SOK: a comprehensive survey on distributed ledger technologies

Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi, Abdellatif Mezrioui

SOK: A Comprehensive Survey on Distributed Ledger Technologies

Badr Bellaj [*] [$], Aafaf Ouaddah[*], Emmanuel Bertin [§](IEEE Senior), Noel Crespi[$](IEEE Senior), Abdellatif Mezrioui[*]
[$]Telecom SudParis, Paris, France, [*]Institut National des Postes et Télécommunications, Rabat, Morocco, [§]Orange, France
Corresponding e-mail: bellaj.badr@mchain.uk

***Abstract*—In recent years, Blockchain arose as a key technology in building autonomous decentralised financial systems. Its ability to digitize trust enables building trustless systems such as cryptocurrencies where users do not need to rely on any third party to exchange value. The success of cryptocurrencies to operate without any intermediaries draw the interest of business operators who seek to bypass intermediation and thus to reduce cost and gain competitive advantages. As a result, Blockchain was used outside the crypto-sphere to build decentralized systems. However, this portage led to the inception of new types of Blockchains adapted to different specifications and with different designs. Consequently, the technology has diverged from its baseline (Bitcoin) to the point where some systems marketed as "blockchain" share only a few design concepts with the original Blockchain design proposed by Satoshi Nakamoto. This conceptual divergence alongside the lack of comprehensive models and standards made it difficult for both system designers and decision-makers to clearly understand what is a blockchain or to choose a suitable solution.**

This survey has a double goal; on the one hand, it attempts to contribute to the discussion on the ontological status of DLTs by providing a taxonomy oriented-framework (DCEA) for conceptualizing and examining DLT. On the other hand, it also attempts to present an up-to-date review and evaluation of current blockchains and their variants as constructed of four layers: the data, consensus, execution and application layers.

***Index Terms*—blockchain, blockchain-like, DLT, survey.**

I. Introduction

The blockchain ecosystem is evolving very quickly from the inception of Bitcoin [1] in 2009 to the recent emerging enterprise version of DLT. This lifetime was marked by important milestones. The first evolutionary milestone was the launch of Bitcoin, considered by many as blockchain 1.0, along with its variants (e.g. Litecoin [2], Peercoin [3], etc.). The second major milestone was the launch of Ethereum in 2015, and many other blockchain-like projects such as IOTA [4], Hyperledger [5], and Hashgraph [6] solutions. This second generation introduced radical changes to the initial design of the Nakamoto's blockchain (Bitcoin), giving birth to a new category of technologies which can be considered as blockchain 2.0. The emergence of this category which was heavily inspired by blockchain but without being one, drove the industry to adopt a broader term; "Distributed ledger technology", or DLT when referring to this category. Nevertheless, there was no rigorously defined set of terminologies or commonly acceptable taxonomy delineating the border of each category. As a result, terms like "blockchain", "DLT" or even "distributed database" were misunderstood, misused, and misinterpreted and many projects or enterprises use the word "blockchain" extensively, simply as marketing jargon. Although there are multiple proposals for standardizing blockchain (ISO [7]-[8], IEEE [9], ITU [10]), there is no recognized standard for defining blockchain or DLT, thereby some difference of opinion is possible as to the degree to which a system is blockchain or not. Moreover, we observe the growing use of ambiguous, imprecise, and inconsistent language and terminology across different projects— where the same term may be used to refer to different things— which could hinder the development of the DLT sector and its mass adoption. Another consequence of this discordance and lack of standardization is the lack of interoperability between DLT networks. In fact, the systems within the DLT ecosystem are inter-siloed and unconnected as interoperability is deprioritized in favor of rolling out new systems with better performance, due to fierce competition and commercial pressure. However, in the absence of a technical reference model, it is difficult to measure and compare the quality and performance of these systems. By recognizing all these concerns, we try herein to draw the boundaries between different categories in the DLT ecosystem by proposing a new taxonomy differentiating the distinctive differences between the existing groups. We also undertake a systematic and holistic approach to conceptualize and examine DLTs in general as a functioning system with key layers at four levels of analysis. The structure of the paper is as follows. Section II defines a new taxonomy-oriented framework serving to normalize and deliberates on the classification and taxonomy of DLTs. Sections III, IV, V and VI present, respectively the data, the consensus, the execution and the application layers, where in each section we outline main components and properties as well as the state of the art of the studied layer. Section VII presents a global comparative evaluation and critical analysis of 44 different DLTs in the academic as well as the industrial field through the prism of the proposed referential framework. Finally, we close with a conclusion and future research directions in section VIII.

II. DCEA a taxonomy oriented-framework for conceptualizing and examining DLTs

We propose DCEA, a framework that defines a layered heterogeneous stack for DLT systems. From a design perspective,

our conceptual framework (DCEA) segregates DLT technologies into four essential and distinct layers: data, distributed consensus protocols and network organization, execution and application layers — each one playing a well-defined role in the DLT architecture. The application of the DCEA framework leads to a double level classification of DLTs. The first classification is based on the different settings of key designs and the second is a two-dimension high-level taxonomy (blockchain and blockchain-like) that considers the impact of the different settings of DCEA at three levels (Figure 1). The result of the extended review and taxonomy presented in the sections III, IV, V and VI are invested , in section VII, to evaluate and classify a wide set of DLTs into the two high categories.

A. Presentation of DCEA framework

The framework consists of the DLTs components and their main properties (Table I), with logically related functions grouped together. This layering approach is aligned with the DLT's modular architecture. That is, it will help to provide a better understanding of DLTs, and serves as a baseline to build a comparative analogy between different DLT variants.

In the following, we introduce the four layers that form the DLT stack.

- Data Layer: Represents the data (transactions and states) flowing through the distributed network and stored in the ledger. Data in this layer is represented by entries recorded in the ledger, under consensus and shared amongst the network participants. These records may represent elements defined by the underlying protocols (such as cryptocurrency, or smart contracts), or data received from external environments (such as IOT data). Generally, the data layer covers data stored on the blockchain itself (on-chain storage) as well as data stored in an auxiliary source using a distributed database (off-chain storage).
- Consensus layer: Defines the global software-defined ruleset to ensure agreement among all participants, in a network, on a unified ledger. Consequently, this layer designates the formal rules that govern the system.
- Execution layer: Represents the components responsible for enforcing and executing distributed programs (e.g. smart contracts). Basically, these programs or contracts codify a given logic (e.g. a business logic) as a set of instructions for manipulating the states recorded in the ledger.
- Application layer: Represents an abstraction layer that specifies a variety of protocols and APIs provided by the DLT system to enable the building of distributed applications commonly called DApps. This layer also represents a communication link between the external actors or applications and the code hosted on the DLT ledger.

Based on the above layering, we propose a four-layered taxonomy , to categorize DLT systems. The purpose of the taxonomy is to:

- Classify the typical DLT systems proposed in the academia and in the industry; and to
- the relative strengths and weaknesses of existing systems and identify deficiencies in the current set of DLTs.

At each layer, DLTs adopt different settings for DCEA properties defined in Table I. Based on their combinations, at the four layers, we can define different DLT classes. For instance, at the data layer we differentiate between DAG-based and chain-based DLTs based on the nature of the underlying data structure; at the consensus layer we differentiate between permissioned and permissionless DLTs based on the identity model of the consensus mechanism; at the execution layer we differentiate between Smart-contract based DLTs and script based-DLTs; and at the application layer we can differentiate between DApps-oriented DLTs and Cryptocurrency-oriented. Subsequently, this cross layer wise categorization help us to classify all DLT systems into two major classes, namely, blockchain and blockchain-like systems, with regard to DCEA properties.

III. DATA LAYER

In this section, we lay out the key components, and their characteristics, that construct the data layer as introduced in Table I. A summary of the state of the art of different data structures adopted by major DLT solutions is also presented. We end this section with a discussion of key challenges and tradeoffs.

A. Components and properties

DLT's ledger basically represents a distributed data store where data is duplicated among multiple nodes, by means of data synchronization. In these data stores, the data organization in its macroscopic structure varies from one technology to another. Generally, we distinguish between two main models of data structures in the DLT space; the linear chain of blocks and the chain-less models.

1) Chained model:

a) Chain of blocks: Data in the chain of blocks is organized in elementary units called blocks. Each block is a collection of transactions validated by the network. These units are organized chronologically as a chain of inter-hinged blocks, which are tied by tamper-evident hash pointers. Each new block can only be valid if it is built upon an unchangeable body of previous blocks. Blocks are composed of a header

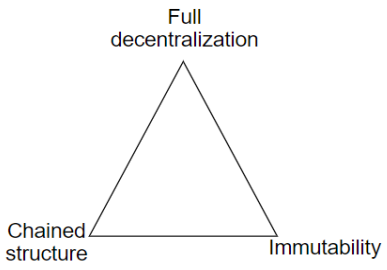

Fig. 1. As per our definition, a blockchain-like system can only have two features of these traits.

TABLE I
LAYERS AND COMPONENTS OF DCEA FRAMEWORK

Application Layer	Integrability			DLT orientation and purpose			Wallet and identity management		
Execution Layer	Execution environment		Turing-completeness		Determinism		Openness	Interoperability	
Consensus Layer	Safety	Liveness	Finality	Network model	Failure model	Adversary model	Governance model	Transaction ordering	Conflict resolution
Data Layer	Data structure		Data shareability		Data immutability		States storage		

and a record of transactions. The block's header contains meaningful metadata such as a cryptographic reference to the previous block and the current time. This linear linkability ensures data integrity through cryptographic connections between blocks and enables each participant in the network to verify and validate data. Data in a chain of blocks is carried over and stored in the ledger using transactions, we therefore consider a transaction as the most elementary data type. At the block level, the transactions are ordered and hashed into a Merkle tree, with the hash root placed in the corresponding block's header. This structure guarantees a cryptographically sealed and tamper proof data vault resistant to any type of data corruption.

b) Skipchain: The data structure of a skipchain is inspired by skip lists [11]. It adapts the skip list idea to the chain of blocks by adding links between blocks both forwards and backwards in time. In skipchain, a block includes not just a single hash link to the previous block, but also an additional hash link to a point farther back in time. Thus, skipchain can build subsequent layers of linked blocks on top of an original linked list of blocks. Skipchain is very useful when one wants to concurrently access the data structure.

