# Blockchain Technology and its Applications
# A Promise Theory view - V0.11*

Jan A. Bergstra
Minstroom Research BV, Utrecht, The Netherlands†

Mark Burgess
MemnonTK Research, Oslo, Norway‡

May 23, 2018

**Abstract**

Blockchain amounts to a form of public voting on the acceptance of transactions, submitted to some kind of ledger, in such a way that an official record, based on a majority decision, is kept permanently for all voluntary participants to see.

We are assembling these notes to discuss blockchains and cryptocurrency applications from the perspective of Promise Theory. The aim is to discuss and document salient issues about blockchain technology in a semi-formal way, and with a minimum level of technical rigour (perhaps to offer some sobriety to the more unbridled claims). We place a special emphasis on the role of trust, particularly in light of claims that blockchain is a trust-free technology, run by democratic and transparent principles rather, than elite opaque institutions. We also seem to arrive at a tentative view that what might be valuable about blockchains today could turn into future liabilities.

Our notes are still quite rough, and are best read in conjunction with an overview of the state of the field, e.g. see [1].

# Contents



*Preliminary version, comments and suggestions for improvements are welcomed.

†janaldertb@gmail.com, info@minstroomresearch.nl

‡mark.burgess.oslo.mb@gmail.com

# 1 Introduction

Blockchain is a class of software technologies, used for building cryptographically-verifiable or tamper-proof 'ledgers' (serial transaction databases), usually built on top of redundant peer to peer (P2P) distributed consensus networks [1, 2]. As such, blockchain is an example of a system that works by *voluntary cooperation* between autonomous agents, which makes it a natural candidate for description in terms of Promise Theory [3]. The BBC reports some 1300 cryptocurrencies, at the time of writing [4]. Additionally, there are applications of blockchain from logistics companies to online universities.

There are multiple blockchain designs, but all of them promise properties such as immutability and redundancy of data, voluntarily cloned across member nodes of the peer networks. They are not unique in making these promises. Blockchain was proposed together with the Bitcoin cryptocurrency [5][1].

In principle, any data could be stored in a blockchain, though applications such as digital currencies, ownable assets, passport records, and logistical hand-overs, are natural candidates for open transactional records. The first national election in Sierra Leone was recently held via a blockchain system [9], there are crowdfunding platforms [10], and financing of small loans is now being offered [11]. The deeper question will remain concerning the longevity of data, and whether open access to these data is truly in the public interest.

We recommend that readers use these notes in tandem with a clear overview of blockchains offered in [1].

# 2 Overview

Many claims have been made about the blockchain and the various applications that have come to be built upon it. The clearest exposition of its motivation and intent is probably still the original paper by Nakomoto [5], but there are also plenty of reviews and discussions clarifying its implications (e.g see [12–15]). Our aim here is to offer a promise theoretic perspective, which underlies the causal factors and their implications, to look for the principles amongst the many promises that are claimed for blockchain. We therefore examine the elementary assumptions, and evaluate some of the claims.

In the modern understanding, a blockchain uses digital signing (also called hashing), along with public-private key (PPkeys) cryptography [16], to build a chain of cleartext evidence about past information transactions, whose

[1] The name derives from the well-known cryptographic method of cypher blockchaining (CBC) [6], in which the encrypted result of a previous block is used as the cipher key for the data of the subsequent block, forming a chain whose informational entropy [7, 8] increases monotonically with the number of blocks. The data structure is said to be a form of Merkle tree.

integrity can be verified in order to prevent or expose tampering. The key promise of a blockchain database is thus *immutability of data with public transparency*. This has obvious applications for the archiving of financial transactions, deeds of ownership, and all forms of digital governance, where disputes may arise or records are at risk of tampering. For news and historical records, it would also be useful to prevent future parties from rewriting history in their own image. On the other hand, irrefutable data with personal implications and consequences may prove harmful to individuals, e.g. witness protection programmes would become untenable if public identity records were ever committed to blockchains.

A blockchain database can be constructed on a single computer, but the real power of the technique lies in it being replicated many times, building immutability by engineering massive redundancy against the possibility of brute force attacks against the cryptographic integrity. This strategy makes the likelihood of tampering vanishingly small.

A replicated blockchain (what we shall call a shared blockchain consensus network) implicates a 'significant' network of communicating agents, which provide redundancy and confirmation by voting. Some notable features of a such a database are:

- A dependency chain of data with cumulative interior integrity checks, within the database.
- A network of peers who promise to keep precise copies of the entire database.
- A consensus mechanism for agreeing on the correctness of a single version of the data, during dynamic updates.
- Incentives for keeping these promises, applicable to so-called untrusted agents.

There might be any number of variations in how these promises are kept, but each application requires distributed members of the network to keep a relatively uniform set of core promises. Some of the promises rely on an assumption about the distributional entropy of peer makeup and location to provide sufficient variety to avoid corruption of the voting mechanisms (just as in any election process).

A key issue will be why a collection of agents would agree to such a system. Any distributed data structure is vulnerable to difference of intent. A consensus system is designed to have singular intent.

# 3 Promise theory

Our lens for these notes is promise theory: a theory of interactions between agents and their intentions (see [17–19]) that documents dynamical and semantic aspects of networks. Promise theory is a tool for reasoning about networked systems, because it exposes network structures at the physical and virtual levels, and their implicit trust relationships. It begins with the idea of autonomous agents that interact through the promises and impositions they make to one another. Agents are autonomous, in the sense that they govern their own behaviour; each agent is thus responsible for keeping the promises it makes, and cannot reasonably be held responsible for keeping promises others might make on its behalf. We write an autonomous promise from $\mathrm{Promiser}$ to $\mathrm{Promisee}$, with body $b$ as:

$$\mathrm{Promiser} \xrightarrow{b} \mathrm{Promisee}, \tag{1}$$

where $b$ is effectively a constraint on the promiser. We denote an imposition by

$$\mathrm{Imposer} \xrightarrow{b}\!\!\blacksquare\ \mathrm{Imposee}, \tag{2}$$

where $b$ is effectively a constraint on the imposee. Promises come in two polarities, denoted with a $\pm$ signs, as below. The $+$ sign gives assertion (offer) semantics:

$$x_1 \quad \xrightarrow{+b} \quad x_2 \ \ (\text{I offer } b) \tag{3}$$

while the $-$ sign gives projection (acceptance) semantics:

$$x_1 \quad \xrightarrow{-b} \quad x_2 \ \ (\text{I will accept } b) \tag{4}$$

where $x_i$ denote autonomous agents. A promise to give or provide a behavior $b$ is denoted by a body $+b$; a promise to accept something is denoted $-b$ (or sometimes $U(b)$, meaning use-$b$). Similarly, an imposition on an agent to give something would have body $+b$, while an imposition to accept something has a body $-b$. In general, intent is not transmitted from one agent to another unless it is both $+$ promised and accepted with a $-$. Such neutral bindings are the exchange symmetry.

Promises are intimately related to the idea of trust. In promise theory, there is no such thing as a trust free architecture, so part of our goal is to expose exactly how trust works in a system or study. Moreover, because agents are a priori autonomous (independent), no other agent can know what a particular agent knows, without the latter explicitly promising to reveal the information. Such a promise could be a lie or a deception. Finally the observer agent has to promise to accept and interpret the information 'properly'. All these concepts reflect the state of affairs in any distributed information system, and underline the amount of collaborative work required to established trusted knowledge between agents. The central aspects of promise theory we shall rely on may be found summarized in the sections of [17]:

1. The axioms and definitions of promises and impositions, chapter 3.
2. Promise assessment, section 5.2 for keeping track of causal influence
3. Conditional and assisted promises, section 6.2.
4. Agreements, section 8.4
5. The value of a promise, chapter 9.
6. Specification of timescales for the lifecycle of a promise, section 7.1.

Any distributed system has so-called horizontal and vertical dependencies, meaning that certain promises are only given conditionally, on the basis that other agents keep promises to them. Horizontal and vertical dependencies reply on exterior peer services and interior stacked dependencies respectively[2].

# 4 Definitions and notation

We follow the notation of [19] as far as possible. Promise theory is about agents and their promises.

1. $A_i$, with subscripts $i, j = 1 \ldots |A|$ running to the maximum number of accounts $|A|$, represents a human agent with an account defined by its PPkey pair, possibly acting through a wallet, or other proxy agent. Thus subscripts $i, j$ effectively represents a private key identities

   in the system. An agent may take on the role of sender $S_i$ or receiver $R_j$ in a transaction.

2. $S_i = A_i \in \{A_i\}$ is an account agent, in the role of sender, who requests a transaction to pay, by making an imposition on a hosting node:

   $$S_i \xrightarrow{+\tau^{(n)}_{ij(\mu,\phi)}} N_\ell \tag{5}$$

   where $R_j = A_j \in \{A_j\}$ is an account agent, in the role of recipient, identified by its own PPkey pair. and $\tau^{(n)}_{ij}(\mu, \phi)$ is the $n$th transaction from agent $S_i$ to agent $R_j$, promising information $\mu$. This transaction is imposed on the peer network, but the amount is promised to the recipient:

   $$S_i \xrightarrow{+\mu} R_j. \tag{6}$$

3. When used explicitly in connection with currency, $\mu$ represents an amount of money to transfer; otherwise, it is general information to be transacted.
4. Similarly, $\phi$ represents a fee offered for processing the transaction.
5. $N_\ell$ is a node running software agents for Bitcoin system, at a location $\ell$, where $\ell$ is a subscript representing a node instance location, which might host several human accounts.

   $$N_\ell \xrightarrow{+\text{SW}} *. \tag{7}$$

6. $S = N_S$ is the node from which the transaction is originated, i.e. the user-account node.

[2] Any conditional dependency is a source of potential risk of failure. In its common meaning, a vertical dependency is often assumed to be a singular critical dependency in software system, which cannot use component-level redundancy to improve systemic reliability. However, this is only a de facto convention. There is no actual hindrance in practice.

7. $R = N_R \in N_M$ is the node which ends up mining the accepted block containing the transaction.

   A node cannot promise the role of a sender or a receiver without having at least one PPkey pair:

$$S_i = A_i \quad \xrightarrow{+S|\text{PPkey}_i} \quad * \tag{8}$$

$$R_j = A_j \quad \xrightarrow{+R|\text{PPkey}_j} \quad * \tag{9}$$

   By throwing away all of its private keys an agent ceases to be a participant.

8. $N_M \equiv M_\ell = N_\ell$ is an alias for the subset of nodes which promises mining the blocks $\beta_a$ to include the transaction

9. $N_W$ is the node on which a user's wallet exists.

10. $B_\sigma$ is block with sequence number $\sigma = 1, 2, \ldots$, accepted into the blockchain, into which a number of transactions is incorporated:

$$B_\sigma \xrightarrow{\{\tau, \tau', \ldots\}} * \tag{10}$$

    Block promises are in scope of all agents.

11. $C_\ell = \{\beta_1, \beta_2(\beta_1), \ldots \beta_\sigma(\beta_{\sigma-1}(\beta_{\sigma-2}(..)))\}$ is the blockchain replica at location $N_\ell$.

12. $\mathbf{N} = (\{N_\ell\}, \{\pi_\ell\})$ is the superagent comprising the total peer consensus network. This is a superagent of nodes and the software promises that bind them to the ledger system [20].

## 4.1 Agents

In the context of promise theory, agents are the irreducible actors, which embody intent, at a particular scale. Agents may be humans, processes, businesses, etc. They are always proxies for human intent. We normally refer to them, in their roles, with respect to information transactions, which are promises to be recorded in the ledger. $S$ is a sender of what is intended (money, information, etc), and $R$ is a receiver (or intended benefactor).

**Definition 1 (Node $N_\ell$)** *A computational instance, running the blockchain software, labelled by its identity location $\ell$, that can keep the promises of the blockchain network.*

Nodes form peers in a network of actors, which are executors of the blockchain system. One copy of blockchain data is kept by each full node; some partial nodes may only observe parts of the chain without the ability to add to it. They can promise to take on roles, such as master, slave, miner, etc, that equate to the keeping of certain sets of promises. The users of the Bitcoin system are clients.

**Definition 2 (Client user)** *A human agent, who imposes a transaction into the blockchain, or who is the recipient of a transaction. Clients thus have the roles of sender $S$ or receiver $R$ of transactions.*

A client promises an identity via a public-private key pair. A single human agent may have multiple key pairs[3].

We use Greek alphabetic symbols for data within the blockchain structure, and for naming promises ($\pi$). The purpose of the blockchain is to record transactions for clients.

**Definition 3 (Transaction $\tau$)** *The imposition $I_\tau$ of a request, by a sender client $S$ to a blockchain node agent $N_\ell$, to commit information $\tau(S, R, \ldots)$ to the blockchain.*

Transaction requests are *impositions* of promise proposals into the blockchain system, in promise theory language. If accepted, those imposed requests become promises $\pi_\tau$ by blockchain nodes to record the transaction proposal at some later time within a data block. Data blocks, thus form the elementary units of a blockchain, are also agents.

**Definition 4 (Blocks $\beta_a$)** *Data blocks are agents that promise to aggregate, store, and validate transactions. Data blocks also promise their own integrity or tamper-proofing, using cryptographic hashing in a number of ways.*

*Within $\beta_a$, transaction encodings promise to use hashes of earlier transactions to form a Merkle tree, providing cumulative verification of integrity. This is a promise of interior integrity. Further, each block $\beta_a$ promises to be a candidate successor of an earlier established block $\beta_{a-1}$ ($a > 0$), thus forming a chain of blocks. Finally, the block promises that all transactions, linkage, and other padding values all are verifiably unaltered. This constitutes a promise of exterior validation.*

[3] Indeed this is encouraged as part of the transactional security of Bitcoin [5].

This final tamper-proofing comes in a number of varieties. Poof of work, and proof of stake are two methods currently discussed. The original idea was that whatever scheme is used, it should levy a huge cost on a potential tamperer. This philsophy is changing however, as the costs of processing such penalties (e.g. in terms of energy expenditure) begins to dominate the discussion [1].

Blocks do not and cannot promise their provenance, i.e. that they originate from a particular location, but through their design they can promise to require a unique investment in truth, using these proof of work or stake schemes.

**Definition 5 (Blockchain $C_\ell$)** *A collection of data blocks, or structures, which are linearly ordered and successively dependent by cryptographic validation, thus forming a dependency chain.*

Blockchains have verification properties that make them difficult to alter without authorized cryptographic credentials. They therefore belong to a set of encryption verification methods. Blockchains are useful as shared data structures, and this is the application that prompted their origin in [5].

**Definition 6 (Shared blockchain consensus network $\mathbf{N}$)** *A collection of nodes $N_\ell$, which form a peer network such that each agent keeps an exact replica of candidate blockchains, until a single chain is chosen by consensus, based on a selection criterion.*

Each specific instance of blockchain technology may provide its own rules about how agents add and delete blocks. To form of shared blockchain consensus network (or what is colloquially referred to simple as a 'blockchain technology'), all agents must do so in such a way that all the agents eventually promise to agree about a single linearized collection of data blocks. This is what we call the distributed ledger.

**Definition 7 (Ledger)** *A totally ordered collection of records, or serial transaction database.*

The term ledger is used in economics for bookkeeping of payment transactions, and has been adopted from the original application of blockchain to construct a distributed payment ledger.

**Definition 8 (Blockchain Distributed Ledger)** *A ledger stored and published in the medium of a blockchain consensus network.*

## 4.2 Peer to peer architecture

A key component to blockchains is the distributed voting mechanism, which is used both to select the official record, and secure it against loss. This is a form of distributed consensus network, more usually implemented by very different technologies.

