Enforcing Fairness in Blockchain Transaction Ordering

Ariel Orda and Ori Rottenstreich

Technion and Orbs (www.orbs.com)

***Abstract*—In Blockchain networks involving multiple applications, the quality of service of an application is affected by the transaction ordering. We study a setting where each application is represented by a node, which might attempt to prioritize its own transactions through including them early in blocks added to the blockchain. A fair block proposal of a node follows a random selection of the transactions among the set of pending transactions the node is aware of. On the contrary, a dishonest node includes more of its transactions at the expense of transactions of other applications. In this work, we propose a toolbox of techniques to enforce such a fair block selection. First, we design an accurate statistical testing for the honesty of a proposal and explain it. Next, we describe a reputation system, documenting honesty of nodes to encourage fairness. Our last technique enforces fair block selection through concise commitments on the set of pending transactions known to a node.**

I. Introduction

Blockchain is a growing list of records (often called *transactions*), managed in a distributed manner among multiple participants. Transactions can either be simple money transfers or some more general pieces of code such as Ethereum smart contracts [1]. The blockchain is organized in *blocks*, each composed of multiple transactions. Typically, participants reach an agreement on the blockchain content through an agreed-upon addition of a new block. The new block is determined as a selection of transactions among the pool of pending transactions, namely transactions that have been issued by one of the participants but do not yet appear in the blockchain. The block selection process implies an order on the transactions. In addition to the agreement on the block order, a consensus is also required with regards to implied updates to the state of the blockchain, namely either account balances or memory values accessed by smart contracts.

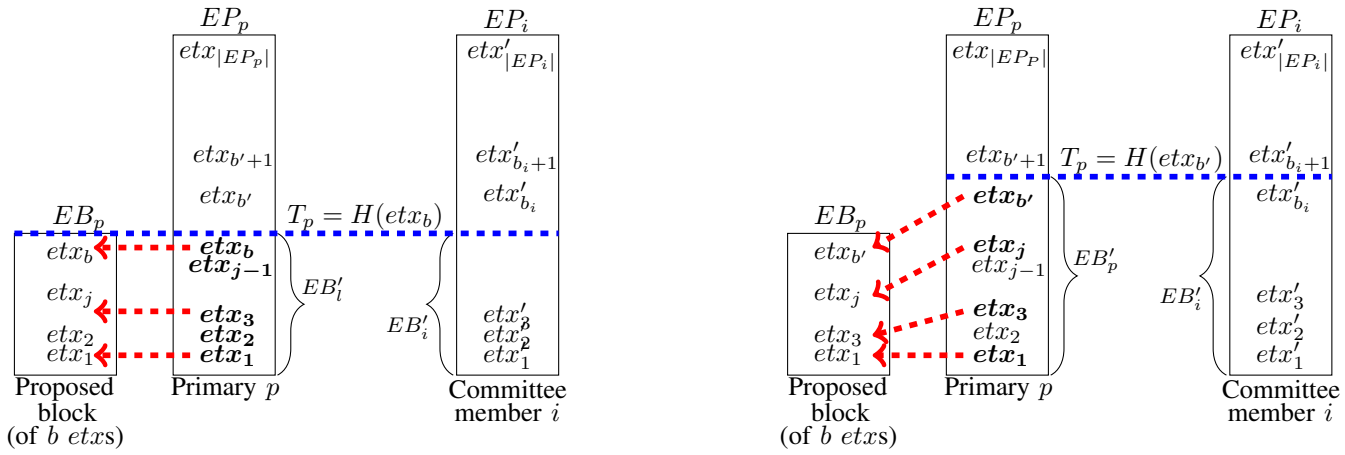
While in some networks the block is jointly determined by multiple participants (e.g., HoneyBadger [2]), typically a block is proposed by a selected node. A node might have a complete freedom in the selection of the transactions blocks (e.g., as in Bitcoin [3] and Ethereum [1], where a miner can select those of highest fees, thus maximizing its profit). A restriction on the ability of a node to consistently manipulate the ordering follows from the entropy in the selection of the proposing node (either uniformly at random, based on computational power in Proof of Work (PoW) [4] or according to one's balance in Proof of Stake (PoS) [5]).

Another approach restricts the freedom in block selection by implying a review process for the block selection by other participants [6]. These nodes, often organized as a *committee*, can validate the selection according to some criteria required from the selection, such that each node indicates whether or not to accept the proposal. A participant not following the required criteria from the block selection takes the risk that its block proposal would be rejected and another participant would be selected to propose an alternative block. Moreover, incentives might be used to encourage nodes to avoid manipulations in the block selection. When there is a particular node allowed to present a block proposal we refer to that node as the *primary*.

A recently suggested protocol, named *Helix* [7], suggested a concrete method for the block selection that the primary is expected to follow for implementing a random block selection. The primary should simply sort its local pool of pending transactions according to a sorting function that changes for every round of block proposal. Then, given a maximal block size of b transactions, the primary should simply include in the proposed block the b transactions with the lowest ranking based on the computed order.