2) Chainless model:

a) Chain of blocks: In order to overcome some limitations imposed by the adoption of the chained block structure, certain DLTs have opted for a chain-less model. Instead, they use new data structures for better scalability or security.

b) DAG: In contrast to using a chain of blocks, some DLTs are using a nonlinear structure such as the Direct Acyclic Graph (DAG) to offer better performance. A DAG is a graph that grows in one direction without cycles connecting the other edges (i.e., there is no path from a vertex back to itself). As with a chain of blocks, a DAG is used as a sorted collection of hierarchically connected transactions where each transaction and sometimes a block of transactions is represented by a node (which refers to a vertex in the graph) and linked to at least another node. The DAG is extended chronologically by appending new transactions to the previous nodes. The ledger is thus an ever-growing tree, starting initially from a root node. The acyclic nature of the DAG and its unidirectional evolution enables participants to confirm transactions automatically based on previous transactions. Based on the representation of its nodes, we identify two types of DAGs:

- Transaction-based DAGs: DAG nodes that represent individual transactions; and
- Block-based DAGs: DAG nodes that represent a block of transactions.

c) Decentralized database model: Some DLTs adopt radical changes in their architecture over the conventional blockchains, to the point they resemble a classical distributed database. We consider these solutions as decentralized databases, as they manage data similar to how conventional databases handle data but they present a different technology. In fact, unlike in a conventional distributed database, where nodes cooperate to maintain a consistent view of data across all systems, a decentralized database is designed to allow multiple parties that may not trust each other to collaborate with their peers to maintain shared records.

d) Hybrid data model: Some DLT projects combine both chains of block models along a block-less model to manage transactions and states in the network. The hybridization is designed to exploit the advantages of each model to enable better scalability and rapid transaction validation. In this model, the states are generally stored in external dedicated key-value databases and the blocks contain only the transactions affecting the ledger's states. Using key-value databases makes it easy to directly access the updated value of a state rather than having to calculate it by traversing trees of transactions.

3) State management: A key distinguishing factor among various DLTs, is how states are managed within the system. Although DLTs serve as distributed ledgers for shared data, in the case of many DLTs, data is stored outside the transactional distributed ledger (off-chain/off-ledger) using auxiliary databases. Conventional blockchains, however, tend to always store data on the shared ledger (on-chain/on-ledger). When we analyze how general states (e.g. user's balance) are managed in existing DLTs, two models emerge: UTXO model, Account model. The first, is a special transactions set, linking new transactions to old transactions, wherein a newly produced transaction (new UTXO) points to a single or multiple ulterior transactions (inputs), whereas, the second is a model where the ledger keeps track of up-to-date global states related to each account. A deep analysis of the UTXO model can be found in [12].

4) Data shareability: All nodes in a DLT network exchange transactions carrying shared data in order to reach consensus, but due to privacy reasons different visions of data shareability have been adopted. Some systems favor complete shareability of all data — which we consider as global shareability—, whereas others restrict the perimeter of shareability including some nodes and excluding others — which we consider as restricted shareability.

5) Data immutability / Atomicity: There is a common belief that records stored on a DLT (especially a blockchain) are

immutable and unalterable. However, that is not necessarily the case, as different DLT systems provide different degrees of immutability depending on the system design. This means that, under some circumstances, nodes can hold inconsistent states, or that a confirmed transaction may be reversed. Section VII provides a more detailed overview of the transaction finality process. For data immutability, we differentiate between:

- Strong immutability. When the state variables or blockchain entries cannot be mutated after their creation; and
- Weak immutability. When the state variables or blockchain entries can be mutated after their creation.

It is worth noting that for some strong immutable systems, their states can be updated without breaking immutability. This is achieved by using tree structures to store persistently both new and old values for a given entry.

B. Data layer: state of the art

This subsection is meant to present an overview of DLTs projects adopting the different data structures previously outlined by our framework as well as an evaluation of their properties.

1) Chained DLTs: Most DLTs follow the linear data chain structure initially defined by Bitcoin. In this broad category, multiple projects define different inner block structures.

a) Bitcoin: In Bitcoin and its clones, transactions are assembled in the block's body and then linked in a Merkle tree. The root of this tree, or the Merkle root, is a hash representing an indirect hash of all the hashed pairs of transactions in the tree and is included in the block header, thereby ensuring transaction verification. In addition to the Merkle root, the block header also contains other important information, including: the timestamp, and the previous block's hash. Moreover, Bitcoin adopts the UTXO model to track the system states (Wallet balances). The UTXO set is stored off-chain in an auxiliary database.

b) Ethereum: The block structure is more complex in Ethereum than in Bitcoin, and the system's state tracking is different than in Bitcoin. In fact, the block's header comprises more metadata and its body englobe multiple types of data, namely: transactions, receipts and system states. Each of these data types is organized into a Merkle tree or a Patricia tree (Radix tree) in the case of the state tree. The state tree is an important component in the Ethereum ledger, as it is used to implement the account model, whereby each account is linked to its related states (account balances, smart contract states, etc.). Any node can parse the tree and get the updated state without any overhead calculation. The state tree grows each time a change occurs in a state. It grows by adding new nodes (stored in the new block) — containing new states— which points to the nodes (stored in the previous block) containing the old value for the same state. To enforce immutability Ethereum keeps its root hash in the block header.

c) Bitcoin-NG: Introduced by Eyal et al. in [13], adopts a chain of block data structure, with two types of blocks — key-blocks and Microblocks. Keyblocks are produced through proof-of-work, with an identical structure to Bitcoin's blocks, in order to determine the block producer (miner or leader). Between two key-blocks, the selected miner creates and signs multiple Microblocks, that contain many smaller batches of collectively signed transactions. The Microblocks reference previous Microblocks and Keyblocks forming a chain.

d) ByzCoin: Inspired by Bitcoin-NG, ByzCoin [14] adopts the decoupling in two blocks but instead of having a single chain, it forms two separate parallel chains of Keyblocks and Microblocks.

2) Skipchain:

a) Chainiac: Nikitin and al. [15] introduced Chainiac to solve offline transaction verification problems (enable nodes to check if a transaction has been committed to a blockchain without having a full copy of the ledger). The Chainiac solution was to add traversability forward in time using a skipchain , where back-pointers in Chainiac are crypto-graphic hashes, whereas forward-pointers are collective signatures. With long-distance forward links and via collective signatures, a client or node can efficiently verify a transaction anywhere in time.

3) Chainless DLT:

a) DAG based chains: The idea of using DAGs as underlying data structure has encountered great interest from DLT designers of multiple projects, including Byteball , DagCoin IOTA, Nano , Phantom and Hedera . Some studies have tried to introduce DAG in conventional blockchain DLTs, for instance the GHOST protocol [16] proposes a modification of the Bitcoin protocol by making the main ledger a tree instead of a blockchain.

b) IOTA: [4] is a DLT system designed primarily for the Internet-of-Things (IoT) industry. IOTA is one of the first projects that adopted a DAG data structure called Tangle —a block-less DAG where individual transactions are tangled together. The Tangle stores transactions as edges; each single transaction references two direct previous transactions and indirectly references a subsection of the Tangle. That is, the tangle is simultaneously constructed of validated transactions, transactions waiting for validation and uncertain validated transactions.

c) Byteball: [17] is another transaction-based DAG, where a transaction references directly one or more previous transactions. Once a new transaction is added to the DAG by a node, it becomes visible to its peers; those peers can append their child transactions on top of this new one.

d) Nano: [18] Formally called RaiBlocks, Nano is based on a special DAG data structure called Blocklattice. Each account has a dedicated chain of blocks) that are replicated to all peers in the network, thereby allowing multiple single chains to grow concurrently. In this structure, only the owner of the wallet can make changes to the individual chains. This means that each wallet can be updated asynchronously.

4) Decentralized Databases:

a) Corda R3 [19]: In the corda network, each node maintains a local database called a "vault" that stores time-

stamped data. Each vault has many different versions (current and historic) of data in the form of state objects.

5) Hybrid DLTs:

a) Hyperledger Fabric [20] : Hyperledger Fabric combines between the usage of a chain of blocks to store only the validated transactions, and the usage of a key-value classical database to store the system's states (transaction outcomes). In the Fabric chain, the block structure resembles the structure of a block in a conventional chain but with an additional part: block metadata. This additional section contains a timestamp, as well as the certificate, public key and signature of the block writer. The block header is straightforward and the transactions are ordered in the block body without Merkilization.

b) EOS [21]: EOS uses a chain of blocks as an immutable store of transactions and a mutable database holding system states. The EOS database adopts an account model similar to a distributed database with permissions, where accounts represent user profiles rather than a simple cryptographic identity. In the EOS database the states can be updated directly in the database tables, thus old values cannot be retrieved from Chainbase, all states are reproducible at any specific time by replaying transactions from the genesis block.

c) BigchainDB: The BigchainDB [22] was introduced as a blockchain database. It aims to combine the key characteristics of "traditional" NoSQL databases (MongoDb) and the key benefits of traditional blockchains. BigchainDB server nodes utilize two distributed databases (transaction set or "backlog") holding incoming transactions and a chain of blocks storing validated transactions (Creation or Transfers). Each transaction represents an immutable asset (represented as JSON documents in MongoDB).

6) Data shareability: Most DLTs operating as global cryptocurrency platforms adopt by design a global shareability of the transactions. In fact, networks such as Bitcoin, Ethereum and many others, operate in relay mode where nodes are relaying transactions to each other, thereby propagating it to the entire network without restrictions. In other DLTs, such as Hashgraph, senders deliver their transactions to a set of selected nodes that are responsible for including them into their DAG and sharing them with others by Gossiping. On the other hand, the DLTs constructed for business purposes, such as Corda or Hyperledger Fabric, impose restricted shareability of the transactions as privacy is an important requirement in such contexts. In Corda, for instance, each node maintains a separate database of data that is only relevant to it. As a result, each peer sees only a subset of the ledger, and no peer is aware of the ledger in its entirety. In Fabric a subset of the ledger restricts data shareability by using the concept of channels [23]. A channel is a private sub network between two or more specific network members. Each transaction on the network is executed on a channel, where only authenticated and authorized parties are able to transact on that channel. Therefore, the network ends up with a different ledger for each channel.

IV. Consensus Layer

DLTs have renewed the interest in the design of new distributed consensus protocols. In fact, a myriad of consensus algorithms, for DLT, have been proposed in the literature presenting different properties and functionalities. In this section, we present the properties and features we consider as part of the DCEA framework for studying and differentiating between the protocols. We also present the state of the art in the second subsection. In section VII we present and discuss the results of a comparative analysis of the studied protocols.

