2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) | 978-1-6654-2534-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ECAI52376.2021.9515090

PoRF: Proof-of-Reputation-based Consensus Scheme for Fair Transaction Ordering

Sidharth Shyamsukha
Dept. of Computer Science and Engg.
Nirma University
Ahmedabad, Gujarat, India
17bit109@nirmauni.ac.in

Pronaya Bhattacharya
Dept. of Computer Science and Engg.
Institute of Technology
Nirma University
Ahmedabad, Gujarat, India
pronoya.bhattacharya@nirmauni.ac.in

Farnazbanu Patel
Dept. of Computer Science and Engg.
Nirma University
Ahmedabad, Gujarat, India
19mcei04@nirmauni.ac.in

Sudeep Tanwar
Dept. of Computer Science and Engg.
Nirma University
Ahmedabad, Gujarat, India
sudeep.tanwar@nirmauni.ac.in

Rajesh Gupta
Dept. of Computer Science and Engg.
Nirma University
Ahmedabad, Gujarat, India
18ftvphde31@nirmauni.ac.in

Emil Pricop
Automatic Control, Comp. and Elect. Dept.
Petroleum-Gas University of Ploiesti
Romania
emil.pricop@upg-ploiesti.ro

***Abstract*—Blockchain (BC)-driven applications ensure trust and transparency among multiple stakeholders through different consensus mechanisms. In consensus mechanisms, a dishonest node (or application), might collude with peer nodes, or miners, to prioritize its transactions over other node transactions, thereby reducing its latency and improving its quality-of-service (QoS). To address the above limitation, this paper proposes a consensus scheme, *PoRF*, based on a reputation-based consensus scheme that allows fair and random transaction selection. The scheme proposes that a sharded BC is considered, managed through shard managers (SM), to address latency and bandwidth issues. A reputation coefficient score is generated, and a reward-penalty setup is considered for nodes based on honest, or dishonest transaction proposals. Based on the reward-penalty outcome, the reputation score for nodes is modified. The scheme is compared with fixed transaction ordering scheme, *Helix* scheme, and practical byzantine fault tolerance (PBFT) consensus for transaction fairness, transaction time, and epochs required for consensus formations. The scheme outperforms the considered schemes for the considered parameters. For example, for 50% dishonest nodes, an improvement of 70 % is recorded against a fixed transaction scheme, and at 4000 nodes, the proposed scheme takes 130 milliseconds (ms) less for the execution of transactions. For consensus formation, at 250 nodes, PBFT takes 167 epochs for consensus, compared to 24 epochs in the proposed scheme, which indicates the scheme's viability in real-world setups.**

***Index Terms*—Blockchain, Consensus, Fair Transactions, Reputation, Mining**

I. Introduction

In decentralized ecosystems, blockchain (BC) technology has led to disruption owing to its adoption to a wide range of applications, ranging from finance, healthcare, Internet-of-Things, and many more. In BC, peer entities add transactions, that are forwarded and added to blocks. Thus, each block contains a list of transactions and is added sequentially, based on linked hashes that connect the previous block hash to the current block. Miners perform transactions and block addition. They are provided an incentive for their mining activity [1]. In BC, all the recorded information is added through consensus, and thus any update in block structure invalidates the entire chain. However, the new block addition is selected from the pending transaction list, and thus, the transaction ordering sequence plays a critical role in determining fair quality-of-service (QoS) to every peer-to-peer (P2P) node.

The fair enforcement of transaction selection in BC might be diluted via dishonest entities (miners and P2P nodes). A P2P application might prioritize and favor its transactions to be added over other node transactions. This might require external compute and energy requirements, that are provided through external colluding parties. As more transactions are recorded for dishonest nodes, the transaction selections increase, thereby improving the block mining rate, and shared node bandwidth [2]. Moreover, some networks allow joint participation of miner and P2P node for proposal of block selection, and thus miner intends to add nodes from where the profits and incentives might be maximized [3]. Thus, there is a requirement of stringent fair enforcement of node participation, to make fair transaction selection.

