Literature Review - Tangle Based Distributed Ledgers

Valerio Di Stefano 1898728 and Giuseppe Prisco 1895709

21/03/2023

1 Introduction

The following is a review of the papers:

- S Popov. The tangle - White paper, 2018
  http://cryptoverze.s3.us-east-2.amazonaws.com/wp-content/uploads/2018/11/10012054/IOTA-MIOTA-Whitepaper.pdf
- Sebastian Müller, Andreas Penzkofer, Nikita Polyanskii, Jonas Theis, William Sanders, Hans Moog: Tangle 2.0 Leaderless Nakamoto Consensus on the Heaviest DAG. IEEE Access 10: 105807-105842 (2022)
  https://ieeexplore.ieee.org/document/9907014

We will focus on the analysis of investigated problems and proposed solutions of the aforementioned papers, while also highlighting their differences and similarities.

We conclude the review by also providing an introduction to a novel technology for Tangle based DLTs: the "Coordicide", the main concept behind the IOTA 2.0 protocol. An in-depth explanation of the aforementioned protocol is provided in the working paper:

- S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer, O. Saa, W. Sanders, L. Vigneri, W. Welz, and V. Attias: "The coordicide" (2019)
  https://files.iota.org/papers/20200120_Coordicide_WP.pdf.

We also note that, in the following review, we will refer to the paper "The Tangle" as the "first paper", and to the paper "Tangle 2.0 Leaderless Nakamoto Consensus on the Heaviest DAG" as the "second paper".

1.1 Distributed Ledgers

A distributed ledger technology (DLT) is a shared, replicated and synchronized data structure containing information distributed across many sites, which, in contrast to a central database, does not require a central administrator (which represents a single point of failure), but instead relies on consensus mechanisms to ensure agreement and validity of data.

A distributed ledger usually requires a peer-to-peer (P2P) computer network and an appropriate consensus algorithm so that the ledger is reliably replicated across every node of the network.
Each node typically replicates and stores an identical copy of the ledger data which is updated independently of other nodes.
Nodes issue new transactions on their local copy of the ledger independently, and then collectively all working nodes use the designated consensus algorithm to determine the correct copy of the updated ledger and update their local one.
Security is enforced through cryptographic keys and signatures.

Distributed ledgers can be categorized in terms of:

- Data structures used to store data (linear data structure, such as a blockchain, or more complex data structures, like DAG-based distributed ledgers);
- Consensus algorithms used to ensure ensure agreement and validity of shared data ("Proof of X" approaches, typically used for blockchain based distributed ledgers, or DAG consensus-building and voting algorithms);
- Permissions on data access (distributed ledgers can either be permissioned, i.e. private, or permissionless, i.e. public)

1.2 Blockchain

The most common form of distributed ledger technology is the blockchain.
A blockchain-based distributed ledger consists of lists of records (called "blocks") that are securely linked together via cryptographic hashes of previously created blocks (all of which store a timestamp along with transaction data).
This "chain" structure makes blockchain transactions irreversible (once recorded, data in any given block cannot be altered retroactively without altering all subsequent blocks).
Blockchain records themselves, however, are not unalterable, since blockchain forks are possible (i.e. the existence of two or more branches originating from the blockchain).

A blockchain distributed ledger can either be on a public network (permissionless blockchain) or a private network (permissioned blockchain).

1.2.1 Permissionless blockchain

In a permissionless, and therefore public blockchain, no access restrictions are defined for data stored on the ledger, instead, nodes of a peer-to-peer (P2P) computer network manages blocks of the chain, collectively adhering to a consensus protocol which allows nodes to add and validate new transaction blocks.
Since new blocks can be created by every node of the network, there is the need for a consensus mechanism that allows nodes to agree on which block will be the next one to be attached to the chain: in order to reach consensus, public blockchain-based distributed ledgers require the election of a leader, which will be the designated proposer of the next block, through a specific consensus mechanisms.

This type of blockchain-based ledger results in a distributed computing system with high Byzantine fault tolerance and high security by design.

Two of the most common consensus mechanism algorithms for blockchains are the Proof of Work (PoW) and the Proof of Stake (PoS) protocols.

A "Proof of Work" (PoW) based consensus protocol consists of proving to other parties that a certain amount of computational effort has been expended.
When used in a blockchain-based permissionless distributed ledger system, this mechanism forces each node willing to propose a block (also called "miner") to compete for the possibility to append its own block to the chain ("mining" of blocks), with a success probability proportional to the amount of computational effort expended.

The "Proof of Work" mechanism is used as a protection measure for deterring manipulation of data, for which large energy and hardware-control requirements are established.

Blockchain-based systems used to record transactions for cryptocurrencies (such as Bitcoin or Ethereum 1.0) typically offer rewards to successful "miners" of blocks, in the form of percentage fees on issued transactions to compensate for the resources spent during the "mining" process of new blocks.

A "Proof of Stake" (PoS) based consensus protocol, used in blockchain-based cryptocurrency systems, consists of selecting validators of blocks in proportion to their "stake".
The "stake" of a validator can be equal to the quantity of holdings in the associated cryptocurrency, or to the product of the number of tokens held with the amount of time that a single user has held them (called "coin age"), or to other possible amounts based on the implementation of the protocol.

The "Proof of Stake" mechanism is used as a protection measure against malicious users or groups from taking over a majority of validation: this is accomplished by requiring potential attackers to acquire a large fraction of the tokens on the blockchain to successfully perform an attack.

Other less common "Proof of X" consensus protocols exist, like "Proof of Capacity" (which allows for sharing of the memory space of the nodes), "Proof of Activity" (a hybrid mechanism that implements both aspects of PoW and PoS), "Proof of Burn" (requires the nodes that issue transactions to send a small amount of tokens to inaccessible wallet addresses).

1.2.2 Permissioned blockchain

In a permissioned, and therefore private blockchain, access to stored data is restricted to a defined group of users: an access control layer is used to govern who has access to the network, and new nodes cannot join the system unless invited by the network administrators.
This results in closed, less decentralized systems as compared to public blockchain systems, but with a higher level of control over nodes creating and attaching blocks to the blockchain.

1.2.3 Advantages and Disadvantages of blockchain based systems

Blockchain-based distributed ledger systems have several advantages, and among the main and most obvious ones we have:

- Decentralization, with which the blockchain eliminates some risks that come with data being held centrally (as for centralized databases), such as the existence of a single (central) point-of-failure and the requirement for a central administration controlling the system;
- Immutability, with which data tampering is prevented and that allows for a high level of confidence that the well-formed blocks appended to the blockchain will not be revoked in the future (are "finalized") and thus can be trusted;
- Transparency, which allows for easy verification of data recorded in the blockchain by any node of the network, and increases public trust in the network itself;
- Traceability, which allows for tracing of changes on the network by inspecting past blocks of the irreversible audit of blocks.

Along with advantages, blockchain-based distributed ledger systems also have a number of disadvantages which can't be overlooked.
Three main drawbacks of the blockchain-based technology can be identified:

- Transaction Fees: fees are needed to incentivize the creation of new blocks (and thus the validation of transactions needed for the system to work properly), and, especially in "Proof of Work" based blockchains, these fees can be very high to compensate for resources spent for the mining of new blocks (hence fees discourage micropayments, often needed for IoT systems)
- Heterogeneous System: two different roles for nodes in the network must be defined, namely "transaction issuers" and "transaction validators", which lead to conflicts that must be solved, often by spending resources;
- Scalability issues (caused by a low transaction throughput): in order to issue a transaction, a larger amount of operations, often sequential (not parallelizable), than traditional database systems need to be performed (cryptographic signature and verification, consensus mechanism operations, propagation of new data over the entire network).