An inherent challenge in validating a block proposal follows directly from the nature of distributed blockchain networks. While different nodes (typically) agree on the content of the blockchain, they are not fully aware of all pending transactions, such that nodes are often exposed to non-identical sets of pending transactions. This makes it hard, or indeed impossible, for a validator to simply reject a proposal when it does not include a transaction that by the view of the validator was expected to be included. The validator cannot clearly indicate that the primary was aware of that transaction and ignored it on purpose in order to serve other transactions it prioritizes.

Helix describes a statistical test to examine whether a proposed block followed these instructions. It relies on a (simplifying) model where a node has a fixed probability to be aware of some pending transaction. For a given transaction, this event is independent for the various nodes. In Helix a committee member examines some level of similarity between the proposed block and the locally computed one (while making use of information from the actual block proposal). This helps the committee member to make a *binary* decision whether to accept the proposal or not. A minimal number of accepting nodes among the committee members is required for the block proposal for being approved and added to the blockchain. Helix does not compute the probability that, based on the validation process, a proposal was made in a honest



(a) Fair block selection

(b) Unfair block selection

Fig. 1. Illustration of the block selection scheme (as of Helix). In each Epool, etxs are sorted based on their hash values. A honest primary fairly constructs a block from the b transactions with the minimal hash value. An unfair block selection includes skipping transactions, selecting the b block transactions among some $b' > b$ transactions with minimal hash values. A committee member i examines the block proposal by computing the overlap between the proposed block EB_p and the set of etxs in a block computed locally based on EP_i.

way. Indeed, such a computation seems challenging.

Our first contribution is an improvement of the specific scheme of Helix. Specifically, we explain that, following the statistical model for transaction dissemination, the validation process in Helix does not fully utilize the information of the committee members. We show that a more accurate decision should be a *joint decision* of the various nodes rather than simply being based on the number of independent approvals among the members. We suggest how to determine the validity of the proposal based on the aggregated information from the members. We describe a simple formula for the probability that a proposal is honest following the complete information (unlike the difficulty to do so given the independent decisions of Helix with partial information).

Our second contribution is a generic scheme that enhances the effectiveness of committee-voting schemes at large. Specifically, to provide incentives for the nodes to be honest and behave as they are expected, we suggest a *reputation system*. We describe how such a system can take into account block proposals based on the outcome of their validation process. The system can also take into account other factors, such as evaluation on the accuracy of nodes' indications while serving as committee members.

Our third contribution takes a different approach and enforces honesty upfront. Specifically, we establish a technique that effectively eliminates nodes from the option to ignore any transactions for increasing the number of prioritized transactions in a proposed blocks. We explain that a periodic *concise* report of the set of known transactions to a node can be useful towards such a goal. The technique relies on the observation that since the transaction sorting cannot be predicted, a node is not aware in advance of the specific transactions it would like to ignore in a particular round. The techniques we describe make use of various data structures such as the Bloom filter [8] or Merkle trees [9] and their variants. Hashing is a useful technique in such data structures, for mapping elements to areas of the report as well as for providing a signature for a reported element.

The rest of the paper is organized as follows. Section II overviews the block proposal and validation procedure in Helix. Then, Section III details the suggested alternative validation process of a block proposal. Next, Section IV describes the node reputation system. Then, Section V presents the technique to enhance fairness in block proposals through node declarations regarding their pending transactions. Finally, concluding remarks are presented in Section VI.

II. Background - Block Validation in Helix

In this section we provide an overview of the block selection and its validation process in Helix. A block is selected by a primary p from the pool of pending transactions it is aware of EP_p. In such a pool, transactions are maintained in an encrypted form. Due to network latency, the pools EP_i and EP_j of two different nodes i, j might differ. However, we may assume a measure of similarity between two pools of pending transactions. To model this similarity, we use a probabilistic model satisfying the following property. For any two correct nodes i, j each etx in EP_i is in EP_j with probability at least α. We refer to α as the *similarity parameter* of the network.

The Helix block selection scheme uses a hash function in order to serialize candidate etxs for the next block. The hash function is tweaked with a random seed RS to eliminate its predictability, yielding a common, random and unpredictable serialization of the etxs. The random seed is a function of the content the block from the previous round. Till a block is selected, transactions appear in an encrypted form such that it is difficult to predict the random seed from the transactions following their decryption. Formally, when considering the

block in term r, the nodes order the pending etxs according to the values $H(RS^{r-1}, etx)$, and refer to these values as the *hash values* of the etxs. We use the notation $H(etx)$ for brevity.

Let EB_p be a block proposed by the primary p and T_p be the maximal hash of an etx in EB_p, i.e., $T_p := \max\{H(etx)|etx \in EB_p\}$. Likewise, denote by $EB'_i := \{etx \in EP_i|H(etx) \leq T_p\}$ the set of etxs in EP_i with hash values lower than T_p, and $b_i := \max\{|EB'_i|, b\}$. We further denote by b' the size of $EB'_p = \{etx \in EP_p|H(etx) \leq T_p\}$ and say that EB_p was constructed under a b'-construction. This illustrates the fact that EB_p was selected as a subset of size b among the b' lowest hashed etxs in EP_p such that the b'^{th} etx was included. The setting is illustrated in Fig. 1.

