Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH PAPER

A Flexible $n/2$ Adversary Node Resistant and Halting Recoverable Blockchain Sharding Protocol

Yibin Xu*[1] | Yangyu Huang[2] | Jianhua Shao[1] | George Theodorakopoulos[1]

[1]School of Computer Science and Informatics, Cardiff University, Cardiff, UK
[2]Guilin University of Electronic and Technology, Guilin, China

Correspondence
Yibin Xu, School of Computer Science and Informatics, Cardiff University, Cardiff, UK, Email: work@xuyibin.top

Summary

Blockchain sharding is a promising approach to solving the dilemma between decentralisation and high performance (transaction throughput) for blockchain. The main challenge of Blockchain sharding systems is how to reach a decision on a statement among a sub-group (shard) of people while ensuring the whole population recognises this statement. Namely, the challenge is to prevent an adversary who does not have the majority of nodes globally but have the majority of nodes inside a shard. Most Blockchain sharding approaches can only reach a correct consensus inside a shard with at most $n/3$ evil nodes in a n node system. There is a blockchain sharding approach which can prevent an incorrect decision to be reached when the adversary does not have $n/2$ nodes globally. However, the system can be stopped from reaching consensus (become deadlocked) if the adversary controls a smaller number of nodes.

In this paper, we present an improved Blockchain sharding approach that can withstand $n/2$ adversarial nodes and recover from deadlocks. The recovery is made by dynamically adjusting the number of shards and the shard size. A performance analysis suggests our approach has a high performance (transaction throughput) while requiring little bandwidth for synchronisation.

KEYWORDS:
Blockchain sharding, Halting, Distributed Ledger

1 | INTRODUCTION

During the thrive of Cryptocurrency in the past ten years, researchers presented various kinds of Distributed ledgers, e.g., Permissionless Blockchain[1,2,3], Directed Acyclic Graph (DAG)[4,5,6]. Many are working on extending the usage of Permissionless blockchain to power Decentralised Autonomous Organizations (DAOs)[7], or Decentralised Autonomous Companies (DAC)[8], which gather the anonymous resources throughout the network to collectively perform tests without any centralised manager and the result integrity is guaranteed. For example, by using a scalable blockchain as the foundation technology, a data grid or distributed database can outsource the jobs to the users. Many IoT devices (e,g. Amazon assistants, Google home services) on the one hand, live in the centre of people's privacy (e,g. home, office), while on the other hand they require a significant amount of data being sent to the cloud for analysis. Blockchain may help IoT devices function in a decentralised manner following a well-defined protocol (smart contract), while reaping the benefits of big data and alleviating the concern over privacy or even espionage.

However, it has been a long-standing question how to open the membership in a distributed system while maintaining the performance, integrity, and correctness of the distributed job results[9]. Traditional blockchains can only process a limited number of transactions per second, and the decentralisation will be compromised if the number of transactions per second is increased. Adding more workload to

existing devices in a network would force the resource-constrained devices which are already exhausted to download and verify updates happening throughout the network continually. Of course, the participants can ease the work burden and increase the reward rate by indirectly mining through mining pools where the minimal computation power is gathered and used collectively by one representative. However, the mining pool participants cannot make a judgment on the way their computation power is used, and as a result the system becomes more centralised, which we would like to avoid. Various approaches have been explored in hopes of solving the blockchain's scalability problem, out of these approaches [10,4,6,11,12,13,14,15], Blockchain sharding [13] is a promising one.

Blockchain sharding is a method that splits the transactions amongst all the participating nodes. By allowing multiple node committees to process incoming transactions in parallel, a sharding-based blockchain protocol can increase its throughput with the increase of the number of participants joining the network. Blockchain sharding has also been used to reduce the storage requirement for non-sharding blockchains [16], which helps the blockchain to be implemented in IoT devices that are lacking storage space. Financial models [17] can be built into the blockchain sharding approach to link the digital labour and the market behaviour with the changes in pay and service prices. Blockchain sharding can also increase the security of blockchain systems by strengthening decentralisation. A recent analysis [18] shows more than 36% of bitcoin nodes are now hosted on only five primary cloud services: This phenomenon can raise a high-security concern, as the cloud services can shut down the bitcoin network suddenly. By employing blockchain sharding, many more users can host blockchain nodes in resource-constrained devices with less processing power than mining-specialised devices and participate as independent miners in blockchain's mining game.

An important aim when designing a blockchain sharding architecture is to reduce the chance for many adversary nodes to be assigned into the same sub-network. Situations in which the adversary does not hold the majority in the whole network but dominates a sub-network should be strictly prevented to safeguard the entire blockchain system. Elastico [19] is known as the first sharding-based consensus protocol designed for blockchain. Elastico has several drawbacks: firstly, all nodes need to reset their identities after every iteration, and all the committees (shards) are rebuilt. Secondly, the latency increases linearly with the increase of the network size because it requires much more time to fill up all the shards by solving enough PoWs. Thirdly, the process of the committee selection can be misled by the adversary: the adversary can pre-compute PoW puzzles. Fourthly, the probability of failure is primarily increased because a small-sized committee (of around 100 members) is required to restrict the running Practical adversary Fault Tolerance (PBFT) [20] in each committee. Elastico has been proved to be insecure in practice [13], and the failure probability can be as high as 0.97 after merely six iterations. Every block must be broadcast to all parties, though Elastico allows each participant only to verify a subset of transactions. OmniLedger [13] improves on Elastico, but it can only tolerate $\mathsf{n}/4$ adversary nodes [21,13]; its security is safeguarded by a large number of participants inside a shard. Due to the limited number of shards, the ceiling of transaction throughput is still relatively low overall. As there are multiple levels of committees, several elections are needed for the system to form this contribution which slows down the system. RSCoin [22] is a sharding-based protocol for helping centrally-banked cryptocurrencies to scale. It is an approach to combine a distributed network with a centralised monetary supply bringing transparency to the traditional centralised banking systems. However, it relies on a trusted source. RSCoin is not adversary fault-tolerant because each shard runs on a two-phase committee protocol. RapidChain [21] improves the security threshold to $\mathsf{n}/3$ adversary node resistant; however, the shard size is still significant.