To enforce fair transaction ordering (FTO) and dependence, researchers globally have proposed schemes to enforce fairness in transaction ordering. One possible approach is to design a consensus protocol where nodes mutually agree on the validity of ledger data and verify that false appends are not considered. A double cross-verification is required before approval of the new block addition. Earlier, a proposed protocol, Asayag *et al.* [4] proposed a scheme named *Helix*, that strengthened the FTO through a round-based transaction addition scheme. In *Helix*, each round consists of a primary node selector, which gets elected through majority vote participation in the BC network. The selector elects the new block addition, and the election committee validates the newly added block ledger entries. Once the block addition is deemed fair, then the block is added to the main chain structure. To avoid collusions in the selection committee process, nodes are added randomly, and thus next node selection is equally likely for all nodes.

However, *Helix* scheme suffered an inherent limitation. The scheme did not utilize the entire information of committee structure and focus more on independent approval counts for a majority decision. To address the limitation, a joint decision protocol is proposed, that enhanced the effectiveness of the voting committee. The scheme proposed incentives to nodes on basis of a reputation score, that encourages nodes to act fairly for primary selector [5]. Another study by Orda *et al.* [6] suggested using bloom filters or Merkle tree as a primary data structure to eliminate dishonest nodes that prioritize their transaction selection, that jeopardize the fairness of BC. However, the proposed schemes suffered from inherent limitations of low transaction rate, and thus mining scalability was a major concern.

To address the scalability issues, sharding was proposed in BC ledgers. The shards are constructed based on node distance, energy consideration, and minimizing latency in oral propagation updates [7]. The sharding segments follow the practical byzantine fault tolerance (PBFT) consensus, and thus dual benefits of transaction selection fairness and scalability are addressed. To address the issue of transaction age, an age-aware protocol is proposed by Sokolik *et al.* [8], that prioritizes transactions based on longest waiting time, regardless of the identity of the application node. This ensures fairness for old transactions and ensures uniformity in transaction ordering. However, dishonest nodes can modify timestamps to forge the waiting time, that increases the transaction age, and subsequently, scheduling priority of dishonest transaction increases.

A. Discussion on existing schemes

In BC network, for FTO, two schemes *Fairledger* [9] and *Optimistic fair exchange* [10] are proposed. The schemes assume that for transaction addition, participating nodes are honest, and do not perform any collusion activity. The inherent assumption implies that meta-information of the transactions are not altered, and only those nodes that are caught violating the above principles are punished. As the baseline assumption is all nodes are honest and fair, any treacherous act is taken into strict action and all nodes are punished in terms of transaction bandwidth. Due to slow executions, there are frequent transaction timeouts and rollbacks in the network, which reduces the overall throughput speed of the network and increases mining latency.

Next, we present the fixed transaction [11], and the age-aware [8] approaches. In the fixed transaction approach, all nodes are proposed equal bandwidth and resources to add transactions. In the age-aware approach, a timer is attached with the transactions that are not currently added as part of the chain structure for a particular node. Once the transactions become part of the proposed block structure, the timer is pruned. However, the scheme does not provide any mechanism for timestamp modifications, that increases the transaction wait time, that increases the node priority to add the next transaction. A modified approach is proposed by authors in [12], and a consensus-based decision in [6], that addresses the limitations via verification of the node timestamp, through a global notion of timer. However, the approach adds extra overhead on the network to monitor the transaction age for all nodes, which significantly reduces the scalability of the chain. To address this, smaller chains, or shards, are proposed, that improves the mining latency and scalability of the overall chain. Thus, the proposed scheme, *PoRF*, exploits the benefits of both age-aware consensus, and sharding, which reduces the overall latency, redundancy, and improves the block processing time.

