Colosseum: A Scalable Permissioned Blockchain over Structured Network

Himanshu Gupta
Department of CSE
Indian Institute of Technology Madras
himanshg@cse.iitm.ac.in

Dharanipragada Janakiram
Department of CSE
Indian Institute of Technology Madras
djram@iitm.ac.in

ABSTRACT

Permissioned blockchain protocols generally use voting based algorithms to reach consensus. Scaling such protocols is hard because of their high message complexity and thus, have a limited number of applications. Use cases like banking and healthcare blockchain require a scalable distributed protocol to maintain the consistency of the ledger and ensure its safety.

This paper presents Colosseum, a scalable consensus protocol designed for permissioned blockchain systems. It is a knockout tournament based protocol over a structured ring network to reach consensus on the next set of block proposers. Participants compete in the rounds of a novel two-player game in each tournament to become eligible to propose block. Colosseum introduces Proof-of-Win as the certificate for the result of matches to have an efficient and easy way to propose trusted blocks and verify them.

CCS CONCEPTS

• **Computer systems organization** → *Peer-to-peer architectures*; • **Theory of computation** → Cryptographic protocols.

KEYWORDS

Proof-of-Win, permissioned blockchain, knockout tournaments, consensus

ACM Reference Format:
Himanshu Gupta and Dharanipragada Janakiram. 2019. Colosseum: A Scalable Permissioned Blockchain over Structured Network. In *Third ACM Workshop on Blockchains, Cryptocurrencies and Contracts (BCC ’19), July 8, 2019, Auckland, New Zealand*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3327959.3329539

1 INTRODUCTION

Distributed ledger technologies have gained much prominence in the last decade. It started when the white paper of Bitcoin [6] surfaced in 2008, written by a pseudonym named *Satoshi Nakamoto*. It introduced a decentralized protocol to avoid trusted third parties to regulate transactions by using a distributed ledger known as blockchain. Nodes in a mutually untrusting peer-to-peer network collectively work for the progress, consistency, and security of the blockchain, providing a “trusted” service to reach agreements.

The notion of membership varies for different blockchain systems. In a *permissionless* or *public* blockchain anyone can join the network to participate. Thus, permissionless blockchains use consensus protocols based on proof-of-work (PoW) or proof-of-stake (PoS) [2] to prevent the system against *Sybils* [4]. In contrast, *permissioned* blockchains have a well-defined notion of membership. All the participating users are known, have a shared incentive towards the progress of the ledger, and may not trust others completely for their assurance against a business transaction that includes a transfer of information, funds or goods and services. Regulating membership allow permissioned blockchains to use the traditional voting based Crash fault tolerant (CFT) consensus algorithms (e.g., [5, 7]) or Byzantine fault tolerant (BFT) consensus algorithms (e.g., [3]) to achieve state-machine replication [10] among the peers.

Voting based CFT or BFT consensus is a costly operation. Scalability becomes an issue for such protocols as they exchange $O(N^2)$ (N is the number of validators) messages to agree on a set of valid transactions. Currently existing permissioned blockchains like Ripple [9], Tendermint [1], and Hyperledger [11] use voting, limiting the number of validator nodes that can participate. It affects the decentralized nature of the blockchain systems and reduces the attack surface for an adversary trying to manipulate the ledger. As the immutability of blockchain is a result of its excessive replication over a large peer-to-peer network, a smaller blockchain network is prone to get compromised by a comparably strong adversary where only a few nodes are required to control the system.

In this paper, we describe Colosseum, a consensus algorithm for scaling permissioned blockchains. At the core, it uses synchronous knockout tournaments with asynchronous rounds over a Distributed Hash Table (DHT) to agree on the next set of block proposers. Participants play matches of a two player game to eliminate others and qualify to propose a block. Independent matches between pair of nodes decrease the interaction between members of the network and filter out nodes in comparatively lesser number of messages. The winners then propose blocks for the blockchain providing a unique and verifiable *Proof-of-Win* (PoWin) certificate generated for each match ensuring the safety of the protocol.

2 RELATED WORK

Traditional consensus protocols are well known to manage a state machine replication over a set of known participants in a synchronous setting, such as PBFT [3]. Publicity of permissioned blockchain in many industries brought attention towards BFT protocols to achieve consensus because of their high-performance statistics on replicated servers. One downside of using the traditional BFT protocols to maintain the consistency of blockchain is their scalability with respect to the number of participants in the system. The number of messages exchanged to reach consensus on a set of transactions is the bottleneck in most of the permissioned blockchains and thus, limits their possible applications.

Colosseum, on the other hand, uses knockout tournaments to select block proposers. Nodes interact with opponents and random nodes on the network to decide the qualifying nodes and generate verifiable proofs to check the authenticity of their proposed blocks. It brings down the interaction between nodes to commit one set of transactions to the ledger and therefore can easily scale.

