PeloPartition: Improving Blockchain Resilience to Network Partitioning

Juncheng Fang, Farzad Habibi, Kevin Bruhwiler, Fayzah Alshammari,
Abhishek Singh, Yinan Zhou and Faisal Nawab
Department of Computer Science
University of California, Irvine
{*junchf1, habibif, kbruhwil, fayzaha, abhishas, yinanz17, nawabf*}*@uci.edu*

***Abstract*—Blockchain has gained considerable traction over the last few years and plays a critical role in realizing decentralized and cryptocurrency applications. A challenge that has been overlooked in prior blockchain algorithms is that they do not consider large-scale network outages and relied on the assumption of a reliable global network connectivity. In the event of a large scale network partition, forks may occur between partitioned regions. After the partition ends they will be discarded, leading to the loss of many blocks and a considerable amount of wasted work. This paper presents** PeloPartition**, which provides a sharding mechanism to improve blockchain's resilience to the possibility of a global internet outage. In** PeloPartition **we form consensus groups dynamically and consider the partitioning of the group as a hint to split the blockchain into branches and guarantee that all of them will be merged after the network is recovered. We indicate different methodologies to ensure blockchain security while partitioning occurs. Our experiments use simulations to show how this approach can improve the performance of blockchain algorithms and prevent wasted computational power during partitioning.**

***Index Terms*—resilience, partitioning, blockchain, collective signing, PeloPartition**

I. Introduction

Blockchain, initially invented for the Bitcoin [1] cryptocurrency, is a decentralized ledger. This permissionless infrastructure is used for recording different transactions without needing a central authority. Particular nodes called miners can create a block consisting of multiple transactions and, using proof-of-work, nodes reach consensus to append new data to the existing ledger. All transactions are maintained and can be retrieved by any node with a copy of the ledger. Many applications have been implemented on blockchain infrastructure, and their adoption is growing rapidly. For example, blockchain can be used for fault-tolerant communication middleware [2], or as a global trusted log [3]. IBM researchers forecast that the market based on blockchain is expected to reach $60.7 billion by 2024 [4]. This wide adoption makes it important to consider the challenges associated with making blockchain resilient to different types of failures. Recent work [5] has shown the possibility of massive disruptions to the Internet caused by solar superstorms in the near future. Such events can disconnect different geographical regions from each other, potentially for a period of months.

Blockchain systems create blocks approximately in prespecified time intervals (for instance, Bitcoin generates one block every 10 minutes). During a network partition the main chain may fork, resulting in miners being segregated into disconnected regions. Such a partition would result in the creation of parallel blocks in affected regions by the miners. Since blockchain algorithms use the longest chain rule to accept only one of the branches when the network partition is resolved all the blocks in other branches will be discarded. This wastes significant hash power and affects the reliability of the ledger. We aim to reduce such impacts by enabling different forks to be merged in our work.

Blockchain sharding has been explored by many previous works [6]. Sharding technologies partition the blockchain network into various groups, while these groups maintain their decentralized ledger and use a secure cross-shard communication protocol. For instance, RapidChain [7] partitions the set of nodes into multiple smaller groups of nodes called committees that operate in parallel on disjoint blocks of transactions and maintain disjoint ledgers. Although these sharding algorithms provide a partitioning mechanism for blockchain systems, they do not consider network disruption because sharding is predetermined and nodes in one shard can still be partitioned. Their primary focus is improving the scalability and performance of blockchain by increasing parallelism and the number of transactions per second.

This paper presents a partitioning algorithm to tolerate global-scale network partitioning. At a high level we design PeloPartition, a blockchain system that splits block creation into different branches when a network partition happens and merges existing branches into one when the network partition is resolved. We also present the security mechanisms to prevent conflicts between branches and tolerate malicious behaviors by both miners and clients. This paper aims to ensure that each fork works independently when the network is partitioned. After a network recovery, the previous progress in each partition is preserved and conflicts between the partitions are resolved.

II. Background

Motivation Traditional blockchain protocol relies on a globally connected network. It is not designed to handle a network partition that will cause all blocks in shorter forks to be dropped after the network recovers. In this scenario, part of the ledger will be lost, which greatly impacts the

reliability of blockchain system. PeloPartition is designed to tolerate large-scale network partitions which can be caused by both natural phenomena and network attacks performed by malicious entities.

We haven't encountered severe natural events that can completely partition the network infrastructure since the advent of modern Internet technology. However, the author in [5] presents the possibility of a large solar storm in the near future. Such event can destroy the submarine fiber-optic cables connecting different continents, cutting down intra-continental network communication. The global-scale network partition caused by such a solar storm could last for several months.