B. Novelty

The proposed work addresses the research gaps in earlier approaches through a reputation score, and reward-penalty setup for nodes in the BC network. There is a linear increase in reputation score for honest transaction additions, and in the case of dishonest additions, the reputation score is halved. This ensures a fair selection of nodes for different requirements. We have considered a sharded BC, and nodes are continuously monitored inside the shard, in the presence of SM. In case SM oversights, or collude with any node, the entire shard is penalized, due to incorrect transaction updates caught by miners. In case of a penalty, the reputation score of SM is reduced, and if the score falls below a threshold limit, the SM is dissolved, and a new SM is appointed through a fair election. Through the reward-penalty scheme, and reputation score of SM, we ensure a dual-monitoring of activities inside a shard.

C. Research Contributions

Following are the major contributions of the scheme.

- An FTO scheme is proposed which uses timestamps and reputation coefficient to ensure honest transaction addition inside the shard boundary structure.
- A mutual consensus between inter-shard managers is proposed which selects the miners and new shard managers based on their reputation.
- Based on mutual consensus, dishonest and colluding nodes are identified in the scheme, and subsequently, reward-penalty is applied to them. For the score, a mutual consensus and validations of previous transactions are committed.

D. Layout

The paper is organized into five sections. Section II presents the proposed scheme *PoRF* consensus for fair transaction ordering. Section III presents the performance evaluation of the proposed scheme, and finally section IV concludes the paper.

II. *PoRF*: The Proposed framework

The section presents the discussion of the proposed scheme, *PoRF* for FTO of nodes in BC network. We first present the system model, and then present the proposed consensus scheme. The details are presented as follows.

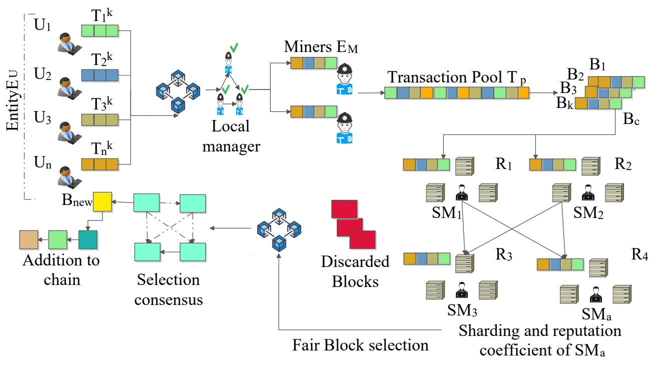


Fig. 1: *PoRF*: System model

A. System Model

The section presents the proposed consensus scheme, *PoRF*, that presents a reputation-based scheme for addition of fair transactions. Fig. 1 presents the schematics of the model. The scheme consists of entities $E = \{E_U, E_M, E_{SM}\}$, where E_U denotes the users, E_M are the miner nodes and E_{SM} denotes the shard manager (SM). We consider that n users are connected in the network, represented as $\{U_1, U_2, \ldots, U_n\}$. Any i^{th} user U_i adds k transactions, denoted as T_k^i, based on mapping $M_i : E_{U_i} \rightarrow T_k$. The collected data from M_i is presented to the transaction pool T_p, aggregated $\forall n$ users. with the usual constraint of $T_p > n$. The transactions are added by E_M to T_p, once the E_{SM} agree unanimously on the same. The scheme assumes $q \in E_M$ miners, and $b \in E_{SM}$ the shard managers as validators. Out of q miners, t miners are malicious, with constraint $\{t \in E_M, t < q\}$. These t miners might collude with l malicious validators, with $\{l \in E_V, l < b\}$, or E_U, to add unfair transactions in T_p.

Based on added transactions to T_p, and fair ordering scheme, m transactions are added to a block B, and overall z blocks $B_c = \{B_1, B_2, \ldots, B_z\}$ are added to . To allow FTO sequence in block addition, the scheme assumes B_c is subdivided into shards S_{B_c}. A total of a shards are present, denoted as $\{S_1, S_2, \ldots, S_a\}$, where $a < z$. Each shard contains a sub-sequence $\{B_x, B_{x+1}, \ldots, B_y\}$ of the original B_c, with the usual constraints $1 < x < z, 1 < y < z\}$. Every S_{B_c} contains E_{SM}, as head node that supervises the fair transaction addition in its own shard. With the help of sharding, the maximum transactions per second (TPS), which is depicted and obtained as in [13],

$$TPS_{max} = k\lfloor B_{size} - H_{size}/Tx_{avg}\rfloor/T_{interval} \tag{1}$$

Here, B_{size} is the block size (bytes), H_{size} is the block header size, Tx_{avg} is the average transaction size and $T_{interval}$ is the block interval period. So, by gradual increase of block size/No. of shards, TPS can be increased.

