Building Blockchains for Tomorrow

Independent Study Report - Monsoon 2020

Sanidhay Arora Sujit Gujar

Abstract. Blockchain faces some major challenges in scalability and fairness in its protocols. This report includes an analysis of current protocols and studies focused on these issues. It aims to build and propose new protocols, by utilizing these studies and insights from Game Theory, to overcome such challenges faced in blockchain technology. The papers summarized in this report include four sections. The first section discusses the game-theoretic analysis of consensus and shard based blockchains. The second part discusses the applicability analysis for blockchains considering reward-structure and off-chain payments. The third part comprises new and existing blockchain consensus protocols like PeerCensus, EOS, and DFINITY. Lastly, we propose our novel blockchain consensus protocol, namely ASHWAChain, that provides high scalability and strong-consistency.

Contents

1

Introduction

Blockchain has revolutionized the way we store data. Especially, it offers a decentralized, distributed, append-only, and trustworthy database. It leads to the building of a plethora of interesting applications, called *distributed applications* (dApps). However, the big challenges with most of the existing blockchains are (i) performance issues in terms of transactions per second (tps) and (ii) consistency (time to finality). In this report, first, we present the recent research/white papers addressing these issues [1] and later propose our blockchain approach, *ASHWAChain*.

Game-theoretic Blockchain Analysis

2

Rational vs Byzantine Players in Consensus-based Blockchains

Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni in [Read7], published in AAMAS 2020, provides a game-theoretic analysis considering Rational and Byzantine participants in consensus-based blockchains.

2.1 Problem Addressed

Byzantine consensus-based blockchains have the advantage to guarantee strong consistency and offer an alternative that has the advantage of being economically viable. To understand the performance and limits of Byzantine consensus-based blockchains with rational players, provide a game-theoretical approach. It provides the analysis of interactions between Rational and Byzantine members in Consensus-based blockchains, and to derive conditions under which the consensus properties in equilibrium are satisfied or not.

2.2 Related Work

Most of the proposals before, assume participants as either honest (satisfying the specifications) or Byzantine, failing to explore the effects of rational participants thoroughly.

[10] is the first to consider rational participants by proposing an incentive-compatible BFT protocol, which introduces incentive mechanisms; however, the work does not provide a game-theoretic framework for their analysis.

[4] proposes a rational analysis of Bitcoin based on the rational protocol design framework [11]. The game is at an upper level of abstraction w.r.t this paper, proposing a two-player game between the protocol designer and the adversary.

[2] models Bitcoin as a coordination game with only rational players, whereas, in this paper, Byzantine players are also a part of a coordination game along with rational players.

2.3 System Model

The system is modeled as a committee coordination game consisting of a total of n, only Rational and Byzantine, participants. It is composed of a finite and ordered set Π, called committee, of synchronous sequential processes or players, $\Pi = \{p_1, ..., p_n\}$ where process p_i is said to have index i. The consensus is said to be reached if more than v players agree on a value.
Consensus-based blockchains should satisfy the following consensus properties:
Termination: every non-Byzantine process decides on a value (a block);
Agreement: if there is a non-Byzantine participant that decides a value B, then all the non-Byzantine participants decide B;
Validity: a decided value by any non-Byzantine process is valid, it satisfies the predefined predicate.

2.3.1 Communication

Each process evolves in rounds each with 2 phases: *Propose* and *Vote* Phase. Further, each phase is divided into three sequential steps, in order: *send*, *delivery* and *compute* step. Each phase has a fixed duration, and it is assumed that the send and compute steps are atomically executed at the beginning and end of each phase, respectively.

[1] Note that, the summaries presented are our understanding of the papers; if any errors found, we are open to revisiting.

2.3.2 Protocol

Steps explained for any round t, at any given height h for any process i:
PROPOSE; Send: If the process i is the proposer of the current block at height h and round t, it creates a block and broadcasts it. A rational process, however, considers if it should create a valid block or not.
PROPOSE; Delivery: The collection of the proposed block by the process.
PROPOSE; Compute: This process checks the validity of the block and stores its *vote* accordingly. A rational process, since there is a cost for checking and voting for blocks, considers if it should check the validity of the block or not. It also considers if it should store the *vote* based on all the previous validity checks for the rounds at the current height h.
VOTE; Send: The process broadcasts its vote stored in the previous phase to the entire network.
VOTE; Delivery: The collection of all the votes broadcasted in the network.
VOTE; Compute: If the number of Votes for a block satisfies the threshold v, and if a block is yet not decided, the protocol terminates on this round, and this block is accepted.

2.4 Equilibria

Term	Meaning
f	The number of Byzantine processes
$n-f$	The number of Rational processes
T	Round where block is accepted
R	Reward gained by any player if it voted for the current accepted block
ψ	Expected number of times a player expects to check validity
ϕ	Expected number of times a player sends expects to send messages before a block is accepted
c_{check}	The cost of checking validity
c_{send}	The cost of sending the vote

Here ψ and ϕ functions satisfy the following property: $g(t) = 1 + \frac{f-t+1}{n-t+1} g(t+1)$, where g is a function.

- In 2 condition settings, A) $f \geq v$ and $n - f \geq v + 1$ and B) $f < v$ and $v < n - f - 1$, there exists a PBE in which rational players' strategy is as follows; 1) propose a valid block 2) send a vote without checking the validity. In this equilibrium *Validity* is not satisfied.
- When $f < v$ and $n - f \geq v$, there exists a Nash equilibrium in which rational players never check blocks' validity nor send messages, so that no block is ever accepted. Hence, *Termination* is not satisfied.
- When $f < v$ and $n - f > v$, if

$$\kappa > \alpha(t) c_{check} - \beta(t) c_{send}, \forall t < f \qquad (1)$$

$$R \geq \max[\frac{n}{n-f} c_{send}, c_{send} + \frac{n}{n-f} c_{check}], \qquad (2)$$

where (x = n - v + f + 2),

$\alpha(t) = \frac{(n-t+1)\phi(t) - (f-t+1) Pr(i_B \geq x | T \geq t) \phi(t+1)}{(f-t+1) Pr(i_B < x | T \geq t)}$, and

$\beta(t) = \frac{Pr(i_B \geq x | T \geq t)}{Pr(i_B < x | T \geq t)}$, there exists a PBE in which certain rational players understand themselves as pivotal and both *Validity* and *Termination* is satisfied. Termination is reached no after than round f + 1.

2.5 Assumptions

- The objective of Byzantine processes is to prevent the protocol from achieving its goal, and to reduce the rational processes utility, no matter the cost. Further, all Byzantine players are symmetric, i.e., their behavior is perceived identically by all non-Byzantine processes.
- There is a cost for checking the validity of a block and sending the vote for a block. Also, all rational players incur a cost κ if any invalid block is accepted.

2.6 Novel Ideas and Takeaways

- Byzantine Consensus-based blockchains could be modeled as a committee coordination game between Rational and Byzantine players. This dynamic game with asymmetric information is analyzed as a Perfect Bayesian Equilibrium for equilibrium.
- If the cost and reward of an invalid and valid block being accepted respectively are large enough, under certain conditions, there exists a PBE in which *Termination*, *Agreement* and *Validity* are satisfied. *Termination*; every non-Byzantine process decides on a value (a block). *Agreement*; if there is a non-Byzantine participant that decides a value B, then all the non-Byzantine participants decide B. *Validity*; a decided value by any non-Byzantine process is valid, it satisfies the predefined predicate.
- In different game settings based on the number of Total players (n), Minimum players required for agreement (v), and Byzantine players (f), there exist equilibria in which *Validity* and *Termination* were not satisfied. In case where A) $f >= v$ and $n-f >= v+1$ and B) $f < v$ and $v < n-f-1$, Validity is not satisfied. In case of $f < v$ and $n - f >= v$, Termination is not satisfied.

3

A Game-Theoretic Analysis of Shard-Based Permissionless Blockchains

3.1 Problem Addressed

Mohammad Hossein Manshaei, Murtuza Jadliwala, Anindya Maiti, and Mahdi Fooladgar in [Read4], publised in IEEE Access 6 (2018) , addresses the research gap of understanding the strategic behavior of rational processors in shard-based consensus protocols for public permissionless blockchains. This analysis is critical for designing appropriate incentives that will foster cooperation within committees and prevent free-riding.

3.2 Novel Ideas and Takeaways

- Addressing the conditions for having a new class of equilibrium, where a subset of processors is forced to cooperate.
- A fair and incentive-compatible reward sharing protocol in shard-based blockchains with only honest but rational players. It introduces a shard-coordinator which imposes derived conditions for cooperative participation. The protocol denies all rewards to any fake cooperating processor, which is a novel punishment approach.

3.3 Assumptions

- Processors do not collude or coordinate to jointly maximize their combined utility and all processors are similar to each other in terms of computational capabilities.
- $l_j \geq \tau$, i.e. number of cooperating processors in shard j is more than the required threshold for consensus and each shard j will submit a non-empty y^j (vector of transactions submitted by shard j) set to the blockchain.

3.4 System Model

The system is based on a classical shard-based consensus protocol, Elastico [12], with only honest but rational players. Time is divided into fixed-sized epochs. The network accepts transactions in blocks, and at the end of each epoch, the network accepts and commits a new block of transactions.
The protocol proceeds in 2 main phases in each epoch:

1. **Organisation Phase;** Each of the total N processors generates a publicly verifiable identity by solving a Proof-of-Work puzzle to participate in the network and form a fully connected overlay for each committee.
2. **Participation phase;** The processors run a standard Byzantine agreement protocol like PBFT in each shard j and reach consensus on a set of transactions B_j, to be included in the block B.

Processor Costs; The total computational cost for a processor P_i is $c_i^t = c^m + c_i^o$, the sum of a mandatory cost (of PoW based on difficulty) to join the committee and the optional cost of participation respectively. Here $c_i^o = c^f + |x_i^j|c^v$, the sum of a fixed cost (of intra-committee consensus) in the participation phase and total cost of verification of transactions, where x_i^j is vector of transactions received by P_i, and c^v is cost of verification of a transaction.

3.4.1 Game Model

The game is (P, S, U), set of players, strategies (cooperate C or deviate D), and payoffs. The total transaction reward in each shard is $b_j = r|y^j|$, where y^j is a set of accepted transactions by shard j, and r is some function of the average transaction fee included in and the number of committee members that processed the transaction. The total reward to be distributed is $TR = BR + \sum_j^k b_j$, Block-Reward + transaction fees for all k shards.

Equal Sharing Reward for each player in the network $b_i = TR/N$, and utility for P_i is, $u_i^j(C) = b_i - c_i^t$ and $u_i^j(D) = b_i - c^m$. This game reduces to a Public Goods Game (PGG) and the only Nash Equilibrium profile of (C^L, D^{N-L}) is if $l_j = \tau$, where C^L and D^{N-L} are set of cooperative and defective processors respectively and $L = \sum_j^k l_j$.

Fair Sharing The rewards are shared equally only among all processors in C^L, and utility of P_i is, $u_i^j(C) = \frac{BR}{kl_j} + \frac{r|y^j|}{l_j} - (c^m + c^f + |x_i^j|c^v)$, and $u_i^D = -c^m$.