Partitioning attacks are notable concerns that should be considered in regards to blockchain resilience. Different attackers can partition the blockchain's network by intercepting a small number of key messages. Authors in [8] demonstrate the feasibility of routing attacks on blockchain. The goal of partitioning attacks is to cut connections between a set of nodes and the rest of the network. The authors present a practical way of using BGP hijacking to perform such an attack on the Bitcoin network. Authors in [9] present a stealthier partitioning attack on the blockchain called EREBUS. The EREBUS attack partitions the Bitcoin network without any routing manipulations, making the attack undetectable to the control plane and data plane. This attack makes the autonomous adversary system a natural man-in-the-middle network of all the peer connections of one or more targeted Bitcoin nodes by influencing the targeted nodes' peering decisions. In both works the authors suggest different ways to mitigate routing attacks by making the attacks more challenging. However, they do not entirely prevent attackers from performing partitioning attacks on the network.

Bitcoin Blockchain systems were introduced by Bitcoin [1]. Bitcoin is a decentralized ledger that maintains proposed transactions as a chain of blocks mined by different miners. Miners collect transactions from the transaction pool and include them in a new block. Each block consists of a cryptographic hash of the previous block, a Merkle tree [10] of new transactions to be committed, and a solution for a cryptographic puzzle as proof of work (PoW). PoW validates blocks in this system. A block is considered valid if it has the correct solution for the cryptographic puzzle, which varies based on the network's current difficulty parameter. In Bitcoin the difficulty is tuned periodically (every 2016 blocks). The adjustment of the difficulty is performed to maintain the frequency of blocks to be close to one block every 10 minutes.

Any miner could add a new block to the blockchain by simply publishing it into the overlay network. When multiple miners create a new block with the same position, the blockchain will be divided into two branches. This phenomenon is called a *fork* in the blockchain systems. The Bitcoin protocol proposes mining on the heaviest chain and discarding the other chain's transactions to resolve the fork. Therefore, branches and blocks outside the main chain will be discarded.

A fork can happen when different network parts are disconnected, or messages are delayed. In an ideal network, block dissemination takes seconds. Forks occur on average about every 60 blocks [11], which can be ignored as a insignificant problem. However, the cases we discussed for network partitioning can take a long time to be resolved [5], which has the potential of increasing the frequency and severity of forks. This leads to wasting a lot of computational power.

In this work, we replace the longest chain rule by proposing a solution to consider all chains as part of the blockchain. Our solution helps merge the different forks into the main fork.

Byzantine Fault Tolerance The Byzantine Generals problem [12], [13] refers to to the problem of reaching agreement among a group of nodes that might act in arbitrary (e.g., malicious) ways. Authors in [13] show that at least $3f + 1$ participants are required to tolerate f malicious participants. The Practical Byzantine Fault Tolerance (PBFT) [14] is one of the most influential solutions to the Byzantine Generals problem. The normal-case PBFT protocol consists of 3 separate rounds:

(1) **Pre-prepare**: The leader proposes the next record be committed by broadcasting a message consisting of the record.
(2) **Prepare**: When participants receive a PrePrepare, they enter this phase and send $Prepare(m)$, where m is the PrePrepare message they have received. The nodes will wait for $(2f + 1)$ to prepare messages and then publish this observation with a $Commit(m)$ message.
(3) **Commit**: Participants wait for $(2f + 1)$ messages to confirm that enough nodes have reached an agreement and consider it the final decision.

The normal-case protocol works correctly when the leader is non-faulty. To tolerate faulty leaders, view-change protocol is introduced to transit leadership and ensure the correctness of the PBFT protocol. We refer the interested reader to the full paper [14] for the details about the view-change protocol.

The PBFT protocol can reach consensus much faster than the traditional blockchain protocol. With PBFT commitment is irreversible, while with blockchain there is only probabilistic guarantee for a block to be committed. However, the PBFT protocol is limited to a static group of participants and it cannot scale to a large cluster. In our work we construct consensus group dynamically to leverage the benefit of the PBFT protocol.

Collective Signing CoSi [15] is a protocol for scalable collective signing, which ensures that a leader statement is validated and publicly signed by a diverse group of witnesses. CoSi uses Schnorr multi-signatures [16] with a communication tree for scalability. For each message that the leader wants to be collectively signed, he runs a four-phase protocol requiring two round-trips over the leader and its witnesses tree (Announcement, Commitment, Challenge, and Response). The result of this protocol is a signature that can be verified efficiently by anyone. We refer interested reader to their work [15] for more technical details.

