Improvements of Blockchain's Block Broadcasting: An Incentive Approach

***Abstract*—In order to achieve a truthful distributed ledger, homogeneous nodes in Blockchain systems will propagate messages on a P2P network so that they can synchronize the status of the ledger. Currently, blockchain systems target on achieving better scalability and higher throughput to support divergent applications which will lead to heavier message propagation, especially the broadcasting of blocks. The heavier traffic on the P2P network will cause longer latency of block synchronization, which may damage system consistency and expose the system to potential attacks. Even worse, when heavy communication consumes a lot of network capacity, nodes in the P2P network may not relay blocks to save their bandwidth. This may damage the efficiency of network synchronization.**

In order to alleviate the problems, we propose an improved block broadcasting protocol which elaborates block data sharding and financial incentive mechanisms. In the proposed scheme, a block is sliced into pieces in order to keep the network traffic smooth and speed up content delivery. Any node which relays a piece of the block will get benefits with financial rewards. By applying data sharding, our proposed scheme speed up the block broadcasting and therefore shorten the synchronization time by 90%, which is shown in our simulation experiments. In addition, we carry out game theoretical analysis to prove that nodes are efficiently incentivized to relay blocks honestly and actively.

I. Introduction

In 2008, Nakamoto published his celebrated paper [1] in which introduced a practical blockchain consensus protocol and later was known as *Bitcoin* protocol. Bitcoin protocol is a novel scheme to maintain a distributed ledger which is safeguarded by all peers in a decentralized system [2]. The advantages of blockchain, such as decentralization, irreversibility, and undeniableness, provide an approach to leverage verifiable interactions among non-trust peers in decentralized systems. As a trade-off, such decentralized consensus is expensive to achieve. Bitcoin, for instance, consumes enormous mining power but only generate a block about every 10 minutes which will provide about 7 Transaction Per Second (TPS) on average. Therefore, the urgent need for fast and cheap transaction processing in blockchain attracts lots of researches. A new version of Bitcoin is proposed which will increase block size [3] and use the lightning network to process frequent small transactions more efficiently [4]. Other works try to integrate DAG [5, 6] and network sharding [7, 8] into Bitcoin scheme and improve the throughput of blockchain using clever consensus design. Above improvements on throughput mainly have two effects: larger block size [3] and higher frequency of block generation [9].

However, we observe that the bottleneck of blockchain will also come to the network layer even if the consensus makes a breakthrough to provide high throughput. A simple calculation can illustrate this concern. The size of a basic transaction with one input and two outputs in Bitcoin is about 250 bytes [10]. If we want to achieve 10000 TPS (Visa achieves about 24000 TPS) in blockchain, the overall throughput is at least 2.5 MB/s, which is a nonnegligible bandwidth cost. In fact, the size of transactions for smart contracts will be even much larger. In current blockchain consensuses, nodes with verification capability (aka full node) are usually required to synchronize all chain data. Blockchain systems are robust only with a significant number of full nodes. As a result, as long as the chain data is generated faster and a large number of full nodes are in function, the message propagation in blockchain P2P network should be much faster to maintain in-time synchronization. The overloaded P2P network layer can cause *long synchronization latency* and *absence of incentive for broadcasting* challenges. We analyze the above two challenges and improve the block broadcasting scheme to solve them.

Synchronization latency means the latency from the generation of a message to confirmation of the same message by all honest nodes. [11] addresses the strong correlation between the size of message and synchronization latency. Since there is the trend of larger blocks, we conduct that synchronization latency will significantly grow in future blockchain network. Slow synchronization can damage security properties of blockchain systems [12, 13], waste mining power [14], cause blockchain forks [11] and even expose the vulnerability of adversary attacks like selfish mining [15, 16]. In order to shorten synchronization latency, we divide each block into pieces and the piece is the basic unit of data transfer. Similar to P2P content delivery applications, transfer in pieces makes the most of each node's communication bandwidth and make the broadcast more efficient [17]. In our simulation in Section VII-A, block sharding shorten the synchronization latency by about 90%.

We also analyze block broadcasting in an incentive way. In the former analysis on blockchain P2P network, it is assumed that honest nodes will correctly perform block verification and broadcasting. However, in P2P broadcast network with heavy workloads, the broadcast consumes significant network bandwidth. As a result, rational nodes may refuse to relay new blocks to others or reduce the number of neighbors in network topology, which also damage the efficiency of block broadcasting. In order to incentivize nodes to verify and broadcast blocks, we introduce blockchain-based financial incentive [18, 19, 20] into block broadcasting. We notice that cryptocurrencies, as the initial application of blockchain, can be an effective, compatible financial resource. We name the nodes who send out block data as uploader and nodes who download the block data as the downloader. If uploaders earn money and downloaders bear a financial cost in the broadcast process, it is the simplest prototype of blockchain-based pecu-

niary incentive. In intuition, nodes are intended to download blocks because of the requirement of synchronization and meanwhile uploading is incentivized by the financial rewards. We formally analyze the incentive mechanism through game theoretical models in Section VI. From our simulation in Section VII, the blockchain-based incentive is sound and effective. The system will converge to the equilibrium of honest behavior as long as there is a sufficient initial fraction of honest nodes.

As for considerations of scalability, we keep the interaction between uploaders and downloaders simple. Since the transfer of cryptocurrency requires extra transactions, we also minimize the number of such transaction calls using micropayment channels [21]. Since micro-payments require the submission of only one transaction after a large number of payments between two nodes, the number of extra transactions will not significantly increase in a long period, therefore the extra cost will not significantly influence the original performance of system scalability. We demonstrate our proposed block broadcasting protocol in Bitcoin environment and the protocol is suitable for other blockchain systems with block-based consensus.

Our contributions. First, we introduce a novel block broadcasting scheme with blockchain-based incentive mechanism. The scheme is a generic approach for block-based blockchain systems to further improve the efficiency of block propagation. Section V elaborately depicts the protocol design, including interactions during block broadcasting.