Another crucial disadvantage, undermining both blockchain-based cryptocurrency's adoption and a wide-spread usage of the blockchain technology across industries, is the so called "blockchain trilemma" (term coined by "Vitalik Buterin", founder of the "Ethereum" blockchain).
The "blockchain trilemma" refers to the difficulty of finding a balance between security, decentralization, and scalability of a blockchain based distributed ledger system: scalability and decentralization are often held back by security, but security tends to be compromised by any shifts on a network that offer scalability.
The blockchain trilemma can be seen as an evolution of the well known "CAP theorem", which proves the difficulty for a distributed shared data-system to simultaneously guarantee consistency, availability, and partition tolerance.

Current blockchain-based systems usually focus on just two out of three elements (the "Bitcoin" blockchain, for example, is currently secure and decentralized, but not well scalable) or on finding a solution able to provide the perfect balance of all the elements ("sharding", "side-chains" and "state channels" are experimental solutions proposed to address the trilemma).

2 The Tangle

The drawbacks and general defects of the blockchain-based DLTs led to the need for alternative technologies capable of solving the main problems arising from the said data structure while maintaining the desired discussed features of a distributed ledger.
A solution proposed in 2015 by Serguei Popov in its related white paper relies on the concept of "The Tangle", which is the main mathematical concept behind IOTA, a cryptocurrency for the Internet-of-Things (IoT) industry aimed to ease machine-to-machine micropayments.
"The Tangle" is the name given to the data structure proposed in the white paper: a directed acyclic graph (DAG), i.e. a graph structure with directed edges and no directed cycles, in which transactions are stored and which is proposed as the next evolutionary step of the linear blockchain data structure.

In a "tangle-based" cryptocurrency, the DAG structure replaces the global blockchain structure, shared by all nodes of the system, as the ledger in which transactions are stored. Transactions issued by nodes of the networks represent the "sites" of the DAG (i.e. vertices of the graph).
In order for a new transaction to be issued, it must "approve" two other already existing transactions stored as sites on the graph: the approval of a transaction A by a transaction B is represented as a directed edge from site B to site A in the Tangle DAG (we say that B "directly approves" A).
A transaction A is also said to be "indirectly approved" by a transaction B if there exists a directed path of length at least two from transaction B to transaction A.
A "genesis site" of the Tangle must also exist, which is defined as the only site of the graph that does not approve any transaction and is directly or indirectly approved by every other site: the genesis transaction contains all the tokens that can be managed by the nodes of the network and no new token is allowed to be generated ("mined" or "minted").

With this "tangle-based" approach, in order to issue new transactions, nodes must "work" to approve other transactions (after checking whether the transactions to approve are valid or if they present conflicts with the Tangle's transactions history), and thus will contribute to the network's security.

Tangle-based ledger's networks are usually asynchronous, and nodes can see different sets of transactions at different times in their local view of the ledger: despite this, nodes do not need to reach consensus on which transaction can be issued and which node can attach its transaction to the Tangle structure (as it is instead required in blockchain-based distributed ledgers), and therefore each node is allowed to issue transactions (valid in respect to the past history of the ledger) at any time, which allows for a high degree of parallelisation (with no need for total ordering of transactions).

A consensus mechanism is needed, however, because of possible "conflicting transactions", i.e. two or more transactions attached to the ledger DAG which are in conflict with the past history of the ledger, like "double spending" transactions.
In case of two or more conflicting transactions, only one of them will continue to be approved (directly or indirectly) by upcoming transactions, while the other ones will be "orphaned" (won't be approved by incoming transactions in the future).
To solve possible conflicts of transactions of the ledger, a "tip selection" algorithm can be defined: this algorithm allows nodes to choose which "tip" (i.e. a transaction not yet approved by two other attached transactions) to attach newly issued transactions to.
Because no rule is imposed on which "tip selection" algorithm must be used by nodes of the network, a "reference" algorithm must be defined such that it is reasonable to assume that the majority of nodes of the network will use it as their designated algorithm to select attachment sites.

Other than a consensus mechanism to solve conflicts of the ledger, a mechanism to incentivize nodes to participate in the approval of new transactions is needed.
This mechanism solves problems in which nodes might not participate in the approval of transactions when they don't need to issue their own transactions ("lazy nodes") or they might prefer to approve old and already validated transactions instead of new ones which represent "tips": these situations lead to a slow down of newly issued transactions' confirmation, and therefore to an increasingly growing number of "tips" of the network.

2.1 First paper's specifications

In order to provide a solution for both consensus finding on conflict resolution and the slow down of tips' confirmation in the network, the first analyzed paper introduces the concept of "weight" of a transaction.
The "weight" of a transaction is a value associated to the transaction itself (a positive integer, usually a power of 3 in the original IOTA implementation), which is proportional to the amount of work that the issuing node invested into it, and represents the "importance" given by nodes of the network to the related transaction.
The "cumulative weight" of a transaction can also be defined, calculated as the sum of the own weight of a particular transaction and the own weights of all transactions that directly or indirectly approve the transaction itself.

The paper also introduces other concepts related to transactions and their weights:

- Height: the length of the longest oriented path from the transaction to the genesis;
- Depth: the length of the longest reverse-oriented path from the transaction to some tip;
- Score: the sum of the own weight of the transaction and the own weights of all other transactions approved by it.

After discussing some performance metrics of Tangle-based DLTs systems, related to the concept of "weight" (transactions issuance rate and time to confirmation, analyzed later in this review), the first paper introduces some possible attacks scenario on the system and proposes solutions to mitigate these types of attacks, in the form of "tip

selection" algorithms which allow to choose new transactions' attachment sites on the Tangle DAG structure.
The proposed algorithms are in the family of "Markov Chain Monte Carlo" algorithms, which select "tips" based on a certain degree of randomness with probability proportional to the cumulative weight of transactions in the network (the proposed algorithms are further analyzed later in this review).

The proposed solutions make it possible for nodes of the network to find consensus on the "heaviest sub-graph" of the Tangle DAG structure (i.e. the set of connected sites of the Tangle with the highest cumulative weight), and are assumed to be adopted by the majority of nodes of the network (because, as explained in the paper, there is no incentive for neither malicious nodes nor regular nodes to deviate for the "reference" algorithms).
This "heaviest sub-graph" concept, which is an alternative to the "longest chain" rule of blockchain-based DLTs systems, can be considered equivalent to the "largest sub-DAG" (i.e. the set of connected sites of the Tangle containing the highest number of transactions), because of the assumption, made in the first paper, which assumes the own weight of transactions to be capped to a maximum amount to avoid some kind of attacks (namely the "parasite chain" attack).

The first paper also proposes a solution for assuring that nodes participate in the approval of new transactions, and therefore to solve the problem of "lazy nodes" and nodes attaching issued transactions to already confirmed transactions.
The proposed solution requires nodes of the network to keep track of the amount of transactions issued by their neighbors, which allows them to drop the "inactive" neighbors ("lazy nodes"), while assures that nodes will attach newly issued transactions to recently attached tips thanks to the proposed MCMC "tip selection" algorithm (which selects tips based on the cumulative weight of their directly or indirectly approved transactions, and therefore makes the approval of a tip arbitrarily attached to an "old" transaction unlikely because of the cumulative weight of the latter).

2.2 Second paper's specifications

The second paper we analyze expands on the concept of "The Tangle" introduced in its related "white paper" (the aforementioned "first paper").
It provides a generalization of tangle-based systems and their related distributed ledger DAG structure discussed in the first paper, while also introducing a consensus protocol that works efficiently and fast in an asynchronous model and allows for a high degree of parallelisation.

The Tangle-based approach proposed in the second paper uses an "Unspent Transaction Output" model (UTXO) for balance keeping, in which each transaction of the ledger corresponds to a set of inputs and a set of outputs (the total value of outputs must be equal to the total value of all inputs).
The UTXO model is used as an alternative to the "account-based" balance keeping model (used by Ethereum), in which funds are directly associated with an account of a user.

