Grassroots Consensus

Idit Keidar
Technion, Israel

Andrew Lewis-Pye
London School of Economics, UK

Ehud Shapiro
Weizmann Institute of Science, Israel
London School of Economics, UK

Abstract

Grassroots platforms aim to offer an egalitarian alternative to global platforms—centralized/autocratic and decentralized/plutocratic alike. Within the grassroots architecture, consensus is needed to realize platforms that employ digital social contracts, which are like smart contracts except that they are among people not accounts and are executed by these people's smartphones not by high-performance servers controlled by parties outside to the contract. Key envisioned grassroots platforms include sovereign democratic digital communities and federations, community banks and their grassroots cryptocurrencies, and digital cooperatives.

The grassroots architecture can benefit from a consensus protocol that is (*i*) quiescent, (*ii*) efficient during both low- and high-throughput, (*iii*) responsive, (*iv*) blocklace-based, (*v*) UDP-ready, and (*vi*) grassroots. The Grassroots Consensus protocol addresses all these requirements while having competitive performance in both low- and high-throughput scenarios and being one of the most concise and elegant consensus protocols for partial synchrony. It achieves that by building on two cutting-edge consensus protocols—the quiescent high-performance Morpheus and the blocklace-based Cordial Miners, improving the latter's dissemination protocol and making it UDP-ready, and extending the protocol with a constitution and a constitutional amendment component, making it grassroots.

2012 ACM Subject Classification Computer systems organization → Peer-to-peer architectures; Networks → Network protocol design; Networks → Formal specifications; Software and its engineering → Distributed systems organizing principles

Keywords and phrases Consensus Protocols, Grassroots Platforms, Blocklace, Digital Social Contracts, Democratic Digital Communities

1 Introduction

Grassroots. Grassroots platforms [44, 42, 40, 45, 46] aim to offer an egalitarian alternative to global platforms—centralized/autocratic and decentralized/plutocratic alike. A grassroots platform is different from a global platform in that it may have multiple instances that can operate independently of each other, without coordination or reliance on any third-party or global resource other than the network itself; furthermore, different instances of the same grassroots platform may interoperate and coalesce into ever-larger instances, possibly, but not necessarily, resulting in a single global instance.

Decentralized platforms for global cryptocurrencies [31, 52] and smart contracts [15, 9, 30] supporting Decentralized Finance (DeFi) [36, 21] and Decentralized Autonomous Organizations (DAOs) [19, 16, 49] often employ a consensus protocol as their foundation. Hence the surge in interest in high-performance and high-throughput consensus protocols [53, 23, 24, 27, 5, 22].

Grassroots platforms, on the other hand, are not consensus-based. Their foundational platform consists of a grassroots social graph implemented via a *blocklace*—a DAG-like generalization of the linear blockchain that is a universal Conflict-free Replicated Data Type (CRDT) [3]. The grassroots architecture [42, 44] facilitates the implementation of a grassroots social network [41] and grassroots cryptocurrencies [43, 27], which are also consensus-free, on top of the grassroots social graph. So where does consensus come into play?

Consensus. Within the grassroots architecture, consensus is needed to realize higher-level platforms that employ *digital social contracts*—the grassroots counterpart of smart contracts. Briefly, while a smart contract [15] is among anonymous accounts and runs on a consensus protocol executed by third-party servers, a social contract is among people known to each other and runs on a consensus protocol executed by the smartphones of these very same people [44, 46]. The two types of consensus-based contracts are compared in Table 1.

Table 1 Comparing Consensus-Based Smart Contracts and Social Contracts

	Smart Contract [15]	**Digital Social Contract** [11]
Among:	Accounts	People
Executed by:	Third-party servers	The people's smartphones
In return for:	Gas	Love
Applications:	Decentralized Finance (DeFi) [36, 21], Decentralized Autonomous Organizations (DAOs) [19, 16], Non-Fungible Tokens (NFTs) [51]	Grassroots democratic digital communities [33, 39] and their federation [44, 46], community banks and digital cooperatives [43]
Amended via:	Self-modifying code	Constitutional amendments

Example: Grassroots Federation. A key application of digital social contracts mentioned in Table 1 is the grassroots federation of digital communities. Grassroots Federation [46] aims to address the egalitarian formation and the fair democratic governance of large-scale, decentralized, sovereign digital communities, the size of the EU, the US, existing social networks, and even humanity at large. A grassroots federation evolves via the grassroots formation of digital communities and their consensual federation. Such digital communities may form according to geography, jurisdiction, affiliations, relations, interests, causes, and more. Small communities (say up to 100 members) govern themselves; larger communities—no matter how large—are governed by a similarly-small assembly elected by sortition [20, 46] among its members. It is expected that the constitutional democratic governance of each community would be specified by a digital social contract running on top of a grassroots consensus protocol, which in turn would be executed by the members of said small community or its assembly, one protocol instance per community. Such a consensus protocol should satisfy all the requirements explained next.

Requirements of Grassroots Consensus. Small communities engaged in democratic conduct, small community banks, small cooperatives, etc., would have a transaction rate much lower than network latency. Hence they would need a ***(i)* quiescent** protocol in which participants do nothing (in particular, send no messages) until there is a new transaction, and when subsequent transactions are issued at low-throughput, finalize each quickly and efficiently. As during low-throughput transactions are issued one-at-a-time, there is no symmetry to break or conflicts to resolve and hence—ideally—the agent issuing a transaction should be able to finalize it without the help of a leader, namely during low-throughput the protocol should be **leaderless**.

At the same time, the protocol should also work efficiently and with low-latency on smartphones in a high-throughput scenario. Namely, it should be an ***(ii)* efficient** consensus protocol during both low- and high-throughput. In addition, a standard requirement, termed ***(iii)* responsiveness**, is that in the good case the protocol operates at network speed, with performance unaffected by an overly-conservative estimation Δ of the actual least upper bound on network delay δ.

A fourth requirement relates to the blocklace: As the lower-level grassroots platforms (Social Graph and Social Networks [41], Cryptocurrencies [43, 27]) are blocklace-based, it would be very beneficial and integrable to also have a ***(iv)* blocklace-based** consensus protocol. This is not a limitation as there are plenty of efficient and high-throughout blocklace-based protocols already, starting from Cordial Miners [24] and its extensions and improvements [5, 22]. We note that a blocklace [3] is not "just a DAG" the same way a blockchain is not "just a sequence". In a blocklace, directed edges are realized by *signed cryptographic hash pointers*: A *p-block* (block by agent p) b includes a cryptographic hash of the rest of b, signed by p, which serves as its unique (whp) identifier, also referred to as a *pointer* to b. An edge from the p-block b to the q-block b' is realized by b including a pointer to b'. Thus, similarly to a blockchain, the blocklace is:

1. **Tamper-proof:** A bit in a block cannot be changed without this being detected;
2. **Non-repudiable:** The creator of a block cannot deny having created it;
3. **Acyclic:** Even Byzantine agents cannot create blocks that form/close a cycle.

However, unlike the blockchain, which is extended competitively, with the point-of-contention being adding the next block to the most-recent block, a blocklace is a naturally-cooperative data type, as agents may add blocks to the blocklace at will without conflicting with each other. In fact, the blocklace is a Universal Conflict-free Replicated Data Type (CRDT) [3].

The grassroots architecture in general, and the consensus protocol presented here in particular, are geared for smartphone-based execution. Namely, the intention is that the parties to a social contract will run both the contract and its underlying consensus protocol on their smartphones. The vagaries of the Internet—NATs and firewalls—and the fact that mobile phones change their IP address as they roam about, make phone-to-phone communication difficult. In particular, opening a direct TCP connection between two phones is either impossible, or impossible without the help of a proxy server, and even if established, it has to be re-established whenever one of the phones changes their IP address. Hence, UDP is a much preferable protocol for phone-to-phone communication, and is the one typically used for audio and video phone-to-phone conversations. Thus, the fifth requirement is that the protocol work not only on reliable networks, as typically assumed by consensus protocols, but also on unreliable networks namely be ***(v)* UDP-ready**.

The final sixth requirement for a protocol for grassroots platforms is, naturally, that the protocol be ***(vi)* grassroots**. Informally, this means two things: (*i*) That a group of agents P running the protocol can do so unscathed if embedded within a larger group of agents $P' \supset P$. Any permissioned consensus protocol satisfies that. (*ii*) That the protocol can do more if P is embedded within a larger group of agents P'.

Any platform that operates on a shared global resource or employs a global replicated (Blockchain [31]), or distributed (IPFS [7], DHT [35]) shared data structure, or distributed pub/sub systems with a global directory [13, 14, 8], are all not grassroots. Server-based federated systems such as Mastodon [34] are also not grassroots, informally due to the control exerted by each server-operator on its members; formally, since a group of people of any size cannot function without first connecting to such a server.

For a permissioned consensus protocol to be grassroots, the set of agents P needs to be "reconfigured" to include agents in $P' \setminus P$. The question of reconfiguration has been studied extensively for decades [26, 17, 2, 47]. While the original informal exposition of the problem in the Paxos paper was with respect to a self-governed parliament [26], subsequent works in the context of permissioned consensus assume that the initial 'permissioned' agents are determined by an external authority, and so is their 'reconfiguration'. Proof-of-Stake protocols such as Ethereum 2.0 offer a stake-based approach to reconfigure the participating agents on every epoch [25, 10].

A grassroots system cannot have external authorities and hence protocol reconfiguration must be entirely under the control of the very same agents that execute the protocol, as envisioned in the Paxos paper [26]. Furthermore, to be egalitarian rather than plutocratic, agents should have equal power in determining reconfiguration decisions. Specifically, in the grassroots context the challenge relates to the consensus protocol being executed by a self-governed community (the parties to the social contract running on top of the consensus protocol), allowing them to add or remove members at will, as well decide to change other aspects of its operation, for example the supermajority required to finalize a decision. To address that, we extend the protocol with a constitution that specifies the agents that execute the protocol as well as two key protocol parameters—the supermajority σ by which decisions, including constitutional amendment decisions, are binding, and a time Δ estimating the unknown least upper bound on network delay during synchrony, δ.

