A Minimal Agency Scheme for Proof-of-Stake Consensus

This is preliminary work that is subject to change.

David Galindo and Jonathan Ward*

Fetch.AI, St. John's Innovation Centre, Cowley Road, Cambridge CB4 0WS, UK

We outline a Proof-of-Stake (PoS) consensus scheme for a decentralised ledger that is designed to achieve strict transaction ordering, rapid confirmation times and greater security than existing technologies. The protocol uses a cryptographic random beacon for electing a committee of nodes that are tasked with reaching an agreement on a candidate transaction set. This is achieved by the committee members appending lists of proposed transactions as vertices of a Directed Acyclic Graph (DAG). The DAG is closed by a member of the committee, elected as leader, who proposes a set of tip vertices that serves as a compact representation of the candidate transaction set. After this transaction set is notarised and subsequently reaches finality, the next block to be appended to the blockchain is constructed by executing a function that performs a deterministic mapping from the partially ordered transactions stored on the DAG to a block with a strict ordering. The deterministic notion of finality is used to provide fast confirmation times and guard against well known avenues of attack on Proof-of-Stake consensus schemes. The protocol is designed with a *minimal agency* principle to restrict the control that any individual participant has on the changes of state that are applied to the ledger. The random beacon, collaborative block production, deterministic mapping and other elements of mechanism design restrict the agency of individual nodes, and thereby enhance the ledger's security.

* url: `https://fetch.ai/`, e-mail: info@fetch.ai

I. INTRODUCTION

One of the principal benefits of blockchain technologies is that they are decentralised, which means that rather than being controlled by a central authority, a collection of participants make decisions on the basis of a mutually agreed set of rules [1]. This set of rules is used to agree on changes to the state of a distributed ledger that serves as a record of the activities of the participants in the system (such as a list of financial transactions). Reaching agreement on the basis of an accepted set of rules is referred to as *consensus*. Ideally, consensus would involve all of the participants in the network but this goal is difficult to achieve in practice. The technical obstacles to a fully-inclusive consensus include the finite time required for messages to be propagated across a peer-to-peer network [2, 3] and the possibility that some of the actors on the network are either faulty or malicious [4]. This implies that the time required for all participants to agree on the same state can be prohibitively long if some of them deviate from their expected behaviour.

The Fischer-Lynch-Paterson theorem [5] states that it is impossible to guarantee convergence to a single state *(security)* in a finite time *(liveness)* in the presence of failures *(fault tolerance)*. It further states that any implementation of a distributed system can only hope to simultaneously attain two out of these three properties under all conditions. Most consensus protocols therefore attempt to achieve these three desirable properties under conditions that are likely to be encountered in practice, such as ledger replicas failing at a particular rate or being under attack from an adversary who controls only a certain proportion of the replicas within the distributed network. Many contemporary blockchain systems prioritise *liveness* and *fault tolerance* over *security*, with convergence to a single state occurring only provisionally and under a specific set of assumptions [6]. In these systems, the distributed consensus is achieved by creating a competition or lottery between the different network participants. The winner of these contests is then recognised by other nodes as having earned the right to append the next block to the chain.

In most blockchain designs, participants gain entry to the contest by provably making some kind of costly sacrifice. In Proof-of-Work blockchains the competition involves winning a race against other nodes to solve an expensive but otherwise pointless computation [7–9]. The sacrifice in Proof-of-Stake involves locking funds for a significant amount of time [10, 11], while in some versions being at risk of losing this stake if they engage in dishonest behaviour [12].

The process of adding a new block to the chain can therefore be seen as giving the winner a one-shot monopoly over the transactions that are added to the ledger [13]. The computers or collections of computers that attempt to add blocks to the chain in order to participate in consensus are frequently called *miners*. In the original blockchain, Bitcoin, the monopoly power is restricted by subsequent blocks being added by different miners [1]. If a miner submits an invalid block, e.g. because the digital signatures are incorrect or transaction values have been calculated wrongly, then it will be ignored by other miners who will continue to add blocks sequentially to the last valid

block that they received. More recent protocols also include additional validation steps to resolve these types of "forks" between different competing blocks in the chain [14–16].

Although the implicit or explicit validation of blocks does, to some extent, restrict the misconduct that is possible on the part of the block producer, recent theoretical studies and empirical observations of several existing blockchains have revealed that they do not completely remove it. Examples in which miners' incentives are in conflict with proper functioning of the system include the group of mining pools on the Bitcoin network that collude with each other to share and validate blocks to gain rewards at the expense of other miners on the network [17], or the practice of mining of so-called "empty" blocks [18] that are devoid of transactions[1]. Finally, there are also plausible theoretical avenues for miners to be bribed to initiate double-spending attacks (i.e. fraud) [19] and to deny service to either specific users or the entire network [20]. Collectively, these issues indicate that current blockchain designs potentially cause them to be subject to many of the same flaws that have been ascribed to centralised systems [21].

One way of reducing the potential for misconduct is to reduce the *agency* that any particular actor, in this case block producers, has over the transactions that are entered into the chain. In an ideal system, processing nodes would be incentivised to propagate all transactions that they encounter across the decentralised network [22], and some objective criteria would be used to decide which of these transactions and in what order should be recorded. It is with these design goals in mind that we propose the *minimal agency* consensus protocol.