Also, we use game theoretical model to analyze incentive for broadcasting and discuss the fairness of data transfer. We introduce an evolutionary game model to analyze the proposed incentive mechanism formally, which shows the stable equilibrium of active cooperation (Section VI-A). Besides, we use a repeated game model and learning model to analyze the fairness and cooperative behavior in the process of data transfer by pieces (Section VI-B).

At last, we carry out simulations and experiments in Section VII, which prove that proposed scheme speeds up synchronization and the incentive is sound and practical.

II. Background

In this section, we introduce backgrounds of some related conceptions and technologies.

Bitcoin and blockchain. Blockchain is a distributed digital ledger over a peer-to-peer network that achieves decentralized agreements between nodes with only minimal trust [2]. The ledger is organized in a special chain-like data structure which is maintained by all participants (i.e. nodes) in the system. The blockchain supports offline-verification but adversary parties cannot reverse the ledger once it is admitted by consensus, given the consumption that the majority of participants is honest.

In this paper, we use Bitcoin as an example of typical blockchain system to illustrate our proposed block broadcasting scheme. Bitcoin is the first application built on blockchain, also the first cryptocurrency, which is proposed in 2008 [1]. Therefore, Bitcoin presents all the nature of blockchain and the soundness of Bitcoin is proved both practically and theoretically.

Block broadcasting in Bitcoin. Blockchain P2P network is homogeneous and all full nodes (i.e. nodes who are able to perform verification on the ledger) keep a complete replica of the necessary information for verification, including blocks. A node who finds a new block take three methods, *unsolicited block push*, *standard block relay* and *direct headers announcement*, to broadcast it [10]. *Unsolicited block push* usually happens when the node is exactly the generator of the found block. In this way, the node knows that no neighbor possesses this newly generated block so he directly sends the block data to neighbors. Except for the special situation that the node is the block generator, nodes execute *standard block relay*. We focus on improving *standard block relay* in this work. To be detailed, a typical *standard block relay* broadcast from node A to B contains following steps: 1) After A discovers a new block, A sends an *inv* message to its neighbor B, telling him the existence of the new block. 2) B checks whether the block has already been included in its local storage. If not, B request the header from A. 3) B checks the integrity of *headers* and request full block data.

Optionally, nodes can skip step 1 to directly broadcast headers to neighbors, and this method is called *direct headers announcement*.

Micropayment channel. Micropayment channel in Bitcoin allows users to make multiple blockchain transactions without committing all those transactions to the ledger. Lightning network [4] implement the Hashed Time-Locked Contract (HTLC), which is an implementation of two-party micropayment channel, to leverage fast off-chain transactions. The two parties can repeatedly update the state of the contract to perform transactions on small amounts, rather than publish each transaction on the ledger. In our design, we also apply HTLC to leverage frequent interactions between uploaders and downloaders.

III. Related Work

Bitcoin P2P network. Blockchain systems have special P2P protocols to leverage message propagation, which has been analyzed through real-world data [14, 11]. [14] performs thorough data collection and analysis on Bitcoin's P2P network in 2014 and provides lots of interesting statistic results related to block propagation. The authors observed the significant propagation latency which may cause miners to mine dated blocks and waste computing power. [11] analyzed the Bitcoin broadcast protocol in details and proved that propagation latency is the main cause of chain forks. Also, the paper proposed some modifications on the broadcast protocol to speed up block propagation.

Our work not only further improve the efficiency of block broadcasting but also integrate an incentive mechanism for block relaying.

Incentive for P2P content delivery. When P2P digital content flows faster as the scale of systems grows, the content providers face the limitation of bandwidth and therefore have lower motivation to provide data. To overcome the lack of

incentive for providing data in P2P systems, [20] integrates financial rewards into data transfer.

Also, blockchain provides a practical tool leveraging distributed financial exchange and blockchain-based incentive in decentralized applications is explored by several works [18, 19, 22, 23, 24, 25]. In the above systems, Blockchain plays the role as a transparent and reliable ledger which confirms the truthfulness of incentive mechanisms in a decentralized environment. Most of those systems also use the cryptocurrency or other blockchain-based tokens as the direct incentive.

In our work, we address block broadcasting in blockchain systems as a specific case of P2P content delivery and integrates a blockchain-based incentive model.

IV. Model Formulation

In this section, we present a formulated model of blockchain's block broadcasting in Section IV-A. We also address two main challenges in Section IV-B and specify our design goals as potential solutions to challenges in Section IV-C.

A. Model of Block Broadcasting

First, we introduce some conceptions in the process of block broadcasting. We then define a formulated model of blockchain message propagation.

Node. The node is the basic unit of synchronization in blockchain systems. Each node manages a local state of chain and tries to keep updating the state to the latest. Each node possesses multiple blockchain identities (aka address). Especially, only full nodes are able to generate new blocks. Our model focuses on full nodes.

Block. The block is one of basic data structures in blockchain. Blocks are generated by miners in the system, containing a set of transactions. The block usually has a hard-coded limitation of size, such as 1MB for Bitcoin in 2010. Each block has a header, which is a serialized piece of data containing crucial properties of a block, such as previous block hash, the nonce and the Merkle root. For block b, we denote by $b.header$ as header part and $b.body$ as the main body containing transactions.

P2P network. Blockchain's P2P network is consist of homogeneous nodes. Each node has a few neighbors with whom the node exchange messages to achieve synchronization. Nodes are supposed to keep as few neighbors as possible to save communication bandwidth, as long as successful synchronization is guaranteed.

Next, we introduce some notations to formally model blockchain's message propagation. We denote by $G = \langle P, \mathcal{E} \rangle$ as the topology of blockchain P2P network where P is the set of nodes and $\mathcal{E}(p)$, $p \in P$ presents the set of neighbors of the node p. For any $e \in \mathcal{E}(p)$, $e \in P$. In addition, we denote by $R = \langle G, p, m, \mathcal{R} \rangle$ as one blockchain message propagation process, in which G denotes the topology of a blockchain P2P network, $p \in G.P$ presents the generator of the message m and $\mathcal{R}$ denotes a broadcast protocol. In real-world blockchain applications like Bitcoin, the message m should be transactions or blocks, which is required to be propagated to all nodes. Since the data of blocks is in a much larger size than single transactions, which can affect the throughput of P2P propagation more significantly, our proposed broadcast protocol focuses on the propagation of newly generated blocks.