To summarize, the grassroots architecture can benefit from a consensus protocol that is **(*i*) quiescent, (*ii*) efficient during low- and high-throughput, (*iii*) responsive, (*iv*) blocklace-based, (*v*) UDP-ready, and (*vi*) grassroots**. The Grassroots Consensus protocol was conceived to address all these requirements.

Paper structure. Section 2 presents the Grassroots Consensus protocol assuming a reliable network, as well as a *prevailing constitution* (P, σ, Δ), that specifies the set of agents P, the supermajority σ among the agents by which the protocol operates, and a presumed upper-bound Δ on network delay during synchrony. Section 3 proves the safety and liveness of Grassroots Consensus. Section 4 extends the protocol for eventually-reliable networks, making it UDP-ready. Section 5 proves that the Grassroots Consensus protocol is indeed grassroots. Section 6 extends Grassroots Consensus with constitutional amendment, with which the agents may amend these three protocol parameters, and present democratic processes geared for amending each of the three parameters. Section 7 concludes.

2 Grassroots Consensus

2.1 Blocklace Preliminaries

We use $a \neq b \in X$ as a shorthand for $a \in X \land b \in X \land a \neq b$. We assume a set of *agents* Π, each endowed with a unique key-pair and identified by its public key $p \in \Pi$. While Π is potentially-infinite, we refer only to finite subsets $P \subset \Pi$ of it.

A *p-block* b, with $p \in \Pi$, is a triple $b = (h, x, H)$ where h is the hash of the pair (x, H) signed by p, referred to as the *identifier* of b, x is an arbitrary *payload*, which may be *empty*, $x = \bot$, in which case we refer to b as an *empty block*, and H a finite set of block identifiers, the blocks of which are *pointed to* by b, with b called *initial* if $H = \emptyset$. An encoding of such a triple $b = (h, x, H)$ via a sequence of bits in an agreed-upon form is referred to as a *well-formed block*. We assume the hash function is collision-free whp and cryptographic, so that blocks cannot form cycles and we may identify a block b with its identifier h. Furthermore, the method of signature allows the public key p to be recovered from a signature by p.

A block b *observes* itself, any block b' pointed to by b, and any block observed by b'. The 'observes' relation induces a partial order on any set of blocks. Two blocks that do not observe each other are *conflicting*, and if by the same agent they are *equivocating*. A block b *approves* a block b' if b observes b' and does not observe any block equivocating with b', and a set of blocks B *approves* b' if every $b \in B$ approves b'.

The *closure* of a block b, denoted $[b]$, is the set of blocks observed by b. If a block variable b is undefined then we deem its closure $[b] := \emptyset$ to be empty rather than undefined. For a set of blocks B the *closure* $[B]$ is defined by $[B] := \bigcup_{b \in B}[b]$, which is also the set of blocks *observed by* B. A set of blocks B is *closed* if $[b] \subseteq B$ for every $b \in B$, equivalently if $B = [B]$. A *blocklace* is a closed set of blocks.

To help avoid confusion, we use B below to denote a closed set of blocks (a blocklace) and D to denote any set of blocks, not necessarily closed. Given a set of blocks D, a pointer to a block b is *dangling* in D if $b \notin D$. Thus, a blocklace is a set of blocks with no dangling pointers. A block $b \in D$ is a *tip* of D if no other block $b' \neq b \in D$ observes b.

The *depth* $d(b)$ of a block b is 1 if b is initial else $d(b) = d(b')+1$ where b' is a maximal-depth block pointed to by b. The *depth* $d(B)$ of a set of blocks B is defined by $d(B) := \max_{b \in B} d(b)$. A *round* $r \geq 1$ in B is the set of all blocks $b \in B$ of depth $d(b) = r$, and the *r-prefix of* B, B_r, consists of all blocks $b \in B$ with depth $d(b) \leq r$, equivalently all rounds in B up to and including r. Note that, by definition, all blocks of the same round do not observe each other.

2.2 Protocol Concepts

Models. We consider two models. The first is the standard model of *eventual synchrony* (aka *partial synchrony)* that assumes a reliable network in which every message sent among correct agents is eventually received. Message latency is unbounded, but in every run there is a time called the *Global Stabilization Time (GST)*, which is unknown to the agents, after which all messages arrive within a least upper bound δ time for some $\delta > 0$. In particular, all messages sent any time before GST arrive by time GST+δ.

In the second model the assumption of a reliable network is relaxed. It assumes an *eventually-reliable network*, in which any message sent infinitely often between correct agents eventually arrives, and a message sent after GST between correct agents arrives within δ. This is the model assumed in the original Paxos paper [26] and many sequels. We refer to this model as the *UDP-ready* model.

Parameters. The protocol employs three key parameters (P, σ, Δ):

1. $P \subset \Pi$, a set of agents,
2. $\frac{1}{2} \leq \sigma < 1$, a fraction, used to specify a supermajority among the agents
3. $\Delta > 0$, a presumed upper-bound on message delay after GST, known to the agents.

We refer to the parameters (P, σ, Δ) employed by the protocol for a given block b or blocklace B, with values as above, as the *prevailing constitution* in b, resp. B; when the value of Δ is immaterial we may refer to the prevailing constitution (P, σ). We begin by assuming the prevailing constitution to consist of arbitrary constant values for all blocks during the protocol run, as is usually the case. Later, in Section 6, we augment the protocol with constitutional amendment blocks by which the prevailing constitution may be amended.

Supermajorities. Given agents $P \subset \Pi$, $|P| = n$, and $\frac{1}{2} \leq \sigma < 1$, a *$\sigma$-supermajority* among P is a fraction $Q \subseteq P$ such that $|Q| > \sigma n$, or, equivalently, that $\frac{|Q|}{|P|} > \sigma$. A set of blocks D is referred to as a *σ-supermajority* if there is a σ-supermajority $Q \subseteq P$ such that for every $q \in Q$, D includes at least one q-block.

Typically, a permissioned consensus protocol assumes that there are at most $f < n$ faulty agents among P. Given f, we say that a blocklace B is *f-safe* if it includes equivocations by at most f agents, a block b is *f-safe* if $[b]$ is f-safe, and define $\sigma := \frac{n+f}{2n}$. For example, if $f = 0$ then $\sigma = \frac{1}{2}$; if $f = \frac{1}{2}n$ then $\sigma = \frac{3}{4}$; if $n = 3f + 1$ (the standard assumption), then $\sigma = \frac{n + \frac{n-1}{3}}{2n} = \frac{2}{3} - \frac{1}{6n}$, namely a supermajority greater-or-equal to $\frac{2}{3}n$ is required.

Under this assumption, a σ-supermajority includes a majority of the correct agents, and hence the intersection of any two σ-supermajorities includes at least one correct agent. Typically, permissioned consensus protocols refer to “blocks by at least $n - f$ agents among P”, which is equivalent to “a σ-supermajority among P” in the standard case of $n = 3f + 1$.

Waves. A blocklace is seen as a sequence of waves, where the *k^{th}-wave* in B , $k \geq 1$, consists of three consecutive rounds r, $r + 1$, $r + 2$ in B, $r = 3(k - 1) + 1$, referred to as the *first*, *second*, and *third round* of the wave.

For a given $P \subset \Pi$, the partial function $leader_P$ assigns a leader $p \in P$ for the first round of every wave, where a first-round p-block b with prevailing constitution (P, σ) is a *leader block* if $p = leader_P(d(b))$. We assume the leader function to be *fair* in that for every wave and any $p \in P$ there is a subsequent wave for which p is the leader of its first round.

The consensus protocol progresses in waves. In each wave, a first round block is a candidate for finality in two cases—if it is a leader block or the only block in its round. These two possibilities typically arise in *high- and low-throughput* scenarios, respectively. During low throughput, agents quiesce, not producing a first-round block until they have a new transaction; and when an agent produces a first-round block b that does not conflict with others, all agents produce second and third round blocks that finalize b. During high throughput, conflicting blocks may be produced and the leader function is used to break symmetry; leader blocks are finalized and induce the total ordering of all blocks.

A second-round block b of a wave may approve any number of first-round blocks, but *endorses* (towards finality) at most one of them, defined as follows. Note that the next two definitions are mutually-recursive:

▶ **Definition 1** (Endorse). *Let b be a first-round block with prevailing constitution (P, σ) and b' a second-round block of the same wave in B. Then b'* ***endorses*** *b if the previous wave is:*

1. ***quiescent*** *and b is the only block in its round approved by b', or*
2. ***non-quiescent*** *and b is a leader block approved by b'.*

A set of blocks D endorses *b if every $b' \in D$ endorses b.*

A first-round block b is final if the third round of the same wave has a σ-supermajority that approves a second-round σ-supermajority that endorses b. Formally (text in parenthesis is explanatory, the definition is valid also without it):

▶ **Definition 2** (Ratified, Final, Ordered, Quiescent). *Given a blocklace B, a block $b \in B$ with prevailing constitution (P, σ) is:*

1. ***ratified*** *by a block b' if b' approves a σ-supermajority that endorses b;*
2. ***final*** *in B if there is a σ-supermajority in the third-round of b's wave in B, in which each block ratifies b, referred to as a **finalizing σ-supermajority**.*

A wave in such a blocklace B is:

1. ***finalizing*** *if it includes a (first-round) final block b; and*
2. ***quiescent*** *if in addition all the wave's blocks except perhaps b are empty and b does not conflict with any block in B.*

*As an initial condition, we refer to the (non-existent) wave that ends in round 0 as **quiescent**. A block b is* slow *if it is both finalized by a quiescent wave and follows a quiescent wave.*

Note that since only first-round blocks are endorsed, they are the only ones that can be ratified or final. We observe that the block properties defined above are *monotonic*:

▶ **Observation 3** (Monotonicity). *If a block is ratified or final in some blocklace B, it is also ratified or final, respectively, in any blocklace $B' \supset B$.*

Note, however, that quiescence is not monotonic, both because conflicting blocks may be added to a round, and because more rounds can be added to the blocklace.