A reduced reliance on a single block producer, has been attempted by other "leaderless" protocols such as SPECTRE [23] and the IOTA cryptocurrency [24]. The key idea of these two proposals is to generalise the concept of a *linear blockchain* to a Directed Acyclic Graph (DAG), where each block, containing information from one or more transaction, is added to at least one previous block. In IOTA, the vertices that are added to the DAG contain individual transactions while the links are hash-references to transactions that were added earlier to the DAG. Unfortunately, fully decentralised DAG-based protocols only have weak liveness guarantees, which means that the time required for a transaction to be accepted, referred to as *finality*, is variable and could sometimes be excessively long. IOTA resolves this difficulty by introducing a coordinator to validate transactions that would never reach finality according to the rules of the decentralised network. However, the presence of a coordinator means that the protocol is *de facto* centralised. Moreover, the partial ordering of events and variable time-to-finality also make pure DAG-based schemes unsuitable as platforms for smart contracts, as these require a strict ordering of transactions to properly maintain complex state databases.

[1] The practice of publishing empty blocks can give the miners an increased probability of earning a block reward as it enables them to begin solving the hash-puzzle without first preparing a list of transactions. Other miners also prefer building on empty blocks as it means that they are not required to verify or execute transactions. The high value of the block reward compared with transaction fees makes these behaviours rational.

Our Contribution. The purpose of this document is to provide a high-level description of a consensus protocol for building a distributed ledger that possesses the strict ordering and uniform time-to-finality of linear blockchains, while at the same time integrating the robustness and scalability of DAG-based consensus schemes to achieve a protocol that offers the "Best of Both Worlds". As we explain in the following text, this is in achieved by notarising a DAG between successive block rounds in a procedure that makes use of a cryptographically secure source of randomness.

The protocol is based on a Proof-of-Stake (PoS) consensus, and is designed to provide secure and rapid block confirmation times. This contains five elements:

(i) A *DAG* that is used to temporarily store transactions; this serves as a compact means of collaboratively agreeing on a set of candidate transactions for inclusion in a subsequent block. The purpose of the DAG is therefore to serve as a mechanism for constructing an agreed transaction memory pool.

(ii) A *cryptographic source of randomness*, known as a Decentralised Random Beacon, for selecting a committee of stakeholders that reach an agreement on the current transaction set.

(iii) A *coordination mechanism* for DAG assembly based on a game-theoretic construct.

(iv) A *DAG notarisation layer* where the transaction set for each block epoch is constructed from the DAG by the leader or other high-ranking member of the committee, followed by notarisation of the proposals by other committee members. Each proposal references a single proposal from the previous block epoch so that the transaction proposals are chained in a similar way to blocks in a conventional blockchain. For proposals beyond a fixed number of rounds in the past, this provides a deterministic notion of finality to prevent "nothing-at-stake" [11] and "long-range" [25] attacks.

(v) A *blockchain layer* After the transaction proposal has reached finality, a deterministic function that provides a mapping from a partially ordered set of candidate transactions to a subset of agreed transactions with strict ordering is used to produce a block. This agreed set of transactions can be entered into linked blocks forming a chain.

One of the main advantages of DAGs is that they enable transactions to be communicated asynchronously across a peer-to-peer network, which is in contrast with the single block propagation events that are required in linear chains. This offers increased security at higher block production rates [23]. A potential difficulty of notarising a DAG between successive block rounds is that the incentives for adding vertices may vary significantly between the beginning and the end of a particular round. This could lead to a loss of asynchronicity, with the majority of the vertices potentially being added to the DAG during a disproportionately small time window, which could in turn lead to a smaller number of nodes contributing to the consensus (in the remainder of the text we use the term 'node' to refer generically to full or partial replicas of the ledger). This would decrease the system's security and its resistance to bribery and other types of side-channel

manipulation [20, 26]. We resolve this limitation by incorporating a novel game-theoretic construct that coordinates DAG assembly to increase security and throughput. This construct is currently the subject of a patent application and will be described in later publications.

Recent protocols such as Algorand, Dfinity, Orbs, OmniLedger and Ethereum 2.0 [14–16, 27, 28] incorporate features that are analogous to elements (ii), (iv) and (v). However, our approach offers higher throughput and a reduced potential for bribery or other types of misconduct by processing nodes. The greater resistance to misconduct is achieved by two complementary mechanisms that significantly restrict the nodes' *agency*, i.e. the ability for an individual node to choose which transactions contribute to the global state. The first mechanism is an algorithmic means for selecting transactions from a larger candidate set that best fulfil specific criteria, such as being associated with a sufficiently high transaction fee. This can introduce complexity in the mapping between the transactions stored on the DAG and the finalised block [29], which partially obscures its structure from the node. The second mechanism is the inclusion of economic incentives that encourage nodes to endorse transaction sets that truthfully represent their current knowledge. This is a property known as *incentive compatibility*, and if this goal is successfully realised then deviation from a "passive" strategy (i.e. following the protocol exactly) will typically lead to financial losses for the node. These two elements together limit the nodes' capability for misconduct, and constitutes an example of what we call *minimal-agency by design*, thereby increasing the security of the protocol.