Using the above notations, we formally describe the process of blockchain's block broadcasting. We assume the P2P network G is set up and all nodes in $G.P$ reach a consensus on a broadcast protocol $\mathcal{R}$ before a broadcast B starts. Note that the topology G must be a connected graph, which means each node must have at least one neighbor in function to exchange messages, or formally declared as $p \in G.P$, $\mathcal{E}(p) \neq \varnothing$. Once a node, denoted by p_0, successfully generate a new block b, a blockchain message propagation R takes place with $R.p = p_0$ and $R.m = b$. Broadcast protocol $\mathcal{R}$ defines the behavior of broadcast, which is triggered by the arrival of a message, for all nodes. The typical behavior has three parts:

- Checks on the message. First, the node checks whether the message is already available in his local storage. If so, there is no need to process the message further. Otherwise, the node verifies the integrity of the message. If the message is a block, for example, the node checks the validity of block header and transactions.
- Process the message. After verification passes, the node updates the status of his local chain according to the message. For example, add new blocks onto the chain graph or put new transactions into the transaction pool.
- Propagate the message. The node propagates the original message to all neighbors, except for the message source, so that invokes the same behavior of his neighbors.

A synchronization starts at the generation of m and it is successful when all honest nodes have processed m. Especially, the message m refers to a block in block broadcasting.

B. Challenges

According to the trend of higher throughput and scalability of Blockchain systems, we conduct that the Blockchain P2P network will be overloaded as mentioned in Section I. The overloaded P2P network causes two challenges: *long synchronization latency* and *absence of incentive for broadcasting*.

Long synchronization latency. *Long synchronization latency*, which means a long time for the network to reach a consensus on one message. In a typical Blockchain system like Bitcoin, long synchronization latency may be caused by: 1) *Larger block size.* As an empirical experiment [11] shows, the larger the blocks are, the longer time a point-to-point transfer takes so that the synchronization takes longer. 2) *Number of full nodes.* If the scale of the P2P network is large, a message takes longer to reach all nodes.

Unfortunately, in order to improve blockchain's throughput and scalability, current blockchain systems trend to support larger blocks and more nodes and that is why *long synchronization latency* will emerge as a problem. The problem causes several unsatisfying results and potentially corrupts blockchain's liveness and consistency. For example: 1) *Waste of mining power*. Miners are supposed to mine after the best

chain (longest chain in Bitcoin). If a new block is late to be observed and synchronized, the miner waste more time mining on the previous block without contributing to the best chain. 2) *Soft forks*. When more miners are mining on previous blocks because the message of the latest block does not arrive, there will be more soft forks on the previous blocks. 3) *Selfish mining attack*. The basic idea of selfish mining is to develop a private chain containing more than one block and broadcast multiple blocks at one time [16]. Different from normal miners, selfish miner avoids synchronization latency. As a result, selfish miner develops his private chain faster and demonstrates more mining power as he should have.

Absence of incentive for broadcasting. The communication cost of block broadcasting will be significant as Blockchain's throughput increases. but in conventional Blockchain P2P network layer, there is no direct incentive to incentivize nodes propagate the messages they received. As a result, rational nodes have the motivation to refuse to relay blocks so that the efficiency of synchronization is further damaged.

C. Design Goals

To address two challenges of Blockchain P2P network, we declare design goals of the block broadcasting scheme from three aspects: *fast synchronization*, *incentive for broadcasting* as well as *fairness in propagation process*.

1) *Fast synchronization*. The new block broadcasting scheme should have better synchronization efficiency. In other words, it requires less time to broadcast a new block to honest nodes in the system. The system should be feasible to implement and stable to use on large scale.
2) *Incentive for broadcasting*. Under the incentive mechanism, nodes are encouraged to relay blocks they received. The system should have a stable equilibrium on which a majority of system participants would like to follow broadcast protocol honestly and actively.
3) *Fairness in propagation process*. It is hard to guarantee strict fairness of interactions between nodes without Trusted Third Party (TTP) but the unfairness remained in the system should be minimized by incentive. Also, the unfairness should not affect the normal execution of system protocol and incentive mechanism.

V. Block Broadcasting Protocol

In this section, we elaborately illustrate our proposed protocol of block broadcasting. The key idea of our protocol is slicing blocks into pieces which are transferred and verified independently. Along with the transfer of each piece, payments are also processed through micro-payment channels.

The illustration of the protocol is based on Bitcoin but the protocol is suitable for other block-based blockchain systems and fits in the model of blockchain's block broadcasting mentioned in Section IV-A. When a block flows between two nodes, we name the sender and the receiver of the block data as *uploader* and *downloader*, respectively.

We introduce related data structures and tools in Section V-A, the block sharding method in Section V-B, P2P messages in Section V-C and the overall workflow in Section V-D.

A. Related Tools

We first introduce some tools used in protocol design, including *micro-payment channel*, *Merkle Tree* and *bloom filter*.

Micro-payment channel. We define the functions of micro-payment channel [10] between two parties, which is a crucial application for our protocol to leverage financial incentive practically.

- $\mathcal{C} \leftarrow Setup(A, B, s)$: Two parties A and B setup the channel and allocate an initial distribution of funds s. For example, initially A and B each owns 0.5 BTC and the total of fund is 1 BTC;
- $\mathcal{C}' \leftarrow Transfer(\mathcal{C}, s')$: When a transfer of fund happens, both parties perform a serial of interactions to update the state of the channel to assign a new fund distribution. For example, the new fund distribution $s' = (A : 0.4, B : 0.6)$ is assigned when A transfer 0.1 BTC to B.
- $Terminate(\mathcal{C})$: Terminate the micro-payment channel by submitting the final fund distribution to the ledger through one transaction.

$Transfer$ method can be called by each side and for multiple times as long as the channel is in function, without changing the status of blockchain ledger. Therefore, $Transfer$ is cheap to perform so that the micro-payment channel makes the frequent transfer of cryptocurrency scalable.

