An n/2 Byzantine node tolerate Blockchain Sharding approach*

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

Yangyu Huang
School of Electronic Engineering and Automation, Guilin University of Electronic Technology
Guilin, Guangxi, China
i@hyy0591.me

ABSTRACT

Traditional Blockchain Sharding approaches can only tolerate up to n/3 of nodes being adversary because they rely on the hypergeometric distribution to make a failure (an adversary does not have n/3 of nodes globally but can manipulate the consensus of a Shard) hard to happen. The system must maintain a large Shard size (the number of nodes inside a Shard) to sustain the low failure probability so that only a small number of Shards may exist. In this paper, we present a new approach of Blockchain Sharding that can withstand up to n/2 of nodes being bad. We categorise the nodes into different classes, and every Shard has a fixed number of nodes from different classes. We prove that this design is much more secure than the traditional models (only have one class) and the Shard size can be reduced significantly. In this way, many more Shards can exist, and the transaction throughput can be largely increased. The improved Blockchain Sharding approach is promising to serve as the foundation for decentralised autonomous organisations and decentralised database.

KEYWORDS

Decentralised ledger, Blockchain, Blockchain Sharding, PBFT

ACM Reference Format:
Yibin Xu and Yangyu Huang. 2020. An n/2 Byzantine node tolerate Blockchain Sharding approach. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3341105.3374069

1 INTRODUCTION

Blockchain Sharding is an approach that implements the idea of *Sharding* [1] in blockchain to increase the transaction throughput without raising the bandwidth and processing requirements of nodes. By allowing multiple committees (Shards) running in parallel, the nodes inside every Shard solely process the data in their Shard, which leads to the system throughput increasing a lot.

Because the essence of a blockchain is in being decentralised and permissionless, and so it should allow as many devices as possible to participate in the system, the idea of Blockchain Sharding is a promising solution to solve the dilemma between increasing performance on one hand and increasing decentralisation on the other hand. Previous work on Sharding has explored various ideas (Elastico [2], RSCoin [3], OmniLedger [4], RapidChain [5]) that can withstand up to 1/4 or 1/3 of network nodes being malicious. These approaches only support a small number of Shards in the system, or, equivalently, they require a large number of nodes in each Shard, both of which impact performance negatively.

In this paper, we propose a new Blockchain Sharding approach that can withstand up to $n/2$ of malicious nodes in the system. Compared to other methods, the probability that the malicious nodes will control a Shard is lower, and only a small number of nodes are required for every Shard to function securely. So that the communication costs inside every Shard are smaller and more Shards can exist in parallel, and that improves the transaction per second globally.

2 BLOCKCHAIN SHARDING HYPOTHESIS

If we are inside a forest recording the time when trees fall, it is not necessarily for everyone to hear every fall of the tree to maintain the fairness of the system. 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 assumed these persons have not colluded. With a sufficient number of people, if they are assigned randomly and completely distributed to subareas in the forest and are reassigned time by time to avoid the accumulation of adversary power, collusion is hard to happen (expected to occur in years). As long as the random and distributed assignment is secured, follow the principle of proportionality, taking control of a subarea requires a significant effort similar to taken the whole system when there is only one area.

In particular, this proposal is secure when (1) only people assigned to a subarea of the forest are legal to record the information about this subarea. (2) any person cannot control or predict which subarea it is about to be assigned in. (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 inside every Shard is large enough.

If the above criteria are fulfilled, and with a sufficient number of honest people, one would only need to check what is the common recognised time of falling for a tree of their interest from the subarea where this tree belongs to, it is not necessary for themselves to hear the falling. In this way, people do not need to have super hearing power when the forest is dense. Instead, they only need to focus on monitoring the subarea where they are assigned to.

*Produces the permission block, and copyright information

ACM ISBN 978-1-4503-6866-7/20/03.
https://doi.org/10.1145/3341105.3374069

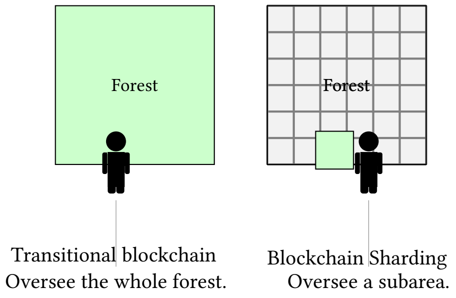


Figure 1: The philosophy of Blockchain Sharding

2.1 Failure Probability

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 group. The failure probability for one committee 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 group of m nodes and t is the number of nodes controlled by the adversary globally.