In the remainder of this document, we outline the protocol design but note that this is subject to modification before release of the Fetch.AI main network launch at the end of 2019 to reflect issues such as the development time and complexity, security testing or novel theoretical or experimental results.

II. PRELIMINARIES

In the following sections we describe two key components that are used in our consensus; Directed Acyclic Graphs and Decentralised Random Beacons.

A. Storing Transactions on a Directed Acyclic Graph (DAG)

The fundamental data structure that is used to record transaction events is a DAG that we denote as $G = (V, E)$, where $v_i \in V$ are the vertices that contain collections of transactions and E represents the set of edges. For each vertex, the out-going edges are hash-references to exactly $k_{\text{out}} = 2$ previous vertices[2]. Each vertex also includes the round number of the block and a digital signature from the node that appended it to the DAG. The set of nodes that have the authority

[2] In general, the vertex out-degree $k_{\text{out}} \geq 2$. The lower bound is chosen to minimise additional storage requirements but since the DAG is a transient data structure the cost of choosing a larger value is small. There is no restriction on the number of incoming edges to a vertex.

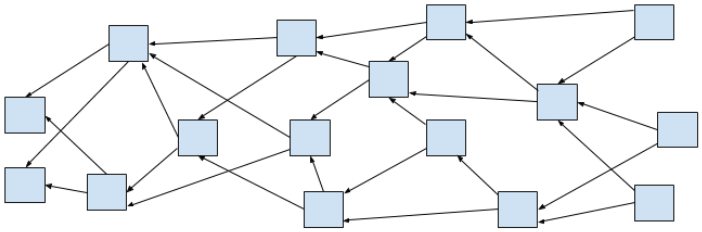

Figure 1. **A directed acyclic graph (DAG).** A pair of genesis blocks are represented by the two left-most vertices.

to append elements to the DAG and the number of vertices that they can append in any round is restricted (see section II B). This allows many types of misconduct such as large numbers of invalid vertices being posted by a malicious node to be easily identified and either penalised through stake slashing or ignored.

The diagram in Figure 1 shows a transaction DAG with the genesis vertices at the left-most positions[3]. Nodes can asynchronously add vertices to the DAG that reference these genesis vertices over the time period between blocks being produced, which we refer to as an epoch. These vertices are then referenced by further new vertices, which leads to right-ward extension of the DAG over time. In this context, asynchronous simply means that each node can keep a slightly different copy of the DAG but that this does not prevent the addition of new vertices. These vertices can then be synchronised across the peer-to-peer network to update the DAG replicas stored by the other nodes.

A DAG provides partial ordering in the sense that the set of vertices that are both directly and indirectly referenced by a specific vertex, v_i, which we denote $past(v_i, G)$, occurred with certainty in its past. A similar argument applies to vertices added in vertex v_i's future but there is also, in general, a further set of vertices that are not present in either the past or future set and that therefore have an ordering that is undefined with respect to v_i.

The compact representation of vertex sets is the key property of the DAG that is utilised in the protocol to increase block throughput. This allows the DAG to be extended throughout the block epoch, and for communications that involve only a few vertices to be used to decide on the final transaction set (Figure 2). Since $\forall v_j \in past(v_i, G) : past(v_j, G) \subset past(v_i, G)$ the most compact representation of the transaction set involves tip vertices that are yet to be referenced by any other vertex.

An important question concerning the design of the system is how to incentivise nodes to append transactions to the DAG that are (i) valid and associated with a sufficiently high transaction fee[4]

[3] In the first block round these are specified by the protocol but in subsequent rounds these are vertices from the previous block round

[4] Non-zero fees are necessary to prevent denial of service attacks on the ledger and to provide a sufficiently large incentive for operators of nodes.

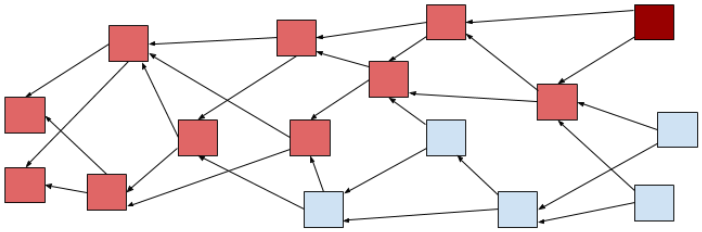

Figure 2. **Compact representation of vertex sets.** A single vertex (dark red) can be used to compactly represent a set of vertices that were previously added to the data structure in its past (light red squares).

(ii) can be represented compactly with a small number of tip vertices and (iii) are agreed by a large number of participants in the consensus. A protocol for achieving these objectives is discussed in the following sections.

