Improving Scalability in Blockchain: Sharding

Sajan Maharjan (20182095)
(thesajan@postech.ac.kr)

1. Introduction:

The year 2017 saw the rise of blockchain and cryptocurrency. Prior to that Bitcoin and cryptocurrencies were limited to software developers, geeks and cyberpunks. About 3 billion dollars were invested into different blockchain systems in the year 2017. Although blockchain based cryptocurrencies and payment systems promise a future of distributed, decentralized, trustless payments without the need of a 3rd party, scalability issues in blockchains have presented a hindrance before they can replace traditional payment solutions such as VISA, MasterCard and PayPal. Currently two of the most popular blockchains in use- Bitcoin and Ethereum can only process transactions at the speed of 3 to 7 TPS and 15 to 20 TPS respectively. On the other hand, existing traditional centralized payment systems such as VISA and MasterCard process transactions at the speed of 1200 to 24000 TPS. This paper explores various possible available exploits to enhance the transaction processing capability in blockchain based payment systems. Particularly, one of the widely proposed scaling solutions- sharding has been described in detail.

2. Problem:

As mentioned in the previous section, the problem of blockchain lies in its transaction processing capability. Transactions are collected in blocks. There is a limit on the size of blocks in Bitcoin which defines an upper bound on the number of transactions that added in a single block. While there is no inherent block size limit in Ethereum, there is gas limit of 6.7 million per block. Also, block generation in Bitcoin, takes an average of 10 minutes and requires at least 6 confirmations while an Ethereum block is generated every 15-20 seconds and requires at least 12 confirmations to be permanently written on the blockchain. This scenario presents a difficulty when considering the use of cryptocurrency in day-to-day life.

3. Proposed Solutions:

A variety of solutions have been proposed to tackle the blockchain scalability problem. These scalability solutions can be broadly categorized as- Layer 1 solutions and Layer 2 solutions. Layer 1 solutions involve forking- which is a change or update to the core blockchain protocols or divergence from the previous version of blockchain. Examples of Layer 1 solutions include- increase in block size, change of consensus algorithms, sharding, etc. Layer 2 solutions do not require forking, but make changes to the layer built on top of the base blockchain layer. Off-chain and side-chain are examples of Layer 2 solutions.

Bitcoin Classic and Bitcoin Cash, are forks of the original Bitcoin protocol, which aims to improve scalability by increasing the block size of Bitcoin block. The developers of Bitcoin did not expect its widespread use as is now and had opted for a block size of 1MB to prevent spam transactions. In the case of Bitcoin classic, the block size was increased to 4MB which could hold 4000-8000 transactions per block and thereby increasing transaction processing speed of the block. Segregated Witness is another soft fork to the Bitcoin blockchain which reduces the size and repetition of signature in transactions, thereby allowing more transactions to be included in the block. Since, block size increase is a vertical scaling solution i.e. nodes require more processing to produce a single block, only powerful nodes will continue operating and cause centralization around powerful nodes in the network.

Another solution that can be used to address scalability, involves the change of consensus protocols. Currently, Bitcoin and Ethereum employ Proof-of-Work consensus algorithms. In this algorithm, consensus becomes more and more difficult to achieve as the size of network grows. Participation of every participating node in the transaction validation makes transaction processing slower. Another set of consensus algorithms called Proof-of-Stake and Delegated Proof-of-Stake establish consensus via a small group of stakeholders/delegates. Approval of blocks from a certain fraction of the delegates confirms block generation. As a result, it blocks are generated quicker and require less confirmation time. An example of a blockchain employing DPoS consensus algorithm is the EOS blockchain that has 21 delegates and processes 3000 transactions per second.

The literal meaning of a shard means a piece or fragment. Sharding involves splitting a blockchain into different sections/parts called shards, each of which can independently process transactions. Sharding is one of the horizontal scaling solutions i.e. it does not require additional processing by nodes rather scaling capabilities increase with the increase in the number of nodes. The sharding process has been discussed in detail in the later contents of this paper.

