P-sharding: Streamline Emergency Medical Transactions via Priority Sharding

Akanksha Saini[1(✉)], Navneesh Kaur[2], Navneet Singh[2], and Dimaz Wijaya[3]

[1] Deakin Blockchain Innovation Lab, School of Information Technology, Geelong, Australia
sainiakan@deakin.edu.au

[2] Department of Information, Communication and Electronics Engineering, Catholic University of Korea, Seoul, South Korea
navneetsingh@catholic.ac.kr

[3] School of Information Technology, Deakin University, Geelong, Australia
dimaz.wijaya@deakin.edu.au

Abstract. In recent years, several blockchain-based access control models have been emerged to give individuals control over their sensitive Electronic Medical Records (EMRs) in the healthcare sector. From our extensive literature review, we observe that currently, these models have no provision of prioritising the emergency transactions. This critically affects the quick and streamline sharing of EMRs in a multi-domain network environment. Further, it restricts the optimal usage of blockchain network affecting scalability. Sharding has arisen as a viable option for addressing the issue of blockchain's scalability and performance. Motivated from this, in this paper, we first propose prioritised sharding (P-sharding), a novel mechanism to streamline the processing of priority or emergency transactions in blockchain-based access control models by improving the throughput of each prioritised shard, in the context of multi-domain healthcare networks. Finally, the performance of the model is verified, validated, and also compared with the existing sharding mechanism. The obtained results are promising and encourage to further sparkle this direction.

Keywords: Blockchain-based access model · Electronic medical records · Healthcare · Priority transactions · Scalability · Sharding

1 Introduction

Blockchain technology is defined as a peer-to-peer decentralised network having distributed ledger of transactions. The network of nodes validates the transactions using cryptography means solving a mathematical puzzle via mining. With its secure features, blockchain has found many applications in various sectors ranging from financial services, supply chain, insurance to healthcare. Our research primarily focuses on the healthcare domain. There exists highly sensitive Electronic Medical Records (EMRs) need to securely access and streamline

W. Bao et al. (Eds.): ADHOCNETS 2021/TridentCom 2021, LNICST 428, pp. 242–259, 2022.
https://doi.org/10.1007/978-3-030-98005-4_18

sharing among healthcare professionals working in multi-domain healthcare facilities. EMRs are conventionally recorded in cloud-based health repositories, with the strategic initiatives of data sharing across different registries. However, it constitutes a very high privacy risk of a security breach to occur posing a major challenge for the digital trust in e-Health where storing, accessing, and exchanging sensitive patient-related data must comply with several regulations, while remaining accessible to authorized health practitioners [1]. Cloud-based access control models [2,3] validate the access right through a centralized entity suffer a single point of failure.

Healthcare is one of the domains with the biggest investment in blockchain technology due to its secure transaction processing to transfer sensitive EMR. Decentralised blockchain technology provides a solution to make the medical data secure, achieves patient-eccentricity, and makes it accessible across the health departments. In recent years, many blockchain-based access control mechanisms have been proposed for EMRs. All these models rule out the validation of access rights by a centralised server. An architecture for scalable access management has been proposed in Internet of Things (IoT) context [4]. Some other methodologies for managing medical records have been proposed in [5] and [6] using smart contracts [7] considering the issue of interoperability and making their system more compatible. In [8], the authors have developed an access control framework based on smart contract, which is built on the top of distributed ledger, to secure the sharing of EMRs among different entities involved in the smart healthcare system.

However, during our study, we identified that among healthcare data, not all are of equal importance, they have different service requirements on the blockchain-based access model. Emergency EMRs need to get processed faster as per their priority in the access control mechanism which the current blockchain is not fully capable to do unless for high transaction fees. In existing blockchain systems, all the transactions are considered evenly and processed as First-In-First-Out (FIFO) invariant of the type of consensus used. We believe that a) it does not only restricts the optimal usage of blockchain's capacity but also selfish (malicious) validators can flood the network with less important transactions preventing emergency transactions from being processed in a timely manner, b) without prioritising the emergency transactions, fatal loss incurs to the patients and hospitals as it hinders the real-time access of patient's EMR in Emergency Medical Services (EMS), c) finally, to optimize OPEX and CAPEX, EMS have to consider the prioritised transactions prior than the regular ones. Some early attempts have been made by [9,10] to analyse the performance of blockchains in the context of scalability without the provision of transaction prioritisation.

To the best of our knowledge, none of the existing blockchain-based access models have the provision of prioritising the emergency transactions which eventually restrict them to get scaled [11,12]. Since existing blockchain platforms like Bitcoin [13], and Ethereum [14] only process limited rate of transactions per second (tps) in FIFO manner leading to the increased overall latency. It takes longer time for emergency transactions to get fetched from the memory pool because there is no provision for prioritized scheduling. Hence it cannot be immediately

applied to the healthcare system in its present state. It is of utmost importance to tackle these challenges before commercially integrating blockchain-based access control into the healthcare system. A blockchain-based healthcare system can be optimized in terms of scalability in the following ways to handle a growing number of EMR transactions, a) reducing the communication and computation overhead; b) adding resources to a single node, i.e., vertical scaling; and c) adding more nodes to the blockchain, i.e., horizontal scaling, which include the concept of sharding.