We previously proposed a new approach [23] that classifies the nodes into different classes and maintains the number of nodes of different classes in every shard. This approach raises the security level of Blockchain sharding to $\mathsf{n}/2$ adversary node resistant, and mostly shrinks the shard size and improves the throughput. However, every shard must have one and only one node from each class, the global rearrangement of the nodes is often needed when the nodes of some classes of a shard go offline. Many new nodes might be left in a pending state when there are not enough nodes from different classes to form new shards. The shard number does not fit into the data flow so that the shard size is only relevant to the number of nodes in the system instead of the real-time workload. It might result in a situation where many shards have no tests to run while the system is idle, and there are not enough shards when the system is busy. Furthermore, the system might be halted by the adversary who has at least $\frac{\mathsf{m}-\mathsf{T}+1}{\mathsf{n}/\mathsf{m}}$ of nodes when the system of n nodes rules that the consensus in a shard sized m is reached with approval from at least T nodes in that shard.

In this paper, we present a flexible $\mathsf{n}/2$ adversary node resistant blockchain protocol that can adjust the class number, shard number, and shard size based on the workload. Our design in this paper serves as the extension of our previous work [23], and it solves the halting problem in that work when the adversary has fewer than $\mathsf{n}/2$ of nodes. We can achieve the security levels of both the $\mathsf{n}/2$ tamper-resistant and $\mathsf{n}/2$ halting resistant. Our protocol not only increases the security level compared to other methods at $\mathsf{n}/3$ security level at most, but it also increases the number of shards and reduces the size of shards. With an increased shard number (more shards running in parallel, and less workload per shard) and a reduced shard size (the less number of nodes in the shard, less communication among nodes in the shard), we strengthen the decentralisation (universal joinability) and increase the performance of the blockchain.

In the remaining of the paper, we first discuss a traditional blockchain sharding approach in section 1.1. We then discuss the $\mathsf{n}/2$ adversary resistant blockchain sharding approach in section 2, and why it may be deadlocked when the adversary has $\frac{\mathsf{m}-\mathsf{T}+1}{\mathsf{n}/\mathsf{m}}$ of nodes.

We then propose a new blockchain sharding approach in section 3, which can recover from a deadlock (global halting) and achieve the n/2 halting resistant security level. In section 4, we show results from a data and performance analysis, discussing the performance (transaction throughput) and the data requirement for nodes.

1.1 | Blockchain sharding Hypothesis

If people are inside of a forest recording the time when trees fall to the ground, there is no need for everyone in the forest to hear and record every fall to maintain the fairness and integrity of the record. The fact that a tree falls and the time when a tree falls is correct when it is recognised by most people around the tree, assuming these people have not colluded. With a sufficient number of people, if they are assigned randomly and distributed to subareas in the forest and are relocated time by time to avoid the accumulation of adversary power, collusion is hard to happen (expected to occur in hundreds of years).

In particular, this proposal is secure when (1) only people assigned to a subarea of the forest can record the information about this subarea. (2) No person can control or predict which subarea they will be assigned to. (3) The assignment follows a globally recognised rule, not by the arbitrary willing of some specific group of superior people. (4) People are periodically reassigned. (5) The number of people in every shard is a sufficient size.

When people assume there is a sufficient number of people acting honestly; people would only need to check what is the typically recognised time of falling for a tree of their interest from the subarea where this tree belongs. It is not necessary for themselves to hear the falling. In this regard, people do not need to have excellent hearing ability when the forest is dense. Instead, they only need to focus on monitoring the subarea in which they are assigned to.

The challenges in this model are as follows: (1) How to distribute people to subareas in a decentralised and unpredictable way? (2) How can people determine if a record of a subarea is made by people assigned to that area? (3) Without monitoring what happened in a subarea, how can people outside know if the majority of the population in that area support a record or not? (4) How large is a population and how many subareas does the forest need to make a "collusion" unlikely to happen?

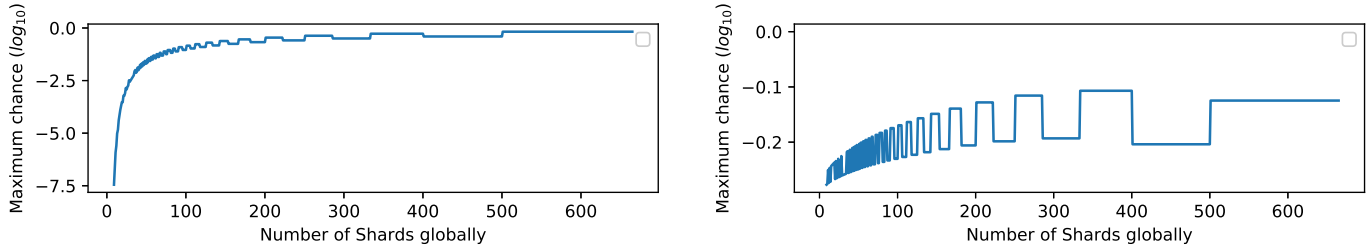
1.2 | Failure Probability

Assuming there exist methods to solve all or some of these challenges above, we calculate the probability of a "collusion" to happen. The probability of obtaining no less than X adversary nodes when randomly picking a shard sized m (m is the number of nodes inside the shard) can be calculated by the cumulative hypergeometric distribution function without replacement from a population of n nodes. Let X denote the random variable corresponding to the number of adversary nodes in the sampled shard. The failure probability for one shard is at most:

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

which calculates the probability that no less than X nodes are adversary in a shard sized m and the adversary has t number of nodes globally in a system of n nodes. Rapidchain [21] is designed with this failure probability.

Figure 1 shows the maximum probability to fail with n = 2000 and m = n/s where s is the number of shards.



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

FIGURE 1 The chance to fail when 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 taken $\mathsf{n}/2$ of nodes. The high failure probability is the main reason why most Blockchain sharding approaches can only withstand up to $\mathsf{n}/3$ of nodes being evil, and the shard size must be enormous.

2 | N/2 BLOCKCHAIN SHARDING PPROACH

2.1 | Hypothesis

We proposed a $\mathsf{n}/2$ adversary node resistant Blockchain sharding approach[23] using a different Hypothesis to build the model. In this hypothesis, nodes are juries inside the courtrooms, and a sentence is made when more than a pre-defined T number of people inside the jury sized m reached a consensus, $\mathsf{T} > \frac{\mathsf{m}}{2}$. In order to make the sentences convincing, people inside the jury come from different occupations; united they represent the whole of society. Assuming the rule says a jury should have five people, a teacher, a social worker, a doctor, a businessman, and a police officer. There are five teachers, five social workers, five doctors, five business-people and five police officers for five juries to run in parallel. The *Jury* hypothesis is different from the *Forest* hypothesis because if the adversary controls two doctors, they cannot live inside the same jury. However, they can be inside the same sub-area of the forest when recording the time of the tree falling.

The challenge of the *Jury* hypothesis is (1) How to give every node a different occupation; and (2) How to make the nodes equally divided into different occupations.