A. Components and properties

1) Basic Properties: The concepts of safety and liveness properties were introduced initially by Lamport in 1977 [24] and have been well adopted in the distributed computing community. All consensus algorithms provide these properties under different assumptions of synchrony, adversary model, etc.

a) Safety: Safety represents in the context of DLT networks, the guarantee that the correct nodes will not validate conflicting outputs (or make conflicting decisions) at the same time (e.g. chain forks). The safety property ensures; Availability whereby transactions submitted by an honest user get incorporated into the ledger sufficiently fast [25].

b) Liveness: A consensus protocol guarantees liveness if requests (transactions) from correct clients are eventually processed.

c) Finality: In the DLT settings, we define the finality property as the affirmation and the guarantee for a transaction to be considered by the system as final and irreversible. The Finality as property can be divided into two types:

- Probabilistic finality, where the probability that a validated transaction will not be reverted, increases with time once the transaction is recorded onto the ledger.
- Absolute finality, where a transaction is considered finalized once it is validated by the honest majority.

2) Network models: In both traditional distributed systems literature and DLT consensus protocols, we consider the message passing model in which nodes exchange messages over the network, under differing assumptions of network synchrony. We adopt in this survey the following taxonomy defined by [26].

- Synchronous, where we assume the existence of a known upper bound on message delay. That means, messages are always delivered within sometime after being sent.
- Partially-synchronous, where we assume there is some known Global Stabilization Time (GST), after which the messages sent are received by their recipients within some fixed time bound. Before the GST, the messages may be delayed arbitrarily.
- Asynchronous, where messages sent by parties are eventually delivered. They may be arbitrarily delayed and no bound is assumed on the delay of messages to be delivered.

3) Failure Models: Different failure models have been considered in the literature; we list hereafter two major types.

- Fail stop failure (Also known as benign or crash faults): Where nodes go offline because of a hardware or software crash.
- Byzantine faults: This category of faults was introduced and characterized by Leslie Lamport in the Byzantine Generals Problem [27] to represent nodes behaving arbitrarily due to software bugs or a malicious compromise. A Byzantine node may take arbitrary actions, provide ambivalent responses or intentionally mislead other nodes by sending sequences of messages that can defeat properties of the consensus protocol.

We consider, therefore, a protocol as fault tolerant, if it can gracefully continue operating without interruption in the presence of failing nodes.

4) Adversary models: We consider a message passing model where a node communicates by exchanging messages with its neighbors. The adversary is able to learn the message exchange and to corrupt different parts of the network.

- The Threshold Adversary Model (Hirt and Maurer) : This model is the most common adversary assumption used in the traditional distributed computing literature, which assumes that the Byzantine adversary can corrupt up to any f nodes among a fixed set of n nodes. Under this model, the network usually has a closed membership requiring a permission to join. The consensus protocol should be able to operate correctly and reach consensus in the presence of Byzantine nodes as long as their numbers do not exceed a given threshold.
- Computational Threshold Adversary: A new model introduced by Bitcoin, where the control of the adversary over the network is bounded by the computational power –requiring concrete computational material— instead of the number of nodes he can control. In this model, typically the membership is open and multiples parties and the bounding computation is a brute force calculation.
- Stake Threshold Adversary : In this model the adversary control is bound by his proportion of a finite financial resource. In networks managing cryptocurrencies, the underlying protocol can ensure consensus based on cryptocurrency deposits, thus the adversary is bound by the share of cryptocurrency he owns. In addition, in these protocols' punishment rules (e.g. stake slashing) could be put in place to deter bad behaviour.

Adversary Modes Protocols assume the existence of different types of adversaries based on their ability and the time they need to corrupt a node.

- Static adversary: A Byzantine user who is able to corrupt a certain number of network nodes ahead of time and exercise complete control over them. However, he is not able to change which nodes they have corrupted or to corrupt new nodes over time.
- Adaptive adversary: A Byzantine user who has the ability to control nodes and dynamically change, depending on the circumstances, the nodes under his control to gain more power.
- Mildly adaptive adversary: A Byzantine user who can only corrupt nodes based on past messages, or its anticipations, and cannot alter messages already sent. Moreover, the adversary may mildly corrupt groups, but this corruption takes longer than the activity period of the group.
- Strongly adaptive adversary: A Byzantine user can learn of all messages sent by honest parties, and based on their content, he can decide whether or not to corrupt a party by altering its message or delaying message delivery.

5) Identity Model: Protocols manages nodes membership differently, but in general two opposite sides are adopted:

- Permissionless, where the membership is open and any node can join the network and validate new entries.
- permissioned, where the membership is closed and only a restricted set of approved members is able to validate new entries.

6) Governance Model: Governance model refers to the process of decision-making adopted by a DLT network to decide on the protocol rules and their upgrade. Hence, the governance of the system boils down to a social concept, we find it appropriate to identify some of the possible governance model, from a social perspective:

- Anarchic, where protocols upgrade proposals are approved by every participant in the network. Each participant chooses to accept or reject a given proposal, thus leading to potential splits in the network.
- Democratic, where participants vote on new rules and protocol upgrades proposals and at the end all participants have to follow the decision of the majority, even for those participants who voted against them.
- Oligarchic, where new rules and protocol upgrades are proposed and approved by a group of participants.

As most DLTs move governance and related issues "on-chain" or "off-chain" we consider also the differentiation between; Built-in (or on-chain governance), where the decision-making process in the network is defined as part of the underlying consensus protocol; External governance (or off-chain governance), where the decision-making process is based on procedures independently performed without involving the DLT mechanisms.

7) Transactions ordering: Whether for a linear or a non-linear DLT (e.g. DAGs), the stored transaction should be ordered chronologically to avoid frauds and inconsistencies. Different approaches have been introduced by the consensus protocols to provide reliable and fair transaction ordering. Usually, in DLTs the ordering is an integrating part of the consensus mechanism but in some cases, it can be decoupled from the execution and validation of transactions. Ordering is an important property with direct impacts on the security and the usage of a DLT, thus the need to evaluate this feature separately.

8) Conflict resolution model: In some DLT networks, conflicting temporary versions of the ledger (known as forks) can coexist for different reasons (e.g. network latency, parallel validation of blocks, etc.). To converge toward a canonical ledger or chain, networks and consensus mechanisms adopt different rules. The most notable rule is defined by Bitcoin protocol as "longest chain rule", whereby in the presence of conflicting orders, the network converges to one order following the longest chain — the chain with the largest accumulated PoW in case of PoW-based systems— and discards the rest. The longest chain rule is adopted by different protocols and each may adopt a different cumulative parameter (witnesses votes, endorsement, etc.).

B. Consensus layer: state of the art

In this subsection, we present multiple consensus mechanisms and their properties. Although, is out of scope of this paper to present a detailed taxonomy of the existing protocols (fig. 2), we consider to group all the reviewed protocols in six categories. This protocol categorization serves us as a basis to categorize the DLTs.

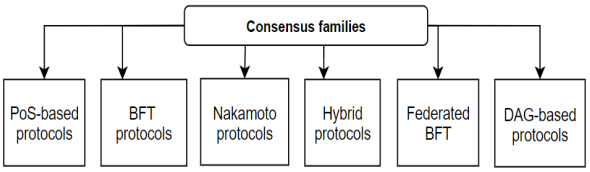

Fig. 2. Taxonomy of consensus protocols

1) BFT consensus family (PBFT-like): This family refers to the classical consensus mechanisms introduced in the traditional distributed computing literature and their recent variants. The BFT family is easily recognized due to their property: all-to-all voting rounds , the identity of the nodes in the network is known, the number of participants is limited. Due to the big number of the protocols belonging to this family, we limit our review, in this paper, on the most used algorithms in DLT context namely; PBFT, RAFT, IBFT, DBFT, POA (AURA, Clique), HoneyBadgerBFT and Hotstuff.

2) Nakamoto consensus family: We consider that Nakamoto's consensus family represents protocols using a chain of block data structure and adopting the longest chain fork choice rule (or a variant like GHOST), to ensure safety, along economic incentives. These protocols were introduced primarily to enable secure currency transfer over the internet. Conversely to PBFT, they are conceptually simple and tolerate important corruptions up to *n/2*. Besides, they are known for being permission-less (open enrollment) — they do not require node authentication and allow nodes to arbitrarily join or leave the network. We review hereafter some of the most discussed protocols in this category namely: PoW, memory bound PoW and BitcoinNG.

3) Proof of stake and its variants: Proof-of-Stake (PoS) was first proposed as an alternative to the costly PoW for use in the PPCoin . Instead of a hash-calculation competition between validators, participants who desire to join validators board and forge the new block have to lock a certain amount of coins into the network as a financial stake. Thus, the chances for a node to be selected as the next validator depends on the size of the stake. Different implementations of PoS exist. We present here some of the typical representatives including Ethereum PoS, DPoS (EOS), Ouroboros and its variants, and Snow white.

4) DAG-based Protocols:

a) IOTA [4]: is a hybrid consensus protocol, marrying between PoW at the entry level and a custom transaction validation algorithm. IOTA relies on PoW to protect the network against spamming as the transactions are fee-less. In order to add a new transaction to the IOTA underlying DAG (called Tangle), the end-user (or IOT end-devise) has to perform a proof-of-work. Moreover, the new transaction should randomly approve two previous valid transactions (called tips) by extending them and thus increasing their initial weights. To choose the tips a user can eventually(as no rule is imposed by the protocol) use the recommended Markov Chain Monte Carlo (MCMC) weighted random walk, which is biased toward transactions with more weight. Moreover, the IOTA network has special nodes called Coordinators —managed by the IOTA foundation— responsible for protecting the network by recognizing periodically (often indirectly) the valid transactions in the tangle by issuing a signed milestone transaction. These milestones checkpoints serve to finalize transactions and to indicate the main tangle to the nodes that might be building a sub-tangle.

b) Avalanche: Avalanche is a recent leaderless Byzantine fault tolerance protocol built on a metastable mechanism via network subsampling. Avalanche protocol is based on a metastable mechanism, whereby a node repeatedly takes a uniform random sample from the network, sends queries repeatedly in multiple rounds and collects responses. Once a given threshold is met the node adopts the decision transmitted by the majority. Consequently, the correct nodes converge, with low communication complexity, to the same decision without requiring a leader. To improve its efficiency and security, Avalanche organises transactions in a DAG, where a new transaction extends one or more parent transactions. In presence of conflicting transactions, Avalanche harnesses the DAG structure and uses a transaction's progeny (all children transactions) such that corrected nodes vote positively on valid transactions based on the validity of the entire ancestors. With regard to safety, Avalanche provides a strong probabilistic safety in the presence of $f \leq n/3$ Byzantine nodes. Further, the concurrent random sampling and leaderless nature enables it to achieve high throughput and scalability. Furthermore, Avalanche adopts a financial mechanism (AVA token) to protect the network against sybil attacks, since it provides open membership, as well as enabling economic governance.

c) Hashgraph [28]: is an asynchronous Byzantine fault torrent protocol. Hashgraph uses a DAG as data structure for storing events (transaction and its related details) and a voting algorithm combined with gossip protocol to reach consensus. In Hashgraph a node randomly gossips about its known events with other nodes either received from other nodes or initiated by the node itself. This gossip process allows for an exponential diffusion of the transactions. Moreover, the shared event includes two hashes referencing two past events (the previous event created by the gossip receiver and the event created by the sender), which is then signed by the Gossiping node. Therefore, every node of the network ends up with a copy of the transaction history along with information about the other nodes that previously received the information. To ensure agreement on the transactions order and validity, nodes proceed to a virtual voting, where no actual votes are cast or exchanged but instead a node calculates what other nodes should vote, based on its knowledge of the DAG (the gossip history) and events timestamping. If any transaction is validated by 2/3 of the nodes in the network, then it is considered as a valid transaction. It is worth noting that the vote is possible as currently the number of nodes is known because of the closed membership.

