Scalable Sharded Blockchain Network Department of Computer Science — University of Wisconsin-Whitewater

Abdoul-Nourou Yigo,
Haki Dehari

April 2019

Abstract
In this research, we have explored the structure of a Blockchain network. Currently some research has been done in the Blockchain domain on how to improve the efficiency of the transmission in a Blockchain ecosystem. The effectiveness of transmission within a Blockchain network depends on the execution of the appropriate consensus to ensure the security and fairness of transactions. The most common consensus algorithms that have been used are Proof of Work (PoW) and Proof of Stake (PoS) for the early versions of the Blockchain system. These algorithms have been used in the first phase of Blockchain systems such as Bitcoin. The issue is that those consensus algorithms are inefficient because they require a significant amount of energy consumption to enable the validation of transactions in a complete Blockchain platform. Therefore, the amount of transactions that can be processed and validated is limited and expensive in terms of time complexity. As a result, scalable approaches need to be designed to sharpen the efficiency of the Blockchain system. The introduction of the sharding concept has given us the possibility to divide the network into small portions to enable the validation of transactions in parallel using the Byzantine-fault tolerant (BFT) consensus on a subset of nodes. Sharding the network can significantly improve the Communication Cost per Transaction ($CCPT$) because each transaction is validated in its specific shard using validator nodes with appropriate proof size. In our implementation, the transaction would follow the shortest path, traversing nodes with small proof sizes (pz). In our network structure, proof sizes are randomly assigned to the nodes to ensure unbiased properties within the network. Therefore, we have designed a Blockchain network structure to stimulate and evaluate our approach to perform a global optimization within a sharded Blockchain network.

1 Introduction

The revolution of Blockchain technology is undeniable because its potentialities to improve human interactions in different aspects could open a new era of digital communication. For instance, the way organizations handle business transactions and privacy could take another lift with the use of the Blockchain plaform. Therefore, making sure that this kind of system can effectively and efficiently process information could be crucial. As a result, the need of a Blockchain structure that could reduce the overhead of consensual agreement when it comes to the validation of transactions could have a profound impact on the efficiency of the Blockchain system.

Different Blockchain network structures have been proposed to try to improve this lack of efficiency and at the same time ensure that the Blockchain network's main properties such as security and decentralization are maintained accordingly [2]. This profound need of scalable schemes is due to the fact that the consensus algorithms created by Satoshi Nakamoto [1] in the *Bitcoin* ecoystem are inefficient and costly in terms of energy consumption. Researchers have tried to come up with different optimal schemes to enable the implementation of sharding. The concept of sharding is the ability to split the Blockchain network into small portions that would gather a subset of nodes. In [3], they have come out with a sharding technique called "RapidChain" that aims to achieve scalability considering the sharding of computation, communication and storage space using the appropriate consensus technique such as Practical Byzantine Fault Tolerance ($PBFT$). According To their principles, "RapidChain" could be the appropriate approach that could be taken to implement the sharding technique.

The manipulation of the sharding technique could give different results. Most of its implementations are able to ensure local optimas for the transmission of values from a sender node to a receiver node within a shard [5]. In [5], they have structured a sharding network concept that could validate transactions according to the supply of computing power. Assumably, this type of sharding approach can perform well with the presence of BFT adversaries within a permissionless Blockchain network. This solution could be practical with the Bitcoin ecosystem that suffers from a lack of scalability due to the fact that PoW and PoS require a tremendous amount of computing power.

The sharding approach can be implemented using different structural concepts. The main objective is the ability to optimize the throughput of the Blockchain network to increase the validation of a transaction by maintaining the fundamental properties of a Blockchain network. Since the structural approach of sharding is the creation of subsets of nodes in the Blockchain network, every shard needs to be bias resistant [6], to ensure fairness in the network when it comes to transaction validation using complex security concepts to guarantee uniqueness and immutability of the Blockchain system. With this architectural design, the OmniLedger [6] can achieve optimal performance within a shard, cross-shard and intra-shard for guaranteeing the atomicity of the transaction within a permission-less Blockchain network.