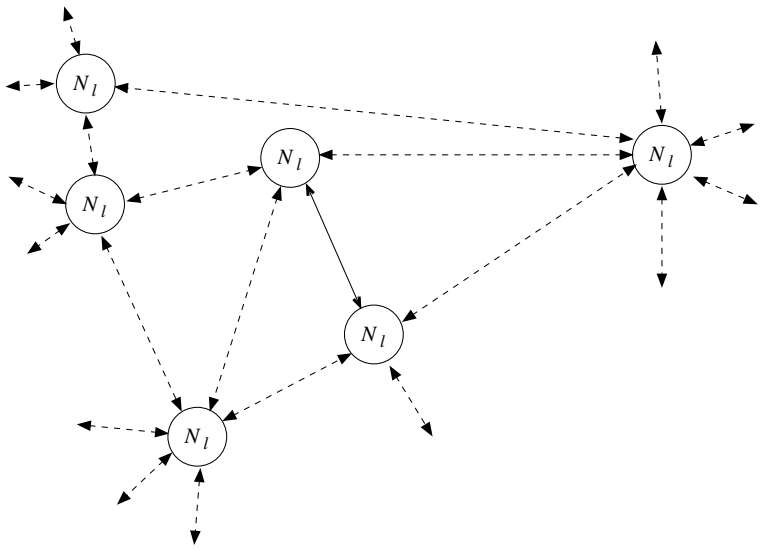


Figure 1: A peer to peer network $\mathbf{N} = \{N_\ell\}$, in which each node (or instance of blockchain software, running on a computer) communicates with its nearest neighbours. This is also the basic picture of agents in promise theory; however, in that case, directed links would refer to specific promises rather than a generic ability to communicate.

A *consensus network*, or *quorum* is a network of nodes in which every node promises to hold the same information at all observable times (see figure 1), and all agents agree on what 'the same information' means. In the case of blockchain, the nodes agree about the structure and content of a single data structure called the blockchain, which they are (a priori) free to read and append.

In an open blockchain, nodes promise to forward or broadcast all blocks $\beta_a$ and transactions $\tau$ to all agents of their peer network:

$$\pi_{\text{consensus}} = \left\{ \begin{array}{c} \{N_i\} \xrightarrow{\pm\beta,\tau} \{N\} \\ \mathbf{N} \xrightarrow{\pm\beta,\tau} \mathbf{N} \end{array} \right. . \tag{11}$$

though this is too vague to deliver much insight. In practice, nodes make many promises, e.g. the software promises to listen on port 8333, to receive messages, and to service imposed transactions by remote parties, etc. Nodes must also promise sufficient memory to hold a complete copy of the datastructure $C$, else they cannot be allowed to represent it faithfully.

$$N_\ell \xrightarrow{+(\text{memory}_\ell \gg C_\ell)} N_{\ell' \neq \ell}, \quad \forall \ell \tag{12}$$

These promises are trusted requirements for the blockchain. Thus nodes, and the software they run, have the status of trusted agents[4] [21].

The blockchain datastructure is replicated on every node that keeps the full set of promises. It remains somewhat unclear whether validating all blocks from the beginning is a mandatory requirement of blockchains, or whether a chain could be picked up at any location to verify subsequent promises. It is composed of blocks $\beta_a$, which resides within a local copy of the globally agreed chain. Each local chain $C_\ell$, on the node $N_\ell$ is a connected collection of all approved blocks $\beta_a$:

$$C_\ell = (\beta_a, \pi_{\text{chain}}). \tag{13}$$

Approved blocks are not independent, but form a chain (i.e. a recursively defined pipeline, which is used for validation), by promising to base their content on the content of prior blocks. Each block $\beta_a$ promises a hash $h_a$ of the preceding blocks.

$$\pi_{\text{chain}} = \left\{ \begin{array}{ll} \beta_a \xrightarrow{+h_a(h_{a-1})} \mathbf{N}, & \\ \beta_{a+1} \xrightarrow{-h_a} \beta_a, & a > 0 \\ \beta_{a+1} \xrightarrow{+h_{a+1}(h_a)} \mathbf{N}, & \end{array} \right. \tag{14}$$

making the blockchain promise a kind of totally ordered peer network for the block data, at the level of the datastructure. In a closed blockchain, only privileged agents would be able to assess the state of the chains. In modern developments, the use of chains is being extended to the use of generalized tree structures (Directed Acyclic Graphs, or DAGs).

Blocks make structural promises so that any node can verify the validity of a block quickly. Conversely, it takes a costly effort by a highly committed node to produce this validation, making it difficult to create valid blocks that might be used to tamper with existing data. There is an increasing flora of varieties for how this works, with different optimizations.

**Example 1** *In the original Bitcoin ledger, for instance, this promise is called 'proof of work': updating the datastructure is necessarily a relatively slow operation, because achieving a consistent certainty and agreement (consensus) on the content of the data, distributed amongst so many nodes, is a slow process. In the Bitcoin case again, each mined block promises its own hashed verification value by adjusting a 'nonce' dummy field; this amounts to a self-consistency promise that is quite calculate, and thus blocks cannot be altered without recalculating this validation.*

Self-consistency promises, in turn, make it very difficult to alter any information that has already been accepted into a blockchain. So the blockchains approximate (very well) an immutable data structure that is replicated with a high level of redundancy.

[4]The common rhetoric that blockchain is a trust-free technology is overstated and quite misleading. The aim of the design, however, is to be able to detect when nodes do not keep the promises of the software (by use of cryptographic hashes), so that the trustworthiness is somewhat verifiable. In practice this requires all agents to trust a single calibrated source for the software, thus the software origin becomes the single trusted party. Trust in its integrity is built on the transparency of the software and its promises.

# 5 Distributed consensus systems

A 'shared blockchain consensus network' belongs to a class of technologies called distributed consensus systems, whose purpose is to distribute precise replicas of a database to redundant locations. In other words, the goal is to engineering a common knowledge about data across a realm composed of multiple independent agents[5]. Well known consensus schemes of this kind include View-stamped Replication [23], Paxos [24–26], Zookeeper [27], and Raft [28]; all of the foregoing make use of an authoritative master node, through which all transactions must pass. Blockchain works differently to most such scheme, however, so we review the concepts briefly here.

Arriving at a consensus is basically the same problem as calibrating data to an agreed standard. First one has to define the standard, then ensure that all agents follow it. Calibration thus implies that a single standard of behaviour is upheld with respect to processing of data too. The most obvious way to do this is to send all data to the same place, i.e. serialize data through a single service gateway with constant semantics. This simulates a simple causal determinism, and avoids unwanted effects of relativity, where different messages arrive at different times due to the finite speed at which messages can travel. It helps to select a single spatial location for this point of calibration (called a master node) , else one may have to deal with these signal delays, complicating the selection of a standard[6]. However, this is not the approach used by most blockchains.

Distributed consensus systems thus come is two main varieties: single point of definition and equilibration by averaging, making use of the two fundamental aspects of a system, dynamics and semantics:

- **Local dynamical ordering (local spacetime causality)**. Most consensus schemes involve the election of a master node from a privileged set of custodians serving as calibration service. As data change, only a single location needs to be updated, i.e. a single point of change funnels all change requests, which it then copies exactly to nodes that keep slave copies (assuming no sharding of data[7]). A master system relies on a predefined quorum on the authoritative data, by identifying authority with the identity of a master source for all subsequent transactions.

  Causality assures that data, where order is important, arrive at a single location in a unique FCFS order. This would not be possible for more than a single node, without some form of memory of previous data. So, in this approach, arrivals are treated as a first order Markov process, i.e. a memoryless transaction process.

  This approach is relatively fast, because the semantic selection is determined by the election of a master node, and the decision is cached and reused for multiple transactions. If data arrive quickly, this is an advantage, because serialization through a fixed point makes it $O(N)$ in the length $N$ of the queue.

- **Semantic ordering (criterion ranking).**

  Where one cannot rely on correct order of arrival, or even a fixed location, we have to order arrivals based on semantic attributes. This may require a computation for the selection process, applied to every transaction. All agents receive data independently, from different sources, and possibly in different orders. There is a 'race' for transactions to arrive in a particular order, and a semantic selection has to be used settle any contention about ordering, e.g. labelling the data in order, selection of the best or longest chains, etc. These require agents to remember prior data, and vote on a post-defined quorum for which version of ordering is authoritative. Blockchain takes this approach, allowing it to reach consensus by broadcasting and averaging the state of all neighbours.

  This is a *masterless* system. It has no single point of validation, and achieves consensus by *equilibration* [29, 30] based on linkage to previous states of consensus (hence a chain). There is no causal ordering, only semantic ordering, so we cannot be sure when equilibrium will be reached. However, since blocks are added sequentially, equilibrium of past blocks gradually decouples from the ordering of newer blocks, allowing a gradual increase of trust in stability of the data. This form of data chain is the exact opposite of a Markov chain: each step depends cumulatively on a growing memory of past transactions.

  Distributed averaging is a slow process, because it may be up to $O(N^2)$ in the length $N$ of the queue.

In both cases, a voting process is used to build a quorum (before or after accepting transactions) understanding of which version of data is the authoritative one. There is a similar situation in thermodynamics, To unpick the

[5] The distributed consensus here refers to the promises about stored data. It does not necessarily apply to other promises, such as software promises, versions, or other resources. Note also that a blockchain is not a 'journal' which can be rolled back and replayed as in other databases [22].

[6] Getting the same data in the end is relatively easy. Getting the order of a sequence with distributed (non-calibrated) origin is much harder, so ordering invariance is the reason for the linearization criteria above. Conflict free replicated data types - merge all time lines (like git/version control) requires a notion of orthogonality of changes. Consensus means to engineer only one correct outcome.

[7] A sharding of a service is a partitioning into non-overlapping sub-services, each of which covers its subset independently, for parallel scalability.

entropy of mixing or equilibrium in a large reservoir of transactions with different labels[8], and order them into a stream, 'Maxwell's Daemon' examines every single packet by brute force (a master server) and separates them by their state or temperature. The work cost is very high, rescuing the second law of dynamics from potential brute force attack.

**Example 2** *The integrity of the ordering is protected by a cost of entry for agents broadcasting blocks, which is paid for in thermodynamic work. Bitcoin's daemon can quickly separate valid from invalid states, once they are in the stream, because blocks cost a lot to prepare (about 10 minutes of CPU time). It is thus not 'possible' (likely in practice) to spoof fake information into the broadcast stream, without a large effort. This is not a foolproof discriminator of trusted membership, but it is considered effective in practice for now.*

Distributed data replica systems are typically discussed according to two semantic standards: so-called ACID (Atomicity, Consistency, Isolation, and Durability) or BASE (Basically Available [31], Soft state, Eventual consistency) characteristics. The former implies that data are immediately consistent; the latter implies that data may become consistent after some equilibration time has elapsed, which assumes that intentionally equivalent data are held immutable once submitted. All systems have eventual consistency in practice. The primary issue is whether or not observers are allowed to see the states of intermediate consistency or not. This is controlled by the software service's promises to its clients.

Since equilibration time is crucial to the promise of observational invariance (not just covariance), data propagation speed (or latency) is a key factor, as is the definition of the event horizon for data. Transport congestion and observational security both play key roles in this.

## 5.1 Key promises for consensus and blockchain

Ledgers may make a number of promises about their particular characteristics. These are not always clearly stated. Reference [32] is a helpful paper in that it makes very clear statements about its claims. Typical issues include:

**Open or Closed** The policy for the ledger. If any agent can join the system, and is possibly rewarded for participation, the system may be called 'open'. If special credentials are needed to participate, then the system may be called 'closed'. Closed systems are more natural in cases where personal information or privacy concerns are paramount.

**Atomic transactions** All databases promise some kind of atomicity. A transaction is an atomic computation, (not necessarily just an atomic nugget of information). A transactional system should make clear promises about how the execution of contracts, or promise-keeping events, will be encapsulated, and what will be observable by whom.

**Transparency** Various levels of transparency may be promised. The level of transparency of an information system refers to the extent to which source code, transactions, and states of computation can be inspected by all agents (or just some).

In any decentralized system, we have to assure the security not only of intent but also assessment. Promises (operations) and assessments (monitoring) can both be distorted by incentives, faults, and errors.

**Integrity and Non-Repudiation** A promise of integrity refers to the idea that transactions will be executed exactly once, accurately, with a total ordering defined across the cluster. Non-repupdiation is a promise claiming tamper-proofing, or at least detectability of tampering.

**Encapsulation (smart contracts)** In the case of transactional computations, different architectures may promise difference isolation mechanisms. Transactions should be clearly separated, without unintended leakage between separate intentions.

**Auditability and regulation friendliness** Another promise of transparency refers to auditability. No coded software can be trusted completely to encapsulate something as complicated as systems, which today are governed by complex legal frameworks. Complying with complex societal intentions and guidelines is non-trivial, so there needs to be room for assessment.

Blockchains have been called 'trustless', but this does not mean that they can replace trusted human institutions like the law.

[8] The term entropy is used in this document in the meaning of Shannon entropy, i.e. high entropy means a state of highly mixed variations. The term is often used colloquially in security to mean the amount of randomness in a key, which is consistent. Here desirable cryptographic entropy arises from input quality, while unwanted entropy in distributed consistency lies in promised data for output.

**Proof of X** , e.g. proof of work, proof of stake, etc. Each blockchain promises to validate participants somehow.

Any ledger is a partial ordering of items. A chain is the simplest such structure, with maximal fragility. Tree structures, or directed acyclic graphs (DAG) generalize the properties of chains, and have already come into use within 'blockchain' systems, through sharding, etc [1]. The degree of decentralization in the operations performed on ledgers is therefore a continuum of possibilities that range from complete master-node centralization with pre-ordering, to complete peer decentralization with post hoc ordering.

In addition to these technical promises, there is any number of proposed benefits claimed. We shall There is a tendency towards overpromising what a mere data structure can achieve in software circles, namely that software alone can replace the malfunctioning and even possibly corrupt institutions of society. We shall comment on some of these below.

## 5.2 Ledgers versus random access data: workflows and signatures

The key promise of a ledger, as opposed to a random access database, is the partial ordering of items within it. There is a direction for time, i.e. a distinction between earlier and later transaction records. This might be coincidental or by causal dependency. Data, ordered by conditional dependency, represent workflows and procedures. They are associated with functional computations beyond the scale of whatever processes are incurred to prosecute transactions. Chains might converge to a final result, or diverge into an ever increasing space of outcomes.

The causal nature of dependent transactions have both dynamical and semantic significance. Earlier transactions form a kind of basis for the later ones to exist. Blockchains use the dynamical coupling to trace origin, as a kind of non-repudiative security. However, functionally it is the semantics of records that are most important. The signifiance of transactions fades gradually over time, as the context that defines them is lost. This amounts to an increase of informational entropy is the semantics of the transactions. As the entropy increases, the point of remembering data approaches zero, and remembering the data becomes pure overhead. To out knowledge, blockchain technologies do not currently support the forgetting (or garbage collection) of old records without loss of the entire structure[9].

As computational structures, a chain or DAG represents a structure whose growth is the result of a computation. The structure might be randomized, like a DNA string (the result of an evolutionary computation), or it might be totally ordered like a story narrative. In the latter case, the blockchain, on reaching some kind of final state, can be equated with a signature for the process. This property seems to remain undiscussed in the blockchain literature.

## 5.3 Comparing centralized and decentralized ledgers

What promises might a blockchain keep that a centralized master-slave system (with backups and redundancy) might not keep? Both the operation and the storage of data within master and non-master consensus systems are quite different. We can use promises to compare and contrast these approaches. Figures 2 and 3 are simplified, schematic promise graphs that attempt to illustrate the difference in information architectures between the two cases.

- A centralized redundant ledger, built on a master-slave principle, accepts transactions (unconditionally) $\tau_{ij}$ submitted only to the master node $M$ by clients $C_i$, and these are broadcast serially to slave nodes.

$$C_i \xrightarrow{+\tau ij} M \tag{15}$$

$$M \xrightarrow{-\tau_* \mid \text{quorum}(M,S_*)} C_i \tag{16}$$

  The master node promise replication to its slaves:

$$M \xrightarrow{+\tau_*} S_* \tag{17}$$

$$S_* \xrightarrow{-\tau*} M \tag{18}$$

  Transactions are stored directly in the master node, and the master node's version of reality is the authoritative one.

  An election quorum, about selection of an authoritative master, routing transactions through the master, and replication of information has to exist BEFORE acceptance of transactions by the elected master, to avoid

[9] Indeed, ironically, in May 2018 the NASDAQ reported companies working on blockchains for tracking the collection of real world garbage.