Off-chain, a layer 2 solution, cuts down data processing on the blockchain by running computations off-chain. Transfer agreement, escrow mechanisms, coupon based payment channels can be used in which both parties locally maintain their copies of transactions/records off-chain and only record transactions into the blockchain while opening and closing of deal between two parties. This solution is applicable if two parties regular transact/deal with one another. Say A buys a cup of coffee from B on a daily basis. Instead of writing daily records of each transaction between A and B, an off-chain solution can be used in which A and B record transaction into the blockchain only at the beginning and record daily transactions in locally maintained ledgers. In the event that A and B want to close accounts between them, then they both verify one another’s local ledger and close the deal by writing the final transaction to the blockchain. Off-chain transactions are instantaneous. Another layer 2 solution is called a sidechain, in which a separate blockchain is attached to the main chain that facilitates trading of assets both ways. Sidechains improve transactions scalability adhering to the fact that they are independent of the main chain, can run in parallel with the main-net, and does not boggle the main-net. For a given state of Ethereum blockchain, assets and tokens can be moved to the sidechain in order to de-congest Ethereum network. This allows block validation to be done on the sidechain and not on the main net. In this way, sidechains can take all the work and at some point in time give its state to Ethereum, so the previous state can be updated with the current state.

4. Sharding Model:

As mentioned in the previous section of this paper, Sharding involves the partitioning of network into smaller committees, each of which can independently process a disjoint set of transactions or "shards". Sharding in blockchain derives its concept from "sharding" in databases. A shard is a horizontal portion of a database, with each shard stored in a separate server instance. This spreads the load and makes the database more efficient. In the case of blockchain, when sharding is implemented each node will only have a part of the data on the blockchain and not the entire information. Nodes that maintain a shard, maintain information only on that shard in a shared manner, so within each shard, decentralization is still achieved. ELASTICO is the first secure sharding protocol for open blockchain proposed by the researchers at the National University of Singapore, which makes use of Proof-of-Work and Byzantine consensus algorithms. ZILIQA, a new blockchain platform designed to scale in transaction rates was introduced in 2017 based on the working principles of ELASTICO. The sharding model proposed by these protocols is explained below.

The sharding model proposed by ELASTICO and ZILIQA, make the following assumptions. Given that there are n miners in the network, the network can tolerate upto $f <= n/4$ byzantine adversaries. Honest nodes are reliable during protocol runs and failed or disconnected nodes are counted as byzantine nodes. Each shard outputs a different set of transactions. Finally, the network is assumed to be partially synchronous i.e. any broadcasted message will reach a node within a bounded delay of δt seconds.

The sharding protocol consists of two types of nodes- a directory service committee node and a regular shard node. Each shard of size c and the directory service committee run a byzantine consensus protocol to agree on a set of transactions. Each shard and the directory service committee is assigned a leader. Sharding protocol proceeds in epochs or rounds. Each epoch involves-

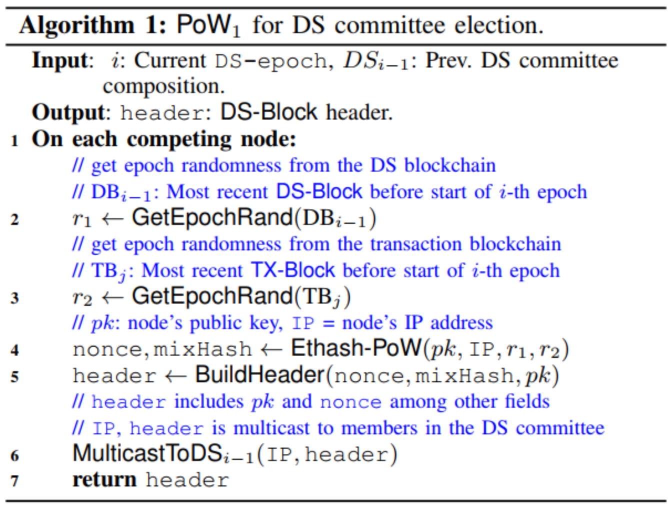
- Identity Establishment and Committee Formation
- Shard Assignments
- Sharding Logic Publishing
- Transaction Sharding and Building Micro-Block Consensus
- Final Block Proposal and Consensus
- Block Confirmation and Epoch Randomness Generation

4.1 Identity Establishment and Committee Formation

In this step, nodes are required to establish a pseudonymous identity using their public key and IP address. Identity establishment is a required step for both the DS committee nodes as well as regular shard nodes. Nodes are required to find a PoW solutions corresponding to their identity in order to participate in mining. A PoW solution can be of the following form-

