_X = \frac{SRP_X}{Rp_X}$, where $SRP_X = sum(CP_{i \in k})$, k is the set of miners who have sent two valid Shares for a block of block height X and are not violating the restriction for branch choosing (once a miner sends two Shares for a block, it should not send Shares for another block of the same block height); Rp_X is the registered power at block height X.

5.2. Game overview

Miners need to claim an amount of calculation power they intend to put into every round of the mining game before participating in the game, and they do so by sending a New Join to the network. A New Join is considered valid when (1) the hash difficulty of it is equal or greater than the calculation power indicated in it; (2) The HashPrevBlock is the hash of the preceding block of the latest block in the mainchain. Each miner is given a unique *TR* based on the calculation power it claims. When a miner creates a block and finds a Nonce that fulfills Entrance Difficulty in its *TR*, it broadcasts the block as well as the Nonce in a Share. Then other miners will attempt to find a Nonce in their *TR* to make this block fulfill Acceptance Difficulty if they acknowledge this block as a valid one. Ideally, miners should announce a block collectively by doing PoW in their *TR* in parallel. When a Share of a block is broadcast, if the Nonce inside enables the block to reach Acceptance Difficulty, then this block is announced. The first block reaches Acceptance Difficulty is the winner block, and miners who have contributed Shares to this block will divide the remuneration from mining.

During the announcement, miners should send Shares that do not fulfill the acceptance difficulty but fulfill at least 25% of the power they have claimed previously as the proof of contribution. A miner can only send up to four Shares to the network per round. If more than one block is successfully announced in one round of the game, miners should mine on top one on a branch which has reached Acceptance Difficulty first. Miners may have different views in terms of which block has reached Acceptance Difficulty first due to network delays. Assuming this winner block is the block X, the blocks of the next block height (block $X+1$) will embed Shares of block X. According to the Shares integrated, if a miner failed to find the Shares which together weigh more than 50% of the power it has previously claimed, this miner will be expelled from the game. This expulsion means that the miner's *TR* will be canceled since block height is $X+1$. The remuneration for the miners of block X is given in the block height $X+1$ as Coinbase transactions. All valid miners of block X divide the reward based on the difficulty of the Shares they have sent. As every miner oversees different Try Range, it is easy to determine which miner should receive what amount of remuneration.

5.3. Game procedure

The mining game for every miner is carried out as follows:

- **Register Power.** A new miner creates and submits a New Join to the system.
- **Get a Try Range.** Miners whose New Joins are embedded into a block will be assigned with Try Ranges.
- **Mining.** Try to create a block and find a Nonce that fulfills Entrance Difficulty in miner's *TR*. If a miner's block has reached Entrance Difficulty and miners have approved this block, miners will try to find a Nonce of Acceptance Difficulty in their Try Ranges.
- **Getting Reward.** If a miner has submitted an adequate number of valid Shares for the winner block, the amount of reward would be given at the next block height.
- **Re-arrange Try Range and Start Over.** After one round of the game, invalid miners will be globally expelled. Miners who have failed to send Shares which stand for at least 50% of the power they have claimed will get their Try Ranges cancelled. New miners will be added as well as Try Ranges for all the valid miners to be re-arranged. After that, a new round of game starts. Miners who have submitted New Join before and have not been expelled do not need to register power again to participate in the new round of the game.

5.4. Block simplification

We use a block simplification algorithm Graphene [5] to simplify a block when its size increases due to embedding New Joins and Shares. Graphene [5] combines Bloom filter [30] and IBLT [31]. Graphene can encode dozens of thousands of transactions into several Kbytes. Graphene encoded blocks can be decoded using previously received information. Graphene has detailed mechanisms to deal with the failure of decoding. The structure of MWPoW block is given in Figure 8. It is important to simplify MWPoW blocks so that increase in participants will not significantly affect block sizes. A block of extended size can slow down block broadcasting and may, as a result, affect the fairness of the system, i.e. some miners may receive a block faster and start to mine the next block earlier than others.

Note that while a block is simplified, nodes still need to synchronise with all New Joins and Shares in the system to decode simplified blocks. However, the bandwidth requirement in this case is not as heave as it seems because the New Joins and Shares would not have been sent at the same time, but spread over many iterations. A New Join can be a size of 102 bytes and a Share $36bytes$, according to [19]. Compared to Nakamoto blockchain, MWPoW only requires around additional $2Kbytes/s$ of bandwidth for a participant to synchronise all the New Joins and Shares in a bitcoin-like system with 8000 nodes and the block interval setting of 10 minutes.

5.5. Distributed remuneration

According to the Shares embedded in a block, the compensation for announcing a preceding block is given to miners in the winner group directly in Coinbase transactions. Figure 9 shows an example of a remuneration distribution, where the sum of difficulty of the shares sent by Miner *A* and Miner *B* are 212 and 49 respectively, and the sum of the difficulties of all valid shares of the block is 1000. The total reward amount from the last block height is 100. Miner *A* and Miner *B* receive 21.2 coins and 4.9 coins, respectively.

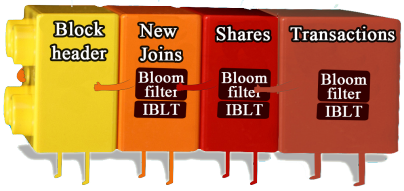


Figure 8: The structure of a MWPoW block

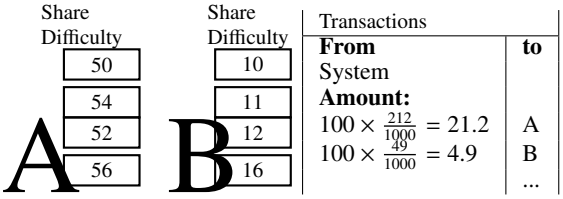


Figure 9: Reward assignment

5.6. Fast block confirmation

In the Nakamoto blockchain, nodes have no information about the Support Rate of a block. Miners hold the blocks until the difference of accumulated difficulty between different fork branches is large enough for nodes to accept the most difficult one as the mainchain. In this case, the blocks in that branch are accepted. However, in MWPoW, by registering power, we know the overall calculation power in the game. By counting Shares, we can acquire how much calculation power has agreed on which branch of the blockchain and the Support Rate of a block can be easily calculated and compared. This procedure waives the need for later block confirmations.

It is pre-defined that if a miner has sent two Shares for a block, this miner will not be allowed to change branches in this round of the game. Otherwise, it will be expelled, and its contribution will not be counted toward the Support Rate. It is also pre-defined that the miners should mine on the block, which, to their knowledge, first reaches Acceptance Difficulty. A miner can move to mine on another block when this miner has not yet sent two Shares for a specific block. A miner will likely do so if there is a block of more Support Rate.

A block is eventually accepted when:

- it is announced and is inside the highest branch of the mainchain;
- the Statement Rate of the latest block height is larger than 50%;
- the amount of power that has supported this block is significantly more than the amount of power that has backed the second largest block plus 25% of the registered power of the latest block height.

Figure 10 shows an example of branch choosing where *D* stands for the difficulty, and *SR* stands for Support Rate. In (*a*), when blocks *A*, *B* and *C* are announced, none of them get a more than 50% Support Rate; thus, we cannot determine which block is to be accepted. In (*b*), when there are succession blocks of block *A*, *B* and *C*, the Support Rates of block *A*, *B*, *C* are changed. Blocks *C* and *D* are eventually accepted because they have more than 50% of the Support Rate. Meanwhile, this Support Rate is larger than the Support Rate of either block of the same block height plus 25% of the registered power.