In order to add this provision, we are among the early ones to propose a novel mechanism named priority sharding (P-sharding), fundamentally based on the sharding principle, to prioritise the processing of emergency transactions by applying tags to EMR transactions in blockchain-based access control models. The key idea behind P-sharding is to automatically divide the available computational resources into smaller groups or committees, each processing a prioritised sharded block containing a set of emergency EMRs. The major contribution of P-sharding is to process the transactions faster and alleviate the scalability issues. In sharding [15], data is broken into different shards and instead of all nodes verifying the entire data individually, they verify one shard each side-by-side. The amount of time is saved exponentially through sharding [16]. In this paper, we have applied prioritised sharding on Ethereum-based permissioned blockchain[1]. We take advantage of sharding concept and with our analysis, we observe that the prioritisation of transactions directly impacts the scalability of the system. Hence, our model of P-sharding prioritises the emergency transactions and processes them faster contributing to the development of a streamlined and scalable system. The main contributions of our paper are highlighted as follows.

1. We propose the first-ever novel approach of prioritisation in sharded blockchain to process emergency transactions faster as per their priority. This improves the processing rate of prioritised emergency transactions drastically.
2. We simulate our proposed approach through permissioned blockchain in a controlled environment. Evaluation results signify that our model is scalable and efficient to achieve priority-based weighted fair queuing using a probabilistic approach.
3. We identify a significant use-case of the above contribution in the healthcare sector to have scalable and efficient blockchain-based access control.

The remaining of this paper is organized as follows. Section 2 gives an overview of the related works. In Sect. 3, we introduce our proposed P-sharding model and provide the details. In Sect. 4, the implementation and performance analysis are given. Finally, this paper is concluded in Sect. 5 and provides further discussion.

[1] https://github.com/ethereum/sharding.

2 Preliminaries

Before we provide in depth details of P-sharding, in this section we show a typical behavior of blockchain-based access control models. We observe that workflow of the majority of these models [4–8] is relatively the same. Based on this observation, now we present a high-level generic blockchain-based access control framework, as shown in Fig. 1.

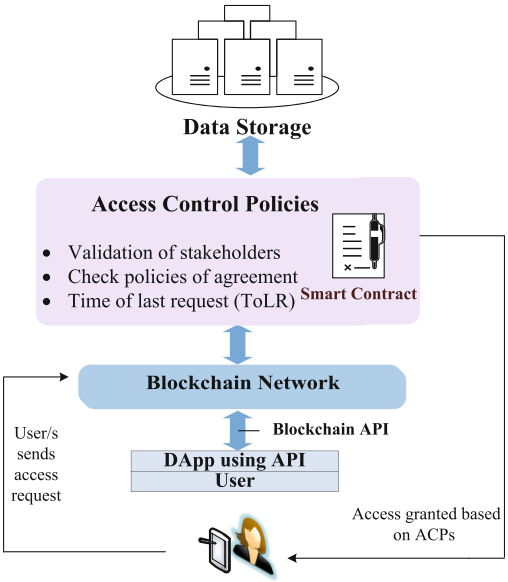


Fig. 1. High level illustration of blockchain-based access control framework.

2.1 Blockchain-Based Access Control Mechanism

There are mainly four entities involved in the blockchain-based access control models. Here, we discuss the functionality of each sub-module as below.

– ***Users with DApp.*** It is a decentralised web application used by front-end users to render the blockchain-based access control application. It contains peer decision makers, rules, and policies scripted in a specific language (Solidity, Vyper etc.) about how the peers are allowed to access information [17]. The interface between users and DApp is mainly via JavaScript Application Programming Interface (API) to establish communication with the blockchain network. It contains a wallet that manages the cryptographic keys and keeps a record of blockchain addresses. A user sends an access request through the DApp. In DApps, users can directly send requests and access data without a single server controlling it like in client-server model. Once the user has sent

the request through DApp, it cannot be tampered or deleted. This leads to secure and open governance. DApps offers various applications in healthcare, financial sectors, gaming, supply chain etc.

- ***Blockchain network.*** After the user sends the access request, it is broadcasted among each peer in the blockchain network. Then, the network of nodes verifies the legitimacy of access requests based on a consensus mechanism. The transactions are cryptographically signed and appended on the main chain after the consensus. Blockchain consensus can be broadly classified into Proof-based and Vote-based. Proof-based consensus are fully decentralised and permissionless. They elect a leader by introducing a game approach to propose a final block value. For example, cryptocurrencies like Bitcoin and Ethereum utilise Proof of Work (PoW) consensus where miners with varying computational power compete against each other to solve a mathematical puzzle to confirm transactions on the blockchain network, and the miner with sufficient proof gets rewarded. Other such consensus are Proof of Stake (PoS) [18], Delegated Proof of Stake (DPoS) [19], etc. They are hard to scale with the growing number of nodes across the network [20]. While the vote-based consensus are simpler than the proof-based as they achieve consensus based on the round of votes. Byzantine fault-tolerant (BFT) consensus protocols are among the vote-based consensus such as Practical Byzantine fault-tolerance (PBFT) [21]. These consensus protocols have high performance but the degree of decentralisation is low. Due to decentralised consensus, blockchain achieves a great level of security. Blockchain network also famously called distributed ledger technology (DLT) records and replicates the transactions across each node. It provides ledger and smart contract or chain code services to various applications. It records the provenance of a digital asset in a distributed, shared, and immutable ledger. The blockchain network operates across a peer-to-peer network of computers without a central authority or intermediary.
- ***Access control policies.*** Each peer in the system has a list of Access Control Policies (ACPs) constituting access agreements between the data owner and data requester. In a blockchain-based system, the involved entities define the access control policies in the smart contract and manage access. Smart contract-based ACPs check any kind of misconduct, time of last request (ToLR), etc. and grants access permission to the requester. It also revokes access control in case of any misconduct. The smart contract-based access policies are not only decentralised but also self-executable. It eliminates the threat of any internal or external attack due to the SSL certificates located at each node rather than using the traditional passwords. Multiple smart contract-based access policies have been proposed for Internet of Things (IoT) systems to achieve distributed and trustworthy access control [22].
- ***Data storage.*** As blockchain is a distributed system, data is stored in computers or nodes across the whole network. Each of these nodes contains a copy of the blockchain ledger. In blockchain-based access control, data can be stored inside or outside the blockchain. Considering the limited block size of the blockchain, some of the existing frameworks come up with the idea of storing data in the cloud while their corresponding hash is packed into