Theorem 3.1. *(C^L, D^{N-L}) is a Nash Equilibrium profile if 3 conditions are satisfied:*
1) $\forall j, l_j \geq \tau$.
2) If $x_i^j = y^j$, then $|x_i^j| > \theta_c^1 = \frac{c^f - \frac{BR}{kl_j}}{r/l_j - c^v}$.
3) if $x_i^j \neq y^j$, then $|x_i^j < \theta_c^2 = \frac{\frac{BR}{kl_j} + \frac{r|y^j|}{l_j - c^f}}{c^v}$.

3.5 Protocol

It is an incentive-compatible reward sharing protocol that uses the results derived in fair-reward sharing. The protocol uses 2 novel ideas: a) it first announces to processors whether the cooperation would be in their interests, b) a punishment approach that denies all reward to any processor that was supposed to cooperate but deviated. The coordinator in this could be a centralized trusted third

party as well, as it does not receive any sensitive information.
It proceeds in 4 steps:

1. **Initialisation;** Processors solve PoW puzzle to gain an identity and form committees.
2. **Node Selection;** P_i sends a predetermined hash of set of received transactions, $H(x_i^j)$ to the shard coordinator. It determines the maximum processors with common transactions and if that is more than $\tau \forall j$, then it calculates θ_c^1 and θ_c^2 according to theorem and announce the set of processors that has the incentive and will gain rewards for each shard, $C_j^{l_j}$, and if not, then protocol fails for that epoch.
3. **Participation;** If $P_i \in C_j^{l_j}$ and the theorem is satisfied, then it participates as the system model and reaches a consensus on a set of transactions y^j, and if not, then the node defects.
4. **Reward and Punishment;** Verify whether the $P_i \in C^L$ cooperated or not and distribute the rewards according to fair-sharing.

3.5.1 Numerical Analysis

The simulation results were showed that in an equal reward sharing system, all the processors were defective for all parameter changes. The incentive-compatible protocol similar in fashion to the fair reward sharing but a little bit better for each of the different parameters like N, BR and $|y^j|$ in every aspect.

3.6 Limitations

Two major limitations can get addressed in future works:

- **Inter-Shard Communication;** The processors could lose all their utility and incur a loss if any one of the shards fails, so lack of intra-shard communication is necessary to ensure the system works correctly.
- **Byzantine Participants;** The consideration of the malicious behavior of players is very important. It is required to model the game for real-world scenarios, which have not been addressed here.

Blockchainś Applicability Analysis

4

Optimizing Off-Chain Payment Networks in Cryptocurrencies

Yotam Sali and Aviv Zohar in [Read6], published as ArXiv preprint, focuses on optimizing and analyzing off-chain payment networks built on cryptocurrencies. Off-chain transaction channels represent one of the leading techniques to scale the transaction throughput in cryptocurrencies such as Bitcoin.

4.1 Problem Addressed

Exploration of network topology and construction of efficient off-chain transaction channels with optimal maintenance costs for these networks is worth considering to overcome the issues such as scalability and real-time micro-payments with low transaction fees.

4.2 Novel Ideas and Takeaways

- A closed model for symmetric off-chain channels, and efficient algorithms for constructing minimal cost spanning-tree networks.
- A simple hub topology provides a 2-approximation to the minimal maintenance cost showing that spanning trees, in general, are efficient.

4.3 Assumptions

- Channels are reset whenever liquidity has fully shifted to either side of the channel.
- Transaction amounts are all identical (for convenience, all transfers are for 1 unit of money), and that transaction rates are symmetric, i.e. symmetric-demand and payment networks.

4.4 System Model

Creation/Reset of a transaction channel requires a write on the blockchain. The state of a payment channel e between 2 participants A and B is a pair (ω_A, ω_B), allocation of funds by both players (must be non-negative) to create and maintain the channel. $\omega_e = \omega_1 + \omega_2$ is the liquidity of the channel. A transaction of x from A is feasible if and only if $0 \leq x < \omega_A$.

The **Cost** of locking liquidity ω in the channel is $\alpha \cdot \omega$, where α reflects interest rate in economy. The cost to write a record on the blockchain is ϕ. The total cost of maintaining the channel (per unit time) is $\alpha \cdot \omega + \phi \cdot RPS$, where RPS is the rate of blockchain records per second.

Payment Network is defined as a graph $G = (V, E)$, set of transactors and payment channels respectively, and liquidity allocation $\omega : E \to R^+$, where $\omega(e)$ is liquidity of channel e. Total sum of all liquidity is denoted by W. A transaction of x coins via path $(e_1, ..., e_n)$, changes the state of every channel e_i in the path as $(\omega_{i_1} - x, \omega_{i_2} + x)$ and all transactions must be feasible.

A **Routing Policy** π is a function such that $\pi(i, j)$ is a legal path in G between i and j with no cycles. This policy is in reference to a *state-independent symmetric*

routing protocol (current way payment networks are implemented in Bitcoin).

$(\lambda_{i,j})$ denote the **Demand Matrix** for transactions. A participant v_i transfers to v_j an amount from a Poisson distribution $\lambda_{i,j}$. Considered symmetric-demand networks, $\lambda_{i,j} = \lambda_{j,i}$, since such payments often cancel out and represent an optimistic case for transaction channels' lifetimes.

The blockchain cost per second for channel e is $\phi \cdot RPS_e$, where $RPS_e = \lim_{k\to\infty} E[\frac{k}{T_1+\ldots+T_k}]$ and T_i is the channel lifetime (in seconds) after $i-1$ resets. $\mu_e = E[T_i]$, and proved using convergence that $RPS_e = 1/\mu_e$, and so $\sum_{e\in E} RPS_e = \sum_{e\in E} 1/\mu_e$.

From [15], equal allocation in liquidity maximizes expected time for reset of a symmetric channel with $E[T] = \frac{\omega^2}{8\lambda}$, where λ is the Poisson rate of transfer of one coin in one direction.

The total single-directional traffic over an edge e, as derived from the routing policy is defined as **balanced channel Poisson rate**, $\lambda_e = \sum_{\{i,j\} s.t.\ e\in\pi(i,j)} \lambda_{ij}$.

4.5 Optimization

4.5.1 Hub is 2-Approximation

For any off-chain network $G = (V, E)$ and $H = (V, E_H)$ as a hub topology centered at an arbitrary vertex $v_o \in V$, there exists a routing policy π_H and liquidity allocation $\omega_H : E_H \to R^+$ for H, such that for every series of transactions, the number of channel resets is at most twice the number of channel resets in G, and the required liquidity W_H is at most doubled: $\sum_{e'\in E_H} \omega_H(e') \le 2 \cdot \sum_{e\in E} \omega(e)$. The tightness is proved by the following theorem.

Theorem 4.1. *For every $\epsilon > 0$, there exists a set of transaction demands (λ_{ij}), in which the optimal maintenance cost per second is at least $(2-\epsilon)$ times lower than in every hub.*

For any transaction demands, a hub over an arbitrary node is a 2-approximation for the optimal maintenance cost network. This also raises the concern of large monopolistic players and could out-perform decentralized networks.

4.5.2 Maintenance Costs

Liquidity The optimal allocation of liquidity that minimizes network's maintenance costs with λ_e as one-side transaction rate of edge e is: $\omega(e) = W\frac{\lambda^{1/3}}{\sum_{\bar{e}\in E}\lambda_{\bar{e}}^{1/3}}$. The optimal blockchain record fees per second, $\phi \cdot \sum RPS_e$, is: $\frac{8\phi}{W}(\sum_{e\in E}\lambda^{1/3})^3$, which is obtained by using Lagrange multipliers on the chain record fees, $\frac{8\phi}{W^2}\sum_{e\in E}\frac{\lambda_e}{x_e^2}$, where $x_e = \frac{\omega_e}{W}$.

Spanning Trees For a network $G = (V, E)$, and a cut graph $(X, V \setminus X)$, the cut's λ − *capacity* is defined as: $\sum_{i,j\in E s.t.\ i\in X, j\in V\setminus X} \lambda_{i,j}$. The Poisson rate of a channel e in a spanning tree T, λ_e equals the λ − *capacity* of the cut between the 2 connectivity components of $T - e$. Using this notion to optimize the communication traffic, $\sum_{e\in E_T}\lambda_e^{1/3}$ needs to be minimised which is obtained by using results from Gomory-Hu tree [9]. A Gomory-Hu tree T over weight function of Poisson rate over edges, $\lambda : E \to R^+$, is the optimal spanning tree over the value of blockchain record fees per second, and it can be calculated in polynomial time.

Greedy Game The optimal maintenance cost of the network at the optimal liquidity W_{opt}: $M.C(W_{opt}) \propto RPS_0^{1/3}\cdot\phi^{1/3}\cdot\alpha^{2/3}$, where RPS_o is the optimal blockchain records per second when $W = 1$. The result is obtained by minimizing the maintenance cost: $MC(W) = \phi \cdot \frac{RPS_0}{W^2} + \alpha\cdot W$. The minimal maintenance cost for a channel e with λ transaction Poisson rate is: $M.C_e = 3\sqrt[3]{2}\lambda^{1/3}\phi^{1/3}\alpha^{2/3}$.

Considering rational players, each player will minimize their own costs, with restriction to spanning tree topology. The share of maintenance cost of player v over an edge with λ_e balanced transaction Poisson rate is: $M.C_e(v) = \frac{\lambda_e(v)}{\lambda_e}\cdot M.C_e$, where $\lambda_e(v)$ is transfer rate by v on e.

The total maintenance costs of a player v is : $M.C(v) = 3\sqrt[3]{2}\phi^{1/3}\alpha^{2/3}\sum_{e\in E}\frac{\lambda_e(v)}{\lambda_e}\lambda_e^{1/3}$, clearly depend on the routing policy which is unique in a spanning tree. This property restricts the players' legal moves to reconnect a single edge that is incident on its vertex to maintain the spanning-tree topology. An equilibrium is said to be reached when none of the players can reduce their costs by a legal move.

Price of Anarchy There is an unbounded price of anarchy for every $K > 0$ there exists an equilibrium point in the spanning tree greedy game, T^*, in which there holds $M.C(T^*) > K \cdot M.C(T_G)$ where T_G is the optimal maintenance cost spanning tree.

4.5.3 Simulation Results

Simulation with 100 agents, demand matrix in a scale-free graph model: generating a random scale-free graph between the agents with a certain power-law coefficient, and setting a non-zero transaction rate for every pair of connected nodes.

The RPS ratio of a complete graph to optimal spanning tree was high (around 8) that reduced to almost 2. The RPS of Hub to optimal spanning tree started almost equal and increased up to two as the power coefficient is increased. It indicates that hub topology achieves better results than a tight 8-approximation complete graph, with an optimal spanning tree being the best strategy.

4.6 Limitations and Future Work

- The assumption of balanced channels as channels in payment networks might be unbalanced and hence their lifetime may be closer to a linear factor of the liquidity they hold, which will imply blockchain costs will be higher.
- Possibility of different routing algorithms for asymmetric demands matrices remains open.
- The theory of online algorithms for balancing channels persists as balancing algorithms can reduce the channel's lifetime significantly, and the analytical estimation question remains.

5

Redesigning Bitcoin's fee market

Ron Lavi, Or Sattath, and Aviv Zohar in [Read5], publised in WWW '19, proposes two auction-based frameworks to Redesign Bitcoin's Fee Market considering impatient users.