Merkle Tree. Merkle Tree provides an approach of effective and secure verification on a large data structure. Note that $Hash$ is a predefined safe hashing function. We adopt Merkle Tree to perform the verification on a single data piece, described in Section V-B.

- $(r, \mathcal{P}) \leftarrow GenMerkle(X)$: Generate a Merkle Tree taking a set of inputs X as leaves and return Merkle root r and a mapping function $\mathcal{P}$. $\mathcal{P}(x), x \in X$ presents the corresponding path of the Merkle Tree from root r to the leave x.
- $\{0, 1\} \leftarrow VrfyMerkle(x, r, p)$: Verify x is a valid leave in the Merkle Tree given Merkle root r and the path of x as p ($p \leftarrow \mathcal{P}(x)$).

Bloom filter. Suppose there is a large set with a size of M. *Bloom filter* is used to test whether an element is among the M elements in the set, but the bloom filter here is a M'-length vector of bits and M' could be much smaller than M.

- $f' \leftarrow Add(f, x)$: Use an element x to update bloom filter f;
- $\{0, 1\} \leftarrow Test(f, x)$: Test whether x has been used as an input to update f.

Similarly, if the blocks are sliced to M pieces at most, uploaders can use the data structure of the bloom filter to inform downloaders which pieces are available. In our protocol, x is the index of a specific piece. Note that $Test$ has false positives, whose probability depends on the ratio of M and M', but false negatives are impossible. We name the vector f as *vector of possession* in our protocol.

B. Block Sharding

In our block broadcasting protocol, for each block b, $b.body$ is sliced to multiple pieces. Given a sharding method $\mathcal{S}$, the set of pieces is formally expressed as $\mathcal{S}(b) = \{b_0, b_1, \ldots, b_n\}$. The benefits of the sharding are straightforward. Similar to P2P content delivery or file sharing, transferring data in pieces is much faster in the P2P network, especially for large files. In the P2P environment, the multiple pieces can have various sources to download so that the method bypasses the limitation of uploaders' bandwidth.

But different from most P2P applications, data in broadcasting, such as transactions and blocks, is crucial for blockchain consensus and therefore requires more strict verification. We intend to make the transfer of each piece independently verifiable with the minimum communication cost. To achieve this, we adopt a Merkle Tree proof on the pieces.

We add another block property named as *piece root* into the block headers. The *piece root*, as the name indicates, is a Merkle root from a Merkle Tree where each leaf is the hash of one piece. For a block b, the *piece root* $b.header.pr$ is generated by $(b.header.pr, \mathcal{P}) \leftarrow GenMerkle(\{Hash(p_i) \mid p_i \in \mathcal{S}(b)\})$. Before the uploader transfers any piece of b to the downloader, downloader should have received and verified $b.header$ and therefore possess a valid *piece root*, as Section V-D depicts. When transferring a piece $p_t \in \mathcal{S}(b)$, the uploader offers a Merkle path $path_{p_t} \leftarrow \mathcal{P}(Hash(p_t))$ along with the piece data to the downloader. After receving p_t and $path_{p_t}$, the downloader runs $VrfyMerkle$ to verify p_t's validity.

Since the block header, in Bitcoin, directly participants in mining process (i.e. block generation), under the assumption that $Hash$ performs safe hashing, attackers cannot modify in-header *piece root* at any case. Meanwhile, given a verified *piece root*, the attacker cannot produce any malicious $p'_t \neq p_t$ that passes either the verification of Merkle Tree of pieces.

C. P2P Messages

We define some important P2P messages used in our block broadcasting protocol. Our protocol inherits some messages used in Bitcoin, such as *inv* indicating the discovery of a new item; *getHeaders* requesting block headers; *headers* transferring the data of block headers.

Similar to *getHeaders* and *headers*, we have *getPieces* and *pieces* for requesting and transferring piece data. In details, message *getPieces* contains one block hash ($Hash(b.header)$ for block b) and at least one piece index x ($0 \leq x < |\mathcal{S}(b)|$). Message *pieces* contains at least one piece data $p_x \in \mathcal{S}(b)$ and correlated Merkle path $path_{p_x}$.

Besides, we define message *getPossession* and *possession* to leverage fast queries on *vector of possession* from downloaders to uploaders. Message *getPossession* simply contains one block hash and message *possession* responses the correlated *vector of possession*.

D. Workflow of Block Broadcasting

Given defined tools and P2P messages, we describe the whole workflow of our block broadcasting protocol, focusing on interactions between one uploader and one downloader.

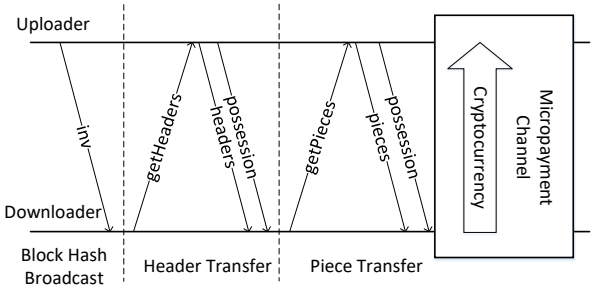


Fig. 1: Standard workflow of block broadcasting protocol. The arrows denote messages between the uploader and the downloader.

As Figure 1 depicts, block broadcasting process between one uploader and one downloader has several interactions. We denote by A as the uploader and B as the downloader for simplicity. We assume all nodes agree on a set of global configurations, such as the size of pieces and the format of messages. To simplify the presentation of the global configuration, we suppose all honest nodes are homogeneous and each piece of block data is in the same size. Also, the size of blocks is limited so that each block is only sliced to a finite number of pieces.

Setup. A and B initialize their micro-payment channel $\mathcal{C}$ through `Setup` and configure their broadcast strategy.

Block hash broadcast. After A receives or generates a valid block header $b.header$, A sends B an inv message containing block hash $h \leftarrow HASH(b.header)$, which declares the identity of block b.

Header transfer. B receives message *inv* and checks whether the block header $b.header$ is valid and does not exist in his storage. If so, B sends back a *getHeaders* message to request the header. In response of *getHeaders* message, A sends back the *headers* message containing complete data of $b.header$. A can optionally send *possession* message to inform b which pieces are available for requesting.