All of the above implementations of sharding we have described suffer from deterioration of the decentralization and the effectiveness of the fault tolerance [2]. Therefore, they have come out with another structural design of sharding called "spontaneous sharding" [2]. This principle is the following: Nodes in the network would only try to prove that their transaction is not double-spent making sure the transmission cost is minimized [2]. According to their theoretical results, they were able to achieve local optimization within a shard. They have proposed a theoretical scheme to perform a global optimization.

The inspiration for our research came from this concept of performing a global optimization in a sharded Blockchain network by minimizing the $CCPT$ in the shard. Since we are trying to translate theoretical results into practice, we have designed our Blockchain network with specified number of nodes n and edges M.

2 Sharding Concept

The sharding concept has some crucial challenges. It gives the ability to divide the network into different sectors that could be manageable in terms of transaction executions. However, some questions could arise:

1. How can security be maintained within a shard?
2. How can the fundamental property of a Blockchain, which is decentralization, be maintained within a collection of nodes?
3. How can fairness be maintained within the whole Blockchain after the network has been partitioned into small portions?
4. How can scalability and accuracy be assured during inter-shard communication?

These questions should be seriously taken into account due to the fact they are essential and fundamental to the core concept of Blockchain technology. For instance, the usability of Blockchain networks could produce some technical concerns if those questions are not taken into consideration to ensure the robustness of the Blockchain platform.

2.1 Sharding Security Maintainability

With a partitioned Blockchain network, the security and faithfulness of the network also dwindles along with the partition. Although there are many who argue that decentralization can still be maintained along with many of the other security properties of a Blockchain when Sharding, we must not ignore the implications of cutting down the "strength in numbers", so to speak. With each shard, there comes a calculated risk of developing a sub-network of corrupt nodes. This is known as a "Shard takeover". A Shard takeover results in the complete loss of all the data and transactions within the shard. This can be combated by applying the same technique used by the Ethereum Blockchain network. The Ethereum network randomly assigns nodes to shards to prevent the deliberate corruption of a shard by a corrupt group of nodes. This in turn, will incapacitate the ability of corrupt nodes to completely eliminate data from the network, thus conserving data integrity.

2.2 Maintainability of the Decentralization of a Sharded Blockchain Network

The main concern when applying sharding techniques to a Blockchain network is the maintainability of the decentralization of the Blockchain network. When validating a transaction, a traditional Blockchain network will try to achieve a majority consensus from all of the nodes to decide on whether a transaction is deemed valid or not, which is then added to the Blockchain if it is considered valid. Sharding introduces a fortunate and unfortunate scenario where the Blockchain has less of a majority to arrive at a validation consensus. There is much research being done in this area to improve the likelihood of a successful validation occurring within individual shards. Since decentralization is a core concept of the Blockchain network, one must be wary of implementing such techniques where decentralization is inherently lost. To combat this, researchers have begun investigating potential connections between different shards which is termed "Cross-shard Communication(CSC)". In a network model that contains CSC, a shard has connections to a set of other shards, in turn fortifying the decentralization concept by including other shards in a validation. The number of connections that a shard has to other shards is up to the creators of the model to decide. There are drawbacks to having too many connections as well as having too little connections, which means the decision to decide on the number of CSC a shard has must be made extremely carefully.

2.3 Considering the Challenges of Cross Shard Communication

Cross Shard Communication is already in the works for multiple Blockchain networks. The most noteworthy of these networks to implement CSC being the Ethereum Blockchain network. The implications that CSC has on a sharded Blockchain network cannot be ignored. Not only are the implications security related, they are also relative to the integrity of a whole network. By allowing other shards to play a role in the validation of a specific shard's transactions, the network model is therefore increasing the integrity value of itself, since integrity is one of the most important pillars of the Blockchain network. A specialized technique such as this one comes with its own set of challenges. One of these challenges being whether to make CSC connections randomized or should they be chosen by a system based on specific parameters (e.g. Connections based on proximity to a specific shard). Randomized CSC is not likely because it defeats the whole purpose of sharding. Instead, the Ethereum network decided to follow the $ReceiptParadigm$ for CSC. The $ReceiptParadigm$ for CSC is when each shard generates a receipt upon validation of a transaction which displays the state of the generating shard which can then be viewed by other shards which have a connection to the origin shard through CSC. The receipts are stored in a shared memory which cannot be modified and can only be viewed by the connected shards. This provides verification capabilities along with visibility on the connected shards through CSC.