the blockchain. Various existing blockchain platforms have different ways of storing data. For example, Ethereum stores the transaction data in trie only when the transaction is confirmed. Corda [23] uses the concept of states to store data rather than broadcasting to each peer, while Hyperledger [24] uses LevelDB and CouchDB to store the state data. The whole process of data storage can take anywhere ranging from few minutes to hours depending on the congestion in the network.

The bottleneck in the existing [4–6] and our proposed smart contract-based access control model [8] is the slower transaction processing rate for emergency or priority EMR transactions due to the FIFO scheduling, despite having higher importance. In next subsection, we will explain the issue of scalability in detail to get an insight to understand and analyse the performance and scalability of blockchain-based access models.

2.2 Scalability and Performance Issue

Blockchain networks such as Bitcoin avoid the double-spending problem through the consensus protocols where each transaction is recorded and validated by every node on the chain. This ensures transparency, data integrity, and immutability in a decentralised blockchain environment. But, it restricts the scalability as current blockchain systems can process only a few transactions per second. It is mainly affected by key factors, i.e., block mining rate, transaction processing, and waiting time of the transaction in the queue to get processed. This needs to be addressed before blockchain is adopted in real-time systems such as smart healthcare [4]. Blockchain network deals with the mining congestion because of higher transaction generation rate than the transaction processing rate, leading to a longer waiting time for the transaction to get mined. We emphasis that none of the given access models have the provision to process the prioritised transactions or access requests faster and they do not consider the allocation of adequate resources (resource fairness). This factor is of utmost importance and hence cannot be ignored in terms of increasing scalability [25] or optimizing the blockchain network resources. Some of the solutions proposed to address this challenge are a) Off-chain solutions, b) Directed Acyclic Graph (DAG), c) Sharding.

1. *Off-chain:* This approach to tackle scalability in the blockchain network is to store transaction-related data in the local nodes, which are often referred to as off-chain [26]. These local nodes only send a summary or outcome of the transactions to the main chain. Another such solution is creating a network of micro-payment channels to instantly confirm a payment transaction [27]. Such off-chain solutions cannot guarantee the validity or legitimacy of off-chain transactions. To tackle that, often validator nodes are introduced to endorse the transactions but still, the validity is compromised due to the centralisation.

2. *Directed Acyclic Graph:* Another approach is to design blockchain as Directed Acyclic Graph (DAG) network [28,29], where each transaction is linked to multiple transactions rather than the blocks. Theoretically, the higher, the volume of transactions, the faster a DAG network can validate them.
3. *Sharding:* Sharding has been extensively explored in the distributed database systems to alleviate scalability and performance. Database sharding refers to the horizontal partitioning that splits large databases into smaller chunks called shards across multiple servers.

To address scalability problems in conventional blockchain, many blockchain sharding protocols have been introduced. In the standard blockchain sharding, the entire blockchain's state is broken into shards that contain their independent history of state and transaction. Omniledger is among the earliest work conducted to achieve high Visa-level performance in distributed ledgers through parallel intra-shard transaction processing [30]. It introduces a cross-shard protocol to handle transactions affecting multiple shards. Optchain [31] seeks to enhance the placement of transactions in order to minimize the adverse effect of cross-shard transactions on the efficiency of existing sharding proposals. It has implemented the temporal fitness score to measure the probability of transaction should be placed into the shard without causing further cross-shard transactions. However, these sharding techniques enhance scalability by compromising the very core properties of blockchain i.e. decentralisation. Spontaneous sharding has been suggested as a solution where transactions are sharded by the nature of the value transfer system [32]. With each value, a proof is associated that grows with the number of nodes passing. Hence, to keep the transmission costs low, nodes would prefer to keep the transaction in smaller shards than the entire network. It does, however, bring out the downside of low storage capacity.

3 P-sharding: Proposed Framework

In this section, we propose and discuss our novel framework to prioritise and efficiently process emergency transactions in blockchain-based access control mechanisms in smart healthcare. We name this model P-sharding and explain it as follows in detail.

Our proposed model of P-sharding introduces the priority in the sharding mechanism proposed by Ethereum. In this methodology, the transaction is split up among smaller groups of nodes based on the prioritised data, to get validated. Using the assumption that only trauma and emergency departments of any hospital can put priority tag to the EMR transactions. And the emergency EMR transactions get accumulated in the memory pool. The idea behind the P-sharding is illustrated in Fig. 2. The detailed step-wise workflow sequence is given below.

1. Transactions are grouped in the memory pool as they arrive in the system. In every mining iteration, the blockchain is divided into shards. The number of shards which is the length of the transaction array is divided according to

two factors: (a) the transactions load and (b) the number of validators in the network. Then the transactions are randomly divided among all the shards and get collected into their respective sub-block. Each of the shards has its own ledger. These shards process and store a disjoint set of transactions. Validator nodes are randomly assigned to each shard.