The hypergeometric distribution depends directly on the total population size (i.e., n). Due to n can change time by time in a permissionless network (open-membership), the failure probability might be affected consequently. To maintain the desired failure probability, each Shard in RapidChain runs a consensus in pre-determined intervals (e.g. once a week), to agree on a new committee size, based on which, the committee will accept more nodes to join the committee in future epochs.

Figure 2 shows the maximum probability to fail with $n = 2000$ 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 $n/2$ of nodes, even if there are only 7 Shards. That is the main reason why all the Blockchain Sharding approaches so far are only withstanding up to $n/3$ of nodes being bad.

If every iteration lasts for 30 minutes, then the time to secure a fail with the 10^{-6} failure chance with $n/3$ of nodes being bad is over 57 years. There cannot be more than 10 Shards when $n = 2000$ and a 10^{-6} failure chance is maintained. The block interval (length of every synchronisation iteration) cannot be shortened; otherwise, it also reduces the time to fail. In Nakamoto blockchain, a block is published in every 10 minute with over 1000 transactions inside the block. Thus, the transactions embedded to the Nakamoto blockchain per hour is over 6000. If we set the same block size for the Blockchain Sharding approaches, with 30 minutes block interval, it can only process over 20000 transactions per hour with 10 Shards. However, considering all the additional designs, the lowered Byzantine fault tolerate rate, the slowed block interval and the failure in the next

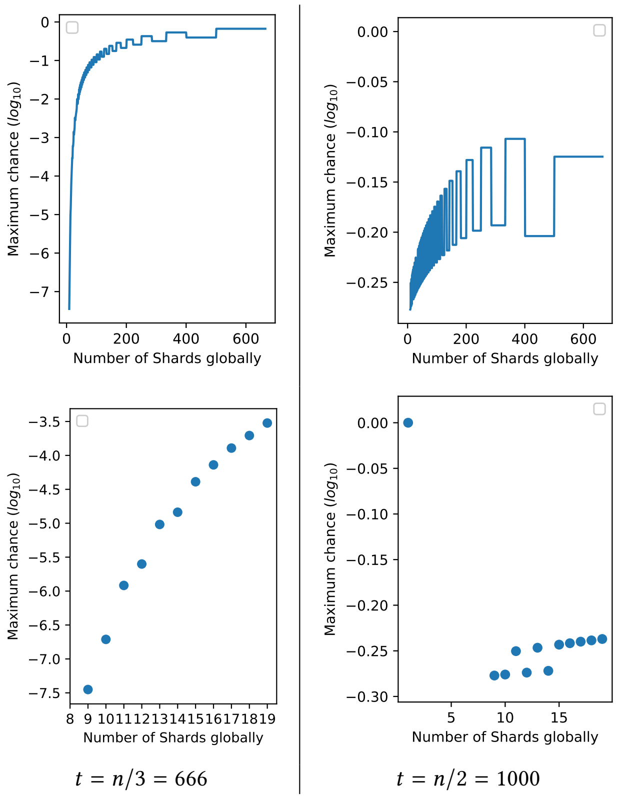


Figure 2: the chance to fail when $n = 2000$, $t = n/3$, $t = n/2$ and $m = n/s$ where s is the number of Shards;

57 years, people may wonder whether a tripled performance worth all the costs.

3 THE N/2 BYZANTINE NODE TOLERATED BLOCKCHAIN SHARDING APPROACH

In this section, we our approach in more detail.

3.1 Our Hypothesis

Instead of recording the time of tree fallings inside a forest, imagine nodes are juries inside the courtrooms. We rule that a sentence is made when more than a predefined T number of people inside the jury sized m reached a consensus (the predefined T must larger than $0.5 \times m$). Every jury should have m people inside, and these people are from m different occupations. For example, let's assume $m = 5$, a jury should have five people: a teacher, a social worker, a doctor, a police officer, and a businessperson. Then, there are at least ten teachers, ten social workers, ten doctors, ten police officers, and ten businesspeople for ten juries to run in parallel. A person can choose an occupation by itself before being assigned to a jury, and it cannot change its occupation inside the jury. There is a court office which in charge of the jury membership issues. Whenever there come m new people in m occupations, the court office will add these people by reorganising the membership of every existing jury and then form a new jury. Jury hypothesis is distinguished from the forest hypothesis because if the adversary controls two social workers, they cannot live inside the same jury; however, they can be inside the same sub-area of the forest when recording the tree falling time. The difficulty in fulfilling the Jury hypothesis is to divide the people into m occupations equally.

