Characterising Global Platforms: Centralised, Decentralised, Federated, and Grassroots

Ehud Shapiro
Department of Mathematics and Data Science Institute
London School of Economics and Political Science, UK
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Israel

Abstract

Global digital platforms are software systems designed to serve entire populations, with some already serving billions of people. We propose atomic transactions-based multiagent transition systems and protocols as a formal framework to study them; introduce essential agents—minimal sets of agents the removal of which makes communication impossible; and show that the cardinality of essential agents partitions all global platforms into four classes:
(1) Centralised – one (the server)
(2) Decentralised – finite > 1 (bootstrap nodes)
(3) Federated – infinite but not universal (all servers)
(4) Grassroots – universal (all agents)
Our illustrative formal example is a global social network, for which we provide centralised, decentralised, federated, and grassroots specifications via multiagent atomic transactions, and prove they all satisfy the same basic correctness properties. We discuss informally additional global platforms—currencies, "sharing economy" apps, AI, and more.

While this may be the first characterisation of centralised, decentralised, and federated global platforms, grassroots platforms have been formally defined previously, but using different notions. Here, we prove that their original definition implies that all agents are essential, placing grassroots platforms in a distinct class within the broader formal context that includes all global platforms.

This work provides the first mathematical framework for classifying any global platform—existing or imagined—by providing a multiagent atomic-transactions specification of it and determining the cardinality of the minimal set of essential agents in the ensuing multiagent protocol. It thus provides a unifying mathematical approach for the study of global digital platforms, perhaps the most important class of computer systems today.

1 Introduction

Global digital platforms are software systems that aim to serve entire populations—all of humanity in some, entire nations in others. Global platforms have emerged as humanity's primary infrastructure for social interaction, economic exchange, and information flow. Despite their ubiquity and profound influence on billions of lives, we lack a mathematical framework to study their fundamental architectures.

Class	Essential Agents	Cardinality
Centralised	The server	One
Decentralised	Bootstrap nodes	Finite, > 1
Federated	All servers	Infinite, not universal
Grassroots	All agents	Universal

Table 1: Classes of global platforms by cardinality of essential agents

Our Framework. We propose atomic transactions-based multiagent transition systems and protocols [56] as a formal framework to study and characterize global platforms. In this framework, platforms are modelled as sets of agents whose interactions are governed by atomic transactions—indivisible state changes that can involve multiple participants.

To demonstrate the viability of this framework for specifying platforms of all four classes, we use a Facebook/X-like social network—conceived as centralised platforms—as our running example, providing atomic-transactions specifications of centralised, decentralised, federated, and grassroots social networks with feeds and followers.

Essential Agents: A Lens for Classification. We then introduce *essential agents*—minimal sets of agents the removal of which makes communication impossible (only unary transitions may occur). This concept provides a mathematical lens for understanding platform dependencies. The cardinality of essential agents in multiagent protocols provides the first formal characterisation of global platforms, partitioning them into four classes, as shown in Table 1.

Global Platforms in Practice. We review the "common knowledge" on global platforms of the four classes, exposing their fundamental importance and influence. See also Table 2.[1]

(1) **Centralised platforms** employ the centralised client-server architecture, where corporations control cloud servers that mediate all interactions among people who use them. Five companies command platforms with over one billion users each: Google/Alphabet (5Bn users; $3Trn market cap), Meta (3.8Bn; $2Trn), ByteDance (2Bn; $350Bn), Apple (1.5Bn; $4Trn), and Tencent (1.4Bn; $750Bn). Collectively, these companies have over $10Trn market value and reach over 5Bn people. Their governance is *autocratic*: one entity holds all decision-making power, resulting in a handful of people controlling together a substantial portion of the digital lives of more than half of humanity. Facebook, Twitter, Uber, and traditional web services exemplify this architecture, where a server with a designated identity (e.g. facebook.com) remains permanently essential for platform operation. This architecture enables what may be characterized as *corporate control* [65, 66].

[1] All numbers in the paper are approximate, as of November 2025; compiled from public reporting and industry trackers and cross-verified by AI.

Class	Control	Examples
Centralised	**Corporate/** Autocratic	Google Search (5Bn), YouTube (2.5Bn), Gmail (2.5Bn), Facebook (3Bn), Instagram (3Bn), WhatsApp (3Bn), Messenger (1Bn), TikTok (1.6Bn), Douyin (750M), iPhone (1.5Bn), WeChat (1.4Bn), X/Twitter (500M), Discord (200M) [21], Web servers, Database servers
Decentralised	**Capital/** Plutocratic	Bitcoin ($2.3Trn) [43], Ethereum ($500Bn) [13], Tether ($180Bn), BNB ($160Bn), Solana ($110Bn), (IPFS [7], DHT/Kademlia [39])
Federated	**Local Operators/** Cooperative	Mastodon (15M) [48], Matrix (80M) [38], ActivityPub networks, Email (SMTP servers)
Grassroots	**Participants/** Social Contracts [15] Democratic [30, 57]	Grassroots social networks* [53], Grassroots cryptocurrencies* [54], Grassroots federations* [57], Grassroots Logic Programs* [55], Scuttlebutt [58], BitTorrent [60] * Mathematically specified, yet to be implemented

Table 2: Classification of global platforms, with user counts and market valuations where available.

(2) **Decentralised platforms** include global cryptocurrency platforms that employ the blockchain architecture, where distributed networks of nodes maintain replicated ledgers through consensus protocols while users access services through wallets and decentralised applications. Bitcoin [43] ($2.3Trn market cap) and Ethereum [13] ($500Bn) exemplify this architecture. The top five platforms collectively exceed $3Trn in market value. While envisioned as decentralised alternatives to centralised financial systems, in practice cryptocurrencies exhibit extreme concentration of both ownership and control: Regarding ownership, less than 5% of cryptocurrency holders, which is less than 0.25% of the human population, own more than 90% of their value, exacerbating an already-unprecedented concentration of wealth. Regarding control, it is not only *plutocratic*, but also far from being decentralised: In Bitcoin the top two mining pools control 55% of hashrate, and in Ethereum the top four operators control 50% of staked ETH. In both mechanisms, control is realized via capital—computational resources (PoW) or staked assets (PoS). We may term this *capital control* [14].