The paper points out two major problems of the solutions presented in the original proposal of the first paper:

1. Vulnerability to various types of attacks and too much reliance on the Proof of Work needed to issue new transactions (and used to mitigate some of the attacks);

2. Liveness problem, consisting of the impossibility to add honest transactions to the current ledger state when they refer to transactions that turned out to be malicious.

2.2.1 First problem's solution

The first major problem is solved by proposing a new consensus mechanism, the main contribution of the second paper: a probabilistic asynchronous leaderless protocol for consensus which employs a weight-based voting scheme on the Tangle and can provide synchronization of nodes at certain time intervals using a random common coin (the protocol and underlying algorithms are discussed later in this review).

2.2.2 Second problem's solution

The second major problem of the original proposal, the liveness problem, is instead solved by separating transactions from their containers (blocks, sites), and also separating the DAG structure of the system into two distinct graphs used for different purposes.

The separation of the concepts of "sites" and "transactions" of the DAG structure is achieved by introducing "blocks" that are allowed to contain more than a single transaction: these blocks become the new vertices of the corresponding DAG structure.

It also separates the concept of "ledger" and "Tangle" by introducing a "Ledger DAG", i.e. a DAG structure in which all transactions are stored (containing vertices representing transactions and directed edges representing transactions inputs and outputs, from and to the incident vertex respectively), and a "Tangle DAG", i.e. a DAG structure in which issued blocks are stored, in a similar way to the original proposal of the first paper.

By combining the "Ledger DAG" and the "Tangle DAG", a "Voting DAG" can be constructed, which is used for determining voting cones, and that results in a graph whose vertex set is the union of the set of blocks of the "Voting DAG" and the set of transactions of the "Ledger DAG", and which retains the same edges of the two original graphs: this allows for a better representation of votes of each node to reach consensus, and to correctly execute the voting protocol presented in the paper (described in the next section of this review).

Separating Ledger and Tangle DAG also allows for retrieving missing blocks in the "past cone" of a block (i.e. the set of vertices to which a directed path from the current block exists), which can be lost during broadcast of messages in the network: this is called "solidification" process, and works thanks to the Tangle DAG with which nodes can identify missing parents of the block and ask for their retransmission to their peer nodes.

Finally, the separation of the two DAG structure serves as a solution to the liveness problem introduced by the original Tangle proposal of the first paper: because any

transaction is allowed to be issued and attached to nodes of the Tangle, honest transactions that directly or indirectly approve transactions that turn out to be malicious in the future can not be added to the ledger state.
By providing a "Ledger DAG" separate from the "Tangle DAG", and in which all transactions are stored, retrieving non-conflicting orphaned transactions becomes possible.

Similarly to the first paper, in the second paper blocks of the Tangle DAG are assumed to contain exactly one transaction for simplicity of analysis: it is then specified, however, that this constraint can be relaxed to allow more than one transaction per block.
It is worth noting, though, that the possibility of including a single transaction in each block, given by DAG based DLTs, is a relevant advantage in respect to blockchain based DLTs: the constraint which imposes to wait for a certain amount of transactions to fill each block before propagating it, in fact, results in a generally lower transaction throughput and can significantly slow down the network.

3 Algorithms and Protocols

This section of the review describes and analyzes the algorithms for consensus finding proposed by the first paper and also the protocols and respective algorithms proposed by the second paper.

3.1 First paper's proposed algorithms

The first analyzed paper provides mechanisms and algorithms to provide a solution for consensus finding on conflicting transactions waiting to be approved, in the form of "tip selection" algorithms, which allow to choose new transactions' attachment sites on the Tangle DAG structure, and therefore allow to solve possible conflicts generated by two or more tips waiting for an approval.

It is also noted in the paper that no enforcing of a tip selection algorithm is made for nodes of the network: this leads to nodes being able to choose attachment sites on the Tangle arbitrarily, and therefore malicious nodes can choose the most convenient tips to approve while honest nodes should voluntarily choose to follow the reference algorithm (such that nodes do not have reasons to abandon the reference algorithm in favor of a more advantageous one, and so that the majority of nodes can be assumed to follow the reference algorithm).

The first analyzed tip selection strategy consists of a purely random selection of two of the current "tips" of the Tangle as the attachment sites of a node's newly issued transaction.
This approach, as noted in the paper, is not practical, because it does not offer enough protection against "lazy" or malicious nodes: if a node was to approve a transaction issued by a node which is not contributing to the network security (i.e. is not approving new tips), or issued by a malicious node trying to double spend, the said transaction would have the same probability of being approved as any other "honest" transaction.
Despite being impractical, this simple mechanism is still described in the first paper and used to provide insights into the system's behavior for more complicated tip selection strategies because of its simplicity.

Another possibility for a tip selection strategy would be to randomly choose tips based only on their cumulative weight or score.
Although being better than the purely random strategy in presence of "lazy nodes", this approach is vulnerable against a large number of attacks described in the paper (parasite chain attack, splitting attack, ecc...), which result in attackers being able to obtain large weights for their newly issued transactions, and therefore get their malicious transactions approved.

Better algorithms than the purely random tip selection and the weight/score based tip selection are proposed in the paper, all being in the family of Markov Chain Monte Carlo algorithms (MCMC): these algorithms use random variables to construct sequence of possible events (Markov Chains) in which the probability of each event depends only on the state attained in the previous event.
In particular, the MCMC algorithms proposed in the paper select new transactions' attachment sites on the Tangle at random with a probability proportional to the sites' cumulative weight and the cumulative weight of all their directly or indirectly approved transactions.

The MCMC algorithms proposed in the paper are based on a "Random Walk" approach, in which N independent "particles", called "walkers", are placed on a previously issued transactions' sites in the Tangle, and then move towards the tips of the Tangle by executing "transitions" from transaction x to transaction y (a transition from x to y is possible if and only if y approves x).
Transitions of walkers of the MCMC algorithm are executed according to a certain "transition probability" function which defines the transaction's site towards which the particle will transition to.

The paper notes that the N starting sites of the N independent "walkers" should be chosen at random from a set of sites representing transactions issued on a time in the interval [W, 2W], where previous time W should be "large": this is to avoid cases in which walkers are already "too close" to tips when they start their "random walk".

A tip selection strategy can also be defined by the different mechanisms with which the MCMC algorithm is used to select the tips:

- A possible way of choosing tips with the MCMC algorithm would be to select the first two tips reached by two particles among the N.
- A variation of the previous method would be to discard the tips reached "too fast" by walkers (i.e. reached by walkers in a number of steps lower than a defined value). This would help ignoring tips representing transactions' sites issued by "lazy nodes" and therefore arbitrarily attached to already approved "old" transactions' sites in the Tangle.
- Another approach described in the paper would be to first run a "random walk" to choose a "model tip" such that at each step the particle transition towards the transaction with the higher cumulative weight, and then run a second "random walk" with the defined "transition probability" function to choose the two designated tips while verifying that all the sites' transactions visited are consistent with the "model tip".

This approach is described in the paper as an "additional protection measure" against "parasite chain" attacks (described later in this review) .

- The final approach presented in the paper proposes to run a "random walk" many times to then choose as attachment sites tips reached by walkers more frequently.

Transition probability rules can be defined in different ways for MCMC with Random Walk algorithms, and the paper proposes four different approaches:

1. A transition probability proportional to $e^{-\alpha(H_x - H_y)}$, with H_x and H_y are the cumulative weight of the current site x and the destination site y respectively, and $\alpha > 0$ is a parameter to be chosen.

   This rule assigns to each of the attached sites of the current transaction x a probability that depends on their cumulative weight, so that transactions with a higher cumulative weight are more likely to be chosen for the "transition".