2.4 Scalability and Accuracy Within a Shard

Sharding as a concept is actually much older than Blockchain technology and has been used by various systems from business database optimizations to Google's global Spanner database. While the Blockchain provides high level security through decentralization, it leads to issues of scalability that cannot be ignored. In a normal Blockchain, due to every node having to validate a transaction, the more nodes that get added to the network increases the number of validations required for a transaction by a linear factor. Due to the fact that every node must validate a transaction in a normal Blockchain, the network is only as fast as its slowest node. Combine this with the communication needed between nodes to reach a consensus, the network ends up facing an interesting scaling dilemma when confronted with a large number of nodes. Sharding helps alleviate these problems by providing an elegant solution to the scaling problem. Sharding, as discussed earlier, groups together a specific set of nodes and processes transaactions specific to that shard in parallel to other shards. This increases throughput tremendously, which increases even more with the existence of more shards. The question one may pose to this is: How is accuracy maintained within a shard and within the whole network due to the lesser number of validations required to validate a transaction within a shard? Let us discuss this concept with regards to the Ethereum network. The Ethereum network is in the works to implement a technique called "State Sharding". The current state of the Blockchain network is known as the "global state" and that is what everyone can see when they look at the Blockchain at a specific point in time. The tricky part of using this technique is that each shard must validate their respective transactions all while updating the global state of the network. When a specific transaction has been validated, the shard must update the global state of the network to inform the other shards and the network as a whole of the completed transaction. This helps retain a variant of a global consensus within the Blockchain network.

3 Network Model

For the implementation of the network structure, we have modelled a directed Blockchain network with n nodes and M edges to simulate the Communication Cost per Transaction ($CCPT$) within a shard. The shard would be a collection of specific nodes that are sectorially close to each other in the Blockchain network. The nodes in the networks would have a proof size pz. All the nodes in network are assumably honest because it is a permissioned Blockchain platform.

During the transaction from one node to another node, the nodes in the network would try to validate the transaction. The network would have the origin(supply node) and destination node (demand node) during the transaction. The other nodes in the network would play the role of transshipment nodes. The transshipment nodes that would be traversed by the transaction flow are supposed to have small proof sizes (pz). Also, the nodes in the network are supposed to be honest nodes. The edges (arcs) in the network would have the same communication cost. Since the objective is to minimize the $CCPT$ within the network, the transshipment nodes that would be traversed would be included along with their proof sizes on validating the transactions and would determine the trajectory of the transaction. The trajectory of the transaction would create the transaction path. We could try to determine the minimum cost to transact from one node to another node by applying some constraints that we would define accordingly.

We would assign random values as proof sizes to the transshipment nodes within the sharded network. These proof sizes could be considered as validation cost for each transshipment node. By randomly assigning the proof size to the nodes to evaluate how the transaction is transmitted across the network, we are conceptually evaluating how transactions are manipulated within Blockchain ecosystem. The random property would permit unbiased results that could ensure accurate transaction cost calculations within the Blockchain network. To accomplish this implementation, we needed to formulate a problem whose resolution could permit the satisfaction of these Blockchain network properties.

Let take tx_i as a transaction from a node to another node in the network and n_i a set of transshipment nodes. Let pz_i be proof sizes that would be randomly assigned to nodes within the network.

Theorem 1. *A transaction tx_i traverses a node n_i if and only if the node n_i has a small proof size pz_i.*

Proof. **We would perform a proof by contradiction.** Let us suppose that to minimize the transaction cost, the transaction flow could use any transshipment nodes with any proof sizes.