The next observation asserts that a wave's final block is unique.

▶ **Observation 4** (Uniqueness). *A wave can have at most one ratified or final block.*

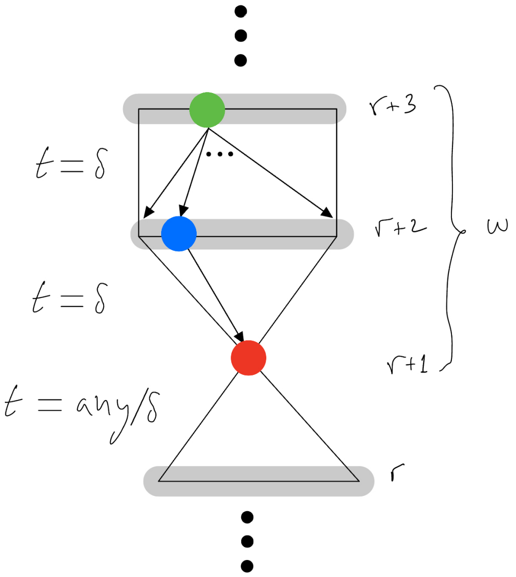
Quiescence, low- and high-throughput scenarios. The protocol is initially quiescent. It begins operations and continues to do so as long as agents have payloads to send. If multiple agents send payloads in the same wave, the protocol switches to high-throughput mode. It returns to being quiescent following a quiescent wave.

During low-throughput, in the good-case scenario, all waves are quiescent. A wave commences when some agent has a new transaction. This agent issues a (first-round) non-empty block, and all agents follow by issuing two rounds of blocks: Second-round blocks that endorse the first-round block, and third-round blocks each approving all second-round blocks, upon which the first-round block becomes final. The new wave is also quiescent provided all second- and third-round blocks are empty. The agent that issued a finalized, non-empty, first-round block in this scenario is referred to as a *spontaneous leader*, in contrast with the *formal leader* assigned by the $leader_P$ function.

During high-throughput, no wave is quiescent, and the good-case scenario unrolls as above, except that the first-round block is issued promptly by the wave's (formal) leader, whether or not it has any payload, and second- and third-round blocks are typically non-empty (contain transactions to be ordered by the next-wave's final leader block). The good-case scenarios are summarized in Figure 1.

Going beyond the good case, if during high-throughput agents fail to receive a first-round leader block in a timely manner, they issue first-round blocks of their own. Once a supermajority of agents do so, agents can continue to the next round without waiting for a leader block. And if during low-throughput there are multiple agents issuing first-round blocks, then it is possible that none of the first round blocks will become final. In this case, the wave is not quiescent and the protocol proceeds as in the high-throughput scenario.

To formalize these progress rules, we define a round r as *advanced* in a blocklace B if B includes enough blocks to allow round $r+1$ blocks to be added to the blocklace safely:



Round $r+1$ has a single leader block (red), round $r+2$ consists of a supermajority of blocks (blue as an example) pointing only to (and thus endorsing) the first-round leader block, and round $r+3$ consists of a supermajority of blocks (green as an example) each pointing to (and thus approving) all and only the second-round blocks, and thus ratifying the first-round leader block.

In the low-throughput scenario, the previous wave ending in round r is quiescent, the spontaneous leader block (red) may be produced at any time after the previous round r has advanced, and all other blocks are empty, thus the wave w is also quiescent.

In the high-throughput scenario, the formal leader produces a block (red) within δ after the previous round r is advanced, and all (or many) of the blocks are non-empty.

Figure 1 A good-case wave.

▶ **Definition 5** (Advanced Round). *Given a blocklace B with prevailing constitution (P, σ), a round $r > 0$ in B is* ***advanced*** *if r is a:*

1. *first round that either (i) includes a σ-supermajority of blocks among P, or (ii) includes a leader block or (iii) follows a quiescent wave and includes at least one block, or a*
2. *second or third round that includes a σ-supermajority of blocks among P.*

Round 0 is trivially advanced in all blocklaces.

Agents incorporate blocks in the blocklace only if they are valid, namely could have been produced by a correct agent during the run of the protocol. Formally, an ordinary block b is *valid* if it is well-formed and round $d(b) - 1$ is advanced in $[b]$.

Dissemination. The protocol uses three types of *dissemination-inducing blocks*: ACK, NACK, and NUDGE, defined below; other blocks are referred to as *ordinary.*

Dissemination-inducing blocks are not incorporated in the blocklace of their sender or recipient. Hence, they contribute only to the liveness of the protocol and do not affect its safety, which is derived from structural properties of the blocklace.

The Grassroots Consensus protocol we present in Section 2.3 assumes reliable links and uses NACK and NUDGE blocks for dissemination. Unlike Cordial Miners, which incorporates a blocklace-based dissemination protocol with $O(n^2)$ amortized communication complexity in the good case, Grassroots Consensus for this model achieves $O(n)$ amortized communication complexity in the good case during high-throughput. The UDP-ready protocol (given in Section 4) uses also ACK blocks for dissemination.

NACK blocks are used to achieve reliable broadcast of ordinary blocks. They are needed in order to address partial dissemination by Byzantine agents. For example, consider a Byzantine agent that sends a block b to correct agent p but not to another correct agent q, and p adds b into its blocklace and sends a block b' that depends on b. Then when q receives b', it cannot add it to its blocklace because its predecessor b is missing. To this end, q sends a NACK block to p specifying the blocks it is missing.

Note that out-of-order blocks may naturally arise on a reliable network even during periods of synchrony, even when there are no Byzantine agents, because there is no bound on how *quickly* messages can arrive — in the example above, b's sender might be correct and b may still be under way to q. We therefore introduce a delay before sending a NACK. Thus, in the good case, when the network is synchronous and there is no partial-dissemination by Byzantine agents, every correct agent sends its newly created blocks to all other agents and no further communication is needed.

The following definition relates to a setting in which an agent p has a blocklace B and it receives a new block b. If $[b] \not\subseteq B \cup \{b\}$, then p cannot incorporate b into B since some pointers in b are dangling in B. While obviously p cannot know all the blocks it does not know, it can know the boundary of its knowledge—where p's ignorance starts in relation to b. These are the tips of $[b] \setminus (B \cup \{b\})$. Note that if no block pointed to by b is in B, then the pointers in b point exactly to tips in $[b] \setminus (B \cup \{b\})$. Hence the following definition:

▶ **Definition 6** (NACK-block). *Given a blocklace B, a* NACK-***block** for a block $b = (h, x, H)$ is the block $b' = (h', x', H')$ where $x' = ($NACK$, h)$ and $H' \neq \emptyset$ contains the tips of $[b] \setminus (B \cup \{b\})$.*

Note that as defined above, $[b'] \subseteq [b]$. Also note that, given a block $b = (h, x, H)$ and a blocklace B, the tips of $[b] \setminus (B \cup \{b\})$ are observed directly from b: These are the pointers in H that are dangling in B, namely point to blocks not in B. These dangling pointers attest to the 'knowledge gap' of the recipient of b, namely, which blocks need to be added to B so that b will be in order following B. If there is no such gap, namely $[b] \subseteq B \cup \{b\}$, then the NACK-block for b and B is undefined.

In the reliable network model, the reason for the asymmetry of having NACK-blocks but not ACK-blocks is that any ordinary p-block is in fact an ACK-block, as it attests to all the blocks known to p at the time of its creation; ACK-blocks will be added in the UDP-ready protocol described in Section 4, when eventually-reliable networks are considered.

NACK blocks enable pull-based forwarding, where an agent who knows it misses a block asks for it. In addition, we make use of push-based forwarding to facilitate progress using NUDGE-blocks.

▶ **Definition 7** (NUDGE-Block). *Given a blocklace B, a* NUDGE-block *(h, x, H) for round $r + 1 > 1$ has payload $x =$ NUDGE and pointers H to the blocks of depth r in B.*

If an agent p suspects that the leader of the next round $r + 1$ does not proceed since it is missing some round r blocks, p sends the leader a NUDGE-block pointing to the r blocks in its blocklace. If the leader indeed misses any of these blocks, which after GST may happen due to Byzantine partial dissemination, it will respond to p with a NACK-block, which will in turn cause p to send to the leader the blocks it needs.

2.3 The Protocol

Here we present Grassroots Consensus for eventual synchrony with a reliable network. We prove the protocol's safety and liveness in Section 3, and present its UDP-ready extension in Section 4. In this section, we present the protocol using arbitrarily-fixed values for the parameters (P, σ, Δ). These can be changed using constitutional amendments, discussed in Section 6.

In the Grassroots Consensus protocol, each agent p maintains two sets of blocks: An input buffer D and a local blocklace B. The protocol is event-driven as described in Algorithm 1.

Overview. Each three consecutive rounds are grouped into a wave, which aims to finalize one block of its first round. Similarly to Morpheus [28], the protocol has two modes of operation, high-throughput and low-throughput, but here they are implicit— depending on whether the preceding wave is quiescent. In high-throughput mode, in the good-case scenario, only the leader sends a first round block. If the leader is faulty or does not communicate with other agents in a timely manner (bad-case scenario), other agents time-out on the leader and send blocks of their own. Agents wait either for a first-round leader block or for a supermajority of blocks in a round before producing a block in the next round.

Low-throughput is manifested by a quiescent wave, which finalizes its first-round spontaneous leader using two rounds of empty blocks. Following a quiescent wave, any agent may initiate a new wave by issuing a non-empty block. If two such blocks collide the protocol enters high throughput mode until, if ever, it quiesces again.