B. Decentralised Random Beacon *via* Elected Committees

Our consensus protocol uses a Decentralised Random Beacon (DRB) as a mechanism to elect a group of nodes that we refer to as a *committee* who collaboratively build and decide upon the next block that will be entered into the ledger. The DRB is a special case of Multi-Party Computation protocol. The DRB is initiated during a Setup phase that involves a set of ℓ nodes. This set, representing all nodes that can participate in the consensus, engage in an interaction that culminates with the creation of a pair of verification and secret keys $(\mathsf{vk}_i, \mathsf{sk}_i)$ for the i-th node, as well as a global key pair $(\mathsf{pk}, \mathsf{sk})$. The latter key pair implicitly defines $f_{\mathsf{sk}}(\cdot)$, a Verifiable Random Function (VRF), where sk can be thought of as a virtual secret key that is never computed explicitly. The set of nodes' secret keys thereby computed enable any subset of nodes of size $h+1$ to compute the verifiable random value $f_{\mathsf{sk}}(x)$ for a publicly known input x. Conversely, any set of at most h nodes cannot learn any information on $f_{\mathsf{sk}}(x)$ for any x not previously evaluated. It is assumed that an adversary that wants to predict the random value for future rounds will not control more than h nodes at any point in time, where typically $\frac{h}{\ell} \in [1/3, 1/2]$.

Similarly to [14, 15], the initial set of ℓ nodes can be divided into multiple disjoint subsets to increase the rate at which the values of the random beacon are produced. The committees would then run a setup phase independently of each other, so that each committee generates a pair of global keys $(\mathsf{pk}_j, \mathsf{sk}_j)$ along with verification and secret keys $(\mathsf{vk}_i^j, \mathsf{sk}_i^j)$, for $j = 1, \ldots, \ell/c$ and $i = 1, \ldots, c$, where c is a prescribed committee size[5]. The threshold value h could be adjusted to $\frac{h}{c} < 1/3$ if availability is given precedence over security. An alternative is to choose $\frac{h}{c} < 1/2$, which prioritises security over availability. At each new round, a different committee is selected randomly to compute the next value of the random beacon.

[5] For ease of exposition, we assume that ℓ is a multiple of c

A decentralised source of randomness is then obtained by parallel distributed computations of the random function $f_{\mathsf{sk}}(\cdot)$ by a $(h+1)$-subset of nodes (to simplify the notation we drop the index, j, of the committee generating the random beacon). In each round, indexed by $r \geq 1$, every node computes a seed s_r as the value $s_r := f_{\mathsf{sk}}(s_{r-1}||r)$, where s_0 can be any constant that is publicly known before the Setup phase. The output s_r of the VRF, together with a proof of correctness π_r, are included in the proposed block as $\langle s_r, \pi_r \rangle$. Once agreement is reached on random value $s_{r-1} := f_{\mathsf{sk}}(s_{r-2}||r-1)$ in round $r-1$, the seed s_{r-1} for the subsequent round of the DRB is known, and the process can be repeated to compute s_r. For a group of ℓ nodes that have been divided into ℓ/c different committees, the seed s_{r-1} for round $r-1$ is used to select the committee for computing the next random value and deciding on the next block to be added to the ledger.

All of the ℓ consensus participants contribute an identical stake S to gain entry to the consensus. This design choice is implemented to ensure that the contributions that each stakeholder makes to the consensus are associated with uniform financial risks and rewards. In unpermissioned platforms this does not prevent a single entity controlling multiple stakes, and hence the system needs to be resistant to concerted attacks from more than 50% of the stakeholders. In our case, the precise setting of S remains to be determined, but possibilities include the use of k^{th}-price auctions [30] or a minimal threshold specified by the Fetch.AI foundation.

III. MINIMAL-AGENCY CONSENSUS PROPOSAL

In the following section we provide a brief overview of our consensus protocol proposal. As mentioned previously, this incorporates a *minimal agency* design principle, in which the nodes' agency (or ability to deviate from the prescribed protocol) is reduced to a minimum through a combination of cryptography and mechanism design.

A. Agreeing a Common Memory Pool with a DAG

The pseudo-random number generator $\sigma(\cdot)$ that is used to select the indices [31] of a subset of nodes that will form the committee for deciding on the next block to be added to the ledger can also be used to obtain a ranking of the committee members. The node with the highest priority represents the *leader* of the current round, who can propose k_{final} vertices as transaction sets for that round. We refer to the element specifying a set of vertices for closing the DAG as a "tip proposal", which includes a reference to a single tip proposal from the previous block epoch and a digital signature from the *block proposer*. The value of k_{final} is a parameter that is specified at the protocol level with low values being more "exclusive" by restricting the vertices to only those associated with the greatest stake while larger values lead to inclusion of a larger number of vertices.