ByzCoin [17] is a blockchain system that provides a stronger guarantee on the commitment of blocks. It uses CoSi

to implement a scalable Practical Byzantine Fault Tolerance (PBFT) [14] algorithm to reach consensus between a dynamically formed consensus group. CoSi does not directly implement the PBFT protocol, but it can be used as a primitive that leaders can utilize to collect and aggregate the PBFT protocol's messages. Authors in ByzCoin combine two sequential rounds of CoSi to implement a single round of PBFT protocol. The first run of CoSi implements the pre-prepare and prepare phase of PBFT, in which the leader obtains proof from a two-thirds super-majority quorum of consensus group members that the leader's proposal can enter the commit phase. Then, the leader initiates the second round of CoSi to implement the commit step of PBFT. In ByzCoin, the miners of previous *m* blocks of a new block will become the consensus group to verify and co-sign this block. Once a block is signed, every node can safely confirm that this block will be preserved in the main chain without waiting for multiple blocks to be published. By using this algorithm as a building block, authors implement ByzCoin, a blockchain system to optimize transaction commitment and verification as their primary goal.

We extend ByzCoin's proposed scalable PBFT algorithm to collectively sign new blocks by a window of previous miners as a consensus group. The partition of the consensus group is used as a hint to determine if a network partition happened.

III. System Design

A. System and Security Model

PeloPartition is a blockchain protocol that aims to tolerate network partitioning. It consists of N miners working on appending new blocks to the existing chain, and an arbitrary number of clients that send transactions to miners to include in the chain. Each miner i has a finite amount of hash power resembling the number of hash operations a node can perform in a fixed amount of time.

A subset of miners could be malicious and act arbitrarily at any time. Therefore, Byzantine faults can happen during the execution of the algorithm. As the PBFT protocol can tolerate up to 1/3 of faulty nodes, we assume that the total hash power of all malicious nodes is less than 33% of the total hash power in the system.

During network partitions, the set of nodes is divided into different regions. We assume that each node, including both miners and clients, can only communicate with other nodes that are in the same region. There is no intra-region communication allowed. We call this type of network partition as *Network Hard Partition*. We also assume that the malicious nodes are distributed uniformly among regions so that the assumption of no more than 33% faulty nodes still holds in each region.

B. Overview

In PeloPartition, a window of w previous miners will form a consensus group and sign the new block using the PBFT protocol introduced in ByzCoin [17]. For instance, Figure 1 shows when the window size is set to 3, each block will be collectively signed by the miner of the block and the miners that created three previous blocks. When a network partition happens, the consensus group will be divided and the new block in each partition is signed by only part of the group. In Figure 1, two branches are created after the network partition happens and the first blocks of them are signed by disjoint sets of miners.

After the network recovers, miners will receive all branches. The miner of the new block forms a consensus group by selecting a set of miners from each branch. For instance, in Figure 1, a merge between two branches happens, and nodes from both partitions sign the new block. However, there can be conflicts between branches so merging them into one chain will affect the integrity of the ledger. We introduce two mechanisms to prevent conflicts, ensuring that merging can be performed safely. We will also discuss the defense against possible issues of using consensus group.

In the following subsections we will delve into the details of the design and discuss each part in more detail.

C. Signing Blocks With CoSi

We adopt the design of ByzCoin [17] to verify and sign blocks with a dynamic group of consensus miners using CoSi. The dynamic consensus group for a new block is formed by the miners of the previous *w* blocks and the miner of the new block itself. Since it is a sliding window on the chain, the window will be moved right by one every time a block is mined. The window size *w* can be defined by the number of blocks in a specific time window. For example, bitcoin produces 144 blocks per day, so with *w=143*, the miners of blocks in one day will be responsible for signing new blocks. It ensures that the consensus group members are recently active. Also, the mining rewards and transaction fees of a new block will be split across all members who participate in the signing procedure, which encourages miners to be active. So consensus group members will rarely be unavailable with the above two mechanisms.

The miner of the new block will be the leader of the consensus protocol. As in ByzCoin, the BFT consensus protocol is implemented using two consecutive rounds of CoSi initiated by the leader. The first round implements the pre-prepare and prepare phase of the BFT protocol, ensuring that the super-majority of the nodes agree on the validity of the transactions in new blocks. The second round implements the commit phase so the super-majority knows that the proposal is accepted and the new block is committed. The collective signature from the second phase will be the commitment signature of the new block as proof to the network that the new block is valid. Once a block is signed, we can safely treat it as committed without waiting for multiple blocks to be mined after this block.