The selection of E_{SM} in any S_a is done by the network through bloom filters. We consider the membership $M(E_{SM})$ of any SM. A bit vector B_V is considered, with k cells. The index of any k^{th} cell is represented by $I(k)$. In case E_{SM} is present in S_a, $M(E_{SM})$ is ***TRUE***, and value at $I(k)$ is presented as follows.

$$V(I(k)) = SHA - 256(ID(E_{SM}) \tag{2}$$

where $V(I(k))$ is the value at $I(k)$, and $SHA - 256(ID(E_{SM})$ presents the hashed reference of ID of E_{SM}.The reason of usage of bloom filters is the membership independence and uniformity. The false-positive rates are significantly reduced as follows.

$$FP = (1 - e^{-kn/m})^k \tag{3}$$

where k represents the number of hash functions required for membership, n denotes the range of values, and m denotes the number of bits in $ID(E_{SM})$. Given m and n values, the optimization problem is to choose a suitable k [4]. Based on this, every shard node consists of E_M, and E_V, which are responsible for mining and validation of added transactions.

Whenever an E_U completes a transaction T_p, it updates the information based on the order of metadata stored in the main node to the ledger. That information is verified by the E_{SM} based on the calculation done by the node, also it checks the timestamp of the transaction and other details by cross verifying the details with all the E_{SM}'s of the shards $\{S_1, S_2, \ldots, S_a\}$. if any discrepancy is found then the reputation Coefficient R_k, where $1 <= k <= n$ of the E_U value is reduced and it is penalized with timeout or fines.

As depicted in Fig. 2, a node is elected as E_{SM} by other nodes in every shard. Post election of E_{SM}, whenever a new node seeks to join through the shard, it needs the approval of the E_{SM} to join the respective shard. For the same, E_{SM} verifies the hash of the data of the new node to be added, and R of the node, and then stores the data in its ledger. The data is stored based on the timestamp of the transaction, as suggested in Silva *et al* [11], post verification by E_{SM}. The transactions are divided into two parts, first, the older-transaction part P_{ot}, which includes the senior transactions (ST), and the next is randomly selected transactions, denoted by P_{rand}. The details are presented as follows.

$$\begin{aligned} P_{ot} &= ST_{max} \\ ST_{max} &< B_{size} \end{aligned} \tag{4}$$

The random selected transaction addition is presented as follows.

$$P_{rand} = B_{size} - ST_{max} \tag{5}$$

where, ST_{max} is maximum ST included in the shard. Older transactions are given higher priority so that they get updated the E_{SM}'s ledger first. This ensures fairness in sequencing transactions in the block.The proposed block passes the older transactions part validation as follows,

$$\prod_{Senior\ Transaction\ \alpha} Pr(G_\alpha^{Eu} \geq g_\alpha^{Eu} | Q_{tlearn}^{Esm,\alpha}) > v \tag{6}$$

Here α denotes any ST observed by E_{SM}. Probability of the ST selected should be greater than some threshold value v to justify fairness.G denotes a random variable representing the