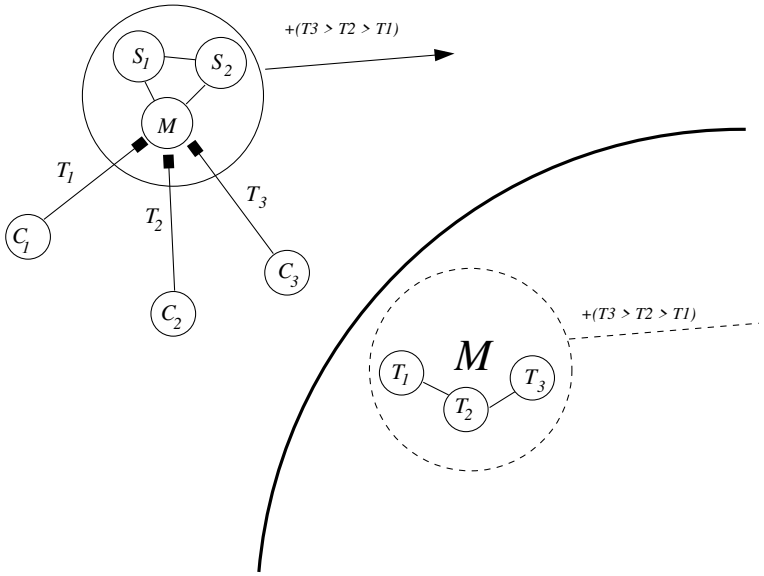


Figure 2: A centralized redundant ledger built on a master-slave principle. Transactions are submitted only to the master node, and these are broadcast serially to slave nodes. Transactions are stored directly in nodes, and the master node's version of reality is the authoritative one. A quorum has to exist BEFORE acceptance of transactions by the master, to avoid accepting an inconsistent state. Thus equilibration takes place in a typically small set of master-slave nodes. On the right, an exploded view showing the arrangement of transaction agents within each of the master nodes, ordered for each client by First Come First Served.

accepting an inconsistent state. Thus equilibration takes place in a typically small set of master-slave nodes. On the right of the figure, an exploded view showing the arrangement of transaction agents within each of the master nodes, ordered for each client by First Come First Served.

A master node is a single point of failure, but if it should fail, a new election can take place to replace it, taking over where the master left off, uninterrupted. So, with sufficient redundancy centralization is not a specific risk.

Thus the causal sequence is:

**Definition 9 (Master causation)**

*Global quorum for all transactions* $\rightarrow$ *Feed into master node*$\rightarrow$ *Replicate to redundant cluster*

- A decentralized ledger, based on blockchain, works without the tight coordination of a master-slave cluster. A chain is a conditionally dependent structure, as analyzed in section 11.3.3 of [17], with the its inherent fragility. We shall refer to that analysis here, to avoid repetition. What the analysis concludes is that the number of promises, explicitly written, to ensure precise promise keeping through a dependency chain grows as fast as $O(N^2)$ in the number of agents (e.g. the number of blocks in a blockchain). The cost of adding a block might be high ($O(N)$) or low ($O(1)$), depending on how the promises are kept, but the potential vulnerability to failing to keep the promises of the full chain grow like $N^2$ [17].

  In a blockchain peer network, transactions are submitted to any or all the nodes, and they are broadcast to all nodes. The nodes select the transactions they like the look of, possibly swayed by incentives, and blocks are mined by racing nodes. The winner extends the blockchain, and broadcasts the new block change. Other nodes voluntarily accept the blocks and promise to break ties impartially using a longest chain criterion to select an authoritative version. Transactions are thus aggregative in blocks, which or stored as a chain within nodes. Each node's blockchain promises an authorized (and eventually common knowledge) total ordering of transactions, by the agreed acceptance of a longest chain. Each block is a single point of failure for the chain, but copies of missing or broken blocks can be obtained from other peers to counter this threat.

  A quorum is formed (eventually) AFTER transactions have been accepted into blocks on nodes, and competition amongst all the nodes establishes an equilibration of state across a much larger set of authoritative nodes than for a typical master-slave system. Inside blocks, transactions are ordered for each client by First Accepted First Served. The order is not First Come First Served, because service order may be distorted by differing path length latencies and incentive fees.

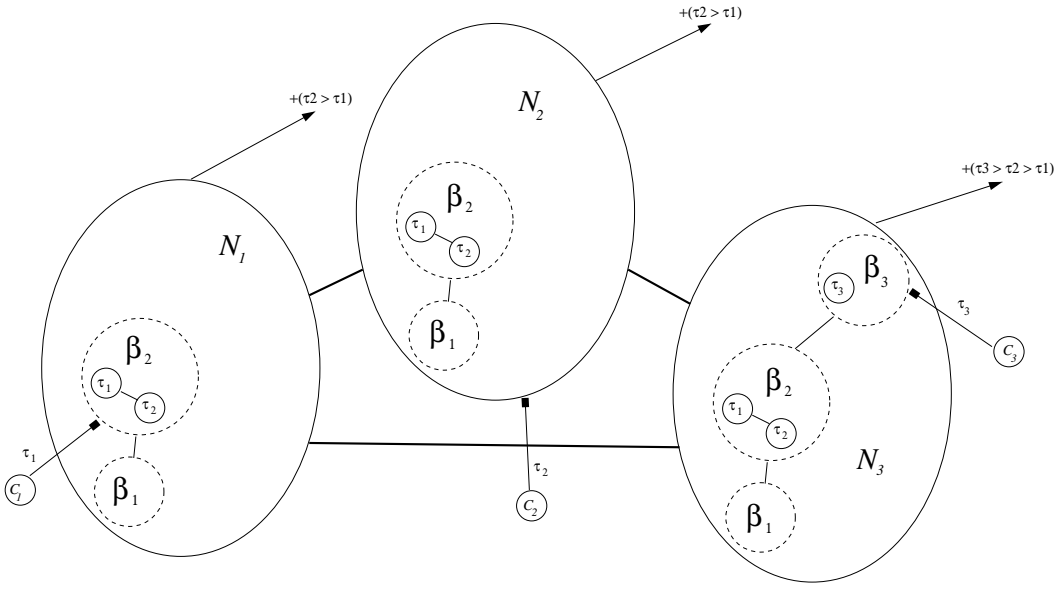


Figure 3: A decentralized redundant ledger built on a blockchain principle. Transactions are submitted to any and all nodes, and are broadcast to all nodes. Blocks are mined by racing nodes, and the winner extends the blockchain, and broadcasts the change. Other nodes promise to adopt the longest blockchain they can see as the official one. Transactions are thus stored in blocks, which or stored as a chain in nodes. Each node's blockchain promises an authorized (and eventually common knowledge) total ordering of transactions, by the agreed acceptance of a longest chain. A quorum is formed (eventually) AFTER transactions have been accepted into blocks on nodes, and competition amongst all the nodes establishes an equilibration of state across a much larger set of authoritative nodes than for a typical master-slave system. Inside blocks, transactions are ordered for each client by First Accepted First Served. Order may be distorted by differing incentive fees.

**Definition 10 (Blockchain causation)**

*Replicate transactions to peer network* $\rightarrow$ *Mine into blocks* $\rightarrow$ *Find quorum per block*

A summary of comparative points is shown in table 1.

| Promise | Master | Blockchain |
|---|---|---|
| Single points of failure | $\{M, S, ...\}$ cluster | $\{\beta....\}$ blocks |
| Redundant failover recovery | Yes | Yes |
| Vulnerability to network partitioning | Yes | Yes |
| Observability locking | possible | impossible |
| Consistency of data | Eventual or blocking | Eventual |
| Mean time to equilibration | indeterminate | indeterminate |
| Scaling of equilibration | $O(M) \times O(\tau)$ | up to $O(N^2) \times O(\tau)$ |
| Reliance on user entropy for trust | None | High |
| Favouritism of an elite | none / ad hoc policy | by CPU power |

Table 1: Comparison of some promises for master and masterless consensus.

## 5.4 Scaling of available consensus

Another way to compare the architectures is to consider the scaling of costs[10]. The cost of inserting or retrieving data can be estimated in general terms based on the scaling variables. Let $N$ be the number of peer to peer nodes, $B$ be the number of blocks, $T$ be the number of transactions, $M$ be the number of master-slave nodes.

- The locality of a master cluster makes it suitable for scaling geographically (see figure 4).

[10]Blockchains based on proof of work have a reputation for being extreme in their computational energy usage. Here we do not consider absolute cost, only cost scaling as a function of system size.

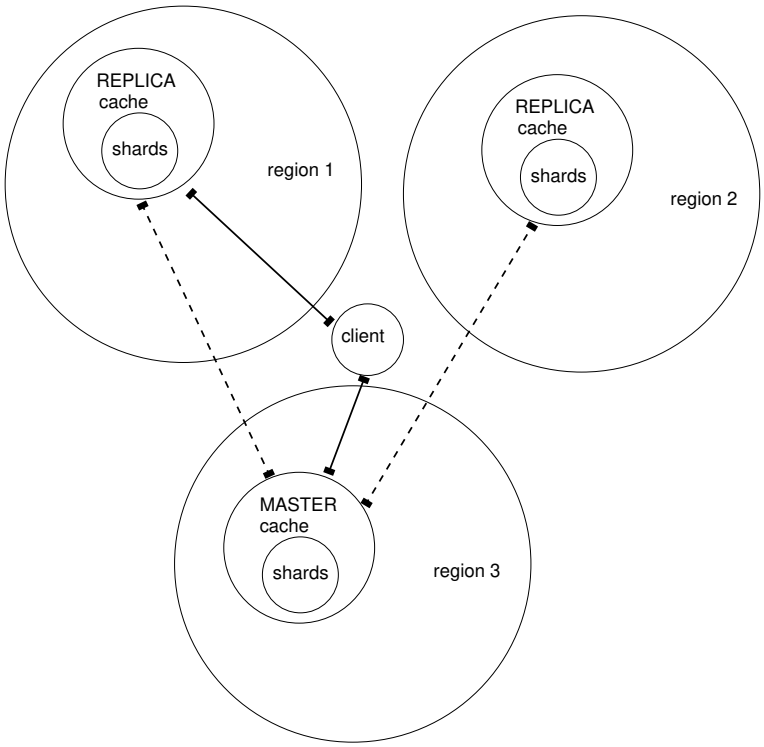


Figure 4: An agent scaling schematic, showing how replication of caches and shards may cover a space by dividing it into regions. Latency along access routes (thick lines), as well as bottlenecking to a central point, may reduce efficiency for a client writing to a central master at a single point. If the replicas are peer nodes, all of which can accept transactions (as in a blockchain) and each region has a peer, then a peer network may be faster for a user to register with, but then the time to equilibrate data between the peers may be significantly higher. It is difficult to predict which kind of architecture would be fastest, without knowing underlying network promises. Sharding of data can improve efficency of replication and read lookup. If shards are specific to the region in which they are located, this partitioning is maximally efficient. Caches may be sharded at every location, to improve indexing and lookup times.

- Blockchain's promise of control decentralization would suggest avoidance of geographical clustering.
- The same latencies exist for both, but the order in which they are applied is different. For a write once. read many times, it pays to have a local cache, so local copy of all data
- Master systems are used for metadata rather than for large amounts. Block transactions are small.
- Neither system promises a deadline, when agents will finish processing transactions.
- Master-based consensus may make varying levels of guarantee about what intermediate states can or cannot be seen in a ledger (locks on the observability), so while it may appear to behave deterministically the whole time, it cannot promise how long it will take to reach this observable state. Blockchain is not just eventual consistency, but eventual determinism. Intermediate states may be observed, but no promise is given concerning their permanence. They may be rescinded (in principle) at any later time.
- In a master system, speed can be improved by sharding the data storage, in the front-end or the back-end. So the master clusters can be parallelized by partitioning data into non-overlapping sets, split by some criterion $\chi(\tau)$, and served by a deterministically selected master $M(\chi(\tau))$. Insertion of data is quick.
  - Blockchain
    - Cost of retrieving data $O(1)/O(N)$ (denoting pick one of $N$ possible) assuming constant path length latency (pick closest node)
    - Cost of writing $O(N)$ (some fraction races in parallel) assuming constant path length latency, since each block mining adds cost to write.
    - To reach equilibrium $O(N) \times O(T) - O(N^2) \times O(T)$. However some schemes using distributed proof of stake (DPOS) have identical scaling characteristics to a master scheme, due to deterministic sharing [33].

– Central master,
  * to reach quorum $O(M^2)$,
  * Cost of writing $O(1)$ assuming constant path length latency (single write to single target).
  * Cost of retrieving data $O(1)/O(M)$ assuming constant path length latency (pick a mirror)
  * and to equilibrate data $O(M) \times O(T)$

  How big the master-slave network is compared to the peer node network is not defined, but typically master slave systems are not larger than $O(M) \simeq 10$ machines, due to the rapidly increasing cost of transaction locking.

  It seems reasonable to assume that $O(N) \gg O(M)$.

The length of the block chain does not affect the results until complete validation of the chain is required. Once performed, this should not need to be repeated.

### 5.5 Idempotence of promises and transactions

Transactions must have uniquely identifiable intent to avoid a single intention from resulting in multiple actions unwittingly, For example, an IOU for 100 dollars implies only a single payment, but if several copies of the IOU are made we have to be able to determine that the correct IOU is for 100 dollars, not 200 or 300. In the context of money, we need to prevent multiple spending of the same money. This can be avoided by labelling money with information that distinguishes independent intent (e.g. by stakeholders, promises, and the time at which an intent was 'committed' to). With such individual labels, promises are idempotent, meaning that no matter how many times we mention a promise, it is the same promise and need only be kept once.

The fundamental question with replication is: when does an arbitrary observer of data, within the permitted bounds of the system, know that the outcome of a promise has already been secured? If information travels to different places at different speeds, multiple world views are possible. This question can only be answered probabilistically, relative to the processing capabilites and speeds of network transmission.

These properties are important in blockchain, because they address the questions of whether intended transactions have been recorded, and how many times.

**Example 3** *In the case of Bitcoin, we shall see that agents simply come to believe that they can trust committed data after a few ticks of the Bitcoin clock (after a few blocks have been added). This is an* ad hoc *trust horizon of 30-40 minutes, based on experience.*

## 6 Trust and blockchain

Trust is the level of belief an agent has in predicting the outcome of a promise, before any assessment or validation is made [21]. If every action or behaviour is verified, this is an indication of zero trust. If no actions are verified, this is an indication of complete trust (which may be normalized to a value of 1).

Whether agents can be considered trustworthy or not is a matter of much interest and debate in information systems. A common attitude is that trust is a sign of vulnerability and that verification is a necessary and sufficient cure for this weakness (one might call this trust in trust)[11]. In fact, trust is something of a phantom, which can never truly be eliminated. So, to evaluate trust in blockchain, we need to be clear about what is actually promised amongst these various claims.

### 6.1 Trust in data

Blockchain technology has been called trustless (sic) or trust-free, in relation to its use of agent consensus, block validation, open access to new participants in the network, and the incorporation of data integrity validation by all participants as part of the peer protocols. A blockchain network is self-validating in this limited sense. Nevertheless, the idea that no trust is needed (which paradoxically implies that one should trust it completely) is a bold and quite misleading assertion. A more pertinent question is to ask which promises we are trusting.

Trust in services is usually approached by appealing to the integrity of a single master agent for the authorized handling of key promises according to a common standard. It is not common to include reliability in trustworthiness, only semantic correctness. This is more problematic in a distributed system, because a correct response is more sensitive to whether data arrive at a particular location. A singular agent who performs a service at an authorized location, is often referred to as a Trusted Third Party (TTP). Banks play the role of this trusted intermediary

[11] We shall not try to evaluate this belief here (see [34]), only point out the consequences of the belief.

for transactions of regular bank money (either as a direct electronic service point, or as the validator of authorized coins and notes). A single TTP calibrator allows clients to believe in the *consistency* of the agent, with respect to all clients.

If one does not believe in the integrity of a single agent, then multiple agent oversight (exterior regulation) is possible. With multiple independent agents cross checking, one can seek a quorum or consensus in the assessment of promises[12]. In blockchain, promises are cumulative and thus trust is built cumulatively. Older blocks are more trusted than newer blocks, since the leading edge of the chain may still be disputed.