2.2 | Membership management

Let there be a shard acting as the court office which in charge of Jury schedule arrangement. We refer a pending person as the person who reported to the court office but has not been assigned to a court. Pending people can claim their occupations when reporting to the court office, and they cannot change the occupations once they are assigned into a court. The court office periodically publishes the number of pending people in every occupation. Line up the number of pending people in every occupation in ascending order of the time when they claimed an occupation. Every time there comes F pending people in every occupation, the front F people in every occupation is added to the courts. In the meantime, F more courts are formed, and all the jurors are reassigned into different courts. When assigning people, the court office can also be seen as a court. Namely, every time the juror membership is adjusted, the people inside the court office is also reassigned to other courts. If a pending person changes its occupation, it will go to the tail of the other pending queue, losing the position in the original pending queue. Thus, we expect new participants to check the pending queue of every occupation and choose an unpopular occupation to get into the courts quicker. Otherwise, they will be kept in pending status longer until other people fill in unpopular occupations. As new participants line in the shorter queues, the number of people in every occupation is automatically close to each other (tend to be equal in the long run). The person, regardless if they are inside a court or in the waiting queue, should work (generate PoWs in the case of blockchain) in every fixed time window so that the adversary who has half of the overall energy can only have half of the people in the system.

Table 1,2,3,4 and 5 show the procedure of adding people into the system. In this procedure, we rule that whenever there are at least four pending people in every occupation, the adding starts ($\mathsf{F} = 4$). Table 1 shows the pending queue published by the court office in moment one. Table 2 shows the pending queue when the adding conditions are met after moment one and before moment two. Table 3 shows the pending queue in moment two. Table 4 shows the people in the court system at in moment one. Table 5 shows the people in the court system at the moment two where some new people are added.

TABLE 1 The pending queue published by the court office at moment 1. Letters in red colors are the pending people. Following the alphabetical order, the letters also reflect the time when people report to the office.

Occupation	I	II	III	IV	V
	A				
	B				
	C	E			
	D	F	G	H	I
Number of Pending people	4	2	1	1	1

TABLE 2 The pending queue after moment 1 before moment 2. letters in the blue colour represent the people who reported to the office after moment 1 before moment 2. Some nodes will be added to the system afterward because the minimum length of the queues is equal to four (pre-defined adding parameters)

Occupation	I	II	III	IV	V
			U		V
Add to court system ->	A	Q	R	S	T
	B	M	N	O	P
	C	E	J	K	L
	D	F	G	H	I
Number of Pending people	4	4	5	4	5

TABLE 3 The pending queue published by the court office at moment 2. Selected people in Table 2 has been assigned to the system.

Occupation	I	II	III	IV	V
			U		V
Number of Pending people	0	0	1	0	1

TABLE 4 People in the court system at moment 1. There is only one jury running.

Ocp \ Court	1
Occupation I	!
Occupation II	@
Occupation III	#
Occupation IV	$
Occupation V	*

TABLE 5 People in the court system at moment 2. All membership is adjusted with four more juries formed.

Ocp \ Court	1	2	3	4	5
Occupation I	A	C	B	D	!
Occupation II	@	Q	M	F	E
Occupation III	R	N	#	J	G
Occupation IV	S	O	H	K	$
Occupation V	T	*	P	I	L

As can be seen from the adding procedure, if the adversary is not in a very long pending queue, there is no gain for the adversary to change the occupation once it reports to the court office. If it does so, it goes to the tail of another queue, leaving its original place to others. With a sufficient number of participants, only an insignificant number of people will be left outside the system eventually.

2.3 | Failure Probability

Table 6 shows a court schedule table for five courts run in parallel with the jury sized five (five people in different occupations). A refers to an adversary person. H refers to an honest person.

For a s number of courts to be held in parallel, there is an s number of people in each occupation. To manipulate a sentence in of a court, the adversary must gain control of at least T people inside this court. Because the adversary not having more than $\frac{n}{2}$ of people globally and $\mathsf{T} > \frac{m}{2}$, so

$$T \times s > t \tag{2}$$

TABLE 6 Court Jury Schedule

Ocp \ Court	0	1	2	3	4
Occupation I	A	A	A	A	A
Occupation II	H	A	H	A	H
Occupation III	A	H	A	H	A
Occupation IV	H	A	H	H	A
Occupation V	H	H	H	H	A

where t is the number of adversary persons. If the adversary has $\mathsf{A_i}$ number of the person in occupation i; then the chance for the adversary to secure a manipulated sentence is (assuming it place all the adversary person in the front T occupations):

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

To derive the maximised value of $\mathsf{Pr[T]}$, we want $\prod_{\mathsf{i=1}}^{\mathsf{T}} \mathsf{A_i}$ to be maximised because s is the same number (every occupation has s people inside the system). If the adversary has t number of people inside the system (Court Jury Schedule), then

$$t = \sum_{i=1}^{m} A_i \tag{4}$$

To let the value of $\prod_{\mathsf{i=1}}^{\mathsf{T}} \mathsf{A_i}$ maximised, we consider

$$\mathsf{A_i} = \lceil (\mathsf{t/T}) \rceil, \quad \mathsf{i} \in [1, \mathsf{t} \bmod \mathsf{T}] \tag{5}$$

$$\mathsf{A_i} = \lfloor (\mathsf{t/T}) \rfloor, \quad \mathsf{i} \in (\mathsf{t} \bmod \mathsf{T}, \mathsf{T}] \tag{6}$$

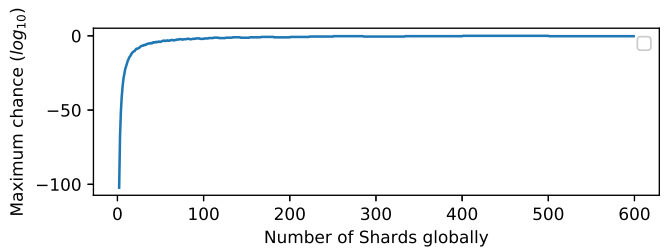
This scenario is maximised because given any positive Integer X,

$$X \times X > (X-1) \times (X+1) = X \times X - 1 \tag{7}$$

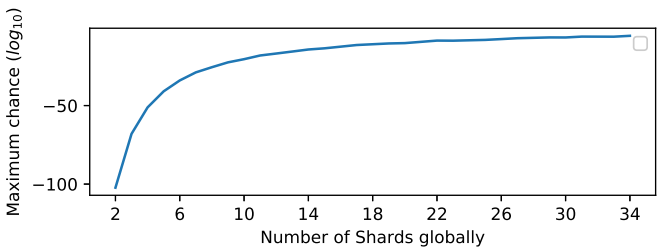
Thus,

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

Though the adversary cannot manipulate a sentence when it does not have T people inside a shard, it can halt a sentence to be reached when it has m-T+1 number of the nodes in a shard. This sentence cannot then be made until the next court (the group of juries is re-selected). Thus, to make the system function more smoothly, we want $\mathsf{T \approx [m/2]}$ whilst meeting the security threshold (e.g., 10^{-6} failure chance). Figure 2 shows the maximum failure chance with different s,$\mathsf{n = s \times m = 2000}$, $\mathsf{T = 0.7 \times m}$ and $\mathsf{t = 1000}$ (1/2 fraction of the overall population).