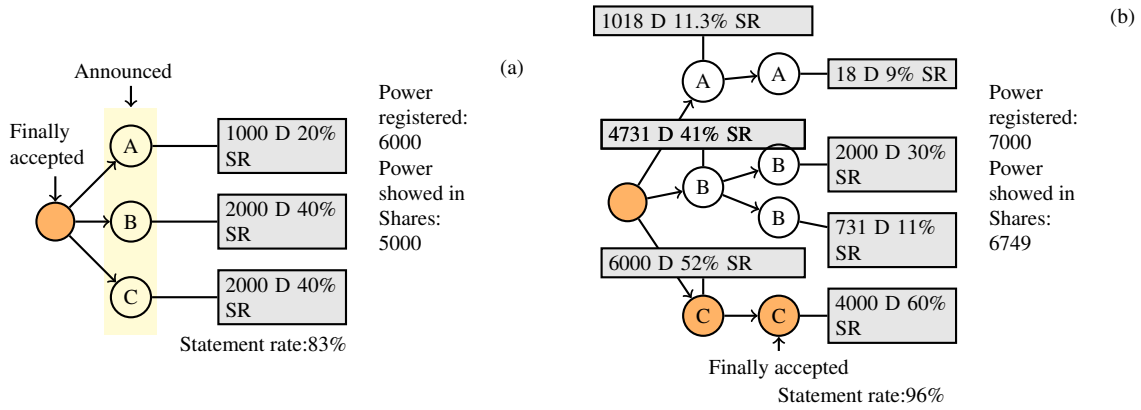


Figure 10: Finally accepting a block

6. Multichain MWPoW

MWPoW provides less powerful nodes the ability to profit from mining games and to judge blocks collectively. An accepted block can be confirmed quickly if half of the mining power has voted for it. However, because there are New Joins and Shares, verifying a block is resource-demanding, especially when block intervals are shortened, since the bandwidth of individual nodes needs to be high to synchronise all the data in time. Multichain MWPoW addresses this issue and increases the scalability of the system by splitting transactions into multiple parallel MWPoWs.

6.1. Multichain MWPoW outlines

6.1.1. Definitions

- **Chain ID.** Chain ID is formatted as C+*digits*. Chain ID is given and changed based on the history of split/merge of chains from the start of the system. A chain can only be split into two at the same time. The two new chains use new names by appending a "0" or "1" at the end of the ID in binary format, for example, $C1$ split into $C2$ and $C3$. When two chains are merged, if they stem from the same branch, the ID after merging is the old name of that branch. For example, $C2$ and $C3$ become $C1$ again, if they are merged. If two chains are merged into one and they did not stem from the same branch, the name for the merged chain is the smaller one of the two Chains' IDs. For example, if $C5$ and $C3$ are merged, the new ID would be $C3$.

- **Lifelength.** Lifelength refers to the time (continuous iterations of mining game) that a miner can play in a chain after being assignedto this chain. There is a pre-defined Lifelength Ti; $Ti \ mod \ 4 = 0$.

- **New-Assign-Join.** New-Assign-Join is like New Join in MWPoW but with one additional field: "Identity_Key". To make a New-Assign-Join valid, the hash of this New-Assign-Join must fulfill T times of its Calculation Power Claim.

- **Chain Limit.** Every chain has an upper limit of K and a lower limit $\frac{K}{2}$ of the number of transactions and New-Assign-Joins per block. When a pending transaction and New-Assign-Join number exceeds or breaks the upper/lower limit, a chain will be split into two or merge with others.

- **Ordinary Block.** An Ordinary block (Ob) records the same information as the block in MWPoW records, except that it does not record New Joins.

- **Power-assignment Block.** Apart from information of an ordinary block, Power-assignment block (Pab) additionally records New-Assign-Joins. Pab is used to assign the owners of the recorded New-Assign-Joins into different chains. Pab records up to K New-Assign-Joins while recording up to K transactions. The preceding block of a Pab is an Ob.

- **Assignment Box.** An assignment box is the container for New-Assign-Joins, and it is embedded only in a Pab. There are two sections in an assignment box: a New participant section and a Re-assignment section.
- **New Join.** New Join in Multichain MWPoW is a dataset that contains a New-assign-Join and a Merkle branch. The Merkle branch must prove this New-Assign-Join has been written to a Pab of a particular chain.
- **Fuel-up Block.** Fuel-up block (Fub) of a chain records the New Joins that are assigned by a Pab to this chain after the previous Fub of this chain. The creator of the New Joins recorded in this Fub can start a mining game in this chain after the current block height (Try Ranges are assigned). The preceding block of a Fub is an Ob, where this Ob's preceding block is a Pab.
- **History / OffSptring Chain.** When a chain is merged/split, the new chain(s) are the Offspring chain(s) of this chain. This chain becomes a history chain of its Offspring chain(s).
- **Duty Range.** A range of transactions/New-Assign-Joins, which should be processed by a chain.
- **TransOnhold.** TransOnhold is a number added to the block header. This number stands for the number of transactions/New-Assign-Join received by the creator of the block. These transactions/New-Assign-Joins should be legal and within the Duty Ranges. In the meantime, they should have not yet been written into a block in the mainchain of this chain.
- **Chainpower.** The amount of the overall registered power (in PoW difficulty form) inside a chain.
- **Threshold Chainpower.** This ranks the CP of the participants inside a chain in ascending sequence, place the ranked sequence in a list $RCP_{0...NPC-1}$. NPC is the number of registered participants inside this chain. Threshold Chainpower is the sum all the values from $RCP_{0...\lfloor \frac{2}{3} \times NPC-1 \rfloor}$
- **Bl_candidate.** An integer array of Sg items indicated in the block header,

$$Bl\_candidate(i) = RCP_{\lfloor i \times (NPC/Sg) \rfloor}, i \in [0, Sg) \tag{21}$$

- **Global Block Header.** Global block header is a Merkle root of the hash of all the latest accepted blocks of all chains.
- **Crosschain Section.** When one transfers a transaction between chains, the transaction is written into the Cross-chain section.

6.1.2. Amendment to the designs of MWPoW

The following amendments have been made to the designs of MWPoW:

1. Three types of blocks: Ordinary block (Ob), Power-assignment block (Pab), and Fuel-up block (Fub) take turns to be written into a chain, i.e. they repeat the writing sequence of Ob, Pab, Ob, Fub.
2. Fub records New Joins, where as Ob and Pab do not.
3. The Block interval time of every chain is set to be the same. Thus all chains generate blocks in an approximately same time window.
4. $TransOnHold$ is placed into the block header.
5. $Share$ is signed by the private key of the Indentity_Key of its creator.
6. $Chainpower$ is added to the block header.
7. $Bl\_candidate$ is added to the block header.
8. $Threshold\ chainpower$ is added to the block header.
9. $Number\ of\ participants$ is added to the block header, which states the number of valid registered miners inside a chain.