During normal operation of the ledger, it is expected that a single tip proposal will be made by the leader, and that a failure of the leader to make a proposal will be dealt with by other nodes with

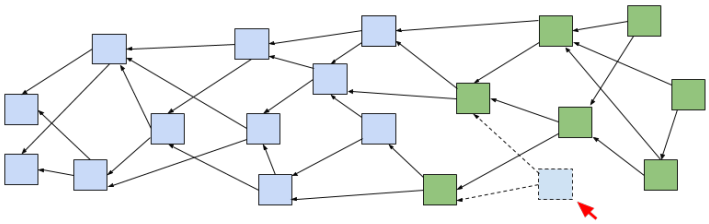

Figure 3. **A simple rule enables transaction vertices for the next block time epoch to be appended before notarisation of the current block.** Vertices appended to the DAG by nodes elected in round r are coloured in pale blue and vertices from round $r+1$ in green. All of the pale blue vertices remain in the *past* or have undefined ordering with respect to the green vertices provided that the rule of ascending round numbers is obeyed. The pale blue node with a dashed line and marked by a red arrow violates this rule and is invalid.

high priority generating alternative proposals. In some circumstances, this can lead to multiple tip proposals being made for a single block epoch but these forks can be resolved in later epochs. The notarisation of the tip proposals is achieved by each member of the committee appending a digital signature that is applied to data that includes the round number, r, and public key of the block proposer. Honest members of the committee notarise tip proposals that have a greater priority than the highest priority tip proposals that they have received during that round, and that reference the proposal from the previous round with highest priority.

Since messages passed across the network have different degrees of latency, it is possible that the value s_{r+1} of the DRB for the next epoch corresponding to round $r+1$ will be known to some nodes before notarisation of the DAG in round r. Nodes that are elected to become members of the committee for the next epoch can continue to add vertices asynchronously to the DAG, which will contribute to the subsequent block epoch. These will assuredly be considered for the next block epoch if they are appended with the rule that the vertex can only reference vertices with the same or lower round number as shown in Fig. 3. The incentive design and mechanism for converting the notarised DAG to a block is described in the following section.

B. Constructing and closing a DAG

In each epoch, it is expected that the leader will publish a vertex that references a tip-set V_{tip} that specifies a candidate vertex set V_c for inclusion in the current block defined as

$$V_c = \bigcup_{v_i \in V_{\text{tip}}} past(v_i) \tag{1}$$

and that a leader correctly following the protocol will propose a vertex set that maximises $|V_c|$ for their local copy of the DAG as this is equivalent to maximising the stake associated with the

transaction set[6]. This is consistent with notion of allocating "voting" rights on the consensus in proportion to the amount of cost that has been incurred by the different participants [1]. The leader shares a fixed fraction of the block fee, which is comprised of a block reward and transaction fees, with the members of the committee that proposed vertices that were included in V_c.

Since the nodes appending vertices to the DAG also receive a proportion of the transaction fee, and only a small number of tips will be used to define the transaction set, nodes have an incentive to choose references that maximise the number of tips that include the vertex in their *past*. This can be achieved by (i) choosing a set of transactions that are associated with high fees as this increases the reward that is shared with other nodes, and (ii) maximising the stake that is referenced by the vertex. The first property is associated with the transactions contained in the vertex while the second property is associated with its edges.

At the end of the epoch and assuming that the vertex to be appended is certain to be the final entry to the DAG, the second property suggests (for $k_{\text{out}} = 2$) choosing a pair of edges, $e_n \in \{v_i, v_j | i \neq j\}$, that maximise $|past(v_n, G')|$ where v_n is the vertex to be added and G' is the resultant DAG, which implies the following calculation for e_n,

$$e_n = \underset{i,j}{\text{argmax}} \ |past(v_i, G) \cup past(v_j, G)| \tag{2}$$

This also suggests a simple greedy, but potentially sub-optimal, algorithm for appending transactions to the DAG earlier in the block epoch.

After the tip proposal for the vertices that close the block epoch have been announced by the leader, the subsequent block epoch can be initiated by nodes that have received sufficient shares to compute the next value of the random beacon s_{r+1}, and are identified as being part of the committee.

IV. IMPLEMENTATION CONSIDERATIONS

The identity of each node is specified by a public key that is known in advance by all other nodes. This identity is then used in the setup phase of the Decentralised Random Beacon that instantiates the cryptographic source of randomness. Nodes who have not participated at any point in time in a (per committee) setup phase for the DRB cannot append vertices to the DAG. Nodes are also only entitled to append a single vertex to the DAG in a particular block epoch, with a penalty levied through a special transaction if they deviate from this expected behaviour. The protocol will include mechanisms that allow nodes to join or leave existing committees and for entirely new committees to be created.

[6] Each fixed-cost stake is associated with the right or obligation to append a single vertex to the DAG as a committee member in a specified epoch.

After the set of transactions has been decided by the communication phase of the protocol that includes subsequent rounds of DAG notarisation and a finalisation procedure, the nodes then compute a deterministic mapping between the transaction set, as defined by the DAG, and a block. In unsharded blockchains the selection of transactions could involve simply ordering them according to the ratio of the fees paid to transaction size, and rejecting any transactions that would lead to the block exceeding its size limit. Other possibilities include more sophisticated auction-like mechanisms such as, for example, k^{th}-price auctions [32]. In addition to the transaction information, the block would also include the signatures of the committee members that were used to notarise the tip proposals from that round.