5.1 Problem Addressed

As the block reward decreases over time, the revenue derived from transaction fees starts to have an increasingly important role in incentivizing honest miners. This process is essential for the security of the network. The revenue from transaction fees is not extracted well if the blocks are not congested. This effect has implications on the scalability debate as increasing block size could lower transaction fees. Thus, addressing this issue by proposing novel frameworks for the fee market.

5.2 Novel Ideas and Takeaways

- Using RSOP mechanism based on [1], removes the ability of an individual transaction to affect the price it pays, promoting honest bidding.
- Using auction-based frameworks to propose new models for Bitcoin's fee market, which decouples the issue of miners' revenue from the block-size.

5.3 Assumptions and Limitations

- Assumption of only impatient users, users that have utility only if their transactions get accepted in the next block. This limitation is important to tackle as patient users exist in real-world scenarios. The distribution of bids changes if rejected bids stay in the system, unlike in the current analysis.
- After off-chain transaction channels develop, it will radically change the transaction fee market. Several aspects of the demand for blockchain transactions would be affected: the level of patient users for transactions that set up payment channels, the willingness to pay fees, and the basic-demand for transacting on-chain may differ from the pre-lightning fee market.
- In the RSOP mechanism, blocks may contain many transactions that do not get accepted, hence wasting the network storage.

5.4 The Monopolistic Price Mechanism

This auction-based mechanism for the fee market overcomes the two main issues with the current fee market of (i) bid shading and frequent bid updates, and (ii) poor revenue extraction. It decouples the block-size concern from the miners' fee and provides several key properties like (i) high revenue extraction, (ii) incentivizing auctioneer, and (iii) honest bidding of true preference by users.

Mechanism For n impatient users each with a transaction, with each user's maximum willingness to pay as v_i, given the vector of bids for a transaction fee for each transaction submitted by all users, $b = (b_1, ..., b_n)$, the miner chooses a subset of transactions to be included in the block of size k (number of transactions), and all the users included in the block pay the minimum bid of b_k (supposing that $b_1 \geq b_2 \geq ... \geq b_k$) as the transaction fees.
The monopolistic revenue is defined as $R(b) = max_{k\in[n]} k \cdot b_k$ and monopolistic price as $p^{monopolistic}(b) = b_{k^*(b)}$, where $k^*(b)$ is number of users that maximise $R(b)$. Considering that a user i can report a strategic bid lower than v_i and still gets the transaction accepted, the strategic price is defined as $p^{strategic}(b_{-i}) \equiv min\{b_i \in R \mid p^{monopolistic}(b_i, b_{-i}) \leq b_i)\}$. To analyze the strategic behaviour, notion of discount ratio is introduced to capture the gain a user can obtain over honest bidding, defined as $\delta_i(v_i, b_{-i}) = 1 - \frac{p^{strategic}(b_{-i})}{p^{honest}(v_i, b_{-i})}$ if $v_i \geq p^{strategic}(b_{-i})$ and 0 otherwise, where $p^{honest} \equiv p^{monopolist}$.

To achieve the goal of quantifying how large δ will be, 2 parameters are considered in evaluation and empirical analysis, $\Delta_n^{average} = E_{(v_1,...,v_n)F}[\delta_1(v_1, v_{-1})]$ and $\Delta_n^{max} = E_{(v_1,...,v_n)F}[\delta_{max}(v)]$, where $\delta_{max}(v) = max_i \delta_i(v_i, v_{-i})$ and F is some distribution on $R_{>0}$. It is assumed that values drawn from F are i.i.d. As it clearly holds that $\Delta_n^{max} \geq \Delta_n^{average}$, the following theorem is proved:

Theorem 5.1. *For any distribution F with a finite support size, $lim_{n\to\infty} \Delta_n^{max} = 0$.*

It is also proved here that even in the worst case expectation of the discount ratio, it goes to zero as the number of users increases, which is the desired property in the mechanism.

Multiple Strategic Bids As the users can also benefit from splitting the bids, $p^{strategic}$ is generalised to capture this strategy as $p^{multibid}(b_{-i}) = min\{u \cdot b_i^{(u)} \mid u \in N^+, b_i^{(1)} \geq ..., \geq b_i^{(u)} \in R, b_i^{(u)} \geq p^{monopolistic}(b_i^{(1)}, ..., b_i^{(u)}, b_{-i})\}$. As the user splits the bid in u parts, and from strict inequality of $p^{strategic}(b_{-i}) \geq p^{multibid}(b_{-i})$, this case is considered under the theorem above. Further is it easy to show the equivalence of the definition above and the case where user splits each bid equally, as b_i, and maximises $u \cdot b_i$.

An efficient way to compute $p^{multibid}$, is fix a player i, and let $w = v_{-i}$ where $w_1 \geq w_2 ... \geq w_{n-1}$,

Theorem 5.2. *If* $p^{multibid}(w) = min_{k^*(w) \leq j \leq n-1} \frac{R(w)}{f(j)}(f(j) - j)$, *further if* j^* *minimizes this term, then taking* $b^* = \frac{R(w)}{f(j^*)}$ *and* $u^* = f(j^*) - j^*$ *minimizes the total bid of* $u \cdot b_i$, *where* $f(j) \equiv max\{\lceil \frac{R(w)}{w_j} \rceil, j+1\}$.

Empirical Evaluation The empirical evaluation uses both synthetic data as well as transaction data taken from the Bitcoin blockchain. The synthetic data is generated using 4 distributions: (i) uniform discrete over 1 to 100, (ii) uniform over [0, 1], (iii) half normal, probability of max fee decreases exponentially, and (iv) inverse distribution, $F(x) = 1 - 1/x$ for $x \in [1, \infty]$. The willingness to pay for the 1000 considered bitcoin blocks is taken as a function of transaction size $v(x)$, for 3 alternatives: (i) log(x), (ii) $\sqrt{x}$, and (iii) $v(x) = x$.
In all the cases both $\Delta_n^{average}$ and Δ_n^{max} decreases approaching to 0 as n increases from 2^3 to 2^{17}, except for the case in inverse distribution for Δ_n^{max} which doesn't decrease much with n.
This difference between $\Delta_n^{average}$ and Δ_n^{max} is because of two different reasons: (i) $\Delta_n^{average}$ takes into account also the losing bidders, whose discount ratio is zero, and (ii) Δ_n^{max} takes into account only the single winning bidder with the maximal discount ratio.

5.5 RSOP Mechanism

The main difference between the two mechanisms is that Random Sampling Optimal Price (RSOP) removes (by design) the ability of an individual transaction to affect the price that it pays. To instantiate the RSOP-based mechanism, users must specify their maximal willingness to pay for the transactions. The miners create a block with all transactions they wish to include. Unlike the current Bitcoin protocol, here, not all transactions in a block are valid.
Upon receiving n bids vector b, the auctioneer does the following:
(i) randomly splits transactions in 2 sets, A and B, with equal probability,
(ii) compute $p_A^{monopolistic}$ and $p_B^{monopolistic}$, where $p^{monopolistic}$ for empty set is 0,
(iii) set of winning bids is $A' \cup B'$ where $A' = \{i \in A : b_i \geq p_B^{monopolistic}\}$ and $B' = \{i \in B : b_i \geq p_A^{monopolistic}\}$. The total revenue hence is $RSOP(b) = |A'| \cdot p_B^{monopolistic} + |B'| \cdot p_A^{monopolistic}$. This mechanism provides truthfulness and maximal revenue ($lim_{n \to \infty} max_b \frac{R(b)}{RSOP(b)} = 1$).

Algorithm The algorithm goes as follows:
(i) The node receives a block B and checks the validity of this block and transactions, considering only valid transactions according to the mechanism described below.
(ii) Compute the sets A and B using the block hash as a seed to a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) and determine valid transactions to be as the ones in A' and B'.
(iii) A fraction $1-\alpha$ of the revenue goes to the miner who mined the current block, and the rest goes to the future miner who would mine the next valid block. The parameter $0 \leq \alpha \leq 1$ needs to be specified as part of the protocol.

As the source of randomness is the same, each node will reach a consensus on the set of valid transactions. The empirical analysis shows that the gain-ratio from performing this strategy decreases as the number of users increases. This analysis is similar to the monopolistic mechanism.

Issues The main concerns with this algorithm is the assumption of:
(i) user truthfulness; that the users cannot send multiple bids, further it is also not clear on how to construct even tailored worst-case examples or if there exists an efficient algorithm to find beneficial deviations as users cannot control the their bids' association to set A or B, and
(ii) auctioneer honesty; as it can create false bids and/or remove bids for his benefit. Determining α is crucial as it prevents miners from adding false bids and eliminates potential profit from this strategy as users increase.

Blockchain Consensus Protocols

6

Bitcoin Meets Strong Consistency

C. Decker, J. Seidel, and R. Wattenhofer in [Read2], published in ICDCN '16, provides a new system that can get built on top of Bitcoin or similar blockchains. It enables real-time payments with fast confirmations.

6.1 Problem Addressed

Most Proof-of-* blockchains like Bitcoin, only provide eventual consistency (a questionable property for a protocol to handle financial transactions). A signed transac-

tion, not yet included in a block, can be vulnerable to a double-spend attack. A transaction might require a significant amount of time to get included in a block due to slow block-time and over-flooded transaction-pool.

Further, in the case of blockchain-forks, only "enough" subsequent block confirmations determine the eventually accepted chain. This process requires a significant amount of time for the transaction to get confirmed for consistency. This time delay prevents efficient real-time micro-payments in any general setting.

6.2 Novel Ideas and Takeaways

- Enable the creation of a cryptocurrency that provides *forward security*, i.e., once a transaction is committed it cannot be reverted at any future time, and hence supports fast/real-time confirmations.
- Using the Blockchain to limit and certify new identities joining the system and provide strong guarantees regarding their assignment to entities participating in the system.
- Decoupling block-creation and transaction confirmations by leveraging the techniques from Bitcoin (proof-of-* consensus algorithms) and byzantine agreement protocols, to achieve strong consistency.

6.3 System Model

The system operates in a setting of following three components:
a) A peer-to-peer system: The identity of a peer p is its public key. Every message is verified by the signature of the sender.
b) The notion of controlling entities: Each peer, p, is assigned to exactly one entity, e, who controls them. All entities are rational.
c) The notion of computational resources at the disposal of an entity: To model computational limitations of entities, the notion of a *computational unit-resource* is introduced, each of which possesses the same computational power.

6.4 PeerCensus

The PeerCensus system acts as a trust-less decentralized certification authority, manages peer identities in a peer-to-peer network, and ultimately enhances Bitcoin and similar systems with strong consistency. PeerCensus consists of two components: the Blockchain (BC) and the Chain Agreement (CA).

Blockchain (BC) Peers are either non-voting or voting peers. Each new peers start as non-voting until promoted to voting by appending a block to the collaboratively maintained blockchain. The share of identities controlled by an entity converges to the fraction of computational resources that the entity controls in the network.

Chain Agreement (CA) Responsible for two tasks: (A) To track the system membership by running a byzantine agreement protocol. (B) To resolve conflicts in case of a blockchain-fork, by only committing one block using standard agreement protocol techniques and obtain strong consistency.

6.4.1 Protocol

Besides the two layers of PeerCensus, the protocol consists of a third, Application (APP) layer.