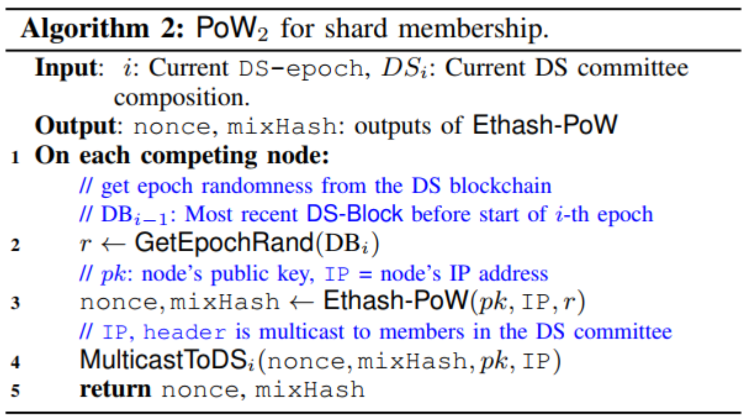
$$H(\text{epochRandomness} \,||\, \text{IP} \,||\, \text{PK} \,||\, \text{nonce}) <= 2^{\gamma-D}$$

γ is the length of the hash output and D defines the number of leading zeros required. This PoW solution prevents Sybil nodes from participating in the network. Epochrandomness defines a random number for the given epoch which prevents byzantine adversary from precomputing the hash value output before a specified time or epoch. As previously mentioned, there are two sets of nodes in the sharding protocol- directory service nodes and regular nodes. Sharding protocol first proceeds with the election of directory service nodes. DS Committee has a fixed number of nodes and the first N_O nodes to solve the PoW solutions are selected into the DS Committee. A leader is elected from one of the nodes and a new leader or member is added in each epoch churning the oldest member in the DS committee. Once the election of the DS nodes is complete, election of regular nodes take place. The following figures show the algorithm for DS committee election and regular shard node in ZILIQA.



```
Algorithm 1: PoW$_1$ for DS committee election.
  Input: $i$: Current DS-epoch, $DS_{i-1}$: Prev. DS committee
         composition.
  Output: header: DS-Block header.
1 On each competing node:
      // get epoch randomness from the DS blockchain
      // DB$_{i-1}$: Most recent DS-Block before start of $i$-th epoch
2     $r_1 \leftarrow$ GetEpochRand(DB$_{i-1}$)
      // get epoch randomness from the transaction blockchain
      // TB$_j$: Most recent TX-Block before start of $i$-th epoch
3     $r_2 \leftarrow$ GetEpochRand(TB$_j$)
      // $pk$: node's public key, IP = node's IP address
4     nonce, mixHash $\leftarrow$ Ethash-PoW($pk$, IP, $r_1$, $r_2$)
5     header $\leftarrow$ BuildHeader(nonce, mixHash, $pk$)
      // header includes $pk$ and nonce among other fields
      // IP, header is multicast to members in the DS committee
6     MulticastToDS$_{i-1}$(IP, header)
7     return header
```

Fig 1. Algorithm for DS Committee election



```
Algorithm 2: PoW$_2$ for shard membership.
  Input: $i$: Current DS-epoch, $DS_i$: Current DS committee
         composition.
  Output: nonce, mixHash: outputs of Ethash-PoW
1 On each competing node:
      // get epoch randomness from the DS blockchain
      // DB$_{i-1}$: Most recent DS-Block before start of $i$-th epoch
2     $r \leftarrow$ GetEpochRand(DB$_i$)
      // $pk$: node's public key, IP = node's IP address
3     nonce, mixHash $\leftarrow$ Ethash-PoW($pk$, IP, $r$)
      // IP, header is multicast to members in the DS committee
4     MulticastToDS$_i$(nonce, mixHash, $pk$, IP)
5     return nonce, mixHash
```

Fig 2. Algorithm for regular shard node selection

4.2 Shard Assignments

Once the required number of DS nodes and regular shard nodes are elected, the protocol now requires assigning regular shard nodes to different shards. One of the straightforward method to assign nodes to different shard involves comparing the nonce values of the PoW solution in step 1 and then mapping the nonce values to a corresponding shard number. Another approach to shard assignment is- given that there are 2^s shards in the network, use the last s-bits of the

PoW solutions and map them to the corresponding 2^s shards. Since computing the hash solution for PoW involves a random function, we can assume that the last s-bits of the PoW solution is also random. This in turn ensures that assignment of a node to a particular shard is also random and helps in restraining the number of malicious nodes in a shard to less than 1/3. A leader for an individual shard can be selected by comparing the nonce values of nodes within that shard.