3 ARCHITECTURE

Tournaments in Colosseum are the basis for the selection of block proposers. Users participate in these tournaments to qualify and propose blocks for the ledger. A tournament consists of multiple asynchronous rounds. Users play matches of a two-player game for each round to win and move forward, and receive *PoWin* as the certificate of their win. Nodes with valid PoWin advance to the next round and the winners of $\alpha \leq \log(N)$ (N is the number of participants) number of rounds are eligible to propose a block in Colosseum.

The two-player game plays a vital role in maintaining the integrity of the system. It is an algorithm to select a random winner amongst two participants and rely on random nodes in the network for validation and verification. The randomness of the game comes from the cryptography involved to generate the game proposals and Ids of the third party nodes on the ring. The dependency on random nodes in the network prevents a player to defraud others and ensures fairness among the players.

3.1 Building-blocks of the tournament

Tournaments require two major structural components in order to work - first, an efficient and independent way to find the validating and verifying nodes for the matches. Second, a construct to store the result of matches on the network that is easily verifiable. Colosseum uses a structured ring based *Distributed Hash Table* (DHT) to counter the first issue and introduces *Proof-of-Win* (PoWin) to address the second.

DHT as the Structure network. Colosseum arranges all its users in a structured ring network where each position on the ring corresponds to a hash. They are assigned unique Id's by the trusted oracle which maps them to an exclusive position on the ring. These users form a DHT, i.e., given a key (hash of any of the positions on the ring) one can unambiguously associate a node with it at any given point in time. Colosseum uses a simpler form of DHT where all the participants have the information about every other participant in the network and send direct messages instead of routing them over the network. It is a reasonable assumption for a permissioned blockchain network and does not have a large storage overhead.

Proof-of-Win. *PoWin* is a cryptographically secured certificate designed to maintain the state of users on a peer-to-peer network. It is tamper-proof as it contains the digitally signed game proposals of both the nodes involved in a match, as shown in Figure 1. PoWin is a single certificate generated as the result of matches in Colosseum

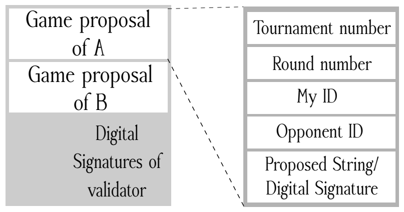

Figure 1: Proof-of-Win for a match

and gets distributed over the network. Winners of a round use their PoWin as a ticket to play the next round, whereas it works as a restriction for the losers and prevents them from moving ahead.

PoWin also proves the occurrence of a match. A node on receiving conflicting PoWins of a user alert others with verifiable proofs (multiple PoWins for the same round or PoWin for a higher round of a player already lost in previous rounds). Since the cryptography involved does not allow to generate fake PoWin certificates, it becomes evident that the user itself has performed such a task. Peers discard blocks proposed by these nodes and individually block them for a certain number of tournaments on multiple violations. Moreover, nodes are blocked on the basis of proofs available with the peers and therefore does not require a consensus between nodes.

3.2 Two Player Game

Colosseum uses a two-player game to eliminate nodes participating in the tournaments and select the final set of block proposers. Each node in the network performs three roles during a tournament. A *Player* forms a pair with an opponent and generate their game proposals. A *Validator* validates the game proposal of both the involved players and generate PoWin certificate as a result, whereas a *Keeper* is responsible for storing the PoWin and verifying that the players are playing authorized matches.

Finding Competitors: Pairing for the tournaments is not absolute in Colosseum. Participants find their opponents on the fly to avoid targeted attacks by malicious opponents. They send TCP requests to other nodes on the ring to play and provide the PoWin of the previous round as a token of their eligibility. Eligible nodes respond with their PoWin while others send a negative response.

Generating game proposals: After pairing confirmation nodes generate their *Game Proposal*. Both the players digitally sign a common string $S_{AB_{tr}}$ with their private key p_k to generate *proposed string* (*PS*), refer equation (2). Validator compares *PS* of both the opponents with its signed $S_{AB_{tr}}$ and the player with the value closer to that of the validator wins.

$$S_{AB_{tr}} = concat(I_A, I_B, t, r) \tag{1}$$

$$PS = sign_{p_k}(\, S_{AB_{tr}}) \tag{2}$$

Here, I_A and I_B are the Ids of the players (I_A lesser than I_B), and t and r is the tournament and round number respectively. Digital signatures used to generate the proposals must be deterministic to ensure safety of the protocol. Moreover, the Public Key infrastructure should not be compromised, i.e., the players should not have multiple private keys to generate various game proposals which

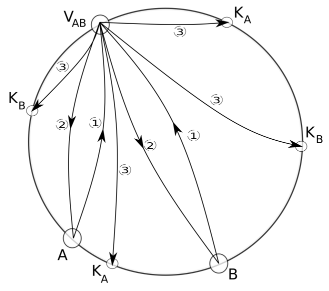

Figure 2: Message flow of a match in Colosseum. A,B are the players, V_{AB} is the validator, and K_A, K_B are the keepers

can result in a probabilistic advantage. It also prevents the validators from using different keys that can affect the outcome of a match.