Algorithm 1 Grassroots consensus protocol, code for agent $p \in P$, given (P, σ, Δ).

```
Initialize: $D \leftarrow \phi$, $B \leftarrow \phi$, $payload \leftarrow \bot$.
    Let $r$ always be the maximal advanced round in $B$, which is initially $0$.
Receive: Upon receipt of $q$-block $b$, if $b$ a NACK-block then judiciously send $[b]$ to $q$,
    else if $b$ a NUDGE-block such that $p = leader(d(b))$ and $[b] \not\subseteq B \cup \{b\}$ then send
    once a NACK-block for $b$ to $q$,
    else if $b$ is valid then add $b$ to $D$.
Accept & Nack: If $[b] \subseteq B \cup \{b\}$ for some $b \in D$ then $B \leftarrow B \cup \{b\}$, $D \leftarrow D \setminus \{b\}$,
    else if a $q$-block $b'$ was added to $D$ more than $\Delta$ time ago then send once a
    NACK-block for $b'$ to $q$.
Issue: Issue once a new block if $r + 1$ is a second or third round or a first round that
    follows a wave that is:
    1. quiescent (or $r = 0$) and $payload \neq \bot$, or
    2. non-quiescent and $(i)$ $p = leader_P(r + 1)$ or $(ii)$ $r$ has been advanced for $9\Delta$.
Nudge: If $r$ is a third-round of a non-quiescent wave that has been advanced for $2\Delta$
    then send once to $leader_P(r + 1)$ a NUDGE-block for round $r + 1$.
Output: If $B$ contains a final block $b$ of depth greater than the previous final block
    (if any) then output every block in $\tau(b)$ unless it has already been output.

- issue a new block means create a block $b$ with $payload$ and pointers to the tips of
  $B_r$, add $b$ to $B$, send $b$ to every $q \neq p \in P$, and set $payload \leftarrow \bot$.
- judiciously send $[b]$ to $q$ means send every $b' \in [b]$ to $q$ unless $(i)$ $p$ has already
  sent $b'$ to $q$, or $(ii)$ $b' \in [b_q]$ for some $q$-block $b_q \in B \cup D$.
```

Delays. The protocol makes judicious use of delays, to ensure both liveness and efficiency. Here is the rationale for the various protocol delays, all expressed as multiples of Δ:

Δ for issuing nack-blocks. In the good case we wish no NACK-blocks to be sent at all. Agents wait Δ before complaining, so if a correct agent sends a block b to p and q at the same time, q may receive b up to Δ sooner than q and send a block b' that depends on b to p; but p will receive b within Δ from receiving b'.This delay is an optimization to avoid sending NACK-blocks in the good case; eliminating it does not hamper safety or liveness.

2Δ for nudging the ensuing round's leader. In the good case whence all agents are correct after GST, whenever an agent receives a block, all other agents receive it within Δ time. Thus, once a third round is advanced at the blocklace of a correct non-leader agent p, p expects to receive a leader block of the ensuing (first) round within 2Δ. If this does not happen, this could be because (*i*) it is before GST, (*ii*) the leader is incorrect, or (*iii*) the leader has not seen yet that the previous round is advanced. This last third case can happen after GST even if the leader is correct, due to Byzantine partial dissemination of third-round blocks. Since p knows that the previous third round is advanced, it sends a NUDGE-block to the leader, which will cause the leader to send a NACK-block requesting the blocks it is missing. This 2Δ delay is an optimization to avoid spurious nudges in the good case; eliminating it does not hamper safety or liveness.

9Δ for an unresponsive leader. Following a non-quiescent wave after GST, if the leader p_ℓ is correct, t all correct agents should wait for p_ℓ's first round block before proceeding to the second round in order to allow that block to be finalized. But because p_ℓ may be faulty, they need to eventually time-out and stop waiting. To determine the time-out, we consider the worst-case latencies as follows: Assume a third round r is advanced at p at time t. If p does not receive a leader block by $t + 2\Delta$, it nudges p_ℓ, which issues a NACK-block, upon receipt of which p sends p_ℓ any missing blocks. This exchange takes at most 3Δ, so by time $t + 5\Delta$, p_ℓ issues a leader block b for round $r + 1$, which is received by p by time $t + 6\Delta$. Upon receipt of b, p might be missing some blocks in $[b]$ and wait Δ before sending a NACK and getting them from q by time $t + 9\Delta$. If p does not accept any leader-block by $t + 9\Delta$, it concludes that either p_ℓ is faulty or GST has not arrived yet, and sends a first-round block of its own in order to allow the protocol to proceed to the next round.

Protocol walk-through. Incoming blocks are inserted to D; if any of their predecessors are missing, p waits Δ to issue a NACK, and once a block in D has no missing predecessors it is *accepted* to B and removed from B.

The variable r always holds the highest advanced round in B. If r is a first or second round in its wave, then p immediately (upon r advancing) issues a round $r + 1$ block. If r is a third round, then block issuing (in the ensuing wave) depends on the mode. During low throughput (initially and following a quiescent wave), p issues a block only if it has payload to send. During high-throughput, p issues a block in case (*i*) it is the leader, or (*ii*) it times-out on the leader.

An agent p always issues blocks that follow the latest advanced round p saw. This means that rounds can be skipped if the agent has been slow sending while the blocklace has advanced. This is useful when an agent that has been offline for some time catches up with the current blocklace before beginning to send blocks of its own.

After GST, whenever there is a wave with a correct leader following a non-quiescent wave, all correct agents accept and endorse the leader's first-round block (without timing out), and so no correct non-leader agent sends a first-round block. Therefore, no agent can produce a valid second round block that does not endorse the leader block. So the leader block is endorsed by all second round blocks, ratified by all third round blocks, and finalized.

Ordering with τ. Once the protocol finalizes a block b, it orders the predecessors of b that have not been ordered yet using the function τ. The definition of τ is similar to Cordial Miners: It starts from the most-recent final block in B and iterates through ratified blocks.

The output of $\tau(b)$ is a sequence that includes all blocks observed by b, with equivocating blocks excluded. In Section 3 we prove that τ is monotonic and safe. The monotonicity of τ ensures that consecutive calls to τ by a correct agent produce ever-increasing sequences, each a prefix of its successor. Hence, the protocol needs only output the increment of the output sequence of τ over its output from the earlier call, if any. The safety of τ means that blocks are ordered the same way at all correct agents.

In anticipation of the protocol being constitutional, we already assume the notion of the *prevailing constitution*. For now, assume (P, σ, Δ) is fixed arbitrarily for all blocks. In

Section 6, we define the constitution prevailing in a block, and have each call (initial and recursive) to τ with a block b employ the parameters of the constitution prevailing in b.

▶ **Definition 8** (τ). *We assume a fixed topological sort function $xsort(b, B)$ (exclude and sort) that takes a block b and a blocklace B, and returns a sequence that preserves the 'observes' partial order and includes all the blocks in B that are approved by b. The function τ takes a block as input and returns a sequence of blocks in $[b]$ as output, as follows:*

$$\tau(b) := \begin{cases} \tau(b') \cdot xsort(b, [b] \setminus [b']) & \textit{if } b' \textit{ is the maximal-depth block ratified in } [b] \\ & \quad \textit{according to the constitution prevailing in } b \\ xsort(b, [b]) & \textit{if no such } b' \textit{ exists} \end{cases}$$

When τ is called with a block b, it makes a recursive call with a block b' ratified in $[b]$. Note that b' is not necessarily final in $[b]$, nor final in the closure of the final block with which τ was called initially. In the implementation of τ, the results of recursive calls can be cached and used in subsequent calls, saving the need to recompute them.

Latency and complexity. Regarding latency, we note that after GST and in the good case a (formal or spontaneous) leader block is finalized by its wave within 3δ, during both low- and high-throughput (See Figure 1).

Regarding communication complexity, during low throughput, in the good case the spontaneous leader block is sent to all, constant-size endorsement blocks are sent all-to-all, and the culprit is the third round, where blocks each with $O(n)$-pointers are sent to all-to-all. In Morpheus [28], this issue is alleviated by having second-round blocks be replaced by threshold signatures on the first block (in low throughput mode): a σ-supermajority of these signatures can be amalgamated to form a single signature of constant length, and then third round blocks can be replaced with threshold signatures on that single signature, giving communication complexity $O(n^2)$. A similar idea can be applied here, by having hitherto-empty blocks carry a threshold signature instead. Specifically, a second-round block, instead of being empty, should carry as payload a threshold signature of the pointer to the first-round leader block. A third-round block that observes a supermajority of second-round blocks with threshold signatures, instead of being empty and pointing to all second-round blocks, should carry as payload an amalgamation of these threshold signatures, and point only to its own second-round block. Thus, in the good case, all blocks of a low-throughput wave are of constant size, resulting in $O(n^2)$ communication complexity. To accommodate this change, the definition of a third-round p-block b *ratifying* a first-round leader block b' has to include the case that b points only to the second round p-block, but its payload is an amalgamation of threshold signatures of a supermajority of second-round blocks that endorse b'.k

During high-throughput, all blocks carry payloads. If the payloads have at least $O(n)$ transactions each, then, in the good case, the amortized communication complexity per transaction is $O(n)$.

3 Safety and Liveness of Grassroots Consensus

Here we prove the safety and liveness of the Grassroots Consensus protocol.

We say that two sequences x and y are *consistent* if one is a prefix of the other, namely there is a sequence z such that $x \cdot z = y$ or $x = y \cdot z$, with $\cdot$ denoting sequence concatenation.

▶ **Definition 9** (Safety and Liveness of an Ordering Consensus Protocol). *An ordering consensus protocol is:*

Safe *if output sequences of correct agents are always consistent.*

Live *if every non-empty block issued by a correct agent is eventually included in the output of every correct agent.*

Next we prove safety and liveness of the basic Grassroots Consensus protocol.

3.1 Safety

▶ **Theorem 10.** *Grassroots Consensus is safe.*

To prove the theorem, we follow a condensed and simplified proof of the safety of Cordial Miners.