There is some overlap between these two apparent positions, if we take dependencies into account. Most services rely on a standard kind of software, even if there are many service points distributed around in space. Agents are usually not completely independent; they typically run the same software, so the software itself becomes the single point of trust in many cases. Following the 'collapse of trust' in banking institutions after the 2008 financial crisis, software communities therefore proposed that exchanging trust in a banking elite for trust in a software elite was a change worth exploring.

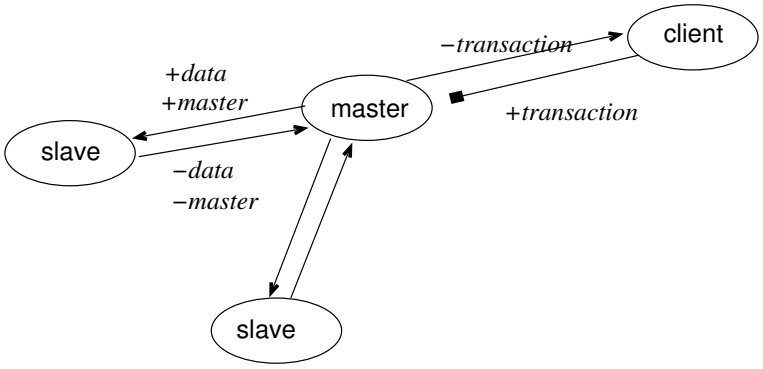


Figure 5: A master-slave consensus network. When a node is elected master, that node promises to accept transactions from clients on behalf of the others, and the others promise to subordinate themselves, accepting only authorized updates from the master. In this single-master scheme, transactions are (quickly) serialized by virtue of there being a single queue.

By focusing on a 'single point of trust' as a 'single source of truth' (SST), we form a calibrated standard whose invariance (in terms of promises kept) becomes the same issue as the invariance of the agent itself. A single point removes the issue of horizontal (spacelike) variability within a system, but it does nothing to eliminate any longitudinal (timelike) variability.

A master node may promise to be a single source of truth, but its promise depends on the sources of factual information fed to it (from distributed sources, see figure 2):

$$\text{Source}_1 \xrightarrow{+\text{fact}_1} \text{Master} \tag{19}$$

$$\text{Source}_2 \xrightarrow{+\text{fact}_2} \text{Master} \tag{20}$$

$$\text{Master} \xrightarrow{+\text{SST} \mid \text{fact}_1, \text{fact}_2, \ldots} \text{Clients}. \tag{21}$$

The keeping of all these promises is what leads to the semantic and dynamic keeping of the promise by the master node. So we can never really separate trust from reliability. Finally, there is still the possibility of a single source of truth of being a single point of failure. Data corruption can still lead to semantic unreliability of the most calibrated of sources. Redundant systems are only a security against a breach of this trust if clients actually check redundant sources (like good scientists or journalists); it is not enough for there to be multiple copies if only one is used.

The calibration afforded by channelling facts through a single point is not possible in a distributed service, where there is less reason to believe in horizontal consistency of facts between different service points: agent independence and a lack of common causality cast doubt on the consistency. Even if the agent promises are globally invariant with respect to data, they are potentially uncalibrated with respect to behaviour (they may not even be using the same software). Elaborate schemes for data consistency (distributed consensus) may be used to make limited promises about horizontal variability of data values.

[12] Assembling a body of independently verified evidence is the essence of the scientific method, and the Gaussian theory of statistical uncertainty.

## 6.2 Validation of data, and risks of immutability

One of the notable promises of blockchain is to make transactions immutable. Should the inputs to an immutable ledger be validated somehow? The growing phenomenon of 'fake news' and the shift from a trusted information society to a reputation society means that immutability could quickly become a liability. Impartiality, as journalists, news organizations, and scientists once promised, seems harder and harder to trust. Blockchain is irreversible computing.

Tamper-proofing cannot be promised, because we cannot promise that something will not happen. What can be promised is the cost incurred by an agent trying to tamper with data in each system. Blockchain tries to make this cost high, while master-based systems allow it cheaply.

How do we assess the potential damages caused by undoing information? References and dependencies on earlier promises can become orphaned if those earlier promises are rescinded. Cleaning up all subsequent dependencies of a change may be very difficult. It is usually impossible. However, this can also happen in a blockchain if a user loses their PPkey, so the risk of loss is present with different profiles in both systems.

A key feature of interest in blockchain is that horizontal variability in blockchain data is both detectable and resolvable, provided there are sufficient nodes online, and that one assumes the horizontal consistency in the software promises kept by the nodes.

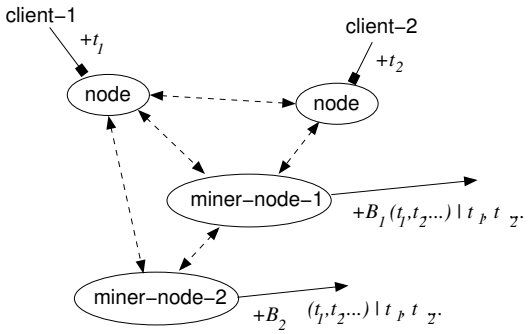


Figure 6: A masterless (blockchain) consensus network. Any node can accept transaction proposals from clients at random. All nodes promise to broadcast these proposals to the network by gossiping to their neighbours in the P2P network. Nodes that promise the role of 'miner' may accept the transactions and try to aggregate them into a block. Such blocks can be produced by any node, and are also broadcast to all nodes, who promise to accept them if and only if it satisfies criteria authorized by the blockchain software. In the end, only one block from one miner will be accepted, and there is no guarantee that a given transaction proposal will be processed. In this masterless scheme, there are multiple queues in competition, and transactions are (slowly) serialized implicitly by a final selection criterion. Non-determinism implies that users cannot automatically trust that their transactions will be honoured.

**Definition 11 (Blockchain's trusted promise)** *Blockchain's major claim is that, malicious nodes do not need to be independently trusted because any foul play on their part would be reflected in block data that would fail verification, and thence be overwritten by honest and rational agents, according to the incentives offered within the system; this relies on the availability of a quorum of sufficient size at any time in the future at which validation might be sought. This assumes that clients actually check multiple sources rather than trusting a single node.*

The data consistency is handled by a validation mechanism based on:

- Trusted calibration of node software according to behavioural rules nodes have promised to obey.
- The trusted broadcasting of transaction and block data to all peers.
- A deliberate competition between agents, with trusted cryptographic tie breaker.
- The cryptographic validation of data blocks using a trusted hash function.
- Agreement (consensus or quorum) by a majority of nodes checked, and trust in this quorum.

All these steps assume the consistent keeping of promises encoded into the blockchain software, and the belief in safety in numbers.

In practice, the cumulative nature of blockchain means that the threat of undoing blocks dwindles as they become older and the threat of dominating and overwriting the official blockchain becomes less. In other words,

there are grounds to trust the integrity of older blocks more. However, it is not beyond the realm of all speculation that a major attack could be mounted against either a single user or a partition of the network: a gradual attrition of interest in participation, or legal rulings causing nodes to drop out of the Bitcoin network, followed denial of service attack on the network to rewrite records is not beyond the scope of plausibility. Believing that what goes into a blockchain is immutable could be a serious new error of trust. The counter point is also important. The inability of blockchain to forget anything could be equally problematic, as malicious users abuse the immutability of the medium, like graffiti vandals use indelible markers to despoil public spaces [35].

## 6.3 The role of trust in blockchain

We see that it is quite inaccurate to call a blockchain a *trust-free* system, as is sometimes claimed. Its software may not formally need to be sourced from a single origin, but agents expecting to join the network must keep a set of promises that are effectively sourced from a single specification. There is thus an implicit contract involved in using a blockchain. The policing of this contract is built into the software as far as possible[13]. Most users take this software promise on trust.

It is not the avoidance of a single master node (or trusted third party) which makes a network 'trust-free', rather that no single node acts alone or unverified by others, as opposed to being regulated by a possibly (un)Trusted Third Party (TTP). In a master node consensus system, the TTP is the transaction kernel and the software. For monetary matters it is usually a bank or a broker. A monolithic system (monolithic in composition) is easier to associate with a legal entity.

In a blockchain, the Trusted Third Party is a collective formed by a majority of nodes running the trusted software. In all cases an endless chain of validating turtles (all the way down) would be required to completely avoid trust. Clearly, this is not possible. One can only improve the odds of a trusted outcome being favourable, by cumulative experience. At some point one must give up verification and simply trust.

Blockchains promise that the outcomes of transactions recorded within them will be verifiable, no matter their origin, provided all nodes use the approved software version.

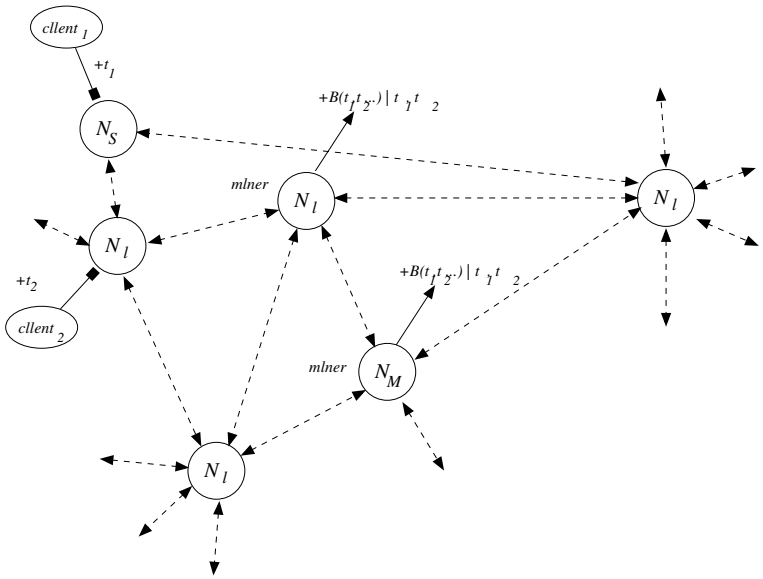


Figure 7: In a blockchain network, nodes promise to accept transactions imposed by clients as 'request for transaction', some nodes promise the role of miners. Miners promise to select and aggregate transactions into a new data block, which can be incorporated into the blockchain. Several miners may compete, in a slow race, with the same or different sets of transactions to fill a block which can be incorporated into the blockchain. All nodes promise to broadcast or relay transactions and blocks to their neighbours.

A distributed blockchain, based on peer to peer, requires the cooperation of many different parties. In cases where it's implemented as transparent open source software, it has a certain level of verifiability, but ultimately no one can see which software any node is running.

[13] Ostensibly out of the hands of untrusted human clients, though this assumption turns out to be naive.

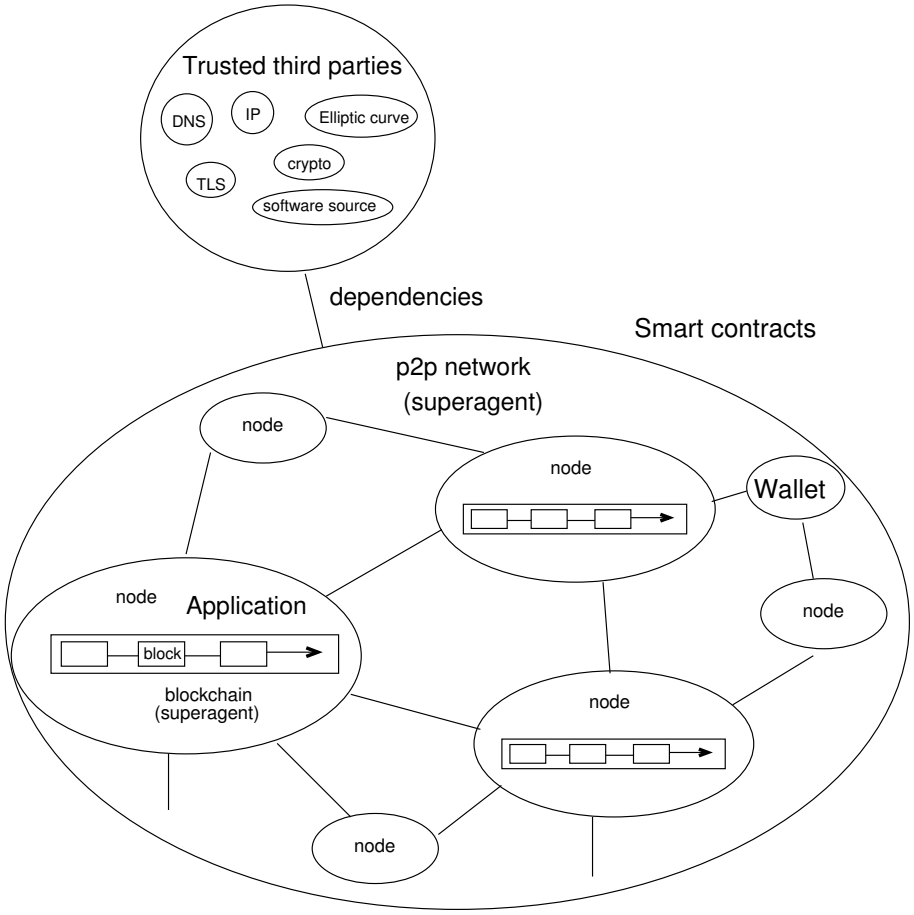


Figure 8: The Bitcoin blockchain schematic world view. Blockchain is part of the interior promises of a node. The peer consensus network is a part of the exterior promises of a node, securing the replicated integrity of the blockchain data. Applications are interior to nodes, and make use of the implicit exterior promises to secure the integrity of their data within the blockchain.

## 6.4 Use-cases for blockchain and consensus

The wide range of promises offered and employed across the space of distributed consensus systems, even those calling themselves 'blockchain' makes predicting their properties hard to generalize. How should anyone choose a system for consensual record keeping? The main selling point for blockchains is often touted as transparency and immutability without trust. Whatever one might think of the semantics, the practical consequences are worth questioning.

Permanence is not a unilaterally positive property. While technologists tend to focus on the hard problem of keeping data, they often forget that systems need to forget information as part of a natural reduction of cost (energy). This is a purely (thermo)dynamical necessity.

There is also a semantic need to forget information. Permanent defamatory information of a personal nature may cause harm to reputation: to individuals, companies, or families (during someone's lifetime and beyond). When is it ethical to place information in the public sphere? For example, untrustworthy governments have tried to rewrite the history of their countries to eliminate records of unfavourable events, such as the atrocities, weaknesses, or eras of political upheaval. Having an immutable public records 'for humankind' about public history and public news would be a security, where governments and institutions could not be trusted because of political motivations. Witness protection programs would become impossible if identity records were made on public blockchains. Secrecy is still a necessity in some cases to protect parties. To whom do we entrust the decision about this selection and its initial impartiality? There is no technological solution to this matter.

Trust has been shifting away from government institutions (who have been slow and resistant to adopt change) towards to commercial enterprises who offer quick solutions to pressing needs, but who may have fundamental conflicts of interest (e.g. social media companies). Conversely, public institutions may seem more trustworthy as arbiters of economic transactions, based in legal frameworks and the traditions of a long-standing justice system. The introduction of a vigilante system of approvals, or a flash mob, signing off on transactions (which is one way

of characterizing a Bitcoin peer network) would signal a major shift in political power.

# 7 The Bitcoin blockchain

Bitcoin was the first form of money whose transactions were executed entirely without the possibility of human intervention [36, 37]. This idealized experiment is not representative of how Bitcoin is used today. Many intermediaries and handlers are involved in operating the Bitcoin system on behalf of clients. These have taken on the roles of the very banks that Bitcoin hoped to eliminate from money transactions.

Today blockchain is considered a novel architecture for all kinds of distributed data structures, where transparency and immutability of records are valued. The claim that systems built on a blockchain do not rely on any form of centralized leadership is often touted, but this is somewhat misleading, since human leadership writes the software that determines how transactions are made, but that software may be open source and contain tamper-proofing.

The name Bitcoin (collective) represents the name of the currency (like USD or Yuan). The denomination of Bitcoin is Satoshis (analogous to cents or RMB). Bitcoins, as imagined by [5] are transaction histories rather than single agents for exchange. The schematic architecture of Bitcoin is shown in figure 8. The blocks $\beta_a$ are shown in figure 9. Each block is an upside down Merkle tree, which is used to aggregate the transactions into a flattened hash.