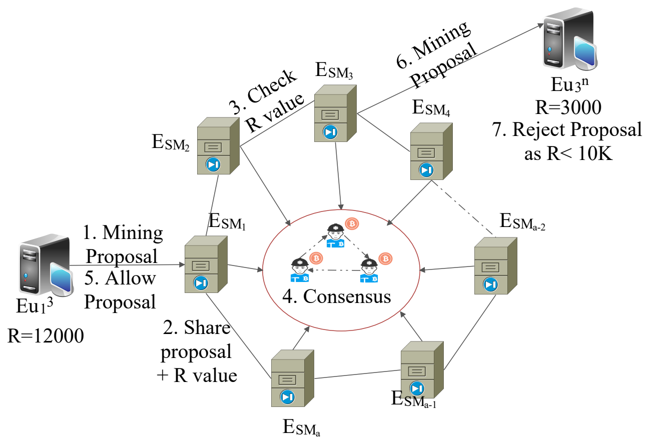


Fig. 2: *PoRF*: Transaction Validity through reputation values

age of the transaction added by the E_u, while g is the actual age. $tlearn$ is the time at which the E_{SM} learned about the transaction [8].

The ST satisifies the following constraints

$$C1 : Tx\ local\ age > T_{threshold} + T_{propagation} \quad (7)$$

where, $T_{threshold}$ is threshold Tx age, and $T_{propagation}$ is

Algorithm 1 Checking Transaction Validity

Input: Transaction Details,R value ,node
Output: Condition for adding transaction$TrueFalse$

```
1: procedure VERIFICATION( )
2:     $a \leftarrow$ Total Shards
3:     for $j = 1$ to $j = a$ do
4:         $x \leftarrow$ Nodes in  a Shard
5:         Algorithm 2.Initialization() called
6:         for  $l = 1$ to $l = x$ do
7:             $k \leftarrow$ Number of transactions added by a node
8:             for $m = 1$ to $m = k$ do
9:                 $t \leftarrow$ time
10:                $amount \leftarrow$ Transaction amount
11:                $s \leftarrow$ Sender
12:                $r \leftarrow$ Receiver
13:                if ($s.R < 10000$ or $r.R < 10000$) then
14:                    Return False
15:                end if
16:                if ($k_s.hash() == k_r.hash()$) then
17:                    if  ($s.time()$  ==  $r.time()$  and  $s.amount()$  == $r.amount()$) then
18:                        Algorithm 2.Calculation() called
19:                        Return True
20:                    else
21:                        Return False
22:                    end if
23:                else
24:                    Return False
25:                end if
26:            end for
27:        end for
28:    end for
29: end procedure
```

the time a Tx is proposed by a user until it is communicated to the other user. After completing a block of data the E_{SM} hashes the block, and then passes it on to all the other main nodes $\{S_{SM1}, S_{SM2}, \ldots, S_{SMa}\}$, which is total of a shards. All E_{SM} verifies the hashes with the old hashes of

Algorithm 2 Calculating Reputation Coefficient

Input: Transaction Details,R value,node
Output: Condition for adding transaction$TrueFalse$

```
1: procedure INITIALIZATION( )
2:     $E_u^i \leftarrow$ New nodes added to
3:     for Every new node do
4:         assign $E_u^i.R = 10000$
5:     end for
6: end procedure
7: procedure CALCULATION( )
8:     Poll nodes for transactions
9:     $k \leftarrow$ Number of transactions added by a node
10:    for $i = 1$ to $i = k$ do
11:        if $T^[i]_k$.validity==$True$ then
12:            $R = R + 1$
13:        else
14:            $R = R2$
15:        end if
16:    end for
17: end procedure
```

the block stored within their own ledger and then add the new block in their respective ledgers. Algorithm 1 presents the schematics of transaction validity. Initially, a verification process is called on a shards, and transactions are added. Next, based on computed value of R, the transactions are added. Algorithm 1 has a total of a shards, and x nodes in a given shard, that adds k transactions in total. Hence, the time complexity is $O(k.x.a)$. $\forall$ a shards, k transactions are stored, and hence the space complexity is $O(ka)$.