First, observe that Grassroots Consensus produces only f-safe blocks; if a block created by the protocol is not f-safe it means that its closure includes equivocations by more than f agents, a contradiction. For the following proofs we generalize τ to blocklaces, so that $\tau(B)$ is $\tau(b)$ if b is the most-recent final block in B if there is one, else it is the empty sequence Λ.

▶ **Definition 11** (τ Safety). *A blocklace B is τ-**safe** if every final block in B is ratified by every higher-depth block ratified by some block in B.*

▶ **Proposition 12.** *An f-safe blocklace is τ-safe.*

Proof. Let B be an f-safe blocklace, let b be a block of wave k final in B and b' a block of a later wave $k' > k$ ratified by some block in B, which means that b' is endorsed by a σ-supermajority in B. We have to show that b is ratified by b'.

Since b' is ratified by some block it means that it is valid, since correct agents do not include invalid blocks in their blocklace, let alone endorse them, and a σ-supermajority includes (a majority of) correct agents. Hence, being a first-round block of a wave, b' must observe a σ-supermajority of the last round of the previous wave, let's call it the blue supermajority.

Then the blue supermajority and the finalizing σ-supermajority of b, let's call it the green supermajority, must have a correct agent in their intersection. Namely, there is a correct agent p with a blue p-block that must observe a green p-block (which may actually be the same block if k' happens to be $k+1$), which in turn must observe a supermajority of blocks that endorse b. Since b' observes the blue p-block that observes the green p-block that ratifies b, then b is ratified by b'. ◀

▶ **Proposition 13** (τ Monotonicity). *τ is monotonic wrt $\supset$ over f-safe blocklaces.*

Proof. We have to show that if $B \subset B'$ for f-safe blocklaces B, B' then $\tau(B)$ is a prefix of $\tau(B')$. By Observation 3, all the blocks final in B are also final in B'. By Observation 4, each wave in B, B' includes at most one ratified or final block, hence the ratified and final blocks are totally ordered by their round number. If B, B' have the same most-recent final block, or if B has no final block, then the proposition holds trivially. Assume $b \neq b'$ are the most-recent final blocks in B, B', respectively, so necessarily b' is more recent than b. Let $b = b_1, \ldots, b_k = b'$, $k \geq 2$, be the sequence of all ratified blocks in B' between and including b and b'. By Proposition 12 and the definition of τ, $\tau(b')$ will eventually call recursively $\tau(b)$ and hence $\tau(b)$ is a prefix of $\tau(b')$, implying that $\tau(B)$ is a prefix of $\tau(B')$. ◀

▶ **Observation 14.** *If the union $B \cup B'$ of two blocklaces is f-safe then $\tau(B)$ and $\tau(B')$ are consistent.*

Proof. By monotonicity of τ over safe blocklaces, $\tau(B)$ and $\tau(B')$ are both prefixes of $\tau(B \cup B')$, which implies that one is a prefix of the other, namely they are consistent. ◀

Proof of Theorem 10. Since in a run of Grassroots Consensus there are at most f faulty agents in total, the union of the blocklaces of two correct agents is always f-safe. Hence, by Observation 14, the outputs of any two correct agents running Grassroots Consensus are always consistent. ◀

3.2 Liveness

Next, we discuss liveness. We first prove that Grassroots Consensus disseminates all correct agents' blocks and the blocks they depend on.

▶ **Proposition 15** (Dissemination). *If a block b is issued by a correct agent in Grassroots Consensus then $[b]$ is eventually received by all correct agents.*

Proof. By way of contradiction assume that $p, q \in P$ are correct, b_p is issued by p (and received by q since the network is reliable) and some block $b \in [b_p]$ is never received by q. Since by assumption b (and possibly other blocks in $[b_p]$) have not been received by q, then upon receipt of b_p, q issues a NACK-block b_q for which $b \in [b_q]$. Upon receipt of b_q by p, p will send (once) $[b_q]$, which includes b, to q. Since the network is reliable, q will eventually received b, a contradiction. ◀

The following observation follows directly from the definition of block validity:

▶ **Observation 16** (Sequential Advance). *If round r is advanced in a blocklace B, then every round $0 < r' < r$ is also advanced in B.*

We next show that the protocol does not get "stuck", i.e., agents continue to produce blocks when they are needed for progress. Because Grassroots Consensus is quiescent, agents do not have to generate infinitely many blocks. But if some agent issues a block in a round, that round eventually advances.

▶ **Proposition 17** (Advance). *If a correct agent p sends a block in some round r in Grassroots Consensus, then r eventually becomes advanced at every correct agent.*

Proof. Let p_1 be an agent that sends a block b in round r. Then when p_1 does so, $r - 1$ is advanced in p_1's blocklace.

Assume by way of contradiction that r does not advance at some correct agent p_2. By Observation 16, no blocks of depth greater than r are added to p_2's blocklace. By Proposition 15, this means that no correct agent sends blocks in rounds $> r$.

Moreover, because r does not advance at p_2, it must be the case that p_2 does not accept a supermajority of blocks in round r, which by Proposition 15 implies that there is some correct agent p_3 that does not send a round r block.

By Proposition 15, p_3 receives all the blocks in $[b]$. Therefore, p_3 accepts all these blocks into its blocklace. Thus, round $r - 1$ is advanced in p_3's blocklace.

In case r is a first round following a quiescent wave at p_3, r advances at p_3 upon receipt of b, and p_3 issues a round $r + 1$ block, a contradiction.

Otherwise, r is the maximum advanced round at p_3 and is not a first round following a quiescent wave at p_3. Therefore, p_3 issues a round r block, a contradiction. ◀

We next prove liveness for finite runs.

▶ **Proposition 18** (Finite implies quiescence). *Consider a run of Grassroots Consensus in which the blocklace B of a correct agent p is finite in a suffix of the run. Then B is quiescent.*

Proof. Let r be the last advanced round in B. (Note that round 0 is by definition advanced, so r is always defined). Assume by way of contradiction that B is not quiescent, then p eventually issues a round $r+1$ block b. By Proposition 17, $r+1$ eventually becomes advanced at p. A contradiction. ◀

▶ **Proposition 19** (Liveness in finite runs). *Consider a run of Grassroots Consensus in which the blocklace B of a correct agent is finite in a suffix of the run. Then all non-empty ordinary blocks sent by correct agents in this run are ordered in B.*

Proof. By Proposition 15, B includes all ordinary blocks sent by correct agents. By Proposition 18, B is quiescent. This means that its last wave includes a first round block b that is final in B, and all other blocks in the wave are empty. Because b is final, p calls $\tau(b)$. Also by assumption of B being quiescent, all non-empty ordinary blocks in B are in $[b]$, and hence ordered by $\tau(b)$. ◀

Next, we show that after GST, all waves that have correct leaders in high throughput mode are finalizing. Because quiescence is not monotonic, we consider here only waves k for which the preceding wave, $k-1$, is not quiescent at any correct agent at any time.

▶ **Proposition 20** (First round progress after GST). *Let r be a first round with a correct leader, let p be a correct agent at which round $r-1$ advances at time t>GST, and assume that the wave that ends in round $r-1$ is not quiescent in any correct agent's blocklace at any time. Then p sends a block in round r if and only if it is the round's leader. Moreover, p endorses a round r leader block in round $r+1$.*

Proof. Let p_ℓ be the correct leader of round r. If $p = p_\ell$ then once $r-1$ is advanced, p immediately issues a leader block for round r, causing round r to advance, and issues a block endorsing it in round $r+1$.

Otherwise, p waits for a block from p_ℓ until time $t+2\Delta$ and then sends p_ℓ a NUDGE-block, which is received by $t+3\Delta$, causing p_ℓ to send p a NACK-block and get the blocks it is missing by time $t+5\Delta$. Because round $r-1$ is advanced and its wave is not quiescent, p_ℓ sends a leader block, which is received by p by time $t+6\Delta$. Some predecessors of this block might be missing, causing p to wait Δ and then send a NACK-block and get the missing blocks allowing it to accept the leader block by time $t+9\Delta$.

When the leader block is received, the round advances and p's round $r+1$ block endorses the leader block. ◀

Below, we say that a round r *advances at time t* if there is some correct agent p such that round r is advanced in p's blocklace at time t and r is not advanced at any correct agent before time t.

▶ **Proposition 21** (Progress after GST). *Let r be a first round with a correct leader s.t. round $r-1$ advances after GST and the wave that ends in round $r-1$ is not quiescent in any correct agent's blocklace at any time. Then a round r leader block is finalized in its wave at all correct agents.*

Proof. By Proposition 20, the only correct agent that sends a round r block is the round's leader and all correct agents endorse this block in round $r+1$. This means that it is impossible to create a valid round $r+1$ block that does not endorse the leader, as such a block would have to point to a super-majority of round r blocks excluding the leader block. Therefore, every valid round $r+2$ block ratifies the leader block, and the block is finalized in round $r+2$ as soon as it advances at any correct agent. ◀

Next, we consider runs with infinitely many quiescent waves.

▶ **Proposition 22** (Infinite quiescent waves). *Consider a run of Grassroots Consensus in which infinitely many waves are quiescent in at least one correct agent's blocklace at some point in time. Then all the blocks of correct agents are ordered by all agents in this run.*

Proof. Let B be the (infinite) union of all blocklaces in the run. Let b_1 be a block of a correct agent in B. By Proposition 15, b_1 is eventually included in all correct agents' blocklaces. Therefore, it is approved by all correct agents' blocks from some wave onward. Because every quiescent wave is finalizing, and finalization is monotonic, there are infinitely many finalizing waves in B. Let b_2 be a final block that approves b_1 in some wave k. Once b_2 is final at some agent's blocklace, b_1 is ordered at that agent. We will show that b_2 is finalized at all agents.

Observe that since b_2 is finalized in wave k, it is approved by a super-majority in wave k. Because every valid first round block approves a super-majority in the preceding round and there is at least one correct agent at the intersection of any two super-majorities, every valid first round block in any round $k' > k$ observes b_2.