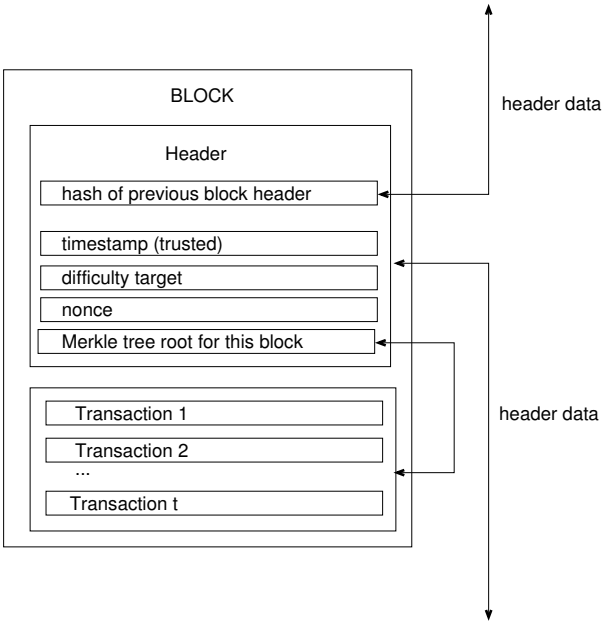


Figure 9: A single block within a blockchain may promise the following structure. It is linked in two ways: to previous and subsequent blocks by pointers and hashes to verify the integrity of the chain, and the header of the block is linked to its payload of transactions, with a hash of the hashes of the transactions, to detect any tampering.

## 7.1 Payment transactions

It is somewhat confusing to speak of Bitcoin as entities. Unlike a physically denominated currency, owners of Bitcoin never hold the currency. Nor do the coins represent fixed amounts. Moreover, nothing actually happens between users when a payment is made. No information passes from the payer to the payee. Everything happens inside the blockchain's distributed database. Miner nodes act as virtual bankers, simply making new 'pointer references' between amounts and owners of PPkeys. Thus, the total amount of Bitcoin currency remains referenced inside the blockchain, even if users lose their keys and access to it forever. Bitcoin currency is created and reassigned to different owners as transactions are processed. The Bitcoin blockchain is effectively a collection of IOUs, represented by transactions.

A transaction request is a broadcast message about the intent to transfer a measure of currency, owned by one agent $S$, to another agent $R$. An exchange currency that uses blockchain is like bank money (money of account)

insofar as it is simple a record of promised transfers of monetary amounts on a single consistent book-keeping ledger.

**Definition 12 (Bitcoin asset)** *An unspent amount of Bitcoin that is owned by some user, and is documented by a transaction in the blockchain that has not already been assigned to another user.*

Finding the money for a payment involves collecting proof of ownership of sufficient assets or funds, by assembling pointers to the outputs of previous transaction records. These assets form the inputs for a payment transaction. After a miner has recorded a transaction into a new block and that block has been accepted for sufficient time, a set of outputs (or promised outcomes) effectively reassign funds from payer to payee, and where 'change' is given, from payer back to payer (for the difference in amounts).

1. The inputs are thus a set of previous transactions, pointing to transfers to the account $S$, that 'prove' ownership of a sufficient amount of Bitcoin currency.

2. This set may add up to an amount greater than or equal to the amount to be paid. So in general, "change" must be redirected back to the sender $S$ by the software teller. Note that it is not up to the payee to return the change; that is done by the trusted agent of the Bitcoin mining software.

3. The outputs are a set of amounts that point to the intended recipient's PPkey (i.e. the payee of an amount), and any residual 'change' resulting from the difference of inputs and intended payment, directed back to the payer.

4. Transactions promise an intended transfer of satoshis, with input address as the PPkey of the sending agent to the PPkey of the receiving agent. Once accepted, the transaction remains in a transaction pool. Once a block containing the transaction is accepted into the approved blockchain, the transaction record remains forever in the blockchain unless the block is removed during a reorganization, in which case a new attempt will be made to incorporate it into a block by some other miner.

5. Each transaction request promises a sender, which is implicitly characterized by its private key $i$, not encoded in the block in any manner, and each transaction request has a public key of some node $j$ as its intended destination.

$$\tau_{ij}(\mu, \phi) \xrightarrow{+(i,j,\mu,\phi)} \mathbf{N}. \tag{22}$$

   All agents assume that the PPkeys are unique.

Things (goods and services) change hands, together with payments. In the world of physical money, money also changes hands. However, bank money and electronic currencies do not change hands: such money never leaves the bank that creates it; it is simply relabeled on a ledger.

## 7.2 Blocks

Blocks are signed aggregations of transactions, protected by a 'proof of work' validation code, which can be verified by any agent. The Bitcoin blockchain is tuned to grow by approximately one block per 10 minutes. Miners can include as many transactions as they like into a block, but there is a 1MB block size limit promised, which constrains how many transactions can be included in a block. This limit is intended to prevent huge blocks from clogging the network, but could be changed in future.

> Maximum Block Size / Average Transaction Size = Average Transactions Per Block. 1000000 Bytes / 495 Bytes = 2020

As Bitcoin network grows, the rate of transactions per 10 minute tick might grow larger and this number could overflow, unless different agents shard their blocks along orthogonal sets. Transactions that are not processed are rolled over into the next block. Priority is given to transactions that include a higher transaction fee. This is an ad hoc scheme, which is the subject of some debate.

## 7.3 Node promises

The Bitcoin consensus peer network $\mathbf{N} = (\{N_\ell\}, \{\pi_\ell\})$ is a superagent [20] of nodes running the Bitcoin software[14]. The broad promises of this collective may be summarized by:

1. The slow emergence of a transaction ledger with a unique and consistent timeline across all nodes[15].
2. The long term immutability of the ledger, assuming the continued existence of the peer network.
3. Conservation of Bitcoin assets so that duplication and Bitcoin assets is not allowed, with a maximum limit on the total amount of Bitcoin assets for all time, introducing new assets slowly in payment for services rendered.

The clearest summary of individual node behavioural promises has been given in [5]:

1. All nodes promise to broadcast transactions to all neighbouring nodes.

$$N_i \xrightarrow{+\text{replicate}} N_j, \qquad \forall N_j \in \text{peers}(N_i) \tag{23}$$

   Here we use the promise text 'replicate' to represent ledger transactions or block publications.

2. Each agent $N_\ell$ can at any time become a member of the peer network.

$$N_i \xrightarrow{-\text{replicate}} N_j, \qquad \forall N_i, N_j \in \mathbf{N}. \tag{24}$$

3. Each node $N_\ell$ is either a pure client $N_\ell \notin M_\ell$ or it is a mining client $N_\ell \in M_\ell$.
4. Each miner node promises to collect some number of transactions into a block.
5. When producing a block, a miner node promises to find a solution to the (non-unique) 'proof of work' problem, which involves fitting the hash of the mined block to a set of non-unique acceptance criteria. The solution to the hash problem may or may not be unique for each unique block, but it is verifiable by any agent, and designed to cost about 10 minutes of CPU time, thus proving the existence of a unique agent.
6. Each miner promises to broadcast finished blocks (correctly fitted to the proof of work solution) to all other nodes. Blockchain broadcasts promise to be tolerant of dropped blocks, since agents can request missed blocks from neighbours once they realize there is a link in the chain missing. Keeping this promise might take an unknown number of ticks of the 10 minute block clock to reach consistency. Based on empirical data, nodes expect that after 40 minutes equilibration can be promised.
7. Nodes promise to accept blocks from miners if and only if all transactions can be verified as valid and do not refer to spending of Bitcoin assets that have already been spent in earlier blocks.
8. Nodes accept the blocks received or mined locally by adding them to their local copy of the blockchain, and then starting a new empty block, whose 'previous hash' pointer is the accepted block.
9. If several blocks arrive at the basically same time (within a 10 minute tick of the clock), there is a conflict, because there might be two versions of the blockchain growing in different pockets of the total peer network. These pockets may order the blocks differently, so nodes needs to promise to reach an eventual consensus about the ordering. This selection is based on the longest chain available to the node:

$$B_i \xrightarrow{+\text{verified}} * \quad \text{Block promise validation} \tag{25}$$

$$\{...B_i, ...\} = BC_i \xrightarrow{+\text{len}_i} * \quad \text{Blockchain superagents promise a verifiable length} \tag{26}$$

$$N_j \xrightarrow{-\text{len}_i} * \quad \text{Nodes check the length} \tag{27}$$

$$N_j \xrightarrow{+BC_i \mid \text{len}_i > \text{len}_*} * \quad \text{Nodes promise the longest blockchain} \tag{28}$$

[14]The promise of consensus cannot be promised by the software on any single node. It is an entanglement property of the emergent state of a collective, which depends mutually on the promises of each node, including their acquired data, but none in particular. So while each node's promise can break the promise the consensus of the superagent, that consensus is not simply a function of the individual node promises. Could they be contradictory? Possibly...? Say the collective promises that it can only trust the majority of the whole collective, but locally it needs to trust each one separately to verify that. All this is based on the assumption that blocks are reported/broadcast transparently. What if there are liars? I haven't thought very hard about this yet, but it seems there may be some naivete about this validation/trust issue.

[15]Technically, Bitcoin blockchain gives a probabilistic solution to the general's problem.

10. Each node promises to serialize blocks arriving, and then promises to work on the first block their input queue. However, the nodes also promise to keep a version of other possible candidate chains, which have not been excluded, in case it should turn out to be voted the chosen one.

11. Any node may submit a transaction request into the network (as an imposition). In other words, all nodes promise to accept transactions from any source with a PPkey pair.

$$S_i \xrightarrow{+\tau_{ij}^{(n)}(\mu,\phi) \mid \text{PPkey}_i\text{PPkey}_j} N_\ell \tag{29}$$

$$N_\ell \xrightarrow{-\tau^*} * \tag{30}$$

12. Although we write an amount of Bitcoin as $\mu$, in keeping with the notation of [19], this is a mnemonic for a more complicated promise. In order to transfer an amount $\mu_{\text{intended}}$, a client agent, such as a wallet application, must find a set of previous transactions that promised amounts greater than or equal to the amount $\mu_{\text{intended}}$, and collect these block references as inputs.

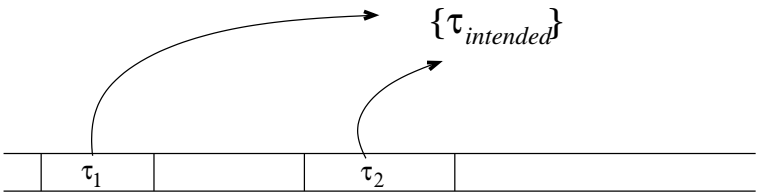


Figure 10: Previous transactions assigning a user funds are aggregated during spending to cover the payment amount. Wallet software handles this record keeping by keeping a list of pointers to transactions that assign Bitcoin to the user. Once used for payment, these same transactions cannot be spent again because a reference to their usage is now encoded later in the chain.

This is a unique set of transactions, which we denote $\{\tau_{\text{inputs}}\}$, and so can only be spent once (see figure 10). Because the blocks are labelled, they can only be spent once on a FCFS basis. If the amount $\mu(\{\tau_{\text{inputs}}\}) > \mu_{\text{intended}}$, then two transactions will be promised as outputs:

$$S \xrightarrow{+\mu_{\text{intended}}} R \tag{31}$$

$$S \xrightarrow{+(\{\tau_{\text{inputs}}\}-\mu_{\text{intended}})} S. \tag{32}$$

13. Blocks are aggregated by miners from outstanding transaction requests. However, even after having been incorporated in a block, however often, and by however many miners, any request is still present and may yet be incorporated once more an a new block by a miner who expects to profit form doing so. Multiple transactions are idempotent, and thus have no effect. REF??

14. Thus miners promise to accept transactions broadcast from any source, and to mine a block from them, provided the sum of the fees is sufficient to overcome a threshold $\Phi_{limit}$:

$$N_M \xrightarrow{-\tau} N_i, \forall i \tag{33}$$

$$N_M \xrightarrow{+\beta_\sigma(\tau^{(1)},\tau^{(2)}\ldots) \mid \phi(\tau^{(1)})+\phi(\tau^{(2)})+\ldots>\Phi_{\text{limit}}} \mathbf{N} \tag{34}$$

15. Because the nodes of the blockchain network act autonomously, and are motivated only by incentives, there is no time limit promised for how quickly transactions might be accepted by a miner for incorporation into a block. Transactions and blocks compete for acceptance in the peer network community.

    A miner $N_\ell$ creates a new block $\beta_a$ by aggregating a number of outstanding transactions into a pre-block. This candidate block is combined with a solution of the puzzle based on its local chain replica $C_\ell$, and the combination of both is signed by the $N_\ell$ in such a manner that taking out the solution and adding that to any other candidate block could only be done by a miner $M$ if $M$ used the private key of $N_\ell$. Each candidate block is thus equipped with a trusted timestamp, and a solution of the validation puzzle, created by the local chain of preceding blocks, and it is signed by the miner who made it. Then the block $\beta_a$ is broadcast by the miner for others to consider.

16. When a node receives a new block $\beta_a$, the receiver node promises to verify the consistency of the node with respect top the block chain hashes (which can be done easily by each node) and subsequently a consistency check is performed with the node's local blockchain. If the block fulfills the expected promises to fit as an extension to the local blockchain, it is included by the client.

17. The Bitcoin daemon $N_\ell$ makes the transaction with the source PPKey and the destination PPKey, then broadcasts the transaction to the network.

$$N_\ell \xrightarrow{+\tau\blacksquare} N_j, \qquad j \in \text{peers}(N_i). \tag{35}$$

18. The Bitcoin miners confirm the transaction and include it in the next(s) blockchain blocks. The miners receive the transaction fee as the reward for this.

19. When the receiver downloads a new Bitcoin blockchain block and sees incoming transaction to his/her address, the receiver knows the payment was made.

20. So all Bitcoin nodes share the same blockchain data (approx. 27.5 GB at the moment). This blockchain as eventually consistent, append-only, database of all Bitcoin transactions.

## 7.4 Remarks on these promises

High level promises indicate the goals the designers intend for user experience:

**Stabilization of each local blockchain.** The blockchain consensus network promises each client (with all other clients in scope of the promise) that a block which has been present in its local blockchain for more than 45 minutes is 'extremely unlikely' to be dropped in the future. In other words, transactions which are accepted and documented within a block which has not been eliminated by that time are 'almost definitive'. While this is a not a provable assertion, no case has yet been observed where a block older that 45 minutes has been subsequently removed. The probability of future removal decreases exponentially with time.

**Transaction processing speed up through fees.** A transaction request is an imposition, which miner nodes are free to reject, but requesters can try to sweeten the deal for miners by offering a fee to miner nodes willing to prosecute transaction[16]. The fee may be claimed by the miner, after an elapsed time, which is registered as a new transaction, visible to each client in whose local blockchain that particular block has survived.

**Double spending protection for spenders.** If a client who imposes a transaction request (i.e. attempts a payment), is disappointed by not receiving a new block incorporating that transaction, after an arbitrary time, the client may issue a new transaction with a higher fee, such that the Bitcoin software agents promise to exclude the previous transaction (this is an application of the way in which blockchain based system resolves the so-called double spending problem). Thus the software promises clients that the risk of doubly committing a transaction, double paying, or spending an amount twice is not possible.

**Double spending protection for promisees (receivers).** "No Bitcoin highwaymen". The collective network superagent running the Bitcoin software promises agents that transactions intended for $R$ cannot be diverted to another agent surreptitiously, by any other agent. The transaction $\tau(S, R, \mu, \phi)$ cannot be overridden later by 'fake' transactions, i.e. any transactions involving a transfer of the same assets from $S$ to some other account $R'$, immediately before broadcasting it to the network for processing. This is detectable because each transaction points to specific Bitcoin references, and has a timestamp, which should be able to discriminate between arrival order with a high probability due to the ten minute throttle, making the payment profile unique.

Since each node can only promise its own behaviour, this amounts to an algorithm for the detection of races involving the same assets. This promise naturally relies on the integrity of the blockchain, and the verifiability of the timestamps on transactions by the collective nodes. If a node could claim one transaction was earlier than another, while managing to win a race, this diversion of funds would be possible in principle. Any transactions can be undone, in principle, but not duplicated.