Repeated piece transfer. B randomly selects a piece through $Test$ method on the latest A's possession vector f. If B does not have any vector of possession, he can send a *getPossession* message to request one from A. Then B sends a *getPieces* message which announces the block hash h and the index of a specific piece x.

A sends back *pieces* message containing block hash h, piece data p and Merkle path $path_p$. Also, A can respond to another *possession* message if A wants to update his possession status. Note that A may receive pieces of b simultaneously during interactions with B and A should use new obtained pieces to update the possession vector through Add.

Received expected piece data p and its path $path_p$, B first locally calls $MerkleVrfy(Hash(p), b.header.r, path_p)$ to verify the piece. After a successful verification, B starts a payment to A by calling micropayment channel's function $Transfer$. The function of $Transfer$ requires several times of interactions between the two sides. After both sides achieve agreement on the payment, B is supposed to start another *piece*

transfer if he wants.

Termination The repeated piece transfer can be terminated from any side by sending a special message or just waiting for the timeout. Assuming both sides are honest and follow the protocol, the termination is likely to happen when the downloader has no other pieces to request.

VI. Analysis

In this section, we use game theoretical models to analyze our incentive mechanism. We first deploy an evolutionary model to analyze the efficiency of incentive in broadcasting. The peers should have incentive to become active honest uploaders and downloaders.

However, in the process of piece transfer, both sides have the possibility to cheat the other side, for instance terminating protocol in advance. The downloader, especially, has the chance to refuse paying the uploader. For this potential unfairness, we use a repeated game model to demonstrate how cooperation can take place and whether the threat can affect overall system performance.

A. Evolutionary Model of Incentive

We consider an evolutionary game model of block transfer process. One block transfer means the whole interactions between two nodes during processing one specific block, including header transfer and repeated piece transfer. Each node can act as downloader and uploader simultaneously, and we assume that nodes are strategic for the most profit. Each uploader or downloader has two strategies: cooperate (C) and defect (D), and the transfer is successful only when both sides take strategy C. Also, we assume one successful block transfer produces benefit α for downloader and costs β for the uploader. Meanwhile, the downloader pays π to the uploader.

Cost and payoff. In one single block transfer process, the downloader bears a cost of communication bandwidth and computation power but obtains necessary data for synchronization. So α should be data value minus the costs. In the same way, uploader's cost β includes bandwidth cost and computation cost. Meanwhile, if the opposite side unexpectedly aborts the block transfer process, both downloader and uploader have an extra cost. Downloader must find another uploader while uploader may lose the last payment as described in Section VI-B. The extra costs for downloader and uploader are respectively denoted by t_d and t_u.

$$P = \begin{bmatrix} & C & D \\ C & \alpha-\pi, -\beta+\pi & -t_d, 0 \\ D & 0, -t_u & 0, 0 \end{bmatrix} \tag{1}$$

Matrix P in Equation 1 shows the payoff matrix for one block transfer (P_{ij} denotes payoff when downloader uses strategy i and uploader uses strategy j). In one generation of the evolutionary model, each node plays games with his neighbors, so the distribution of strategy C and D has an important influence on average payoff in one generation. We use x_d denotes the fraction of strategy C among downloaders while x_u denotes the fraction of strategy C in uploaders. Equation 2 shows expected payoff in one generation for each role and each strategy, in which P_i^S denotes payoff for role i (downloader or uploader) with strategy S. Therefore, the total payoff in one generation with for a strategy set $S = (S_d, S_u)$ is $P^S = P_d^{S_d} + P_u^{S_u}$.

$$\begin{cases} P_d^C = x_u(\alpha - \pi + t_d) - t_d \\ P_d^D = 0 \\ P_u^C = x_d(-\beta + \pi + t_u) - t_u \\ P_u^D = 0 \end{cases} \tag{2}$$

Equilibrium Points. From the payoff we list above, we found: 1) When x_u is small, which indicates restricted resource, nodes trend to shift to strategy D as downloader since P_d^C may below 0. Otherwise, cooperation is a better choice. 2) When x_d is small, which indicates inactive downloader group and few profits for uploaders, nodes trend to shift to strategy D as uploader since P_d^C may below 0. Otherwise, continuously providing download service earns more.

To further analyze this model, we use replicate dynamic equations [26]: $\dot{x_i} = x_i[f(x_i) - \Phi(x)]$, $\Phi(x) = \sum_{j=1}^{n} x_j f(x_j)$. x_i denotes proportion of a strategy. f is fitness of a strategy, which equals to payoff analyzed in our model.

$$\begin{cases} \dot{x_d} = x_d(1-x_d)P_d^C \\ \dot{x_u} = x_u(1-x_u)P_u^C \end{cases} \tag{3}$$

To find an Evolutionary Stable Strategy (ESS), the replicator dynamics equation should be equal to 0. Strategy C for uploader is stable only if $x_u = 0, 1$ or $P_u^C = 0$. In the same way, strategy C for downloader is stable when $x_d = 0, 1$ or $P_d^C = 0$. We can use Jacobian matrix 4 to investigate ESS in evolutionary game model. Possible equilibrium points are listed in Table I.

$$J = \begin{bmatrix} (1-2x_d)[x_u(\alpha-\pi+t_d) - t_d], \\ x_d(1-x_d)(\alpha-\pi-t_d); \\ x_u(1-x_u)(-\beta+\lambda\pi+t_u), \\ (1-2x_u)[x_d(-\beta+\lambda\pi+t_u) - t_u] \end{bmatrix} \tag{4}$$

TABLE I: Analysis of equilibrium points

Equilibrium point	$det(J)$	$tr(J)$	result
$x_d = 0\ x_u = 0$	+	-	ESS
$x_d = 0\ x_u = 1$	+	+	Not stable
$x_d = 1\ x_u = 0$	+	+	Not stable
$x_d = 1\ x_u = 1$	+	-	ESS
$x_d = \frac{t_u}{\pi-\beta+t_u}\ x_u = \frac{t_d}{\alpha-\pi+t_d}$	+	0	Saddle point

We paint five equilibrium points $O(0,0)$, $A(0,1)$, $B(1,0)$, $C(1,1)$, $D(x_{d0}, x_{u0})$ in one coordinate plate (Figure 2). From above analysis, the evolutionary game model has two ESS point: (0,0) and (1,1). Point $(\frac{t_u}{\pi-\beta+t_u}, \frac{t_d}{\alpha-\pi+t_d})$, denoted by (x_{d0}, x_{u0}), is the saddle point. If initial state of system locates inside area OADB, system is more likely to converge to O. Otherwise, system has larger probability to evolve to C, which indicates cooperation equilibrium. We notice that free riding (point A) is unstable equilibrium point.