Consider a finalizing wave $k' > k$ in B and let b_3 be the first-round block that is finalized in wave k'. Then b_3 is endorsed by a super-majority in the second round. Because this super-majority includes correct agents, all correct agents eventually accept b_3 (by Proposition 15). If b_3 is not a leader block, then to be validly endorsed, $[b_3]$ must be quiescent, and so upon receiving b_3, all the correct agents learn that b_2 is finalized.

By the same token, if any non-leader first round block b is endorsed by a valid block of any agent in the second round of wave $k+1$, then $[b]$ is quiescent. This, in turn, means that every third round block in the wave either ratifies the leader block or observes a block whose closure is quiescent.

So when the third round of wave $k+1$ advances, every correct agent either observes a super-majority of blocks that ratify the leader block or observes a block whose closure is quiescent. In the former case the leader block is finalized, ordering all preceding ordinary blocks including b_2, and in the latter b_2 is already final in the block's closure. ◀

We are now ready to prove the liveness theorem.

▶ **Theorem 23.** *Grassroots Consensus is live.*

Proof. Let b be a non-empty ordinary p-block produced by a correct agent p in a run of Grassroots Consensus. By Proposition 15, every correct agent receives and incorporates b in its local blocklace.

We consider three cases: First, assume the run is finite. By Proposition 19, all non-empty ordinary blocks sent by correct agents in this run are ordered at all correct agents, and the theorem holds. Second, consider a run in which infinitely many waves are quiescent in at least one correct agent's blocklace at some point in time. By Proposition 22, b is ordered.

Finally, assume the run is infinite and contains only finitely many waves that are quiescent in at least one correct agent's blocklace at some point in time. Let k be a wave whose last round advances after GST and after which no wave is quiescent at any correct agent's blocklace. By fairness of leader selection, there are inifinitely many waves $k' > k$ with a correct leader. By Proposition 21, a leader block is finalized at all correct agents in each of these waves, ordering all non-empty ordinary blocks from earlier rounds. ◀

4 UDP-Ready Grassroots Consensus for Eventually-Reliable Networks

Grassroots platforms and protocols are geared for smartphone-based serverless implementation, and hence depend on smartphone-to-smartphone communication. Smartphones often reside behind NATs and Firewalls, which in turn are programmed with policies that make smartphones finding each other difficult. To overcome this, servers with known IP addresses (or domain names), running the STUN protocol help devices behind NATs find each other. However, once they do, establishing a direct TCP connection is difficult, and hence direct phone-to-phone communication typically employs UDP.

Grassroots Consensus (Algorithm 1) was specified assuming a reliable network. Here, we relax the assumption to eventually-reliable networks, and extend the protocol to be UDP-ready, by adding ACK-blocks and block resending. This protocol resends only ordinary blocks, as the reliable communication of dissemination-inducing blocks (NACK, ACK, and NUDGE) before GST is not required. Note that the protocol is very different from the trivial solution of "implementing TCP upon UDP", as during correct execution the same block can arrive via multiple paths and blocks may arrive in many possible orders. The protocol does not need to to impose an ordering on agent-to-agent block communication.

▶ **Definition 24** (ACK-block). *Given a block $b = (h, x, H)$, an ACK-**block** for b is the block $b' = (h', (\text{ACK}, h), \emptyset)$.*

The changes to make Grassroots Consensus UDP-ready are adding ACK to the Receive rule, adding resending to the NACK-rule, a new Resend rule, and revising the notion of *judiciously send*, as detailed in Algorithm 2 below (additions to Algorithm 1 are highlighted). In Algorithm 2, $k \geq 2$ is a parameter chosen to taste. In principle, it could be added as the fourth parameter of the constitution and amended similarly to Δ.

Algorithm 2 UDP-Ready Extension to Grassroots Consensus.

Receive: Upon receipt of q-block b, if b a NACK-block then judiciously send $[b]$ to q, else if b a NUDGE-block such that $p = leader(d(b))$ and $[b] \not\subseteq B \cup \{b\}$ then send once a NACK-block for b to q, else if b is valid then add b to D and send an ACK-block for b to q.

Accept & Nack: If $[b] \subseteq B \cup \{b\}$ for some $b \in D$ then $B \leftarrow B \cup \{b\}$, $D \leftarrow D \setminus \{b\}$, else if a q-block b was added to D more than Δ time ago, and no NACK-block for b was sent during the last $k\Delta$, then send a NACK-block for b to q.

Resend: If $k\Delta$ has passed since the most-recent ordinary p-block b was sent to q, no ACK-block or NACK-block was received for b, and $b \notin [b_q]$ for any q-block $b_q \in B \cup D$, then resend b to q.

- *judiciously send* $[b]$ *to* q means send every $b' \in [b]$ to q unless (i) an ACK-block or a NACK-block was received for b', or (ii) $b' \in [b_q]$ for some q-block $b_q \in B \cup D$.

Mobile devices. Smartphones are mobile, and when they move their IP address may change.

Hence, even if a phone-to-phone connection is established, it has to be re-established every time one of the phones changes their IP address. Blocklace dissemination can support mobile agents recovering each other's IP address via a joint stationary friend [41]. The problem is already present at protocols of lower-level than consensus such as the grassroots social graph [44] and the blocklace-based solution presented there [41] is applicable here as well.

Safety and liveness. The UDP-ready extension (Algorithm 2) to Grassroots Consensus only relates to dissemination-inducing blocks (an ACK-block and its handing) and does not affect ordinary blocks. Hence the definitions, propositions, and proofs regarding the safety of Grassroots Consensus—all stated in terms of ordinary blocks only—remain valid verbatim. Safety of the UDP-ready extension is therefore a corollary of Theorem 10.

Regarding liveness, prior to GST ordinary blocks issued by correct agents are resent to correct agents until received. Thus, there is a time $t \geq$ GST by which every block sent by a correct agent priot to GST either has been received, or has been resent after GST but before t. Thus, any message sent before t would be received by $t + \delta$. After GST, the two models under consideration are the same. Thus, the time t in the UDP-ready protocol has the same properties as GST in the Grassroots Consensus protocol: Any message between correct agents, if sent before t arrives by $t + \delta$, and if sent after t arrives within δ. Hence, the liveness proofs of Grassroots Consensus hold for the UDP-ready protocol, replacing GST by t thus defined, with the liveness of the UDP-ready protocol being a corollary of Theorem 23.

5 Grassroots Consensus is Grassroots

We recall the formal definition of a grassroots protocol from reference [44], instantiate it informally (bypassing the formalities of the definition) to the constitutional Grassroots Consensus protocol, and then argue that the protocol is grassroots.

▶ **Definition 25** (Oblivious, Interactive, Grassroots). *A protocol $\mathcal{F}$ is:*

1. ***oblivious** if for every $\emptyset \subset P \subset P' \subseteq \Pi$, a run of the protocol over P can proceed just the same in the presence of agents in P'.*
2. ***interactive** if for every $\emptyset \subset P \subset P' \subseteq \Pi$ and every run of the protocol over P, the protocol can proceed from any configuration c in r to interact with members in $P' \setminus P$, namely produce a run that is not possible only over P.*
3. ***grassroots** if it is oblivious and interactive.*

Note that the definition is stated in terms of any two populations $\emptyset \subset P \subset P' \subseteq \Pi$.

Being oblivious, in the context of Grassroots Consensus, means that if the prevailing populations of the protocol during a run are always included in P, then the presence of additional agents hanging around, those in $P' \setminus P$, which do not participate in the protocol, should not interfere with the protocol or affect its behaviour. Clearly, this is the case with Grassroots Consensus, as its prevailing populations may always choose not to amend the constitution to include members outside P. Hence Grassroots Consensus satisfies the requirement of being oblivious.

Being interactive, in the context of Grassroots Consensus, means that if the prevailing populations of the protocol in some prefix of a run ending in a configuration c are included in a set P, then the protocol has a computation starting from c that interacts with members outside P in a meaningful way, namely the protocol has a computation from c that reaches a configuration c', in which members of $P' \setminus P$ are included in the prevailing population, and thus c' cannot be reached if the prevailing populations are all restricted to P. Clearly, members of the prevailing population P, starting from configuration c, may amend the constitution to include in P a new member $p' \in (P' \setminus P)$ and thus reach a configuration c' in which $p' \notin P$ is a member. Hence Grassroots Consensus satisfies the requirement of being interactive. Together, these two properties imply that constitutional Grassroots Consensus is indeed grassroots. We thus conclude:

▶ **Theorem 26.** *Grassroots Consensus is grassroots.*

Note that the two requirements cannot even be expressed in the context of the basic (non-constitutional) Grassroots Consensus protocol, as the population P is given and fixed. Also note that the constitutional protocol being interactive critically-depends on the ability of the prevailing population P to decide to add new members.

6 Constitutional Grassroots Consensus

The problem of reconfiguration of a consensus protocol—namely changing the set of its participants—has engaged the distributed computing research community for decades [47, 17, 2, 47], ever since it was first formulated in Lamport's Paxos paper [26]. Here we explore the more general problem of constitutional amendment. Supporting constitutional amendments makes the protocol grassroots, as we formally prove in Section 5.

6.1 The Constitution

Generally, a *constitutional consensus protocol* is provided with an initial constitution that specifies an initial population, an initial consensus protocol, a rule for amending both, and a rule for amending the constitution itself. The notion of a constitution and its amendment is different from the well-studied notion of *reconfiguration* in at least two respects:

1. Reconfiguration typically refers to changing the set of active agents (servers, processes), whereas a constitution of a consensus protocol may address any aspect of it, including of course the agents executing it but also the protocol itself and its parameters.
2. Reconfiguration studies are not concerned with who makes reconfiguration decisions: They are typically assumed to be coming "out of the blue". Amending the constitution of a protocol, on the other hand, is done by the agents executing the protocol themselves, in accordance with the prevailing constitution's constitutional amendment rule.