Under these notations, the validation checks (in the context of selection fairness) performed by a committee member i upon receiving a proposed block, EB_p (from primary p), are:

1) $|EB_p| = b$
2) $|EB_p \cap EB'_i| \geq \beta_\alpha(b_i)$ for $\beta_\alpha(b_i) := \alpha b_i - \sqrt{10b_i}$

The second condition encourages primaries to construct blocks with low b'. The minimal value of b' is b; in the event that the value of b' is in fact b, the selection scheme is perfectly fair. Intuitively, a larger b' allows the primary more freedom in the selection of EB_p (rather than selecting it as the b minimal etxs). However, since we can expect $|EB'_p| \approx |EB'_i|$, large b' yields large b_i and accordingly large $\beta_\alpha(b_i)$, reducing the chances of EB_p to pass validation. $\beta_\alpha(b_i)$ is the maximal value for which blocks constructed with $b' = b$ pass validation w.o.p., as implied by Hoeffding's bound.

The extra validation process dictated by the Helix block selection scheme bears risk to the liveness of the protocol. Blocks that would have passed validation might get rejected once the statistical validation is enforced. It was shown that w.o.p. a block compliant with the b-construction passes validation of a committee member that follows the protocol.

Property 1 (Liveness under b-construction.)**.** *Let EB_p be a block constructed according to the b-construction, and let i be a committee member following the protocol. Then, EB_p passes i's validation w.o.p. (under the assumption that α bounds from below the similarity parameter of the network).*

The property follows from the correctness of the following lemma.

Lemma 1. *For two correct nodes k, l and a general set $A \subset EP_l$, $|A \cap EP_k| > \beta_\alpha(|A|)$ with probability greater than $1 - 2\exp(-20)$, where $\beta_\alpha(x) = \alpha x - \sqrt{10x}$.*

III. An Alternative Joint Block Validation

In this section we present an improvement of the Helix scheme. As summarized in Section II, the block validation of Helix relies on an independent evaluation of the proposed block by each of the committee members. A committee member votes in favor of the proposal when the conditional probability for the block proposal to be fair is above a required lower bound. The probability is computed based on the content of the transaction pool of the committee member. A minimal number of votes in favor of the proposal are required for the block to be approved but finding the optimal number of such required number seems difficult and is not answered in Helix. On the positive side, this scheme does not require communication between the committee members earlier to their voting.

In this section, we explain that such voting criteria does not utilize all information available by the committee members. We follow the assumption of Helix, namely that an issued transaction has the a fixed probability α to appear in a pool of pending transactions by each node, such that for a given transaction this probability is independent among the various nodes. Accordingly, a block proposal of a honest primary should include, with probability α, a transaction issued by other nodes satisfying a bound on its hash value. On the other hand, a proposal of a dishonest primary would be selective and its number of such included transactions is expected to be lower than that implied by such a distribution. We point out an alternative and more accurate validation process than that proposed for Helix. The process requires communication among the committee members earlier to a *joint* indication regarding the honesty of the block proposal.

Intuitively, the evaluation of the block proposal in Helix decreases *by each committee member* for each transaction the primary could be aware of but did not include in the proposal. Consider for simplicity the case where a block proposal of the primary p is evaluated by two committee members i, j. Following the definition of the probability α and its assumed model, we claim that the evaluation should make a distinction between the two following cases: *Case I* - Node i holds a single transaction etx_1 that was not included in the block although being expected to and another node j also holds a transaction $etx_2 \neq etx_1$ with the same property. *Case II* - Both nodes i, j hold a single identical transaction etx_3 not included in the block although being expected to. The reason a distinction should be made is the different probabilities for the scenarios to occur with a honest primary given the existence of the transactions by the committee members. While in Case I, this happens w.p. of $(1-\alpha)^2$, in Case II the probability for that is larger and equals $1 - \alpha$.

However, the evaluation in Helix by each committee member does not make any distinction based on the identity of the missing transactions and accordingly cannot distinguish between such cases since in both each committee member observed the missing of a single transaction. We propose an alternative approach, according to which a right evaluation should be based on examining the ratio of included transactions among those expected given the content of the local pools among the committee members. We show that the related probability of the primary being honest is affected by the identity of missing transactions in the block according to each committee member and not just by their number by each of the members. Namely, the probability is a function of the total *number of distinct missing transactions* rather than their sum among the committee members.

More specifically, the approval decision should determine whether the selection is fair. Thus, each transaction that has to be included should contribute equally to that decision, either positively if it is indeed included, or negatively if it is not included. This can be computed by the number of expected to be included transactions and the number of those among them that are indeed included. Determining these numbers exactly, should ignore multiplicities of the same transaction among nodes and thus requires communication among them. Accordingly, the joint decision for the nodes is made based on these numbers. Unlike the scheme of Helix, there is no notion of a block approval by a single committee member and thus computing the minimal required number of such nodes is not necessary.