The BC layer works on a Proof-of-Work consensus mechanism. A legal block is of the form $b_i =< h, d, p, x >$, the hash of the previous block, difficulty, a new identity, and the nonce.

The CA layer basically keep track of some shared state that can be modified by certain predetermined *operations* by using SGMP [14] and PBFT [3] agreement protocols. The shared state encompasses an operation log O, a set of online voters I, and blockchain C. The three operations are:
• **block(b)**; used to append a new block b to the Blockchain, thus promoting the peer in b to voting.
• **join(p)**; used by a previously offline voting peer p to re-join the set I of online voters.
• **leave(p)**; used to remove offline peers from I.

The APP layer makes use of the membership information from the CA to implement the application logic. The applications can use the full capabilities of PBFT, including snapshots, by the use of rankings, current membership, and timestamp provided by CA. A single instance of CA and BC can get shared among any number of applications.

6.4.2 Safety

Since SGMP ensures that I tracks the voters in $P(t)$ (online voters), so as long as the inequality $\phi_i := \frac{|I_A|}{|I_D|} < \frac{1}{2}$ is satisfied (for PBFT to be secure), PeerCensus is considered in a secure state, where I_A and I_D are the corresponding partition of I into online peers controlled A (attacker), and D (meta-entity; all entities not A). The analysis relies on a steady state of the system. In a steady-state, the number of online peers, and resources, is governed by the respective expected value. To ensure the secure state, the root theorem needed for the proof is as follows:

Theorem 6.1. *Let ϕ_R denote the fraction of resources associated with A over resources not associated with A, and let $0 < \epsilon < 1/2$ be a constant. If PeerCensus reaches*

a steady-state and $\phi_R < \frac{1}{2} - \epsilon$*, then PeerCensus is in a secure state with high probability.*

PeerCensus can get bootstrapped by retrofitting the Bitcoin blockchain, providing the initial resources, blocks (voting identities), and peers. By considering first 350,000 Bitcoin blocks and 25,000 online peers (from latest of those blocks), the Prob(PeerCensus in secure state) $>= 1 - 4.26 \times 10^{-15}$.

6.4.3 Related Work

PeerCensus solves problems arising from inconsistent state views, such as double-spendings [16, 8]. It does not address problems like transaction malleability [5], scalability [6] and privacy issues, e.g., [7, 13].

6.5 Discoin

Discoin is a cryptocurrency, an exemplary application built on PeerCensus. It tracks the balances of *accounts*, denominated in *coins*. The shared state in Discoin consists of account balances B. Discoin has a single operation transaction(tx) which, if committed, applies the transaction to the account balances. It distributes the reward, of r newly generated coins, in equal parts to each identity $i \in I$. The reward is triggered by a timestamp change each time a block is found and does not necessitate a new transaction.

It explicitly tracks account balances that result in a smaller shared state and a more intuitive concept of account balances and faster confirmations. Committing a transaction is independent of the block generation, and it provides *forward security*.

Once PeerCensus gets bootstrapped, Discoin can get bootstrapped by computing the account address balances in the Bitcoin blockchain up-to height l_m.

7

DFINITY Technology Overview Series Consensus System

Hanke, Timo and Movahedi, Mahnush and Williams, Dominic in [Read3] proposes the consensus protocol of DFINITY, a committee-based blockchain protocol.

7.1 Problem Addressed

DFINITY is a decentralized public compute platform that aims to host the next-generation software and internet services. The Dfinity blockchain computer provides a secure, performant, and flexible consensus mechanism. The goal is for the virtual computer to finalize computations quickly, to provide predictable performance, and for computational and storage capacity to scale up without bounds as demand for its services increases.

7.2 Novel Ideas and Takeaways

- Use of a registration layer using any method like a stake (or PoW) providing permanent Sybil resistant identities.
- The Decentralised Random Beacon provides a verifiable unpredictable pseudo-random output used in creating a random committee for running the consensus mechanism. This randomness makes it highly scalable and decentralized secure.
- Use of notary to achieve fast finality and predictable block times and using t-of-n threshold DKG signature scheme with properties like non-interactivity for fast networking and the uniqueness of group signature.

7.3 Consensus Mechanism

Dfinity peer-to-peer network consists of *clients* connected by a broadcast network over which they can send messages to everyone. Clients observe blocks and build their view of the finalized chain and fulfill three active functions: (a) participate in the *decentralized random beacon*, (b) participate in the decentralized notary, (c) propose blocks.

Dfinity's consensus mechanism has four layers:

1) Identities and Registry: The responsibility of the first layer is to register and provide Sybil-resistant client identities via stake deposit with a lock-up period (any Sybil resistance method, like PoW, can also be used). All registered clients have permanent & pseudonymous identities.

2) Random Beacon: This is the core layer of Dfinity. The random beacon is an unbiasedly *verifiable random function* (VRF), produced jointly by registered clients who produce an unpredictable output for each round.

3) Blockchain: This layer deploys the *probabilistic slot protocol* (PSP) providing constant block time and homogeneous bandwidth utilization by a ranking mechanism driven by the random beacon.

4) Notarization: The fourth layer is a *decentralized notary* that provides time-stamping and publication guarantees and is ultimately responsible for near-instant finality.

7.3.1 System Model and Primitives

Clients or **Replicas** are labeled as $i \in N$ and U be the finite set of labels of all replicas, called the universe. $i \in U$ with public/secret key pair (pk_i, sk_i).

Committee: At any given time, some or all $i \in U$, are arranged into one or more subsets, $G_1, G_2, ... \subseteq U$ called groups. Out of all the groups, a single one is active to drive progress and ensure consensus by proposing & notarising blocks and running the Decentralised Beacon Protocol. This group is called the *committee*. All groups G_j have the same size n, which is a system parameter called the group size.

The system evolves in **rounds**. Replicas advance to the next round based on events. Dfinity assumes a semi-synchronous network, i.e., that the network traversal time can be modeled by a random variable Y whose probability distribution is known. The two constants are responsible for liveness ($BlockTime$) and safety (T) of the system, respectively.

A cryptographically secure **pseudo-random number generator** PRG which turns a seed ξ into a sequence of values PRG(ξ, i) for $i \in N$. The sequence PRG(ξ,i) is used as input to the Fisher-Yates shuffle to produce a **random permutation** of U, a bijective map $\{1, ..., |U|\} \to U$ denoted by $Perm_U(\xi)$.

Blocks A block B, apart from genesis B_0, is a tuple $B = (p, r, z, d, o)$ where $prvB := p \in \{0,1\}^l$ is the hash reference to the previous block, $rdB := r \in N$ is the round number, $ntB := z \in \{0,1\}$ is the notarization of the previous block, $datB := d \in \{0,1\}$ is the data payload ("transactions" and "state"), $ownB := o \in U$ is the creator (or "owner"). A notarization is a signature on the previous block created by a "notary".

Chain $C(B)$ denotes the uniquely defined chain C with $headC = B$. $C \leq C'$ denotes C is prefix of C'. For any non-empty set S of blocks we denote by $C(S)$ the largest common prefix of all chains $C(B)$ with $B \in S$. For any sets of blocks S, T with $S \subseteq T$ we have $C(T) \leq C(S)$. $prvS := \{prvB | B \in S \setminus \{B_0\}\} \neq \phi$, then $C(prvS) \leq C(S)$.

7.3.2 Threat Model

Adversarial strength: For any $G \subseteq U$,let $f(G)$ denote the number of Byzantine replicas in G. *Assumption 1;* There is $\beta > 2$ such that $|U| > \beta f(U)$,
where value $1/\beta$ is called the adversarial strength.

Honest groups: For group size n, G is called honest if $n > 2f(G)$. The protocol relies on *Assumption 2;* Each group G used in the system is honest.

U is honest from Assumption 1. For randomly selected G, $Prob[Ghonest]$ is calculated via the cumulative distribution function of the hyper-geometric probability distribution. For failure probability ρ, we can calculate the probability for the minimal group size $n = n(\beta, \rho, |U|)$ such that $Prob[Ghonest] > 1-\rho$ for each random sample $G \subseteq U$ with $|G| = n$.

7.4 Protocol

7.4.1 Decentralised Random Beacon

The beacon allows replicas to agree on a verifiable random function (VRF) and to jointly produce one new deterministic pseudo-random output ξ_r of the VRF in every round r. ξ_r is unpredictable, given the knowledge of all prior outputs, til at least one honest replica advances to r unbiasedly due to its deterministic pseudo-random nature.

The DRB protocol uses *non-interactive* t-of-n threshold signatures scheme used by group G as the source of randomness to generate keys. The threshold scheme used in the DRB protocol is setup using a distributed key generation (DKG) mechanism, which does not rely on trusted parties. The signature scheme is the pairing-based scheme derived from BLS. G runs a DKG for BLS to set up the group public key and the secret key shares during the initialization of the blockchain system. The threshold t is a parameter of the setup. A group signature can be recovered by any t shares from the group as input because of *uniqueness* property.

7.4.2 Probabilistic Slot Protocol

Based on ξ_r, the protocol assigns a rank to each $i \in U$. The rank of the proposer defines the weight of a block as follows:

The ranking permutation for round r is defined as $\pi_r := Perm_U(\xi_r)$, rank of $i \in U$ as $\pi_r(i)$ and rank of a block B as $rkB := \pi_r(ownB)$. $rkB < rkB'$ means B has higher priority level than B'.

The weight of a block B is defined as $wtB := w(rkB)$ and weight of a chain $C = (B_0, ..., B_r)$ as $wtC := \sum_{h=0}^{r} wtB_h$. In Dfinity $w(x) = 2^{-x}$. The heaviest chain is considered by the replicas to be notarised.

7.4.3 Notarization and Finality

A notarization of block B is an aggregated signature by a majority subset of U on the message B. A notarized block is a block concatenated with notarization of itself. The replica continues to sign all highest priority block proposals as soon as it receives more block proposals after $BlockTime$. It advances to the next round as soon as a notarization for the current round has been observed by the replica. A proposed block B is considered valid for round r if $rdB = r$ and there is a valid block B' such that: (1) $prvB = H(B')$ and $rdB' = rdB-1$, (2) ntB is notarization of B', and (3) $datB$ is valid.

Timely Publication: An artifact of round r was timely published (within d rounds) if it was broadcasted while at least one honest replica was in round $\leq r$ (resp. in round $\leq r + d$).

Only timely published block proposals can get notarized and Only notarizations that are timely published within 1 round can get referenced. A round has **normal operation** if only one block for that round gets notarized. The publication of both block proposals and notarizations is enforced.

An adversary cannot build a private chain because a chain can only survive if: (a) All its blocks were timely published and (b) All its notarizations were timely published within 1 round.

Algorithm: It assumes that the observer receives all-round (r − 1) notarizations that can get referenced before time T after having received the first notarization for round r. The notarization algorithm confirms finality as follows:

1) N_r be the bucket for all notarized blocks for round r out of which only highest ranked blocks concerning the publicly verifiable ranking algorithm driven by the random beacon are notarized.
2) As a block cannot be validated without knowing its predecessor, every pair of blocks that reference each other, the predecessor is processed first.
3) So the receipt of any further notarized blocks for N_r that can get referenced is removed.
4) A round r is said to be finalized because we know that N_r already contains all chain tips that can possibly survive beyond round r. So notarization proceeds in rounds, and hence the output is the longest-common-prefix $C(N_r)$ as being final.