5) Federated BFT: Ripple [29] was the first implementation of a federated Byzantine agreement system (FBAS, for short), which was extended later by Stellar [30] protocol. FBA revisits BFT settings by providing an open membership service based on a trust model. In fact, FBA protocols depart from the concept that each node interacts only with a limited group of its trusted peers — the unique node list (UNL) in Ripple and the quorum slice in Stellar. Thus, unlike traditional BFT protocols, the federated Byzantine agreement (FBA) does not require a global and unanimous agreement among the network participants. Ripple consensus (or RPCA) proceeds in rounds where validators of a server's UNL try to reach a supermajority (at least 80%) on a set of transactions. Thus, as long as 80% or its equivalent of the UNL are honest, validated transactions are applied to the new last closed ledger. Therefore, Ripple tolerates a Byzantine minority of $f < n/5$ nodes to operate correctly. However, if more than 20% of nodes in the network disagree with the rest, the network may temporarily halt. In addition, the good configuration of servers and the intersection of a pair of correct nodes' UNLs determine the network's safety and liveness. According to the ripple project, the minimum overlap requirement is 20% of the UNL by any two nodes in different UNLs. However, an independent analysis [31] suggested that the correct requirement was instead about 40%. Stellar Consensus Protocol (SCP) is an evolution of Ripple protocol providing first provably safe consensus —assuming the transitivity and the strong connectedness of the network. Unlike Ripple, SCP allows a lot of flexibility in terms of how nodes configure their quorums. In order to assure network consensus, SCP relies on Federated voting [32], which allows the network to reach consensus as long as quorum slices intersect with each other.

V. Execution Layer

In this section we identify the fundamental components of the execution layer, and their properties. Then we present the execution component widely adopted in the state-of-the-art.

A. Components and properties

In a DLT system, business logic, agreed to by counterparties, can be codified using a set of instructions and embedded into the ledger in specific format. The ruleset execution is enforced by the distributed consensus mechanism, thereby an external actor cannot influence or corrupt the execution of the instruction set to get advantageous results. Generally, we distinguish between two main models for rules codification: Smart contracts and built-in scripts.

1) Execution environment:

a) Smart contract model: In this model, clauses between counterparties are codified as a self-executed program acting on a defined set of states. Typically, this program (known as smart contract) is implemented either in a dedicated language or using an existing programming language such as Java or C++. The smart contract execution is handled by a dedicated environment such as a virtual machine or a compiler, which proceeds the clauses defined in the triggering transaction, returns an output and often results in updated states. Commonly, the smart contracts live and executes on the DLT as an independent entity with reserved states and tends to offer a generic oriented tool to implement versatile logic beyond the manipulation of a native asset (Token or cryptocurrency). Although they are qualified as 'smart', they are not autonomous programs, as they need external triggering transactions, nor contracts in a legal sense.

b) Scripting model: Unlike the smart contract model, the scripting model enables codifying a desired logic using only a usage-oriented and predefined set of rules defined by the protocol, which limits the possible scenarios to implement. The idea behind this limitation is to avoid security problems and reduce the complexity of the system. Typically, scripting model is implemented in the DLTs that focuses on securing the manipulation of built-in assets rather than providing a platform for running universal programs

2) Turing completeness: Generally speaking, a given environment or programming language is said to be Turing-complete if it is computationally equivalent to a Turing machine [33]. That is, a Turing-complete smart contract language or environment is capable of performing any possible calculation using a finite amount of resources. Some DLTs are capable of supporting a Turing-complete execution environment, which provides its users with the flexibility to define complex smart contracts, Whereas other DLTs provides Non-Turing complete execution environments, because they suffer from some inherent limitations (For example, the impossibility to have an iteration structure with an arbitrarily high upper bound).

3) Determinism: Determinism is an essential characteristic of the execution environment in DLT systems. Since the distributed program (e.g. smart contract) is executed across multiple nodes, the deterministic behaviour is needed to yield

coherent and identical outputs to obviate discrepancies in the network. In order to ensure determinism, DLTs have to handle non-deterministic operations (e.g. Floating-point arithmetic, or random number generation, etc.) either by disabling these features or by enabling them in a controlled environment.

4) Runtime openness: In most DLTs, the execution environment or runtime is by design an isolated component without connections with external networks (e.g. Internet). However, in many case scenarios, the need for accessing information (e.g. weather forecast, stock price, or exchange rate) from outside the DLT, manifested as a necessity. Thus, to allow such a feature, different design choices were introduced. At this level we distinguish between three approaches:

- Isolated: where interactions between the smart contract execution environment and the external environments are not allowed
- Oracle-based: where interactions with external environments are managed by members of the network who are called oracles. An oracle refers to a third-party or a decentralized data feed service that provides external data to the network.
- Open: The execution layer is able to connect to the external environments.

5) Interoperability: DLT operating networks are currently by-design siloed and isolated from each other. The interoperability, which we consider as the ability to exchange data, assets or transactions between different DLTs, is a complex operation that requires passing transactions between them, in a trustless manner without the intervention of third parties. Interoperability is a highly desired property as it improves performance, allows cross value transfer and represents an indispensable direction of scaling. Due to its importance, multiple solutions were developed to enable the interoperability between different existing DLTs. These solutions can be categorized in the following groups:

- Sidechain [34]: is a blockchain running in parallel with another chain (known as main chain) that allows transferring data (cryptocurrency) from the main chain to itself. Sidechains operate generally in two modes; one-way pegged or two-way peg way. Such that in the former data can be moved from and back to the main blockchain by using locking mechanisms, and in the later data is moved only toward the sidechain.
- Multichain [35]: is a network of interconnect chains, upon which other chains can be built. In a multichain one major ledger rules all the sub-ledgers.
- Interoperability protocols: represent protocols and means (e.g. smart contracts) added to the original DLT to enable interoperability with other DLTs.
- Interoperable DLT: represents a DLT designed with the goal to enable interoperability between other DLTS.

B. Execution layer: state of the art

In this section, we provide an overview of the most widely-used execution environments implemented in the industry and the literature with a discussion of their properties. We pay special attention to the Ethereum virtual machine as it is being adopted by a large number of the existing DLTs. Broadly speaking, we can consider the current DLT-based smart contract platforms divided into two main groups: EVM-compatible platform and non EVM-compatible platform.

1) Execution environments:

a) Ethereum Virtual machine: In Ethereum, smart contracts represent a computer program written in a high-level language (e.g. Solidity, LLL, Viper, Bamboo, etc.) [36] and compiled into a low-level machine bytecode using an Ethereum compiler. This bytecode is stored in a dedicated account —therefore has an address— in the blockchain. Then, it is loaded and run reliably in a stack-based virtual machine called Ethereum Virtual Machine (EVM for short), by each validating node when it is invoked. The interactions with smart contract clauses or functions happen through transactions carrying inputs and the designation of the called function. Then, the corresponding code is executed, simultaneously, in the Ethereum virtual machines of the network, according to the invoking transactions payload. If the execution terminates correctly, the contract states are updated on the blockchain (State tree).

To enable the execution of the bytecode and state update, the EVM operates as a stack-based virtual machine. It uses a 256-bit register stack from which the most recent 16 items can be accessed or manipulated at once. The stack has a maximum size of 1024 possible entries of 256-bits words. The EVM has a volatile memory operating as a word-addressed byte array, where each byte is assigned its own memory address. The EVM has also a persistent storage space which is a word-addressable word array. The EVM storage is a key-value mapping of 2^{256} slots of 32 bytes each. Unlike the memory which is volatile, storage is non-volatile and it is maintained as part of the system state. The EVM is a sandboxed runtime and a completely isolated environment. That is, every smart contract running inside the EVM has no access to the network, file system, or other processes running on the computer hosting the EVM. The EVM is a security-oriented virtual machine, designed to permit the execution of unsafe code. Thus, to prevent Denial-of-Service (DoS) attack, EVM adopts the gas system, whereby every computation of a program must be paid for upfront in a dedicated unit called gas as defined by the protocol. If the provided amount of gas does not cover the cost of execution, the transaction fails. Assuming given enough memory and gas, the EVM can be considered a Turing-complete machine as it enables to perform all sorts of calculations.

b) Bitcoin scripting: Bitcoin uses a simple stack-based machine to execute Bitcoin scripts. A Bitcoin script is written using a basic Forth-like language. It consists of a sequence of instructions (opcodes), loaded into a stack and executed sequentially. The script is run from left-to-right using a push-pop stack. A script is valid if the top stack item is true (non-zero) at the end of its execution. Bitcoin defines two main scripts to validate the transaction and transfer the coins,

namely ScriptPubKey and ScriptSig. ScriptPubKey is a locking script placed on the output of a Bitcoin transaction that requires certain conditions to be met in order for a recipient to redeem the output. Conversely, ScriptSig is the unlocking script that satisfies the conditions placed on the output by the ScriptPubKey.

c) Stellar: Stellar does not provide a smart contract language nor a built-in virtual machine to execute general-purpose code. A Stellar Smart Contract (SSC) is expressed as compositions of transactions that are executed under various defined constraints. A participant in a Stellar smart contract, is not directly interacting with code on chain, but instead agreeing to the conditions of a transaction. Transactions are constructed out of a predefined set of 13 operations, where each operation is an individual command that mutates the ledger. On top of these operations, various constraints are defined to the transaction - Stellar has support for built-in Multisignature, Batching, Atomicity, Sequence ,Time bounds, and more [37]. SSCs are not Turing complete and can be written in multiple languages(Python, C#, Ruby, Scala, C++) using an SDK.