**Return currency 'change' for excess amounts.** The Bitcoin software promises to accept only previous transactions as inputs. These might not match the amount a client wants to spend, so the software promises to transform a number of owned 'input' assets, owned by the payee, into a number of 'output' assets with

[16]These fees were originally considered a formality, like an anti-spam measure, but fees have spiked on the Bitcoin markets in response to speculation [38].

different owners, such that any amount of currency included in the inputs, but not promised to another, will be returned to the initiator of the transaction as a 'change' transaction (see figure 10).

**Profits for miners.** Each miner is promised a reward for mining any block accepted by the blockchain consensus network. This consists of a fixed reward for each new block (a newly created amount of Bitcoin currency, 'thank you for playing') promised by the software, plus an amount equal to the sum of fees promised in all transaction proposals included in the block, by their respective senders.

**Success for miners.** The node software promises miner nodes that, when producing a new block, all other nodes will honour the effort expended to include that block as an extension of their blockchain, subject to the condition that there is consensus, which means: if and only if no higher ranked chains have been mined (where ranking is defined below).

**Chain competition.** Due to its slow and distributed nature, variously distributed miners might successfully mine blocks 'simultaneously' on the 10-minute clock. This may lead to multiple candidate blockchains, originating from different miners. These are broadcast around the network; however, the chains should converge to a single one, after a short number of newly mined blocks unless extraordinary coincidences persist.

Mined blocks propagate (at different speeds) through the network and are thus appended to the candidate chains at different times. Over time, one chain is likely to grow slightly faster than the others. As this happens, diffusion of the blocks will mean that more and more nodes consider this the consensus blockchain to which miners start adding new blocks until its status as the blockchain is firmly established. (The approximately ten minutes time taken to mine a block helps ensure that such forks are infrequent and, if they occur, are resolved within a few blocks length.)

Blocks that are orphaned, when alternative chains are discarded effectively have their transactions returned to the pool to be mined, so they will be picked up after a delay. It is possible, but unlikely, that contention can arise in all these phases to hinder the appearance of transactions.

**Example 4** *Imagine that the blockchain is 210000 blocks long and TWO miners both find valid blocks within a few seconds of each other and broadcast them to the network. You now have two chains, each of length 210001. Neither of these are longer than each other. Some Bitcoin nodes will see the first miner's block and some Bitcoin nodes will see the second. Eventually both chains will reach all nodes.*

*Temporarily you have two forks of the blockchain, each of length 210001 blocks long. They are identical for 210000 blocks, but the 210001st is different on the two forks.*

*Sometime later another miner finds another valid block, the 210002nd block, and that will be attached to only one of the forks, making it the longest. Of course, it is also possible that this could also happen simultaneously in two places, for different chains. Other criteria can be promised to reduce this likelihood, however. The new chain is now the longest at 210002 blocks and becomes the longest chain. It becomes the "definitive" blockchain.*

*Transactions in the alternate fork don't disappear - they get put back into the pool of unconfirmed transactions and miners will put them into a subsequent block.*

## 7.5 Maintenance promises

From the point of view of software engineering, design and development, and system maintenance Bitcoin is a classical instance of open source software, under permanent development and with an unclear obligation for user to upgrade.

**Bitcoin Foundation.** A foundation (the Bitcoin Foundation) represents the ongoing design and development manifesto of Bitcoin. This has an elected board of directors, and has the ultimate mandate to determine policy for future changes. Some controversy has already been raised around the composition of this organization [39].

**Bitcoin client distribution.** Bitcoin users must promise to keep their client up to date. This is an obligation imposed upon them by the software development group. New releases are posted on a software repository by a group of software engineers who is in control of the official software repository. Only that particular repository promises an authorized version of the client software.

Note that this structure implies the existence of an elite group whose role it is to decide the outcome of future promises made by Bitcoin. This group is determined by the Bitcoin Foundation.

**Bitcoin client authoring.** A core developer team constantly maintains and enhances client software, with inherent risks. Within the original specification, some degrees of freedom exist, and all users must trust that the core developer team promises that such degrees of freedom are made use of in a systematic and coherent manner.

**Bitcoin design and development democracy.** The core developer team promises to use a voting system, which provides the most successful miners with the most influence when choices have to be made about the further development.

At this stage of development of blockchain based informational money, including Bitcoin, there is no substitute for trust in its core developer team.

## 7.6 Remarks on Bitcoin ideology

Bitcoin arose from the aftermath of the global financial crisis in 2008, in which unruly bankers and the failures of financial regulation led to a significant loss of trust in human organizations. Amongst technological elites the argument arose that replacing these fickle human bankers with more dependable and dispassionate software could prevent such problems from arising again in the future.

Although this reasoning has a basis in history of inventing trusted third party institutions, independent of a personal stake, and with impartial regulation, a promise theoretic view quickly shows that the use of an individualized technological proxy for such organizations does not insulate anyone from such risks, essentially because any such proxy has to promise a consensus of rules which must be determined by a new human stakeholder. In spite of some flawed reasoning, Bitcoin has a number of interesting architectural features:

**No single point of failure.** All users, are free to operate as clients and as miners, and all miners have equal access to the flux of transaction requests. In other words, the software promises not to reject the participation of any node. Thus payments could not be overruled by a single organization[17].

**Robust against participant failure.** Even if 75% of the users and of the miners would become inactive the system proceeds as if nothing had happened.[18] This promise is not a promise as such, but an implicit property of quorum requirements.

**No participant needs to be trusted by other participants.** No single participant, or small group of participants to the P2P network needs to be trusted by other participants, insofar as payment is concerned, but all participants need to trust the software agents in the peer network.

Agents *do* need to trust one another in the real world however; for example, as regarding fitness for purpose of goods or services purchased, legality of goods or services purchased, and so on.

**Adequate security.** No security breach has occurred with the Bitcoin system until now since its inception in 2009[19].

**Untrusted intermediaries.** The explosion of intermediaries who promise to act as trusted proxies for buying and selling basically invalidates this security from the perspective of a client of such a proxy. Any intermediary simply becomes a trusted party, usually with far less regulation and control than any bank. This begs the question: is there any point at all to the use of such an expensive and unfamiliar technology, when its promises work mainly for the benefit of untrusted intermediaries?

**Inflation protection.** Unlike conventional monies, inflation cannot be caused by any organization in charge of the system, for the simple reason that the production of Bitcoin is both limited by and fully specified as a functional property of the Bitcoin design.[20]

**User pseudonimity.** A user uses private keys as pseudonyms. Although inspection of the entire blockchain allows inferences to be made about user identity, it may be quite difficult to find out who is in control of an account.[21]

**Universality or critical dependency.** Bitcoin transactions works if and only if the Internet is available.

[17] One can imagine other cryptocurrencies in which the software permitted certain users to be treated unequally; so it is not the fact of blockchain or cryptocurrency per se that leads to this democratization, rather it is a policy of the software.

[18] It is important, however, that a significant number of so-called full nodes remains active. A full node stores an entire (local) blockchain from its inception.

[19] Doubts have been raised about the quality of the application of elliptic curve cryptography in Bitcoin, but the system has been flawless until now.

[20] Conversely, there is no organization either which can or will see to it that in terms of its buying power Bitcoin will not degrade or collapse.

[21] The literature is unequivocal in warning prospective users against the expectation that Bitcoin transactions can be kept secret. Nevertheless, Bitcoin has proven quite useful for uses with criminal objectives such as blackmail.

**Limited performance.** Bitcoin at its current stage of development successfully processes some 20 transactions per second world wide. This comes nowhere near what is needed to play an important role in the international financial processing. In order to acquire such a role some $10^5$ transactions per second may suffice, while $10^4$ transactions per second does not.

## 7.7 The presumption of immutability of the blockchain

Once a block has been accepted it can still be removed, in principle, at any time. Indeed, there is no guarantee that blocks and their payments will persist over time. In practice, however, they do persist, which we shall explain here. This non-deterministic aspect of blockchain is troubling to some users, and is the price one pays for decentralization of transaction. See section 7.8 for further discussion.

In the long run, once a transaction has persisted for a certain time horizon (of about 30-40 minutes), single transactions cannot plausibly be removed or reversed without a new transaction to repay it, by voluntary consent of the payee (the best one can do is ask the payee nicely to repay the amount). Coercion and post hoc legal regulation of transactions is thus impossible technologically but possible sociologically.

## 7.8 Bitcoin risks

**Loss of money.** Lost digital money is unrecoverable, whereas central banks and governments can restore lost funds and lost property in principle.

**Cartelization** Trust may shift from individuals or collective action to a cartelization of the network, if a malicious user can control more than half the collective CPU power of the P2P network.

**A new elite to trust** . Bitcoin relies on incentives to those who run the network, thus it creates an elite structure to motivates the corruption of the miners, just as bankers.

**Social back-channels.** All security systems are vulnerable to covert channels. In the real world, these channels can easily dominate. Governments, press gangs, bullies, or other back-channel coercive forces can still appropriate or destroy property belonging to a blockchain user, so it offers no real protection. Trust in banks as third parties is a distraction, and the impact of the decentralization is minimal in practice.

Indelible, public proof of transaction is the strongest argument for blockchain, but this might also be limited....

**Legal risks.** There is a risk that cryptocurrencies (or specifically Bitcoin) may be either forbidden or that usage may become severely restricted, as they conflict with other directives and protections of a society. The use made of Bitcoin by criminal individuals or groups seems to justify adverse action from financial authorities. Legal risk vary from country to country.

**Internet access and stability risks.** A person who "owns" some Bitcoin cannot get access to this resource without being able to work on the Internet in a secure manner, and in such a way that private information can be kept private indeed.

**Software robustness risks.** Once the client software (including its specific brand of elliptic curve cryptography and the specific application of hash functions) turns out to be problematic, unreliable or plainly wrong, the whole Bitcoin blockchain may become irrelevant.

**Blockchain forking.** The blockchain may at any time fork in such a manner that 2 or more successors of the system coexist from that moment. Users of a specific wallet may be forced to choose a particular direction while the other direction of further development may turn out to be more fruitful.

Will money be compatible, and accepted?

**Technical innovation.** Another design of a blockchain based informational money may turn out to be superior with the effect that Bitcoin becomes obsolete. In that case it is highly likely that Bitcoin looses much of its value.

**Security/integrity problems with Bitcoin service providers.** Most Bitcoin users make use of auxiliary software such as so-called wallets. This also plays the role of a trusted third party.

It is difficult to do without such services. Auxiliary software which has been provided by others than the core developer team may be fraudulent or defective or both.[22]

[22]The public at large seems not to grasp the difference between Bitcoin being hacked and a Bitcoin service provider being hacked. The first

## 7.9 Differences between Bitcoin and banking

Some remarks concerning the Bitcoin model versus our model of the external world.

**Money and work** In the Bitcoin mythology, money is created by mining or performing work (the work of integrating and publishing blocks securely to leave an indelible trace). This suggests a Marxist view of the economy which is not representative of how modern capitalist money works. In the modern era, currency money is created by borrowing from authorized banks (through the creation of debt).

The Bitcoin ideology perpetuates a common and historical misconception that our work is what creates monetary value, and thus we all become richer by doing honest work[23] . The ethics of this can be discussed. Pragmatically, it seems to be somewhat misleading and potentially manipulative.

The creation of Bitcoin does cost real energy, however, and so one can easily argue that Bitcoin is simply exchanged for electricity acquired in the real world. This means that Bitcoin money is available to those who already have money outside the Bitcoin system. No new age of fair distribution is forthcoming[24].

**Limited amount of currency.** Cryptocurrencies are generated by software systems, governed by (currently unregulated) private companies and foundations. In the case of Bitcoin, the total amount of currency in the world is controlled and limited by design, even though the size of the activity economy may grow or shrink. Bitcoin is therefore not a sustainable currency, because it can only be shared amongst a growing number of things and people, becoming increasingly diluted. As the activity of the world increases, the total amount of money also has to increase (just as the number of messages increases). Alternatively, the efficiency, costs of production, and thus price of goods would have to decrease to compensate. However, there is no causal relationship between these matters, so the currency must be unstable unless used as a single component in a diversified portfolio of currencies.

One could argue Bitcoin's limitation is a deliberate design feature, but ordinary population growth will spoil this view. Most likely, this feature will drive Bitcoin to become an elite currency, and an object for speculation, as observed even in the initial market speculation throughout 2007.

**Loans and debt.** Because the Bitcoin money supply cannot be expanded, it is not possible to get a loan of Bitcoin, without borrowing from the amassed savings of individuals. If Bitcoin were the only currency available to an agent, loans could only be made through personal arrangements between individuals, one-to-one and unregulated, like "loan sharks", or by the introduction of regulated banks.

This lack of elasticity may cast users back into the 'dark ages' of tribal debts, honour systems, grudges, and quite possibly violence. The regulation of money lending by an authorized entity, however impractical, would undo much of the narrative about interference of state and third parties.

The protection afforded to individuals, without the need for personal trust between fickle individuals, families, and tribes, by the impersonal intermediary of a powerful state, is the value that trusted third parties such as authorized banks add to a monetary system. This point is entirely missing from Bitcoin's more libertarian raison d'être.

**Escrow during transactions.** If Bitcoin is paid, but goods are not delivered, the funds are unrecoverable without the cooperation and moral position of the payee. Thus one needs the possibility for neutral lockboxes. For some virtual assets, features can be added by 'smart contracts', for instance in the Ethereum platform.

**The right to forget.** The forgetting of data is not a bug but a feature. Without the ability to forget, information becomes garbage, burdening society with a new sanitation problem. When should data be forgotten? We may need this for legal and practical reasons.

## 7.10 Macroeconomic and capitalist considerations

The functions of money reproduced by Bitcoin are limited to transactional payments, common to consumers. In the wider economy, money acts as a lubricant not only for buying but for investing in longer term projects. Financing is an essential aspeect of money, to imbue enterprise with the freedom to act towards larger goals, without the the need for prior saving. The lack of banking functions for creating new money, means that Bitcoin cannot practically

has yet to be done, while the second problem has been around since the inception of Bitcoin. Even services with a wold-wide name had to be closed because of irregularities.

[23]The expenditure of actual energy does lead to the creation of new and potentially desirable things. Demand for those things may lead to them being sold for a price. The determination of that price has no simple explanation [19]. None of the foregoing steps results in the creation of money by which to buy goods.

[24]This is in spite of what the Bitcoin manifesto claims.

be used for expanding economic activity, or for realizing significant returns on investment, in the absence of other currencies. Without such financing, no one would have built cities, or the railways, and developmental progress, like plant building, research facilities, capital expansion could not happen.

### 7.11 Short summary for Bitcoin

Figure 11 sketches out a number of key promises in how Bitcoin works, and the hierarchy of dependency. Low level promises can invalidate higher level ones. This serves as a rough guide to its vulnerabilities.

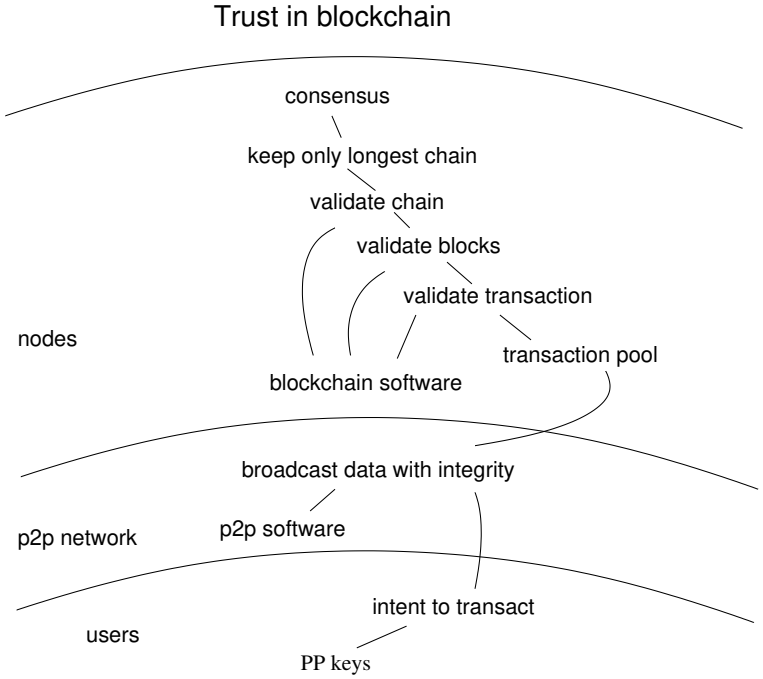


Figure 11: The hierarchy of promises on which Bitcoin is based. The higher level promises depend on the lower level ones.