We assume that all of the nodes in our Blockchain network are honest without biased transactions. Let define n as a set of nodes in the network such that n_1, n_2, ..., n_i are in V (the set of vertices in the graph). Let randomly assign proof sizes pz_1, pz_2, ..., pz_i to the set of nodes in the network. Let us assume a transaction tx_i could be performed from n_1 to n_7 in a defined directed graph. Assumably the transaction from n_1 to n_7 can take the following path n_1, n_3, n_6, n_4, n_7 consider as $path_1$ with the respective proof size $pz_3 = 100$, $pz_6 = 60$, and $pz_4 = 20$ such that:

$$\sum_{i \in n}^{i} path_{1i} pz_i = 180$$

Therefore we would have a proof size of 180 that would reduce the transaction tx_i cost if $path_1$ taken. Let take another path n_1, n_2, n_4, n_5, n_7 that could be taken by tx_i from n_1 to n_7 consider as $path_2$ with the respective proof sizes $pz_2 = 30$, $pz_4 = 20$, $pz_5 = 25$ such that:

$$\sum_{i \in n}^{i} path_{2i} pz_i = 75$$

We could see that the transaction tx_i could not be minimized by taking $path_1$ because $\sum_{i \in n}^{i} path_{1i} pz_i > \sum_{i \in n}^{i} path_{2i} pz_i$. Therefore that is contradictory to our supposition that a tx_i cost could be minimized by using any transshipment nodes with any random proof sizes. As a result, the Communication Cost Per Transaction could be minimized if the transaction tx_i takes $path_2$ trajectory. We could say that the Communication Cost Per Transaction could be minimized if and only if the transaction path utilizes validation nodes with small proof sizes. $\square$

3.1 Network Flow Problem Formulation

To realize our implementation, we would define a directed graph that would represent our Blockchain network. Let V be a set of vertices and A be a set of arcs in the network. Let us define a directed graph $G = (V, A)$, a supply vector $s = (s_i)$ for $i \in V$ and fixed cost and the the proof size f_{ij} and pz_i for $(i, j) \in A$. Using these representations, we need the model to find a set of nodes with small proof size within the network that minimizes the communication cost per transaction $CCPT$ at the the same time ensuring the shortest path.

We have defined the transaction trajectory variable as a binary variable. The transshipment nodes that would be used for the transaction would be represented by 1 and the other nodes that do not participate in the transaction procedures would be represented by 0. Therefore, this model could be defined as mixed-integer program.

As a result, we could have this formulation define x_{ij} to be the flow on arc (i, j) and $y_{ij} = 1$ if the $arc(i, j)$ is in the transaction path. Otherwise $y_{ij} = 0$. Therefore, we could write the following constraint.

$$min \sum_{(i,j) \in A} f_{ij} y_{ij} pz_i \quad (1)$$

$$s.t.$$

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = s_i \quad (2)$$

$$x_{ij} \geq 0 \quad \forall \quad (i, j) \in A, \quad y_{ij} \in \{0, 1\} \quad \forall \quad (i, j) \in A \quad (3)$$

In our implementation, the model would try to minimize constraint (1) using the conservation constraint (2).
As additional notation, we use $V_s = \{i \in V : s_i > 0\}$ and $V_d = \{i \in V : s_i < 0\}$ as sets of demand and supply nodes respectively. The supply and demand nodes

would be randomly chosen to stimulate the transaction manipulations within the network.

3.2 Sharded Network Within the Network

Since the network would be sharded, the model needs to maintain the same properties within each shard to ensure the model effectiveness. Therefore, subsets of the network would be evaluated according to their structures. We have modelled our Blockchain network to make sure that all of the nodes within a shard are connected. Also, the shards in the network need to be connected because transactions could be done from shard to shard maintaining the network properties.

4 Implementation

We implemented Blockchain sharding to evaluate its performance on a simulated Blockchain network of 150 nodes. We simulated the Blockchain using a special type of optimization language and software called GAMS. Using GAMS, we used the Cplex solver to perform our optimization. GAMS is available at the following website and proper licensing is required to use the software: https://www.gams.com/.