10. The block in every chain records a list of valid registered miners inside its chain. [1]
11. Every block embeds a global block header; the hash of the global block header is written into the block header of every block. Global block header records the hashes of the latest finally accepted block of all the chains. We allow these hashes to be the second latest one because the block generation among chains goes incompletely synchronised.
12. Nodes only hear the blocks of the chain they are assigned to as well as the block header of the announced blocks from other chains. When a block is announced, miners in all chains should download the block header of this block.
13. Apart from the rules of MWPoW regarding final acceptance of a block, a block reaches Acceptance Difficulty is accepted when the chance for the block to be incorrect is lower than the required security threshold. This chance is calculated using Equation 12. If there are two blocks which reach the Acceptance Difficulty at the same epoch in a chain and the chance for them to be incorrect are lower than the security threshold, the one (say *Alice*) with more Support Rate is accepted if the differences between the Support Rate of the two blocks is more than a specific value. This value is defined as one that the adversary can gain with a pre-defined security threshold probability. The chance for the adversary to control this particular value of Support Rate differences can also be calculated using Equation 12 by enumerating some voters of *Alice* and assuming the enumerated voters are controlled by the adversary (only use the enumerated votes to calculate the $DG(i)$ and $NgS(i, j)$). The enumerated adversary voters together should contribute the amount of differences between the Support Rate of *Alice* and the other block. The chance for the enumerated voters to be adversary should be lower than the security threshold probability.

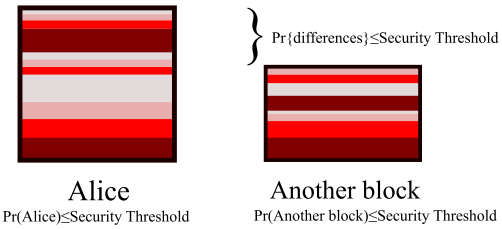


Figure 11: The explanation of the additional rules for accepting a block

Figure 12 shows the structure of Multichain MWPoW.

6.1.3. Game procedure

- **Register power.** A participant can create a New-Assign-Join based on a Fub of a chain (HashPrevblock should be the hash of that Fub). After the New-Assign-Join is constructed, usually after Ti iterations of the game as the hash difficulty of this New-Assign-Join must reflect T times of its Calculation Power Claim. The participant then sends the New-Assign-Join to that chain.

- **Wait for the power assignment.** In every four iterations (whenever a Pab is created), up to K qualified New-Assign-Joins of new participants is selected by miners in a chain. A random assignment protocol is used to place all the selected New-Assign-Joins into the assignment box of the new Pab.

- **Register with the chain assigned to.** After a Pab *Alice*, which embedded the participant's New-Assign-Join, is announced (reaches Acceptance Difficulty), the participant then creates a New Join, which contains that New-Assign-Join and a Merkle branch. The Merkle branch should prove this New-Assign-Join has been assigned to a specific chain by *Alice*. Finally, the participant should submit this New Join to the chain assigned by *Alice*.

[1] In the original MWPoW, the participant list is not written in the block, which can be derived by counting the New Joins and Shares since the beginning of the system. Including participant list does not increase the bandwidth demand significantly because the block is encoded using Graphene. Nodes do not need to swap any clear text of the participant list unless a discrepancy is detected.

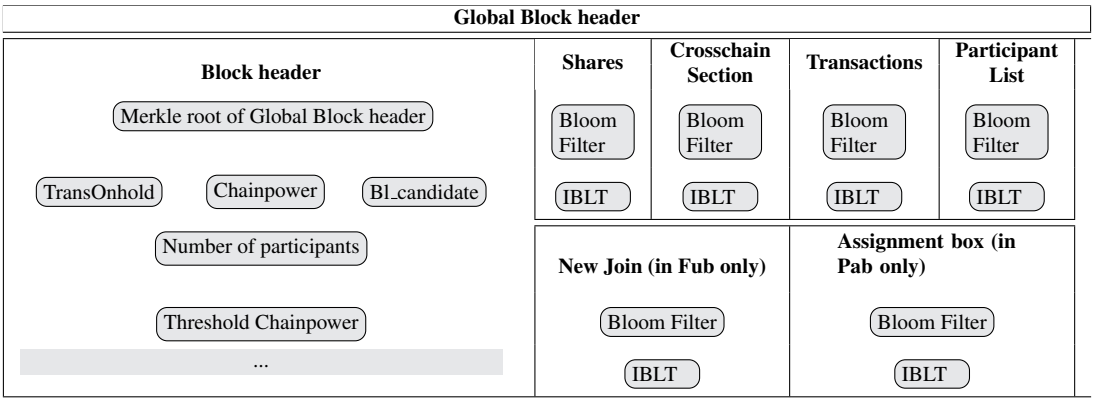


Figure 12: The Block of Multichain MWPoW

- **Get a Try Range.** Miners in the assigned chain check if the New Join they have received is valid. They also check if the Pab (*Alice*), which has made the assignment, is the latest finally accepted Pab in its chain. A Try Range is given to the participant at the next Fub, then the original rules of MWPoW begin to apply.
- **Split and merge the chains.** When the chain violates chain limits, the latest block will indicate if the chain should be split or merged. The miner then enters the split or merged chain following the rule of split/merge.
- **Reassignment.** When a miner has inside a chain for Ti rounds of the game, it is reassigned to another chain.
- **Expel.** The same rule as MWPoW, if a miner has not sent at least three valid Shares per iteration that are successfully embedded in the block, it is expelled.

Figure 13 shows an example of the game procedure of Multichain MWPoW.

6.2. Global parameters

6.2.1. Group boundary

The number of nodes in a system can be derived by adding together the number of participants indicated in the block headers of the latest finally accepted blocks in every chain. Let NC be the number of chains,

$$bl(i) = min(Bl\_candidate(i, j)), i \in [0, Sg), j \in [0, NC) \tag{22}$$

where $Bl\_candidate(i, j)$ refers to the $Bl\_candidate(i)$ of the latest finally accepted block in chain j. Miners of chain j then classify the nodes inside the chain according to the bl derived. Every time the group boundary is determined, miners should examine if some restrictions are met. The restrictions includes:

1. $2 \times Threshold\ Chainpower \geq Chainpower$.
2. There is at least one node from every group in this chain.
3. $Max(Pr(j)) \leq Threshold$, where $Threshold$ is the predefined security threshold.

If these restrictions are not met, then chain j should be merged with others.

6.2.2. Global block header and dispute resolution

Because nodes only synchronise information with its chain and the block headers of the announced blocks of other chains, nodes are unable to determine if a block of another chain is genuine and can be finally accepted. To solve this, we propose a mechanism:

1. When a block is announced or finally accepted, relevant miners should broadcast this information to miners of other chains.