d) NXT and Ardor: NXT provides a suit of built-in smart contracts templates or smart transactions [38] in order to eliminate code insecurity. Its evolution, Ardor [39], adopts Java virtual machine (JVM) as a smart contract runtime. Besides it introduces the Lightweight Contracts as a framework for developing a layer of automation on top of the existing Ardor APIs. Concretely, the lightweight contract is a Java class uploaded to the blockchain and executed by a subset of nodes selected to run the ContractRunner addon. Unlike NXT, Ardor's contracts are considered to be Turing-complete.

e) NEO virtual machine: NEO proposes the lightweight NeoVM (NEO Virtual Machine) , a virtual stack-based machine for processing smart contracts. The NeoVM provides a general-purpose solution where, the NEO's compiler (Neo-Compiler), compiles contract source code written in Java, C# or other high-level languages (with limitations into a unified bytecode achieving thus cross-platform programming. Notably, NeoVM provides a stack isolation mechanism which allows each contract to only access its own stack area, thereby securing execution of smart contracts. NeoVM is Turing-complete by design and also relies on a gas concept identical to Ethereum's gas concept.

f) EOS virtual machine: The EOS VM is based on the WASM (WebAssembly)-LLVM (Low Level Virtual Machine) architecture. EOS adopts C++ as the contract's development language and compiles them using EOS's tool-chain into a low-level bytecode, which is executed by block producers. Therefore, the smart contracts are stored in the blockchain in the form of pre-compiled Web Assembly and acts as an application registered in the EOS blockchain and runs on the EOS nodes.

g) Cardano CCL: Cardano's smart contracts platform, or Cardano Computation Layer (CCL), was designed to address security and correctness concerns in Ethereum. The CCL consists of two layers: a language framework and a formally specified virtual machine. Also, CCL provides two virtual machines; IELE [40] a register-based machine, like LLVM, empowered with formal K semantics; and KEVM, a complete formal K semantics of the Ethereum Virtual Machine. The CCL supports solidity and a new general purpose smart contract language, inspired from Hashkall, called Plutus [41]. Similarly, to the EVM, KVM and IELE use gas to limit resource usage and prevent DoS attacks.

h) Zilliqa virtual machine: Zilliqa [42]delivers a smart contract platform based on a programming language called Scilia , a safety-oriented and intermediate-level language for writing formally verified smart contracts. Since Scilia is an intermediate-level language, it can serve as a compilation target for high-level languages (such as Solidity) and also as an independent programming framework.

i) Java virtual machine: Hyperledger Fabric embraces a different approach by adopting existing runtimes and languages instead of building new ones. In fact, Hyperledger Fabric adopts Java Virtual Machine (JVM for short) and Node.js runtime as smart contract environments. Therefore, smart contracts (Known as Chaincodes) are written in JavaScript, TypeScript, Java, Go or any other language that can run on one of the supported runtimes. adopts Kotlin and Java as the main language for smart contracts.

j) Stratis CLR: Stratis utilizes the popular Microsoft .NET Common Language Runtime (CLR) as an execution environment for its smart contracts rather than using a virtual machine. Its smart contracts are coded using C# and compiled into Common Intermediate Language (CIL) [43]. Hence, Stratis supports Common Intermediate Language (CIL), theoretically any language that can be translated into CIL can be used for writing smart contracts. Stratis also relies on the "gas" concept for payable execution which is identical to Ethereum's gas concept.

2) Interoperability: The inability for siloed DLTs to communicate with one another has been a major hindrance to the development of the blockchain space, therefore different proposals have aimed to solve this problem. In this subsection we present the most important approaches implemented at the execution layer to solve this problem.

a) Sidechains: Multiple sidechains have been proposed in the DLT ecosystem. Rootstock [44] is a sidechain of Bitcoin, equipped with RVM a built-in compatible Ethereum virtual machine. Rootstock chain is connected to the Bitcoin (BTC) blockchain via a two-way peg enabling transfers from BTC to SBTC (Rootstock's built-in currency) and vice versa using Bitcoin scripts, whereby users lock up their BTC in a special address and get an equivalent amount of RBTC on the sidechain. Similarly, Counterparty is another sidechain of Bitcoin where coins to be transferred are burned or locked by sending them to an unspendable address and generating the equivalent in the Counterparty chain. Drivechain is another proposal for transferring BTC between Bitcoin blockchain and sidechains. Unlike most DLTs where the sidechain is a separate project, Cardano has introduced Cardano KMZ sidechain as part of its ecosystem. Cardano KMZ is a protocol

which serves for moving assets from its two-layer CSL to the CCL (Cardano Computation Layer), or other blockchains that support the Cardano KMZ protocol. Another sidechain-based project is Plasma . It aims for creating hierarchical trees of sidechains (or child blockchains) using a combination of smart contracts running on the root chain (Ethereum). The idea is to build connected and interoperable chains operated by individuals or a group of validators rather than by the entire underlying network. Thus, Plasma helps scaling Ethereum by moving transactions toward the sidechains.

b) Interoperability protocols: Interledger (ITL) [45]is a protocol standardized by the World Wide Web Consortium for sending payments across different ledgers. Interledger is constructed of a network of untrusted connectors linking different ledgers and leveraging escrow transactions (conditional locks of funds) to make transfers between accounts on different ledgers. Moreover, Atomic swap enables trading digital assets across unrelated blockchains. Atomic swaps leverage Hashed Time-Lock Contracts (HTLC) to allow coordinating operations (e.g. trading digital assets), on different chains. These operations will have the same trigger —usually the revelation of the preimage of a particular hash. Alternatively, Hyperledger project proposes Hyperledger labs blockchain integration framework [46], a communication model to enable permissioned blockchain ecosystems to exchange any on-chain data independent of the platform (e.g. Hyperledger Fabric, Quorum, etc.) without a middleman.

c) Multi-chains: Polkadot [47] is a network of interconnected chains constructed of a central connector called Relay chain to which are connected multiple ledgers called 'Parachains'. The relay-chain is responsible for finalizing all the transactions, completing cross-chain transactions [48] and sharing states. Besides, in order to connect the Relay chain with external chains (e.g. Ethereum or Bitcoin), Polkadot provides bridge Parachains [49] that enable two-way compatibility. Similarly, to Polkadot, COSMOS [50] is constructed of "Zones" which are networks of blockchains interconnected via a central hub called Cosmos Hub Network. Each zone maintains its own states, and operates with its own validators. Alongside, all zones can exchange messages and tokens via the cosmos hub using a protocol called Inter-Blockchain Communication (IBC) [51].

d) Interoperable chains: Gravity Hub, is a blockchain with the inherent ability to communicate with other blockchains, such as Waves[52], or Ethereum. Gravity Hub nodes are able, for example, to obtain block headers from the Ethereum network and send them to Waves Platform to prove that a specific transaction has taken place on Ethereum. Another DLT with built-in interoperability is Wanchain [53]. Wanchain is a cross-chain blockchain infrastructure designed to provide interoperability between Bitcoin, Ethereum and ERC-20 tokens, EOS. Recently, the project introduced the T-Bridge framework [53], a generalized framework for data and asset transfer between heterogeneous public and private blockchains. More details about the interoperability solutions discussed in this subsection along other projects (such Blocknet, ARK, the POA network, AIO, etc.) can be found in this survey [54].

3) Determinism : To deal with the non-determinism issue, three general approaches are adopted. The first approach is to guarantee determinism by design. For instance, in Ethereum the EVM does not support, by-design, any non-deterministic operations (e.g. floating point, randomness, etc.). Nevertheless, due to the importance of randomness, the RANDAO [55] project has been proposed as an RNG (Random number generator) of Ethereum based on an economically secure coin toss protocol. The idea behind is to build DAO (decentralized autonomous organisation) for registering random data on the blockchain. The second approach, adopted by other projects such as Multichain [35], Corda, or Stratis which use existing runtime environments, ensure determinism by adapting these environments to force determinism processing.

C. Environment openness

Most DLTs rely on oracles to read data from external sources. Simply put, an oracle is a smart contract maintained by an operator that is able to interact with the outside world. Several data feeds are deployed today for smart contract systems such as Ethereum. Examples include , Town Crier [56] and oracle Oraclize.it [57].

VI. Application layer

In this section we briefly introduce the components and properties our DCEA framework defines for the application layer as well as an overview of the state-of-the-art.

A. components and properties

a) Integrability: As a new technology, which is often perceived as hard to adopt, DLT systems try to offer a better user-experience by providing necessary tools (APIs, frameworks, protocols.) to enable better integrability with existing technologies and systems (e.g. Web, mobile). The integrability of a DLT can be considered as a qualitative property, thus it is possible to deduce a "Level of Integrability ". That is, we establish a small integrability scale from "high" to "low":

- High: Indicates that the DLT ecosystem has a good integrability with other technologies, especially web and programming technologies.
- Low: Indicates that the DLT ecosystem does not provide any official integrability tools or provides few with limited ability.
- Medium: Defines an intermediate level between the two extremes.

b) DApp orientation and DLT's purpose : Decentralized applications (or DApps for short) are software applications whose server and client tiers are decentralized and operating autonomously, with no controlling entity. Broadly speaking, DApps are two-tier applications which include a front-end client connected to a back-end running on the blockchain. The key-advantage of DApps is the distribution of its execution and potentially the hosting of front-end components, a design feature that improves reliability and security. Therefore, we

can assume that a DApp is unstoppable and free from external control or manipulation. We do not consider an application as DApp, if it uses the DLT as a mere database to store data or to exploit its timestamping while its components live outside the DLT. Due to the importance of DApps, we consider a DLT as DApp-oriented if it focuses on offering the necessary tools for building and maintaining decentralized applications, using different protocols and APIs.

c) Wallet and identity management: Wallets are an important component of the application layer. Generally, they manage user's cryptographic identities. Since in most DLTs, identities and ownership are determined by their public/private pairs, a wallet represents a high-level entry-point to the network. Wallets are responsible for all cryptographic operations related to the creation or storage of the user's keys or digital certificates as well as the management of transactions.