4.1 Model Components

Using GAMS, we developed a complex, yet elegant optimization model to test the sharding concept on a simulated Blockchain network. The model consists of two different GAMS files. One of these files stores the large parameter values of the network. The values being the nodes and all of their respective connections to other nodes. The nodes in the network are not considered to be part of a complete graph data structure. Rather, they are part of an incomplete graph. The nodes and their respective connections have been randomly generated by a script written by us in the Python programming language. Implementing the model using a complete graph will be left to future research. The other GAMS file contains the implementation of the optimization of the simulated network. It contains the constraints revealed in the Network Model section of this paper. The file also contains the sharding portion of the model. We have used a randomized sharding approach using the GAMS Set operator and its respective powerful mechanisms used for producing dynamic sets. Using a randomized approach to sharding and to produce the network itself ensures that our calculations are not biased. Aside from the randomized connections and sharding, we have given each node a proof size that has also been randomly generated using the GAMS uniform operator. This proof size has been taken into account when traversing from one node to another node in the Blockchain network when performing a transaction. Along with the shortest path and lowest cost, we have implemented sharding to further limit the total distance travelled by a transaction within a specific shard. By doing this, the network will expend the lowest energy possible when traversing the nodes for a transaction, further strengthening the argument behind the sharding concept.

4.2 Constraints

As if with every model, we have set a specific set of rules for the simulated Blockchain network. For this specific model, we have only implemented two constraints required for a transaction to traverse the sharded blockchain to reach its destination node. The first constraint is the objective constraint and the cost constraint. This constraint calculates the total cost of traversing a sharded network from the source node to the destination node, all the while taking into account the proof size of each node traversed and the cost of the edge traversed, which are all the same in our model and was done this way for the sake of simplicity. This is also called the objective constraint due to the fact that this constraint is the constraint being minimized to find the optimal solution for each shard in the overall model. The second constraint is called the balance constraint. This constraint is what defines a completed transaction. It is implemented using the difference of sums of the constrained transaction path to find the truly minimized solution to the path taken by the model. This, combined with the objective or cost constraint, directs the optimization model to the correct path taken for the traversal of the transaction within a shard.

4.3 Sharding Implementation

The tricky part for the model was not figuring out the correct constraints, but figuring out how to split an already randomly generated Blockchain network into smaller parts or shards. We had to perform sharding all the while maintaining connectivity between the nodes in a respective shard. Using simple probability, we were able to deduce that a shard will most likely maintain connectivity if each node in the total network had at least seven connections to other nodes. When randomly generating our network, we made sure that the algorithm maintained at least seven different connections from each node to other nodes. By satisfying this rule, we were able to generate many different shards with different respective nodes within those shards and still maintain connectivity between the nodes. While performing randomized sharding, we created two different sets of nodes which housed the nodes that were used in previous shards and the nodes that were not used, respectively. This helped us ensure the uniqueness of a shard and that no shard shared a duplicate node between them. Once sharding was complete, we ran the model on each specific shard to obtain the desired results.

5 Evaluation

5.1 The Complete Network

During the original execution of the model, we performed the optimization on the complete set of nodes. This run was made to make sure the constraints put in place for the model performed in the desired way. Not only did the model have to find the shortest path, but it also had to take into account the randomly assigned proof size for each node. By doing this, the model had to find the shortest path that also had the least cost to get to the destination node. As discussed in the Network Model section of this paper, the cost was calculated by multiplying the proof size of each transition node by the cost of the edge or sub-path taken from one node to another node (Which we all set to 50 for simplicity. In future research, these can be easily changed to different values but for our model, it was not necessary). As mentioned previously, our optimization model had 150 nodes each numbered from $n_1 ... n_{150}$. We tested these nodes 15 different times on randomly assigned supply nodes and demand nodes. That is, for each run, we assigned one supply node and one destination node (where the transaction was originating from and where it was going to) and ran the model to see whether it took the shortest possible path given the cost constraint. Since the network was randomly generated, we had to verify by hand whether the path the transaction took was the shortest least cost path. Taking an example of one of the solutions during this test run, we had the following values:

Supply Node: n_{97}

Demand Node: n_{111}

Path Taken: $n_{97} \rightarrow n_{62} \rightarrow n_{83} \rightarrow n_{16} \rightarrow n_{111}$

Total Cost: 5750

Given this is a complete set of all the nodes, this ends up being the shortest path with the lowest cost to get from the supply node to the destination node.