4.3 Sharding Logic Publishing
In this step, the DS nodes publish information about node identities, connection information, list of selected nodes in each shard in a public channel. Sharding logic for a transaction is also defined in this phase i.e. since each individual shard needs to agree on a different set of transactions, a logic is necessary that defines which shard stores the transaction details. Assuming that there are 2^s shards in the network, use the rightmost s-bits of sender/receiver address to determine which shard to store at.

4.4 Transaction Sharding and Building Micro-Block Consensus
At this point in the sharding protocol, it has already been determined as to which shard stores which transactions. Transactions are collected by nodes in a shard and sent to the shard leader. Collected transactions are then grouped into a block called the micro-block. Once the leader has prepared a micro-block, the prepared micro-block needs to be approved by 2/3 of the members of the shard. Therefore, consensus takes place among the nodes in the shard. Upon success, the shard leader sends the micro-block header (containing transaction hashes) and bitmap object (signifying signatures of members of the shard) to some of the DS nodes.

4.5 Final Block Proposal and Consensus
DS nodes collect the micro-block headers sent by different shard leaders and forwards them to the DS committee leader. Then DS committee leader validates each of the micro-blocks and combines them to generate the final block. The final block proposed by the DS committee leader needs to be approved by at least 2/3 member of the DS committee. Once approved, the DS committee leader publishes a final block header and bitmap object (containing signatures of the members of the DS committee) and sends it to few of the nodes in each shard.

4.6 Block Confirmation and Epoch Generation
After nodes in individual shards have received the final block header sent by DS committee leader, the nodes then confirm the signature in the bitmap object by comparing it against the signature of DS nodes, which was published in the public channel in step 3. Individual nodes also check the transaction hash in the final block header against the transaction hash in the micro-block proposed by that shard. If the transaction hash matches, the transaction data for the proposed micro-block is published in the local shard. Once transactions have been finalized, corresponding account and global states are updated for the sender and receiver.

As mentioned earlier, at the end of each epoch, some random value is generated (epochRandomness) to prevent malicious nodes from pre-computing hash values. There can be cases when the elected leader is a byzantine node, such that they can delay or drop messages from honest nodes. Due to the possibility of such events, leaders are changed periodically. Leader of each shard is changed after every micro-block confirmation whereas leader of DS committee is changed after every final block confirmation. If size of DS consensus group is *n*, then epoch lasts for generation of *n* final blocks with leader change after every final block. Each final block consists of a single micro-block proposed from each shard.

One cumbersome issue while using sharding is the existence of cross-shard transactions. Given a transaction from A to B, if the account address of both A and B are within the same shard, there is no need to communicate with another shard and transaction is finalized with a single shard. However, consider the case of a cross-shard transaction i.e. Alice (address in shard #1) wants to send some tokens to Bob (address in shard #2), requires updating state and transaction information in both the shards. Cross-shard transactions are implemented using two approaches- synchronous and asynchronous. In synchronous approach, state transition related information is updated in both the shards at the same time. In the case of asynchronous approach, state transition information is updated one after another. The validity of cross-shard transactions are questionable in the case of forking in individual shards. However, given that the network has less than 1/3 of the total nodes as malicious, the probability of fork is extremely low.

Let us suppose that by an infinitesimal probability more than 1/3 of malicious nodes collude on a shard. In such cases, they can obfuscate an invalid B on top of a valid block C which involves a cross-shard transaction with another shard. i.e.