(3) **Federated platforms** employ a distributed server architecture where multiple independent servers interoperate through shared protocols, with users attached to their chosen home server. Examples include Mastodon [48] (15M users across 10,000 servers), Matrix [38] (80M users), and the broader ActivityPub ecosystem. While being more distributed than centralised platforms, federated systems maintain a fundamental architectural distinction between servers and clients. Users cannot function without their home server, and server operators exercise control over their domains—deciding moderation policies, user access, and federation relationships. This may be characterized as *control by local operators*.

(4) **Grassroots platforms** aim to provide an *egalitarian, cooperative and democratic* alternative to centralised/autocratic and decentralised/plutocratic global platforms [52, 56]. Unlike the global platforms that may only have a single instance (one Facebook, one Bitcoin), grassroots platforms enable any agent to initiate an instance, have multiple instances operate independently, and have instances coalesce, possibly (but not necessarily) resulting in a single global instance. Grassroots platforms were specified for social networks [53], cryptocurrencies [32, 54] and democratic federations [32, 57]. Existing grassroots platforms include Scuttlebutt [58] and the original BitTorrent [60]

Contributions. Our introduction of essential agents as a mathematically-founded notion provides two fundamental contributions to the study of global platforms. The cardinality of minimal sets of essential agents yields a comprehensive and mutually exclusive characterisation of all four known platform classes—centralised (one), decentralised (finite, > 1), federated (infinite, but not universal), and grassroots (universal). This is the first formal characterisation of centralized, decentralized, and federated global platforms, now unified with grassroots computing within a single mathematical framework. While grassroots platforms have been formally characterized previously using different notions, we show that their definition implies that all agents are essential.

Paper organization. Section 2 introduces atomic transactions, establishes correctness properties that social networks should satisfy, and provides transaction-based specifications of platforms across all four categories, using social networks as a running example. We prove in Appendix A that all four specifications satisfy these properties despite their architectural differences. Section 3 presents the theory of atomic transactions-based multiagent transition systems and multiagent protocols; formally defines essential agents; proves the equivalence between having all agents essential and being interactive, and hence grassroots; and establishes the comprehensive characterisation. Section 4 reviews additional global platforms of all classes, Section 5 reviews related work, and Section 6 concludes.

2 Multiagent Atomic Transactions-Based Specifications of Global Social Networks

Here we recall the notion of multiagent atomic transactions [56], and use it to provide specifications for a Facebook/X-like social

network with feeds and followers for each platform class, exposing their architectural differences.

2.1 Multiagent Atomic Transactions

We assume a potentially infinite set of agents Π (think of all the agents that are yet to be born), but consider only finite subsets of it, so when we refer to a particular set of agents $P \subset \Pi$ we assume P to be nonempty and finite. We use $\subset$ to denote the strict subset relation and $\subseteq$ when equality is also possible.

In the context of multiagent transition systems it is common to refer to the state of the system as *configuration*, so as not to confuse it with the *local states* of the agents. As standard, we use S^P to denote the set S indexed by the set P, and if $c \in S^P$ is a configuration over S and P, we use c_p to denote the member of c indexed by $p \in P$.

Definition 2.1 (Local States, Configuration, Transaction, Active & Stationary Participants). Given agents $Q \subset \Pi$ and an arbitrary set S of *local states*, a *configuration* over Q and S is a member of $C := S^Q$. An *atomic transaction* over Q and S is a pair of configurations $t = c \to c' \in C^2$ s.t. $c \neq c'$, with $t_p := c_p \to c'_p$ for any $p \in Q$, and with p being an *active participant* in t if $c_p \neq c'_p$, *stationary participant* otherwise. A set of transactions R is *over* Q and S if every $t \in R$ is over some $Q' \subseteq Q$ and S.

Definition 2.2 (Degree, Concise Set of Transactions). Given agents $P \subset \Pi$, local states S, and a set of transactions T over some P and S, the *degree* of $t \in T$ (unary, binary,…k-ary) is the number of active participants in t, and the *degree* of T is the maximal degree of any $t \in T$. A stationary participant $q \in Q$ in a transaction $t \in T$ over $Q \subseteq P$ is *redundant* in t (given T) if for $\forall s \in S\ \exists t' \in T$ such that $t'_q = s \to s$ and $t_p = t'_p$ for every $p \neq q \in Q$. A set of transactions is *concise* if it has no transactions with redundant participants.

2.2 Correctness of a Social Network

The social network we specify enables agents to post messages and follow other agents to receive their posts. Regardless of architecture, any specification of such network should guarantee two fundamental properties:

(1) **Follower Correctness**: If agent p follows agent q, then p sees exactly q's posts in the correct order and eventually sees all of q's posts.
(2) **Agent Autonomy**: An agent can always post a message and follow any other agent.

To ensure liveness properties hold, each platform specification includes weak fairness requirements (stating that a continuously-enabled transaction must be taken) that capture the necessary progress conditions without forcing voluntary agent actions.

We next present social network specifications for each platform class—centralised, decentralised, federated, and grassroots. As the formal specification of these properties depends on the introduction of the notion of a multiagent protocol (Section 3), we prove that all four specifications satisfy these properties in Appendix A.

2.3 Centralised Platforms

Centralised platforms employ a named server agent that all other agents must interact with. We illustrate this with a social network where a central server maintains feeds for all registered users. Users can post locally but need the server to exchange content. For a post by p to reach q the following transactions have to take place: (1) p registers with the server (2) q registers with the server (3) q follows p (4) p posts (5) p syncs with the server (6) q syncs with the server. Subsequent posting by p require only the last three transactions.

Definition 2.3 (Centralised Social Network Transactions). Given agents $P \subset \Pi$ with a named server $p \in P$ and users $P \setminus \{p\}$. Local states are sets of feeds, where a feed is a pair $(agent, posts)$ with $posts$ a sequence. Initial local state is $\emptyset$. The transactions are $c \to c'$ where:

- **Register** over $\{p, q\}$:
  - $c_q = \emptyset$ and $c'_q := \{(q, \Lambda)\}$
  - $c'_p := c_p \cup \{(q, \Lambda)\}$
- **Post**(m) over $\{q\}$: $c'_q := c_q$ with $(q, posts) \in c_q$ amended to $(q, posts \cdot m) \in c'_q$.
- **Follow**(q') over $\{q\}$: $c_q \neq \emptyset$, $(q', \cdot) \notin c_q$, and $c'_q := c_q \cup \{(q', \Lambda)\}$
- **Sync** over $\{q, p\}$: $(q, \cdot) \in c_p$ and:
  - $c'_p := c_p$ with q's posts missing in c_p added to $(q, posts)$
  - $c'_q := c_q$ with posts by followed agents missing in c_q added

2.4 Decentralised Platforms

Bitcoin. We illustrate decentralised platforms with bootstrap agents using an abstract Bitcoin-like protocol. In particular, we do not specify preconditions for the nondeterministic AddBlock transaction. While decentralised systems typically distinguish between full nodes and lightweight nodes, we abstract away this distinction since any node can in principle become a full node. The genesis block and bootstrap node addresses are public information hard-coded in the protocol.