Here, we consider a restricted type of *constitution* (P, σ, Δ), with $\emptyset \subset P \subset \Pi$ being the population, $\frac{1}{2} \leq \sigma < 1$ being a supermajority among the population the approval of which is needed to amend the constitution, which is also the supermajority used by the underlying consensus protocol, and Δ being the agreed-upon estimate of the prevailing population P regarding the delay needed to ensure the liveness of the protocol.

The three components of the constitution affect the safety, liveness, and performance of almost any consensus protocol for eventual synchrony. Even more so, P, σ, and Δ are key components of Grassroots Consensus, given that its primary aim is to execute digital social contracts, as explained next. The people P are the parties to the contract—members of the democratic digital community, community bank, cooperative, etc. Clearly, they should be sovereign to accept new members and remove existing members, namely to amend P.

As the level of knowledge and trust among the people P evolves, possibly as a result of amending P, so should σ be amended to reflect that: If the people in P conclude that they can trust each other more than initially anticipated, σ could be decreased. Adding new people to P, initially known to and trusted by only a subset of P, may require increasing

σ, even if temporarily. The parameter Δ is also highly-sensitive to the composition of the population P: If all have high-performance smartphones that are well-connected, Δ can be decreased; if new members are admitted with lower-capacity devices, Δ may need to be increased. In addition, technology advanced may allow decreasing Δ. Thus, the prevailing population P should be sovereign to amend the values of σ and Δ as they see fit.

6.2 Democratic Constitutional Amendment

Here we describe meta-level democratic processes for deciding on amending each of the components of a constitution (P, σ, Δ).

(1) Amending P. Regarding amending P to P', requiring σ-supermajorities among both P and P' to approve the amendment has been proposed before [17], to avoid, among other problems, the 'dead sailors problem' (in which all of P' are in fact dead, resulting in a constitutional stalemate) presented in the original Paxos paper [26].

Once P can be amended, the question of sybils (fake and duplicate identities) entering P, in addition to Byzantines, comes to the fore. First, since one Byzantine agent, having penetrated P with multiple identities, can cause as much harm as multiple Byzantine agents. In fact, the principal goal of permissionless protocols, both Proof-of-Work [32] and Proof-of-Stake [25, 10], is to augment consensus with mechanisms that make it sybil-resilient. Second, since a person controlling multiple digital identities in P—even if all correct at the consensus level—will have undue influence on the meta-level democratic processes, violating one person – one vote.

Existing large-scale digital communities suffer from extensive penetration of sybils. For example, Facebook removes around 1Bn fake accounts every quarter [48]. Most work on sybil detection and elimination employs top-down approaches suitable for global platforms [50, 18, 37, 12, 4, 6], but by their nature do not provide support for the democratic formation and conduct of digital communities. For grassroots consensus, egalitarian sybil-repelling protocols for the grassroots formation of digital communities [33, 39] may be more appropriate. They are based on mutual trust among the people forming a community and their willingness to backup this trust with mutual sureties. These protocols ensure, under certain conditions, that a digital community can grow indefinitely while retaining a bounded fraction of sybils.

A complementary line of work [38, 29], relevant to the meta-level democratic processes, shows how a community can employ sybil-resilient democratic governance: How the safety and liveness of digital democratic processes can be ensured despite a bounded penetration of sybils, even in the face of partial participation by its non-sybil members. Together, the two lines of work offer a comprehensive solution for (*i*) sybil-repellence during the egalitarian formation of digital communities and (*ii*) sybil-resilience in the democratic governance of digital communities, both needed during by the meta-level processes of constitutional amendment of Grassroots Consensus, as discussed next.

(2) Amending σ. Amending the supermajority by which a constitution can be amended has been studies by Abramowitz et al. [1], who showed that the h-rule (similar to the h-index of author citation metrics) is the only rule that satisfies certain intuitive and self-evident axioms. According to the h-rule, each agent in the prevailing population P votes by stating their most-preferred new value of σ. The value of σ is increased to the maximal $\sigma' > \sigma$ for which there is a σ'-supermajority among the prevailing population P who voted for a value greater or equal to σ'. Else σ is decreased to the minimal σ' for which there is a σ-supermajority among the prevailing population P who voted for a value less than or equal to σ'. Else σ remains unchanged.

(3) Amending Δ. The value of the parameter Δ should be increased or decreased only if the majority of the correct agents wish so. To this end, we use the *Suppress Outer-f* parameter update rule by Shahaf et al. [38][1], as follows: Each member of the prevailing population P votes by stating their most-preferred new value of Δ, through a process left unspecified. The vote on Δ should not be based on the subjective experience of each member of P, but rather on the perception of each member of P of the "common good".

If the median of the votes is larger than Δ, then the maximal f votes are discarded, the median of the remaining votes Δ' is calculated, and if $\Delta' > \Delta$ then the value of Δ is updated to Δ'. Symmetrically, if the median of the votes is smaller than Δ, then the minimal f votes are discarded, the median of the remaining votes Δ' is calculated, and if $\Delta' < \Delta$ then the value of Δ is updated to Δ'. Else Δ remains unchanged.

We note several desirable properties of the Suppress Outer-f rule: If all Byzantines wish to increase Δ to slow-down the protocol, and the correct agents do not, then Δ will not be increased. Similarly, if all Byzantines wish to decrease Δ to undermine the liveness of the protocol, and the correct agents do not, then Δ will not be decreased. If all correct agents unanimously wish to update Δ to Δ' then Δ will change to Δ' no matter what the Byzantine agents vote, provided $f < \frac{1}{3}n$. In particular, if the protocol is not making progress, it can be expected that all correct agents will vote to increase Δ and will succeed in doing so.

6.3 Grassroots Consensus with Constitutional Amendment

While we suggested specific meta-level democratic processes for amending the three components of the constitution, the safety and liveness of constitutional amendment does not depend on their specifics.

We assume that the constitutional amendment process produces a possibly-infinite ordered sequence of *constitutional amendment decisions* $d_1, d_2, \ldots$, where $d_i = \text{AMEND}(Id, i, P_i, \sigma_i, \Delta_i, s_i)$, $i \geq 1$, with s_i being a multisignature of $(Id, i, P_i, \sigma_i, \Delta_i)$ by some $Q_i \subset \Pi$, $P_i \subset \Pi$, $\frac{1}{2} \leq \sigma_i < 1$, $\Delta_i > 0$, and Id being the identifier of the instance of the constitutional consensus protocol (Grassroots Consensus in our case). The triple $(P_i, \sigma_i, \Delta_i)$ is referred to as its *constitution*. The constitution $(P_1, \sigma_1, \Delta_1)$ is referred to as the *initial constitution* and P_1 as its *founders*.

In a grassroots setup, the identifier of each instance of a grassroots consensus protocol should be unique in order to allow for multiple instances to operate concurrently without interference, even if they share some or all of the agents. This can be achieved, for example, by having the identifier Id of a protocol instance to consists of the founders P_1 together with the maximal serial number of any grassroots consensus protocol any founder has founded. This ensures that if P_1 has at least one correct agent, then Id would be unique.

▶ **Definition 27** (Valid Constitutional Amendment). *Given a sequence of constitutional amendment decisions $d_1, d_2, \ldots$, d_i is **valid** if $i = 1$ and $Q_1 = P_1$, or $i > 1$, d_{i-1} is valid, and Q_i is a $\max(\sigma_{i-1}, \sigma_i)$-supermajority in P_{i-1} and a $\max(\sigma_{i-1}, \sigma_i)$-supermajority in P_i. The sequence is **valid** if all its members are valid. Two non-identical constitutional amendment decisions are **conflicting** if they have the same index.*

Note that the two consecutive supermajorities are possibly overlapping, even identical. In the special case that only Δ_i changes, a σ_i-supermajority among P_i is needed; if only σ_i changes then a single $\max(\sigma_{i_1}, \sigma_i)$-supermajority among P_i is needed; and if only P_i changes then a σ_i-supermajority among P_{i-1} and a σ_i-supermajority among P_i is needed.

[1] In reference [38] the rule refers to 'outer-σ', however the definition of σ there is $\frac{f}{n}$, so we changed the name here to avoid confusion.

This definition ensures that the h-rule is followed and that the 'dead sailors' scenario of Paxos (that the new P' does not exists) [26] has not happened.

We assume that correct agents do not sign invalid or two conflicting constitutional amendment decisions, and hence can make the following observation.

▶ **Observation 28** (Safety of Constitutional Amendment). *Given a valid sequence of constitutional amendment decisions $d_1, \ldots d_i$, $i \geq 1$, if the number f of faulty agents in P_i among the total number of agents $n = |P_i|$ satisfies $\sigma_i \geq \frac{n+f}{2n}$, then the constitutional amendment process cannot extend this sequence with two conflicting valid constitutional amendment decisions.*

A constitutional amendment decision is enacted by the consensus protocol via a block that includes it as payload. To ensure orderly transition among constitutions, we assume that a correct agent would not sign d_{i+1} before observing that the amendment decision d_i has been finalized by the underlying consensus protocol, namely that a block containing d_i as payload has been finalized (as we explain below, only leader blocks carry constitutional amendments as payload). We exploit this in the consensus protocol by assuming that a d_{i+1} amendment decision cannot occur as a new payload of a correct agent p before a block with d_i is final in the blocklace of p.

▶ **Definition 29** (Valid Constitutional Amendment Block). *A block b is a **valid constitutional amendment block** if it is a valid initial block with an initial constitution d_1 as payload, in which case it is called a **founding constitution block**, or with a valid constitution d_i as payload, $i > 1$, and for which there is a valid constitutional block b' with d_{i-1} as payload and b' is final in $[b]$.*

▶ **Definition 30** (Constitutional, Founding, Prevailing, Pending). *A blocklace B is **constitutional** if it has a sole founding constitution block, with its constitution referred to as the **founding constitution** of B. Given a constitutional blocklace B, the **prevailing constitution** of B is the constitution of the most-recent final valid constitutional amendment block in B. The sole **pending constitution** of B, if any, is the constitution of the most-recent non-final valid constitutional amendment block in B, if there is one. When these terms apply to a block b they refer to the blocklace $[b]$.*

Importantly, we amend the definition of quiescence (Definition 2) to state that a blocklace with a pending constitution is *not quiescent*. Beyond that, Definitions 1, 2, and 5 are kept verbatim, with the prevailing constitution of each block interpreted as in Definition 30, and the σ-supermajority required for rounds to advance (in Definition 5) counted out of the prevailing constitution of the agent's entire blockchain.