In the Fetch.AI smart ledger, blocks have a particular structure that is designed to optimise parallel transaction execution, which greatly increases throughput in systems with complex smart contract languages [29]. Blocks consist of a header and a body, where the body is a Merkle tree that references a number of transactions. The header contains the block hash and a hash pointer to the previous block, so that, collectively, the block headers form a cryptographically secured linked list (the blockchain). The block's body is referenced from the header by the Merkle hash root. The novel aspects of the block data structure which are designed to make the ledger more scalable are found in the block body. Whereas the leaves of the Merkle tree used in a conventional blockchain each reference a single transaction, each leaf of the Merkle tree inside a Fetch ledger block references a list of transactions that correspond to block slices [29].

At high transaction volumes, computing this deterministic mapping could prove to be resource-intensive, and beyond the capabilities of a single node. At low transaction throughput, however, it is likely that a single node will have sufficient computational resources to find a good solution by using e.g. constraint-satisfaction problems or stochastic optimisation techniques. Since nodes have access to a virtual machine, the agreed procedure for mapping the DAG entries to a block can be executed by all participants, which means that the block itself does not need to be synchronised across the peer-to-peer network. However, the block will need to be communicated to light client nodes and it is possible that this will be more efficient than each node independently computing the mapping. Each block will consist of the tip proposal, committee notarisations and the set of transactions selected by the block packing algorithm.

The block packing algorithm defines a function that performs a mapping from a set of candidate transactions X_c^r to a set of accepted transactions $X_a^r \subseteq X_c^r$. This means that there is a further set of discarded transactions $X_d^r = X_c^r - X_a^r$ that could not be entered into a block. In the protocol, this set of discarded transactions is added to the set of candidate transactions for the next block epoch. These discarded transactions are also eligible for inclusion in a block over a window w of block epochs before being deleted from the nodes' memory pool and thus rejected from inclusion in the blockchain. This design choice means that the DAG must be retained for at least w epochs to maintain a full record of the discarded transactions.

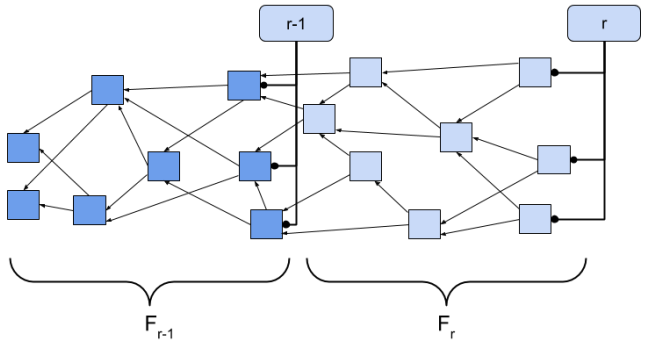

Figure 4. **Block fee shares between successive block epochs.** The vertices in dark blue were notarised in block epoch $r-1$, and resulted in a block that is associated with a total reward F_{r-1} comprised of transaction fees and a stationary block reward. This is followed by another set of vertices that being notarised and mapped to a block in round r.

Since the DAG is a transient data structure that does not affect the state of the blockchain after a block has been created and the discarded transactions rejected it does not need to be kept indefinitely. This means that wasted storage associated with redundant transaction hashes (present in multiple DAG vertices) and hash-references that define the edges in the DAG can be saved thereby reducing the cost of operating the blockchain. The other advantage of this approach is that nodes earn rewards from participating in the consensus in many epochs, which reduces the temporal variability or risk of operating a node and thereby increases participation in the protocol.

An issue associated with blockchains that are sustained by transaction fees rather than inflationary block rewards is the potential for the incentives for block-producing nodes to be in conflict with the efficient operation of the ledger. In the current ledger design, this situation could occur when several alternative transaction sets are notarised by the committee in the previous round. Efficient operation of the ledger would imply accepting the DAG notarisation that is associated with the largest stake and by implication the highest reward for the previous block. However, by accepting a smaller vertex set, the nodes in the subsequent committee can ensure that a larger set of transactions with correspondingly higher reward is available for incorporation into their block [33].

A straightforward approach to making the process *incentive compatible* is to split the block reward F_r that is payable in round r equally between the committee elected in that round and the committee from the subsequent round. This is depicted in figure Fig. 4 and would mean that the reward, R_r for the committee elected in round r would be given by $R_r = \frac{1}{2}(F_r + F_{r-1})$[7]. This places additional constraints on the function $F(\cdot)$, which maps set of candidate transactions X_c to a set of accepted transactions $X_a \subseteq X_c$ with an associated fee. The first condition is that the fee should increase monotonically with the addition of new transactions to candidate set $\forall X_c \supset$

[7] The committee producing the genesis block would receive $F_1/2$.

$X'_c : F(X_c) \geq F(X'_c)$. Another necessary condition is that higher transaction fees are recovered by splitting transaction sets between different block epochs so that $F(X_c^1) + F(X_c^2) \geq F(X_c^1 \cup X_c^2)$.

V. DISCUSSION AND FUTURE WORK

In this document we have outlined the Fetch.AI consensus protocol, which uses a minimal agency principle as a design philosophy to enhance the security and performance of a ledger. It is directed at achieving similar performance and security guarantees to systems under the control of a single entity, where consistent behaviour of the different nodes is assured. This is accomplished by using a combination of: (i) cryptography for allocating tasks incorruptibly to different participants; (ii) distributed computing for collaboratively agreeing blocks and executing transactions; and (iii) mechanism design for aligning the incentives of the nodes with efficient operation of the ledger. Another important property of the consensus is that it is highly resistant to Distributed Denial-of-Service (DDoS) attacks on any individual node as redundancy is present in all aspects of the consensus such as proposing transactions, notarisation or block construction.