7.4.4 Threshold Relay

Group Derivation: For group size n, groups are derived from a random seed ξ where the j^{th} derived group is $Group(\xi, j) := Perm_U(PRG(\xi, j))(\{1, ..., n\})$. . A number m and a seed ξ is chosen and form groups $G_j \forall j \in \{1, ..., m\}$ which runs the DKG to create keys stored in blocks.
Committee Selection: The sequence (ξ_i) is bootstrapped by defining an initial value for ξ_0. Then, in round r, $G^{(r)} := G_j$, $j := \xi_r \mathrm{mod} m$, as the committee for round r. In the random beacon protocol, the members of $G^{(r)}$ jointly produce the output ξ_r , which is then used to select the next committee $G^{(r+1)}$.

The rounds are split into non-overlapping epochs of length l, which is a fixed system parameter. The block produced in the first round of each epoch is called a registry block, which is also called a keyframe. It contains a summary of all new registrations and de-registrations of replicas that happened during the previous epoch. The "summary" is a deterministic result of all the blocks in the preceding epoch so that the block maker of the keyframe has no opportunity to censor registrations.

8

EOS: Consensus

EOS developers in [Read1] proposes the EOS consensus mechanism, providing high scalability and fast confirmations.

8.1 Problem Addressed

Existing blockchain platforms are held back by large fees and limited computational capacity that prevent widespread blockchain adoption. The EOS.IO software introduces a new blockchain architecture designed to enable vertical and horizontal scaling of decentralized applications. It can ultimately scale to millions of transactions per second, eliminates user fees, and allows for quick and easy deployment and maintenance of decentralized applications in the context of a governed blockchain.

8.2 Consensus Protocol

8.2.1 Overview

An EOSIO blockchain is a highly efficient, deterministic, distributed state machine that can operate in a decentralized fashion. EOS has achieved to provide functional decentralized applications by increasing performance to thousands of transactions per second, reducing latency to 1.5 seconds, eliminating per-transaction fees, and providing a user experience similar to those that are currently made available by existing centralized services.

8.2.2 Model

The nodes responsible for block production and block validation are called *block producers* which are elected by EOSIO stakeholders (users of the network). There are two layers, which are functionally separate, involved in the EOSIO consensus model:

1. The Native Consensus Model (aBFT) This layer is responsible for the actual process of block finality, performed in an asynchronous byzantine fault tolerant (aBFT) way. This layer uses a two-stage block confirmation process by which a two-thirds supermajority of producers from the currently scheduled set confirms each block twice. The consensus model achieves algorithmic finality through the signatures from this chosen set of participants (active producers). They are arranged in a schedule to determine which nodes are authorized to sign the block at a particular time slot. Changes to this schedule are initiated by privileged smart contracts running on the EOSIO blockchain.

2. Delegated Proof of Stake (DPoS) This layer elects the active producers who will be authorized to sign valid blocks based on their stake in the network and responsible for the scheduling. The layer introduces the concepts of tokens, staking, voting, vote decay, vote tallying, producer ranking, and inflation pay. Blocks get produced in rounds of 126 seconds (6 blocks each, times 21 producers). DPoS layer gets enabled by WASM smart contracts. The actual selection of the active producers (the producer schedule) is open for voting every scheduled round.

8.2.3 Voting

A token holder must first stake some tokens to become a stakeholder and thus be able to vote with a given staking power. Each stakeholder can vote for up to 30 block producers in one voting action, out of which the top 21 elected producers will then act as DPoS delegates as block producers. The remaining producers are placed on

a standby list in the order of votes obtained.
The **voting weight** of each stakeholder is directly proportional tokens staked and the base-2 exponentially proportional to the time elapsed since the EOSIO block timestamp epoch, defined as January 1, 2000. As increasing the voting weight produces depreciation of the current votes held by each producer, a **vote decay** is intentional is introduced for two reasons: (a) Encourage participation by allowing newer votes to have more weight than older ones, and (b) Give more voice to those users who are actively involved in significant governance matters.
Producers who don't get voted on will keep their old votes, though depreciated due to vote decay. Producers who get voted on keep their old votes. The contribution of the last voting weight gets replaced by their new voting weight.

8.2.4 Scheduling

After the producers get selected for the next scheduled round, they are sorted alphabetically by producer name. This order determines the production order. Each producer receives the proposed set of producers for the next scheduled round within the first block to get validated from the currently scheduled round. The proposed schedule becomes active for the next scheduled round when that first block is finalized.

8.2.5 Block Lifecycle

Blocks undergo three main phases during their lifespan: production, validation, and finality. A block contains the producer's name and signature along with a schedule version and list of producers. The block production time (500ms) is further split into two configurable parameters based on maximum processing interval and minimum propagation time.
Blocks go through various stages during production: apply - create the block with metadata and transactions; finalize - validate the authenticity of the block, sign, and commit.
Block validation is about reaching enough quorum among active producers to agree upon; (a) The integrity of the block and the transactions it contains and (b) The deterministic, chronological order of transactions within each block.

Block Finality is the outcome of EOSIO consensus. It is achieved when a supermajority of active producers has validated the block according to the layer one consensus rules. The last irreversible block (LIB) in the chain refers to the most recent block that has become final.

Original Work

9

ASHWAChain: A Fast, Scalable and Strategy-proof Committee-based Blockchain Protocol

With our insight from the above report, we have understood that it is not possible to have a truly decentralized and highly scalable blockchain and secure system at the same time. With this in mind, we propose a novel committee-based blockchain consensus protocol, namely ASHWAChain, that balances between decentralization and centralization.
ASHWAChain provides the following:

1. **High scalability**: It can support up-to 300 Transactions Per Second (TPS), which is significantly more than most cryptocurrencies like Bitcoin and Ethereum.
2. **Strong consistency**: Transaction confirmation time is fast and within a few seconds. This is an improvement over existing cryptocurrency protocols where this time is several minutes, around 60 minutes in Bitcoin.

We provide scalability, security, and game-theoretic analysis of ASHWAChain. We consider three types of participants in the system, being Honest, Rational, or Byzantine. This consideration makes our analysis more practical as compared to existing literature where, in most cases, only Honest or Rational participants are considered.

Our paper "ASHWAChain: A Fast, Scalable and Strategy-proof Committee-based Blockchain Protocol" was accepted in Workshop on Game Theory in Blockchain at the 16th Conference on Web and Internet Economics (WINE), 2020, which we presented live in WINE 2020. This paper is provided in the appendix.

10

Acknowledgements

We would like to express special thanks to Anurag Jain and Sankarshan Damle for useful discussions and insightful comments on ASHWAChain.

List of Papers Summarized

[Read1] Eos: Consensus protocol. https://developers.eos.io/welcome/latest/protocol/consensus_protocol.

[Read2] J. Seidel C. Decker and R. Wattenhofer. Bitcoin meets strong consistency. In *In Proceedings of the 17th International Conference on Distributed Computing and Networking Conference (ICDCN)*. ICDCN, 2016.

[Read3] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system, 05 2018.

[Read4] Anindya Maiti Mohammad Hossein Manshaei, Murtuza Jadliwala and Mahdi Fooladgar. A game-theoretic analysis of shard-based permissionless blockchains. In *IEEE Access 6 (2018)*, page 78100–78112. IEEE, 2018.

[Read5] Or Sattath Ron Lavi and Aviv Zohar. Redesigning bitcoin's fee market. In *WWW '19: The World Wide Web Conference, May 2019*, pages 2950–2956, 2019.

[Read6] Yotam Sali and Aviv Zohar. Optimizing off-chain payment networks in cryptocurrencies. arXiv preprint arXiv:2007.09410, 2020.

[Read7] Maria Potop-Butucaru Yackolley Amoussou-Guenou, Bruno Biais and Sara Tucci-Piergiovanni. Rational vs byzantine players in consensus-based blockchains. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020, IFAAMAS, 9 pages*. IFAAMAS, 2020.

References

[1] A. R. Karlin M. E. Saks A. V. Goldberg, J. D. Hartline and A. Wright. Competitive auctions. In *Games and Economic Behavior, 55(2)*, page 242–269, 2006.

[2] Matthieu Bouvard Bruno Biais, Christophe Bisière and Catherine Casamatta. The blockchain folk theorem. In *The Review of Financial Studies (2019)*, 2019.

[3] Miguel Castro. Practical byzantine fault tolerance. In *Barbara Liskov, et al.* OSDI, 1999.

[4] Ueli Maurer Daniel Tschudi Christian Badertscher, Juan A. Garay and Vassilis Zikas. But why does it work? a rational protocol design treatment of bitcoin. In *In Advances in Cryptology - EUROCRYPT 2018*, page Part II. 34–65. 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, 2018.

[5] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. In *19th European Symposium on Research in Computer Security (ESORICS), Wroclaw, Poland, September 2014*, 2014.

[6] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Edmonton, Canada, 2015*, 2015.

[7] Marc Roeschlin Tobias Scherer Elli Androulaki, Ghassan Karame and Srdjan Capkun. Evaluating user privacy in bitcoin. In *IACR Cryptology ePrint Archive, 2012:596*, 2012.

[8] E. Androulaki G.O. Karame and S. Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. In *Computer and Communication Security*, 2012.

[9] Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. In *Journal of the Society for Industrial and Applied Mathematics, 9(4)*, page 551–570, (1961).

[10] Kartik Nayak Ling Ren Ittai Abraham, Dahlia Malkhi and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. CoRR abs/1612.02916v1, 2016.

[11] Ueli Maurer Björn Tackmann Juan A. Garay, Jonathan Katz and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *In 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA. 648–657*, page 648–657, 2013.

[12] C. Zheng K. Baweja S. Gilbert L. Luu, V. Narayanan and P. Saxena. A secure sharding protocol for open blockchains. In *the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[13] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Social Computing, 2011*, 2011.

[14] Michael K. Reiter. A secure group membership protocol. In *Transactions on Software Engineering*, 1996.

[15] Erel Segal-Halevi Simina Branzei and Aviv Zohar. How to charge lightning. CoRR, abs/1712.10222, 2017.

[16] Lennart Elsen Samuel Welten Tobias Bamert, Christian Decker and Roger Wattenhofer. Have a snack, pay with bitcoin. In *IEEE Internation Conference on Peer-to-Peer Computing (P2P), Trento, Italy*, 2013.

Appendix

Below is our paper "Sanidhay Arora; Anurag Jain; Sankarshan Damle; Sujit Gujar: ASHWAChain: A Fast, Scalable and Strategy-proof Committee-based Blockchain Protocol. Workshop on Game Theory in Blockchain at WINE 2020 (GTiB@WINE 2020), 2020.", which got accepted in Workshop on Game Theory in Blockchain at the 16th Conference on Web and Internet Economics (WINE), 2020.