The Grassroots Consensus protocol remains the same except in the way it assigns payloads to blocks, which is changed to facilitate constitutional amendments and avoid conflicting decisions during the transition between constitutions. This change is effected without modifying the Grassroots Consensus protocol (Algorithm 1), but through constraining the *payload* variable while there are pending amendments, as described in Algorithm 3 – an agent that knows of a constitutional amendment d_i that is not prevailing does not send the usual payload. If the agent is a round leader sending a leader block, it sets its *payload* to d_i, and otherwise it sets it to $\perp$.

The word *knows* in the Constitutional Amendment rule has a double meaning: Either p knows that the meta-level process of constitutional amendment has produced a new valid constitutional amendment decision d not yet prevailing in B, or p knows that its blocklace B includes a valid pending constitutional amendment block with payload d.

Algorithm 3 Grassroots Consensus with Constitutional Amendment.

```
Constitutional Amendment: If $p$ knows of a constitutional amendment decision $d$
    not prevailing in $B$ then prior to issuing a block of round $r+1$, if $p$ is the leader of
    round $r+1$ then $payload \leftarrow d$ else $payload \leftarrow \perp$ (retaining the outstanding value
    of $payload$, if not $\perp$, for the future).
```

To ensure liveness, an correct agent p that is removed by a constitution amendment d_i (namely, p is in d_{i-1} and not in d_i) must continue to participate in the protocol until it observes a final leader block b whose prevailing constitution is d_i. In particular, the constitution amendment d_i is final in $[b]$.

Excluding faulty agents. One natural application of constitutional amendment is excluding exposed faulty agents, in particular equivocators or agents that produce invalid blocks. The protocol is amended so that a correct agent p does not sign a constitutional amendment i if P_i includes an agent exposed as faulty in p's blocklace B. As a block known to a correct agent is eventually known to every correct agent, every agent exposed as faulty by one correct agent will eventually be exposed as such by every correct agent, and all exposed faulty agents will eventually be excluded from the prevailing population.

7 Conclusions

The Grassroots Consensus protocol and its extensions are novel in several respects, all devised in order to serve the needs of future grassroots platforms that operate digital social contracts. Initially, we expect low-throughput applications to prevail, as small digital communities engaged in constitutional democratic conduct use the platform to grow and evolve [46]. Even small community banks and cooperatives [43] may still be, at least initially, on the low-throughput side. As communities federate to form larger communities, and local community banks federate to form regional and national banks, higher-throughput execution would follow. Until then, simulating the various scenarios would be productive future work.

References

1 Ben Abramowitz, Ehud Shapiro, and Nimrod Talmon. In the beginning there were n agents: Founding and amending a constitution. In *Proceedings of ADT '21*, pages 119–131, 2021.

2 Marcos K Aguilera, Idit Keidar, Dahlia Malkhi, and Alexander Shraer. Dynamic atomic storage without consensus. *Journal of the ACM (JACM)*, 58(2):1–32, 2011.

3 Paulo Sérgio Almeida and Ehud Shapiro. The blocklace: A byzantine-repelling and universal conflict-free replicated data type. *arXiv preprint arXiv:2402.08068*, 2024.

4 L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi. Sok: The evolution of sybil defense via social networks. In *S&P '13*, pages 382–396, Los Alamitos, CA, USA, 2013. IEEE Computer Society.

5 Kushal Babel, Andrey Chursin, George Danezis, Lefteris Kokoris-Kogias, and Alberto Sonnino. Mysticeti: Low-latency dag consensus with fast commit path. *arXiv preprint arXiv:2310.14821*, 2023.

6 Leemon Baird. Swirlds and sybil attacks, 2016.

7 Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

8 Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52, 2009.

9 Vitalik Buterin. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

10 Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.

11 Luca Cardelli, Liav Orgad, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Digital social contracts: A foundation for an egalitarian and just digital society. In *CEUR Proceedings of the First International Forum on Digital and Democracy*, volume 2781, pages 51–60. CEUR-WS, 2020.

12 Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, 2005.

13 Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 109–118, 2007.

14 Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 14–25, 2007.

15 Primavera De Filippi, Chris Wray, and Giovanni Sileno. Smart contracts. *Internet Policy Review*, 10(2), 2021.

16 Qinxu Ding, Daniel Liebau, Zhiguo Wang, and Weibiao Xu. A survey on decentralized autonomous organizations (daos) and their governance. *World Scientific Annual Review of Fintech*, 1:2350001, 2023.

17 Danny Dolev, Idit Keidar, and Esti Yeger Lotem. Dynamic voting for consistent primary components. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 63–71, 1997.

18 John R Douceur. The sybil attack. In *Proceedings of the international workshop on peer-to-peer systems*, pages 251–260, 2002.

19 Ethereum. Decentralized autonomous organizations (DAOs) | ethereum.org, 2021, https://ethereum.org/en/dao. URL: `https://ethereum.org/en/dao/`.

20 Daniel Halpern, Ariel D Procaccia, Ehud Shapiro, and Nimrod Talmon. Federated assemblies. *Proc AAAI 2025; arXiv preprint arXiv:2405.19129*, 2024.

21 Johannes Rude Jensen, Victor von Wachter, and Omri Ross. An introduction to decentralized finance (defi). *Complex Systems Informatics and Modeling Quarterly*, (26):46–54, 2021.

22 Philipp Jovanovic, Lefteris Kokoris Kogias, Bryan Kumara, Alberto Sonnino, Pasindu Tennage, and Igor Zablotchi. Mahi-mahi: Low-latency asynchronous bft dag-based consensus. *arXiv preprint arXiv:2410.08670*, 2024.

23 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 165–175, 2021.

24 Idit Keidar, Oded Naor, and Ehud Shapiro. Cordial miners: A family of simple and efficient consensus protocols for every eventuality. In *37th International Symposium on Distributed Computing (DISC 2023)*. LIPICS, 2023.

25 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

26 Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998. `doi:10.1145/279227.279229`.

27 Andrew Lewis-Pye, Oded Naor, and Ehud Shapiro. Grassroots flash: A payment system for grassroots cryptocurrencies. *arXiv preprint arXiv:2309.13191*, 2023.

28 Andrew Lewis-Pye and Ehud Shapiro. Morpheus consensus: Excelling on trails and autobahns. *arXiv preprint arXiv:2502.08465*, 2025.

29 Reshef Meir, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Safe voting: Resilience to abstention and sybils. *arXiv preprint arXiv:2001.05271*, 2024.

30 Mayukh Mukhopadhyay. *Ethereum Smart Contract Development: Build blockchain-based decentralized applications using solidity.* Packt Publishing Ltd, 2018.

31 Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

32 Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

33 Ouri Poupko, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Building a sybil-resilient digital community utilizing trust-graph connectivity. *IEEE/ACM transactions on networking*, 29(5):2215–2227, 2021.

34 Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the decentralised web: The mastodon case. In *Proceedings of the internet measurement conference*, pages 217–229, 2019.

35 Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. Opendht: a public dht service and its uses. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 73–84, 2005.

36 Patrick Schueffel. Defi: Decentralized finance-an introduction and overview. *Journal of Innovation Management*, 9(3):I–XI, 2021.

37 Sven Seuken and David C Parkes. Sybil-proof accounting mechanisms with transitive trust. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 205–212. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

38 Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Sybil-resilient reality-aware social choice. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 572–579, 2019.

39 Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Genuine personal identifiers and mutual sureties for sybil-resilient community growth. In *International Conference on Social Informatics*, pages 320–332. Springer, 2020.

40 Ehud Shapiro. Grassroots distributed systems: Concept, examples, implementation and applications (brief announcement). In *37th International Symposium on Distributed Computing (DISC 2023). (Extended version: arXiv:2301.04391)*. LIPICS, 2023.

41 Ehud Shapiro. Grassroots social networking: Serverless, permissionless protocols for twitter/linkedin/whatsapp. In *OASIS '23*. Association for Computing Machinery, 2023. doi:10.1145/3599696.3612898.

42 Ehud Shapiro. A grassroots architecture to supplant global digital platforms by a global digital democracy. *arXiv:2404.13468, Proceedings of DAWO'24*, 2024.

43 Ehud Shapiro. Grassroots currencies: Foundations for grassroots digital economies. *arXiv preprint arXiv:2202.05619*, 2024.

44 Ehud Shapiro. Grassroots platforms with atomic transactions: Social networks, cryptocurrencies, and democratic federations. *arXiv preprint arXiv:2502.11299*, 2025.

45 Ehud Shapiro and Nimrod Talmon. Foundations for grassroots democratic metaverse. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 1814–1818, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems.

46 Ehud Shapiro and Nimrod Talmon. Grassroots federation: Fair governance of large-scale, decentralized, sovereign digital communities. *arXiv preprint arXiv:2505.02208*, 2025.

47 Alexander Spiegelman, Idit Keidar, and Dahlia Malkhi. Dynamic reconfiguration: A tutorial (tutorial). In *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, pages 2–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

48 Statistica. Facebook fake account removal, 2024, https://www.statista.com/statistics/1013474/facebook-fake-account-removal-quarter/.

49 Tom Taulli. Decentralized autonomous organizations (daos) governance for web3. In *How to Create a Web3 Startup: A Guide for Tomorrow's Breakout Companies*, pages 81–96. Springer, 2022.

50 Dinh Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *Proc. NSDI '09*, volume 9, pages 15–28, 2009.

51 Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. Non-fungible token (nft): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*, 2021.

52 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

53 Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proc. ACM PODC'19*, pages 347–356, 2019.