A key consideration in decentralised ledger design is in how to overcome the so-called *scalability trilemma*, which states informally that trade-offs exist in terms of the *security*, *scalability* and the degree of *decentralisation* that can be realised by any protocol [13]. For example, many recent cryptocurrencies that are capable of attaining high transaction throughput can do so simultaneously with security but at the expense of decentralisation. In practice, this means that block production is controlled by a relatively small number of nodes, which abrogates the principal benefit of blockchain technologies; that they are not under monopoly or oligopoly control. The scalability trilemma is also argued to apply to Proof-of-Work blockchains such as Bitcoin where scalability is sacrificed (i.e. throughput is artificially limited) to provide a sufficient incentive for users to pay transaction fees to nodes in exchange for the provision security and a degree of decentralisation[8]. Perhaps the most important benefit of our *minimal agency* approach is that the deterministic algorithm, used to decide which transactions are entered into the blockchain, enables transaction fees to be levied without restricting throughput.

Separately to the economic issues, a technical constraint that leads to blockchains' limited transaction throughput is the replication of state across a large number of processing nodes. This can be resolved by partitioning state replication across independent nodes in a process known as *sharding* [34]. Theoretically, sharding enables the throughput of a blockchain to be increased linearly with the number of state shards, without affecting the computational requirements of processing nodes, and thereby enables high throughput to be attained without compromising decentralisation. The Fetch.AI scalable ledger implements sharding of state, network and contract execution [29] and is compatible with a variant of this consensus scheme where each shard is associated with a different

[8] The specialised equipment required for Bitcoin mining does lead to centralisation, however. [21]

committee and transaction DAG. This enables processing nodes with limited computational power to participate in the consensus by appending vertices to the relevant DAG associated with their state shards. The nodes with sufficient computational power to coordinate transaction execution must compute a public and deterministic algorithm to produce blocks which greatly restricts their influence over the consensus process.

Although our consensus design has several novel features, it also incorporates elements from other protocols. A common feature of many recent designs is the use of Proof-of-Stake as a criteria for participating in consensus combined with a cryptographic source of randomness for electing leaders or committees that have control over the blockchain for a limited period of time [14–16]. This approach provides an economically efficient and secure alternative to Proof-of-Work, and is likely to become established as an industry standard in permissionless ledger designs. In this protocol, the cryptographic source of randomness enables strict transaction ordering and rapid finality to be delivered while the DAG enables transactions to be communicated and agreed asynchronously.

A. Future Work

An important feature of the Fetch.AI scalable ledger is its optimisation for efficient smart contract execution. A potential bottleneck in the design is obtaining an ordering for the contracts that exploits the sharded state database structure to enable contracts to be executed in parallel [29]. As mentioned previously, at low transaction volumes it will be possible for individual nodes to compute the ordering operation but the complexity of this calculation could act as a bottleneck under conditions of high transaction load. In a future article, we will outline how this computational task can be distributed across the network of processing nodes to facilitate operation of the ledger at high loads. This makes use of Fetch.AI's synergetic computing platform to introduce a competition for nodes to optimise the block organisation. This platform was previously referred to as useful Proof-of-Work (uPoW) but we have renamed it as "synergetic computing" to reflect its greater generality and indirect role in the consensus. The replicated virtual machine allows block optimisation to be performed in parallel by the population of nodes. The minimal agency elements of the consensus design are preserved in this case by the public nature of the optimisation algorithm, restrictions on the function arguments that can be passed to the algorithm and the limited control that individual nodes have over the transactions that are candidates for entry into a block.

There are several aspects of the consensus protocol that are currently the subject of patent applications that will be revealed in future revisions of this document. These include novel cryptographic and game theoretic constructs, which are essential for efficient and secure operation of the consensus. These are related to the decentralised random beacon and to the algorithms and incentives for nodes to append vertices to the transaction DAG, respectively. We are also carrying out ongoing theoretical work to develop formal security proofs for the performance of the consensus when these critical elements are incorporated. In addition to this theoretical work, we are also

testing the performance of the consensus in simulations and after implementation and deployment in the Fetch.AI test network. These will include tests to examine its performance under conditions of high transaction volume and when subjected to attack by different types of adversary.

B. Acknowledgements

We would like to thank Troels Rønnow, Marcin Abram, Frederic Moisan, Daniel Honerkamp, Jin-Mann Wong, David Minarsch and colleagues at Fetch.AI for helpful discussions and their comments on this manuscript.