Definition 2.4 (Bitcoin Transactions). Given a constant set of bootstrap agents $B \subset \Pi$ and agents $P \subset \Pi$. Local states S are pairs (blockchain, peers) where blockchain is a sequence of blocks and peers is a set of agents, and for configuration c over S and P, we let $c_p = (chain_p, peers_p)$. Initial local state is (g, B) where g is the genesis block. Transitions include $c \to c'$ where:

- **AddBlock**(b) over $\{p\}$: $chain_p \neq \Lambda$ and $c'_p := (chain_p \cdot b, peers_p)$.
- **Sync** over $\{p, q\}$: $q \in peers_p$, $|chain_p| < |chain_q|$ and:
  - $|chain'_p| := |chain'_q| := |chain_q|$,
  - $peers'_p := peers'_q := peers_p \cup peers_q \cup \{p\}$

If the chain of p is not a prefix of the chain of q upon a Sync over $\{p, q\}$, then the blocks in p that are inconsistent with the chain in q are *abandoned* and we say that the chain of p has undergone *reorg*. The correctness of the protocol depends on the probability of an AddBlock transaction being much lower than the probability of a Sync transaction [25, 45]. This in turn implies that the probability of a reorg abandoning a block diminishes quickly as a function of how deep the block is "buried", namely how many blocks follow it in the blockchain [27, 42]. Moreover, the probabilistic argument underscores the importance of the bootstrap nodes: It depends on them forming a connected component and on all active agents joining this component; the argument falls apart if multiple connected components form and grow independently [5, 28].

Decentralised Social Network. Each agent maintains their own posts and publishes them directly to the blockchain. Chain reorganizations require rolling back the publication pointer for posts in abandoned blocks.

For a post by p to reach q the following unbounded sequence of transactions have to take place: (1) p posts block b (2) a sequence of Sync transactions in which b is part of the longer chain, the last of which is with q. If a Sync transaction with p entails a reorg in which the block b is abandoned , it has to be posted again.

Definition 2.5 (Decentralised Social Network Transactions). Given agents $P \subset \Pi$ with bootstrap nodes $B \subset P$. Local state is $(blockchain, peers, posts, pointer)$ where blockchain is a sequence of blocks, peers is a set of agents, posts is a sequence of posts, and pointer indexes published posts, and for configuration c over S and P, we let $c_p = (chain_p, peers_p, posts_p, pointer_p)$. Initial local state is $(g, B, \Lambda, 0)$. The transactions are $c \to c'$ where:

- **Post**(m) over $\{p\}$: $c'_p := (chain_p, peers_p, posts_p \cdot m, pointer_p)$.
- **AddBlock**(b) over $\{p\}$: $chain_p \neq \Lambda$, b contains posts from $pointer_p$ to $|posts_p|$, then $c'_p := (chain_p \cdot b, peers_p, posts_p, |posts_p|)$.
- **Sync** over $\{p, q\}$: $q \in peers_p$, $|chain_p| < |chain_q|$ and:
  - $|chain'_p| := |chain'_q| := |chain_q|$
  - $peers'_p := peers'_q := peers_p \cup peers_q \cup \{p\}$
  - $pointer'_q := pointer_q$, $pointer'_p$ is set to the number of posts from $posts_p$ published in x, where x is the maximal prefix of $chain_p$ that is consistent with $chain_q$

While several proposals exist for blockchain-based social networks [12, 16, 17, 47], our specification exposes fundamental architectural mismatches:

- **Replication**: All posts must be replicated to all agents and stored by all agents, regardless of who follows whom
- **Consensus**: Every post requires global consensus.
- **Instability**: Posts may be reordered after publication and responses may appear before their referents.

Blockchain efficiency can be improved by sharding [31, 61], layering [20, 46], and side channels [22, 41], but the issues raised above cannot be addressed short of operating the social network via a different architecture. Perhaps the key reason for that is that social networks do not need consensus to operate (see the grassroots social network protocol below, Definition 2.7), so there is no reason to pay the price for consensus in order to realise them.

2.5 Federated Platforms

Federated Social Network. Multiple servers cooperate through federation. Users bind to home servers with qualified identities (e.g., p@s). Federation occurs when users follow remote users.

For a post by p to reach q the following eight transactions have to take place: (1) p registers with server r (2) q registers with server s (3) $q@s$ follows $p@r$ (4) $q@s$ syncs with the server s. (5) $p@r$ posts (6) $p@r$ syncs with the server r (7) server r syncs with server s (8) $q@s$ syncs with server s. Subsequent posts by p require only the last four transactions.

Definition 2.6 (Federated Social Network Transactions). Given agents $P \subset \Pi$ with designated servers $Q \subset P$ and clients $P \setminus Q$. Local states are sets of feeds, where a feed is $(agent@server, posts)$ with $posts$ a sequence. Initial local state is $\emptyset$. Transactions are $c \to c'$ where:

- **Register** over $\{p, s\}$: $c_p = \emptyset$, $c'_p := \{(p@s, \Lambda)\}$, and $c'_s := c_s \cup \{(p@s, \Lambda)\}$
- **Post**(m) over $\{p\}$: $c'_p := c_p$ with $(p@s, posts) \in c_p$ amended to $(p@s, posts \cdot m) \in c'_p$.
- **Follow**$(q@s')$ over $\{p\}$: $c_p \neq \emptyset$, $(q@s', \cdot) \notin c_p$, and $c'_p := c_p \cup \{(q@s', \Lambda)\}$
- **Sync** over $\{a, b\}$:
  - If $a \in P \setminus Q$ and $b \in Q$ where $(a@b, \cdot) \in c_b$:
    - $c'_b := c_b$ with $(a@b, posts)$ updated to include any posts from c_a
    - $c'_a := c_a$ with feeds of followed users updated from c_b's copies
  - If $a, b \in Q$ and $a \neq b$: For each $(p@a, \cdot) \in c_b$ or $(q@b, \cdot) \in c_a$:
    - If $(p@a, posts) \in c_a$ and $(p@a, posts') \in c_b$: update to longer sequence in both
    - If $(q@b, posts) \in c_b$ and $(q@b, posts') \in c_a$: update to longer sequence in both

2.6 Grassroots Platforms

Grassroots Social Network. For a post by p to reach q the following five transactions have to take place: (1) p initialises (2) q initialises (3) q follows p (4) p posts (5) p syncs with q. Subsequent posting by p require only the last two transactions.