2. The transactions with priority are then chosen from the sub-block based on their priority tag. The tag with 0 indicates a non-priority transaction, while 1 represents a transaction of priority. In our proposed model, the transactions are then sequenced as per the priority and held in the respective priority and non-priority mining queues based on FIFO.
3. After sequencing, the randomly selected validators in the network verify the prioritised transactions by allocating their resources prior to the non-priority transactions.
4. Finally, the verified block is generated and added into the sharded blockchain.

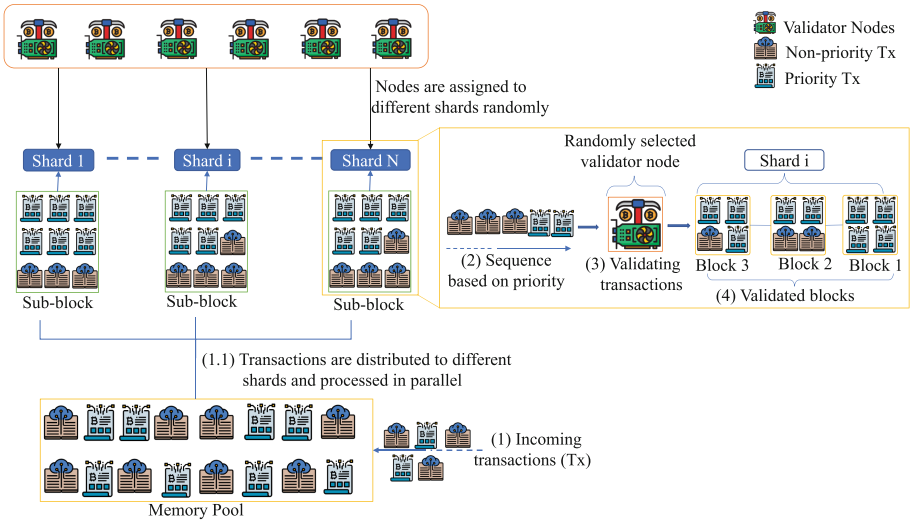


Fig. 2. Proposed P-sharding model.

Since the sharded blockchain has multiple shards, these emergency transactions spend least amount of time to fetch from the memory pool. This results in faster transaction processing and helps in the development of an efficient access control mechanism. Table 1 depicts the structure of the proposed P-shard in which a transaction group is divided into two parts: a) transaction group header and b) transaction group body. The transaction group header is further divided into two sections. The left section is as follows:

- *Shard ID.* Every transaction specifies the ID of the shard it belongs to.
- *Pre-state root.* It represents the state of the root of the shard before the transaction is registered.
- *Priority score.* It is the total count of priority transactions in a block. Our aim of adding a priority score is to reduce the response time and maintain the service quality for the priority transactions.
- *Post-state root.* It shows the state of the root of the shard after the registration of transactions.
- *Receipt root.* It is generated after all the transactions in the shard are registered. It facilitates the cross shard-communication.

While the right part of the transaction group header consists of random validators. They are randomly chosen and responsible to verify the transactions in the shard. And transaction group payload has all the transaction IDs in the shard itself. Each transaction has a field of priority tag of 0 or 1 that represents its non-priority and priority state respectively.

Table 1. Structure of P-shard.

Transaction group header		
Shard ID: 23	<sig #3256>	<sig #4672>
Pre state root: 142c3dfg	<sig #7089>	<sig #8796>
Priority score: 6	<sig #4351>	<sig #2317>
Post state root: 567819ab	<sig #2356>	<sig #6451>
Receipt root: ca4567f7	<sig #1254>	<sig #2478>
Transaction group payload		
Tx a142; PriorityTag:1	Tx a674; PriorityTag:1	Tx a542; PriorityTag:1
Tx a231; PriorityTag:1	Tx a892; PriorityTag:1	Tx a902; PriorityTag:1
Tx a256; PriorityTag:0	Tx a353; PriorityTag:0	Tx a762; PriorityTag:0

Based on the queuing theory principle [33], the incoming shards follow Poisson distribution. Upon arrival, each shard is either in the priority queue with a mean arrival rate of λ_p or in non-priority queue with λ_{np} such that the total rate is

$$\lambda = \lambda_p + \lambda_{np} \tag{1}$$

It is important to notice that the waiting time of the non-priority transactions in the queue is longer than priority transactions, typically depends on the number of transactions waiting to get processed in the priority queue ahead of non-priority queue, such that

$$W_{nq} = \frac{\mu}{\mu - \lambda} W_{pq} \tag{2}$$

where μ is the average processing time of the validators, W_{pq} and W_{nq} are the average waiting time of the transaction in the priority and non-priority queue respectively. General equations for our model where priority transactions are served at the rate μ_1 and non-priority transactions are served at μ_2 are as below:

$$QL_{pq} = \frac{\lambda_p}{\mu_1}\hat{\rho}\frac{\lambda_p/\lambda + (\lambda_{np}/\lambda)(\mu_1^2/\mu_2^2)}{1 - \lambda_p/\mu_1} \tag{3}$$

$$QL_{nq} = \frac{\lambda_{np}/\mu_1}{1 - \lambda_p/\mu_1}\hat{\rho}\frac{\lambda_p/\lambda + (\lambda_{np}/\lambda)(\mu_1^2/\mu_2^2)}{1 - \lambda_p/\mu_1 - \lambda_{np}/\mu_2} \tag{4}$$

where QL_{pq} and QL_{nq} are respective mean queue lengths of prioritised and non-prioritised queue. Total utilization or % system busy time is given by

$$\rho^* = \frac{\lambda_p}{\mu_1} + \frac{\lambda_{np}}{\mu_2} \tag{5}$$

We should observe $\rho^* < \hat{\rho}$ if

$$\frac{\lambda_p}{\mu_1} + \frac{\lambda_{np}}{\mu_2} < \frac{\lambda}{\mu_1} = \frac{\lambda_p + \lambda_{np}}{\mu_1} \tag{6}$$

Both the priority and non-priority mining queues in our P-sharding model preserve non-preemptive scheduling within due to the cases of conflict between transactions of the same priority. The transaction that goes to the head of the shard deemed valid and served first. Non-preemptive priority scheduling is time saving also. Our system is based on the following assumptions.