   This mechanism provides protection against "Sybil attacks" (i.e. attacks in which attackers are able to gain a high level of influence over the rest network) and other similar attacks (described later in this review) because it is assumed that single attackers would not be able to produce sites with a cumulative weight larger than the rest of the network (and because walkers move towards tips to choose based on their approved sites' cumulative weights).

   Thanks to this mechanism, node's "laziness" is discouraged too: newly issued transactions' sites attached to already approved transactions by "lazy" nodes, which are therefore not contributing to the network security, are less likely to be chosen as tips by the "walkers" of the MCMC algorithm because of their low cumulative weights.

2. A transition probability defined in the same way of the previous approach, but with a non zero probability of a possible "backtracking" for the walker, which results in the particle moving back to a random transaction's site x approved by the current transaction's site.

   This variant of the previous approach helps against attacks which exploits the knowledge of the "exit probability distribution" of particles (i.e. the probabilities of each tip to be chosen by the tip selection algorithm): by making particles being able to backtrack, additional randomness can be introduced in the process of selecting a tip, which renders attackers unable to calculate the said "exit probability distribution".

3. A transition probability proportional to H_x - H_y as in the first approach, but with a sharper decrease when two transactions y and z attached to the current transaction x have a cumulative weight that is almost equal (i.e. $H_z \; approx \; H_y$), so that the transition probability for one of the transactions z or y is a lot greater than the transaction probability of the other (y or z respectively).

   This way of defining the transition probability rule helps mitigating the so called "splitting attacks" (analyzed later in this review), in which attackers try to maintain two parallel branches of the DAG in order to spend the same funds on both branches as long as two halves of the network contribute to each branch equally.

4. A transition probability proportional to both H_x - H_y and H_x, so that when the walker is "deep in the tangle" (i.e. is far from reaching a tip) the next choice for a transition will be almost "deterministic" because of the high value of the weight H_x of the current site x, while the randomness of the algorithm will increase when the walker is close to a tip, as the cumulative weight H_x will in this case be smaller.

   This final approach helps avoiding the selection of tips attached to "weakest branches" (i.e. with a low cumulative weight) and therefore mitigating some types of attacks which rely on the construction of a "local sub-DAG" which will then be attached to the actual DAG structure (e.g. "splitting attacks").

MCMC tip selection algorithms are "local", which means that nodes do not need to consider the cumulative weight of the entire Tangle to select a tip (only cumulative weights of sites traversed by the "walkers" are needed).

3.2 Second paper's proposed algorithms

The second paper proposes a consensus protocol described as a "two layer solution" to the consensus problem in presence of conflicting transactions.
The first layer is an "asynchronous layer", which represents a general solution for an asynchronous network setting in order to guarantee eventual consistency for the shared Ledger structure.
The second layer is a "synchronized layer", which allows the synchronization of nodes in certain time intervals in order to provide a solution for the consensus problem even in some edge cases (represented by "impossibility results" for the asynchronous setting, described later in this review) and to also provide liveness and safety guarantees about the protocol.

In the considered system, nodes of the network have two roles:

1. They receive and propagate blocks by and from their network neighbors;
2. They generate their own blocks (containing their own newly issued transactions) and propagate them to their neighbors, while also assigning these blocks a reference to previously generated blocks, which results in an implicit voting mechanism (dubbed "On Tangle Voting", discussed later in this review).

Each node of the network has a certain associated "weight", which is described in the paper as a function of "scarce resources": no enforced definition is given for the concept of "weight", but as suggested by the authors of the paper, it could be defined as a stake or reputation based function which applies to each node a weight to guarantee a certain level of "sybil protection".
The paper proposes two examples for the implementation of the weight function used by the protocol: a "resource testing" approach (nodes must prove the ownership of a certain amount of scarce resource, for example the underlying cryptocurrency's tokens) and a "delegation" approach (in which owners of certain amount of stake can delegate their weight to other nodes and revoke it when they don't behave as expected, thus establishing a "reputation based" mechanism).
The weight of each node is also assumed to be normalized (i.e. the sum of all weights of nodes in the network is exactly 1) to simplify the analysis of the protocol.

The concept of "weight" of each node is used by the proposed protocol to keep track of what portion of nodes referenced a previously issued transaction, contained in block x, with their own issued transactions: this is represented by the "Witness Weight" of block x (denoted by "WW(x)"), which is defined as the total weight of all nodes that issued one or more transactions contained in the "future cone" of block x (i.e. the set of blocks from which a directed path to block x exists in the Tangle DAG).
This provides a measure of the portion of blocks that received and considered valid a certain transaction contained in the corresponding block x, and therefore provides a level of "confidence" with which a certain transaction can be considered permanently accepted in the DAG structure shared by all nodes.
The concept of "Witness Weight" of blocks can therefore be used to define a certain "confirmation threshold" θ such that $\theta \in (0.5, 1]$, which allows blocks with Witness Weight greater than θ to be considered "confirmed" forever by nodes of the network: this in turn allows nodes to consider each transaction contained in these types of blocks as a valid and approved transaction, without need for future nodes to reference its container block.

Two transactions are said to be directly conflicting if they have at least one input in common.
Two transactions are therefore indirectly conflicting if they contain at least two conflicting transactions in their past cone.

In the Ledger DAG structure of the second paper's proposal, conflicting transactions are allowed to coexist.
The sub-DAG structure of the Ledger DAG with the maximum number of non-conflicting transactions is called a "Reality".

Because of the distributed and asynchronous nature of the system, each node may have a different perception of the shared Ledger DAG structure (its "local Ledger DAG"), and therefore a different perception of "Realities" of the Ledger: it follows that each node has to choose a single "preferred reality" from the set of realities of its local Ledger DAG (i.e. a conflict-free part of its local ledger) in which it can issue new transactions and validate existing transactions.
In order for nodes to choose their "preferred reality", the second paper proposes a "Reality Selection" algorithm which, starting from the set U of transactions of the node's local Ledger DAG, iteratively choose the transaction x with the highest "weight" w(x) (i.e. the output value of a specific "weight function", such as the "Approval Weight" function described later) and adds it to the "reality set" R, then "prunes" all transactions conflicting with x from U: this process continues until all transactions of the node's local ledger has been chosen or pruned, i.e. until U is empty.
The set R of chosen transactions at the end of the algorithm constitutes the "preferred reality" of the node.

The described algorithm therefore allows a single node of the network to choose its "preferred reality" among its own local Ledger DAG, while the mechanism with which the node can let other nodes of the network know about its own chosen preferred reality is the "On Tangle Voting" mechanism.

Other than the concept of "Ledger DAG" (which, as said before, is the DAG in which transactions are stored as vertices whose edges represent references to outputs of previously issued transactions), the concept of "Tangle DAG" is also described in the paper, a DAG containing single blocks as vertices, each, in turn, containing one or more transactions.
The edges of the "Tangle DAG" represent instead references to previously created and attached blocks, made by newly created blocks.
When in need to issue a transaction contained in a certain block, each node can attach this block to k other blocks contained in the Tangle DAG, by creating an edge representing either a "transaction reference" (i.e. a reference to a certain transaction contained in a certain block of the Tangle DAG) or a "block reference" (i.e. a reference to another block of the Tangle DAG).
Blocks and transactions that an upcoming block of the Tangle DAG references are determined by the output of a "Tip Selection algorithm", described later in this review. This process represents the aforementioned "On Tangle Voting" mechanism (OTV), which allows nodes of the network to "cast their votes" on the Tangle DAG by creating references to previously created blocks or transactions.