Definition 2.7 (Grassroots Social Network Transactions). Given agents $P \subset \Pi$. Local states are sets of pairs $(agent, posts)$ where $posts$ is a sequence. Initial local state is $\emptyset$. Transactions are $c \to c'$ where:

- **Initialise** over $\{p\}$: $c_p = \emptyset$ and $c'_p := \{(p, \Lambda)\}$
- **Post**(m) over $\{p\}$: $(p, s) \in c_p$ and $c'_p := c_p \setminus \{(p, s)\} \cup \{(p, s \cdot m)\}$
- **Follow**(q) over $\{p\}$: $p \neq q$, $(q, \cdot) \notin c_p$, $c'_p := c_p \cup \{(q, \Lambda)\}$
- **Sync** over $\{p, q\}$: for any r with $(r, s) \in c_p$ and $(r, s') \in c_q$:
  - If $|s| < |s'|$: update (r, s) to (r, s') in c'_p
  - If $|s'| < |s|$: update (r, s') to (r, s) in c'_q

3 Classifying Global Platforms with Essential Agents

Here we recall definitions and results from [56], introduce the key novel notion of essential agents, and use it to classify global platforms specified by multiagent protocols.

3.1 Atomic Transactions-Based Multiagent Transition Systems and Protocols

The following definition uses 'configuration' rather than the more standard 'state' to avoid confusion with the 'local state' of agents in a multiagent transition system.

Definition 3.1 (Transition System). A *transition system* is a tuple $TS = (C, c0, T)$ where:

- C is an arbitrary set of *configurations*
- $c0 \in C$ is a designated *initial configuration*
- $T \subseteq C \times C$ is a *transition relation.* A transition $(c, c') \in T$ is also written as $c \to c' \in T$.

A transition $c \to c' \in T$ is *enabled* from configuration c. A configuration c is *terminal* if no transitions are enabled from c. A *computation* is a (finite or infinite) sequence of configurations where for each two consecutive configurations (c, c') in the sequence, $c \to c' \in T$. A *run* is a computation starting from $c0$, which is *complete* if it is infinite or ends in a terminal configuration.

Given agents $P \subset \Pi$, local states S with initial state $s0 \in S$, we denote configurations as $C := S^P$ and the initial configuration as $c0 := \{s0\}^P$.

A *multiagent transition* over P and S is but an atomic transaction over P and S. However, a *set of multiagent transitions* over P and S has every transition exactly over P, not, as in transactions, over some $Q \subseteq P$. Thus, a transaction over $Q \subseteq P$ defines a (typically infinite) set of multiagent transitions over P in which all members of $P \setminus Q$ are stationary in arbitrary states:

Definition 3.2 (Transaction Closure). Let $P \subset \Pi$, S a set of local states, and $C := S^P$. For a transaction $t = (c \to c')$ over local states S with participants $Q \subseteq P$, the *P-closure of t*, $t{\uparrow}P$, is the set of transitions over P and S defined by:

$$t{\uparrow}P := \{t' \in C^2 : \forall q \in Q.(t_q = t'_q) \wedge \forall p \in P \setminus Q.(p \text{ is stationary in } t')\}$$

If R is a set of transactions, each $t \in R$ over some $Q \subseteq P$ and S, then the *P-closure of R*, $R{\uparrow}P$, is the set of transitions over P and S defined by:

$$R{\uparrow}P := \bigcup_{t \in R} t{\uparrow}P$$

Namely, the closure over $P \supseteq Q$ of a transaction t over Q includes all transitions t' over P in which members of Q do the same in t and in t', and the rest remain in their current (arbitrary) state.

Note that while the set of multiagent transitions $R{\uparrow}P$ in general may be over a larger set of agents than R, T and R are of the same degree, by construction. Also, note that while all transitions of a multiagent transition system over P are also over P, each transaction in a set of transactions is typically over a different set of participants. Thus, a set of transactions R over S, each with participants $Q \subseteq P$, defines a multiagent transition system over S and P as follows:

Definition 3.3 (Transactions-Based Multiagent Transition System). Given agents $P \subset \Pi$, local states S with initial local state $s0 \in S$, and a set of transactions R, each $t \in R$ over some $Q \subseteq P$ and S, the *transactions-based multiagent transition system* over P, S, and R is the transition system $TS = (S^P, \{s0\}^P, R{\uparrow}P)$.

Given an arbitrary set of local states $mathcalS$ with designated initial state includes $s0 \in mathcalS$, a *local-states function* $S : P \mapsto 2^{\mathcal{S}}$ maps every set of agents $P \subset \Pi$ to some $S(P) \subset \mathcal{S}$ that includes $s0$ and satisfies $P \subset P' \subset \Pi \implies S(P) \subset S(P')$.

Definition 3.4 (Multiagent Protocol). A multiagent protocol $\mathcal{F}$ over a local-states function S is a family of multiagent transition systems that has exactly one transition system

$$\mathcal{F}(P) = (C(P), c0(P), T(P))$$

for every $P \subset \Pi$, where $C(P) := S(P)^P$ and $c0(P) := \{s0\}^P$.

Definition 3.5 (Multiagent Protocol Induced by Transactions). Let S be a local-states function and R a set of transactions over S. A multiagent protocol $\mathcal{F}$ is *induced by R* if for each set of agents $P \subset \Pi$:

$$\mathcal{F}(P) = (S(P)^P, \{s0\}^P, R(P){\uparrow}P)$$

where $R(P) := \{t \in R : t \text{ is over } Q \text{ and } S(Q') \text{ for some } Q \subseteq Q' \subseteq P\}$.