- Priority and non-priority mining queues are kept of infinite buffer capacity (B) to avoid transactions being discarded in order to maintain Quality-of-Service (QoS).
- In our simulations, we keep all validators of the same capacity for fair play. Although the above assumptions might not be realistic and we will work on this in our future research as queuing is another research domain.

The process of sharding of prioritized blocks containing priority transactions is explained in Algorithm 1.

4 Implementation and Performance Analysis

In this section, we present the implementation setup and analysis of our simulated results.

Algorithm 1. P-sharding: prioritised sharding

Input: *Transactions (Tx), No of validators, Capacity of each shard, Minimum validators per shard, Previous block hash, No. of transactions in memory pool, Current state*

Required: *Queue capacity, Priority tag, Arrival rate of transactions (λ_p,λ_{np}), Service time (t_s)*

Output: *Minedblocks == (M_B), Priority score (PS), Departure time from queue (t_d)*

1. *Set PriorityTx Arrival rate*$\rightarrow\lambda_p$, *Non-PriorityTx Arrival rate*$\rightarrow\lambda_{np}$
2. *Set Priority MiningQueue capacity*$\rightarrow B$, *Set NonPriority MiningQueue capacity*$\rightarrow B$
3. *Set Validators*$\leftarrow M_T$
4. *Set Minimum validators per Sharded Block*$\leftarrow M_{SB}$
5. *Set Maximum Sharded Block*$\leftarrow S_{MX}$
6. *Current state*$\leftarrow C_{st}$
7. *Memorypool*$\leftarrow$ *transactions(Tx), ServTime(ts)*$\leftarrow$*Start*
8. *Current state*$\leftarrow C_{st}++$
9. *AriTime*$\leftarrow t_a$, *priority tag*$\leftarrow$ 1// 0
10. ***If***(Tx $\leftarrow$priority tag[1])
11. *{*
12. tx_{p1},tx_{p2},tx_{p3}....tx_{pn} =:*tx*
13. ***Priority MiningQueue*** $\leftarrow tx_{p1},tx_{p2},tx_{p3}....tx_{pn}$
14. *}*
15. ***else***
16. *{*
17. $tx_{np1},tx_{np2},tx_{np3}....tx_{npn}$ =:*tx*
18. ***NonPriority MiningQueue***$\leftarrow$ $tx_{np1},tx_{np2},tx_{np3}....tx_{npn}$
19. *}*
20. $t_s \leftarrow end$
21. ***Result***$\leftarrow (M_B)^{new} \leftarrow tx_{p1} + tx_{p2} \cdots + tx_{pn} \cdots + tx_{npn}$, $PS = count(tx_p)$, $DepTime(t_d) \leftarrow t_a + t_s$

4.1 Simulation Setup

We simulate our model on Python in a controlled permissioned blockchain environment. The study designed a blockchain scenario by creating multiple Ethereum virtual nodes on each device with configuration of Lenovo ThinkCentre, Intel(R)CoreTM i5-7500 CPU @3.40 GHz, Windows 10 Enterprise, 64-bit, 16 GB. It mainly aims to measure and compare the overhead of the existing sharding approach [15] and our proposed prioritised sharding model.

Our model consists of a source station: transactions, one queuing station: memory pool, a fork station: P-sharding, two sets of queues: priority mining queue and non-priority mining queue, one join station, and a sink station: discharge block as shown in Fig. 3. The arrival rate for the transactions source station is generated randomly. The memory pool is modeled using M/M/1 where the arrival of transactions is Poisson distributed, service time is exponentially distributed, and the number of server is one. While on the other hand, mining

pool is modeled as M/M/c since, in reality, several validators compete in parallel to solve a single block of puzzle. The fork station named P-sharding splits the transactions based on their priority tag. This station is used to achieve fast processing and reduce the service time of transactions in the mining pool. This task is synchronized and forwarded for processing in the priority and non-priority mining queue respectively. After all these tasks are processed and completed, they are joined again to dispatch the final blocks to the blockchain network.

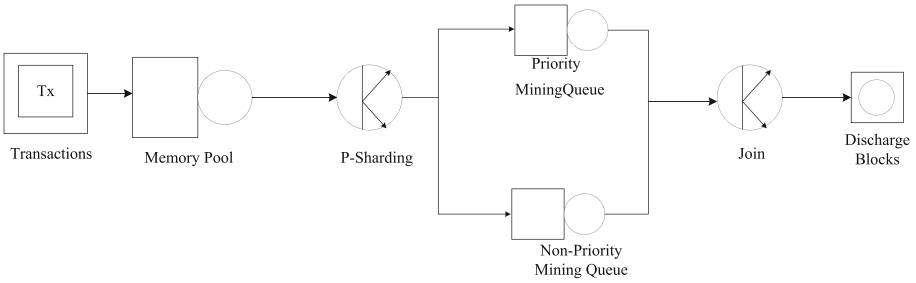


Fig. 3. Simulation model for P-sharding.