The most basic promises behind Bitcoin are those to cooperate in an exchange network, by a number of humans. This is what makes it possible to form a network of peers, deploying software to each. From there, the promises of the software become the defining characteristics of a particular form of monetary cooperation. Once the nodes, owned by the human participants are running the software, they trust the subsequent interactions to that software system, and all of its dependencies. At the edge of the Bitcoin network, users can buy and sell Bitcoins for other currencies

Bitcoin does not promise anything more than transaction records, so the extraordinary fluctuations in the 'exchange value' of Bitcoin seen in the past years are entirely a result of the normal economy's contortions at the periphery of the Bitcoin network. Although it is possible to make some payments using Bitcoin, this is not widespread enough to be more than a curiosity for currency speculators. The status of Bitcoin is more like that of bonds, (even gold and diamonds have utility beyond mere investment objects or currency speculation).

## 8 Other blockchains

As mentioned in the introduction, the number of blockchain related projects grows every day. Estimates of the number of 'altcoins' or microcurrencies based on blockchains is anywhere between 1300-1500 at the time of writing. A number of other blockchains merit special mention:

**Bitcoin Cash (BCH)** A fork of Bitcoin [40], made in the fall of 2017, redesigned blocks with eight times the transactional capacity, for greater throughput, and also a different proof authorization mechanism designed for speed. The goal was to improve the poor performance of Bitcoin which had led to spiralling fees, at the cost of higher demands on client hardware, thus making participation tot the peer to peer system less generally accessible too.

The appearance of BCH may be taken as a 'proof of fork', i.e. that a blockchain can survive a fork, which is only backwards compatible on existing transactions.

**Ethereum** Ethereum blockchain extends the scope of applications built on decentralized consensus; it implements both a digital currency and a distributed execution platform called the Ethereum Virtual Machine. Blockchain mining takes 12-15 seconds on Ethereum [47].

The currency of the Ethereum platform is known as 'Ether'. While Bitcoin has a limit of 21 million coins in circulation, the creation of ethers appears to be unlimited. The virtual machine can use the blockchain mining participants as a source of computation, and the blockchain itself as a form of data storage, including permanent and temporary workspace. Using an 'immutable chain' for scratch space as well as archival data leads to interesting properties.

The notable addition to come from this generalized access to the computational platform is the addition of so-called 'smart contracts'. The impact of mining of currency transactions is easily limited in the core software, but the execution of generic code for smart contracts is another matter. It could be used or even perceived as a Denial of Service attack on nodes, without controls.

The incentive for participation in an Ethereum chain is similar to Bitcoin, but the scope of a request is not only to mine transactions, it can perform other more general computations. Ethereum uses a fuel burning analogy for its nomenclature. Ethereum miners are paid some fees proportional to the computation they perform. Specifically, each instruction in the Ethereum byte-code has a pre-specified amount of 'gas'. When a user sends a transaction to invoke a contract, he or she has to specify how much gas she is willing to provide for the execution (called gasLimit) as well as the price for each gas unit (called gasPrice). A miner who includes the transaction in his proposed block subsequently receives the transaction fee corresponding to the amount of gas the execution actually burns multiplied by a 'gas price' [41].

**EOS** EOS is another platform for smart contracts, with its own implementation of blockchain and its own currency, which links to Ethereum tokens. It claims to solve the scaling issues associated with Bitcoin and Ethereum, using a distributed proof of stake approach [33, 42]. The software promises accounts, authentication, databases, asynchronous communication, and the scheduling of applications across many of CPU cores or clusters. It claims millions of transactions per second, without user fees, and 'quick and easy deployment and maintenance of decentralized applications, in the context of a governed blockchain'. EOS also claims to allow bug fixing of applications running on its platform.

Each participant in an elected group promises to produce a block at a scheduled time. New blocks are promised by the group every 3 seconds, and all members promise not to compete in the production of blocks when it is not their turn. Each of these cooperative promises involves trust in the other members; so, in the language of Bitcoin, an EOS network is not as trustless as Bitcoin.

The EOS platform has been the basis of experiments in the Asian FinTech industry, where many operations are still processed manually [43].

**HyperLedger** An alternative smart contract platform [44]. A permissioned chain, exhibiting a hierarchy of trust: some nodes are fully trusted (called endorsers), while others only partially trusted (ordering nodes). Clients first submit their transactions to endorsers who execute the smart contract redundantly. Clients collect matching signed results and state updates from endorsers, and ordering nodes append to the blockchain. The hyperledger was used in BBVA's microloan application [11].

**Chainspace** This work seems to be a proof of concept, applying traditional consensus protocols to blockchains. This work is notable mainly for its clarity, presenting a technology that makes clear promises [32].

**MakerDao** Addressing the question of loans within the closed currency scope, there is the Maker Dao project, which is an index linked currency, whose purchasing power is linked to the USD. Loans can be made in a staunchly algorithmic manner, and then repaid with nominal interest relative to the dollar price. The Maker Dao project is trying to be economically aware regarding markets and the mechanisms of the economy.

By migrating an entire trading market onto a distributed market infrastructure (based on blockchain or not), a fully linked trade life cycle from order, settlement, all the way to payment and post trade services could be linked as smart contracts. Writers enthusiastic about these idea are generally those working in the stock market sector, where no real world things change hands. For goods traded and transported, once could imagine transport logistics being integrated too, so that confirmation and reconciliations could occur in real time, given that the transaction chain could not be broken. However, once again, this would involve trust in the platform, and voluntary cooperation of all parties involved.

**Etherisc** An example of an blockchain applied to the insurance (risk) market. Insurance is a case where asymmetric information potentially works against customer interests. Insurance companies take money from clients,

but have a vested interest in not paying out claims. The possibilities for fraud are considerable. By making insurance information and claim details 'public' for independent parties to see, the possibilities for regulation or fraud detection may be secured to a higher degree.

# 9 Smart contract platforms

Another application of distributed technologies is for mobility. The invention of society's institutional structures was an important functional development, which moved to depersonalize and reduce the costs of trust, and to accredit promises through representative decision-making, encoded as legal entities. Centralization of transactional functions, was important to promising a consistent calibration of service, but also limited their potential scalability.

With information technology, these functional aspects of centralization and authority need not be physical; they can be virtual instead. Distributed computing platforms and reach users more quickly with a mixture of online and offline services. Blockchain consensus might serve as a useful time-buffering mechanism for eventual consistency. There is thus an interesting proposal in that traditional service points might be replaced by more decentralized agents, written as embedded computer programs, to avoid inefficiencies of scaling.

Smart contracts (on top of blockchains) are computer programs that keep contractual promises as an embedded trusted intermediary, which users are supposed to trust by virtue of trusting the integrity of the blockchain. This is not a full surrogate for truly mobile code, because the agents of the blockchain and the agents being transacted are not formally connected. Placing a stashed or cached version of code close to all locations is a Promise theoretically, this is not a major innovation. It is only superficially different from a centralized service repository with a parallelized back end.

A promise viewpoint points towards using the autonomy of agents, and the keeping of promises by the agents that make them. Why would the code not accompany the very thing being transacted instead of being stored in a blockchain alongside it? The immutability argument is one answer to this question, but this brings as many problems as it solves, particularly with regard to upgrading and rescinding of promises.

## 9.1 Smart contracts

From a promise theoretic viewpoint, there is nothing special about recording monetary transactions in a blockchain, except for their general fungibility [19]. We can easily imagine using immutable structures for the keeping of more complex promises.

A simple innovation is to store executable scripts, which may be executed by the software platform that maintains the blockchain. Smart contracts are digital contracts that have been coded with promises about specific rules and penalties, bound into an agreement, like a traditional contract. Contracts are bilateral promise proposals which may or may not be signed by stakeholders to promise their agreement to the proposed terms [17]. A contract could be automated if nodes in a blockchain peer network, on receiving promise proposals as a transaction, automatically tried to keep those promises. Smart contracts could use public or private blockchains, but it is more likely that requirements of privacy would see their usage more attached to private blockchains, where credentials are needed for access. In that case, one can built namespaces and promises of user-scope into the model. Peer-based namespaces have been discussed in connection with the Internet of Things [45, 46].

With blockchain, there is eventually only a single record on a replicated distributed ledger. So, asymptotically, on convergence, it has the potential to simplify operations by dealing with one single record of truth, Automation of any kind leads to manpower savings, by elimination of account reconciliation and account maintenance work. The specific benefits of a blockchain have yet to be fully explained, but usually involve arguments about tampering and trust. The contents of smart contracts are verified and authenticated by the account agents named in the contracts. All these parties must hold a permission key to the blockchain. If agent wanted to amend or tamper with a contract, they would have to produce all the permission keys held by all the parties.

Smart contracts have many possible applications. For example, a song or film posted on the Internet could be distributed, opened, and copied, all by making a payment which is then returned to its maker. In this way, business projects could live their own independent lives on the Internet, without the need for human intervention. Applications in small scale financing has been discussed [11]. Distributed loaning is nothing new, of course: WeChat and Alipay, in China, offer loans to through social media accounts, quickly and with excellent terms. The main difference is the level of transparency behind the scenes, or perhaps interchangeably the level of trust deemed acceptable in the system. Insurance and logistics companies see possibilities widespread and high speed access to secure services. It remains somewhat controversial just how secure and private Blockchains could be, given the tension between the need for privacy and transparency. A deliberate separation of concerns is needed, which could lead to conflicts of interest.

A brief evaluation of smart contracts has a number of dimensions:

- Smart contracts may be executed in permissionless networks which arbitrary participants can join (i.e., under Byzantine conditions). This is risky and a potential security liabilty.
- Programming languages for smart contracts: Solidity, Serpent, with frameworks for test-driven app development including Truffle (with promises) and Embark.
- Miners and callers have meaningful control over the environment in which the transactions execute (e.g. which transactions to accept, transaction ordering, setting of block timestamps, manipulation of call stack, etc.) As with any distributed service, it is not trivial to make assurances on either side as long as there is a non-local behaviour, in spite of an underlying voluntary cooperation.
- The immutability of blockchain records means that there is enormous pressure to get things right first time. For monetary transactions, mistakes can be fixed by voluntarily adjusting payments based on human interactions (human trust must still play a role). However for smart contracts and other services, this is more challenging.
- There is no way to patch a buggy program or smart contract, no matter how much money it has, without reversing the blockchain (which would be an unrealistic task). Users can, naturally, refrain from using promised code, but this opens a new role for trust in the platform.

  Reasoning about the correctness of smart contracts before deployment is now critical, as is designing a safe smart contract system, by some agreed set of standards. One may explicitly design versioning and upgrade capabilities into smart contracts code, since contracts can promise to call each other ( [48]); but it remains the case that the original byte-code associated with the contract is immutable for all time, with unspecified implications, so the assessment of safety relies on ones's trust in the promises of the platform.

In Asia the financial industry still relies, to a large extent, on manual processes and paper forms, often transmitted through fax. This is obviously inefficient, high cost, and involves operational risks and delayed settlements, which increase during peak processing times [43], and may lead to insider tampering. For example, if an Asian company invests in assets that are originated by entities in a different jurisdiction, then investors in Asia, no matter where they are based, could seek legal recourse in foreign courts, assuming the blockchain were trusted. This remains untested, however. Remarkably, these questions still remain questions of trust, in spite of the trustless rhetoric. In fact, China as already signalled that it should be illegal to trust foreign blockchains.

The timing and convergence rates of blockchain are still in need of more study. Contracts that are executed, based on records that are still under potential dispute, may become invalid as blockchain forks resolve. Thus, effectively, promise outcomes could be turned into lies by future events. This uncertainty is borne by users of the platform, and must trigger consequences. It suggests that blockchain technologies need to be able to promise users when results are in fact immutable, or when convergence has occurred finally. There remains therefore some question as to how useful complicated distributed systems can be compared to central clearing agency promises.

Finally, a general theme in blockchain is that the naive use of immutability can be as much of a liability as a benefit. Indeed, as time goes on, and the length of blockchains grows, that liability grows with it. A proper design must have the ability to deprecate and garbage collect old data blocks. There is, of course, a sense that the implications of a new technology have not been fully thought through. For blockchain, this issue is more serious due to the immutability.

# 10 Conclusions

Blockchains, and other distributed ledgers, blend several ideas to fashion a model for achieving distributed consensus around a ledger of transactions that are not regulated by any central 'master' or authority.

## 10.1 Promises kept

Blockchain solves the following problems:

- It offers a form of tamper-proofing, and eternal immutability, for better or for worse.
- It promises to replace an opaque intermediary with a slightly more transparent intermediary, in practice it is no more transparent to ordinary users, who use trusted 'fourth party' applications like wallets and brokers to handle their transactions.

- Blockchain solves the technical challenge of data serialization in a masterless architecture[25].

It does not solve these remaining problems:

- It does not eliminate the need for trust: it shifts trust onto different focal points.
- It does not answer the question of why users should trust a majority vote or quorum rather than a central governing entity. Couldn't this be manipulated to lead to abuse? There is some naivete concerning the objectivity of software automation.
- Proof-of-X schemes all favour those players who have most resources to begin with: most computing power, most memory, most money(!) This tends to speak against the view of blockchains as being the great democratizers of society.

Crypto currencies solve the following problems:

- They enable transactions to be made without individual discrimination, though systematic discrimination is still possible.
- A number of technical issues about conservation of currency are handled by the 'double spending protections', but these do not protect users from being duped.
- They protect the 'dilution of the currency' against monetary inflation, which is mainly of interest to institutions rather than individuals. There is no monetary inflation with Bitcoin, but this is a purely technical promise with both positive and negative connotations for users.

They do not solve these remaining problems:

- Transaction speed is considerably slower with cryptocurrencies than for most banks, so payment by blockchain may not be suitable for small day to day transactions, without trust in the software to guarantee a transaction time.
- Transactions could still be rejected by the software, once submitted.
- The role of trust in end users becomes more like that in bartering again: cryptocurrency might promise to make an immutable payment, but it doesn't promise that you will get the goods you paid for, nor that a payer will repeat a rejected transaction after receiving the goods. Moreover, the buyer has no recourse to an institution to act as a mediator. The cold logic of the blockchain will be unsympathetic to such broken promises. We have to be aware that this is precisely the reason for human trust relationships in the first place; so, trying to eliminate them will have a cost.
- There is no mechanism for loaning money, or dealing with debt, in Bitcoin without abandoning several of the key tenets of the currency.

## 10.2 Transparency and impact

Transparency is the perhaps the most widely used claim for blockchain. In practice this means having a mechanism that can be inspected by 'anyone' at the source code level, and trusting miners to execute the authorized software (not masquerade as part of a scam). Transparency could also mean the opposite of privacy, so there is a fine balance to be maintained in separating the control channel from the data users want to keep private. This is not necessarily trivial, since it is desirable to use some of the identifying metadata as indexing for referencing data in the blockchain.

The asymmetric information argument, which is used to suggest that blockchain is needed seems spurious and ill-considered. What it really shows is that trust in our institutions has been compromised. This is certainly i crisis for civilization, but it is not an argument for total transparency.

The importance of sharing information depends on its impact. The impact might be positive or negative—indeed, it must be relative to all the parties involved in it. Promise theory's law of conditional promises makes a simple but important point: if an agent $A$ promises information $I$ to anyone:

$$A \xrightarrow{+I} * \tag{36}$$

[25] As a networking technology, it resembles the distributed cooperation of autonomous BGP zones, rather than the centralized management of OpenFlow.

and it is used by some party $X$, who then derives some followup promise $F$ based in $I$

$$X \xrightarrow{-I} A \tag{37}$$

$$X \xrightarrow{+F|I} ? \tag{38}$$

then whether this is beneficial or harmful to any party depends not only on its being shared $+I$, but on whether it is received and understood $-I$, and on how it is used $F|I$. There is bound to be a variety of social impacts as a result of any shared information.