$\mathsf{s} \in [2, 600], \mathsf{t = 1000 = n/2}, \mathsf{T = 0.7 \times m}$



$\mathsf{s} \in [2, 34], \mathsf{t = 1000 = n/2}, \mathsf{T = 0.7 \times m}$

FIGURE 2 The chance to fail with different s when $\mathsf{n = 2000}$ and $\mathsf{m = n/s}$ where s is the number of shards;

Comparing Figure 1 and Figure 2, the Jury hypothesis largely outperforming the forest hypothesis even when there are $\mathsf{n/2}$ of evil nodes in the Jury hypothesis and only $\mathsf{n/3}$ of evil nodes in the forest hypothesis. When $\mathsf{n = 2000}$, if maintaining a 10^{-6} failure chance, Jury ($\mathsf{n/2}$ evil nodes) can have 33 shards at the same time, while forest ($\mathsf{n/3}$ evil nodes) can only have 10 shards.

2.4 | Drawbacks

The system as a whole is halted when the adversary took $s \times (m - T + 1)$ of nodes, and it places all of these nodes into the same $m - T + 1$ occupations. In this scenario, it is guaranteed that the adversary has $m - T + 1$ of nodes inside every shard. Table 7 shows a example of a system halting, where $m = 5$ and $T = 4$, the adversary took $m - T + 1$ number of nodes in every shard.

TABLE 7 A halted scenerio

Ocp \ Court	0	1	2	3	4
Occupation I	A	A	A	A	A
Occupation II	A	A	A	A	A
Occupation III	H	H	H	H	H
Occupation IV	H	H	H	H	H
Occupation V	H	H	H	H	H

Because the court office (the shard which deals with the node membership) is also stopped from generating further blocks that can be approved by T people inside it, the pending nodes cannot be added to the system. Thus, though the $n/2$ Blockchain sharding method secured the system from tampering when the adversary is below $n/2$, the system can stop functioning anymore when $s \times (m - T + 1)$ of nodes are evil.

3 | FLEXIBLE N/2 BLOCKCHAIN SHARDING APPROACH

In this section, we introduce a flexible $n/2$ Blockchain sharding approach that can dynamically adjust m in shards to defeat the adversary. It takes $n/2$ nodes for the adversary to tamper a record or halt the system eventually.

3.1 | Hypothesis

We use a new hypothesis named *colour* Hypothesis. It is ruled that every node should claim a colour from the colour spectrum when joining in the system — every node inside a shard is categorised to its closet *base colour*. If there is a m number of nodes inside a shard, then there is a m number of *base colours* globally which together represent the colour spectrum as a whole. The same as the *Jury* Hypothesis, a consensus of a shard should be approved by at least a pre-defined T people (T must larger than $0.5 \times m$) inside this shard. Figure 3 shows the example of the *base colour*. Every colour categorisation should have the same number of nodes inside. This hypothesis is different from the *jury* Hypothesis, the *jury* Hypothesis requires a fixed number of occupations, however this hypothesis can regroup a *colour* to a different *base colour* base on the change of the number of *base colours*.

The challenge of *colour* Hypothesis is to make the number of nodes in every categorisation the same as each other, especially with the rapid changes in the number of categorisations.

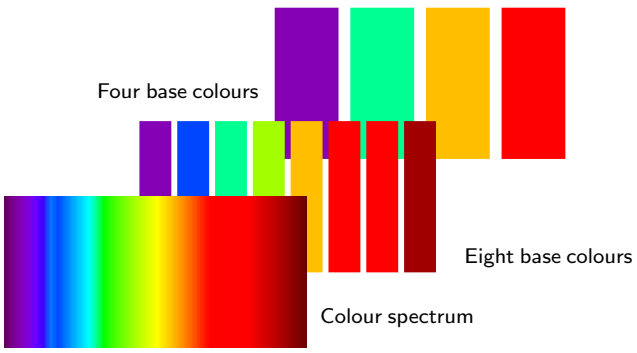


FIGURE 3 The colour spectrum and base colours

3.2 | Failure Probability

Let the adversary nodes has a $\mathsf{t} = \frac{\mathsf{n}}{2} - 1$ number of nodes, s number of shards. The chance for the adversary to take control of a shard in a system which has n nodes and m *base colour* is:

$$Pr[T]_{Max} = (\frac{t}{T \times s})^{T} = (\frac{t/T}{n/m})^{T} \approx (\frac{m}{2 \times T})^{T} \tag{9}$$

As can be seen from Figure 4, T can be adjusted by the number of m when maintaining a fixed threshold failure chance. When m is over 800, $\mathsf{T/m}$ is very close to 0.5 (the adversary needs to take approximately $\mathsf{n/2}$ of nodes to halt the system). We aim to dynamically adjust m to increase the difficulty for the adversary in deadlocking a shard.

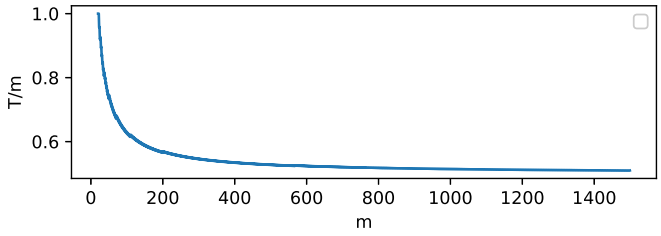


FIGURE 4 T/m for maintaining a 10^{-6} failure chance with different m

3.3 | Model

We run a committee to deal with the membership issues (the same to the court office in *Jury* Hypothesis). The new nodes report to the committee and wait for the assignment. The committee's block contains two sections.

- Pending node section.The node information of nodes which reported to the committee before but have not yet assigned to a shard.
- Member node section. The node information of nodes inside each shard.