As seen before, a block of the Tangle DAG can be considered confirmed when its Witness Weight is large enough to indicate that a conspicuous percentage of nodes of the networks "witnessed" the creation and inclusion of this block in their local "Tangle DAG".
In order to obtain a similar mechanism with which transactions can be considered confirmed, the second paper introduces the concept of "Approval Weight" of a transaction x (denoted as "AW(x)"), which is defined as the sum of the weights of all nodes having issued a block on the Tangle DAG which currently maintains either a transaction reference to x or a block reference to the block containing x.
In other words, the Approval Weight of a transaction defines the percentage of nodes of the network approving a given transaction, while the concept of Witness Weight (described before) is instead related to blocks, and defines the percentage of nodes approving a given block: this distinction is necessary because the protocol allows blocks to contain more than one transaction, and also allows to have two or more blocks in the Tangle DAG containing the same transaction or the same set of transactions (to manage cases in which a fund owner may request several nodes to broadcast a certain transaction x, or a node may need to issue several different blocks containing x).

Analogously to the confirmation rule for blocks, the paper defines a confirmation rule for transactions, which uses a confirmation threshold θ such that $\theta \in (0.5, 1]$ and allows a transaction to be considered confirmed by a node when $\text{AW(x)} \geq \theta$.

Before proceeding with the description of the voting protocol described in the second paper, we need to introduce the concept of "conflict graph" of a node.
The "conflict graph" of node i is the graph whose vertices represent "conflicts" of the Ledger, and whose edges connect two vertices representing "conflicts" (i.e. transactions) which are directly conflicting (i.e. have one or more inputs in common, hence spend the same output of a previous transaction).
A "conflict" is defined as a transaction directly conflicting with another transaction of the ledger.

In order to choose a block or a transaction in its "preferred reality" onto which "casting its vote", a node uses a "Tip Selection Algorithm" proposed in the paper as a "Uniform Random Tip Selection Restricted on Reality R" (R-URTS).
As the name suggests, this algorithm selects tips from the preferred reality of a node uniformly at random, and works as follows:

1. Node i creates a block b containing transactions to be issued in the Ledger DAG;
2. Node i chooses a random tip t from its local Tangle DAG;
3. If the randomly chosen tip t is a block such that its past cone only contains transactions which are in turn all contained in node i's preferred reality R, a "block reference" from the new block b to the chosen block t is created and stored in node i's local Tangle DAG;
4. If instead the randomly chosen tip t is a block which contains a transaction that is in turn a tip in the Ledger DAG, and if for this tip t the only conflicts contained in its past cone are from node i's preferred reality R, then a "transaction reference" from the new block b to the chosen block t's transaction is created and stored in node i's local Tangle DAG;
5. If neither of the two above conditions apply, the block is discarded;

This process goes on until k different tips are selected (where $k \geq 2$ is defined a priori).

By selecting tips to reference with its newly issued blocks, a node can therefore vote for the transactions contained in its preferred reality and allow the other nodes to know about its vote: transactions can then eventually be confirmed by nodes using the "confirmation rule" described above.

As seen before, a consensus algorithm is needed to solve conflicts in the Tangle. Conflicts are represented by two or more conflicting transactions, and their resolution consists of deciding which one of the conflicting transactions will be considered "confirmed" by the network as a whole, thus being kept in the Ledger.
Conflicting transactions are allowed to exist in the Tangle and Ledger DAG, but their state cannot remain "unconfirmed" forever, as this would violate the liveness property of the system (nodes won't eventually agree on the confirmation state of transactions) and could also harm its safety (as conflicting transactions have a non-zero probability of being eventually confirmed by some node, and therefore two conflicting transactions have a small but still non-zero probability of being both eventually confirmed). Furthermore, keeping conflicting transactions in an "unconfirmed" state would also drastically increase the communication required on the voting layer and make it impossible to "prune" unnecessary branches of the Tangle and Ledger DAG.

The second paper discusses some types of attacks on the stability of the system (metastability attacks, discussed in depth later in this review), which exploit the property of the system to allow conflicting transactions to be included in the Ledger and Tangle DAG in order to attempt to keep honest nodes of the system in an undecided state for a long time.
These attacks are shown to be (at least theoretically) possible, and are therefore described as "impossibility results" under the asynchronous communication model, as

the described protocol, along with its underlying algorithms, doesn't guarantee liveness and safety in asynchronous settings in presence of the discussed attacks.
The OTV protocol instead guarantees eventual consistency under the assumptions of both random block issuance and random package delay: these results are summarized in Theorem 1 of the paper, discussed later in this review.

In order to provide liveness and safety guarantees, the paper proposes a modified version of the described protocol which is intended to be used when stronger assumptions about the synchronicity of the system can be made.
In particular, the paper offers a solution to the impossibility scenarios in which nodes cannot come to an agreement between various options, by presenting a mechanism that utilizes a "distributed random number generation" process (dRNG) to provide external randomness and synchronize the nodes, in order to interfere with a possible adversary. In this sense, this modified version of the protocol constitutes a "second layer" of the protocol itself, i.e. a "synchronized layer" of the OTV mechanism seen before (which can in turn be seen as a first "asynchronous layer").

In particular, the "synchronized" version of the protocol uses a new reality selection algorithm which allows a node to choose its preferred local reality, the "Reality Selection Algorithm With Common Coin".
As the name implies, the algorithm utilizes a common random "coin", i.e. a value X generated by the dRNG at a certain time such that $0.5 \leq X \leq \theta$, where θ is the confirmation threshold for transactions (discussed above), and such that this value is received by every node of the network with a non-zero probability before the next random value's generation.
The algorithm is divided in two phases:

1. First, each node starts from the set U of conflicts stored in its local set of conflicts and iteratively chooses the transaction x $\in$ U with the highest approval weight.

   If AW(x) > X, the algorithm adds x to the node's preferred reality transactions set R and prunes all transactions conflicting with x from U, then continues choosing the next transaction x with the highest approval weight for the next iteration.

   If instead AW(x) $\leq$ X (or when U is empty), the algorithm skips to its second phase.

2. In the second phase, the algorithm behaves in the same way as the reality selection algorithm for the "asynchronous" case described above, starting from the reality set R and the local conflict set U updated by the first phase, but choses, at each iteration, the transaction with the max value for "hash(x$\|$X)" where "x$\|$X" denotes the content of transaction x concatenated with the random value X.

This algorithm allows to obtain a transaction set which is indeed a reality, with a non-deterministic procedure that chooses transactions to include in this "preferred reality" set based also on the value of a randomly generated value X.

The new voting protocol for nodes in this "synchronized setting" is then described by the paper as follows: the node chooses its preferred reality using the "Reality Selection Algorithm With Common Coin" described above, then uses the R-URTS algorithm to choose the block and transactions references for the block to issue, and finally waits for the next random value of the dRNG to execute a new iteration of this voting protocol.

As explained in the paper, this mechanism guarantees, with a non-zero probability, to reach a so called "pre-consensus" state, i.e. a state from which, eventually, consensus will be reached for all nodes on a certain reality: this happens when the approval weight of all nodes of a chosen reality R is such that AW(R) $> \theta$ for each honest node of the network.

The results achieved by this "synchronized layer" of the OTV protocol are summarized in Theorem 2 of the paper, discussed later in this review.

4 Performance Analysis

4.1 Introduction to common metrics

The most important metrics to take into account for DLTs are the number of issued transactions per second and the confirmation time of each transaction.
In the first paper's proposed setting, nodes are modeled as independent entities by a Poisson process with rate λ.
The second paper models each node i of the network as an independent entity issuing blocks at a Poisson rate λ_i, with λ_i being proportional to the weight of the node w(i), that is $\lambda_i = \lambda \cdot w(i)$, for some $\lambda > 0$. Since each node's weight is normalized to 1, effectively representing a percentage on the total weight of the network, it follows that:

- $\sum \lambda_i = \lambda$

Therefore the issuing rate of transactions or blocks follows an exponential distribution with rate λ.