We proceed to analyze the probability for a particular scenario based on the above mentioned transaction numbers. Again, let EB_p be a block (of size $|EB_p| = b$) proposed by the primary p such that $T_p = \max\{H(etx)|etx \in EB_p\}$ is the maximal hash of an etx in EB_p. For a committee member i we denote by $EB'_i := \{etx \in EP_i | H(etx) \leq T_p\}$ the set of etxs in EP_i with hash values lower than T_p and also denote $b_i := \max\{|EB'_i|, b\}$. Let $EB' = \bigcup_i EB'_i$ (where the union is computed over the committee members) be the set of all transactions that the the committee members are aware of and should be included in the block as implied by T_p. Denote $b' = |EB'|$. Let $EB' \cap EB_p$ be the set of transactions among those expected that are indeed included in the block and let K be the random variable for their number. For a honest primary, K should follow a binomial distribution (α, EB') such that $Pr(K = k) = \binom{b'}{k}\alpha^k(1-\alpha)^{b'-k}$ and a probability for a honest node that an intersection of size k or less appears is given by $\Phi_\alpha^{k,b'} = \sum_{m=0}^{k}\binom{b'}{m}\alpha^m(1-\alpha)^{b'-m}$. The joint decision of the committee members should then be to accept the block proposal whenever the computed probability satisfies some required lower bound. The bound is selected as a tradeoff between the liveness and the probability to reject a proposal of a dishonest primary. A lower bound Φ_{min} guarantees that a proposal of a honest leader is accepted with at least such probability.

IV. Reputation System

A. Motivation

In this section we construct a generic scheme that enhances the effectiveness of committee-voting schemes at large. As already noted, in typical blockchain systems, the nodes should be assumed to be rational, self-optimizing agents (i.e., "players" in the game theoretic sense). It is therefore not straightforward that they will carry their tasks, in particular as primary nodes or as committee members, in the precise way prescribed by the system protocol. Our goal, then, is to provide incentives to the nodes to behave in the way they are expected, and specifically: when acting as primary, propose blocks according to a fair rule (e.g., according to order of appearance) rather than prioritize some transactions (e.g., their own) over others; and when acting as committee members, vote honestly. Our scheme is based on a *reputation system*. The idea is that, whenever a node is sensed to have performed its task according to the protocol rules, its reputation score would be incremented, while misbehavior should lead to a decrease in that score. The reputation score, in turn, would be translated into corresponding rewards and fines. The latter would be in the form of direct monetary transactions (i.e., getting money as a reward or paying money as a fine), as well is indirectly, e.g., by affecting the probability of being elected in the future as primary or committee member (which, in turn, would lead to a monetary reward for providing this service).

But the introduction of such a reputation system (with the associated rewards scheme) should be done with care. Indeed, self-optimizing agents may now focus on maximizing their reputation, independently of what the protocol dictates. To illustrate the potential trap here, consider a node that is serving as a committee member and detects that the block proposed by the primary does not pass its local test. The protocol dictates that the node would vote against the block acceptance. Yet the node might be concerned about finding itself being outvoted, which would likely cause a decrease in its reputation score. Indeed, if the node presumes that the most probable cause for the block's failure in passing the test is the potential dissimilarity between the system states observed by any two nodes, then, due to the assumed independence of such an event among different pairs of nodes, as well as the assumed small (a priori) probability for this event to occur, the committee member would conclude that, most likely, the block would be accepted by the majority of the other nodes, hence it should better "follow the crowd". Clearly, this would make the whole idea of committee voting futile. One might be tempted to overcome this problem by offering a sufficiently large reward to nodes that ("bravely") vote to reject a block, whenever they reach majority; yet this may end up in another undesirable equilibrium, where all committee members would vote to reject, hence maximizing their reward (note that, while this is not a dominating strategy, each committee member can count on the other members to reach the same conclusion, i.e., understand that this is a pareto optimal equilibrium).

The above problem is circumvented if the committee member would assume that the more likely cause for a block to fail the test is dishonesty of the primary. Hence, we shall employ the following assumption:

Assumption 1 (Dishonesty probability.)**.** *The probability that a primary acts dishonestly is larger than the probability that a block proposed by a honest primary does not pass the validation check.*

We note that the above assumption is quite reasonable. For example, for Helix it has been proven that a honest primary passes validation w.o.p. (Property 1). Moreover, for a given probability of "dishonesty of the primary" and a given value of the similarity probability α, one can enforce the above assumption by appropriately tuning the "slack variable" $\beta_\alpha(b_i)$. We note that, in principle, the dishonesty probability assumption is neither necessary nor sufficient to guarantee that a rational self-optimizing node would make a honest choice

when casting its vote. Indeed, its decision would depend also on the precise values of the reward and the penalty (in terms of reputation) for making the "right" or "wrong" choice (respectively). Yet, together with a well-balanced choice of rewards and penalties, the dishonesty assumption does guarantee a honest vote for nodes whose possible incentive for violating the protocol is solely their reputation scores.