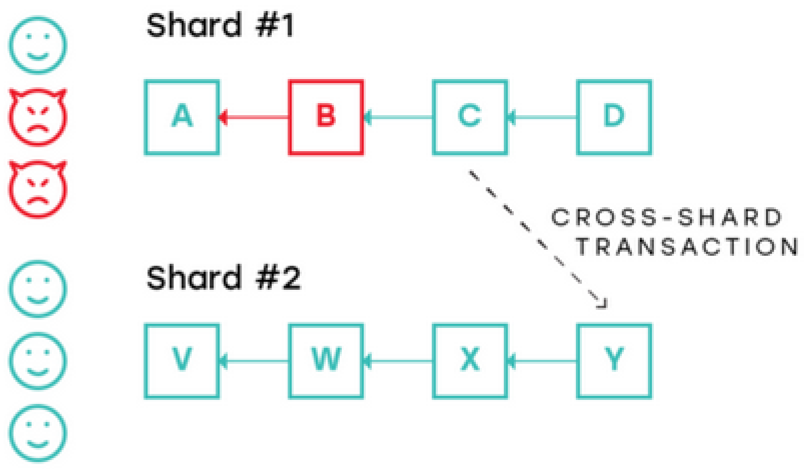


Fig 3. Cross-shard transaction with a malicious shard

In the scenario presented above, shard #2 can confirm that block C is valid because it only has to deal with block C as the cross-shard transaction. In such cases, malicious blocks can be generated at will in a local shard, while only presenting cross-shard transactions as valid to other concerned shards. To prevent such malicious blocks generation, undirected graphs can be used, in which a shard is connected to few another neighboring shards. i.e.

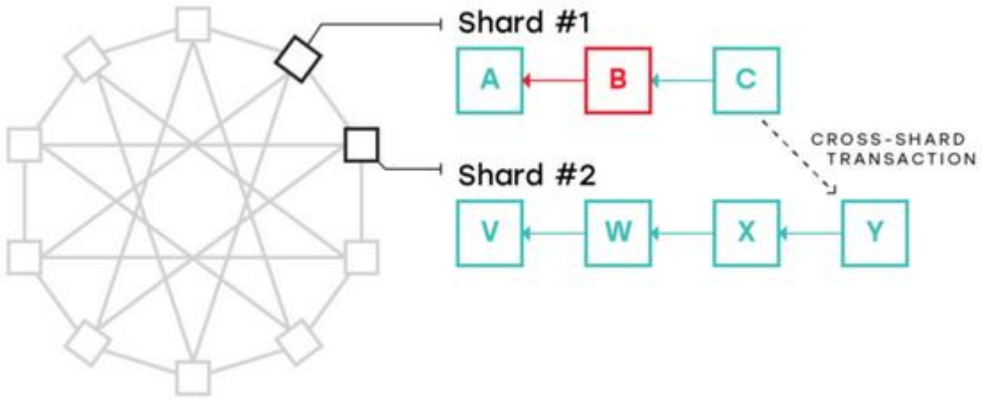


Fig. 4 Use of undirected graph between shards

The use of undirected graphs require that each block generated by shard nodes needs to be validated and checked by neighboring shard nodes that it is connected to. This will prevent invalid blocks to exist in a single shard. Also, the undirected graph allows cross-shard transactions between neighboring shards in a easy manner. If a cross-shard transaction is needed between shards that are not neighbors, such transaction is routed through multiple shards.

The use of undirected graph prevents a single malicious shard from generating invalid blocks. Consider the rare case in which more than one shard colludes on generating invalid blocks.