In this section of the review, we describe the analysis and results provided by the first paper about the aforementioned performance metrics: for the analogous concepts presented by the second paper, the notion of "blocks" can be used in place of the notion of "transactions" in the entirety of the following analysis.

4.2 Number of tips in the system

To compute the time to confirmation (TTC), both papers begin their analysis by first estimating the total number of tips in the system L(t) at any time t.
In order to properly calculate this value, the study starts with the assumption that the number of tips in the system remains roughly stationary in time.

If we denote with λ the arrival rate of new tips in the system and with h the average time needed for a node to issue a transaction, we have that:

- When a node issues a transaction at time t, it does not see the current state of the tangle but only that of h time units in the past, which means that transactions issued after t - h are not yet visible to the node;
- As a result, at any given time t there are on average λh hidden tips not yet visible to the network;
- Moreover, we consider the number of revealed tips r, which are tips attached to the tangle at a time before t - h and still not approved/referenced by any transaction;

- The resulting number of tips in the system is stationary and centered around the value $L_0 = \mathrm{r} + \lambda\mathrm{h}$

Under the assumption that the the number of tips is stationary, at any time t there is a number of transactions equal to λh that were tips at time t - h but are no longer tips at time t (since λh new transactions are issued).

Therefore each newly issued transaction will approve on average $\frac{2r}{(r+\lambda h)}$ tips in case it has to approve two existing tips.
In the general case in which a set of k blocks is referenced instead (i.e. for the second paper's proposal), the average number of tips referenced will be $\frac{kr}{(r+\lambda h)}$.
Since the number of tips is assumed to be constant, the average number of tips chosen should be equal to 1, resulting in the following:
$L_0 = \frac{k\lambda h}{(k-1)}$

This result was only estimated by the first paper and later experimentally confirmed by other works, and is now adopted in the analysis presented by the second paper.

4.3 Time to confirmation and regimes of load

To properly estimate the time to confirmation, both papers distinguish between two types of regimes: a low load regime and a high load regime.

In the case of a low load regime the overall number of tips in the system is relatively small.
This happens when the flow of transactions is small and, as a consequence, it is unlikely for different transactions to reference the same tip. Moreover, in case the network delay is small and the nodes have high computation speed h, a restricted number of tips would be present at any time regardless.

In the high load regime, there would be a large number of tips in the system, and, as a result, L_0 will be large. Due to network delays and/or limited computation speed, it is more likely that different transactions approve the same tips.

The time to wait for the first approval/reference of a specific transaction is different in the two regimes:

- In a low load regime, the tip pool size is small, thus one of the next incoming transactions will reference the transaction in question in an average time of $\frac{1}{\lambda}$;
- In a high load regime, the number of tips L_0 is large, and since at least h time units need to pass in order for a tip to be approved, the time of first approval will be $h + \frac{L_0}{2\lambda} = 2h$ (in particular, the second paper generalizes this time to be $h + \frac{L_0}{k\lambda} = h + \frac{h}{(k-1)}$)

Consequently, the time to confirmation (TTC) will be different in the two regimes.

After a transaction is approved several times in a low load regime, all the newly issued transactions will indirectly approve/reference it, thus the TTC will directly depend on the growth speed of the cumulative weight (while in the second paper it will analogously depend on the WW's growth speed) until it reaches a specific "confirmation threshold". We remind that both the cumulative weight and the WW are quantities that monotonically increase with time.

In a high regime we can distinguish between two phases:

- Adaptation period (i.e. the "confluence time" as denoted in the second paper), which is the time needed for a transaction to be approved by almost all the current tips;
- Issuance time, which is the time in which the cumulative weight grows with speed λw (or analogously, for the second paper, the WW grows with a speed derived from $\sum_{i=1}^{N} w(i)(1 - e^{-\delta\lambda w(i)})$ until it reaches the confirmation threshold.

The first paper computes the adaptation period in the setting in which a transaction has to approve exactly two tips, giving the result for the time t_0:
$t_0 \approx \frac{h}{W(\frac{1}{2})} \cdot (\ln(L_0) - \ln(\epsilon^{-1}))$

Following the approach of the second paper in which blocks can approve a set of k existing blocks, the formula is generalized to:
$t_c \approx \frac{h}{W(\frac{(k-1)^2}{k})} \cdot (\ln(L_0) + \ln(\epsilon))$

where W(.) is the Lambert W-function, which for large values of k can be approximated to log(k), giving the following final result for the confluence time:
$t_c \approx \frac{h}{log(k)} \cdot (\ln(L_0)) \approx \frac{h}{log(k)} \cdot \ln(\lambda h)$

On the other hand, the issuance time is explicitly computed by the second paper in the (analytically simple) case in which every node in the network shares the same value for their weight:
$t_{iss} \approx \frac{N}{\lambda} \cdot (-\ln(1 - \theta))$

However, due to the heterogeneous nature of the network, a more detailed model should be used as the one defined by a Zipf law (i.e. an empirical law describing the frequency of an event E_i, which is part of a specific set, in function of its rank in respect to the event's frequency itself). The Zipf law appears in many fields such as internet traffic data or formation of P2P communities, and is considered to be appropriate for use in our setting by the authors of the second paper. This model is well suited to describe a general network environment where the nodes' weight depends on N (the total number of nodes) and s (the Zipf parameter).

With all the previous results, we obtain:
$TTC \approx \frac{h}{log(k)} \cdot \ln(\lambda h) + \frac{N}{\lambda} \cdot (-\ln(1 - \theta))$

4.3.1 Related comments

As we can see from the formula of the TTC (which is less or equal than the confluence time plus the issuance time), there are three main components that affect the final value and we note the following:

- As one can expect, by having a block reference more tips (k becomes larger) we obtain a lower TTC.

  It is somewhat intuitive to think that for each newly issued block a greater number of tips will be referenced, and therefore the expected time for a given block to be referenced by almost all newer blocks will be smaller.

- We also note that the TTC will be longer as the average computation time h grows or if the mean number of tips in the system L_0 is large.

  The reason for this is that since a block has a greater number of tips to choose from, the probability to reference a given block would be smaller, slowing down the total time to have it referenced by a large percentage of new blocks.

5 Attack scenarios

5.1 Attacks presented by the first paper

The first paper discusses some attack scenarios on the proposed protocol and on the Tangle structure itself, along with how they can be mitigated: this section of the review focuses on the described attack scenarios and mentions their possible solutions as algorithms which were previously discussed and described.

5.1.1 Double-spending attack

The double spending attack consists of the following steps:

- The attacker issues a "honest" transaction in which he should receive some goods, (and receives them upon confirmation, that is, when the transaction gains a relatively large cumulative weight);
- The attacker uses his computing power to either issue a double-spending transaction and produce many small transactions that approve only the double-spending one, or to issue a single double-spending transaction with a large own weight that approves transaction issued before the "honest" one;
- The attacker then hopes for other nodes of the network to converge to its malicious transaction's branch, which will cause its "honest" transation's branch to be orphaned.

The probability of success for this type of attack is equal to the probability of the malicious branch of the Tangle DAG to have a total weight greater than the "honest" branch.
Because the cumulative weight of the network grows linearly with speed λ (the "adaptation period" for the honest transaction issued by the attacker is over, hence this transaction has been confirmed), this probability is equal to:
$P[W_m > W_h] = 1 - e^{\frac{-\mu}{\lambda w}} \approx \frac{-\mu}{\lambda w}$
where λ is the arrival rate of transactions issued by honest nodes, μ is the computing power of the attacker and w is the mean weight of a generic transaction.
It is worth noting that, despite being small, the probability of performing a successful attack of this type is still greater than zero.