Simulation of evolutionary game. We design a simulation of the evolutionary game (see Algorithm 1), we first initialize parameters and the original topology of the network. Each node plays block transfer games with its neighbors. Then

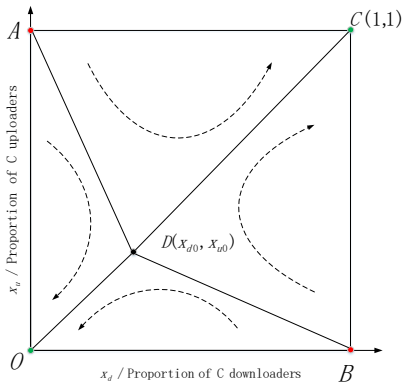


Fig. 2: Diagram of equilibrium points in evolutionary model. Points O, A, B, C and D denotes various possible equilibriums while arrows denotes the path of revolution.

Algorithm 1 Process of evolutionary model

```
Initialize simulation parameters.
Initialize topology graph G = ⟨P, E⟩.
Initialize strategy distribution (x_d, x_u).
loop
  for i ∈ P do
    for j ∈ E(i) do
      node i plays game (block transfer process) with node j.
    end for
  end for
  for i ∈ N do
    Randomly select node j ∈ E(i).
    Compute probability of learning process p_{i→j}.
  end for
  Update strategy of nodes with probability matrix p_{i→j}.
end loop
```

calculate the payoff for each node, run learning process and shift strategies for the next generation.

When nodes learn another node's strategy with a specific probability at the end of each generation, we apply Fermi function [27, 28] as evolutionary updating rule. At the end of each generation, node i learns to follow another node j's strategy with probability $p_{i\to j}$:

$$p_{i\to j} = \frac{1}{1+e^{\omega(P_i-P_j)}}$$

P_i denotes payoff of node i. ω is a selection intensity factor. The larger the ω is, the faster the system evolves.

In conclusion, if parameters are properly set to make (x_{d0}, x_{u0}) close to $(0,0)$ and make sure there are enough proportion of cooperators at the beginning of the system, the whole system will converge to overall cooperation and keep stable in the end. In other words, honest nodes are incentivized to relay blocks honestly and actively. The simulation of Algorithm 1 (Section VII-B) proves this analysis.

B. Repeated Game Model for Piece Transfer

Downloader requests a sequence of pieces from one uploader once the block header is obtained. This procedure is named as *repeated piece transfer*. The procedure can be regarded as a repeated game and one piece is transferred in each stage. The repeated game is finite but the number of stages is not common knowledge for both sides.

One stage consists of at least three steps: *piece request*, *piece response* and *payment*. Both sides take the trigger strategy: quit protocol in the next stage if the other side misbehaves. It accords with the reality that rational nodes will not be cheated twice. In intuition, a greedy downloader can skip *payment* step to get a free piece. Note that downloaders request piece in a randomized order and uploaders cannot predict at which stage the downloader will quit. Therefore the repeated game will satisfy the following conditions (denote by n as the maximum stage of the repeated game):

- In the first t stages, both sides cooperate. $t \in [0, n]$.
- At stage $t+1$, either side quits.
- After stage $t+1$, both sides quit and the protocol is terminated.

Cooperation behavior. In classic game theory, similar to finite repeated prisoner dilemma, such finite repeated game has Subgame Perfect Equilibrium (SPNE) where both sides will not cooperate from the beginning. However, participants are not completely rational. They usually have a belief that the other side will cooperate at the beginning. With the belief, nodes are greedy to take the risk to cooperate longer for better profits. As a result, both sides incline to deviate before the other side but intend to cooperate as much as they can.

There is a classical learning model which models cooperation behavior in finite repeated prisoner dilemma observed in experiments [29]. Above this classic model, we define our learning model of repeated piece transfer game, in which one downloader and one uploader repeatedly play the finite repeated game (piece transfer game), once in one round. Downloader has a random demand λ for each round. Both sides respectively have an intended deviation stage t_d and t_u. Note that if $\lambda < t_d$, downloader deviates in advance. The learning process goes as follows:

- If one side observes that the opponent deviated before he intended to deviate, he has a probability p_1 to shift his intended deviation from t to $t-1$.
- If one side observes that the opponent deviated in the same period as he intended to, he has a probability p_2 to shift his intended deviation from t to $t-1$.
- If one side observes that the opponent hadn't deviate when he intended to deviate, he has a probability p_3 to shift his intended deviation from t to $t+1$.

From this model, we notice that if $t_d = t_u = n$, they all incline to deviate earlier, which represents that downloader wants to cheat for one free piece and uploader want to avoid this. When $t_d < t_u$ downloader terminates the protocol so early that misses more pieces to download. It is same for uploader when $t_d > t_u$ is observed. If they have the belief that the other side wants to cooperate longer, intend to shift their deviation later.

Algorithm 2 Process of repeated game model with learning

Initialize maximum piece count n, topology graph $G = \langle P, \mathcal{E} \rangle$ and probability parameters.
for $i \in P$ **do**
  Randomly initialize expected deviations for node i.
end for
loop
  for $i \in P$ **do**
    Randomly initialize a demand of piece within $[1, n]$.
    Randomly select j from $\mathcal{E}(i)$
    node i plays game (piece transfer process) as downloader with node j.
    Node i and j updates expected deviation as downloader and uploader, respectively.
  end for
end loop

Repeated game with learning. In each round of the repeated game (see Algorithm 2), each node plays repeated piece transfer game as a downloader with a random neighbor, executes learning algorithm and update expected deviation of the node and the neighbor. In this way, end behavior evolves through games between the nodes round by round.