B. Model and Notations

We consider a blockchain system in which blocks are based on a committee-voting principle, namely: a primary proposes a block and committee members cast their (binary) votes, and based on these the proposal is accepted or rejected. The vote of a committee member i on a block proposed by a primary p is based on a self evaluation of the validity of the proposal.

The reputation system is based on *reputations scores* of nodes, as follows.

(i) Associated with each node n there is a reputation score, RS(n), whose value can be either positive or negative.

(ii) RS(n) is initialized to some predetermined, non-negative value ρ.

(iii) RS(n) is updated upon the following:

- Whenever n serves as primary and proposes a block (see more details below).
- Whenever n serves as a committee member and votes on a block (see more details below).
- Over time, the (either negative or positive) value is aged. This can be achieved by incorporating each update through a convex combination of the current value of RS(n) and the new value.
- Additional events may be considered for updating the reputation score of a node. For example: upon agreeing to serve as primary or committee member; providing some periodic reward for "being awake" and "not causing problems" during some time interval (and incurring a penalty for "causing problems" during the interval); etc.

C. Reputation Update

We recall that whenever the primary node p suggests a new block, it is examined by committee members, each member i performs a self evaluation, based on which it indicates on the block acceptance or rejection. The evaluation computed by a committee member i for a proposal of a primary p shall be denoted by RP(p,i), i.e.: the committee member i approves a proposed block when RP$(p,i) \geq 0$. Accordingly, RP(p,i) is taken to be the (positive or negative) "influence" of i to the reputation score of p. For example, in the specific voting scheme of Helix, RP$(p,i) = |EB_p \cap EB'_i| - (\alpha b_i - \sqrt{10 b_i})$; note that, here, RP$(p,i)$ is a (decreasing) function of the similarity probability α, namely, a committee member is more critical for better network performance.

Updating the reputation score after proposing a block as a primary

As mentioned above, the validation of a block by a committee member is a binary decision, according to which a block is admitted *iff* RP$(p,i) \geq 0$. However, for the purpose of providing feedback for the reputation system, we suggest a smoother function, as follows:

- If the threshold condition was met, then RP(p,i) quantifies to what extent, and contributes a positive amount toward the reputation of p.
- Otherwise, it contributes a negative amount, which corresponds to the extent by which the threshold has not been met.

The aggregate contribution to the reputation of the primary p, following a block proposal submitted by it, is computed out of the RP(p,i) of all members i of a committee C, as follows. We first compute the mean value $ERP(p)$, of the various RP(p,i)'s, as well as the mean absolute deviation, MADRP(p):

$$\text{ERP}(p) = \frac{1}{|C|} \cdot \sum_{i \in C} \text{RP}(p,i)$$

$$\text{MADRP}(p) = \frac{1}{|C|} \cdot \sum_{i \in C} |\text{RP}(p,i) - \text{ERP}(p)|$$

We then define the following set:

$$\text{OK}(p) = \{i \in C| \ |\text{RP}(p,i) - \text{ERP}(p)| \leq \gamma \cdot \text{MADRP}(p)\}$$

for some predetermined $\gamma > 0$. Further considerations on the proper choice of γ shall be described in the following.

The contribution to the reputation of p (after its recent block proposal) is set as:

$$\text{RP}(p) = \frac{1}{|C|} \Sigma_{i \in \text{OK}(p)} \text{RP}(p,i).$$

Updating the reputation score after voting as a committee member

After the vote is cast, the reputation of each committee member i is adjusted by adding the following (possibly negative) component:

$$\text{RC}(i) = \text{MADRP}(p) - |\text{RP}(p,i) - \text{ERP}(p)|$$

We note that, for $\gamma \leq 1$, being in OK(p) implies a positive reward in reputation to the committee member, which might support such a choice of γ: indeed, this way committee members that are allowed to contribute to the reputation of the primary are granted an increase in their own reputation.

Further support for choosing a small (smaller than 1) value of γ is provided by the following argument. Suppose that a dishonest primary p colludes with a committee member $\hat{i}$ so that the role of $\hat{i}$ is to contribute to the reputation of p by providing a large value of RP$(p,\hat{i})$. Yet, $\hat{i}$ shall not perform such an action at the price of lowering its reputation score. We claim that, by choosing a sufficiently small (yet still reasonable) value of γ, we can guarantee that $\hat{i}$ will not be able to provide a (false) value of RP$(p,\hat{i})$ that is out of range. To concretize the discussion, suppose that each honest committee members i chooses a value of RP(p,i) that is uniformly distributed over some interval (RP_{min}, RP_{MAX}). Then, it can

be verified that, by choosing $\gamma \leq \frac{|C|}{|C|+2}$, $\hat{i}$ cannot increase $\text{RP}(p, \hat{i})$ beyond the (maximal nominal value of) RP_{MAX} and still be admitted into $\text{OK}(p)$. For example, for a committee of size $|C| = 8$, we would require $\gamma \leq 0.8$.