B. *PoRF: A novel reputation-based consensus protocol*

In this subsection, we present the usage of reputation coefficient R that helps in distinguishing dishonest nodes from honest nodes. Apart from normal nodes, every E_{SM} are assigned R values. All new nodes start with $R = 10000$, which is incremented for every correct transaction and halved for every fraudulent or incorrect transaction. This value of $R = 10000$ is assumed to facilitate the halving punishment for the dishonest nodes since a considerable amount of transactions takes place in a BC so the R value will be maintained. Only entities with $R > 10000$, set as a threshold, can participate as E_M or E_{SM}, rest nodes are eligible to take part in regular transactions. This ensures fairness in selection among the entities.

$$\begin{aligned} Correct\ T^i\ added &\rightarrow R = R + 1 \\ Incorrect\ T^i\ added &\rightarrow R = \frac{R}{2} \end{aligned} \quad (8)$$

Any user E_U that wishes to change its state to E_M, notifies this proposal to the E_{SM}. E_{SM} frequently polls all the nodes under it and collects all these proposals. E_{SM} then broadcasts these proposals to all the other shard managers $\{S_{SM1}, S_{SM2}, \ldots, S_{SMa}\}$. Then all the E_{SM} together gossips among themselves the selection ratio x i.e. the percentage of proposals that would be allowed based on the mining requirements of the system. All these E_{SM}'s compare the R values of all the candidate nodes and priority is given to the nodes having the highest R values, or every node above the threshold value of 10000 are allowed to be a E_M. For example, at the start, the R value of E_M's is below the threshold. All E_{SM} looks at the previous transactions of the nodes from their ledgers, to find any set of discrepancies. The final list of new E_M's is released when consensus is reached $\forall\ E_{SM}$. The respective E_{SM} notify the respective candidate entity of their status of whether they are allowed to be a E_M or not.

In case some E_{SM}'s from a different shard tries to collude with dishonest E_M's to add some fake transactions T_i or change the value of R of some $E_U$nodes, they are not successful in the scheme. Alteration in R value by any E_U is done so that the node can E_M, and gain access to add transactions to BC. Such collusions are not possible as R values and transaction timestamps won't match in other E_{SM}'s ledger, and thus consensus from other E_{SM} would fail. Other E_{SM}'s would be able to notice these discrepancies, and this would result in penalizing the concerned E_{SM} R value to half, and subsequently, the dishonest E_{SM}'s would lose its role as SM. In such cases, for the concerned shard, an election process would be re-initiated and a new E_{SM} would be elected from the remaining nodes, either in the entire network or from E_M that has the maximum R value in the shard. Algorithm 2 presents the computation of R based on the reward-penalty scheme. For every honest addition, the R value increases linearly and is halved for an invalid transaction. The R value is considered for every node polled for transactions in a shards. Thus, the time complexity of algorithm 2 is $O(a.x)$. For all x nodes, storage of R value takes a constant space, thus the space complexity is $O(x)$.

III. Performance evaluation of *PoRF*

The section presents the performance evaluation of the *PoRF* scheme. For simulation purposes, the instances are run on each GPU independently. The ledger stored in the memory of the GPU is used to refer to the previous transactions. We initiated a local sharded BC with 10000 nodes, these nodes were divided equally into 25 shards. In each shard, we have considered 10 nodes, and 1 E_{SM}. Thus, for the entire BC network, a total of 25 E_{SM} are present in the entire network. We have considered a reduced value of R for better visibility of simulation results and have fixed the value at $R = 1000$. We have compared our proposed scheme against fixed transaction ordering, as proposed in Silva *et al.* [11] for transaction fairness, and Asayag *et al.* [4] for transaction time. Next, we compared our scheme against the standard PBFT scheme for a number of epochs that required consensus formation with increasing users. The experimental setup and discussion of the results are now presented in subsequent subsections.

A. Experimental Setup

For simulation purposes, we have considered the Intel Core *i7* processor series that runs the sharding instances of the ethereum mining server. For storage requirements, 16 gigabytes RAM is present, and for mining requirements, we have used *Nvidia RTX 1650* graphics processor units. The ethereum setup is connected with 1 *GbE* ethernet switch, and experiments are performed on Linux operating system.