The advantage of using CoSi for ByzCoin is that it can improve the scalability of consensus protocols like PBFT and provide an easily verifiable signature to other nodes. For PeloPartition, we are more interested in the metadata of the signature, which contains the information of which node has signed and which has not, because the number of signed nodes will be a crucial hint to reflect the status of

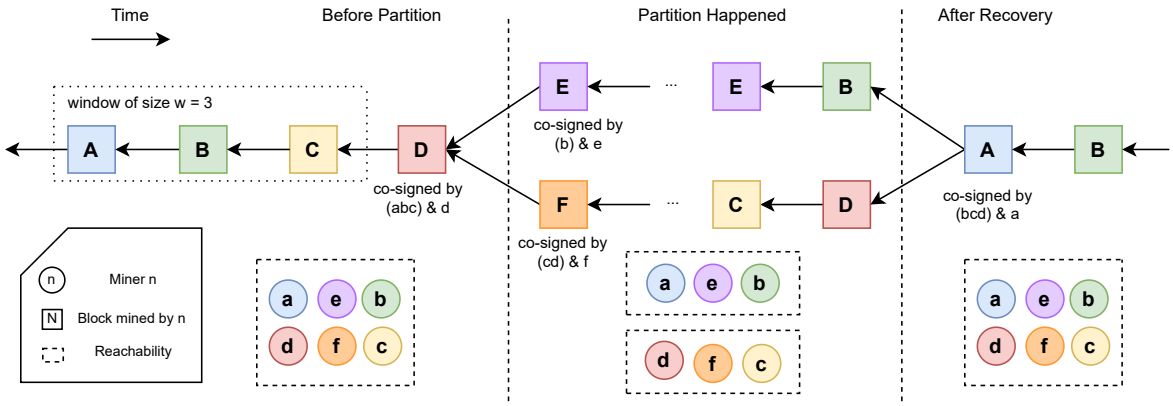


Fig. 1: **PeloPartition Design Overview**–Nodes inside a partition can reach each other. Each block will be signed by a window of size w of previous miners. When partitioning happens, blockchain will be split into different branches. After recovery, branches will be merged into a single chain.

the network. If a new block is signed by only half of the consensus group, a network partition likely happened, and this partition has around half of the hashing power of the whole network, assuming that the hashing power of each partition is proportional to the number of consensus miners in that partition. Because a region with more hash power has is more likely to produce more blocks, within a long enough time frame the number of blocks produced by one region can be used to estimate its relative hash power.

D. Splitting And Merging

Our goal is to design a blockchain system that can be split into branches when a network partition happens and preserve the progress in each branch after network recovery. In this section, we will discuss how the blockchain is split and merged, while taking the network condition into account. We will also discuss how to resolve several threats introduced by the network partition and our new blockchain design.

Splitting into branches When a network partition happens, the consensus group may not be able to reach agreement due to not having enough participating members. We modify the above signing approach with one more phase, discovering phase, before the two-round CoSi to adjust the size of the consensus group according to the network condition.

The purpose of discovering phaseis to verify the availability of each consensus miner. The miner of the new block will broadcast a validation message to all consensus miners and wait for their response until timeout. The set of responding miners will form the actual consensus group. Finally, the miner of the new block initiates the two-round CoSi by sending the new block together with the set of actual consensus members.

The miner of the new block can be malicious, and the consensus group it proposed may be wrong. To ensure the correctness of the set of participants, the consensus protocol will agree on not only the validity of the transactions in the new block but also the set of participants in the first CoSi round. That means each member will broadcast a validation message to all consensus miners like in discovering phase, and check if the set of responding nodes matches the proposed consensus group by the leader. This approach will introduce extra message overhead, and a simple but effective optimization is to only perform this extra verification of participants only when the number of inactive consensus miners is higher than a certain threshold, which means a network partition is likely. Consequently, the modified two-round CoSi only happens at the first block of each branch.

When a block is signed by a consensus group with size smaller than the window size w all nodes receiving this block know that the network has been partitioned. The size of the consensus group w_b indicates the size of the branch, and $\frac{w_b}{w}$ reflects the portion of hashing power owned by that branch. Nodes can also easily identify the branch by the signature of the first block in that branch. Both the branch size and branch identity will be useful to handle some threats introduced by partition, and we will discuss them in the ledger integrity section.

See Figure 1 as an example. When a network partition happens, miner *e* of new block E sends out validation request to miners of previous w=3 blocks which are *bcd*, and can only receive the response from *b*. As a result, the consensus group of block E will be only *be*. They then run the two-round CoSi to verify and sign on block E. On the other hand, another block F is mined, and the corresponding miners in that partition will be signing on F. Now, the network is split into two branches, one with one-thirds of hashing power and another with two-thirds.