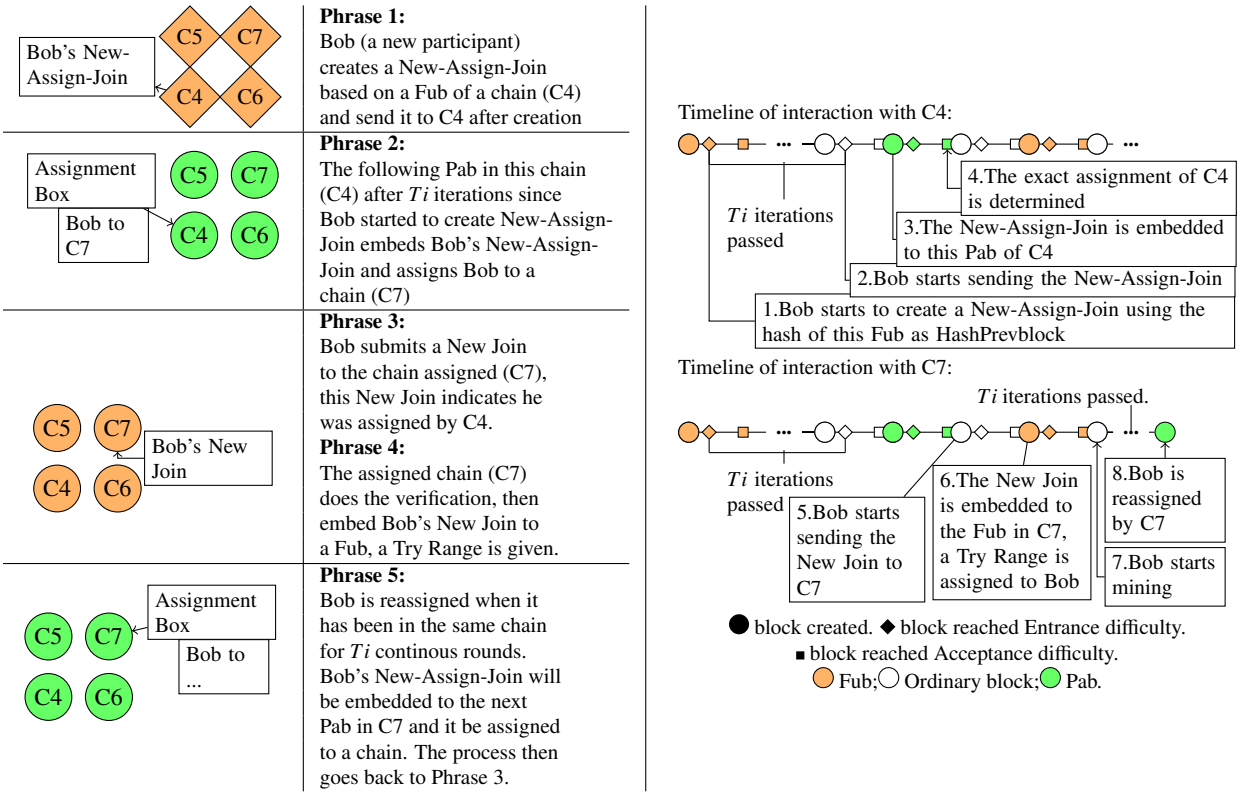


Figure 13: The procedure of Multichain MWPoW

2. Miners should periodically ask several miners in other chains to see if the announced blocks have been finally accepted.
3. If a conflict is known to a node, this node should synchronise with the participant list in the last block before the suspicious one. It should then determine the genuine Shares and calculate the support rate of the blocks.
4. A block carried a wrong finally accepted block hash in its global block header should be rejected. The miner should not mine on this block in any circumstances.

Because the calculation power is distributed in chains, it is easier for Byzantines to over-write specific blocks in a chain using a greater calculation power. To prevent unregistered nodes affecting the generation of blocks, we rule that:

1. The Shares sent to the network should be signed by the Identity_Keys which have been claimed in the New-Assign-Joins;
2. The Nonces should be within the Try Range that is associated with the Identity_Keys.

Under this mechanism, the first block of a fraud chain of blocks can be determined as invalid because Byzantine cannot provide the correct Shares which are signed by previous participants.

When a new block of a chain which fulfills the Entrance difficulty comes out, nodes of that chain should check the global block header of that block before contributing Shares for it. In this way, when a block is announced, at least a certain amount of calculation power agrees with the global block header attached. Because nodes synchronise all the block headers of the announced blocks of all the chains, they can see the differences between the Merkle root of all the global block headers. A node will request and verify the relevant global block headers if it cannot construct the same Merkle root of the global block header. Figure 14 is an example of a global block header, where NC is the Chain ID, and LASH is the hash of the latest finally accepted the block.

In a brief summary, if a Byzantine attempts to change a finally accepted block of a chain, it must place enough power inside this chain through the normal procedure. If the power is not registered before, it cannot generate valid

NC	LHASH
C2	EA232341AEAFEWER2EKWFL23EWRKL
C6	FB1113A122FIAQFXWSLEEF23ERK1LR4
C7	CCA313A152FIAQF1AWLEWAE3WFETQ

Figure 14: Global block header

Shares, nodes inside the chain will not recognise an invalid block which has reached the Acceptance Difficulty. When nodes of other chains ask which block has been finally accepted, or the honest miners inside a chain has received a fraud block of that chain from the network, the registered honest power inside that chain will appoint another block to the network. When a conflict of finally accepted block has occurred, the Byzantine's block cannot pass the verification of other chains.

6.2.3. Duty Range, Chain split and merge

Duty ranges for a chain include all the New-Assign-Joins, transactions, and New Joins which:

- The HashPrevBlock of the New-Assign-Joins is a block inside this chain.
- The HashPrevBlock of the New-Assign-Joins indicates a block in the history chain of the current chain. The hash of this New-Assign-Join is within a specific range.
- All the Input transactions of the transactions have been committed to any block of this chain.
- All the Input transactions of the transactions have been embedded in the history chains of the current chain. The hash of these Input transactions is within a specific range.
- The New Joins which have indicated their creators are assigned to this chain.
- The New Joins, which have indicated their creators, are assigned to the history chains of this chain, and the hash of the New Joins are within a range.

The valid New-Assign-Joins, transactions and New Joins of a chain complies with the following:

- They are under the government of this chain (inside the Duty Range).
- The INPUT transactions are not used before.

When $TransOnhold$ indicated in the latest finally accepted block of a chain is more substantial than $2 \times K$, then this block is split into two due to the next block's height. However, a chain cannot be split when either of the split chains will not meet the chain restrictions stated in section 6.2.1. When a chain $C1$ is split:

- **Duty Ranges.** According to the hash of the transactions written in the blocks of chain $C1$, if the hashes of the transactions are within the range of 0 to 2^{255} then these transactions are governed by chain $C2$. Otherwise, the transactions are governed by chain $C3$. The duty ranges inherited from chain $C1$ are also equally split into two. Chain $C2$ will take the duty ranges of $C1$ with lower half hashes while the chain $C3$ will take the upper half. The rule also applies to the New Joins on hold. If the hashes of which are within 2^{255}, then the New Joins are processed by $C2$. Otherwise, they are processed by $C3$.
- **Participants.** Rank all the participants by the amount of their *Calculation Power Claim* in ascending order, a participant is relocated to $C2$ if $Pindex \bmod 2 =: 0$ where $Pindex$ is the index number of this participant inside the ranked participant sequence, otherwise this participant is relocated to $C3$.

Figure 15 shows an example of the chain split and merge, where the system starts from one chain $C1$, orange squares are blocks in the chains that currently exist while gray squares are the blocks in history chains.