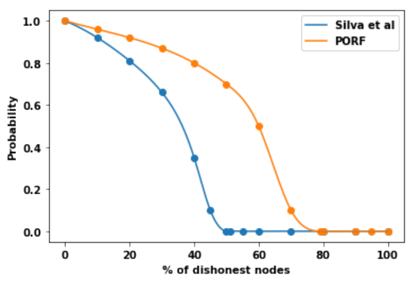
B. Simulation Results

As shown in Fig.3 (a), *PoRF* is compared with fixed transaction scheme proposed by Silva *et al.* [11] based on the probability of transactions that would be added in proportion to the percentage of dishonest nodes. Silva *et al* [11] proposed an epidemic based transaction ordering protocol *EpTO*, that can be manipulated by the dishonest nodes when they reach a majority in the consensus. However, in the proposed scheme, the limitation is addressed, as shard managers are responsible for block communication, and unless all E_{SM} become dishonest, there is slight probability of dishonest additions. The same is stated due to the fact that each E_{SM} has to maintain a consistent reputation value R, otherwise it would not be allowed to add transactions in future. As indicated in Fig. 3 (a), at 50% dishonest nodes, the probability of adding the transaction in Silva *et al.* [11] becomes negligible around 0.0024, while in *PoRF*, the probability value is 0.7011 which indicates a 70% improvement

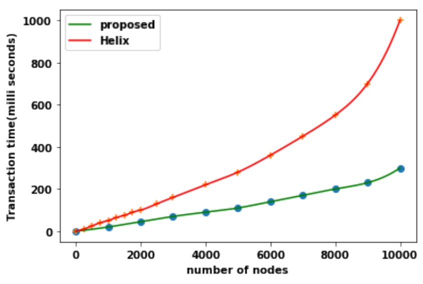
In Fig.3 (b), we compare transaction time against number of nodes. At 4000 nodes Asayag *et al.* [4], *Helix* scheme took approximately 220 milliseconds (ms), while the proposed scheme takes ≈ 90 ms to complete and validate a transaction. As more nodes are added, the improvement is further intensified. As transactions are validated by E_{SM}, less time is required for validation purposes.

Next, we present the impact of reputation coefficient against collusion attacks. Fig.4 (a) shows the impact of R on system. For demonstration purposes, we have considered the value of R as 1000 as threshold. Here, 4 different user type $U = \{U_1, U_2, U_3, U_4\}$ are considered, where they propose 100%, 75%, 50% correct Tx, and 100% of the fraudulent Tx respectively. So, as per the properties of R, U_1 always proposes a fair transaction, hence is rewarded, and the value of R increases linearly. If we consider 14 transactions, the final value would be 1014. U_1 proposes $\frac{3}{4}$ transactions as correct, and hence at 3^{rd} transaction, a penalty of $R/2$ is applied. Similarly, U_3 proposes $\frac{1}{2}$ transactions correctly, and hence there is a drop of $R/2$ after each interleaved transaction. U_4 proposes all transactions as incorrect, and thus each successive transaction, value of R is halved. Hence, increased penalties for E_U results in revocation of rights of E_{SM}, or E_M in the BC network. This reward-penalty step motivates the user to propose correct Tx, eliminating chances of possible collusions between E_{SM} and E_M, ensuring fair selections.

At last, Fig.4 (b) shows the required number of epochs required for consensus formation in the BC network. In consensus formation, we consider all nodes in BC converge to the same and common state of the ledger. The formation is based on the propagation of oral updates in the network. The scheme is compared against PBFT consensus, which requires $2f + 1$ validations, where f is the number of faulty nodes. In the proposed scheme, the block validation is done by E_{SM} only. For example, for 250 U, PBFT requires 167 epochs for convergence, compared to 24 epochs in the proposed scheme. The reduction in the number of epochs is due to fewer oral