A possible approach to mitigate double-spending attacks consists of fixing an upper bound for the own weight of issued transactions, in order to render the attackers unable to choose arbitrarily large weights for their transactions. This limitation results in a much smaller probability for an attacker to successfully perform a double-spending attack.

It is also important to note that the arrival rate λ of transactions issued by honest nodes should be greater than the computing power μ of the attacker for the system to be secure (as seen before, $P[W_m > W_h] \approx \frac{-\mu}{\lambda w}$, thus we need $\lambda \gg \mu$ to obtain a small success probability).

5.1.2 Parasite chain

Another possible attack scenario described in the first paper is the following:

- The attacker issues a malicious transaction and starts to build its own local sub-tangle;
- The attacker grows his local subtangle by forming a long chain of transactions (a "parasite chain") occasionally referencing the main tangle, thus gaining a higher and higher cumulative score for its local chain;
- At the time of the attack, the attacker issues a "honest" transaction on the main tangle and waits for its confirmation, then tries to artificially inflate the number of tips in his parasite chain;
- The attacker hopes that other nodes issuing new transactions reference his subtangle so that the legitimate branch on the main tangle is orphaned.

We note that in this type of attack, the parasite chain cannot reference the main branch after issuing the "honest" transaction.

In order to defend against this type of attack, we must define weights of newly issued transactions such that the cumulative weight of honest nodes of the network is larger than the cumulative weight of malicious nodes.

Based on the results of the previously described MCMC tip selection algorithm, it is easy to see that the attacker's tips will not be selected to be referenced with a high probability.
The cumulative weight of the parasite chain will be in fact small relatively to the main tangle it refers to. Therefore it is unlikely that the random walker will decide to transition to a transaction contained in the parasite chain, thus won't traverse it unless it starts from there, which is, again, very unlikely.

5.1.3 Splitting attack

The last type of attack presented in the first paper refers to the so-called "splitting attack".
The attacker tries to split the main tangle in two "parallel" branches by issuing a double spending transaction, and then keeping their cumulative weight balanced so that both branches can continue to grow: if this succeeds, the attacker will be able to spend the same funds on both branches.

An effective way to prevent this kind of attack is to introduce an analogous rule to the "longest chain rule" of blockchain-based distributed ledgers. If the probability of a honest node to choose one of the two parallel branches was much larger than 0.5, then it would be very difficult for the attacker to maintain the balance equal in the two

branches, since in this case we would have many nodes quickly referencing only one of the parallel branches, thus the network would converge to it by orphaning the other one. We could therefore choose a rapidly decaying function for the transition probability of particles of the MCMC algorithm (described before), and spawn them deep into the tangle so that the random walker has a high probability of starting before the originating double spending transaction of the splitting attack. With this type of transition rule for the tip selection algorithm we obtain a high probability of nodes of the network converging into a single tip of one of the two parallel branches, even if the difference in cumulative weight between them is very small.

It is worth noting that in a realistic setting, this attack would be highly unlikely, since network delays would prevent attackers from being able to immediately know about the preferences of the other network nodes (i.e. their newly issued transactions) and act timely in order to influence the behavior of the rest of the nodes.

5.2 Attacks presented by the second paper

Before providing an analysis of the possible attacks on the protocol described in the second paper, we provide a brief introduction on the considered system and adversarial model.

5.2.1 Communication level

The system's network is composed of nodes that communicate over a peer-to-peer protocol, realizing an overlay network that allows the exchange of "packages" between neighbors.
A node can issue a block by first creating and attaching it to its local Tangle and then by including it in a package to be sent to its neighbors.
Upon receiving a package, a node checks if the contained block was received and analyzed before: if this is not the case, the node checks the block's validity and adds it to its local Tangle DAG.
Packages are sent over directed edges of the overlay network between nodes.
Further assumptions about the communication model made by the second paper include both random block issuance and random package delay.
These assumptions are relative to an asynchronous system model, while the second paper also proposes a "synchronized layer" of the described OTV protocol which allows for a partial synchronization of nodes at fixed time intervals: the following analysis of attacks presented in the second paper takes into account the "asynchronous layer" of the protocol, and therefore assumes a fully asynchronous system model.
The "synchronized layer" of the OTV protocol is instead proposed in the second paper as a solution to mitigate these kinds of attacks.

5.2.2 Adversarial model

In an ordinary environment, other than "honest" and "faulty" nodes, "malicious" nodes can interfere with the rest of the network: "malicious" nodes are considered to be nodes controlled by an "attacker".
Attackers are assumed to have a limited computational power, such that they cannot actively break cryptographic hash functions and signature schemes of other nodes.

Attackers are however considered “omniscient” in the sense that they “know immediately” about all state changes of the honest nodes.
Moreover the adversary nodes do not need to follow the algorithms used for processing blocks and updating their state and can generate malicious packages to be sent to its neighbor nodes.
Finally, in the analysis of attacks of the second paper, a distinction of adversarial strategies is made for attacks on the “voting level” and attacks on the “communication level”.

5.2.3 Security, safety and liveness

The second paper focuses on the security of the algorithms, providing a more in-depth analysis of both the liveness and safety properties.
Liveness is the property of nodes to eventually make a decision on the state of a specific transaction (i.e. reaching a specific confirmation threshold value θ).
Safety is the property of any two nodes reaching an agreement without confirming any conflicting transaction. The paper distinguishes between non-conflicting and conflicting transactions scenarios.

5.2.4 Non-conflicting transactions

The liveness property is intrinsically linked to the tip selection algorithm, under the assumption that the number of tips remains roughly stationary in time.
In the asynchronous model, if we denote with q the weight of the malicious nodes and with $\theta \in (0.5, 1]$ the value of the fixed threshold for approving a transaction, we have the following result: if $q < 1 - \theta$, then every non-conflicting transaction will eventually be confirmed by every honest node.

Despite the ability of an attacker to delay blocks with honest transactions, thus incrementing the number of tips and slowing down the time to confirmation, we note that it is possible to define a synchronization mechanism (like the solidification process) for honest nodes to help mitigate this problem (this mechanism consists of the ”second layer” of the OTV protocol previously described).
Other possible attack scenarios consist of attackers issuing blocks that do not remove any tips, i.e. referencing other already referenced blocks only. As seen before, the number of malicious blocks per each ”honest” block that could be generated by malicious nodes, which have a cumulative weight of q, is q/(1-q) for each of the honest blocks generated by honest nodes. In order to mitigate this type of attack, honest nodes need to keep the number of tips in the system stationary: this can be achieved by allowing and encouraging honest nodes to reference more blocks per each newly issued block.

5.2.5 Conflicting transactions

While both liveness and safety can be guaranteed in many scenarios in case of non-conflicting transactions, the same cannot be stated when conflicting transactions are introduced.
In the case of an asynchronous communication model, many edge cases in which these properties do not hold can be defined: these edge cases consist of attacks on the

stability of the network, labeled "metastability attacks" by the authors of the second paper, in which the network is kept in an "undecided" state and is unable to reach a "stable" consensus state.

5.2.6 Attack on Communication level

For the following scenario, the attacker has the ability to control the exchange of communication packages containing blocks of honest nodes, and does not need to influence the voting level. In fact, there is no assumption on the total weight of the attacker, as it only needs to adjust the delay of the packages.

5.2.7 Metastability attack 1

In the first type of metastability attack, the attacker constantly keeps honest nodes in an "undecided" state, hindering consensus finding. The system is kept in a symmetric state between two conflicting transactions by the attacker, which is assumed to have control over the communication level, and which tries to delay the confirmation of blocks voted by the majority of the other nodes in order to force them to change their vote repeatedly, thus remaining in an "undecided" state.
The attacker can therefore delay consensus finding arbitrarily, by adjusting each blocks' confirmation delay accordingly.