D. Reputation's Reward and Punishment

Maintaining a high reputation should maximize a node's utility, both directly and indirectly: directly - by offering monetary rewards for high values and imposing monetary penalties for small values; indirectly - by increasing the chance of being elected as primary or committee member, which, in turn, should provide monetary rewards. We proceed to discuss these in some more detail.

Primary node election

The primary should be randomly chosen among the set of nodes whose reputation achieves at least some predetermined (non-negative) value RP_{min}. Denote the set of such nodes by S_P. Then, the probability $p_P(n)$, of choosing a node n in S_P as primary, is set to

$$p_P(n) = \frac{RS(n)}{\Sigma_{m \in S_P} RS(m)}.$$

We may want to allow "mutations", so as not to be locked in a local optimum. E.g.:

- $p_P(n)$ is computed as the convex sum of the above term and $1/|S_P|$.
- We might consider allowing also nodes that are out of S_P to be chosen, by assigning them some (sufficiently) small probability ϵ. The probability of choosing a node in S_P is then $p_P(n) = [1 - \epsilon \cdot (N - |S_P|)] \cdot RS(n) / \Sigma_{m \in S_P} RS(m)$ where N is the total number of (active) nodes.

Election of committee members

Should follow similar lines, but incorporating the fact that the primary is out of the game and also that several (namely, some $|C| > 1$) members should be chosen.

Penalties and rewards associated with reputation

For maintaining the system, during each time interval each node n should pay some subscription fee $F(n)$. However, this fee is made dependent on the reputation scores, as follows.

If the reputation score is "neutral", i.e., $\text{RS}(n)$=0, then the fee is set to some nominal value F_{min}, which is the minimum per-node fee that is required for maintaining the system. That is: $\text{F}(n) = F_{min}$, where $F_{min} > 0$ and $F_{min} \cdot N$ is the amount required for maintaining the (N-nodes) system.

Now, denote by S^- and S^+ the sets of nodes with non-positive and strictly positive reputation values (respectively). Let $\psi > 0$ be a fine incurred per unit of negative reputation. Then:

- A node $n \in S^-$ pays a fee F_{min} - $\psi \cdot \text{RS}(n)$.
- A node $n \in S^+$ pays (or gets, if the value is negative) F_{min} - $\psi \cdot RS(n) \cdot \Sigma_{m \in S^-} RS(m)$ / $\Sigma_{m \in S^+} RS(m)$.

E. Possible Extensions

Suppose that we need to guarantee that, no matter how large a (fixed) external reward for deviating from the protocol is, the penalty incurred by the reputation system will ultimately balance it. To that end, we may keep increasing the associated fine (corresponding to ψ) as long as the node keeps maintaining a negative reputation score. The increased revenues from such fines would be distributed among the good nodes, as per above.

Still, a node may misbehave as long as it is beneficial, then it would behave itself for some time (letting the negative score diminish enough through the aging process), and then misbehave again. To address this, we can decrease the rate of aging as the reputation score gets more negative, i.e., we maintain a longer history for significantly misbehaving nodes.

V. Declarations on Pending Transactions

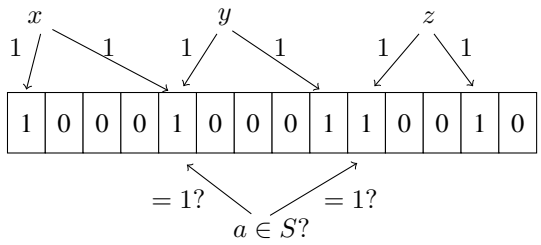
A. Intuition for Declarations

We would like to further enhance the fairness of the block selection. Even with the alternative examination of the block proposal from Section III and the reputation system from Section IV, it may be easy for a primary to ignore a few particular transactions without being detected. In particular, when the block is small it can be easier to manipulate some part of it. Also, when the network conditions imply low values of α, the validation process cannot be strict and manipulation in the block selection is easier to perform.

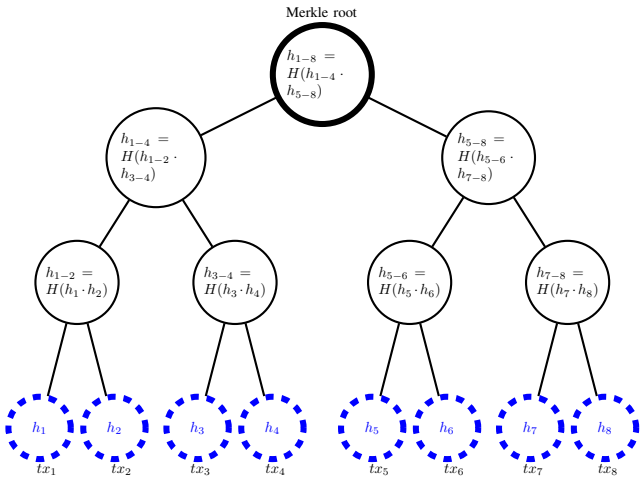
We proceed to suggest an approach that is based on periodically asking nodes to *declare the set of transactions they are aware of*. Then, a block proposal would be tested based on a recent declaration of the primary. When provided, a declaration would be examined to include enough transactions of other nodes. A crucial point in this scheme is that, at the time of the declaration, a node is not aware of the random ordering of the transactions in a specific future round, thus it cannot predict those particular transactions of others it has to include and would like to ignore upon being selected as a primary. To allow itself to make some meaningful manipulation, it would have to make in advance a major self adjustment to its declaration, hence significantly increasing chances for being detected.