3.2 Using Essential Agents to Characterise the Four Classes of Global Platforms

Using the machinery of multiagent protocols developed above, we we define the notion of essential agents and use it to characterise the four classes of global platforms.

Definition 3.6 (Essential Agents). Given a multiagent protocol $\mathcal{F}$ induced by transactions R, a set of agents $E \subseteq \Pi$ is *essential* if it is a minimal set such that for every P with $P \cap E = \emptyset$, every run of $\mathcal{F}(P)$ has only unary transitions.

Note that a protocol may have multiple essential sets.

Definition 3.7 (Global Platform Classes). Let $\mathcal{F}$ be a transactions-based multiagent protocol and E a set of essential agents in $\mathcal{F}$ with minimal cardinality. Then the *class* of $\mathcal{F}$ is defined according to the size of E, as follows:

(1) **Centralised**: $|E| = 1$
(2) **Decentralised**: $1 < |E| < \infty$
(3) **Federated**: $|E| = \infty$ and $E \subset \Pi$
(4) **Grassroots**: $E = \Pi$

Observation 1. *The classes of Definition 3.7 form a partition of all global platforms specified by transactions-based multiagent protocols.*

Namely, to classify any global platform—existing or imagined—it is enough to provide a credible multiagent atomic transactions specification of it and analyse the cardinality of the essential set of the ensuing multiagent protocol.[2]

3.3 Grassroots Platforms: An Alternative Characterisation

Here we recall the original definition of grassroots protocols [52, 56], recasted in our current framework, and relate it to the essential-agents-based characterisation.

Definition 3.8 (Projection). For a configuration $c \in S^{P'}$ and subset $P \subseteq P'$, the *projection* of c to P, denoted c/P, is the configuration in S^P defined by $(c/P)_p = c_p$ for all $p \in P$.

Definition 3.9 (Computation Notation). For configurations c, c' in a transition system, we write $c \xrightarrow{*} c'$ if there exists a finite computation (sequence of transitions) from c to c'. Specifically, there exist configurations $c = c_0, c_1, \ldots, c_n = c'$ where $c_i \to c_{i+1}$ is a transition for each $i < n$.

Definition 3.10 (Interactive Protocol). A protocol $\mathcal{F}$ is *interactive* if for every $\emptyset \subset P \subset P' \subseteq \Pi$ and every configuration $c \in C(P')$

[2] It is theoretically possible for two alternative specifications to vie for being the "right" specification of some known global platform. If the two seem credible yet have different cardinalities of the minimal sets of essential agents, probably the specification with the larger set should be used for classification. We have yet to see or produce such a case.

such that $c/P \in C(P)$, there is a computation $c \xrightarrow{*} c'$ of $\mathcal{F}(P')$ for which $c'/P \notin C(P)$.

The interactive property requires that agents in P can always potentially interact with agents in $P' \setminus P$, leaving "alien traces" in their local states that could not have been produced by P operating alone.

Definition 3.11 (Grassroots Protocol). An atomic transactions-based protocol $\mathcal{F}$ is *grassroots* if it is interactive.

We note that the original definition of grassroots protocols [52] included a notion of obliviousness, however subsequent work [56] showed that any transactions-based multiagent protocol is oblivious. Hence we bypass it here.

Theorem 3.12. *In a grassroots protocol all agents are essential.*

Proof. Assume $\mathcal{F}$ is grassroots, then by definition it is interactive. Suppose for contradiction that some proper subset $E \subsetneq \Pi$ is essential with at least two agents $p, q \notin E$.

Let $P = \{p\}$ and $P' = \{p, q\}$. Note that $P \subset P'$ and $P' \cap E = \emptyset$.

Since E is essential and $P' \cap E = \emptyset$, every run of $\mathcal{F}(P')$ has only unary transitions.

But $\mathcal{F}$ is interactive, so for $P = \{p\} \subset P' = \{p, q\}$ and any $c \in C(P')$ with $c/P \in C(P)$ there must exist a computation $c \xrightarrow{*} c'$ of $\mathcal{F}(P')$ with $c'/P \notin C(P)$.

For $c'/P \notin C(P)$, the computation must have p interacting with q (otherwise p's state would not be reachable in $\mathcal{F}(P)$). This requires a non-unary transition, contradicting that $\mathcal{F}(P')$ has only unary transitions. □

As the inverse of the theorem does not hold, we refer to the original definition of grassroots platforms as *tight* and to the new characterisation via essential agents as *loose*. The question arises, which platforms quality as grassroots under the loose definition but not under the tight one? As bewildering as it may sound, we believe these are platforms in which participants sacrifice their dignity and forgo their basic human rights. One examples is a platform in which participants forgo their freedom of speech—that runs have a suffix in which agents cannot communicate with each other. Another is a platform in which after a finite prefix a leader is chosen that cannot be deposed, and following which the leader functions like a centralised server, an intermediary for all agent-to-agent communication. Fully understanding the mathematical and practical implications of the difference between the tight and loose grassroots notions is an open question.

4 Additional Global Platforms

Beyond social networks, the informal four-class characterisation applies to diverse global platforms.

Currencies and payment systems. Fiat currencies and Central Bank Digital Currencies (CBDCs) are centralised under government control. PayPal and Venmo similarly have single controlling servers. Bitcoin [43] and Ethereum [64] exemplify decentralised systems with bootstrap nodes. Ripple [51] operates as federated with designated validators. Grassroots cryptocurrencies [54, 56] enable communities to bootstrap local economies from trust relationships, with transactions for bilateral IOUs, mutual credit clearing, and trust-based liquidity creation—all operating without external capital or credit.

"Sharing economy" platforms. Uber and Airbnb are centralised with corporate control. OpenBazaar [1] attempted decentralised peer-to-peer commerce using blockchain but ceased operations. Platform cooperatives [50] propose federated models where worker-owned nodes coordinate services. Grassroots digital cooperatives [54] would enable direct peer-to-peer resource sharing—transportation, housing, labour—with transactions managed through mutual credit and democratic governance rather than platform mediation.