Merging branches Merging branches happens after the network recovers. When a miner can see multiple branches simultaneously, it will start mining on a block that is pointing to the last block of each branch, which means the new block has multiple parent blocks instead of one.

For this merging block, the consensus group is formed in a different way. For each branch, the miner of the last w_b (branch size) blocks will be selected, and there will be precisely w

previous miners being selected among all branches. If the miner successfully mined the merging block, it would lead the signing process with the consensus group in the normal way. When any node receives a signed merging block, it knows that the network is recovered. Moreover, with the merging block pointing to all branches, we keep track of all previous blocks and transactions.

See Figure 1 as example. Before the network recovers, two branches exist. After recovery, every node in the network can see both branches. Miner *a* successfully mined a block that is pointing to the last block of two branches, B and D. Since the branch size of the upper branch is 1, miner b is selected, and miner *cd* is selected from the lower branch with size 2. Then miner *a* initiated the signing process with consensus group *abcd* on block A. Once block A is signed, the chain recovers and the progress of the two branches is preserved.

E. Ledger Integrity

This section discusses the possible threats introduced by network partitions and PeloPartition design. We also propose a solution to handle those threats to ensure the ledger's integrity.

Conflicting transactions In the blockchain, clients send their transactions to miners, and miners will add the transaction into the transaction pool. Miners select a large number of transactions from the pool and pack them into one block. When a partition happens, two miners in different partitions may have the same transaction in their pool. In this way one transaction may be committed to two different branches.

The most efficient way is to force all miners to clear their pool after the partition, but there is no way to guarantee that they will do so. To handle this problem, we require clients to attach the branch identifier (the signature of the first block in the branch) when they send out the transaction. A transaction is considered invalid if the branch identifier is wrong. In this way we enforce the property that all transactions included in a branch are sent out after the partition, which means they can not be committed to multiple branches. Since the first block of the branch is clueless about the identifier, we treat it as an empty block where all the included transactions will be ignored and do not need to be verified.

One account in different regions Although each node can only access one partition, it is possible that one account is used by people in different partitions, allowing them double-spend some or all of the balance of an account in multiple branches. We limit the maximum balance an account can spend in one branch according to the branch size. For example, if a branch owns one-third of the hashing power, then the maximum balance of an account in the branch will be one-third of the balance before the partition plus the earnings within that branch. As a result, the wallet is also partitioned so that an account cannot spend more than its total contents across multiple branches.

Inactive consensus miner If a miner in the consensus group of a new block is not responding in the signing procedure, the number of participants will be $w - 1$. Nodes seeing the new block will consider that the network is partitioned and there is another branch with a consensus group of size 1. However, the other branch does not exist and we can never merge that non-exist branch. In that case, part of the wallet will be permanently burned.

As discussed in III-C, the probability of a consensus miner being inactive is very small. An effective way to tolerate this rare case is to set a minimum threshold t of consensus miners to form a branch. For example, if $t = 3$ and there are two miners not responding, we don't consider the network to be partitioned. Note that an actual branch with less than t consensus miners will also be ignored. It helps filter out small regions with too little hash power since they are more vulnerable to be take over by malicious entities.

Maliciously creating a branch without partition If the malicious miner controls more than t of the miners in the sliding window, it is possible to create a branch that is fully controlled by not responding to the validation request of others and mine its block with the signature from controlled miners. We will demonstrate that there is no incentive for the malicious nodes to do that. The malicious node will not do it because no client will be sending the transaction to the malicious branch, so the malicious node earns no transaction fee. If it does not publish the branch to the network, no one will send the transaction to this branch. If the branch is published, since there is no network partition, every client can see another branch with a larger branch size. They still only send the transaction to the other branch since that branch is safer with higher hashing power, and they can spend more balance in that branch. So we conclude that the malicious node has no incentive to create a branch without partition deliberately.

IV. Evaluation

The goal of PeloPartition is to tolerate network partitions and preserve the progress of different branches. In this section we conduct experiments to evaluate how PeloPartition performs with different network events and then compare it to a bitcoin-like blockchain system, with the goal of measuring their respective robustness against network partitions.

A. Simulation Parameters

In this subsection we define and explain default parameters that will be used throughout the evaluation. All experiments are configured with the default parameters unless specified otherwise.

Window Size We simulate PeloPartition running for 120 blocks which is equivalent to running a bitcoin-like blockchain for one day. We consider the window size to be 6, roughly equal to the number of mined blocks after one hour.

Network For our simulation we use 1000 nodes distributed across six regions, in which each of them has a specific mining power derived from [18]. Table I shows the exact percentage of each region's distribution.