When merging, a chain will be merged with another that is closest to it in Chain ID. If there are two chain candidates, select the one with a smaller Chain ID. The duty ranges of the chains are also merged. Assume chain $C5$ is merged into another chain $C3$ after a block *Alice* in $C5$ is announced. When *Alice* is announced, the miners

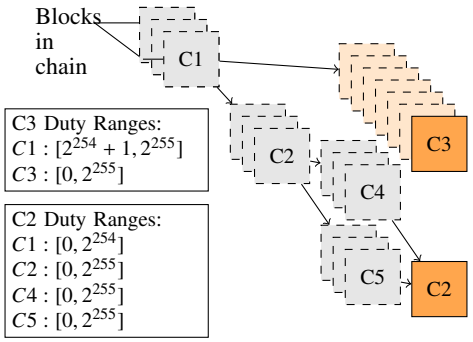


Figure 15: Chains overview

in $C3$ are aware of this merging because they synchronise the block header of all the announced blocks. The miners in $C3$ then synchronise the data in $C5$ between the block interval of the last finally accepted block of $C5$ and *Alice*. The miners use this data to verify *Alice*. If they believe $C5$ should be merged with $C3$ according to the rules, they will mine on the merged chain. When a safe number of nodes in both $C3$ and $C5$ has approved this merge through mining in the merged branch, then the merge is completed. This safe number can be calculated using Equation 12. When chain $C5$ has not generated a finally accepted block for five continuous block interval, the chains into which the chain $C5$ is possible to merge should synchronise the data from $C5$ and determine if they should merge with $C5$. The merged chain starts at the next block height of the highest block height in its history chains. When a chain $C3$ seeking to merge to $C5$, $C5$ is also seeking to merge; if $C5$ is trying to merge with another chain $C6$, then three or more chains merge into one at the same time. Figure 16 shows an example of the chain merge and split, where gray squares are abandoned blocks. If a sufficient number of nodes in $C3$ and $C5$ agree on merging in block height 14, then other branches of them are abandoned.

6.3. Crosschain operation

Because every chain can confirm the situation of blocks in other chains (has been / not yet finally accepted), we take advantage of that to conduct crosschain operations. When a user wants to transfer a transaction to another chain, it first sends the cross-chain-request to the chain that governs the transaction (Origin chain). If this cross-chain-request is written into the crosschain section of a finally accepted block afterward, the user then sends a cross-chain-confirm to the transfer destination chain. The cross-chain-confirm is a Merkle branch that can prove the cross-chain-request has been written into the crosschain section. The destination chain should write the cross-chain-confirm into its cross-chain section, and then the transaction is transferred. The difference in block height between the cross-chain-request and the cross-chain-confirm embedded the blocks should be less than three. If the cross-chain-confirm cannot be written into the destination chain in time, the user will ask the origin chain to cancel the cross-chain request. The miners in the original chain will acquire the cross-chain section of relevant blocks of that destination chain and determine if the transfer should be cancelled. If the user does not send the cancel request, the transaction is being transferred to the destination chain. Figure 17 shows an overview of the cross-chain operation. New transactions of the destination chain can refer to the cross-chain-confirms written in the cross-section of this destination chain as the INPUT transactions.

6.4. Power assignment block

In this section, we show the procedures of forming a Pab for a chain $C5$. There are two parts in forming a Pab: Periodical power re-assignment and New power adding. The structure of the Assignment Box and New-Assign-Join is shown in Figure 18 and Figure 19 respectively.

6.4.1. Periodical power reassignment

Select the nodes which were added to $C5$ at the block height $BH - Ti$, where BH is the current block height. Place the selected nodes' New-Assign-Join into a list PSL by ascending order of calculation power claim indicated in their

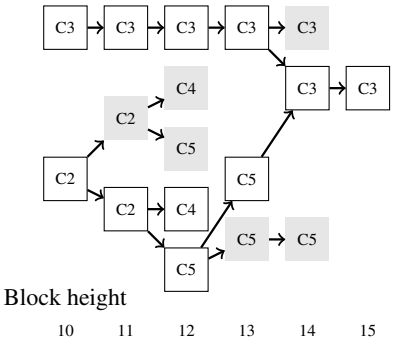


Figure 16: An example of chain merge/split

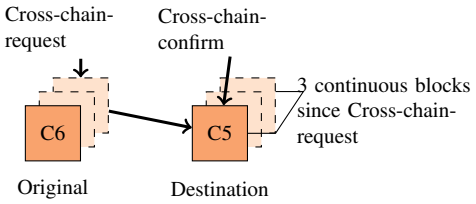


Figure 17: Crosschain operation overview

New-Assign-Joins. Create a new sequence $RPSL$,

$$RPSL_i = Hash(MGBH \oplus Hash(PSL_i)) \tag{23}$$

where $MGBH$ is the Merkle root of the global block header indicated in the latest block of $C5$. Link PSL_i with $RPSL_i$ and rank $RNAJ$ by alphabetical order. After that, a new index of PSL can be reached. Let NC be the number of sub-sections in the "Re-assignment section" of the Assignment box. $Acs(i)$ represents sub-section i ,

$$Acs(i) = \bigcup_{(hash(PSL_j)+j)\ mod\ NC=i} PSL_j \tag{24}$$

Nodes in $Acs(i)$ are assigned to chain i. If chain i becomes a history chain right after, nodes in $Acs(i)$ are assigned to its OffSpring chains according to the Duty Range.

6.4.2. Adding New Power

Miners in $C5$ take the New-Assign-Joins from all unassigned New-Assign-Joins received, which fulfill the following criteria:

- The Nonce inside can make the hash of this New-Assign-Join fulfill the Intended_difficulty.
- The HashPrevBlock is the hash of the Fub at Ti iterations before the current block height.

After selecting the New-Assign-Joins, the following procedure is carried out:

1. Let InD be the Intended Difficulty indicated in a New-Assign-Join. If $bl(i+1) > InD >= bl(i)$ then place this New-Assign-Join to list i. $bl(Sg) = +\infty$.
2. Rank the New-Assign-Joins in every list i by ascending order of $abs(InD - tt)$, where $tt = \frac{bl(i+1)+bl(i)}{2}$. Specially, in this step, $bl(Sg) = bl(Sg-1)$.
3. Select K/Sg New-Assign-Joins from the top of every list. If a list has less than K/Sg New-Assign-Joins, then take all of them.
4. Rank the selected in descending order of their calculation power claim, and sum the front $\frac{1}{3}$. If that is larger than half of the overall power of the selected New-Assign-Joins, then delete the New-Assign-Joins from the top until the front $\frac{1}{3}$ of power claims are not more than half of the overall power of the selected New-Assign-Joins.
5. Rank the remaining New-Assign-Joins according to the alphabetical order of their hashes and place them into a list NAJ. Create a new sequence $RNAJ$,

   $$RNAJ_i = Hash(MGBH \oplus Hash(NAJ_i)) \tag{25}$$

   Link NAJ_i with $RNAJ_i$ and rank $RNAJ$ by alphabetical order. After that, a new index of NAJ can be reached.
6. Let the "New participant section" in the assignment box assign New-Assign-Joins to $min(NC, K)$ chains. $NAJ_{i\ mod\ min(NC,K)=j}$ is assigned to chain j indicated in the assignment box, NC is the number of chains.
7. Write the assignment plan into the "New participant section" in the assignment box.

Assignment Box		
New participant section		
Ll	New-Assign-Joins	Intended_Difficulty
0	$[NAJ_3]$	$[CP_3]$
1	$[NAJ_1]$	$[CP_1]$
2	$[NAJ_2]$	$[CP_2]$
Re-assignment section		

Figure 18: Assignment box

New-Assign-Join	
HashPrevBlock	The hash of the latest block in the mainchain of the chain.
Intended_Difficulty	Calculation Power Claim.
Wallet_address	Used for receiving rewards.
Identity_Key	A public key of a public-private key pair.
Nonce	Number (256bits) that makes the hash of this New-Assign-Join fulfill the Intended_Difficulty.