Content distribution. YouTube and Netflix are centralised. IPFS [7] uses decentralised DHT with bootstrap nodes. PeerTube [24] federates video hosting across servers. BitTorrent [18] and similar protocols approach grassroots with peer-to-peer content sharing, though trackers provide some centralisation.

Messaging. WhatsApp and Telegram are centralised. Signal uses decentralised sealed sender [35] but retains central servers. Matrix [38] and XMPP [49] federate across home servers. Scuttlebutt [58] and Briar [11] are grassroots with peer-to-peer messaging.

Computation. Cloud providers (AWS, Azure) are centralised. Ethereum provides decentralised computation [64]. Grid computing [23] federates across institutions. Grassroots Logic Programs [55] provide concurrent logic programming with single-occurrence reader and writer logic variables for implementing grassroots platforms.

Governance. Existing voting platforms are typically centralised (Change.org). DAOs use decentralised blockchain voting [62]. Grassroots democratic federations [56, 57] support constitutional governance through transactions for community formation, sortition-based assembly selection, and multi-level federation—enabling democratic scaling from local to global without central coordination.

AI and machine learning. OpenAI, Google AI, and Anthropic operate centralised platforms with models hosted exclusively in provider-controlled datacenters. Despite its name, federated learning [40] is architecturally centralised—a single server coordinates all training, aggregates model updates, and controls the global model; only data storage is distributed. PyTorch Distributed [33] and similar frameworks require master nodes for coordination. Bittensor [9] achieves decentralisation through blockchain-based coordination with validators and miners. Petals [10] approaches grassroots with peer-to-peer inference where volunteers host model shards without central coordination.

5 Related Work

The characterisation of distributed systems has been a fundamental challenge in computer science since the field's inception. Our work builds upon and extends several research threads: formal models of distributed computation, atomic transactions in distributed systems, peer-to-peer architectures, and the emerging theory of grassroots computing.

5.1 Formal Models of Distributed Systems

Lynch's comprehensive treatment of distributed algorithms [36] provides foundational definitions for distributed systems, though primarily focused on consensus and synchronization rather than architectural classification. Tel's work [59] similarly emphasizes

algorithmic aspects without addressing the structural characterisation we identify. Attiya and Welch [6] present a computational model based on message passing and shared memory, but do not consider the role of named versus anonymous agents that forms the basis of our essential agents framework.

The closest work to ours in spirit is Angluin's seminal paper on local and global properties in networks of processors [4], which distinguishes between problems solvable with purely local information versus those requiring global coordination. However, Angluin focuses on computational complexity rather than architectural characterisation, and does not address the spectrum from centralized to grassroots that we formalize.

The notion of essential agents is analogous to other threshold concepts in distributed computing, such as the FLP impossibility result requiring failure detectors for consensus, or Byzantine agreement requiring n > 3f participants.

5.2 Atomic Transactions and Distributed Computing

Atomic transactions have been extensively studied in distributed database systems [8, 26, 63]. The integration of atomic transactions into distributed computing models was pioneered by Lynch and colleagues [37], though their focus was on correctness properties rather than using transactions as a basis for system classification.

More recent work has explored atomic transactions in the context of blockchain systems [29], though this literature assumes a specific architectural model (blockchain) rather than using transactions to characterize different architectures. The extension of process calculi with atomic transactions [2, 19] provides formal semantics but has not been applied to classify system architectures.

5.3 Peer-to-Peer and Decentralized Systems

The peer-to-peer systems literature provides extensive practical examples but limited formal characterisation. Lua et al. [34] survey P2P overlay networks, categorizing them by topology (structured vs. unstructured) rather than by essential agent requirements. Androutsellis-Theotokis and Spinellis [3] classify P2P systems by application domain without addressing the fundamental architectural distinctions we identify.

BitTorrent [60] and Scuttlebutt [58] appear to be grassroots, though neither has been formally proven to be so. The Dat protocol [44] and IPFS [7] occupy an intermediate position, requiring bootstrap nodes similar to our characterisation of decentralized platforms.

6 Conclusion

We introduced essential agents—minimal sets whose removal prevents all non-unary transitions—and proved their cardinality partitions global platforms into four classes: centralised (1), decentralised (finite >1), federated (infinite, not universal), and grassroots (universal). We proved that grassroots platforms as originally defined have all agents essential. This work provides the first mathematical framework for studying today's most important family of computer systems.

Acknowledgement

I thank Andy Lewis, Idit Keidar, and Nimrod Talmon for discussions that led to this work, and to Nimrod for feedback on an earlier draft. The writing and revising of the paper involved intensive discussions with Claude.

A Social Network Platforms Correctness

We state and prove the correctness properties for the social network specifications of Section 2. As the local state in each platform is a bit different, we assume that if p follows q, then within the state of p, $posts_p$ and $posts_q$ are well defined, and we refer to them. We state specific fairness requirements for each platform.

Definition A.1 (Social Network Correctness). Let $\mathcal{F}$ be a multiagent protocol induced by transactions R. We say $\mathcal{F}$ is correct if:

(1) **Follower Correctness**: For every $P \subset \Pi$, run r of $\mathcal{F}(P)$, and all $p, q \in P$, if p follows q in configuration $c \in r$, then:
- *Safety*: $posts_q$ within c_p is a prefix of $posts_q$ within c_q.
- *Liveness*: Given the platform's fairness requirements, for any post m in $posts_q$ within c_q, eventually m appears in $posts_q$ within c_p.

(2) **Agent Autonomy**: For all $P \subset \Pi$ and all runs of $\mathcal{F}(P)$:
- *Post autonomy*: For any initialised agent $p \in P$, a Post transaction is always enabled.
- *Follow autonomy*: For any initialised agent $p \in P$ and $q \in P$ with $q \neq p$, if p does not follow q, then Follow(q) is enabled (when Follow is defined).

A.1 Centralised Social Network

Fairness requirement. For any run r, configuration $c \in r$, and agents p, s where $(p, \cdot) \in c_s$, there exists a subsequent configuration $c' \in r$ reached by a Sync over $\{p, s\}$ transition.