Prototype Implementation We implemented our prototype on top of a blockchain simulator called Simblock [19] which allows setting the network connections as well as hash power

Region	Distribution
North America	33.16%
South America	0.9%
Europe	49.98%
Asia	11.77%
Japan	22.4%
Australia	19.5%

TABLE I: Distribution of Nodes in Each Region

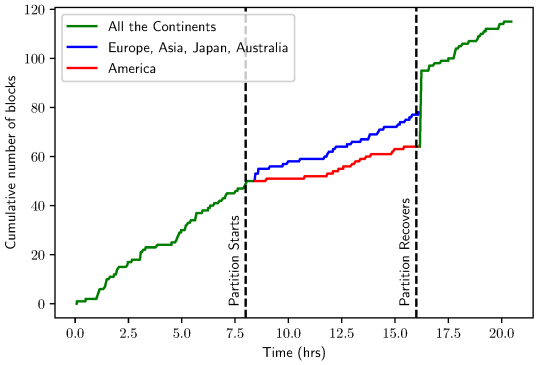


Fig. 2: Size of each branch in blockchain

distribution among different regions. We add a new type of block to allow multiple parents and also add our protocol of signing new blocks. The collective signing procedure is emulated with a fixed latency due to the communication cost. In order to simulate a real world setup we consider this latency to be 15 seconds for each block, equal to the worst-case implementation of the CoSi protocol's performance according to [17]. All algorithms are implemented in Java. Computations are carried out on a computer with an Intel Core $i5-9500T$ processor at $2.2-3.7$ GHz and 8 GB of RAM.

B. Total Number of Produced Blocks

We first show how PeloPartition reacts to the network partition and recovery. In this experiment, we simulate a network partition by isolating North and South America from the other continents by setting the corresponding links' bandwidth to 0. The mining power distribution is set according to section IV-A where America has one-third of the hash power and the remaining world has two-thirds.

Figure 2 reveals that the experiment starts with a regular network until after 8 hours, the network partition occurs and creates two different branches in the blockchain. After 8 hours from partitioning, the network recovers, and only one branch remains. Both of the previous branches will be considered in the final chain and their blocks won't be discarded. This experiment shows that, after partitioning, one branch grows with roughly one-third of the original rate while the other increases with two-thirds of the initial rate. When the network recovers the number of blocks is the accumulated number of blocks generated in both branches plus the blocks mined before partitioning.

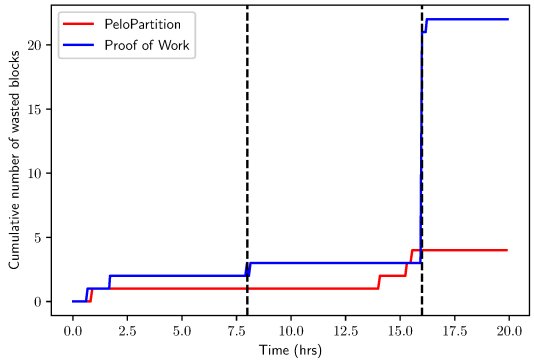


Fig. 3: Number of wasted blocks, comparing Proof of Work with our method in the case of paritiioning

C. Wasted Blocks

In this experiment, we compare PeloPartition with a traditional bitcoin-like blockchain systems on the number of wasted blocks, which means the blocks that are mined but not included in the main chain eventually. We calculate the number of wasted blocks by reducing the number of accepted blocks in the blockchain from all generated blocks during the simulation.

Figure 3 shows that, before the network recovers, both our system and bitcoin incurred a small number of wasted blocks. They include the blocks discarded because of minor forks in Bitcoin and the blocks mined with the same position as a previously accepted block in PeloPartition. This figure denotes that after recovery, one of the partition branches will be discarded entirely in the regular Bitcoin and considered the wasted blocks since. This rapid increase is because the normal Bitcoin only takes the longest chain into account. However, PeloPartition merges two partition branches, and in the end, has fewer wasted blocks than a traditional bitcoin-like blockchain.

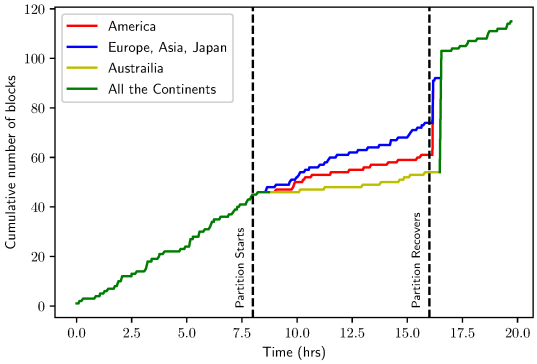
D. Network with Three Partitions

In this experiment, we change the network configuration to reflect the event of three partitions consisting of America, Australia, and the rest of the world.