4.2 Simulated Results

In this section, we present simulated results of our model and compare them with the basic sharding model. We have utilized updated Ethereum[2] actual data parameters to carry out our simulations. The QoS performance metrics with proposed P-sharding model and basic sharding are shown in Fig. 4 and 5 respectively and explained below.

- *Mining queue count after sharding.* These are the accumulated sharded transactions that are waiting in queue to get mined.
  - *Priority mining queue count after sharding.* Due to the probabilistic approach, the priority mining queue has more priority transactions than the non-priority mining queue. In Ethereum simulation, the average priority mining queue count comes out 0.5772, which improves significantly as compared to the basic sharding where all transactions are considered evenly irrespective of their priority as shown in Fig. 4(a), 5(a) respectively.
  - *Non-priority mining queue count after sharding.* The average non-priority mining queue count comes out to be 0.3837 in Fig. 4(a).
- *Response time.* It is the average mining time of a shard, or mining time of both priority and non-priority mining queue which is almost the same with negligible variations due to dynamic arrival and mining rates. In Ethereum, both queues have an average of 0.0712(s) simulated mining rate with the input or actual mining rate of 0.0714(s) shown in Fig. 4(b) which comes in coherence with sharding results in Fig. 5(b).

[2] https://etherscan.io/.

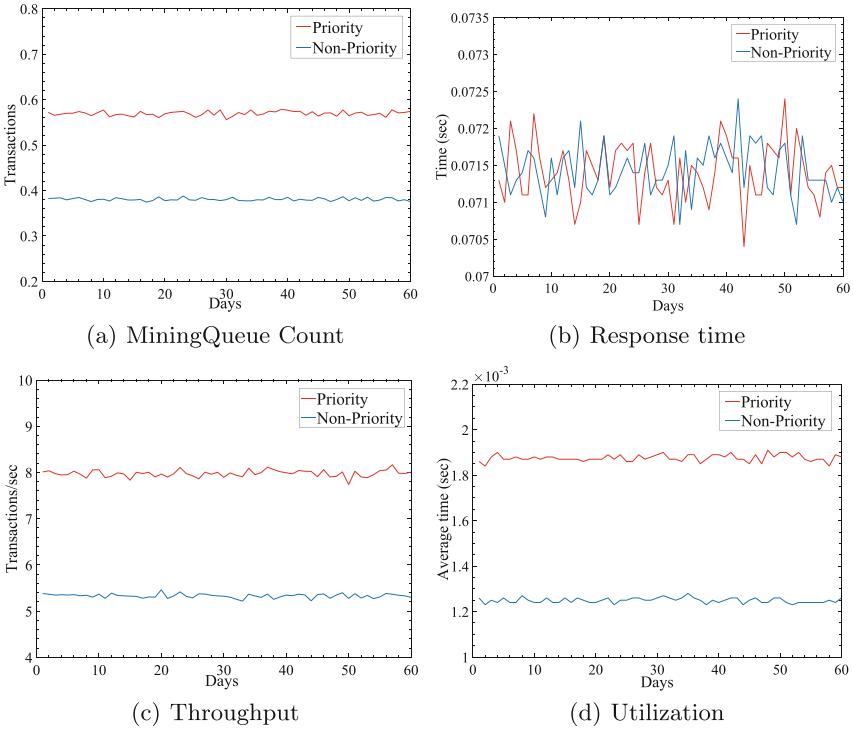
- Throughput: The rate at which transactions depart from a mining queue, i.e., the number of transactions completed in a unit time depicted in Fig. 4(c).
  - *Priority mining queue throughput.* Priority mining queue has higher throughput. In other words, mining of a prioritised transaction is faster as compared to non-priority mining queue. In Ethereum simulation, the average throughput in the priority mining queue is 7.9877 tps (refer Fig. 4(c)) which is faster than the basic sharding where all the transactions irrespective of their priority are treated the same as shown in Fig. 5(c).
  - *Non-priority mining queue throughput.* The departure of transactions in a non-priority mining queue is comparatively slower. In Ethereum simulation, the average throughput in non-priority mining queue is 5.3248 tps.
- *Utilization.* It can be defined as the percentage of time a station is used (i.e., busy). It ranges from 0(0%), when the station is idle, to a maximum of 1(100%), when the station is constantly busy mining transactions for the entire simulation run. The utilization rate with single server S is $\mathrm{U} = l\mathrm{S}$ and subsequently, utilization with m number of servers is $\mathrm{U} = l\mathrm{S}/\mathrm{m}$ where l is arrival rate.
  - *Priority mining queue utilization.* In the priority mining queue, we set up three servers (validator nodes) mining three sharded blocks simultaneously with utilization $1.86{*}10^{-3}$ less than 1 (Fig. 4(d)), which means the priority mining queue is not fully occupied. Hence our system achieves $\rho^* < \hat{\rho}$. In the basic sharding model, the system is congested due to a high utilization rate (refer Fig. 5(d)).

To further validate our model, Table 2 shows the comparison between the actual and simulated results of basic sharding and our P-sharding model. The values indicate the significant improvement in the stated parameters with our P-sharding model in comparison with the basic sharding model.

4.3 Performance Analysis

In this section, we evaluate the performance of our proposed P-sharding mechanism as shown in Fig. 6 and 7.

- *Latency per number of shards.* The latency (or average confirmation time) of a transaction is measured by the time the transaction is sent until it is committed to the blockchain. It is a significant factor in block propagation time. A large number of confirmed sharded blocks will multiply this delay as shown in Fig. 6(a). As the latency changes linearly with the increasing number of shards and transaction rate. This shows that the two mining queues are able to streamline the flow of transactions.