5.2 The Sharded Network

After the original testing was done with the complete network, we moved on to running the optimization model on a sharded network. We manually and randomly(in GAMS) split the network into 3 shards or groups. This was done randomly to ensure that we did not intentionally choose nodes that had connectivity between them. Since a large scale blockchain will have thousands upon thousands of nodes, we had to make sure that when randomly sharding the network, to have some sort of path of connectivity between all of the nodes in the shard. As discussed in the Implementation section of this paper, we chose the number 7 for how many connections each individual node shall have to other nodes in the complete network. After performing the sharding operations, we were still able to maintain connectivity between the nodes in each individual shard. Since connectivity was maintained, the optimization model performed successfully when ran on each shard. Each shard was able to find a solution to minimize the cost by finding the shortest path and the smallest required proof sizes for the transition of the Blockchain transaction. In this run, we split the network into 3 shards. This is an arbitrary number and can be increased or decreased based on the connectivity on the nodes. After running the optimization model and minimizing the cost for each shard, the solutions were as follows:

Shard 1:

Supply Node: n_{95}

Demand Node: n_{91}

Path Taken: $n_{95} \rightarrow n_{109} \rightarrow n_{88} \rightarrow n_{91}$

Total Cost: 7000

Shard 2:

Supply Node: n_{123}

Demand Node: n_{94}

Path Taken: $n_{123} \rightarrow n_{8} \rightarrow n_{32} \rightarrow n_{110} \rightarrow n_{94}$

Total Cost: 5100

Shard 3:

Supply Node: n_{67}

Demand Node: n_{93}

Path Taken: $n_{67} \rightarrow n_{72} \rightarrow n_{62} \rightarrow n_{83} \rightarrow n_{28} \rightarrow n_{93}$

Total Cost: 3100

Given that all the nodes in each shard were unique, we were able to ensure that no nodes were finding a path that led to another shard. The individual networks were contained in this regard. In a real Blockchain network, we assume that all of these transactions occurred parallel to each other along with each of their validations. The results show that a randomized sharding technique can be used to break up the network into smaller regions to improve the throughput of the network drastically. The implications of this small experiment can be scaled to much larger networks if provided with the right parameters for connectivity on the nodes.

6 Conclusion

The concept of Blockchain sharding is still in the early stages of development in Blockchain networks such as the Ethereum network. Global validation schemas work for the time being, but there is no guarantee that they will suffice for when Blockchain networks get incredibly large, as is predicted to be the case in the future. There will need to be in place certain schemas that will be able to handle transactions and transaction validations to keep up with the growing use of Blockchain networks and cryptocurrency. The concept of sharding is a great step in the right direction for structuring the Blockchain network to increase throughput tremendously. Much work still needs to be done to improve the effectiveness and security of sharding for future research. This paper is only a start, and there will be much more research done on this topic in the future. Our optimization model shows that randomized sharding is a possibility to decrease the load on a complete Blockchain network and run the validations and transactions in parallel. This in turn, will lead to quicker validations and less energy wasted for all of the nodes in the network.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Z. Ren and Z. Erkin, "A scale-out blockchain for value transfer with sponteneous sharding," CoRR, vol. abs/1801.02531, 2018. [Online]. Available: http://arxiv.org/abs/1801.02531

[3] M. Zamani, M. Movahedi, and M. Raykova,"Rapidchain: Scaling blockchain via full sharding," IARC Cryptology ePrint Archive, Report 2018/460, 2018, https://eprint.iacr.org/2018/460

[4] L. Luu, V. Narayanan, C. Zhang, A secure sharding protocol for open blockchains. IN CSS, 2016.

[5] E. Kokoris Kogias, P. S. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. A. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in iEEE Symposium on Security and Privacy (SP), San Francisco, CA, May 2018, pp. 583-593.

[6] edChain, "Blockchain FAQ 3: What is Sharding in the Blockchain?", 2018. [Online]. Available: https://medium.com/edchain/what-is-sharding-in-blockchain-8afd9ed4cff0

[7] L. Mearian, "Sharding: What it is and why many blockchain protocols rely on it", 2019. [Online]. Available:

https://www.computerworld.com/article/3336187/sharding-what-it-is-and-why-so-many-blockchain-protocols-rely-on-it.html

[8] B. Curran, “What is Sharding? Guide to this Ethereum Scaling”, 2019. [Online]. Available: https://blockonomi.com/sharding/

A GAMS code

The GAMS code can be found at https://github.com/hakidehari/Blockchain-Optimization