What is important is not whether information is shared or not, but on how society prepares for the impact of this information getting into the hands of any agent. There is a scaling issue here too: one might try to prevent certain harmful outcomes by appealing to macroscale initiatives (legal restrictions, etc), but these might be appropriate on the scale of society, but completely fail to protect an individual from the harm caused. So for every benefit of open information, there might be harm. What matters in a society is how those in power regard information, and how justice is served. In no sense is it possible to work without trust. The only question is who or what do you trust the most?

## 10.3 Fitness for purpose

Blockchain is sometimes criticized as solving a problem that doesn't exist. It may prevent certain perceived risks of tampering with transactions, but in practice the functioning of society requires there to be certain legally authorized entities who provide oversight and regulation of assets and their transaction. In general, the protection of citizens requires trust in more powerful entities than the average citizen. In some cases we want transactions to be undone by the force of a higher authority; of course, this could still be recorded transparently, which is not the case today[26].

Throughout history, history itself has been rewritten many times, by individuals with different motivations. Promise theory exploits a simple premise: the belief in an absolute truth is nonsense. Facts are only promises, which are subject to interpretation. We can choose to put a positive or a negative slant of the events of the past, and we often do this with cynical motivations, not only well intentioned ones.

Using blockchains for scientific journals is an interesting idea. Scientific evidence and train of thought constitutes an important record. Our assessments about it may certainly change. thinking. Immutable records of thinking might encourage authors to be more careful in publishing results, and to reduce numbers of spurious self-plagiarizing publications. Running assessments can be run in parallel. Referee reports then become quite irrelevant, as reputation based commentary can be the measure of peer review—not the corruptible cartel-like barriers imposed by some editors.

As a record of transactions, blockchain cryptocurrencies seem to offer little above and beyond other technologically based currencies. The advantages of speed and accessability are design choices that are independent of the technology of blockchain. The idea that the immutability of transactions might stymie corruption and wrongdoing is naive at best. Power is not wielded through software, but by physical forces, which can still attack users in the real world, whatever the information says. It is a common mistake amongst software writers to assume that software is a world unto itself. States and other powers have other options to coerce and have their way.

As we pointed out in [19], some countries (like China) do not even support the permanent ownership of assets. Property reverts to the state after 70-100 years, without payment. In effect, the state owns all property, and rents it to the people on a temporary basis. The immutability of records, in this regard may need some new conventions, in addition to a new story about money.

## 10.4 Comments and remarks

We have compiled these notes to try to examine some of the assumptions that lie behind the technology of blockchains, and to assess their implications. Most discussions of blockchain are technical discussions, seeking new methods to solve challenges where traditional infrastructure has been slow to adapt and improve. Some are aimed at conceiving of new enterprises; this is egged on by the relationship to money, which inevitably attracts much interest. Then, there is a political dimension to blockchain, given its reputation as a decentralized 'trust free' form of technology. There are clear challenges to the claims made on all these fronts.

The commitment of blockchains to decentralization may turn out to be exaggerated. What might happen, for instance, if political differences drove a wedge between parties over time? Would mechanically enforced consensus

[26] In the UK, for instance, money laundering laws have led to a bizarre level of mistrust between banks and customers. The onus of 'proving' the legitimacy of funds has been pushed onto clients by banks, despite the fact that banks have the information, if they bothered to cooperate, or were willing to be transparent about their own dealings.

then act as glue keeping parties together, or as a territory to be fought over? As a benefit or a liability? The forking of some blockchains already indicates that rifts can occur, for whatever reason.

Auditing or assessment of the extent to which promises are kept (or not) by agents, from within or without will be needed, as the same loss of trust in society catches up with software. The ability to detect tampering could in principle be separated from the mechanisms of data storage and execution. Distributed consensus and its immutability could plausibly be used in an anonymous way to publicly detect tampering of data or computations, without the actual data or computations being stored in the blockchain directly. In this way, it could be a tool for society and regulators to police the ability for agents to keep sensitive promises, where one does not want to make the actual details public.

In other words, instead of adapting the blockchain (with all of its costs) to support permissions and access controls by Clark-Wilson (RBAC) model [49], or the Bell-LaPadula DAC/MAC Disrcretionary/Mandatory Access Controls [50] models, one uses blockchain only for the monitoring aspects of the system—for chain of evidence.

Some of the promises of blockchain may be unsustainable in the long run. Reversals of core principles are not unknown, especially when they take on a political dimension.

**Example 5** *The global preprint system, which post-Internet morphed into the physics paper archive (arXiv) worked effectively as a trusted third party for transparency of research until it was undermined recently by its own popularity. The cultural shift away from refereed journals to an open platform for sharing, combined with publication pressures on researchers, has led to a huge increase in posting of papers in all categories. The result has been a restriction on participation for new authors, who must seek a sponsor in order to be able to post work, rather like the exclusive gentlemans' clubs of ages passed (not not passed).*

**Example 6** *A counter example to the blockchain idea of immutability is Wikipedia, in which there is open access and quality controls. Data usually reach some kind of equilibrium, and undergo 'continuous improvement'. There have been episodes of battles for control of certain pages, but on the whole community policing has kept the content assessed as valid and valuable. If a blockchain had been used to store Wikipedia, topics would be 'first come first served', i.e. a race to take control, for no further edits would be possible after first the version had been committed.*

One cannot avoid the question of whether the decentralization of responsibility, an apparent libertarian dream, is an entirely misconceived notion, given its inevitable side effects. In essence, using a technology to try to eliminate trust is no less than handing over trust from one elite to another (e.g. from regulated bankers to unregulated technologists).

Trust is an essential glue in cooperation that cannot be eliminated without also eliminating society itself. The potential harm of a well-intentioned technology may well lie in the naivete that spawns it. In this case, what does giving up trust say about someone? It seems clear that, in the wrong hands, blockchain could quickly become harmful. The immutability of blockchains will prove to be a liability, on a number of levels. The first is in terms of resources consumed. A mechanism for forgetting transactions needs to be built into any system, if it is to scale and be assured of a continued existence, otherwise the cost of data and processing will grow more than linearly over time, until it collapses under its own weight. On a more sinister note, the inability to forget and to forgive the past is the basis for blackmail, extortion, and any number of destabilizing misbehaviours.

Societies may not have the political maturity to deal with such immutable information. Governments, where power struggles naturally weaponize information, already relish the prospect of using digital technology to track and profile individuals suspected of threatening opinions and behaviours. Paranoia is a spiralling risk. Already today we confuse culture and nationality with intent: a stamp from the wrong country on your passport, or an incidental image on social media, or making an inappropriate comment in jest, can make you an enemy of the state. Being in the wrong place at the wrong time can be interpreted as a marker of hostility, because intent is as much inferred by promisees as it is promised by promisers.

The inability to forget may not be as positive as we think. Records of thoughts and actions have been preserved into familial and tribal grudges over many generations, leading to much violence. A technology with the ability to forgive and forget is not something anyone has so far seen fit to invent.

# References


[1] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis. Sok: Consensus in the age of blockchains. *arXiv:1711.03936v2 [cs.CR]*, 2017.

[2] BlockChain Hub. Blockchains & distributed ledger technologies, 2018.

[3] Mark Burgess. An approach to understanding policy based on autonomy and voluntary cooperation. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775*, pages 97–108, 2005.

[4] BBC. World business report, April 2018.

[5] S. Nakamoto. Bitcoin: a peer to peer electronic cash system. http://nakamotoinstitute.org/bitcoin/, October 31 2008.

[6] NIST. Des modes of operation. Technical report, NIST, 1980.

[7] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.

[8] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. (J.Wiley & Sons., New York), 1991.

[9] John Biggs. Sierra leone just ran the first blockchain-based election. http://tcrn.ch/2GtiaVI, March 2018.

[10] CoinTelegraph. Goodbye kickstarter? the blockchain-based project aims to challenge the crowdfunding sector. https://cointelegraph.com/news/goodbye-kickstarter-the-blockchain-based-project-aims-to-challenge-the-crowdfunding Feb 2018.

[11] BBVA. Bbva and indra deliver the world's first blockchain-supported corporate loan. https://www-bbva-com.cdn.ampproject.org/c/s/www.bbva.com/en/bbva-indra-deliver-worlds-first-blockchain-supported-co April 2018.

[12] Erik Kangas. Understanding blockchains (and bitcoin) – part 1: Concepts. The LuxSci FYI Blog, 2017.

[13] Erik Kangas. Understanding blockchains (and bitcoin) – part 2: Technology. The LuxSci FYI Blog, 2017.

[14] Erik Kangas. Understanding blockchains – part 3: Ethereum, or moving beyond bitcoin. The LuxSci FYI Blog, 2017.

[15] Aleksandr Bulkin. Explaining blockchain: how proof of work enables trustless consensus. Keeping Stock, 2016.

[16] M. Bishop. *Computer Security: Art and Science*. Addison Wesley, New York, 2002.

[17] J.A. Bergstra and M. Burgess. *Promise Theory: Principles and Applications*. $\chi tAxis$ Press, 2014.

[18] M. Burgess. *Thinking in Promises*. O'Reilly, 2015.

[19] J.A. Bergstra and M. Burgess. Money and ownership as an application of promise theory. Published online at markburgess.org.

[20] M. Burgess. Spacetimes with semantics (ii). `http://arxiv.org/abs/1505.01716`, 2015.

[21] J.A. Bergstra and M. Burgess. Local and global trust based on the concept of promises. Technical report, arXiv.org/abs/0912.4637 [cs.MA], 2006.

[22] M. Burgess and A. Couch. On system rollback and totalized fields: An algebraic approach to system change. *J. Log. Algebr. Program.*, 80(8):427–443, 2011.

[23] B.M. Oki and B.H. Liskov. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, PODC '88, pages 8–17, New York, NY, USA, 1988. ACM.

[24] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[25] Leslie Lamport. Paxos Made Simple. *SIGACT News*, 32(4):51–58, December 2001.

[26] Jim Gray and Leslie Lamport. Consensus on transaction commit. *ACM Trans. Database Syst.*, 31(1):133–160, March 2006.

[27] F.P. Junqueira, B.C. Reed, and M. Serafini. Zab: High-performance broadcast for primary backup systems. *IEEE/IFIP 41st International Conference on Dependable Systems and Networks*, pages 245–256, 2011.

[28] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA, 2014. USENIX Association.

[29] M. Burgess. Deconstructing the 'cap theorem' for cm and devops. http://markburgess.org/blog.html, 2012.

[30] M. Burgess. *In Search of Certainty: the science of our information infrastructure*. Xtaxis Press, 2013.

[31] M.P. Herlihy and J.M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463–492, 1990.

[32] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis. Chainspace: A sharded smart contracts platform. *arXiv:1708.03778v1 [cs.CR]*, 2017.

[33] Steemit. Dpos consensus algorithm - the missing white paper. https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper, 2017.

[34] M. Burgess. *A Treatise on Systems: Volume 2: Intentional systems with faults, errors, and flaws*. in progress, 2004-.

[35] Thomas Claburn. Bitcoin's blockchain: Potentially a hazardous waste dump of child abuse, malware, etc. The Register, March.

[36] J.A. Bergstra and K. de Leeuw. Bitcoin and beyond: Exclusively informational money. `ArXive:1304.4758v2 [cs.CY]`, 2013.

[37] J.A. Bergstra and P. Weijland. Bitcoin: a money-like informational commodity. `arXiv:1402.4778 [cs.CY]`, 2014.

[38] T.B. Lee. Bitcoin's transaction fee crisis is over—for now. Ars Technica, 2018.

[39] N. Tiku. Whistleblower threatens to expose corruption at bitcoin foundation. *ValleyWag*, 2014.

[40] S. Bano, M. Al-Bassam, and G. Danezis. The road to scalable blockchain designs. *USENIX ;login:*, 2017.

[41] L. Luu, D.H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, New York, NY, USA, 2016. ACM.

[42] EOS. Eos.io technical white paper v2, March 2018.

[43] G. Lee. Life beyond cryptocurrencies – hong kong and china fintech firms show there is more to blockchain. http://www.scmp.com/business/companies/article/2134488/life-beyond-cryptocurrencies-hong-kong-and-china-fintech-firm February 2018.

[44] C. Cachin. The architecture of hyperledger blockchain fabric. *Workshop on Disrtibuted Cryptocurrencies and Consensus Ledgers*, 2016.

[45] M. Burgess and H. Wildfeuer. Federated multi-tenant service architecture for an internet of things. `https://tools.ietf.org/html/draft-burgess-promise-iot-arch-00`, October 2015.

[46] P. Borrill, M. Burgess, M. Dvorkin, and H. Wildfeuer. Workspaces. Technical report, 2015.

[47] ConsenSys. A 101 noob intro to programming smart contracts on ethereum, October 2015.

[48] Stack Exchange. Upgradeable smart contract question. http://ethereum.stackexchange.com/questions/2404/upgradeable-sm 2016.

[49] D.D. Clark and D.R. Wilson. A comparison of commercial and military computer security policies. *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, page 184, 1987.

[50] D.E. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. *MITRE technical report, MITRE Corporation, Bedford Massachusetts*, 2997:ref A023 588, 1976.

[51] H.T. Kung and C.H. Papadimitriou. An optimality theory of concurrency control for databases. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, pages 116–126, New York, NY, USA, 1979. ACM.

[52] P. Bailis. Highly available, seldom consistent: Data management, distributed systems, and beyond. http://www.bailis.org/blog/linearizability-versus-serializability/, 2014.

# A Appendix about data consistency terminology

Linearizability and serializability are about the interleaving of operations in databases and distributed systems [31, 51]. This appendix is based on the note by Bailis [52].

**Linearizability** is a guarantee about single operations on single agents. It provides a 'real time' i.e. wall-clock promise about the behaviour of a stream of single imposed operations (often reads and writes) onto a single network agent. Linearizability means that writes should appear to be locally instantaneous, i.e. once a write promises completion, all subsequent reads (by the wall-clock) should return the value of that write, until the next once succeeds it.

Linearizability for read and write operations is synonymous with the database term 'atomic consistency' in ACID. Linearizability is *composable* or local (transitive by aggregation) because, if operations at each agent in a system are linearizable, then all operations in the system are linearizable.

**Serializability** is a guarantee about transactions imposed on one or more agents. It promises that the imposition of a set of transactions (read and write operations) across multiple agents is equivalent to a single total ordering of the transactions to the combined single superagent. Serializability is the traditional 'I' or 'isolation' in ACID. If user transactions each preserve application correctness ("C," or consistency, in ACID), a serializable execution also preserves correctness. Therefore, serializability is a promise about database correctness.

Unlike linearizability, serializability does not claim real-time constraints on the ordering of transactions. Serializability is also not composable. Serializability does not imply any kind of absolute time ordering, it simply requires that some equivalent serial execution exists.

**Strict serializability** Combining serializability and linearizability: transaction behavior is equivalent to some serial execution, and the serial order corresponds to real time. For example, say I begin and commit transaction $T_1$, which writes to item $x$, and you later begin and commit transaction $T_2$, which reads from $x$. A database providing strict serializability for these transactions will place $T_1$ before $T_2$ in the serial ordering, and $T_2$ will read $T_1$'s write. A database providing serializability (but not strict serializability) could order $T_2$ before $T_1$.

Linearizability can be viewed as a special case of strict serializability where transactions are restricted to consist of a single operation applied to a single object [31]. Neither linearizability nor serializability can be promised without coordination, i.e. availability of communications (also called availability). Thus they are conditional promises.

In practice, a database is unlikely to provide serializability, and your multi-core processor is unlikely to provide linearizability at least by default. As the above theory hints, achieving these properties requires a lot of expensive coordination. So, instead, real systems often use cheaper-to-implement and often harder-to-understand models. This trade-off between efficiency and programmability represents a fascinating and challenging design space. A note on terminology, and more reading

One of the reasons these definitions are so confusing is that linearizability hails from the distributed systems and concurrent programming communities, and serializability comes from the database community. Today, almost everyone uses both distributed systems and databases, which often leads to overloaded terminology (e.g., 'consistency', 'atomicity').