When a node wants to join the system, the node will check the information in the 'Pending node' section and the information in the Member node section. It then selects a colour code that is more likely to be added to the system. It then tells the committee the colour code it chooses and its public identity key. When a node's information is added to the 'Pending node' section, it needs to continuously send a PoW to the committee per iteration to maintain its spot in the section until it is assigned. After the node is assigned to a shard, it also needs to present one PoW per iteration. In this way, we prevented the node simulation problem: if the adversary has 50% of the overall power, it can only simulate $\mathsf{n/2}$ of nodes. The system will add the pending nodes to make the number of shards fulfilling the current workload. There is a number of pending transactions indicated in the block header of every shard. We pre-define a standard workload K which is also written in the block header of every shard. When the pending transaction of a shard exceeds $2K$ there should be one more shard added to the system. If the pending transaction is below $\frac{\mathsf{K}}{2}$, then the system should cut this shard. In this way, a shard has at least $K/2$ workload and at most $2K$ workload.

Let there be $\mathsf{m} = \frac{\mathsf{n}}{\mathsf{s}}$ colour categorisations currently.

Let

$$mt = \frac{n + snp}{st} \tag{10}$$

where st is the ideal number of shards in the system based on the current workload, snp is the number of pending nodes that should be added to the system for the system to achieve st number of shards (snp can be zero) and mt is the number of colour categorisations corresponding to that situation. Find a snp number of nodes from the pending nodes that after adding these pending nodes, there can be st number of shards with each shard sized mt. If such snp number of nodes are found, then the shard number is adjusted to st, and the selected nodes are added. The deselected nodes remain in the pending section. If there is more than one snp possible, then it should select the largest snp.

If a node inside a shard did not show a PoW in an iteration, it is labelled as *offline*, and this information is written into the next block header of that shard. All the shards (including the committee) synchronise the block headers of the blocks in other shards, and

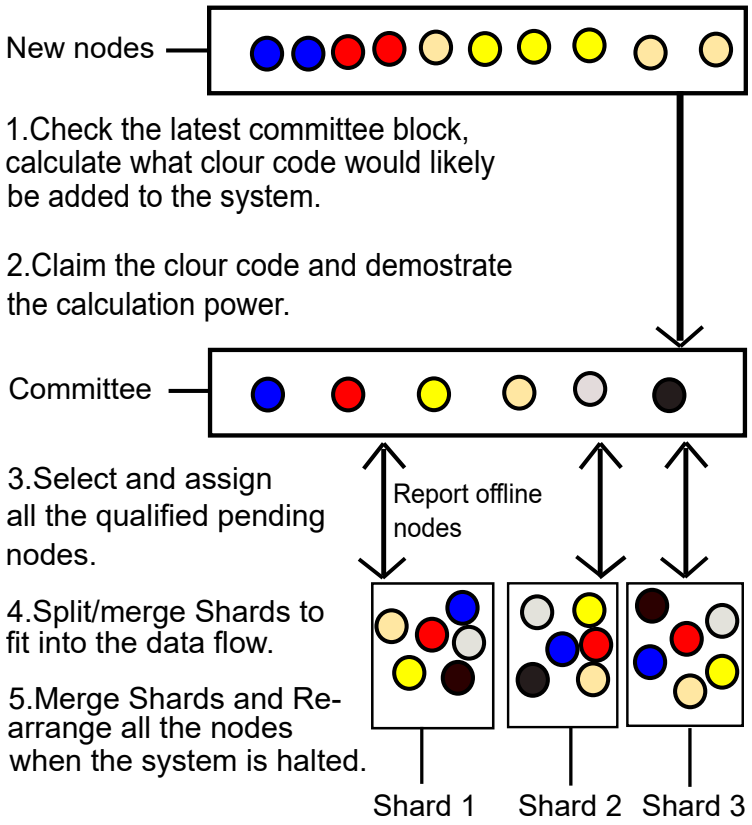


FIGURE 5 The Model.

these blocks should have at least T approvals in the relevant shards. Then, the committee updates its 'member node' section using such information. When a block of a shard is not approved by at least T people inside this shard in a fixed time frame (e.g., *10 minutes*), the block is abandoned. The leader of the next block height will create a new block, and people will vote again for this new block. If the shard cannot reach a consensus in a longer fixed time frame, the shard is halted. All shards know if a shard has halted because they sync the qualified block headers of other shards and the block of the committee. If a shard is halted, the membership of all shards is rearranged (without adjusting the shard number).

If the system is halted (every shard is halted for a fixed time, at the same time), then the nodes use the node information in the committee's latest block to calculate a *st<s* that can make every *base colour* of a *st* number of *base colours* contains mt number of nodes. If such *snp* is not found, then the shard size is directly reduced to one (all the nodes are inside the same shard). If there exists a *snp* (*snp* can be zero), then the system is rearranged to *st* number of shards. By adjusting the number of colour categorisations, we bring more nodes to shards, so that T is respectively reduced. At the extreme, when the shard size is one, the system can withstand the adversary that has a *n/2-1* number of nodes. Especially if it is $\mathsf{s} = 1$, then there is no colour categorisation restriction (number of nodes inside every categorisation can be different). When the system recovers from the halting problem, the shard number can then be increased base on the data flow again. Figure 5 shows an illustration of the model.

3.4 | Node assignment

We refer to the committee as *Shard 0*, and other shards also have their unique shard id. When assigning/rearranging nodes, let $\mathsf{C_{Hash}}$ be the block header hash of the latest block (including the signatures of the nodes) that has been approved by at least T number of nodes in the *Shard 0*. Let *ID[i][j]* be the public identity key of the node in colour categorisation i in the j shard. Let '*hash*' be a hash function that returns a 2^{256} Integer, calculate a *RID*, where

$$RID[i][j] = C_{Hash}\ hash\ ID[i][j] \tag{11}$$

Rank every public key in *ID[i]* by the ascending order of *RID[i]* and then the new assignment will be completed.

3.5 | Pending node selection and node categorisation

Rank all the nodes in the 'Pending node' section and nodes in the 'Member node' section together into a list by the increasing order of their colour code. If node x is originally from the 'Pending node' section then D(x) = 1, otherwise D(x) = 0. Let F[x][y] denote the node assignment scheme for grouping nodes which are selected from the front x nodes in the ranked list; each group should have y people inside it.

$$F[x][y] = Max(F[i \in [y...x-y]][y], select(i, x, y)) \tag{12}$$

where *select* is a function that takes y number of nodes from the node i to node x. The selection rule is: (1) if D(x) = 0 then the node x must be selected. (2) Node x should be selected instead of node $x1$ if node x has been continuously written in the 'Pending node' section longer than node $x1$. Max(a, b) is a function that: (1) calculates the number of nodes from the pending node section in assignment scheme a and b; and (2) returns a or b which has a larger number in the step (1).