According to our simulation results (Section VII-C), though exchange protocol without TTP can hardly be definitely fair, the attractive rewards for cooperating makes the cooperation possible. The learning model shows an evolution of end behavior, and as a result, nodes can perform stable cooperation, which means their expected deviation is dynamically stable around a not small number, after a period of evolution. Besides, the damage of betraying and fraud is limited in only one piece. So the potential unfairness in piece transfer process cannot affect system incentive and overall performance.

VII. Experiments

We have three parts of the experiments. First, we develop a network simulating environment to evaluate how much our proposed protocol can speed up block synchronization. The second experiment is the simulation of the revolutionary model, which proves that blockchain-based incentive is efficient to encourage active and honest block broadcasting. The third experiment is a simulation of the learning model introduced by Section VI-B, aiming to see whether nodes can achieve cooperation in repeated piece transfer.

A. Simulation on Synchronization Efficiency

We simulate a typical P2P network using Network Simulator 3 (NS3). The network contains 1000 nodes and 5000 links among nodes (10 neighbors for each node on average). Each link is assigned a predefined link delay randomly with an average of 100 ms. Also, we set the packet error rate to 0.001% and bandwidth to 70 Mbps, according to an investigation [30]. We simulate both Bitcoin's standard block broadcasting and our proposed block broadcasting. In each execution of the simulation, we use different size of pieces, generate a 4 MB block on one node, propagate the block to the whole network and record the latency of synchronization.

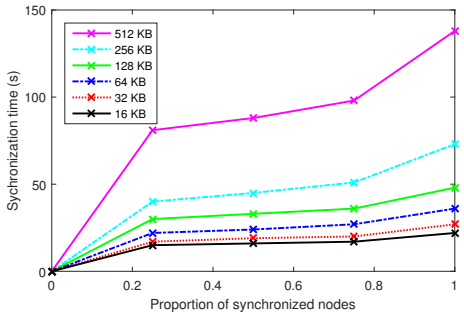


Fig. 3: Synchronization time with various piece size.

TABLE II: Parameters for simulation of system incentive

Parameter	Description	Value
$\|N\|$	Count of nodes	1000
ω	Learning coefficient	0.1
α	Benefit for downloader	1.6
β	Cost for uploader	1
π	Payment from downloader to uploader	1.4
t_d	Downloader cost	0.05
t_u	Uploader cost	0.10

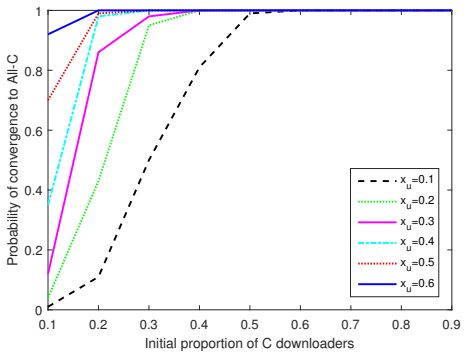
Figure 3 shows a strong correlation between synchronization efficiency and piece size. We use Bitcoin's standard block broadcast as the baseline, which spends 571s, 653s, 729s, 930s to broadcast a 4MB block to 25%, 50%, 75% and 100% nodes, respectively. Compared with the baseline, block sharding can speed up the synchronization for 30 times and shorten the time cost by over 90%. From the simulation, the piece size of 16KB or 32KB is a wise choice for 4MB block broadcasting.

B. Simulation of System Incentive

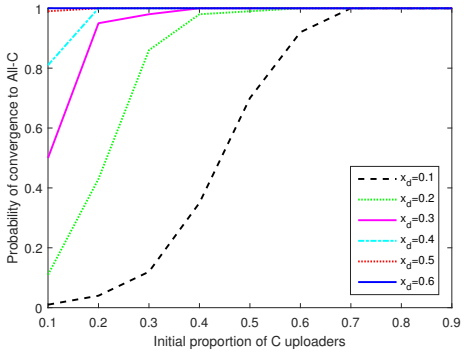
We simulate the evolutionary model defined in Section VI-A using Algorithm 1. We use the same configures about network topology as that in Section VII-A.

Simulation parameters. The values of parameter α, β, t_d and t_u depend on reality. First, we set $\beta = 1$ as a standard. We estimated β much higher than t_u and t_d because one single failed block transfer will not hurt synchronization when the network topology is dense. Payment π should be larger than β and lower than α. We list the parameters in Table II. Assuming blockchain network is homogeneous, we apply the same parameters for all nodes.

Results. The simulation results are consistent with the analysis in Section VI-A. Figure 4 shows that the larger x_d and x_u are, the easier convergence to successful All-C equilibrium is. Also, we can conduct the saddle point showing about 50 % successful convergence to be around $(0.2, 0.2)$, which is close to the prediction given by Table I. Under our parameter setting, if the system has a significant proportion of cooperative nodes, 0.3 for instance, it has over 95% probability to evolve to the status where all nodes are cooperative for block broadcasting.

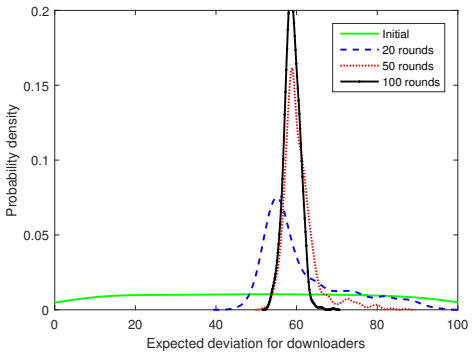


(a) Relationship between probability of successful convergence and x_d.

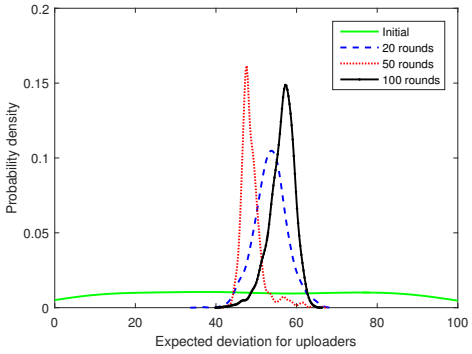


(b) Relationship between probability of successful convergence and x_u.