B. Application layer: state of the art

Due to the vastness of different approaches and tools provided by different DLTs at the application layer, we overview only the application layer of few notorious DLTs.

a) Integrability: DLTs generally introduce a layer of integration between external entities and their data and execution layer. DLTs like Ethereum NEO or EOS and others, have a richer toolset and integration tools. Ethereum offers a robust and lightweight JSON-RPC API with a good support for the JavaScript language. It provides Web3.js, an official feature-rich JavaScript library for interacting with Ethereum compatible nodes over JSON-RPC. Further, for a better integration into legacy systems, Camel-web3j [58] connector provides an easy way to use the capabilities offered by web3j from Apache Camel DSL. In addition, Infura [59] provides online access for external actors to communicate with the Ethereum chain, through Metamask [60], dropping the need for running an Ethereum node or client making the DApp easier for the end-user. Similarly, EOS presents a wide set of tools and features, easing its integration and interaction with external systems.

b) DApp Orientation and DLT purpose: Bitcoin and similar projects (e.g. Zcash, Litecoin) are created with the purpose to serve as mere secure digital cash networks. Thus, they are considered Cryptocurrency-oriented. Other DLTs try to propose along the cryptocurrency other types of P2P value transfers. In the case of storage oriented DLTs such as Sia Network, Storj, FileIo, Ipfs, the network manages data storage alongside a cryptocurrency. Similarly, the service-oriented DLTs propose services consuming the inherent token, such as "Steemit" which runs a social network or Namecoin which aims to provide a decentralized DNS. On the other hand, various DLTs are DApp-oriented and allow developers to build generic applications.

VII. Evaluation and discussion

In this section, we present our comparative analysis and evaluation of the selected DLTs based on the properties defined by our framework.

Table II summarizes the comparison between a large representative sample of DLTs implemented in the industry or introduced by the recent research literature. The comparison covers four aspects: four-component DCEA framework composition, operational scope, decentralization level, and the higher taxon to which it belongs. In this subsection we focus, among the evaluated properties on analyzing the governance, conflict resolution approaches and evaluate decentralization as these properties play a major role to decide whether a system is a blockchain or not.

Decentralization is an essential characteristic that should be at the core of the design of DLTs. In our evaluation, when we investigate the decentralized nature of a DTL, we investigate its architectural topology, the cost of running full nodes and their distribution, alongside the governance scheme adopted in the decision-making process. Thus, we consider a DLT as decentralized if it is not controlled physically or logically by a single entity in terms of the aforementioned considerations. The 44 DLTs, presented in Table II, were reviewed on a three-step scale from centralized, semi-decentralized —where some processes are centralized or some entities in the network have substantial decisional power.— to decentralized. Our comparison shows that multiple DLT projects sacrifice full decentralization in favor of improving performance and scalability. For instance, Ripple, Stellar and Libra choose to accord operational and decisional privileges to central entities to achieve better performance. For better scalability and security, Ripple adopts a "starter" membership list of trusted nodes, which can be updated by users. However, because divergent lists invalidate safety guarantees [31], users rarely update their initial list, leading to a centralized system dominated by the nodes of the starter list. Besides, the Ripple company holds a big amount of XRP (Ripple's token) and thus it is able to influence the network. Similarly, [61] shows that the entire stellar system can fail completely in sequence if only the two nodes operated by the Stellar foundation are deleted. On the other hand, most PoS-based DLTs are decentralized. However, PoS is criticized for favoring entities with a bigger amount of tokens, which lead in case of unfair distribution of tokens to centralized validation. [62] shows that the ratio between the block reward and the total network stake has a significant impact on the decentralization of the network. IOTA is an example of semi-decentralized DLTs as it makes use of coordinator (COO) nodes run by the IOTA foundation to secure the network from 34% attacks and transactions cannot be confirmed unless they are confirmed by the Coordinator (via milestones). DPoS-based DLTs are criticized for being prone to validation centralization because elected validators can collude between them instead of being in competition. For example, in EOS, there is a correlation between votes [63] for different candidates and it is possible to deduce the existence of alliances within the EOS system. Besides, the high concentration of tokens on EOS is worrisome as the top 100 holders collectively own 75.13% [63] of the total of EOS currency. Therefore, the selection of validators can be in the hand of a small fraction of the network. In relation, researchers

TABLE II

A SUMMARY TABLE OF COMPARISON AND ANALYSIS OF THE SELECTED DLTS

	DATA LAYER				CONSENSUS LAYER					EXECUTION LAYER						APPLICATION LAYER		
DLT solution	Data Structure	Data shareability	States management	Immutability	Consensus	Governance	Governance nature	Transaction orderer	Fork management (Non-malicious forks)	Turing completeness	Environment openness	Execution environment	Determinism	Languages	Interoperable	Orientation	Integrability	Purpose
Æternity	CoB	G	On	S	GHOST, Bitcoin-NG (for security) and PoS (for governance)	De	Bl	Randomly selected miner	Lc	Tc	Op (Built-in oracle)	Æternity VM	Dc	Sophia, Varna, solidity	No	Cy and DA	H	Ge
Algorand	CoB	G	On	S	Algorand	Ol	Ex	Randomly selected leader (stake weighted election)	Nf	NTc	Is (OB)	Algorand VM	Dc	TEAL	No	Cy and DA	M	Ge
Ardor	CoB (One parent chain with multiple child chains)	G	On	S	PoS	Ol	Bl	Forging account (NXT forging algorithm)	Lc	NTc	Op	Deterministic Java VM	Dc	Java	Yes (between child chains)	Cy and DA	M	Ge
Bigchaindb 2.0	HDS (a database and a CoB)	G	On/Off	W	Tendermint	Ol or De	Ex	Elected leader orders transactions by arrival time	Nf	Nsp					No	DA	M	Bo
Bitcoin	CoB	G	On	S	PoW	An	Bl and Ex	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Bitcoin scripting, Miniscript	No	Cy	M	Po
BitShares	CoB	G	On	S	DPoS	De	Bl	Elected Witness	Lc	NTc	Is	Bitshares runtime	Dc	C++	No	Cy	L	So
Byzcoin[a]	Skipchain	G	On	S	Byzcoin	An	Ex	Randomly selected miner	Lc	NTc	Is	Byzcoin runtime	Dc	Go	No	Cy	L	Ge
Cardano	CoB	G&R	On	S	Ouroboros	De	Bl	Randomly selected leader	Lc	Tc	Is	IELE VM	Dc	IELE and Plutus	Yes (with cardano sidechain)	Cy and DA	M	Ge
Corda (R3)	DDB	G	Off	W	RAFT, BFT-SMaRt or KAFKA	Di, Ol or De	Ex	BFT-SMaRt distributed notary	Nf	Tc	Is (OB)	Java VM	NDc	Kotlin, Java	No	DA	H	Bo
Cosmos	CoB (Hub and multiple zones)	G&R	On	S	Tendermint	De (with Veto)	Bl	Elected block producer	Nf	Tc	Is	WebAssembly VM	Dc	Wasm languages(cosmos SDK)	Yes (Cross-chain Interoperability)	Cy and DA	M	Ge
Decred	CoB	G	On	S	PoW and PoS	De	Bl	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Modified bitcoin scripting	No	Cy	L	Po
Elrond	SCoB (Metachain and shards)	G	On	S	Secure PoS (variant of Algorand)	De	Bl	Randomly selected proposer	Lc	Tc	Is (OB)	Elrond VM	Dc	IELE, Wasm languages	Yes (between Metachain and shards)	Cy and DA	L	Ge
EOS	CoB	G	Off	W	Transactions-as-PoS	Ol	Bl	Elected block producer	Lc	Tc	Is (OB)	WebAssembly VM	Dc	Wasm languages	Yes (with EOSYS sidechain)	Cy and DA	H	Ge
Ethereum 1.0	CoB	G	On	S	PoW (ETHASH)	An	Ex	Randomly elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	Solidity, Vyper, LLL, Julia	No	Cy and DA	H	Ge
Ethereum Enterprise	CoB	G&R	On	W	PoA, RAFT or IBFT	Di, Ol or De	Ex	Elected leader	Lc	Tc	Is	Ethereum VM	Dc	EVM languages	No	DA	H	Bo
Exonum Enterprise	CoB	G & R	On	S	Exonum protocol	Di, Ol or De	Bl	Predefined leader	Nf	Tc	Op (Built-in oracle)	Java VM and Rust runtime	NDc	Java, Rust	Yes (one way with Bitcoin)	DA	M	Bo
Elastos	CoB (main chain and sidechains)	G	On	S	DPoS and POW (merged mining with Bitcoin)[b]	De	Bl	Randomly elected miner	Lc	Tc	Is	CAR runtime, EVM (Ethereum Sidechain) , NEOVM(NEO sidechain)	Dc	C++, Java, Swift, JavaScript, Golang, solidity	Yes (Sidechains can transact with each other)	Cy and DA	M	Ge
Filecoin	CoB	G	On	S	Proofs-of-Spacetime	Ol	Bl	Elected leader (using Expected Consensus (EC))	Lc/ Hc[c]	Tc	Is	FilecoinVM	Dc	Golang	No[d]	DA	H	DSo
Hashgraph	DAG (Transaction-based DAG)	G&R	On	W	Hashgraph	Ol	Bl	Fair ordering via Consensus Time Stamping	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Hyperledger Fabric)	Cy and DA	M	Ge
Hyperledger fabric	HDS	G&R	Off	W	PBFT	Ol or De	Ex	Ordering service node	Nf	Tc	Op	Java VM and Nodejs runtime	Dc	Go, JavaScript	No	DA	H	Bo
IOTA	DAG (Transaction-based DAG)	G	On	W	IOTA	Ol	Ex	End-user and coordinator node	Hb	Nsp (Qubic a smart contract protocol is under development)[e]					Yes (with Hyperledger Fabric)	Cy	M	IOT
Lisk	CoB (Main and sidechains)	G	On	S	DPoS	De	Bl	Elected leader	Lc	Tc	Is	NodeJs runtime	Dc	JavaScript	No	Cy and DA	H	Ge
Multichain	CoB	G&R	On/Off	W	Mutlichain protocol (variant of PBF)	Ol	Ex	Predefined leader	Lc	Multichain do not support smart contracts					No	DA	L	Ge
NEO	CoB	G&R	On	S	DBFT (Delegated Byzantine Fault Tolerance)	Ol, De	Bl and Ex	Elected leader	Nf	Tc	Is (OB)	NeoVM	Dc	.NET and JVM languages (Java, Kotlin)	Yes (Between private blockchains connected to NEO)	Cy and DA	M	Ge
Omniledger[f]	SCoB	G	On/Off	S	ByzCoinX (Variant of Byzcoin)	Ns	Ns	Randomly elected leader	Nf	Not supported					Yes (with its shards)	-	-	Ge
Parity substrate	CoB	G&R	On	W	Pluggable consensus (Hybrid PBFT, Aurand, Rhododendron, Shaft, ouroboros, PoW)	Ol	Bl	Depends on the chosen consensus mechanism	Nf	Tc	Is	WebAssembly VM	Dc	WASM languages	Yes (with Polkadot)	DA	H	Bo
Polkadot (Relay chain)[g]	CoB (relay chain)[g]	G	On	S	GRANDPA and BABE (PoS)	De	Bl	Randomly elected leader	Lc	Tc	Is	WebAssembly VM[h]	Dc	WASM languages	Yes (Cross-chains interoperability)	Cy and DA	H	Io
Quorum	CoB	G&R	On	W	IBFT or RAFT or Clique POA	Di, Ol or De	Ex	Elected leader	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	DA	H	Ge
Qutum	CoB	G	On	S	PoS	De	Bl	Elected leader	Lc	Tc	Is (OB)	Ethereum VM and X86 VM	Dc	EVM languages, C, C++, Rust, Python	Yes (Atomic swap with bitcoin)	Cy and DA	Hign	Ge
Ripple	DDB (chain of ledgers stored as key-value)	G	On	S	Ripple (FBA)	Ol	Ex	Validating nodes converge toward a canonical order	Nf	NTc	Is	Built-in specialized payment types	Dc	JavaScript	No	Cy	M	Po
Rootstock	CoB	G	On	S	POW (merged mining with Bitcoin)	Ol	Bl	Randomly selected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with bitcoin)	Cy and DA	L	Ge
Steem	CoB	G	On	S	DPoS	De	Bl	Elected leader	Lc	Not supported						Cy	L	So
Stellar	DDB (chain of ledgers stored as key-value)	G	On	S	Stellar (FBA)	Ol	Ex	Validating nodes (using transactions sequence number)	Nf	NTc	Is	Stellar runtime	Dc	Java, JavaScript, Go	Yes (Atomic swap with other blockchains)	Cy	Hign	Po
Sia	CoB	G	On	S	PoW	An	Ex	Randomly elected miner	Lc	Not supported					No	DA	M	DSo
Stratis	CoB (Main chain and sidechains)	G	On	S	PoA or PoS	Ol	Ex	Randomly elected miner	Lc	Tc	Is	.NER runtime	Dc	.Net languages (e.g C#)	Yes (with its sidechains and with bitcoin)	Cy and DA	M	Ge
Nano	DAG (block-lattice)	G	On	S	Open Representative Voting (ORV) (based on DPoS)	Ol	Bl	Users (Sender and recipient)	Nf[i]	Not supported					No	Cy	L	Ge
Tezos	CoB	G	On	S	DPoS (Liquid PoS) and Emmy	De	Bl	Elected miner	Lc	Tc	Is (OB)	Tezos interpreter	Dc	Michelson	No	Cy and DA	M	Ge
Wanchain	CoB	G&R	On	S	PoS	De	Bl	Elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (Cross-Chain Communication Protocol)	Cy and DA	L	Io
Waves	CoB	G	On	S	Waves-NG (PoS based on Bitcoin-NG)	An	Bl	Elected miner (PoS)	Lc	NTc	Is (OB)	Waves runtime	Dc	Rideon	Yes (Atomic swap with other blockchains)	Cy and DA	L	Ge
Zilliqa	SCoB	G	On	S	PBFT and POW (Ethash)	De	Bl	Elected leader	Nf	Tc	Is (OB)	Zilliqa VM	Dc	Scilla	No	Cy and DA	M	Ge
Libra (Facebook)	DDB	G	On	W	LibraBFT	Ol	Ex	Elected leader	Nf	Tc	Is	Libra VM	Dc	Move	No	Cy	M	Po
Artis	CoB	G	On	S	HoneyBadgerBFT	Ol	Bl	Correct nodes	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Ethereum)	DA	M	Ge
VeChain	CoB	G	On	S	PoA	Ol	Bl	Elected leader (deterministic pseudo-random process)	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	Cy and DA	M	Ge
Red Belly	CoB	G	On	S	DBFT Democratic BFT	Ns	Ns	Elected proposers	Nf	NTc	Is	Script runtime	Dc	Bitcoin-like scripting	No	Cy	L	Ge