After the iterations, the assignment scheme in Max(F[st..n][st]) is the new node assignment scheme. If such an assignment scheme does not exist, we do not add the pending nodes into the system in that round of the mining game. If the system is halted, then the colour categorisation is automatically re-grouped to Max(F[ss..n][ss]), where ss is the largest one in $ss<s$ that make a F[ss..n][ss] exist. The system then automatically re-arranges people following the rule discussed in the *model* section. In this way, the system will eventually recover from the halt, at the extreme, when ss = 1.

3.6 | Shard consensus and Global consensus

Let the block header hash of shard i in the block height BH be *Blockhash[i][BH]*. L_i is the index number of the random leader of a shard i in *BH+1*

$$L_i = hash(Blockhash[i-5][BH], ..., Blockhash[i+5][BH])\ mod\ m \tag{13}$$

Specially,

$$Blockhash[j][BH] = Blockhash[abs(NS-j)][BH], j \in [-4, 0] \cup [NS+1, NS+4] \tag{14}$$

If the shard i did not reach a consensus on the block height BH then

$$Blockhash[i][BH] = Blockhash[i][BH-1] \tag{15}$$

When the leader proposed a block *Bob*, all the nodes inside the shard will sync *Bob* and do the verification. If they approve *Bob*, they will sign *Bob* using their private identity key and attach a PoW which fulfils the difficulty they claimed before. If a node does not approve *Bob*, it will not send the signature, but it still needs to send the PoW. When a block receives a higher than T number of signature as well as the corresponding PoWs, then this block (*Bob*) is approved. If a node did not show the PoW, then this node is considered *offline* and its info is written to the block header of the next block. Later this node's information will be synchronised by the committee, and this node is erased from the 'Member node' section.

A *detect* then *verify* mechanism is used for the global synchronous. The nodes of a shard will inform other shards when more than T nodes approved a block of this shard. Then the nodes of the other shards will download the block header of this block. If a node in this shard is opposing this block, it will inform the other shards. When a conflict is *detected*, the honest nodes in this shard will send the signatures of this block to the other shards for *verification*. Other shards will accept this block if there are at least T number of signatures signed by different nodes in this shard. If there is no conflict reported within a given time frame, then other shards will recognise this block as a finally accepted block of this shard.

All shards can simply verify the signatures received to determine the genuine because all nodes sync the block of the committee; they know all the public keys of nodes. To prevent the adversaries from causing additional verification, the node which sends the global consensus information should sign the data using its private key. In this way, the dishonest node can be labelled as *offline* directly after the verification.

4 | PERFORMANCE ANALYSES AND DATA REQUIREMENT

In Figure 6, we show the minimum percentage of nodes which must be taken by the adversary to halt the entirety of the system with the different number of Shards in two systems of 1500 nodes and 20000 nodes. We can observe from the result that if the adversary attempts to halt the system and it has a great number of nodes, the number of shards can be minimal, vise versa. The adjustment of shard number can be done in real time follows the rules of our approach so that we do not need to assume the adversary having n/2 of nodes as like calculating the failure probability.

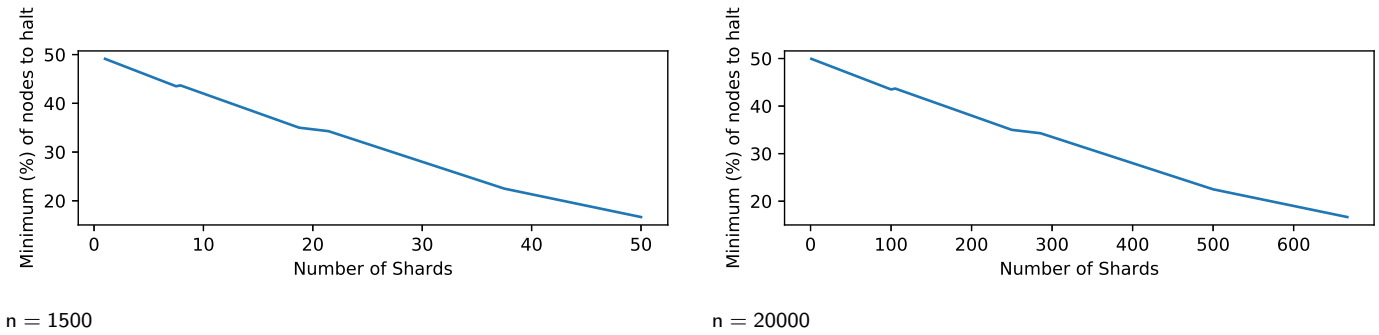


FIGURE 6 Minimum percentage of nodes required to halt a shard

In Figure 7 we show the minimum percentage of nodes must be taken by the adversary to halt the entirety of the system with different number of nodes and different number of shards. As Figure 7 suggests, with a fixed number of nodes, by reducing the number of shards,

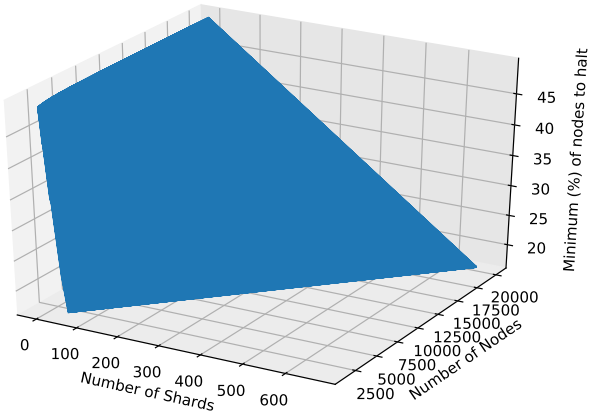


FIGURE 7 Minimum percentage of nodes required to halt the system as a whole

the minimum percentage is increased. Thus, using the halting recovery mechanism introduced in the model, the system will eventually recover from a halt when the number of shards reduced to the point that can defect the adversary. The honest will always win because the honest side has at least *n/2+1* of nodes (the majority), thus meeting the same security level as the Nakamoto blockchain. As the rules discussed, T in every shard is adjusted to keep a 10^{-6} failure chance assuming the adversary has *n/2-1* of the nodes. This mechanism guarantees that the maximum chance to control a shard when the adversary has no more than 50$ of nodes is 10^{-6} with any possible number of shards existed.

In order to get a block accepted by the shard, this block must get T approvals in m nodes of this shard. In Figure 8, we show $\mathsf{T/m}$ in two systems of 1500 and 20000 nodes. For an adversary block to be accepted, the adversary must get T nodes inside the shard. In Figure 9, we show the percentage of nodes (T/M) the adversary must take in order to guarantee an adversary block to be approved.