A clear correlation exists between the effectiveness of a declaration and its size. On the one hand, a detailed declaration can be more helpful for better validation, yet it may require a large communication overhead. On the other hand, a concise declaration (e.g., including partial information, or compressed information with loss) reduces the opportunity to detect missing transactions in the proposal. We believe that a restriction on the allowed amount of communication overhead typically exists, thus we focus on communication-efficient declarations.

A declaration on the set of pending transactions known to a node can be seen as a description of a set. Designing such representations, either exactly or concisely, while losing some information, is a well studied research area with many applications [8], [9], [10]. The choice of the representation scheme is based on the application, requirements, such as the



(a) The Bloom filter, supporting membership queries



(b) The Merkle tree, enabling inclusion and exclusion proofs

Fig. 2. Illustration of the Bloom filter and the Merkle tree, two popular data structures for set representation.

support of answering membership queries, the allowed types of errors, the ability to prove the inclusion or exclusion of an element, and whether the order of elements has significance. For a representation of a set S, there are two kinds of errors in membership queries: a false positive (when an element $x \notin S$ is reported as a member of S) and a false negative (when an element $x \in S$ is reported as a non member).

In the following we give an overview of potential declaration schemes. We examine multiple criteria, such as the declaration size and the ability to easily examine a declaration or a block proposal. We also refer to the ability of a node to prove its honesty (in the block selection) by being able to show that a missing transaction was not included in a declaration. The results are summarized, at a high-level, in Table I.

B. Baseline - Reporting Complete List

As a baseline declaration, one might consider a declaration including the complete list of transactions known to a node. Such a detailed declaration will be long, implying a large communication overhead. On the positive side, such a declaration can easily be tested when proposed and later be useful in a simple validation of the honesty of a block proposal. To validate the declaration, a node examines that the complete list includes a large portion of its transactions. A block proposal is examined by a node by making sure that each of its transactions, missing although being expected to appear in a block following the block hash threshold, does not appear in the declaration, namely is not known to the primary.

Scheme	Declaration size	Declaration testing	Block testing	Proving honesty (non-membership)
Complete list	Large	Local	Local	Complete
Bloom filter	Small	Local	Partial	Partial
Merkle tree	Small	Commun.	Commun.	Complete

TABLE I
FUNDAMENTAL PROPERTIES OF VARIOUS DECLARATION SCHEMES

C. Bloom Filter based Declarations

The Bloom filter [8] is a popular data structure for set representation, supporting element insertion and answering membership queries. It is widely used for multiple blockchain purposes such as summarizing the set of transactions in an Ethereum block [1] or representing the set of addresses a Bitcoin SPV (light) client is interested in [11]. Beyond blockchain, it is also common in many networking device algorithms [12].

The Bloom filter encounters false positives and has no false negatives. The probability of an error (ratio of non-member elements reported as members) decreases when more memory is allocated for the data structure and increases when a larger set S is represented. The Bloom filter, illustrated in Fig. 2(a), stores an array of bits, where a set of hash functions is used to map elements to locations in the bit array. With initial values of zero bits, the elements of S are first inserted to the filter, setting to a value of 1 all bits pointed by the hash functions. Upon a membership query, the bits mapped by the queried element are examined and a positive answer is returned only when all these bits have a value of 1.

We proceed to describe how the Bloom filter can be utilized for the enforcement of fairness. Every few rounds of block selections, each node summarizes in a filter the pending transactions from its local pool that were issued by other nodes (rather than by the node itself). The node then distributes the filter to other nodes, while reporting the Bloom parameters (namely the number of elements, filter length in bits and hash function number). Other nodes (e.g., those that belong to a committee, which is selected for this purpose in a hard-to-predict way) examine the filter validity. They would like to see that the filter includes, in the set represented by it, some portion of their transactions. The filter is approved upon achieving some minimal required support from the committee. A node whose filter was not approved is not selected as the primary for the next several rounds. For a node i, we denote by S_i the represented set of known transactions and by F_i the set of transactions with a positive indication in the Bloom filter such that $S_i \subseteq F_i$.