ACRONYMS

DAG: Directed Acyclic Graph. 4–6, 10

DRB: Decentralised Random Beacon. 4, 7, 9, 10

PoS: Proof-of-Stake. 1, 2

VRF: Verifiable Random Function. 7

GLOSSARY

c: Committee size. 7

k_{final}: Cardinality of the tip vertex set defining the set of candidate vertices. 8

k_{out}: Out-degree (number of outward edges) for vertices in a graph. 5, 10

epoch: Time period during which a block is agreed. 6

node: Full or partial replica of the ledger. 4

round: Index of block or the DAG associated with its generation. 4, 5, 9

tip: Vertex that is not referenced by any other vertices in a DAG. 6

---

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." `https://bitcoin.org/bitcoin.pdf`, 2008. [Online; accessed 8 Feb 2019].

[2] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology – EUROCRYPT 2017* (J.-S. Coron and J. B. Nielsen, eds.), pp. 643–673, Springer International Publishing, 2017.

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

[4] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67189–67205, 2018.

[5] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," in *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems – PODS*, pp. 1–7, ACM, 1983.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security 2016*, pp. 3–16, ACM, 2016.

[7] G. Huberman, J. Leshno, and C. C. Moallemi, "Monopoly Without a Monopolist: An Economic Analysis of the Bitcoin Payment System," 2018. Columbia Business School Research Paper No. 17-92.

[8] J. Ullrich, N. Stifter, A. Judmayer, A. Dabrowski, and E. R. Weippl, "Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids," in *RAID*, vol. 11050 of *Lecture Notes in Computer Science*, pp. 184–203, Springer, 2018.

[9] M. Thum, "The Economic Cost of Bitcoin Mining," *CESifo Forum*, vol. 19, no. 1, pp. 43–45, 2018.

[10] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake." `http://blockchainlab.com/pdf/peercoin-paper.pdf`, 2012. [Online; accessed 8 Feb 2019].

[11] V. Buterin, "On Stake." `https://blog.ethereum.org/2014/07/05/stake/`, 2014. [Online; accessed 8 Feb 2019].

[12] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget," *arXiv:1710.09437*, 2017.

[13] J. Abadi and M. Brunnermeier, "Blockchain Economics." `https://scholar.princeton.edu/sites/default/files/markus/files/blockchain_paper_v6j.pdf`, 2018. [Online; accessed 8 Feb 2019].

[14] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, 2017.

[15] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," *arXiv:1805.04548*, 2018.

[16] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira, "Helix: A scalable and fair consensus algorithm resistant to ordering manipulation." `https://orbs.com/wp-content/uploads/2018/09/Helix-V1.3.pdf`, 2019. [Online; accessed 18 Feb 2019].

[17] I. Eyal and E. G. Sirer, "Majority Is Not Enough: Bitcoin Mining Is Vulnerable," in *Financial Cryptography and Data Security*, pp. 436–454, Springer Berlin Heidelberg, 2014.

[18] S. Yoo, S. Kim, J. Joy, and M. Gerla, "Promoting Cooperative Strategies on Proof-of-Work Blockchain," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[19] G. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *ACM Conference on Computer and Communications Security*, pp. 906–917, ACM, 2012.

[20] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *Bitcoin Workshop at Financial Cryptography*, 2018. [Online; accessed 8 Feb 2019].

[21] N. Arnosti and S. M. Weinberg, "Bitcoin: A natural oligopoly," *arXiv:1811.08572*, 2018.

[22] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On Bitcoin and red balloons," *ACM SIGecom Exchanges*, vol. 10, pp. 5–9, dec 2011.

[23] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[24] S. Popov, "The Tangle." `https://www.iota.org/research/academic-papers`, April 30, 2018. Version 1.4.3. [Online; accessed 1 Feb 2019].

[25] E. Deirmentzoglou, "Rewriting History: A Brief Introduction to Long Range Attacks." `https://blog.positive.com/rewriting-history-a-brief-introduction-to-long-range-attacks-54e473acdba9`, 2018. [Online; accessed 1 Feb 2019].

[26] J. Bonneau, "Why buy when you can rent? - bribery attacks on bitcoin-style consensus," in *Financial Cryptography Workshops*, vol. 9604 of *Lecture Notes in Computer Science*, pp. 19–26, Springer, 2016.

[27] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *IEEE Symposium on Security and Privacy*, pp. 583–598, IEEE Computer Society, 2018.

[28] V. Buterin, "Ethereum 2.0 mauve paper." `https://cdn.hackaday.io/files/10879465447136/Mauve%20Paper%20Vitalik.pdf`, 2018. [Online; accessed 8 Feb 2019].

[29] N. Hutton, J. Maloberti, S. Nickel, T. F. Rønnow, J. J. Ward, and M. Weeks, "Design of a scalable distributed ledger." `https://fetch.ai/publications`, 2018. [Online; accessed 8 Feb 2019].

[30] T. Roughgarden, V. Syrgkanis, and E. Tardos, "The price of anarchy in auctions," *Journal of Artificial Intelligence Research*, vol. 59, pp. 59–101, 2017.

[31] D. E. Knuth, *The art of computer programming: sorting and searching*, vol. 3. Pearson Education, 1997.

[32] R. Lavi, O. Sattath, and A. Zohar, "Redesigning Bitcoin's fee market," *arXiv:1709.08881*, 2017.

[33] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the Instability of Bitcoin Without the Block Reward," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (New York, NY, USA), pp. 154–167, ACM, 2016.

[34] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.