LEGEND :
CoB: Chain of blocks, SCoB: Sharded Chain of blocks, DDB: Distributed database, HDS: Hybrid data structure
G: Global, S: Strong, R: Restricted, W: Weak, Cy: Cryptocurrency, DA: DApps, Ol: Oligarchic, De: Democratic, An: Anarchic, Di: Dictatorship
Bl: Built-in, Ex: External, Ns: Not specified, Nsp: Not supported, Nf: No forks, Lc: Longest chain, Hb: Heaviest branch, Hc: Heaviest chain,
Op: Open, Is: Isolated, OB: Oracle-based, Dc: Deterministic, NDc: Non-Deterministic, H: High, M: Medium, L: Low, Ge: General.
Bo: Business-oriented, Po: Payment-oriented, So: Service-oriented, DSo: Decentralized storage-oriented, IOT: IOT-oriented, Io: Interoperability-oriented

[a] Based on the implementation available on https://github.com/dedis/cothority/tree/master/byzcoin
[b] Elastos sidechains can have any consensus mechanism
[c] Filecoin gives weight to blocks that offer more storage power
[d] Filecoin ensures interoperability between different implementations of Filecoin protocol.
[e] https://qubic.iota.org
[f] Based on the minimalistic implementation available on https://github.com/dedis/student_18_byzcoin
[g] Parachains can have their own data structure
[h] Parachains are individual chains with their own runtime logic.
[i] In Nano, each user maintains its own DAG and a balance-weighted voting system is used to handle conflicting transactions.

such as Micali argue that inappropriate incentive systems may lead to significant centralization [64]. Also, Kwon and al. discuss why it is hard to achieve good decentralization [65] in permissionless blockchains.

In our evaluation, we were also interested in assessing decentralized governance of the selected DLTs. In order to modify the parameters of a DLT protocol and upgrade the network rules, multiple decision-making approaches are defined by different projects ensuring different political forms of governance. Bitcoin and similar networks (e.g. Ethereum) are anarchically governed. In such systems, a proposer initially presents an improvement proposal to solve a problem or to enhance the protocol performance (e.g. rising block size). Afterwards, the proposal is publicly discussed and once it attracts enough favorable peer review, the proposal is implemented into the project's codebase. Depending on the technical requirements a proposal may require a Soft-fork or a Hard-fork implementation, such that in the former the implementation is forward-compatible and does not need to be implemented by all the nodes in the network, whereas the later requires the nodes to imperatively upgrade their software otherwise they will be isolated (intentionally or not) in a different network as upgraded nodes will disconnect and ban nodes which are adopting a different version of the protocol. In Bitcoin, Soft-forks implementations have an on-chain governance feature, called Version bits voting [66] to measure miner support and accept or not the implementation. Other projects like Qutum, try to avoid the problem of hard-fork for minor modifications (e.g. the block size and gas parameters) by providing built-in mechanisms enabling the actors to decide democratically how to tune the system using dedicated smart contracts (Decentralized governance protocol). Similarly, Tezos adopts Tezos governance protocol which enables stakeholders and node operators to democratically decide on upgrading the system's rules. For enterprise-grade DLTs such as Hyperledger, Corda which are intended to be primarily used in private or consortium contexts, the situation is different. In these DLTs, governance is externally managed by a governing body formed by members of the network. For instance, for upgrading the DLT version a technical committee takes the decision, stops the network [67] and applies the upgrade.

Another interesting aspect in the selected DLT systems is their approaches for ordering transactions and resolving conflicts. In the systems based on the Nakamoto consensus protocols, the transactions are ordered freely by the miners in blocks, validated then included in the canonical chain. In presence of conflicting chains, nodes shift toward the longest chain. Other protocols adopt Nakamoto's longest chain rule due to its resilience and reliability. For example, in DPoS protocols, the longest chain rule applies when a block is not voted on by the majority (*2/3 +1*) of the block producers. also some PoS protocols (e.g Ouroboros) and Nevertheless, a recent study [68] argues that the "longest-chain PoS" rule is not safe when applied to PoS protocols. Conversely, Nano's DAG abandons leader election and delegates transaction ordering to users and their representatives to resolve conflicts. In other networks such as Fabric, a special ordering service called the Ordering Service Nodes (OSN) establishes a total order on the transactions before committing the blocks into the ledger. The OSN (or, simply, orderers) is ensured by a set of nodes that collectively establish, using RAFT or KAFKA, the total order of all transactions to be committed by the other nodes, and thus resolve conflicting transactions. Similarly, in Corda, the notary service [69] orders transactions and detects conflicts using multiple consensus mechanisms such as RAFT, PBFT and others. Once the notary service approves a transaction, it is considered as final and committed by all the parties. In IOTA, the ordering is partially ensured by the transaction senders and transactions weights. In fact, the transaction sender randomly selects two previous tips to which his transaction will be appended. The network continuously assigns a cumulative weight to each transaction. Due to this model of cumulative weight, the valid tangle is determined by the heavier branch. Moreover, coordinators in IOTA validate (mostly indirectly) periodically the valid transactions in the tangle by issuing milestone transactions, serving, among other things, to indicate transaction's order and the main tangle to the nodes. Hashgraph aims to guarantee "ordering fairness" based on its gossip-about-gossip protocol, ensuring that transactions are processed in the order of their reception by the nodes. Due to the importance of an accurate and fair ordering, Avi Asayag and al. proposed Helix [70], and Kelkar and al. proposed Aequitas [71] as new blockchain-based consensus protocols providing fair ordering.

Over the last few years, the application of blockchain and blockchain-like technologies has spread to many fields. Although their applications areas overlap, we can definitely deduce that they serve different business scenarios with contrasting requirements. Broadly speaking, we can state that Blockchain systems tend to better serve global decentralized B2C or C2C models whereas blockchain-like platforms are better suited to B2B models or for internal usage within a single organization. Blockchain-like systems are mostly adopted in the corporate sector where there is a need for controlled governance and restriction over shared data. For example, a consortium of banks can build a blockchain-like system where financial transaction details are only shared with the concerned parties. On the other hand, blockchain systems are more suitable for egalitarian networks with a high level of freedom for the end user. For example, most blockchains manage cryptocurrencies and allow global money transfer without involving a middleman.