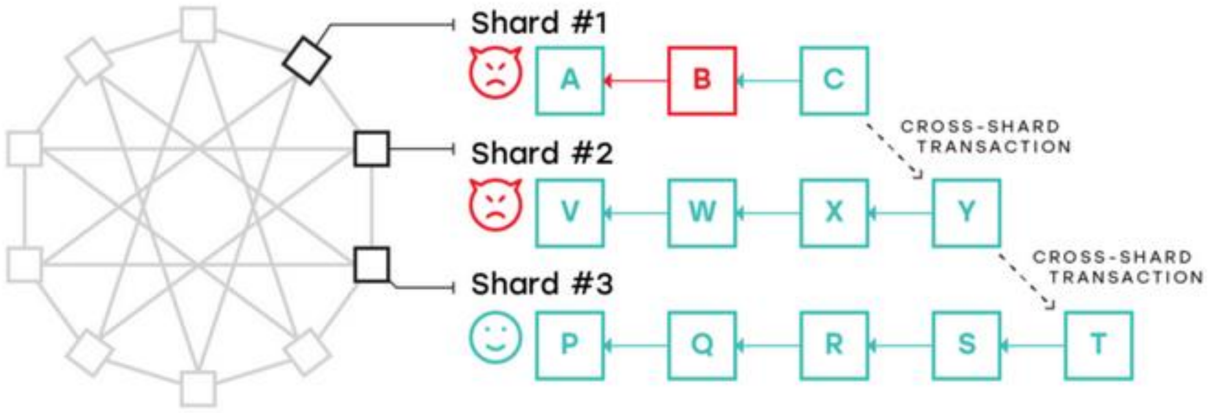


Fig. 5 Multiple malicious shards colluding the network

In the scenario presented above, shard #1 has an invalid block B and shard #2 colludes with shard #1. Shard #3 can only validate blocks from shard #2 but not shard #1, since there is no connection between them. This is a very rare scenario given that shard assignment is completely random. However, such situations can be resolved by using the Fisherman approach. This idea suggests that whenever there is a communication between shards involving cross-shard transactions, a single honest node can issue a challenge or provide a proof that a particular block is invalid for a certain period of time. This approach secures a shard from malicious actors given that there is at least one honest node in the shard.

The use of sharding in blockchain networks has significant improvements in the transaction processing capability of blockchain. Since the transaction data is not transmitted (but just the micro-block header and final block header) between shards or DS nodes, transaction details reside locally on corresponding shard. Due to this, nodes can be maintained in the network without increasing the bandwidth requirement of each node. Also, cryptographic algorithms such as EC-Schnorr can be used to generate multi-signatures that requires less size but achieves higher speed, along with sharding to enhance transaction processing speed. ELASTICO, a secure sharding protocol for open blockchains, shows that throughput scales up linearly in the computation capacity of the network. ZILIQA, another blockchain platform that implements sharding to achieve transaction processing capability, claim that given a network size of Ethereum, they can achieve a scalability of 1000 times in their network.

5. Conclusion:

In this paper, we have introduced the problem of scalability in existing blockchains. To tackle such scalability issues, a number of solutions have been proposed. In particular, sharding has been acquiring a great interest among developers and the blockchain community. Sharding solutions to the blockchain scalability problem was introduced in this paper, the details of its design and how cross-shard transactions are executed and how byzantine adversaries can be prevented from fabricating invalid blocks in the network was discussed. At the end of this paper, the increase in transaction processing speed after implementing sharding was mentioned. Sharding holds a great promise in the blockchain domain and is being researched with deep interest right now.

6. References:

[1] ZILIQA Team, The Ziliqa Technical Whitepaper, https://docs.zilliqa.com/whitepaper.pdf

[2] L. Luu et al, A Secure Sharding Protocol For Open Blockchains, http://delivery.acm.org/10.1145/2980000/2978389/p17-luu.pdf?ip=141.223.124.8&id=2978389&acc=ACTIVE%20SERVICE&key=0EC22F8658578FE1%2E25E83D88E716D18F%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1544631713_e0a9cf3b64b146e76b8dd6a126a46c01

[3] Unsolved Problems in Blockchain Sharding, https://medium.com/nearprotocol/unsolved-problems-in-blockchain-sharding-2327d6517f43

[4] The Authoritative Guide to Blockchain Sharding, Part 1, https://medium.com/nearprotocol/the-authoritative-guide-to-blockchain-sharding-part-1-1b53ed31e060

[5] Blockchain Scalability: The Issues And Proposed Solutions, https://medium.com/@bitrewards/blockchain-scalability-the-issues-and-proposed-solutions-2ec2c7ac98f0

[6] Sidechains: Solving The Blockchain Scaling Problems, https://medium.com/coinmonks/sidechains-solving-the-blockchain-scaling-problem-b3847918b44

[7] Blockchain FAQ #3: What is Sharding in the Blockchain?
https://medium.com/edchain/what-is-sharding-in-blockchain-8afd9ed4cff0

[8] Solving Blockchain Scalability Problems With Layer 2 Solutions
https://hackernoon.com/blockchain-scalability-layer2-bitcoin-ethereum-bb34afd1f9d2

[9] SegWit, https://en.wikipedia.org/wiki/SegWit

[10] Bitcoin Scalability Problem, https://en.wikipedia.org/wiki/Bitcoin_scalability_problem