ASHWAChain: A Fast, Scalable and Strategy-proof Committee-based Blockchain Protocol

Sanidhay Arora[0000−0002−5821−8294], Anurag Jain[0000−0002−4637−4708], Sankarshan Damle[0000−0003−1460−6102], and Sujit Gujar[0000−0003−4634−7862]

Machine Learning Lab,
International Institute of Information Technology, Hyderabad, India
sanidhay.arora@students.iiit.ac.in, {anurag.jain, sankarshan.damle}@research.iiit.ac.in
sujit.gujar@iiit.ac.in

Abstract. Most cryptocurrencies are practically limited, mostly because of their significant time to finality and lack of scalability. Moreover, most of the existing literature for blockchain consensus protocols assumes the miners as honest. The assumption results in the protocols being susceptible to strategic attacks such as selfish mining, undercutting. Consequently, designing scalable, strategy-proof blockchain consensus protocol forms the basis of this work. Towards this, we present ASHWAChain, which deploys committees, generated through an underlying Proof-of-Work blockchain, to reach consensus using PBFT. Through a sophisticated analysis of system performance, we show that ASHWAChain's performance is significantly better than the current state-of-the-art. Additionally, we analyze miners' strategic behavior in ASHWAChain and prove that at equilibrium, the miners will honestly follow the protocol under certain assumptions.

Keywords: Distributed Ledgers · Scalable Blockchains · Peer-to-Peer Networks · Game Theory

1 Introduction

Bitcoin [17] promised to transform the financial system by proposing a decentralized peer-to-peer currency. *Miners* maintain this system by honestly following the underlying *consensus protocol* through appropriate incentives/rewards. In Bitcoin and similar *cryptocurrencies* such as *Ethereum*, every miner validates transactions and tries to append a set of valid transactions to the set of validated transactions, i.e., append a block on the blockchain. These miners compete against one another in a "mining" game. Typically, these miners have to produce Proof-of-Work (PoW) to write new transaction data to the blockchain.

As of October 2020, Bitcoin's and Ethereum's network processes an average of 3-4 and 10 transactions per second (TPS), respectively [1,3]. In contrast, Visa's global payment system handles a reported 1,700 TPS and claims to be capable of handling more than 24,000 TPS [23]. Besides, using such cryptocurrencies for micro-payments, like groceries, coffee, etc., is not feasible as the protocols only provide *eventual consistency*. It means that each transaction requires a certain number of block confirmations for it to be confirmed, i.e., accepted as irreversible, with high probability. For instance, Bitcoin takes at-least 60 minutes to confirm a transaction[1]. Thus, low transaction throughput and only eventual consistency hinder the widespread acceptance of cryptocurrencies.

Note that the miners involved in maintaining the transaction data could be *strategic players*. That is, they may deviate from the prescribed protocol to gain additional *rewards*. Selfish mining, petty mining, undercutting are some of the strategies that may lead to greater rewards for the miners [7,19]. Designing blockchain protocols robust to such strategic deviations is a challenge and a new area of research with limited prior work [21]. In view of this, it may not be a strategic miner's *best response* to follow the protocol honestly. In summary, resolving these practical and game-theoretic limitations to design a decentralized, strategy-proof, and scalable blockchain consensus protocol is a considerable challenge.

Researchers have proposed several protocols to improve blockchain technology's practical performance for better applicability [9, 15, 20, 28]. These protocols provide 3-5 times improvement over Bitcoin in terms of TPS [20]. Chen et al. [10] use game-theory to establish the trade-off between full verification, scalability, and finality-duration. However, they do not consider network delays in their model, and hence, their bounds are quite optimistic. Even with this, it is impossible to have a scalable and consistent, fully decentralized blockchain.

[1] It is also referred to as a time to finality.

With this impossibility as a backdrop, in this paper, we propose a blockchain consensus protocol, namely ASHWAChain[2], which trade-offs some of the decentralization for improved consistency and scalability; thus, increasing its practical applicability. Specifically, we propose to select a *committee* of identities controlled by the miners who mine the blocks for some assigned duration. We refer to such duration as an *epoch*. We elect the committee for every epoch.

Such a *committee-based blockchain* is similar to EOS [2] and DIFINITY [22]. Unlike these protocols, in ASHWAChain, we present a more efficient way to select the committee. Additionally, we also consider the strategic behavior of miners. Specifically, ASHWAChain uses PoW as a method to guarantee Sybil-resistance. The guarantee holds under the standard assumption that no miner will have the majority of the computational resource. In ASHWAChain, after solving a PoW puzzle, a miner gains an *identity* in the committee. We call the identities in this committee as *validators* that run a Byzantine agreement protocol, PBFT [8], to reach consensus.

First, we provide security and scalability analysis of ASHWAChain. We show that ASHWAChain can achieve a throughput of at-least 300 TPS (Section 3.2). Thus outperforming the cryptocurrencies mentioned above. Next, to game-theoretically analyze miners' behavior at equilibrium in ASHWAChain, we consider three types of miners: *honest*, *rational*, and *Byzantine*. We prove that it is Bayesian Nash Equilibrium for rational players to validate and sign the blocks (Theorem 1). That is, ASHWAChain is robust to the strategic behavior of miners. In summary, ASHWAChain provides strong consistency (fast confirmations) and high scalability while maintaining the system's security.

Related Work. Committee-based blockchains like EOS [2] claim to scale up-to 5000 TPS but still have open security concerns. While DFINITY [22] can handle up-to 40 TPS, it is significantly less for a financial transaction platform. Our work is most similar to PeerCensus [6], where identities in the network gain privileges to produce and validate blocks by solving a PoW puzzle. However, as the size of the committee is ever-growing, PeerCensus becomes infeasible to scale. Garay et al. [12] showed that deferring the resolution of blockchain forks is inefficient and strong consistency provides more applicability. Most of these protocols, however, assume participants as either honest or Byzantine, failing to explore the effect of rational participants thoroughly.

Solidus [14] is the first to consider rational participants by proposing an incentive-compatible Byzantine-Fault-Tolerant protocol for blockchain. However, it does not provide a game-theoretic analysis. Biais et al. [5] model Bitcoin as a coordination game but only considering rational participants. Yackolley et al. in [26] provide a game-theoretic analysis in consensus-based blockchains considering rational and Byzantine players and model a dynamic game. Manshaei et al. in [16] consider a non-cooperative static game approach for an intra-committee protocol where they show that rational players can free-ride if rewards are equally shared. Contrarily to previous studies, the consideration of the costs of block validation in the player's utilities makes our study more realistic. We follow the BAR model [4] and study the rational behavior in a non-cooperative setting, as shown by Halpern et al. in [13].

2 ASHWAChain: Protocol

In this section, we present our committee-based blockchain protocol, namely ASHWAChain. We first define the underlying network in it.

Network Model. We consider a peer-to-peer network consisting of miners who control identities in the network. These identities are denoted by their public-private (pk, sk) key pair. Miners are connected by a broadcast network over which they can send messages to everyone.

ASHWAChain comprises of two layers, (i) the PoW Blockchain layer (PBL) and (ii) the Agreement layer (AL). PBL is responsible for providing Sybil-resistant identities, and AL is responsible for maintaining consensus, i.e., fork-resolution and transaction confirmation.

2.1 Proof-of-Work Blockchain Layer (PBL)

The key insight behind PoW mechanisms is that the computational resources needed to solve cryptographic puzzles are not easily acquired and may not be scaled at will. In ASHWAChain, we leverage the same to provide Sybil-resistant identities. PBL consists of a blockchain running a PoW protocol, similar to Bitcoin. Each block at height i, i.e., b_i, is of the form $b_i = \langle h, d, x, p \rangle$. Here $h = H(b_{i-1})$ is the hash of previous block, d is the difficulty of mining, x is the nonce, and p is the identity controlled by a miner that will join the committee to become a *validator*.

[2] *Ashwa* refers to a horse which one can ride with high speed without much hassle [24].

Once a miner mines a block b, PBL provides a `proposeBlock`(b) operation to interact with the Agreement Layer (AL). Specifically, it proposes b to AL. Note that if more than one valid block is proposed at the same time, then only one of them is accepted. Once b is accepted in AL, a `commitBlock`(b) event is triggered in PBL for `proposeBlock`(b). Post this, the identity p in b joins the AL. All the miners in PBL acknowledge the addition of this block. The updated chain is then considered to mine further blocks, i.e., each miner stops mining the current block, chooses a different identity in the block if their own block was accepted, and then continues to mine a new block.

2.2 Agreement Layer (AL)

AL is maintained by a committee of identities. We refer to the identities present in the committee as *validators*. AL comprises a shared state which consists of PowChain, T, ComChain, TxChain, O and B, which are defined later in this subsection and summarised in the Appendix (Table A.1). This shared state is maintained by the committee of validators running SGMP [18] and PBFT [8] agreement protocols, similar to PeerCensus [6]. The shared state can only be modified by three pre-determined operations: `powBlock`(b), `txBlock`(t) and `comBlock`(c) which are defined later in Algorithm 1. For easy reference, summary is provided in Appendix (Table A.2).

AL proceeds in epochs where each epoch consists of t_{epoch} rounds. In ASHWAChain, a round is said to be complete after each commit of the `powBlock`(b) operation (refer Algorithm 1). The committee is *updated* after each epoch. The committee is of a fixed number of identities, $n_C = advRate \times t_{epoch}$, where $advRate$ is a system parameter that determines the size of the committee based on epoch size.

The state ComChain stores a blockchain with each block c containing the list of validators. This state is updated after each epoch with `comBlock`(c), as defined in Algorithm 1. The identities present in the latest block of ComChain are considered as validators for the current epoch. These identities are the same as those present in the latest n_C blocks in the PowChain state, which stores the PoW blockchain from PBL, and is updated with `powBlock`(b) operation (Algorithm 1).

A primary validator is selected to run the agreement protocol, formally presented in Appendix (Algorithm 2). In PowChain, for chain length l and blocks $(b_1, \ldots, b_{l-t_{epoch}+1}, \ldots, b_l)$, the identity in block $b_{l-t_{epoch}+1}$ is considered the *primary* and in case of Byzantine behaviour, next block in the PowChain will used to select the primary validator, using similar procedures as in PBFT. When the miners proceed to the next round after b is committed in PowChain, the next validator in the committee is considered as primary.

The agreement procedure for an identity p in the committee is explained as follows, formally presented in Appendix (Algorithm 2). The Primary validator creates the necessary operation and broadcasts it to the committee. The operation is then validated according to its pre-defined rule in Algorithm 1. Validators wait to collect all the messages for a certain time according to network latency. Once the operation is signed by a super-majority [3] of validators, they append this operation to operation log O. Post this, they commit the operation according to its commit rule as defined in Algorithm 1.

Algorithm 1 defines all the operations of the AL. Validation of any operation involves validating all the signatures, transactions, and the block structure. The `powBlock`(b) operation is broadcasted by the Primary validator after receiving the `proposeBlock`(b) request from a miner in the PBL layer. Then, b is appended to the PowChain once this operation is committed. In `txBlock`(t) operation, t is the block of transactions and each transaction is of the form $tx = \langle from, to, sign, coins, txFee \rangle$, where $from$ and to are the addresses of sender and recipient, $sign$ is the signature of the sender, $coins$ is the number of coins, and $txFee$ is the transaction fee. Once this operation is committed, all the transactions are applied accordingly by updating the state B, which stores the Account Balances. Once all the transactions are applied, the state T, which stores the list of validators (who signed the transactions for each tx) is updated. Note that v is the validator address in Algorithm 1. Post this, Block Reward BR, and the total transaction fee is distributed equally among the validators who signed the respective transactions. Finally, the state TxChain, which stores the transaction blockchain, is updated by appending t to it. In summary, Figure 1 provides the schematic representation of ASHWAChain.