Game Validation: Colosseum delegates the validation and result computation of matches to a random third-party on the network known as *validator*. Validator is a randomly selected participant on the ring network uniquely identifiable for a match. Its responsibility is to validate the game proposals sent by the players involved, produce PoWin and forward it to the contestants and their keepers, as shown in Figure 2.

Delegating the validation process prevents the competitors from leaving matches in-between, foreseeing the result. Selecting a random node to validate ensures that the game is fair unless the validator is biased. Moreover, the outcome of a match is verifiable; therefore a malicious validator can at most drop a match if an ally is losing. In such a case, nodes not receiving results from the validator find other eligible contenders to play after a particular time.

$$V_{AB} = hash(S_{AB_{tr}}) \tag{3}$$

Validator Id V_{AB} depends on $S_{AB_{tr}}$, as shown in equation (3). Dynamic pairing at run time influence $S_{AB_{tr}}$ and therefore prevent targeted attacks on the validator. Using a common string $S_{AB_{tr}}$ also restricts a player to select a validator of choice and hence provides a fair chance to both the players.

Storing Result: Storing PoWin with the *keeper* is also an important design decision to maintain the virtue of the protocol. Keeper manages the verifiable and tamper-proof state of users over a DHT. It restricts the participants to play unauthorized matches (already lost or not eligible) as the verification process which is independent of the players will identify any wrongful attempt by the players and notify it to others with verifiable proofs.

$$K_A = hash(concat(I_A, t, r)) \tag{4}$$

DHT root for the hash K_A is considered as the keeper of a match, as shown in equation (4). To allow independent state verification (i.e., verifying the state of a player irrespective of its competitor) K_A only depends on the Id of the player and the match information. Also, the keepers are different for every match because the same keeper can become a liability if malicious or can get attacked by some adversary. Furthermore, a malicious or faulty keeper can result in loss of data and match verification can get affected. Thus, a PoWin is stored at multiple keepers evenly distributed over the ring to handle faults. Nodes can query any number of replicas to verify the state of a node for a match.

3.3 State Verification in Tournament Tree

Colosseum verifies all the matches of a tournament to make sure that only winning participants of a round proceed towards higher rounds. Its goal is to double-check the entire tournament tree in a top-down approach, and helps to ensure that nodes play honestly and identify any unauthorized match played in a tournament.

To verify matches, keepers (all replicas) of a match query one or more previous round keepers (multiple replicas) of the players included in a PoWin. It is important because both the nodes playing r_{th} round must have won $(r-1)_{th}$ round. The querying keeper provides the PoWin for round r as a proof for a valid query (i.e., node has played the r_{th} round), and the queried keeper reply with the PoWin of round $r-1$ to confirm the previous victory of the players. In case, the proof is not available with the queried keeper and query is valid, it broadcasts a negative vote against the match and queries the previous keeper using the same proof. A match is considered as a *foul* when a majority ($> 2/3rd$ of keepers) of negative votes from valid keepers are against it.

4 CONCLUSION AND FUTURE WORK

In this paper, we discussed a novel approach towards reaching consensus in blockchain systems. Using pairwise elimination to select the proposers with verifiable proofs reduces the interaction between the peers to agree on the proposed blocks and allows the protocol to scale. We successfully implemented and tested a prototype of Colosseum on top of Vishwa [8] a grid computing middleware having network arrangement same as required by Colosseum.

Colosseum proposes multiple blocks in each tournament, and these can have conflicts. None of the known ledgers possess the ability to handle various simultaneous blocks which is essential to utilize the potential of Colosseum fully. Moreover, a rigorous analysis of the scalability and safety of Colosseum against malicious behavior is required to adopt it as a private blockchain system.

REFERENCES

[1] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph.D. Dissertation.
[2] Vitalik Buterin. 2014. What Proof of Stake Is And Why It Matters. (2014). Retrieved 2018-11-05 from https://blog.ethereum.org/2014/07/05/stake/
[3] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.
[4] John R Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer, 251–260.
[5] Leslie Lamport. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.
[6] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008). https://bitcoin.org/bitcoin.pdf
[7] Diego Ongaro and John K Ousterhout. 2014. In search of an understandable consensus algorithm.. In *USENIX Annual Technical Conference*. 305–319.
[8] M Venkateswara Reddy, A Vijay Srinivas, Tarun Gopinath, and D Janakiram. 2006. Vishwa: A reconfigurable P2P middleware for Grid Computations. In *Parallel Processing, 2006. ICPP 2006. International Conference on.* IEEE, 381–390.
[9] Ripple.com. 2018. Ripple. (2018). Retrieved 2019-1-28 from https://ripple.com/
[10] Fred B Schneider. 1990. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)* 22, 4 (1990), 299–319.
[11] Joao Sousa, Alysson Bessani, and Marko Vukolic. 2018. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 51–58.