Figure 19: New-Assign-Join

6.4.3. *Determine the exact assignment*

Any chain accepts the New-Assign-Joins which assigned to it if these New-Assign-Joins are written in a "Re-assignment section". If a New-Assign-Join *Alice* claims she has been assigned to a chain *Ben* by a Pub *Gary* of $C5$ in "New participant section", *Ben* verifies this information by the following steps:

1. Let Ll be the number of subsections in the "New participant section" of the Assignment box of *Gary*, and NC be the number of chains. Rank chains by the alphabetical order of their Chain ID.
2. The New-Assign-Joins in subsection $i, i \in [0, Ll)$ of the Assignment box of *Gary* is assigned to chain $Hash(Gary)\ hash(MGBH + i)\ mod\ NC$ in the ranked sequence.

If it is verified by the above procedure that *Alice* is assigned to *Ben*, then *Ben* should accept *Alice*.

6.5. *Fuel-up block*

Miners need to send a New Join to the chain which they have been assigned to. The New Joins are embedded in the Fuel-up block, and Try Ranges are assigned afterward. Figure 20 is the structure of New Join. The New join for any chain *Ben* is valid when:

1. The Merkle Branch and the hash of the New-Assign-Join attached can form the Merkle root of the Assignment Box of the chain who has made the assignment.
2. The New-Assign-Join is assigned to *Ben*.
3. The Pab which made this assignment is the latest finally accepted Pab of that chain.
4. This New Join has not been used previously.

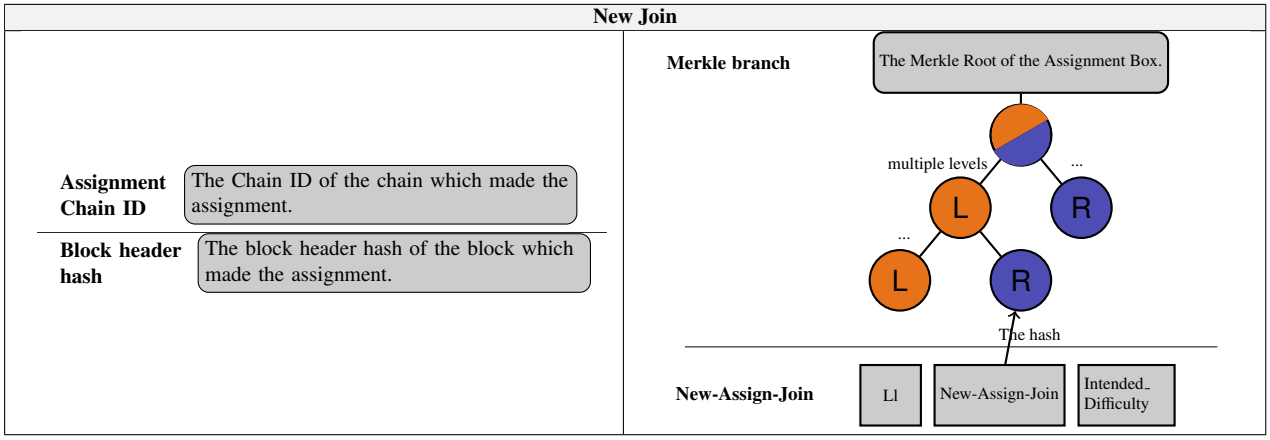


Figure 20: New Join

6.5.1. Power adjustment

When a New Join is valid, it is recorded into a Fub of chain *Ben*. The miners of chain *Ben* acquire the Intended_Difficulty of the New Joins and add the nodes into the chain. However, after adding, if the restriction $2 \times Threshold\ Chainpower >= Chainpower$ is not met, the Intended_Difficulty of the new assigned nodes in $group(Sg - 1)$ is lowered to $bl(Sg - 1)$. If the restriction has still not been met, the chain will be merged with others. The adjusted Intended_Difficulty is then used as the reference for assigning a Try Range.

7. Security analysis

In this section we consider the security features of the proposed Multichain MWPoW in terms of blockchain hypothesis challenges.

Random distribution.

The randomness of the assignment is safeguarded by $MGBH$ in every step. $MGBH$ is changed by every operation and by every transaction embedded in the blocks at the time. An attacker cannot control everything that happens in a system and as a result, making $MGBH$ impossible to be pre-calculated. Also, Bl is hard to be predicted because nodes can join and leave the system freely at any time. A node added or dropped causes a shift in Bl. It is also impossible to predict when the New-Assign-Joins that fulfill the selection criteria would appear on the internet because that is up to the participants globally. For the above result, attackers cannot control the rank of New-Assign-Joins in the sequences, making it impossible to pre-calculate which chains their New-Assign-Joins will be assigned. Also, as it requires Ti times of calculation power claim to qualify a new miner, there will be no gain to quit a chain and repeat the assignment procedure. It takes the same effort to get the miners reassigned to other chains regardless whethr it is a chain or out of a chain currently.

Determine block legibility of other chains.

The protocol discussed in Section 6.2.2 provides a "detect" and "verify" then "synchronise" procedure. When a block is announced, its block header flows to the whole network. When a conflict is detected, miners of other chains can distinguish and recognise a genuine block by acquiring Shares during the conflict and the participant list before the conflict. The recognition is written in the block header as the global block header, which is synchronized and verified by miners globally. We secure the gateway to the inside of every block by the random assignment. Meanwhile, the blocks created by the power outside a chain are not recognised globally, this secures (2) and (3) of the blockchain hypothesis Challenges.

The number of honest participants and the proportion of honest power.

We use the indicator $Threshold\ chainpower$ to describe if the power distribution inside a chain is balanced and secured. If the number of participants in a chain cannot guarantee a safe result, we will merge this chain to others.

Categorisations.

We use the property of a ranked sequence to make the nodes into different categorisations. Nodes can use different powers to make a rough selection of the group, but it is up to the situation of the contemporary nodes to determine the groups eventually. The queues are automatically divided into equal length, few operations to maintain the system is needed, not like previous $n/2$ approaches requiring strict restrictions. The design not only satisfies the challenge of the $n/2$ blockchain sharding hypothesis, but also brings flexibility and stability.

Table 4: The minimum size of structures in Multichain MWPoW

Name	Size	Description
Block header	124 + Sg× 4 bytes	Three 256-bits hashes: the hash of the preceding block,$MGBH$, and the Markle root of the transactions. Seven 32-bits integers: Chainpower,TransOnhold, Threshold Chainpower, Number of participants, Timestamp,Entrance Difficulty,Accetpance Difficulty. Sg number of integers: Bi_candidate.
Share	64.5 bytes	A 4-bits integer(the last four bits of the block hash),a 256-bits integer (Nonce),and a 256-bits signature.
New-Assign-Join	132 bytes	A 256-bits hash (HashPrevBlock), a 32-bits integer (Intended_Difficulty), three 256-bits integers (Wallet address, Identity_Key and Nonce).
New Join	12+32+$log_2(K)\times$ 32 bytes	Three 32-bits integers (Ll,Intended_Difficulty and Assignment Chain ID). A 256-bits hash (Block header hash), and $log_2(K)$ number of 256-bits hashes (Merkle branch).

8. Data analysis

Miners are required to synchronise block headers of the announced blocks of all the chains and the blocks inside the chains they have been assigned to. Table 4 shows the minimum size of a block header, Share, New Join, or New-Assign-Join in Multichain MWPoW.