Figure 10 shows the transaction throughput per iteration with different shards in two systems of 1500 nodes and 20000 nodes (while fulfilling a 10^{-6} failure chance). Figure 11 shows the transaction throughput per iteration with the different number of nodes and shards (while fulfilling a 10^{-6} failure chance). We let every shard block contain *2000* transactions.

Let every block of a shard sized *1Mbytes* (*2000* transactions), every node signs one block per iteration. Let a block header sized *150 bytes*, a transaction sized *500 bytes* and a signature sized *512 bytes* (a SHA256withECDSA signature and *256 bytes PoW* nonce). Let the committee's block $\mathsf{Size}_{\mathsf{block}_{\mathsf{committee}}} \approx 33 * \mathsf{n}$ (contains all the *33 bytes* public identity key of nodes). The quantified data requirement DR in every iteration is:

$$DR = Size_{blockheader} \times s + Size_{transaction} \times 2000 + Size_{signature} \times m + Size_{Block_{committee}} + Size_{Block_{Current\ shard}} \quad (16)$$

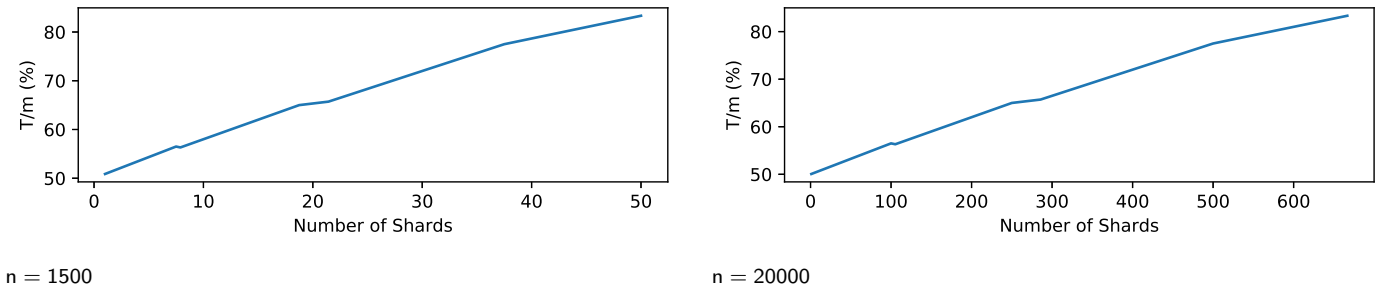


FIGURE 8 T/M

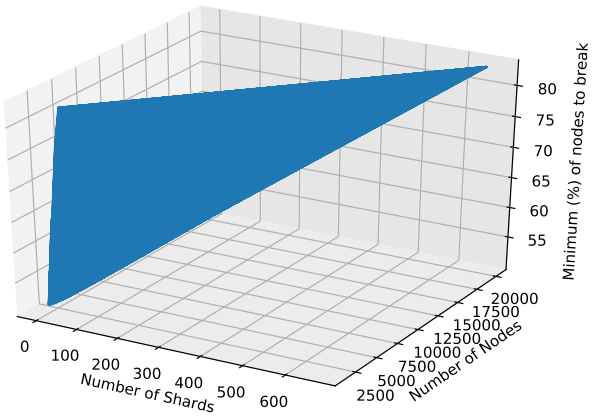


FIGURE 9 Minimum percentage of nodes required to guarantee controlling a shard

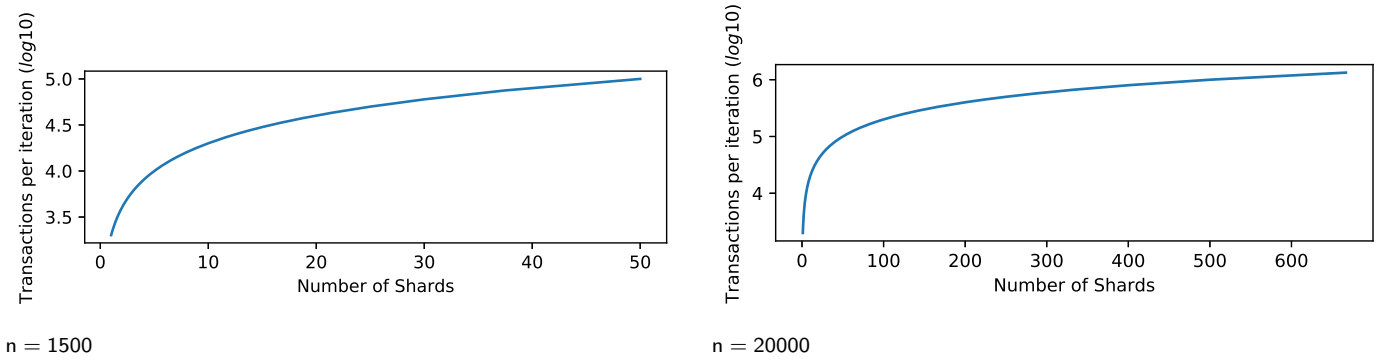


FIGURE 10 Transaction throughput per iteration with 2000 transactions per block

Figure 12 and Figure 13 show the data requirement paired to the situation of Figure 10 and Figure 11.