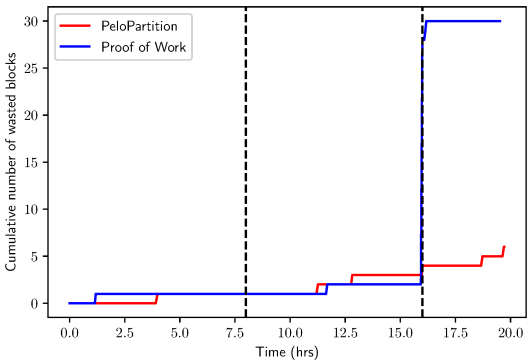
Figure 4a indicates that the experiment starts with a regular network until the network partition occurs and creates three different branches in the blockchain. After recovery, all the branches are eventually merged and are considered in the final chain. This figure shows that the growth rate of the number of blocks in each partition is the same as its corresponding mining power.

In figure 4b we compare the number of wasted blocks for PeloPartition with traditional bitcoin-like blockchain systems when there are three partitions.

This experiment shows that before partitions recover a few blocks are discarded in both PeloPartition and bitcoin. However, after partition recovery the number of wasted blocks for Bitcoin proliferates. This rapid increase is because the



(a) Size of each branch in blockchain



(b) Number of wasted blocks

Fig. 4: The effect of having three isolated partitions in the network

normal Bitcoin only considers the longest branch as the main chain. PeloPartition merges the three branches in partitions, and the final chain includes all the partitions mined blocks. Therefore, there is no jump in the number of wasted blocks. This concludes that PeloPartition has a higher throughput in the case of partitioning and will mine more blocks in the same amount of time.

E. Hash Power and Signing Latency

In this experiment, we analyze the impact of different hash power distributions on the ledger integrity. We also evaluate PeloPartition with different emulated signing latency, which is the average time for new blocks to be signed. We set up a network with two regions and run the simulation with the same partition starting/ending time as in previous experiments. The reported number is the total number of wasted blocks after the network recovers.

Figure 5 shows that Bitcoin loses more blocks when the hash power is equally distributed because the shorter branch will be dropped and it mines more blocks with the higher hash power. For PeloPartition, the number of wasted blocks is independent of the distribution as both branches will be merged into the main chain. With longer signing latency, there is more time to mine blocks at the same position as a block

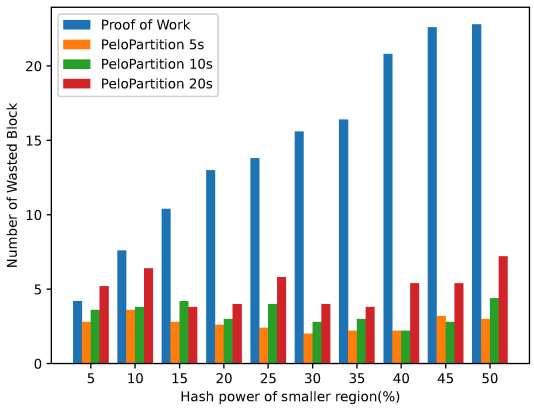


Fig. 5: Number of wasted blocks under different distribution of hash power

under the signing procedure, so there will be more blocks being rejected. Signing latency is determined by the size of the consensus group in real-world implementation, which means that more blocks can be wasted for higher safety guarantee.

V. Related Works

A global-scale network partition can cause severe damage on blockchain [20]. To the best of our knowledge, there is currently no practical solution to overcome the problems of network partitioning. However, blockchain systems that support multiple branches would be a feasible approach since they allow blocks on different branches to be persevered rather than dropping those not on the main chain. There are two types of such systems, sharding-based blockchain, and DAG-based blockchain.

Sharding-based blockchain Sharding blockchain [7], [13], [21] divides all miner nodes into multiple committees, with each committee responsible for verifying transactions and publishing blocks in a single shard. Theoretically, this approach leads to linear scaling of performance with the number of shards. Also, a miner only needs to store part of the ledger that correspond to its shard. Luu et al. [21] propose Elastico, where the time is divided into epochs. In each epoch every miner needs to perform a POW to establish its identity and figure out the shard it can work on. After forming committees, each shard will run a Byzantine fault tolerant (BFT) [13] protocol to reach consensus among the committee and sign each new block. Rapidchain [7] proposes an efficient gossip and intra-shard protocol for faster committee consensus, with a more secure shard assignment generation to prevent the adversary from taking control of one shard. Among all the available implementations of sharding-based blockchain, their primary goal is to achieve better scalability, which depends critically on a reliable network connection for both intra-shard consensus and inter-shard transaction verification.