The participants send shares during every iteration after joining in a chain. The majority of New-Assign-Joins are only broadcast at block heights before a Pab in one iteration interval. The New Joins are broadcast between a Pab is finally accepted, and before the next Fub comes out (also one mining interval). Thus, the minimum upload bandwidth required for a miner in the most data-intensive iteration is $Max(Size_{New\ Join}, Size_{Share} \times 4, Size_{New-Assign-Join})$. The download bandwidth for a participant in the most data-intensive iteration in a system with NPC participants inside the chain is $NPC * Max(Size_{New\ Join}, Size_{Share} \times 4, Size_{New-Assign-Join}) + Size_{Transactions} \times K$. Figure 21 shows the download bandwidth requirement and transaction throughput globally with $n = 8000$ and different number of NC and K. NC is ranged from 1 to 400 ($\frac{n}{20}$, 20 participants per chain). $Sg = 20, NPC = \frac{n}{NC}$, while K ranged from 2 to 1000.

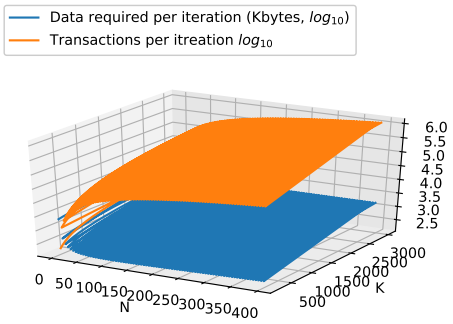


Figure 21: Data requirement and Throughput per iteration with different K and n when $N = 8000$.

9. Experiment

In this section, we experimentally evaluate the overall performance of Multichain MWPoW and test its scalability. We compare its performance with RapidChain [16] and $n/2$ Byzantine node resistant blockchain sharding approaches [28, 29] regarding throughput and transaction confirmation time with different percentage of adversary power in the system. In this experiment, we maintain a 10^{-6} failure chance for every approach. We use a regulated layout of Distributed Ledger Network [32] as the communication protocol used for the essential P2P connections. The connections and the network structure are dynamically adjusted to fit into the data flow to make data propagation fast.

9.1. Experiment setup

We simulated 8000 nodes in a network with $10Mbytes/s$ bandwidth per node in our experiments. For Multichain MWPoW, we gave every connection a random delay time ranging from $1ms$ to $200ms$. The distribution of connection delay time is shown in Figure 22. We have simulated three scenarios A, B, and C of calculation power for every

node, which are shown in Figure 23. In our experiments, we set K to be 2000, meaning that blocks can contain up to 2000 transactions per block, and a Pab can contain up to 2000 New-Assign-Joins in the New participant section of the assignment box. When blocks are broadcast inside a chain, the blocks will be encoded by Graphene [5] like the original MWPoW. If a block is requested by nodes outside the chain or is requested by a new participant when it is synchnising data, the block sent will be the one which is decoded using Graphene.

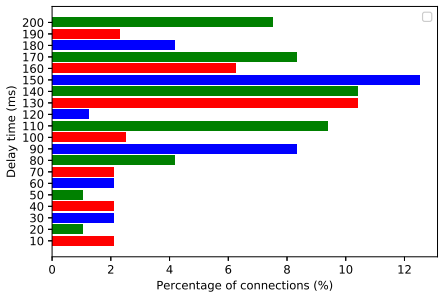


Figure 22: Delay time distribution

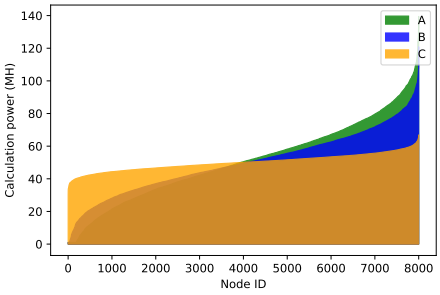


Figure 23: Power distribution

For Other approaches, i.e. the RapidChain and the two existing n/2 blockchain sharding approaches, they are implemented exactly the same as the network setting described above for Multichain MWPoW, except that the protocol runs on every node is not Multichain MWPoW, but the respective protocal and every node is equal in voting.

We send 10^6 transactions per iteration to the network by random nodes (equal allocation). The size of each transaction is fixed at $500bytes$. We record the number of chains (shards) in the system and the throughput (the number of transactions processed globally per iteration). We also record the transaction confirmation time and the frequency of data refreshing (the frequency of a node being re-assigned). For Multichain MWPoW, we increase the amount of malignant power from 0 to 50% of the overall power during an attack, the number of malignant nodes may be higher than half of the node population. For other approaches, we increase the number of malignant nodes from 0 to 50% of all the nodes during an attack. We set the block interval to be 10 seconds globally; the experiments were conducted over 1000 block intervals. Ti is set to 20 block intervals and $Sg = 20$. For $n/2$ blockchain sharding and flexible $n/2$ blockchain sharding, m is set to be 33 at the beginning, and $T = 0.7 \times m$. The flexible $n/2$ blockchain sharding approach may adjust this number during the experiment. The number of shards is set as 55 for RapidChain to maintain the security threshold. For RapidChain, 20% of the transactions are multiple input Shard transactions: a transaction that must be confirmed by all the input shards to proceed. All the approaches used in our experiments maintain a 10^{-6} failure probability.

An adversary node in Multichain MWPoW will function as an honest node if it does not have enough companions in the chain. When there are enough adversary nodes inside a blockchain to halt the shards, the adversary nodes will function maliciously by attempting to create corrupted fork branches. The adversary node in RapidChain has a 50% chance to drop out in every ten iterations. The dropped node will apply to join the RapidChain again immediately. The adversary nodes will start to create wrong blocks when they have gained control over the shard. The adversary nodes for the two $n/2$ blockchain sharding approaches will correctly function when they do not have enough nodes to halt the shards. They will halt a shard immediately when having enough number of adversary companion. We set a transaction in the first block as the initial transaction. The inputs of transactions are randomly selected from the transactions in previous blocks. In our experiments, Multichain MWPoW started with one chain named $C1$. Nodes were added to the system following the rule of Multichain MWPoW as soon as possible. Nodes in other approaches were also added following the rules as quickly as possible. When increasing the adversary percentage, we randomly select the honest nodes in the system and turn them into adversary nodes to obtain the required percentage.

9.2. Experiment results

The experiment lasted 1000 block intervals. Figure 24 shows the changes of the chain (shards) in the progress of block interval. As can be seen from the results, for Multichain MWPoW, the number of chains and the processing

capacity are dynamically adjusted to fit the data flow and to prevent adversary's power from halting the chains. C power distribution scenario is generally more steady than A and B, mostly because the power is more balanced.

From 25, we can see that the $n/2$ sharding approach stops functioning after an adversary took 33% of the nodes, although still produced correct result. We stopped RapidChain functioning after an adversary took 33% of the power because at that point the security of RapidChain was wholly broken. The flexible $n/2$ approach also uses K to indicate pending transactions. We can see that transactions processed by the flexible $n/2$ approach is drastic, and this is different from the pattern in Figure 24. This indicates what number of processed transactions could be. We see this difference because when a shard halts in both $n/2$ and flexible $n/2$ approaches, the shard is frozen until new memberships replace the old nodes in this shard. The halting problem also explains why the $n/2$ approach has slight fluctuation in operation. When a global halting occurred, the transaction per second was reduced to zero, and the system took a few intervals to recover from halting.