(a) MiningQueue Count (b) Response time

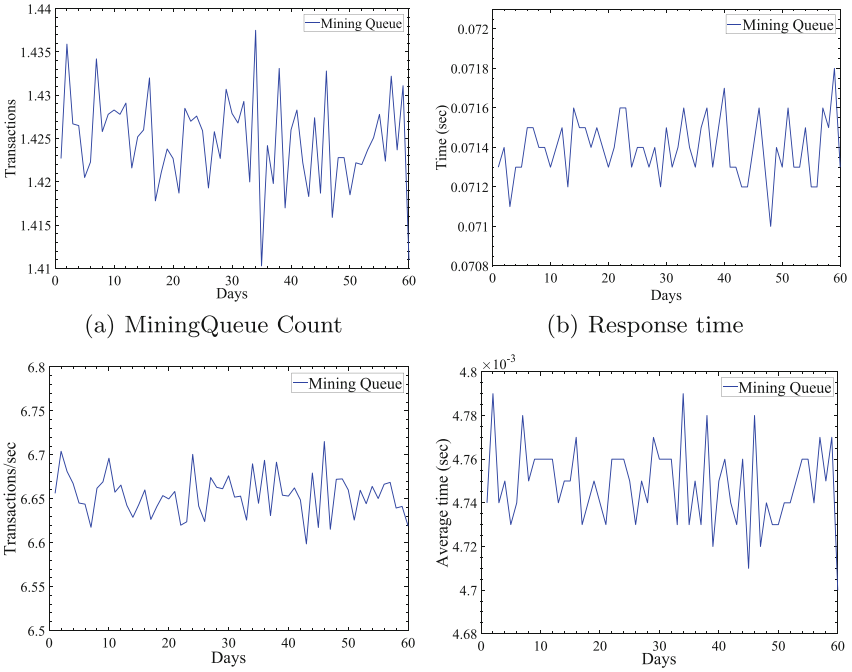
(c) Throughput (d) Utilization

Fig. 4. Simulation results of Ethereum real statistics with P-sharding.

Table 2. Comparison between the simulated results of sharding and proposed P-sharding model.

Parameters	Sharding	P-sharding	
		Priority-mining queue	Non-priority mining queue
Mining queue count (tx)	1.4359	0.5772	0.3837
Response time (sec)	0.0713	0.0712	0.0712
Throughput (tx/sec)	6.4379	7.9877	5.3248
Utilization	0.00479	0.00186	0.00126

– *Latency per transaction rate.* We analyze the latency with respect to the rate of input transactions in the mining queue i.e. the number of transactions coming in unit time for mining. As shown in Fig. 6(b), it increases with the increase in input transactions to the mining queue with respect to the increase in number of shards.



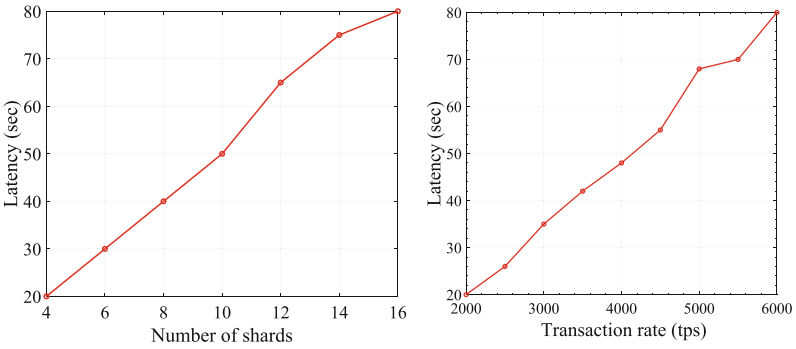
(a) MiningQueue Count (b) Response time

(c) Throughput (d) Utilization

Fig. 5. Simulation results of Ethereum real statistics with sharding.

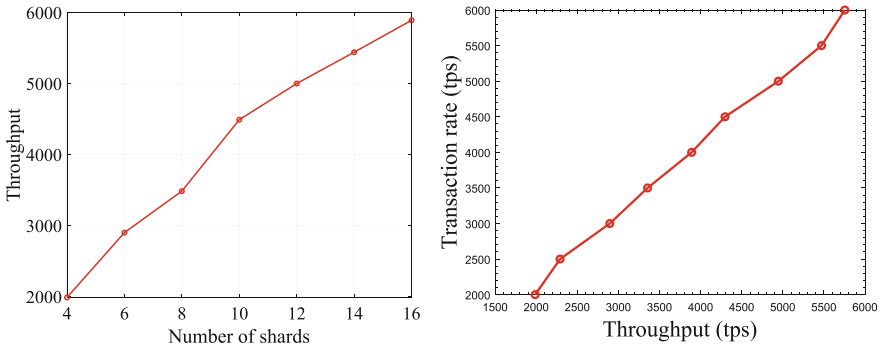
– *Throughput per number of shards.* The transaction throughput is the rate at which valid transactions are committed by the blockchain at one time, i.e., committed at all nodes of the network, usually measured in transactions per second. Throughput decreases with an increase in the number of shards and input transactions as shown in Fig. 7(a). These results indicate that the proposed scheme achieves its highest throughput with 16 shards, when running with different combinations of transaction rates and number of shards.
– *Throughput per transaction rate.* The results in Fig. 7(b) show that our proposed scheme is capable of handling variable transaction rates as the throughput changes linearly without lagging. It guarantees that there is no backlogging or congestion in the system.

Our idea of prioritised sharding with parallel mining divides the overloaded transactions into shards allowing multiple concurrently maintained sub-chains to record them. In this way, we are able to improve the processing rate of emergency transactions significantly while keeping the whole network less congested. This eventually improves the scalability of the network.