3.2 Jury Membership

A new participant needs to choose an occupation and report to the court office before the court office can assign it to a jury (Shard). When adding new people to the court system, the court office should give preference to the people of seniority (the people who reported to the office earlier but has not yet been added to the system). Assuming $m = 5$ and six new people in five different occupations are waiting to be added to the system, person A and person B are in the same occupation. The court office would add person A to the system with the other new people in other occupations if person A reported to the office earlier than B. Person B will then be put in the pending status until there comes four new people in the other four occupations.

The court office periodically publishes the number of people in every occupation who reported to the office and is waiting to be assigned to a jury. So that when a new participant decides its occupation, it will check the pending queue of every occupation and choose an unpopular occupation to get into the system quicker. If it lines in a long queue, it will need to wait until there come enough people to fill in the shorter queues. Because new participants tend to line in the shorter queue, the number of people in every occupation is automatically close to each other (tend to be equal in the long run). If people change its occupation after reporting to the court office, it will be placed to the tail of the pending queue of the new occupation. Thus, changing an occupation wastes the position in the original pending queue.

The person, regardless if it is inside a jury or in the waiting queue, should work (generate $PoWs$) in every fixed time window. Thus the same as the other blockchain sharding models, 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, adding starts. Table 1 shows the pending queue published by the court office at the moment one. Table 2 shows the pending queues when the adding conditions are meet after moment one before moment two. Table 3 shows the pending queue at the moment two. Table 4 shows the people in the court system at the 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 queues published by the court office at moment 1. Letters in red colors are pending people.

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

Table 2: The pending queues after moment 1 before moment 2. letters in blue color stand for the pending people who reported to the office after moment 1 before moment 2. Because the minimum length of the queues reached four (predefined adding parameter), the front four people of every queue should be added to the system in moment 2.

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

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

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

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

Court / Ocp	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.

Court / Ocp	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 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. Then, it still needs to wait until there comes more people in other queues before it can be added to the system.

3.3 Failure Probability

Table 6 shows a court schedule table for ten courts run in parallel with the jury sized five, and ten people in each of every occupation. In Table 6, A refers to the adversary person, H refers to the honest person.

For a s number of the jury meeting to be held in parallel, there is a s number of people in each occupation. Let the adversary has

Table 6: Court Jury Schedule

Ocp \ Court	0	1	2	3	4	5	6	7	8	9
Occupation I	A	A	A	A	A	H	H	H	H	H
Occupation II	H	A	H	A	H	A	H	A	H	A
Occupation III	A	H	A	H	A	H	A	H	A	H
Occupation IV	H	A	H	H	A	A	H	A	H	A
Occupation V	H	H	H	H	A	A	A	A	H	A

A_i number of the person in Occupation i; then the chance for the adversary to secure a manipulated sentence is (assuming without loss of generality that the adversary puts all its nodes into the front T occupations)

$$Pr[T] = \prod_{i=1}^{T} \frac{A_i}{s} \tag{2}$$

where T is the number of the person the adversary must take in a jury to manipulate the sentence.

To derive the maximised $Pr[T]$, we want $\prod_{i=1}^{T} A_i$ to be maximised because s is the same. Let the adversary has AD number of people inside the system (Court Jury Schedule), then $AD = \sum_{i=1}^{m} A_i$. To let the value of $\prod_{i=1}^{T} A_i$ maximise, we consider

$$A_i = \lceil (t/T) \rceil, \quad i \in [1, t \bmod T] \tag{3}$$

$$A_i = \lfloor (t/T) \rfloor, \quad i \in (t \bmod T, T] \tag{4}$$

This scenario is the maximised because given any positive integer X,

$$X * X > (X-1) * (X+1) = X * X - 1. \tag{5}$$

Thus,

$$Pr[T]_{max} \approx (\frac{AD}{T * s})^T \tag{6}$$

If $T = m$ (all the people in the jury should reach the same verdict when making a sentence), then