In Multichain MWPoW, however, the system would not stop processing transactions; the halting is only about when the blocks would be finally confirmed. Figure 28 shows the times of data refreshing in our experiment. Recall from experiment setup, there is $\frac{1}{20}$ chance for the adversary nodes in RapidChain to quit and rejoin a shard immediately. Since when some nodes are assigned to a shard at any time, the same number of old nodes in this shard must be reassigned to other shards. This design causes the majority of refreshing in RapidChain. There is no limitation of how many times a node can join or leave the system so that an attacker can make this attack in reality on a large scale. This attack could also work for both $n/2$ and flexible $n/2$. However, in our experiment, the adjustment for the $n/2$ approaches are used mainly to solve the halting problem. For Multichain MWPoW, the adjustments are primarily there for solving the local halting and for adjusting to data flow (changes in the number of pending transactions).

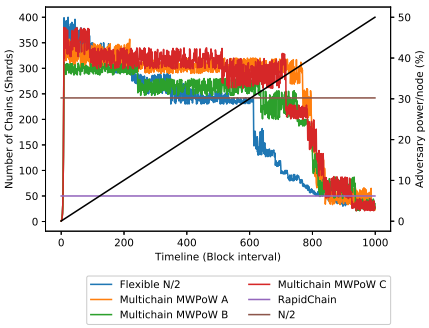


Figure 24: The number of chains/ shards.

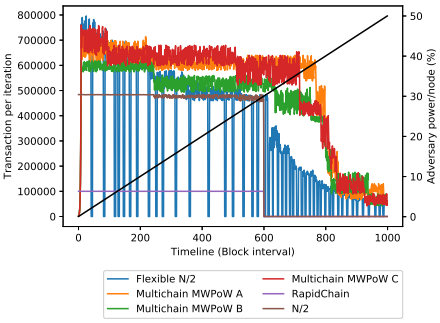


Figure 25: The number of transactions processed per iteration.

Figure 26 shows the average number of transactions per iteration in our experiments. This came to 0 for the $n/2$ blockchain sharding approach and RapidChain after the adversary has taken 33% of the nodes. Figure 27 shows the average pending time for nodes to accept a transaction finally.

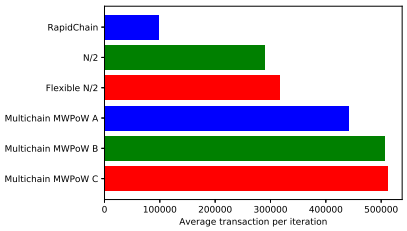


Figure 26: The number of transaction per iteration.

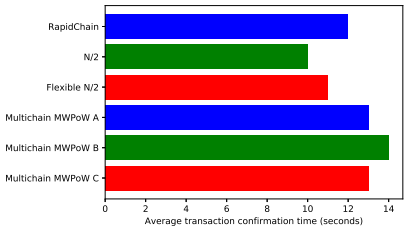


Figure 27: Transaction confirmation time. Recorded from the time a transaction is embedded to a block, and this block is finally accepted.

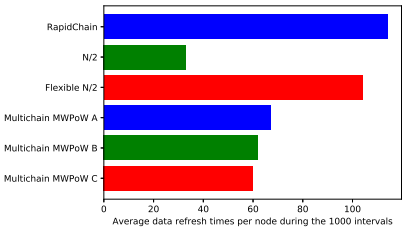


Figure 28: Data refreshing times.

10. Conclusion

We have presented the Multichain MWPoW approach to blockchain sharding in this paper. Our new solution achieves an efficient and robust decentralised autonomous organisation architecture. Multichain MWPoW is the first blockchain sharding approach that can withstand up to 50% of adversary power without assuming that honest people have to create as many nodes in the system as possible. Our experiments show that Multichain MWPoW largely outperforms Rapidchain, the $n/2$ blockchain sharding approach [28] as well as the flexible $n/2$ blockchain sharding approach [29] in terms of stability, throughput and transaction confirmation time. We have proposed a secure random distribution mechanism and maintained a threshold distribution of power inside every chain. We categorise nodes into different classes dynamically and require at least one node per class per chain, the number of participants per chain (shard) is significantly reduced, allowing more chains to be be split. This brings a significant improvement in terms of scalability.

References

[1] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[3] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[4] Jacob Eberhardt and Stefan Tai. On or off the blockchain? insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*, pages 3–15. Springer, 2017.

[5] A Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, and Brian Levine. Graphene: A new protocol for block propagation using set reconciliation. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 420–428. Springer, 2017.

[6] Peter Tschipper. Buip010: Xtreme thinblocks. In *Bitcoin Forum (1 January 2016). https://bitco. in/forum/threads/buip010-passed-xtreme-thinblocks*, volume 774, 2016.

[7] Matt Corallo. Bip 152: compact block relay. *See https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki*, 2016.

[8] Bitcoin, developer-guide. https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv, 2019.

[9] Xinxin Fan and Qi Chai. Roll-dpos: a randomized delegated proof of stake scheme for scalable blockchain-based internet of things systems. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 482–484, 2018.

[10] Yibin Xu. Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. In *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 115–125. IEEE, 2018.

[11] Serguei Popov. The tangle, 2016.

[12] Gerard De Roode, Ikram Ullah, and Paul JM Havinga. How to break iota heart by replaying? In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2018.

[13] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[14] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[15] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. *arXiv preprint arXiv:1505.06895*, 2015.

[16] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 931–948. ACM, 2018.

[17] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[18] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[19] Yibin Xu and Yangyu Huang. Mwpow: Multiple winners proof of work protocol, a decentralisation strengthened fast-confirm blockchain protocol. *Security and Communication Networks*, 2019, 2019.

[20] Yibin Xu and Yangyu Huang. Mwpow-multi-winner proof of work consensus protocol: an immediate block-confirm solution and an incentive for common devices to join blockchain. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 964–971. IEEE, 2018.

[21] Conrad Burchert, Christian Decker, and Roger Wattenhofer. Scalable funding of bitcoin micropayment channel networks. *Royal Society open science*, 5(8):180089, 2018.

[22] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[23] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489. ACM, 2017.

[24] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[25] Yibin Xu and Yangyu Huang. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access*, 2020.

[26] Yibin Xu, Yangyu Huang, and Jianhua Shao. Anchoring the value of cryptocurrency. *arXiv preprint arXiv:2001.08154, 3rd International Workshop on Emerging Trends in Software Engineering for Blockchain*, 2020.

[27] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[28] Yibin Xu and Yangyu Huang. An n/2 byzantine node tolerate blockchain sharding approach. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 349352, New York, NY, USA, 2020. Association for Computing Machinery.

[29] Yibin Xu, Yangyu Huang, Jianhua Shao, and George Theodorakopoulos. A flexible n/2 adversary node resistant and halting recoverable blockchain sharding protocol. *arXiv preprint arXiv:2003.06990, Concurrency and Computation: Practice and Experience, DoI:10.1002/CPE.5773*, 2020.

[30] James K Mullin. A second look at bloom filters. *Communications of the ACM*, 26(8):570–571, 1983.

[31] Michael T Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 792–799. IEEE, 2011.

[32] Yibin Xu and Yangyu Huang. Contract-connection:an efficient communication protocol for distributed ledger technology. *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, Oct 2019.