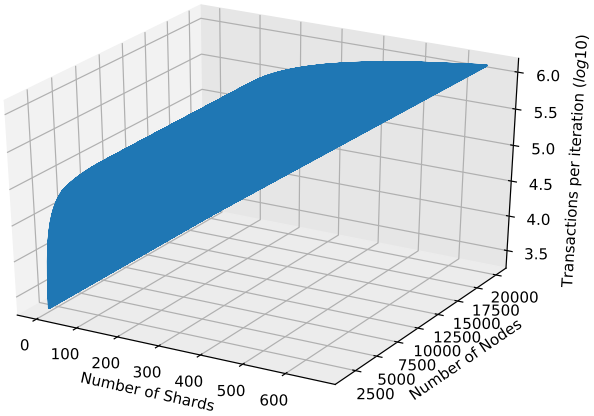


FIGURE 11 Transaction throughput per iteration with 2000 transactions per block

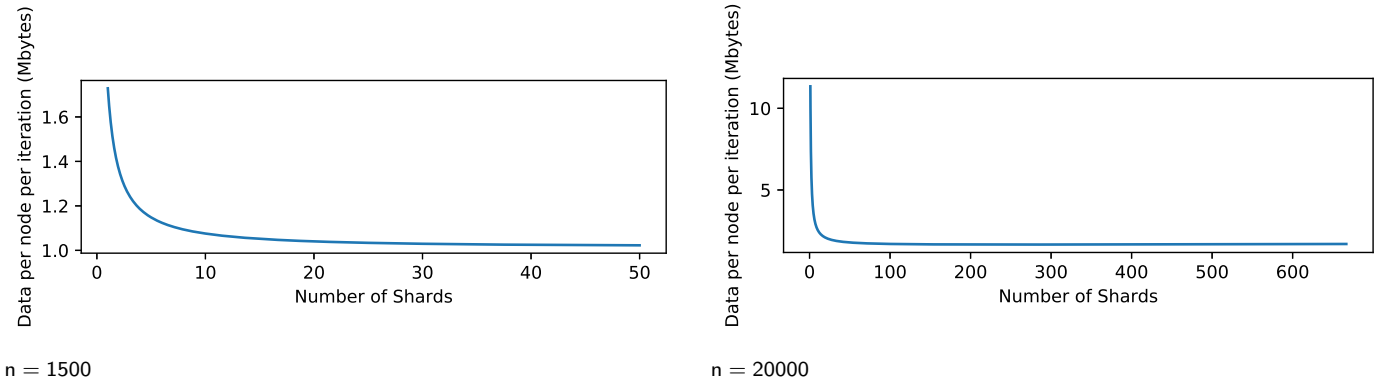


FIGURE 12 Data requirement

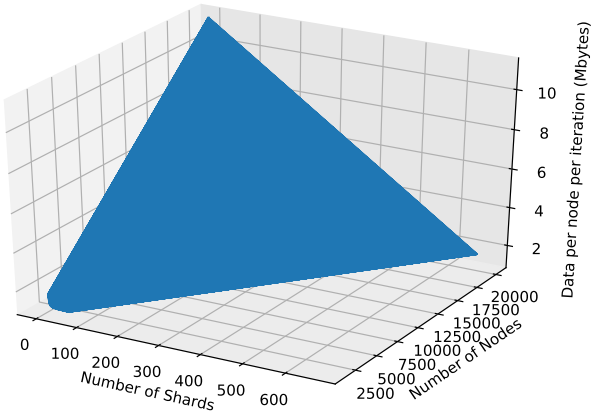


FIGURE 13 Data requirement per node per iteration

5 | CONCLUSION

In this paper, we showed an improved $n/2$ adversary node resistant Blockchain sharding approach; solved the problem of halting when the adversary has a lower than $n/2$ of nodes. In this way, making the Blockchain sharding approach reach both $n/2$ tampering resistance and $n/2$ halting resistant levels. It also makes the shard number become adjustable by the workload, and the workload per shard is balanced. With this improved approach, the performance of DAO systems can be primarily improved while the security of them is sustained. More devices can be added to the system, that extends the usage and potential applications of the blockchain.

We admit that when applying large scale users to this approach, the adjustment of shards maybe frequent because of nodes can join in and leave the shards frequently. We believe such adjustment can be reduced by node regulation especially through the mean of financial tools[17]. In addition, when there are many shards, cross-shard communication can become an issue for the system and causing unnecessary verification or data exchange, the future work of this paper should focus on design and implementing cross-shard communication protocol.

References

1. Nakamoto S, others . Bitcoin: A peer-to-peer electronic cash system. *Working Paper* 2008.

2. Xu Y, Huang Y. MWPoW-multi-winner proof of work consensus protocol: an immediate block-confirm solution and an incentive for common devices to join blockchain. In: IEEE. ; 2018: 964–971.

3. Xu Y, Huang Y. MWPoW: Multiple Winners Proof of Work Protocol, a Decentralisation Strengthened Fast-Confirm Blockchain Protocol. *Security and Communication Networks* 2019; 2019.

4. Popov S. The tangle. *cit. on* 2016: 131.

5. Benčić FM, Žarko IP. Distributed ledger technology: blockchain compared to directed acyclic graph. In: IEEE. ; 2018: 1569–1570.

6. Sompolinsky Y, Lewenberg Y, Zohar A. SPECTRE: A Fast and Scalable Cryptocurrency Protocol.. *IACR Cryptology ePrint Archive* 2016; 2016: 1159.

7. Norta A. Creation of smart-contracting collaborations for decentralized autonomous organizations. In: Springer. ; 2015: 3–17.

8. Barber P, Tomkins C, Graves A. Decentralised site management—a case study. *International Journal of Project Management* 1999; 17(2): 113–120.

9. Ishibuchi H, Nozaki K, Tanaka H. Distributed representation of fuzzy rules and its application to pattern classification. *Fuzzy sets and systems* 1992; 52(1): 21–32.

10. Pass R, Shi E. Hybrid consensus: Efficient consensus in the permissionless model. In: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. ; 2017.

11. Burchert C, Decker C, Wattenhofer R. Scalable funding of Bitcoin micropayment channel networks. *Royal Society open science* 2018; 5(8): 180089.

12. Decker C, Wattenhofer R. A fast and scalable payment network with bitcoin duplex micropayment channels. In: Springer. ; 2015: 3–18.

13. Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly N, Syta E, Ford B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In: IEEE. ; 2018: 583–598.

14. Xu Y, Huang Y. Contract-connection:An efficient communication protocol for Distributed Ledger Technology. *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)* 2019. doi: 10.1109/ipccc47392.2019.8958730

15. Xu Y. Section-Blockchain: A Storage Reduced Blockchain Protocol, the Foundation of an Autotrophic Decentralized Storage Architecture. In: IEEE. ; 2018: 115–125.

16. Xu Y, Huang Y. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access* 2020.

17. Xu Y, Huang Y, Shao J. Anchoring the value of Cryptocurrency. *arXiv preprint arXiv:2001.08154, 3rd International Workshop on Emerging Trends in Software Engineering for Blockchain* 2020.

18. Mariem SB, Casas P, Romiti M, Donnet B, Stutz R, Haslhofer B. All that Glitters is not Bitcoin–Unveiling the Centralized Nature of the BTC (IP) Network. *arXiv preprint arXiv:2001.09105* 2020.

19. Luu L, Narayanan V, Zheng C, Baweja K, Gilbert S, Saxena P. A secure sharding protocol for open blockchains. In: ACM. ; 2016: 17–30.

20. Castro M, Liskov B, others . Practical Byzantine fault tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation* 1999; 99(1999): 173–186.

21. Zamani M, Movahedi M, Raykova M. Rapidchain: Scaling blockchain via full sharding. In: ACM. ; 2018: 931–948.

22. Danezis G, Meiklejohn S. Centrally banked cryptocurrencies. *arXiv preprint arXiv:1505.06895* 2015.

23. Xu Y, Huang Y. An n/2 Byzantine node tolerate Blockchain Sharding approach. *arXiv preprint arXiv:2001.05240, The 35th ACM/SIGAPP Symposium On Applied Computing* 2020.