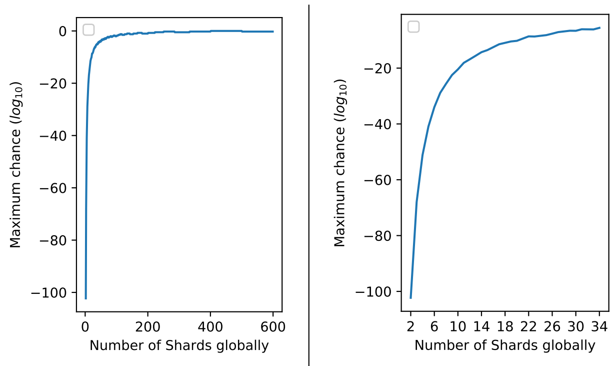
$$Pr[T = m]_{max} \approx (\frac{AD}{s * m})^m \tag{7}$$

Let $AD = \frac{s*m}{2}$ (half of the overall population) then

$$Pr[T = m]_{max} \approx (\frac{1}{2})^m \tag{8}$$

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. Then this sentence cannot be made until the next court (the group of juries are re-selected). Thus, to make the system function more smoothly, we want $T \approx \lceil m/2 \rceil$ while meeting the security threshold (e.g. 10^{-6} failure chance). Figure 3 shows the maximum failure chance with different s, $n = s * m = 2000$, $T = 0.7 * m$ and $AD = 1000$ (1/2 fraction of the overall population).

As can be seen from the result, when there are ten Shards and n/2 people being evil, the failure chance is below 10^{-20}, which significantly outperformed the RapidChain at below 10^{-6} when it has ten Shards and only $n/3$ nodes being evil. If we set the block interval to be 30 *minutes*, then it takes over 10^{15} years to fail the system. If it is 10 *minutes* (the same as Nakamoto blockchain), it still takes over 10^{14} years to fail. If we maintain the 10^{-6} failure chances at this circumstance with $T = 0.7 * m$, then there can be 33 Shards at the same time.



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

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

4 POTENTIAL USAGE

It has been a long-standing question for how to open the membership in a distributed system while maintaining the performance of distributed jobs as well as the integrity and correctness of the job results [6, 7]. How to enable nodes in the different background to participate and tolerant them go offline without notice while stabilised the system as a whole [8]. By increasing the Byzantine-fault-tolerant rate as well as the performance of the Blockchain Sharding approach, the improved Blockchain Sharding approach may solve these standing problems. For example, a data grid and distributed database can allow their users to be a part of the processing system. IoT devices can be governed decentralised by the transparent rule of laws [9–11], in this way, waived the concern over privacy or even espionage for smart home assistants.

5 CONCLUSION

In this paper, we discussed a new Blockchain Sharding approach that maintains the system integrity when there are at most $n/2$ fraction of adversary nodes. Compared to the previous work, the required number of nodes per Shard is much lower and more Shards are allowed to exist with the same security threshold.

REFERENCES

[1] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):8, 2013.
[2] 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.
[3] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. *arXiv preprint arXiv:1505.06895*, 2015.
[4] 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.
[5] 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.

[6] Hisao Ishibuchi, Ken Nozaki, and Hideo Tanaka. Distributed representation of fuzzy rules and its application to pattern classification. *Fuzzy sets and systems*, 52(1):21–32, 1992.
[7] Roy Friedman. Fuzzy group membership. In *Future Directions in Distributed Computing*, pages 114–118. Springer, 2003.
[8] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
[9] Ioannis Psaras. Decentralised edge-computing and iot through distributed trust. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 505–507. ACM, 2018.
[10] Andreas Bogner, Mathieu Chanson, and Arne Meeuw. A decentralised sharing app running a smart contract on the ethereum blockchain. In *Proceedings of the 6th International Conference on the Internet of Things*, pages 177–178. ACM, 2016.
[11] Jozef Mocnej, Winston KG Seah, Adrian Pekar, and Iveta Zolotova. Decentralised iot architecture for efficient resources utilisation. *IFAC-PapersOnLine*, 51(6):168–173, 2018.