3 ASHWAChain: Analysis

We now analyze the security and game-theoretic aspects of ASHWAChain. For this, we begin by defining our adversary and player models.

[3] $\frac{2}{3}^{rd}$ of total validators + 1 as we deploy PBFT.

Algorithm 1: Operations

Validate `powBlock`(b): **begin**
  $b^* \leftarrow$ latest block in *PowChain*
  if *b is child of b^* and b is valid* **then**
    return Valid
  else
    return Invalid

Validate `comBlock`(c): **begin**
  if *All t_{epoch} identities in c match with the PowChain and c is valid* **then**
    return Valid
  else
    return Invalid

Validate `txBlock`(t): **begin**
  if *signatures and all txs are valid* **then**
    return Valid
  else
    return Invalid

Commit `powBlock`(b): **begin**
  Append `powBlock`(b) in O
  Append b to PowChain

Commit `comBlock`(c): **begin**
  Append `comBlock`(c) in O
  Append c to ComChain

Commit `txBlock`(t): **begin**
  Append `txBlock`(t) to O
  for $tx \in t$ **do**
    $B(from) \leftarrow B(from) - coins$
    $B(to) \leftarrow B(to) + coins$
  Append the list of validators of t in T
  for $v \in T(t)$; *nSig* $\leftarrow$ *Total signatures* **do**
    $B(v) \leftarrow B(v) + \frac{\sum txFee + \text{Block Reward}}{nSig}$
  Append t to TxChain

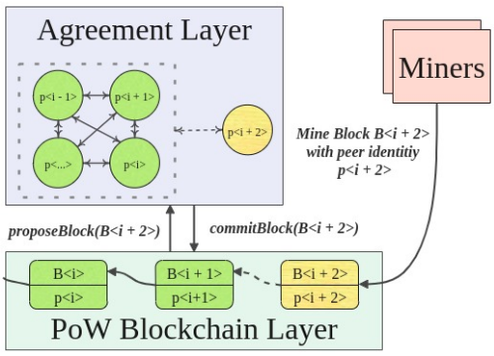

Fig. 1: ASHWAChain overview at height i

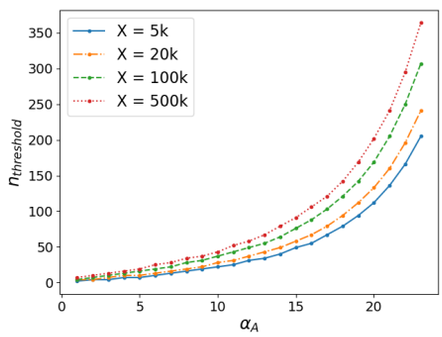

Fig. 2: For a given α_A, required $n_{threshold}$ such that the probability of the system not being in secure state is $1/X$

Adversary Model. We consider an adversary A controlling α_A fraction of computational resources in the network. The remaining resources are controlled by a meta entity H. The goal of the adversary is to control more than n_{ag} identities, i.e., the minimum number of identities needed for agreement. As we deploy PBFT to reach consensus, we have $n_{ag} \geq \frac{2}{3}n_C + 1$. Moreover, the probability of an adversary joining the committee will be directly proportional to its computation power. Similar to [6], we analyze ASHWAChain in its *steady state*, i.e., the number of validators and computational resources are governed by their respective expected value.

Player Model. In ASHWAChain, we consider the miners to be strategic. Towards analyzing their behavior at equilibrium, we consider three *types* of players:

- $\mathcal{H}$ **Honest.** These players do not deviate from the defined protocol.
- $\mathcal{R}$ **Rational.** These players follow the protocol if it yields the highest utility but may switch to alternative strategies, such as double spending, if they yield a higher utility.
- $\mathcal{B}$ **Byzantine.** These players actively try to compromise the system, by trying to double spend etc., irrespective of the utilities they obtain.

As stated, the reward structure of any blockchain consensus protocol induces a game among the participants. To analyze this we define the following notations: Consider a set of miners $\mathcal{M} = \{1, \ldots, m\}$. Let each miner i's strategy be s_i and its type be τ_i, where $\tau_i \in \{\mathcal{H}, \mathcal{R}, \mathcal{B}\}$. Note that, the strategy will depend on the miner's type. We have $\mathcal{S} = \{s_1, \ldots, s_m\}$ as the strategy vector and $\mathcal{S}_{-i}$ as the vector without player i. Let r_i denote miner i's reward. With this, $u_i(\mathcal{S}, \tau_i, r_i)$ represents a miner i's utility from its participation. In ASHWAChain, we game-theoretically analyze the player's behavior at equilibrium using the following notion.

Definition 1 (Bayesian Nash Equilibrium (BNE)). *A strategy vector $\mathcal{S}^* = \{s_1^*, \ldots, s_m^*\}$ is said to be a Bayesian Nash Equilibrium (BNE) if for every player i, it maximizes its expected utility $u_i(\mathcal{S}^*, \tau_i, r_i)$ i.e., $\forall i \in \mathcal{M},\ \forall j$,*

$$\mathbb{E}_{\mathcal{S}_{-i}}\left[u_i(\mathcal{S}^*, \tau_i, r_i)\right] \geq \mathbb{E}_{\mathcal{S}_{-i}}\left[u_i(s_i, \mathcal{S}^*_{-i}, \tau_i, r_i)\right];\ \forall s_i. \tag{1}$$

Intuitively, BNE states that on average it is the best response for a player to follow $\mathcal{S}^*$, assuming every other player is following it.

3.1 Security Analysis

We first analyze the security aspects of ASHWAChain. Our analysis relies on the standard assumption of PBFT that the system will only work correctly if the number of adversary identities in the committee is less then one third of the committee size. To capture this formally, we define the following.

Definition 2 (Secure State). *The system is said to be in a secure state if n_A, i.e., the number of validators controlled by an adversary, is strictly less than $\frac{1}{3} \cdot n_C$.*

Proposition 1. *The system will be in a secure state with high probability depending on $n_{threshold}$ and β if (i) α_A, the fraction of computational resources in the network controlled by the adversary A is upper bounded by a fraction β and (ii) $n_C \geq n_{threshold}$, where $n_{threshold} \in \mathbb{Z}^+$.*

Proposition 1 follows with Definition 2. We provide the formal proof in Appendix (Section A.1).

Selecting n_C. Observe that the probability of a miner being added to PBL is directly proportional to α_{miner}, i.e., the computational power it holds in the system. Consequently, we can model the probability distribution of the identities that will be added to PBL as a *binomial distribution* [25]. We know that, from Proposition 1, the system will not be in a secure state if the number of validators, n_A, controlled by the adversary are $\geq \frac{1}{3} \cdot n_C$. Thus, the probability of the system not being in a secure state becomes the binomial distribution of n_A successes in n_C independent experiments, with α_A as the probability of a each success[4]. With this observation, Figure 2 shows the graph between $n_{threshold}$, i.e., the minimum number of identities needed to ensure secure state with respect to α_A for different probability guarantees. More concretely, Figure 2 presents the threshold of the committee size required, for a given α_A, such that the probability with which the system is *not* in a secure state is $1/X$, where $X \in \mathbb{Z}^+$.

3.2 Scalability Analysis

ASHWAChain runs PBFT agreement protocol which has $O(n^2)$ time complexity where n is the number of identities. Thus, the transaction throughput in our protocol will depend on n_C. Yaqin et al. [27] show that PBFT can scale up-to 200 TPS for 100 identities, where each transaction's size is 50 bytes. We remark that ASHWAChain provides strong consistency, i.e., a transaction is considered as irreversible once it is accepted. This is in contrast to other cryptocurrencies running PoW and PoS protocols like Bitcoin, Ethereum, etc., which need multiple block confirmations for block finality. As a result, the PBFT agreement protocol is the computational bottleneck in ASHWAChain. Note that, the aforementioned cryptocurrencies handle around 20 TPS. Committee-based blockchains like BitShares and DFINITY handle around 18 and 40 TPS respectively. IOTA [11] uses a DAG-based protocol and handles up-to 50 TPS.

From our security analysis, if we consider $\alpha_A = 0.18$ fraction of the computational power controlled by the adversary, then for $n_C = 90$, the probability of the system being in a secure state will be greater than 1.44×10^{-4} (Figure 2). For this n_C, ASHWAChain can support at-least 300 TPS, which is a significant improvement over existing cryptocurrency protocols.

3.3 Equilibrium Analysis

Towards this, we assume that all rational players incur a cost $\kappa > 0$ if any malicious transaction block is committed, similar to [26]. This assumption is based on the premise that the entire ecosystem of the currency inflicts harm when an invalid block is accepted and, since rational players have invested resources and possess a stake in the system, they must incur some cost depending on it. We also assume that the objective of Byzantine players is to minimize the utility of the rational players and prevent the protocol from achieving its goal, regardless of the cost they incur. With these assumptions, we

Term	Definition
N	Set of identities, $\{1, 2, \ldots, n_C\}$
τ_i	Player i's type, i.e.,$\tau_i \in \{\mathcal{H}, \mathcal{R}, \mathcal{B}\}$
u_i	Utility of player i; $u_i : \tau_i \times TR \times S \to \mathbb{R}$
s_i^1	Player i signs the transaction block without validating
s_i^2	Player i signs the transaction block only if its valid
s_i^3	Player i signs and proposes only invalid blocks
S_i	Set of pure strategies of players i, $\{s_i^1, s_i^2, s_i^3\}$

Table 1: Game constituents

Term	Definition
c_{mine}	The cost of mining a block
c_{val}	The cost to validate a tx
$txFee$	Average transaction fee
κ	Cost incurred by rational players if invalid block is accepted
ϕ	Number of transactions in each block
ψ	Number of identities signing a block
BR	Transaction Block reward
$P_{invalid}$	Belief probability of invalid block being accepted, $P_{invalid} : N^{n_C} \times S_i \to [0, 1]$
TR	Total reward gained, $\frac{\phi \cdot txFee + BR}{\psi}$

Table 2: Relevant Notations

model a game between honest, rational and Byzantine players' identities in the committee, defined as: $\Gamma = \langle N, (\tau_i), (S_i), (u_i) \rangle$. Table 1 and Table 2 comprises the notations used in our analysis.

Note that $P_{invalid}(N, s_i^*)$ is player i's belief that an invalid block is accepted after it follows s_i^*. Honest and Byzantine players will always follow s_i^2 and s_i^3, respectively. Hence we only consider rational players' behavior and therefore their equilibrium strategies. The expected utilities of a rational player i for one round are as follows:

$$u_i(\cdot, TR, s_i^1) = TR - \frac{c_{mine}}{(n_C - t_{epoch})} - P_{invalid}(N, s_i^1) \cdot \kappa \tag{2}$$

$$u_i(\cdot, TR, s_i^2) = TR - \frac{\phi \cdot c_{val}}{\psi} - \frac{c_{mine}}{(n_C - t_{epoch})} - P_{invalid}(N, s_i^2) \cdot \kappa \tag{3}$$

In a committee of n_C identities, let n_H, n_R, and n_B be number of honest, rational and Byzantine players' identities respectively. Trivially, $n_C = n_H + n_R + n_B$. We use BNE for analysis, as we have a static game with asymmetric information. As stated, the minimum number of identities needed for agreement are n_{ag}. As we use PBFT, we have, $n_{ag} = \frac{2}{3}n_C + 1$. Our analysis relies on the standard assumption that the system will only work correctly if $n_H + n_R \geq n_{ag}$, i.e., $n_B < n_C - n_{ag}$.