VIII. CONCLUSION & FUTURE RESEARCH DIRECTIONS

In this paper, we have provided a comprehensive survey of the current DLTs with a multi-layered state of the art. Moreover, we have used the proposed framework as a reference to investigate the different approaches adopted by different DLTs at the four layers: data structure, execution, consensus and application layers. Additionally, we conducted

a qualitative and comparative analysis of a large number of existing DLTs. In conducting this survey, we came across a plethora of challenges that we believe had a direct effect on DLT's performance, of which we summarise, at the four layers, the ones most important to us in the following.

a) Blockchain Bloat: While surveying multiple blockchain projects, we identified a common problem known as blockchain bloat. This problem refers to the difficulty of managing the storage of a growing size ledger without harming decentralization. Hence, the cost of this maintenance grows with the size of the blockchain, there will be a level where only few entities will be able to bear the cost of managing the whole ledger.

b) Smart contract security: It is clear that smart contract Security is a real problem in DLT space, particularly for public blockchain where financial risks are at stake. Many solutions have been proposed to further aid in securing smart contract, by providing code analyzers and secure smart contracts libraries.

c) Data immutability in private or consortium DLTs: In permissioned private DLTs it is difficult to guarantee a strong level of immutability as is ensured by public blockchains. Given that immutability is a cornerstone in DLTs, this problem is questioning the usage of DLTs in such environments.

d) Decentralized governance mechanisms: Governance in decentralized networks remains a complex problem. In the absence of any form of authority, the decision-making process can be very challenging. DAO (Decentralized Autonomous Organization) was a primitive attempt to build an organization with decentralized management and heightened transparency, but the experiments showed its limitations including the lack of a decentralized reputation system for DAO, of Sibyl-resistant identities, and of a regulatory framework for DAO actions, etc.

e) Forkless runtime upgrades: Hard forking is a standard method of upgrading public blockchains. However, in large scale networks, this process remains inefficient, and error-prone due to the levels of offline and online coordination required to avoid splitting the network. Some recent solutions[47] are exploring the possibility to deploy the upgrade using a portable technology (e.g. WASM) on-chain to manage nodes upgrade without external intervention.

REFERENCES

[1] S. Nakamoto, "Bitcoin : A Peer-to-Peer Electronic Cash System," pp. 1–9, 2008.

[2] L. Team, "Litecoin." [Online]. Available: https://litecoin.org/

[3] S. King and S. Nadal, "Peercoin–secure & sustainable cryptocoin," *Aug-2012 [Online]. Available: https://peercoin. net/whitepaper ()*, 2012.

[4] S. Popov, "The Tangle," Tech. Rep., 2017.

[5] H. Foundation, "Hyperlegder Project." [Online]. Available: https://www.hyperledger.org/

[6] Hedra, "Hedera Hashgraph." [Online]. Available: https://www.hedera.com/

[7] E. N. Dawson, A. Taylor, and Y. Chen, "ISO/TC 307 Blockchain and distributed ledger technologies." [Online]. Available: https://www.iso.org/committee/6266604.html

[8] ISO/TR, "ISO/TR 23455:2019 Blockchain and distributed ledger technologies — Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems."

[9] I. S. Association, "IEEE blockchain standards." [Online]. Available: https://blockchain.ieee.org/standards

[10] ITU, "Focus Group on Application of Distributed Ledger Technology." [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/dlt/Pages/default.aspx

[11] W. Pugh, "Skip lists: a probabilistic alternative to balanced trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[12] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the Bitcoin UTXO set," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10958 LNCS. Springer Verlag, 2019, pp. 78–91.

[13] I. Eyal, A. Gencer, and E. Sirer, "Bitcoin-ng: A scalable blockchain protocol," *13th USENIX Symposium*, 2016. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf

[14] P. Jovanovic, "ByzCoin: Securely Scaling Blockchains," *Hacking, Distributed, August*, 2016.

[15] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford, "{CHAINIAC}: Proactive software-update transparency via collectively signed skipchains and verified builds," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1271–1287.

[16] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11368 LNCS. Springer Verlag, 2019, pp. 327–351.

[17] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, 2016.

[18] C. LeMahieu, "RaiBlocks: A feeless distributed cryptocurrency network," *URL https://raiblocks. net/media/RaiBlocks_Whitepaper__English. pdf*, 2017.

[19] M. Hearn, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016.

[20] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[21] I. Grigg, "Eos-an introduction," *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.

[22] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, 2016.

[23] S. Brakeville and P. Bhargav, "Blockchain basics: Glossary and use cases," 2016. [Online]. Available: https://developer.ibm.com/technologies/blockchain/tutorials/cl-blockchain-basics-glossary-bluemix-trs/

[24] L. Lamport, "Proving the Correctness of Multiprocess Programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, 1977.

[25] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10211 LNCS. Springer Verlag, 2017, pp. 643–673.

[26] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 4 1988.

[27] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming*, 1982. [Online]. Available: http://dl.acm.org/citation.cfm?id=357176

[28] L. Baird, "THE SWIRLDS HASHGRAPH CONSENSUS ALGORITHM: FAIR, FAST, BYZANTINE FAULT TOLERANCE," Tech. Rep., 2016.

[29] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, 2014.

[30] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, vol. 32, 2015.

[31] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture*

Notes in Bioinformatics), vol. 9229. Springer Verlag, 2015, pp. 163–180.
[32] Stellar, "Intuitive Stellar Consensus Protocol - Developers Blog." [Online]. Available: https://www.stellar.org/developers-blog/intuitive-stellar-consensus-protocol
[33] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.
[34] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," pp. 1–25, 2014. [Online]. Available: http://www.blockstream.com/sidechains.pdf://www.bitcoin.fr/public/divers/docs/sidechains.pdf
[35] G. Greenspan, "Multichain private blockchain-white paper," *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.
[36] S-tikhomirov, "GitHub - s-tikhomirov/smart-contract-languages: A curated collection of resources on smart contract programming languages." [Online]. Available: https://github.com/s-tikhomirov/smart-contract-languages
[37] Stellar, "Stellar Smart Contracts — Stellar Developers." [Online]. Available: https://www.stellar.org/developers/guides/walkthroughs/stellar-smart-contracts.html
[38] Nxter, "IGNIS — NXTER.ORG." [Online]. Available: https://www.nxter.org/understanding-ignis/#smarttransactions
[39] jelurida, "ARDOR Whitepaper." [Online]. Available: https://www.jelurida.com/sites/default/files/JeluridaWhitepaper.pdf
[40] T. Kasampalis, D. Guth, B. Moore, T. F. Șerbănuță, Y. Zhang, D. Filaretti, V. Șerbănuță, R. Johnson, and G. Roşu, "IELE: A rigorously designed language and tool ecosystem for the blockchain," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11800 LNCS. Springer, 10 2019, pp. 593–610.
[41] Plutus, "GitHub - input-output-hk/plutus: The Plutus language implementation and tools." [Online]. Available: https://github.com/input-output-hk/plutus/
[42] Z. Team, "The ZILLIQA technical whitepaper," *Retrieved September*, vol. 16, p. 2019, 2017.
[43] startis academy, "Welcome to Stratis Academy — Stratis Academy documentation." [Online]. Available: https://academy.stratisplatform.com/
[44] N. Mining, "Rootstock (RSK): Smart contracts on Bitcoin. Medium," 2018.
[45] S. Thomas and E. Schwartz, "A protocol for interledger payments," *URL https://interledger. org/interledger. pdf*, 2015.
[46] Catus, "GitHub - hyperledger/cactus: Hyperledger Cactus is a new approach to the blockchain interoperability problem." [Online]. Available: https://github.com/hyperledger/cactus
[47] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, 2016.
[48] Polkadot, "Validator · Polkadot Wiki." [Online]. Available: https://wiki.polkadot.network/docs/en/maintain-validator
[49] Parachain, "Bridges · Polkadot Wiki." [Online]. Available: https://wiki.polkadot.network/docs/en/learn-bridges
[50] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," *URL https://cosmos. network/whitepaper*, 2016.
[51] IBC, "ics/ibc at master · cosmos/ics · GitHub." [Online]. Available: https://github.com/cosmos/ics/tree/master/ibc
[52] Waves, "Open platform for Web 3.0 applications." [Online]. Available: https://wavesprotocol.org/
[53] Wanchain, "Wanchain 4.0 T-Bridge Framework Tech Explainer: Part 1 — General Overview - Wanchain." [Online]. Available: shorturl.at/sZ145
[54] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger Approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.
[55] Randaow, "GitHub - randao/randao: RANDAO: A DAO working as RNG of Ethereum." [Online]. Available: https://github.com/randao/randao
[56] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An Authenticated Data Feed for Smart Contracts," *dl.acm.org*, vol. 24-28-Octo, pp. 270–282, 10 2016. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978326
[57] Provable, "Provable - blockchain oracle service, enabling data-rich smart contracts." [Online]. Available: https://provable.xyz/
[58] Camel-web3j, "camel/web3j-component.adoc at master · apache/camel · GitHub." [Online]. Available: https://github.com/apache/camel/blob/master/components/camel-web3j/src/main/docs/web3j-component.adoc
[59] Infura, "Ethereum API — IPFS API Gateway — ETH Nodes as a Service — Infura." [Online]. Available: https://infura.io/
[60] Metamask, "MetaMask." [Online]. Available: https://metamask.io/
[61] M. Kim, Y. Kwon, and Y. Kim, "Is Stellar as secure as you think?" in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 377–385.
[62] C. Nguyen, D. Hoang, D. Nguyen, D. N. I. ..., and u. 2019, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *ieeexplore.ieee.org*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8746079/
[63] E. Voting, "EOS Voting Pattern Analysis." [Online]. Available: https://eosauthority.com/producers_relation?TB_iframe=true&width=1367.1&height=678.6
[64] Coindesk, "Debate 2017. No Incentive? Algorand Blockchain Sparks Debate at Cryptography Event." [Online]. Available: https://www.google.com/amp/s/www.coindesk.com/
[65] Y. Kwon, J. Liu, M. Kim, D. Song, and Y. Kim, "Impossibility of full decentralization in permissionless blockchains," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 110–123.
[66] Bips, "bips/bip-0009.mediawiki at master · bitcoin/bips · GitHub." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki
[67] H. Fabric, "Procedure for Upgrading from v1.0.x — hyperledger-fabricdocs master documentation." [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/v1.1.0-alpha/upgrade_to_one_point_one.html
[68] J. Brown-Cohen, A. Narayanan, and S. M. Weinberg, "Formal Barriers to Longest-Chain Proof-of-Stake Protocols *," *dl.acm.org*, pp. 459–473, 6 2019. [Online]. Available: https://doi.org/10.1145/3328526.3329567
[69] Corda, "Notaries — Corda OS 4.4 — Corda Documentation." [Online]. Available: https://docs.corda.net/docs/corda-os/4.4/key-concepts-notaries.html
[70] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira, "A fair consensus protocol for transaction ordering," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 55–65.
[71] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-Fairness for Byzantine Consensus," Tech. Rep., 2020.