Theorem A.2. *The centralised social network protocol (Definition 2.3) with its fairness requirement is correct.*

Proof. **Property 1 (Follower Correctness):**
- *Safety*: If $(q, posts) \in c_p$, then either (i) p has not synced with server s since following q, in which case $posts = \Lambda$, or (ii) p has synced, receiving q's posts from c_s. Since the server only appends to $(q, posts)$ in c_s during Sync with q, and p receives these during Sync with s, $posts$ in c_p is always a prefix of $posts$ in c_q.
- *Liveness*: If p follows q and q has posted m, then by the fairness requirement, Sync over $\{q, s\}$ eventually occurs, adding m to $(q, posts)$ in c_s. Subsequently, Sync over $\{p, s\}$ occurs, copying the updated posts to c_p.

Property 2 (Agent Autonomy):
- *Post autonomy*: Post(m) over $\{q\}$ requires only $(q, posts) \in c_q$, which holds after registration.
- *Follow autonomy*: Follow(q') over $\{q\}$ requires $c_q \neq \emptyset$ and $(q', \cdot) \notin c_q$, always enabled for unfollowed agents.

□

Remark: As discussed in Section 2.3, this protocol requires 6 transactions initially and 3 for subsequent posts.

A.2 Decentralised Social Network

Fairness requirement. For any run r, configuration $c \in r$, and agents p, q where $q \in peers_p$, there exists a subsequent configuration $c' \in r$ reached by a Sync over $\{p, q\}$ transition.

Theorem A.3. *The decentralised social network protocol (Definition 2.5) with its fairness requirement is correct.*

Proof. **Property 1 (Follower Correctness):** This holds vacuously as there is no Follow transaction. All agents receive all posts via blockchain replication. User interface filtering provides selective viewing.

Property 2 (Agent Autonomy):

- *Post autonomy*: Post(m) over $\{p\}$ is always enabled as it only appends to $posts_p$.
- *Follow autonomy*: Not applicable (no Follow transaction defined).

□

Remark: As discussed in Section 2.4, this protocol requires an unbounded number of transactions for post propagation through blockchain synchronization.

A.3 Federated Social Network

Fairness requirement. (i) Client-Server: For any run r, configuration $c \in r$, and agents p, s where $(p@s, \cdot) \in c_s$, there exists a subsequent configuration $c' \in r$ reached by a Sync over $\{p, s\}$ transition. (ii) Server-Server: For any run r, configuration $c \in r$, and servers s_1, s_2 where $(p@s_1, \cdot) \in c_{s_2}$ for some agent p, there exists a subsequent configuration $c' \in r$ reached by a Sync over $\{s_1, s_2\}$ transition.

Theorem A.4. *The federated social network protocol (Definition 2.6) with its fairness requirements is correct.*

Proof. **Property 1 (Follower Correctness):**

- *Safety*: If $(q@s', posts) \in c_p$, then $posts$ was obtained via Sync with p's home server, which obtained it via federation from s' (if $s' \neq s$) or directly (if $s' = s$). Server-to-server sync preserves prefix ordering, so $posts$ is a prefix of q's actual posts.
- *Liveness*: If p follows $q@s'$ and q posts m: (i) By fairness, Sync over $\{q, s'\}$ occurs, updating $(q@s', posts)$ in $c_{s'}$. (ii) If p's home server $s \neq s'$, then Sync over $\{s, s'\}$ occurs by fairness, updating $(q@s', posts)$ in c_s. (iii) Sync over $\{p, s\}$ occurs, delivering m to p.

Property 2 (Agent Autonomy):

- *Post autonomy*: Post(m) over $\{p\}$ requires $(p@s, posts) \in c_p$, established at registration.
- *Follow autonomy*: Follow$(q@s')$ over $\{p\}$ requires $c_p \neq \emptyset$ and $(q@s', \cdot) \notin c_p$, enabled for unfollowed agents.

□

Remark: As discussed in Section 2.5, this protocol requires 8 transactions initially and 4 for subsequent posts.

A.4 Grassroots Social Network

Fairness requirement. For any run r, configuration $c \in r$, and agents p, q where $(q, \cdot) \in c_p$, there exists a subsequent configuration $c' \in r$ reached by a Sync over $\{p, q\}$ transition.

Theorem A.5. *The grassroots social network protocol (Definition 2.7) with its fairness requirement is correct.*

Proof. **Property 1 (Follower Correctness):**

- *Safety*: If $(q, posts) \in c_p$, then $posts$ was obtained via direct Sync with q or transitively through other agents. The Sync transaction only updates to longer sequences, preserving prefix ordering.
- *Liveness*: If p follows q ($(q, \cdot) \in c_p$) and q posts m, then by fairness, Sync over $\{p, q\}$ eventually occurs, updating $(q, posts)$ in c_p to include m.

Property 2 (Agent Autonomy):

- *Post autonomy*: Post(m) over $\{p\}$ requires $(p, s) \in c_p$, established by Initialise.
- *Follow autonomy*: Follow(q) over $\{p\}$ requires $p \neq q$ and $(q, \cdot) \notin c_p$, enabled for unfollowed agents.

□

Remark: As discussed in Section 2.6, this protocol requires 3 transactions initially and 2 for subsequent posts.

References

[1] 2014. OpenBazaar: A peer-to-peer marketplace. https://openbazaar.org.

[2] Lucia Acciai, Michele Boreale, and Silvano Dal Zilio. 2007. A concurrent calculus with atomic transactions. In *European Symposium on Programming*. Springer, 48–63.

[3] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A survey of peer-to-peer content distribution technologies. *Comput. Surveys* 36, 4 (2004), 335–371.

[4] Dana Angluin. 1980. Local and global properties in networks of processors. *Proceedings of the 12th Annual ACM Symposium on Theory of Computing* (1980), 82–93.

[5] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In *IEEE Symposium on Security and Privacy*. 375–392.

[6] Hagit Attiya and Jennifer Welch. 2004. *Distributed computing: fundamentals, simulations, and advanced topics*. Vol. 19. John Wiley & Sons.

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

[8] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. 1987. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

[9] Bittensor Foundation. 2024. Bittensor: A Decentralized Machine Learning Protocol. https://bittensor.com.

[10] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2023. Petals: Collaborative Inference and Fine-tuning of Large Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. Association for Computational Linguistics, 558–568.

[11] Briar Project. 2018. Briar: Secure messaging, anywhere. https://briarproject.org.

[12] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. 2009. PeerSoN: P2P social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*. 46–52.

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

[14] Vitalik Buterin. 2018. Governance, Part 2: Plutocracy Is Still Bad. Available at https://vitalik.eth.limo/general/2018/03/28/plutocracy.html.