Claim 1. *For every rational player's identity i, its belief regarding the probability of an invalid block being accepted will be more if it signs a block without validating, as compared to when it validates and then signs, i.e., $\delta = P_{invalid}(N, s_i^1) > P_{invalid}(N, s_i^2)$ s.t. $\delta > 0$.*

Intuitively, Claim 1 follows from the fact that the chance of an invalid block being accepted is more if a rational player decides not to validate a block before signing the block. This is because the size of the committee, N, is finite. We present the formal proof in the Appendix (Section A.2).

Theorem 1. *In ASHWAChain, if (i) $n_B < n_C - n_{ag}$ and (ii) $\phi \leq \frac{\kappa \cdot \psi \delta}{c_{val}}$, then for every rational player i, $s_i^* = s_i^2$ is a BNE. Here, $\delta = P_{invalid}(N, s_i^1) - P_{invalid}(N, s_i^2)$ s.t. $\delta > 0$.*

Theorem 1 follows from Claim 1 and by observing that (3) is greater than (2) when $\phi \leq \frac{\kappa \cdot \psi \delta}{c_{val}}$. The formal proof is provided in the Appendix (Section A.3). However, in a special case where $n_H \geq n_{ag}$, $S^* = (s_i^1)$ is the BNE for rational players where only valid blocks are accepted.

4 Conclusion

In this work, we introduced a blockchain consensus protocol, namely ASHWAChain that provides strong consistency and high scalability, by using a committee-based system for consensus (Section 2). Our security analysis shows that the system is in a secure state (Section 3.2), with high probability (Figure 2). Lastly, we proved that under certain assumptions on the protocol parameters, there exists a Bayesian Nash Equilibrium in which the rational players validate the transactions before signing any block (Theorem 1).

[4] For additional details, we refer the reader to Section A.1

References

1. Blockchain Charts (2020), https://www.blockchain.com/charts/transactions-per-second
2. Documentation/TechnicalWhitepaper.md (2020), https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md
3. Ethereum Daily Transactions Chart | Etherscan (2020), https://etherscan.io/chart/tx
4. Amitanand S. Aiyer, Lorenzo Alvisi, A.C.M.D.J.M., Porth, C.: Bar fault tolerance for cooperative services. In: Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP 2005, Brighton, UK, October 23-26, 2005. p. 45–58 (2005)
5. Bruno Biais, Christophe Bisière, M.B., Casamatta, C.: The blockchain folk theorem. In: The Review of Financial Studies (2019) (2019)
6. C. Decker, J.S., Wattenhofer, R.: Bitcoin meets strong consistency. In: In Proceedings of the 17th International Conference on Distributed Computing and Networking Conference (ICDCN). ICDCN (2016), https://tik-db.ee.ethz.ch/file/ed3e5da74fbca5584920e434d9976a12/peercensus.pdf
7. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167 (2016)
8. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, February 1999. Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139 (1999)
9. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155 – 183 (2019). https://doi.org/https://doi.org/10.1016/j.tcs.2019.02.001, in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I
10. Chen, L., Xu, L., Gao, Z., Kasichainula, K., Shi, W.: Nonlinear blockchain scalability: a game-theoretic perspective. arXiv preprint arXiv:2001.08231 (2020)
11. Divya, M., Biradar, N.: Iota-next generation block chain. International Journal Of Engineering And Computer Science **7**, 23823–23826 (04 2018). https://doi.org/10.18535/ijecs/v7i4.05
12. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. pp. 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
13. Halpern, J.Y., Vilaça, X.: Rational consensus: Extended abstract. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing. p. 137–146. PODC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2933057.2933088
14. Ittai Abraham, Dahlia Malkhi, K.N.L.R., Spiegelman, A.: Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. In: CoRR abs/1612.02916v1 (2016) (2016)
15. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. pp. 357–388 (07 2017). https://doi.org/10.1007/978-3-319-63688-7_12
16. Manshaei, M.H., Jadliwala, M., Maiti, A., Fooladgar, M.: A game-theoretic analysis of shard-based permissionless blockchains. IEEE Access **6**, 78100–78112 (2018). https://doi.org/10.1109/ACCESS.2018.2884764
17. Nakomoto, S.: Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf
18. Reiter, M.K.: A secure group membership protocol. In: Transactions on Software Engineering (1996)
19. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)
20. Siddiqui, S., Gujar, S.: Quicksync: A quickly synchronizing pos-based blockchain protocol. CoRR **abs/2005.03564** (2020), https://arxiv.org/abs/2005.03564
21. Siddiqui, S., Vanahalli, G., Gujar, S.: Bitcoinf: Achieving fairness for bitcoin in transaction fee only model. In: Seghrouchni, A.E.F., Sukthankar, G., An, B., Yorke-Smith, N. (eds.) Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020. pp. 2008–2010. International Foundation for Autonomous Agents and Multiagent Systems (2020)
22. Timo Hanke, M.M., Williams, D.: Dfinity technology overview series consensus system. CoRR abs/1805.04548 (2018) (2018), https://arxiv.org/pdf/1805.04548.pdf
23. Visa Inc.: Visanet booklet, https://usa.visa.com/dam/VCOM/download/corporate/media/visanet-technology/visa-net-booklet.pdf
24. Wikipedia contributors: Ashva — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Ashva&oldid=988520218 (2020)
25. Wikipedia contributors: Binomial distribution — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Binomial_distribution&oldid=989183881 (2020)
26. Yackolley Amoussou-Guenou, Bruno Biais, M.P.B., Tucci-Piergiovanni, S.: Rational vs byzantine players in consensus-based blockchains. In: In Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020, IFAAMAS, 9 pages. IFAAMAS (2020), http://ifaamas.org/Proceedings/aamas2020/pdfs/p43.pdf
27. Yaqin Wu, Pengxin Song, F.W.: Hybrid consensus algorithm optimization: A mathematical method based on pos and pbft and its application in blockchain. Mathematical Problems in Engineering, vol. 2020, Article ID 7270624, 13 pages, 2020 **2020** (2020). https://doi.org/https://doi.org/10.1155/2020/7270624
28. Yu, H., Nikolić, I., Hou, R., Saxena, P.: Ohie: Blockchain scaling made simple. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 90–105. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00008

Appendix

A Proofs

In this section, we restate and present formal proofs of the results presented in the main paper.

A.1 Proposition 1 Proof

Proposition. The system will be in a secure state with high probability depending on $n_{threshold}$ and β if (i) α_A, the fraction of computational resources in the network controlled by the adversary A is upper bounded by a fraction β and (ii) $n_C \geq n_{threshold}$, where $n_{threshold} \in \mathbb{Z}^+$.

Proof. Since we consider PBFT, Byzantine identities should be strictly less than one-third of total identities, i.e., $n_B < \frac{1}{3} \cdot n_C$, for the system to work correctly. We know that, the probability that a miner mines a block is directly proportional to its fraction of computational power in the system, i.e., $Prob$(Miner to mine a block) $\propto \alpha_{miner}$. This follows by observing that the size of the committee, i.e., n_C is finite. Let X be the discrete random variable denoting the number of validators controlled by the adversary and consider the following binomial distribution to derive the probability of the number of validators controlled by the adversary A to be,

$$f(k, n_C, \alpha_A) = Pr(k; n_C, \alpha_A) = Pr(X = k) = \binom{n_C}{k}(\alpha_A)^k(1 - \alpha_A)^{(n_C - k)}$$

Now consider the following Cumulative Distribution Function (CDF),

$$F(x) = Pr(X \leq x).$$

Note that, $F(\frac{1}{3} \cdot n_C)$ gives the probability that the system is in a secure state, i.e. $n_A < \frac{1}{3} \cdot n_C$, which depends on the value of n_C and α_A from the binary distribution. This proves the proposition. □

A.2 Claim 1 Proof

Claim. *In ASHWAChain, for every rational player's identity i, its belief regarding the probability of an invalid block being accepted will be more if it signs a block without validating, as compared to when it validates and then signs, i.e., $\delta = P_{invalid}(N, s_i^1) > P_{invalid}(N, s_i^2)$ s.t. $\delta > 0$.*

Proof. Without loss of generality, for the proof we consider a rational player i. Let $P_{invalid}(N, s_i^1) = k_1$ and $P_{invalid}(N, s_i^2) = k_2$ s.t. $0 < k_1, k_2 < 1$. In the event when player i plays the strategy s_i^1, i.e., signs the block without validating, its belief regarding an invalid block being accepted can only increase. This follows by observing that the size of the committee, i.e., n_C is finite. Thus, player i not validating a block will directly imply that the chance of an invalid block being accepted will be more, i.e., $k_1 > k_2$. Now,

$$\begin{aligned} P_{invalid}(N, s_i^1) - P_{invalid}(N, s_i^2) &= k_1 - k_2 \\ &= \delta > 0. \end{aligned}$$

This proves the claim. □

A.3 Theorem 1 Proof

Theorem. *In ASHWAChain, if*
(i) $n_B < n_C - n_{ag}$ and (ii) $\phi \leq \frac{\kappa \cdot \psi \delta}{c_{val}}$,
then for every rational player i, such that $s_i^* = s_i^2$ is BNE. Here, $\delta = P_{invalid}(N, s_i^1) > P_{invalid}(N, s_i^2)$ s.t. $\delta > 0$.

Proof. The condition $n_B < n_C - nag$ should be true for PBFT to work correctly. For $s*_i = s_i^2$ to be a BNE for every rational player i, $u_i(R, s_i^1) \leq u_i(R, s_i^2)$ should satisfy according to Equation 1. On solving this condition using equations 2 and 3 we get,

$$\kappa \cdot (P_{invalid}(N, s_i^1) - P_{invalid}(N, s_i^2)) \geq \frac{\phi \cdot c_{val}}{\psi}.$$

Substituting $(P_{invalid}(N, s_i^1) - P_{invalid}(N, s_i^2))$ as δ, where $\delta \in R^+$ following from Claim 1, we get our result for the value of ϕ as:

$$\phi \leq \frac{\kappa \cdot \psi \delta}{c_{val}}. \quad \square$$

Name	State
PowChain	Proof of Work Blockchain
T	List of transaction validators
ComChain	Committee Blockchain
TxChain	Transaction Blockchain
O	Operation Log
B	Account Balances

Table A.1: Shared state constituents

Name	Operation
`powBlock`(b)	Adds PoW block b to PowChain
`txBlock`(t)	Adds transaction block t to T
`comBlock`(c)	Adds committee block c to ComChain

Table A.2: Shared state operations

B Notations

For readability, Table A.1 and Table A.2 summarize some of the notations presented in the main paper.

C Algorithms

In this section, we formally present the algorithms used in the main paper.

Algorithm 2: Agreement for identity p

```
begin
    if p = Primary then
        Creates any operation and broadcasts
    else
        Validate operation, sign and broadcast
    wait(LatencyTime)
    if signed by super-majority then
        Append operation to O
        Commit operation according to its commit rule
    else
        Nothing is committed, RESET
```