In the following rounds, one node is selected as the primary among those with an approved filter. The primary p proposes a block EB_p selected from the pending transactions that were available to the primary by the time of computing its most recent (and recently reported) Bloom filter. Assume that the block maximal hash value is $T_p = \max\{H(etx)|etx \in EB_p\}$. A committee member i with local pool EP_i computes $EB'_i =$

$\{etx \in EP_i | H(etx) \leq T_p\}$, the set of etxs in EP_i with hash values lower than T_p. In contrast to the Helix scheme, in its examination, node i does not simply compare $|EB_p \cap EB'_i|$ to some lower bound but rather it carefully examines $\Delta_i = EB'_i \setminus EB_p$, i.e., the set of transactions not included in the proposal although being expected to. If the primary is honest it must not be familiar with any such transactions.

Specifically, node i checks whether Δ_i is aligned with the Bloom filter reported by the primary p. Namely, for each $etx \in \Delta_i$ one out of the following two conditions should hold:

- $etx \notin F_p$, and the transaction was not reported by the primary as a pending transaction in its pool,
- $etx \in F_p$ and the transaction was reported as known to the primary, yet there was a false positive, namely $etx \notin S_p$.

The validation of the declaration restricts the number of transactions of the first type, namely those satisfying $etx \in \Delta_i, etx \notin F_p$. The committee member i examines the transactions in $\Delta_i \subseteq S_i$. It skips those not in F_p, namely those for which the filter returns a negative membership indication. For those in F_p, since they are not included in the proposed block of the primary, they must not be known to the primary and the fact they belong to F_p is due to false positives. The committee member challenges the primary with the list of such transactions in $\Delta_i \cap F_p$. The primary has to demonstrate that they are indeed false positives by indicating on other transactions that the hash functions map to the same bits of the filter. Failing to do so indicates on the primary dishonesty.

Moreover, the Bloom filter properties do not enable the primary to (completely) prove its honesty, showing that all missing transactions were not among the set for which the declaration was computed for. The reason is that the Bloom filter does not enable to prove the non-membership of an element in the represented set for such elements causing the false positives, namely those transactions in Δ_i that also appear in $F_p \setminus S_p$.

D. Merkle Tree based Declarations

Another common data structure in Blockchain networks is the Merkle tree [9], also used for concise set representation while supporting different functionality than the Bloom filter. As illustrated in Fig. 2(b), the Merkle tree is a binary tree where a leaf is associated with a set element and its hash value. An internal node hash value is computed based on those of its direct children. The hash value of the root is the *Merkle root*.

Upon declaring the Merkle root for a represented set S, it is later possible to prove the inclusion of an element in S. The Merkle inclusion proof consists of the values of all the siblings of the nodes in a path to the root from the leaf corresponding to the element. Moreover, elements can be maintained in a sorted manner in the tree leaves. This enables to also demonstrate exclusion of an element through an exclusion proof showing the inclusion as adjacent leaves of a predecessor and successor, with lower and higher hash values, respectively.

A declaration of a node i includes publishing the Merkle root for its pool of pending transactions. Testing the declaration by another node j cannot be done locally by node j and requires sending challenges to node i. Based on sampling, node j repeatedly selects a transaction it issued and asks node i to demonstrate it is included within the tree through a membership proof. This should also include statistically verifying the tree values are sorted through examining the locations of the queried transactions. Node j approves the declaration if a large portion of its challenges are answered by node i. Given a block proposal, a committee member identifies the missing transactions and demands from the primary an exclusion proof demonstrating that such transactions were not included in the declaration. Providing such proofs for all missing transactions establishes the honesty of a primary node.

VI. Conclusions and Future Work

In this work, we proposed a set of new techniques to enhance the fairness of block selection. First, we described an accurate evaluation of a block proposal through a joint decision of committee members. We then presented a reputation system for incentivizing nodes to follow the block selection protocol and to provide a honest feedback on block proposals of other nodes. Last, we showed how declarations of nodes on their pools of pending transactions can dramatically limit their ability to manipulate the block selection. For future work, we would like to find optimal tradeoffs among the characteristics of transaction declaration schemes. In particular, we would like to determine the existence of a scheme implying short declarations that enables local testing of a declaration and a block proposal, while maintaining the ability to show the non-membership of any transaction not part of the declaration.

References

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014. [Online]. Available: http://gavwood.com/Paper.pdf

[2] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *ACM Special Interest Group on Security, Audit and Control (SIGSAC)*, 2016.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[4] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Springer CRYPTO*, 1992.

[5] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012.

[6] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2017.

[7] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira, "A fair consensus protocol for transaction ordering," in *IEEE International Conference on Network Protocols (ICNP)*, 2018.

[8] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[9] R. C. Merkle, "Secrecy, authentication, and public key systems," *PhD thesis, Stanford*, 1979.

[10] R. MacDavid, R. Birkner, O. Rottenstreich, A. Gupta, N. Feamster, and J. Rexford, "Concise encoding of flow attributes in SDN switches," in *ACM Symposium on SDN Research (SOSR)*, 2017.

[11] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber, "On the privacy provisions of bloom filters in lightweight bitcoin clients," in *ACM Annual Computer Security Applications Conference (ACSAC)*, 2014.

[12] A. Z. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2003.