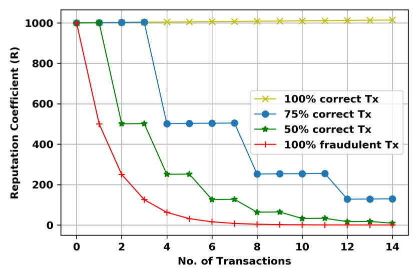


(a) Analysis of transaction fairness

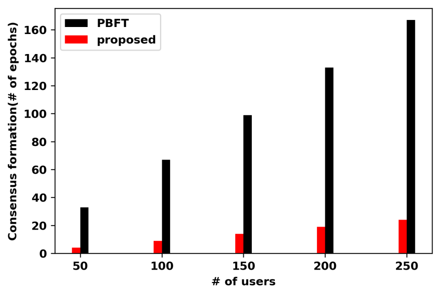


(b) Analysis of transaction time

Fig. 3: Comparative analysis of *PoRF* scheme against existing schemes



(a) Impact of reputation coefficient against collusion attacks



(b) Impact of sharding on validations

Fig. 4: Analysis of impact of reputation coefficient and required validations against PBFT

propagation messages among E_{SM} only, and not all nodes. This also improves the communication latency in the network.

IV. Conclusion

The paper presents a reputation-based consensus scheme, *PoRF*, that addresses the issue of FTO in BC. The scheme addresses the skewness of dishonest nodes, adds more transactions, and fairly improves the QoS for all nodes in BC. To address BC scalability, and mining latency, we divide the entire global chain into shards, each managed through SM. Each SM is assigned a reputation score of R that indicates the honesty of oral communication updates in BC ledgers. Each node is assigned an initial value of R, and a reward-penalty scheme is formulated. Nodes that propose incorrect transactions are penalized and are revoked from proposing further transactions, and honest nodes are rewarded with a linear increase in the R score. Nodes with the highest R value are elected for shard managers. This motivates nodes to participate honestly and increases fairness in the ecosystem.

As part of the future work, the authors would like to investigate further on optimal trade-offs of fairness vs scalability through reduced node participation in shards. We would also like to investigate the effect of Poisson arrival transaction distributions to local shards and measure the effectiveness of fair transactions, to maintain a consistent BC throughput.

References

[1] S. Ferretti and G. D'Angelo, "On the ethereum blockchain structure: A complex networks theory perspective," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 12, p. e5493, 2020. e5493 cpe.5493.

[2] H. Shi, S. Wang, and Y. Xiao, "Queuing without patience: A novel transaction selection mechanism in blockchain for iot enhancement," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7941–7948, 2020.

[3] Y. Liu, Z. Fang, M. H. Cheung, W. Cai, and J. Huang, "A social welfare maximization mechanism for blockchain storage," 2021.

[4] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira, "A fair consensus protocol for transaction ordering," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pp. 55–65, IEEE, 2018.

[5] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "Porx: A reputation incentive scheme for blockchain consensus of iiot," *Future Generation Computer Systems*, vol. 102, pp. 140–151, 2020.

[6] A. Orda and O. Rottenstreich, "Enforcing fairness in blockchain transaction ordering," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 368–375, IEEE, 2019.

[7] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, (New York, NY, USA), p. 41–61, Association for Computing Machinery, 2019.

[8] Y. Sokolik and O. Rottenstreich, "Age-aware fairness in blockchain transaction ordering," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pp. 1–9, IEEE, 2020.

[9] K. Lev-Ari, A. Spiegelman, I. Keidar, and D. Malkhi, "Fairledger: A fair blockchain protocol for financial institutions," *arXiv preprint arXiv:1906.03819*, 2019.

[10] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Toward fairness of cryptocurrency payments," *IEEE Security & Privacy*, vol. 16, no. 3, pp. 81–89, 2018.

[11] P. M. da Silva, M. Matos, and J. Barreto, "Fixed transaction ordering and admission in blockchains," 2019.

[12] L. Zhang, H. Zhang, J. Yu, and H. Xian, "Blockchain-based two-party fair contract signing scheme," *Information Sciences*, vol. 535, pp. 142–155, 2020.

[13] J. Yun, Y. Goh, and J.-M. Chung, "Dqn-based optimization framework for secure sharded blockchain systems," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 708–722, 2021.