(a) Latency per number of shards (b) Latency per transaction rate

Fig. 6. Performance analysis in terms of latency.



(a) Throughput per number of shards (b) Throughput per transaction rate

Fig. 7. Performance analysis in terms of throughput.

5 Conclusion and Further Discussion

We propose a framework to streamline the processing of prioritised transactions in a blockchain-based access model. With our method of prioritised sharding, the emergency transactions were divided among smaller shards and mined in parallel. This improved the processing rate of prioritised transactions drastically. Also, the mining congestion got reduced with our proposed P-sharding model as the waiting time for the transaction confirmation time reduces. The QoS improved in our proposed P-sharding algorithm in the context of prioritised transactions as compared to the existing sharded blockchain. In this paper, the performance of a high-level blockchain-based access control model had also been evaluated which provided insights about developing a prioritised, scalable, and efficient decentralised access control framework for healthcare. The simulation results are promising and give valuable insights into our model.

However, in certain ways, the model needs a more in-depth evaluation of how validators come to a decentralised consensus to select the emergency transactions and avoid selfish mining. In real scenario, there exist transactions with varying priorities. Therefore, we should include them with and implement the whole system on a larger scale. Further, to enhance interoperability, it is important to analyse the work on the cross P-shard communication to efficiently share prioritised EMRs across different blockchain-based access control models. Another challenge is how to join these prioritised sharded transactions and dispatch them to the blockchain network in a decentralised manner, which still needs to be deeply explored. We welcome the research community to contribute to alleviate these issues and explore further challenges in this domain.

References

1. Sun, J., Fang, Y.: Cross-domain data sharing in distributed electronic health record systems. IEEE Trans. Parallel Distrib. Syst. **21**(6), 754–764 (2009)
2. Osborn, S., Sandhu, R., Munawer, Q.: Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM Trans. Inf. Syst. Secur. (TISSEC) **3**(2), 85–106 (2000)
3. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F., Voas, J.: Attribute-based access control. Computer **48**(2), 85–88 (2015)
4. Novo, O.: Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J. **5**(2), 1184–1195 (2018)
5. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: MedRec: using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30. IEEE (2016)
6. Roehrs, A., da Costa, C.A., da Rosa Righi, R.: OmniPHR: a distributed architecture model to integrate personal health records. J. Biomed. Inform. **71**, 70–81 (2017)
7. Introduction to smart contracts. https://solidity.readthedocs.io/en/v0.5.6/introduction-to-smart-contracts.html/
8. Saini, A., Zhu, Q., Singh, N., Xiang, Y., Gao, L., Zhang, Y.: A smart contract based access control framework for cloud smart healthcare system. IEEE Internet Things J. **8**(7), 5914–5925 (2020)
9. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.-L.: Blockbench: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1085–1100. ACM (2017)
10. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. IEEE (2017)
11. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. **18**(3), 2084–2123 (2016)
12. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P 2013 Proceedings, pp. 1–10. IEEE (2013)
13. Nakamoto, S., et al.: Bitcoin: a peer-to-peer electronic cash system (2008)
14. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum project yellow paper, vol. 151, pp. 1–32 (2014)

15. On sharding blockchains. https://github.com/ethereum/wiki/wiki/Sharding-FAQ
16. Dang, H., Dinh, T.T.A., Loghin, D., Chang, E.-C., Lin, Q., Ooi, B.C.: Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 International Conference on Management of Data, pp. 123–140 (2019)
17. Bogner, A., Chanson, M., Meeuw, A.: A decentralised sharing app running a smart contract on the ethereum blockchain. In: Proceedings of the 6th International Conference on the Internet of Things, pp. 177–178 (2016)
18. Saleh, F.: Blockchain without waste: proof-of-stake. Available at SSRN 3183935 (2020)
19. Larimer, D.: Delegated proof-of-stake (DPoS). Bitshare Whitepaper (2014)
20. Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Liu, Y.: A survey on the scalability of blockchain systems. IEEE Network **33**(5), 166–173 (2019)
21. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. (TOCS) **20**(4), 398–461 (2002)
22. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., Wan, J.: Smart contract-based access control for the internet of things. IEEE Internet Things J. **6**(2), 1594–1605 (2018)
23. Brown, R.G., Carlyle, J., Grigg, I., Hearn, M.: Corda: an introduction. R3 CEV, vol. 1, p. 15, August 2016
24. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)
25. Goel, S., Singh, A., Garg, R., Verma, M., Jayachandran, P.: Resource fairness and prioritization of transactions in permissioned blockchain systems (industry track). In: Proceedings of the 19th International Middleware Conference Industry, pp. 46–53 (2018)
26. Wood, G.: Polkadot: vision for a heterogeneous multi-chain framework. White Paper (2016)
27. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
28. Popov, S.: The tangle. White Paper, vol. 1, p. 3 (2018)
29. Churyumov, A.: Byteball: a decentralized system for storage and transfer of value (2016). https://byteball.org/Byteball.pdf
30. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: Omniledger: a secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE (2018)
31. Nguyen, L.N., Nguyen, T.D., Dinh, T.N., Thai, M.T.: Optchain: optimal transactions placement for scalable blockchain sharding. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 525–535. IEEE (2019)
32. Ren, Z., Cong, K., Aerts, T., de Jonge, B., Morais, A., Erkin, Z.: A scale-out blockchain for value transfer with spontaneous sharding. In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), pp. 1–10. IEEE (2018)
33. Kleinrock, L.: Queueing Systems, Volume 2: Computer Applications, vol. 66. Wiley, New York (1976)