DAG-based blockchain Directed Acyclic Graph (DAG) is a directed graph that contains no cycles. Applying DAG to the blockchain means that blocks are organized as a DAG

where a block can have more than one parent block, rather than a linear chain of blocks where each block has only one parent block. GHOST protocol [22] is the first to introduce DAG into the blockchain, but the system still recognizes only one main chain, where the number of blocks referencing a given older block will be used for computing the weight of a chain to determine which is the main chain. Later DAGCoin [23] proposed a blockless solution, where transactions are not grouped into blocks but each become the vertex of DAG directly. The "confidence" of a transaction is determined by the number of transactions referencing it and no POW mining is required. DAG-based blockchains increase the throughput by allowing parallel transaction appending and reducing the computing power required. However, DAG-based blockchain is known to be more vulnerable to double-spending attacks than traditional blockchain. It requires all nodes to maintain the global state of the system for correct transaction validation, which is impossible during a hard network partition.

VI. Limitations and Future Work

This section briefly describes several limitations of our work and possible future work areas.

Soft Network Partition In this paper, we are only considering hard network partitions that limits communication between partitioned nodes completely. If the network is in a soft partition, which means the connections between different regions are limited to a low bandwidth but not fully disconnected, more complex malicious behaviors can happen. For example, a consensus miner may receive blocks from different partitions and reply to all of them, which increases the total number of consensus miners thus affecting the wallet partitioning. We leave the analysis of all possible attacks under a soft partition and the corresponding defense mechanisms for future works.

Single-Shard Takeover Attack We use POW and BFT consensus in each shard, which requires no additional safegaurds beyond the assumption that malicious miners have less than 33% of the total hashing power, so how nodes are distributed into shards will be very important. However, as discussed in the system model section, network partitions may be natural events where the system has no control over which nodes will be in which region of the partition. More importantly, many miners are aggregated into mining pools, and it is possible that in one region there is a powerful mining pool that controls more than 33% of hashing power.

One possible solution is finding alternative consensus protocols that can tolerate more faulty nodes to improve the system's robustness. We only create shards when partitioning happens and quickly merge them after the network recovers, so the impact of the takeover attack in a single shard is limited.

VII. Conclusion

We present PeloPartition, a blockchain system that tolerates network partition by sharding using collective signature as the hint of partition and preserving works in each shard after recovery with merging. Four possible attacks under the hard partition and the solutions are discussed to ensure the ledger's integrity. We developed a prototype on Simblock and validated its workflow under network partition and network recovery events. Experiments show that our system retains the work in different shards after recovery, while standard blockchain systems like bitcoin lose most of the blocks produced during a partition.

VIII. Acknowledgement

This research is supported in part by the NSF under grant CNS-1815212.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[2] F. Nawab and M. Sadoghi, "Blockplane: A global-scale byzantizing middleware," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 124–135.

[3] D. Abadi, O. Arden, F. Nawab, and M. Shadmon, "Anylog: a grand unification of the internet of things," in *Conference on Innovative Data Systems Research (CIDR '20)*, 2020.

[4] W. R. Team *et al.*, "Blockchain market shares, market strategies, and market forecasts, 2018 to 2024. wintergreen research, inc," 2017.

[5] S. A. Jyothi, "Solar superstorms: planning for an internet apocalypse," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 692–704.

[6] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.

[7] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 931–948.

[8] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 375–392.

[9] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang, "A stealthier partitioning attack against bitcoin peer-to-peer network," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 894–909.

[10] R. C. Merkle, *Secrecy, authentication, and public key systems.* Stanford university, 1979.

[11] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.

[12] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 203–226.

[13] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[14] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[15] E. Syta *et al.*, "Keeping authorities" honest or bust" with decentralized witness cosigning," in *2016 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2016, pp. 526–545.

[16] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[17] E. K. Kogias *et al.*, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th usenix security symposium (usenix security 16)*, 2016, pp. 279–296.

[18] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," *et al*, 2015.

[19] Y. Aoki *et al.*, "Simblock: A blockchain network simulator," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 325–329.

[20] K. Bruhwiler *et al.*, "Analyzing soft and hard partitions of global-scale blockchain systems," in *International Symposium on Recent Advances of Blockchain Evolution (BlockchainEvo 2022)*. IEEE, 2022.

[21] L. Luu *et al.*, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30.

[22] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[23] S. D. Lerner, "Dagcoin: a cryptocurrency without blocks," 2015.