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

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

[17] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. 2007. 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*. 14–25.

[18] Bram Cohen. 2003. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Vol. 6. Berkeley, CA, USA, 68–72.

[19] Edsko de Vries, Vasileios Koutavas, and Matthew Hennessy. 2010. Communicating transactions. In *International Conference on Concurrency Theory*. Springer, 569–583.

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

[21] Discord Inc. 2015. Discord - Chat for Communities and Friends. https://discord.com Voice, video, and text communication platform.

[22] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. 2018. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 949–966.

[23] Ian Foster, Carl Kesselman, and Steven Tuecke. 2001. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 15, 3 (2001), 200–222.

[24] Framasoft. 2018. PeerTube: A decentralized video hosting network. https://joinpeertube.org.

[25] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 281–310.

[26] Jim Gray and Andreas Reuter. 1993. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann.

[27] Dongning Guo and Ling Ren. 2022. Bitcoin's Latency–Security Analysis Made Simple. In *ACM Conference on Advances in Financial Technologies*. 100–113.

[28] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *24th USENIX Security Symposium*. 129–144.

[29] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*. 245–254.

[30] Idit Keidar, Andrew Lewis-Pye, and Ehud Shapiro. 2025. Constitutional Consensus. *arXiv preprint arXiv:2505.19216* (2025).

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

[32] Andrew Lewis-Pye, Oded Naor, and Ehud Shapiro. 2023. Grassroots Flash: A Payment System for Grassroots Cryptocurrencies. *arXiv preprint arXiv:2309.13191* (2023).

[33] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training.

[34] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. 2005. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials* 7, 2 (2005), 72–93.

[35] Joshua Lund. 2018. Technology Preview: Sealed Sender for Signal. Signal Blog. https://signal.org/blog/sealed-sender/

[36] Nancy A Lynch. 1996. *Distributed algorithms*. Elsevier.

[37] Nancy A. Lynch, Michael Merritt, William Weihl, and Alan Fekete. 1988. Atomic Transactions. *Morgan Kaufmann* (1988).

[38] Matrix.org Foundation. 2019. *Matrix Specification*. Technical Specification. The Matrix.org Foundation. https://spec.matrix.org Decentralized communication protocol specification.

[39] Petar Maymounkov and David Mazières. 2002. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems: First International Workshop, IPTPS 2002*. Springer, Cambridge, MA, USA, 53–65. doi:10.1007/3-540-45748-8_5

[40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[41] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. 2017. Sprites and state channels: Payment networks that go faster than lightning. In *International Conference on Financial Cryptography and Data Security*. Springer, 508–526.

[42] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008). https://bitcoin.org/bitcoin.pdf

[43] Satoshi Nakamoto and A Bitcoin. 2008. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* 4 (2008).

[44] Maxwell Ogden, Karissa McKelvey, and Mathias Buus. 2017. Dat - Distributed Dataset Synchronization And Versioning. https://github.com/datprotocol/docs. White paper, version 2017-01-17.

[45] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.

[46] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. *Draft version 0.5.9.2* (2016). https://lightning.network/lightning-network-paper.pdf

[47] Project Liberty. 2021. Decentralized Social Networking Protocol (DSNP) Specification. https://dsnp.org/. Version 1.0.

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

[49] Peter Saint-Andre. 2004. *RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core*. Technical Report. IETF.

[50] Trebor Scholz. 2016. *Platform Cooperativism: Challenging the Corporate Sharing Economy*. Rosa Luxemburg Stiftung, New York.

[51] David Schwartz, Noah Youngs, and Arthur Britto. 2014. The Ripple protocol consensus algorithm. *Ripple Labs White Paper* (2014).

[52] Ehud Shapiro. 2023. Grassroots Distributed Systems: Concept, Examples, Implementation and Applications (Brief Announcement), In 37th International Symposium on Distributed Computing (DISC 2023). (Extended version: arXiv:2301.04391) (Italy). *Extended version: arXiv preprint arXiv:2301.04391*.

[53] Ehud Shapiro. 2023. Grassroots Social Networking: Serverless, Permissionless Protocols for Twitter/LinkedIn/WhatsApp. In *OASIS '23* (Rome, Italy). Association for Computing Machinery. doi:10.1145/3599696.3612898

[54] Ehud Shapiro. 2024. Grassroots Currencies: Foundations for Grassroots Digital Economies. *arXiv preprint arXiv:2202.05619* (2024).

[55] Ehud Shapiro. 2025. Grassroots Logic Programs: A Secure, Multiagent, Concurrent, Logic Programming Language. *arXiv preprint arXiv:2510.15747* (2025).

[56] Ehud Shapiro. 2025. Grassroots platforms with atomic transactions: Social networks, cryptocurrencies, and democratic federations. *ICDCN2026, arXiv preprint arXiv:2502.11299* (2025).

[57] Ehud Shapiro and Nimrod Talmon. 2025. Grassroots Federation: Fair Governance of Large-Scale, Decentralized, Sovereign Digital Communities. *arXiv preprint arXiv:2505.02208* (2025).

[58] Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Tschudin. 2019. Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications. In *Proceedings of the 6th ACM conference on information-centric networking*. 1–11.

[59] Gerard Tel. 2000. *Introduction to Distributed Algorithms* (2nd ed.). Cambridge University Press.

[60] TorrentFreak. 2021. *BitTorrent Turns 20: The File-Sharing Revolution Revisited*. https://torrentfreak.com/bittorrent-turns-20-the-file-sharing-revolution-revisited-210702/ Contains Bram Cohen's original statement: "BitTorrent's customer is etree. Etree is a loose-knit community of people who distribute live concert recordings online".

[61] Jiaping Wang and Hao Wang. 2019. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 95–112.

[62] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. 2019. Decentralized autonomous organizations: Concept, model, and applications. In *IEEE Transactions on Computational Social Systems*, Vol. 6. 870–878.

[63] Gerhard Weikum and Gottfried Vossen. 2001. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann.

[64] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger.

[65] Shoshana Zuboff. 2019. *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Public Affairs, US.

[66] Shoshana Zuboff. 2022. Surveillance capitalism or democracy? The death match of institutional orders and the politics of knowledge in our information civilization. *Organization Theory* 3, 3 (2022), 26317877221129290.