Fig. 4: Simulation results of evolutionary model on block transfer process.



(a) Probability density of expected deviation for downloaders.



(b) Probability density of expected deviation for uploaders.

Fig. 5: Simulation results of repeated game model on piece transfer process. The curves present the distribution of expected deviation after various number of rounds.

TABLE III: Parameters for simulation of repeated piece transfer

Parameter	Description	Value
n	Maxinum number of pieces	100
p_{1d}, p_{1u}	Probability parameter of learning model	0.3-0.5, 0.4-0.6
p_{2d}, p_{2u}	Probability parameter of learning model	0.2-0.4, 0.1-0.3
p_{3d}, p_{3u}	Probability parameter of learning model	0.6-0.8, 0.6-0.8

C. Simulation of Repeated Piece Transfer

We simulate the repeated piece transfer in Section VI-B with learning process Algorithm 2. We still use the network topology in Section VII-A.

Parameters. We use the parameters in table III for learning algorithm described in Section VI-B. It is reasonable to assume probability parameters $p_3 > p_1 > p_2$ because nodes are greedy for more profits and incline to last cooperation longer. Respectively, p_{1d}, p_{2d}, p_{3d} denotes the parameters for downloaders and p_{1u}, p_{2u}, p_{3u} is for uploaders. Since nodes have various characteristics, we use different random values within a range as the probability parameters for different nodes. Also, the demand for pieces for each node is randomly selected within the range of the maximum number of pieces.

Results. If both sides are greedy enough (p_{3d} and p_{3u} are large enough), the finite repeated game will reach a dynamic balance point. After 100 rounds of simulation, a significant proportion of nodes have shifted their intended deviation closer to a balance point (around 60). This result represents the cooperation behavior of repeated games. Though the game doesn't have a theoretical equilibrium, cooperation exists when nodes are not completely rational. For example, nodes have a belief in others' cooperation behaviors at the beginning. Also, nodes are greedy for more profits so that the high payoff of cooperation encourages them to cooperate. Therefore, unfairness in piece transfer process will not destruct overall cooperation.

VIII. Conclusion

In this paper, we proposed an improved block broadcasting scheme. We use game theoretical models to analyze the proposed incentive mechanism. Then we carry out experiments to simulate the analysis. As the result shows, the payment mechanism between downloaders and uploaders incentives active broadcasting. Also, the data transfer protocol is almost fair since the potential unfairness is limited in only one piece of the file and this will not affect overall system performance. What is more important, the simulation shows that our proposed block broadcasting scheme can effectively speed up synchronization by over 90% in typical Blockchain P2P network.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 2008.

[2] Marc Pilkington. Blockchain technology: Principles and applications. *Social Science Electronic Publishing*, 2015.

[3] Bitcoin improvement proposals. `https://github.com/bitcoin/bips`, . Accessed December 10, 2018.

[4] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *See https://lightning. network/lightning-network-paper. pdf*, 2016.

[5] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016. `https://byteball.org/Byteball.pdf`.

[6] The zilliqa technical whitepaper, 2017. `https://docs.zilliqa.com/whitepaper.pdf`.

[7] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[8] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[9] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.

[10] Bitcoin - open source p2p money. `https://bitcoin.org`, . Accessed December 10, 2018.

[11] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[12] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 281–310, 2015.

[13] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 291–323, 2017.

[14] Joan Antoni Donet Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. The bitcoin p2p network. In *International Conference on Financial Cryptography and Data Security*, pages 87–102. Springer, 2014.

[15] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[16] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[17] Jin Li. On peer-to-peer (p2p) content delivery. *Peer-to-Peer Networking and Applications*, 1(1):45–63, 2008.

[18] Yunhua He, Hong Li, Xiuzhen Cheng, Yan Liu, Chao Yang, and Limin Sun. A blockchain based truthful incentive mechanism for distributed p2p applications. *IEEE Access*, 6:27324–27335, 2018.

[19] Jingzhong Wang, Mengru Li, Yunhua He, Hong Li, Ke Xiao, and Chao Wang. A blockchain based privacy-preserving incentive mechanism in crowdsensing applications. *IEEE Access*, 6:17545–17556, 2018.

[20] Srijith K Nair, Erik Zentveld, Bruno Crispo, and Andrew S Tanenbaum. Floodgate: A micropayment incentivized p2p content delivery network. In *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, pages 1–7. IEEE, 2008.

[21] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[22] Bing Jia, Tao Zhou, Wuyungerile Li, Zhenchang Liu, and Jiantao Zhang. A blockchain-based location privacy protection incentive mechanism in crowd sensing networks. *Sensors*, 18(11):3894, 2018.

[23] Qingsu He, Yu Xu, Yong Yan, Junsheng Wang, Qingzhi Han, and Lili Li. A consensus and incentive program for charging piles based on consortium blockchain. *CSEE Journal of Power and Energy Systems*, 4(4):452–458, 2018.

[24] J Weng, Jian Weng, J Zhang, M Li, Y Zhang, and W Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. Technical report, Cryptology ePrint Archive, Report 2018/679. 2018. Available online: https …, 2018.

[25] Yongjun Ren, Yepeng Liu, Sai Ji, Arun Kumar Sangaiah, and Jin Wang. Incentive mechanism of data storage based on blockchain for wireless sensor networks. *Mobile Information Systems*, 2018, 2018.

[26] Josef Hofbauer and Karl Sigmund. *Evolutionary games and population dynamics /*. Cambridge University Press,, 1998.

[27] Arne Traulsen, Martin A Nowak, and Jorge M Pacheco. Stochastic dynamics of invasion and fixation. *Physical Review E*, 74(1):011909, 2006.

[28] Philipp M Altrock and Arne Traulsen. Deterministic evolutionary game dynamics in finite populations. *Physical Review E*, 80(1):011909, 2009.

[29] Reinhard Selten and Rolf Stoecker. End behavior in sequences of finite prisoner's dilemma supergames a learning theory approach. *Journal of Economic Behavior & Organization*, 7(1):47–70, 1986.

[30] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. 2018.