5.2.8 Voting Level

When the attacker is assumed to have the ability to issue blocks at a higher rate in respect to honest nodes, he could perform a metastability attack directly at the voting level. For the following considered scenarios, the attacker is assumed not to have any influence on the communication level, and we work under the assumption of a synchronous communication model.

5.2.9 Metastability attack 2

In this kind of attack, the attacker is able to influence the vote of honest nodes by voting for a certain block with its own controlled nodes: when honest nodes start to converge on this block, the attacker can change its nodes' vote in favor of another block, and thus forcing the honest nodes to change their vote too.
The attacker can therefore repeatedly switch its nodes' voted block between the two blocks with the higher number of votes issued by honest nodes, and therefore force the honest nodes to repeatedly change their votes, rendering the network unable to reach a consensus state.

5.2.10 Bait-and-Switch attack

In a "Bait and Switch" attack, the attacker, which is assumed to have a large weight for its controlled nodes, doesn't have to rely on his block issuance rate, but can maintain the network in an undecided state by repeatedly switching its controlled nodes' vote in favor of one of the blocks containing conflicting transactions (continuously issued by the attacker itself).

This causes the honest nodes to repeatedly change their vote as well, because of the large weight of attacker nodes which results in a high level of influence on the rest of the network's decisions.

5.2.11 Communication and Voting level

In the previous attack examples, the attacker could only harm the liveness property: this was achieved by relying on either a strong control of the communication level or a strong control of the voting level.
However, when the attacker has a stronger control on both the communication and the voting level, thus having a strong control on the network in general, the safety property cannot be guaranteed anymore. The second paper proves that if the attacker has a weight q such that $q > \theta - 0.5$, then the safety property can be broken, i.e. there exists a certain time in which two honest nodes confirm different conflicting transactions.
The attacker can in fact issue two conflicting transactions x and y, and convince a group of honest nodes X to confirm transaction x (by exploiting its control on the communication level) which was sent first to nodes in X.
In the meanwhile, the rest of the honest nodes Y of the network were convinced by the attacker to vote for transaction y, which was again sent to nodes in Y first.
The attacker can later change his vote so that the approval rate of transaction x becomes AW(x) < 0.5 (by exploiting its control on the voting level), and let nodes in X know about the preferred transaction y of the other honest nodes in group Y.
Nodes of the network will now eventually confirm transaction y while also having a group of honest nodes (X) previously confirming the conflicting transaction x: the safety property is broken.

5.2.12 Realistic conditions

In realistic conditions, in which votes and blocks are propagated through the network with unpredictable and random delays, the attacker would not have the required level of control over the network to perform the discussed attacks, therefore the system is likely to converge to a consensus state.
It is however worth noting that the time needed for the convergence of the system to a consensus state could be, in general, very high, harming the safety of the system.To mitigate this problem, the second paper introduces a synchronized reality selection algorithm based on a distributed random number generation (dRNG), previously discussed in this review.

5.2.13 Security guarantees

The second paper presents and proves a theorem on the eventual consistency of the OTV protocol under the assumptions of both random block issuance and random package delay (Theorem 1).
Theorem 1 states that if the weight of attackers is less than 50% of the total weight of the network, the system will eventually converge on a consensus state with a probability close to 1.
Under the given (weak) assumptions, neither safety guarantees nor conclusions about the confirmation status of transactions are given by the theorem.

The results of Theorem 1 are proved for the asynchronous leaderless consensus protocol, which employs a weight-based voting scheme on the Tangle ("On Tangle Voting"), discussed previously in this review.

The second paper also presents a second theorem (Theorem 2), with which guarantees about the liveness and safety of the protocol are given, and which makes stronger assumptions about the network and its synchronicity, but does not require random block issuance and random package delay as in the previous theorem.
Theorem 2 states that, under probabilistic synchronicity assumptions, if the portion of the total weight of the network controlled by the attacker is both $q < 1 - \theta$ and $q < \theta - 0.5$, the voting protocol for nodes using the reality selection algorithm with common random coin (presented in the second half of the second paper and discussed in the previous section) converges to a consensus state.

The assumptions made by Theorem 2 include:

- Blocks from honest nodes are received by all honest nodes in a certain time d with a certain probability P
- An adversary controls a portion q of the total weight of the network
- All nodes perceive the same fixed set of conflicts
- A dRNG generates random values at fixed intervals and honest nodes receive them before time d with probability P
- The majority of honest nodes issue transactions with probability P at a rate that allows for their potential confirmation

The result of Theorem 2 is achieved by extending the asynchronous leaderless consensus protocol that employs a weight-based voting scheme on the Tangle, presented in the first part of the second paper, by incorporating the capability to synchronize the nodes at certain intervals with the help of a common random coin (i.e. the synchronized reality selection algorithm based on a distributed random number generation, previously described in this review).
Furthermore, Theorem 2 also allows to estimate the consensus time of the algorithm (in contrast to Theorem 1, which does not offer any bound for the consensus time): an overview of the simulation results that display the performance of the protocol is presented at the end of the second paper.

6 IOTA, Coordicide and additional notes

In this section of the review, we provide a brief history of the IOTA cryptocurrency, for which the original paper's proposal was envisioned for, and on which the second paper bases its proposals (while still presenting a general two layer solution which allows for a high level of configurability of the protocol, making it adaptable to the needs and security requirements of the system in which it should be implemented).

We then introduce the concept of "coordicide", described in detail in the paper by S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer, O. Saa, W. Sanders, L. Vigneri, W. Welz, and V. Attias, named "The coordicide" (2019).
The paper, in its current "Working Paper" version, is available at link https://files.iota.org/papers/20200120_Coordicide_WP.pdf.
The majority of the authors of the paper, researchers at the "IOTA Foundation", are also authors of the first paper (Popov) and the second paper (Moog, Mueller, Penzkofer, Sanders) we analyzed.

Most of the concepts and information discussed in this section are taken from the IOTA foundation website (available at link https://www.iota.org) which provides information about the latest developments of the IOTA protocol, as well as seminars given for the "Introduction to Coordicide Specifications" series (of 2020) and the "IOTA 2021 Research Symposium" (available at the IOTA blog posts at link https://blog.iota.org/introduction-to-coordicide-specifications-f251254ab27d and https://blog.iota.org/iota-research-symposium-2021 respectively).

6.1 IOTA and Coordicide

IOTA is an open-source cryptocurrency designed for the Internet of things industry (IoT).
As seen before, It uses a DAG to store transactions on its ledger, which makes the cryptocurrency and related DLT more scalable than blockchain based distributed ledgers.
Furthermore, IOTA does not use miners to validate transactions, but nodes that issue a new transaction on the network must approve previous transactions instead: as seen before, this solves various problems of blockchain-based DLTS, and in particular it allows to remove the bottleneck represented by the elected leader for the creation of a new block.
Transactions can therefore be issued without fees, which facilitates microtransactions needed for the IoT industry.

6.1.1 IOTA 1.0

The first version of the IOTA network went live in 2016.
This first version was based on the concepts mentioned in its "white paper" proposal (i.e. the first paper analyzed by this review), along with some other concepts as originally envisioned for the Tangle-based distributed ledger of IOTA 1.0.
The main concepts behind the implementation of the first version of the IOTA distributed ledger were:

- DAG based ledger structure
- Cumulative-weight biased random walks as a consensus mechanism
- Fixed PoW rate control (in order to submit information to the Tangle, nodes would need to provide a fixed amount of proof of work)

- Ternary logic (possible thanks to quantum computations, as opposed to binary logic)
- Quantum Resistance (achieved thanks to a "Winternitz One Time Signature" scheme, or "WOTS")

The last 2 main concepts were not explicitly mentioned in the new version of the first paper analyzed in this review, but small hints of these concepts are still given by some briefly mentioned concepts such as the focus on quantum computation in the last section of the white paper (for quantum resistance as well as quantum ternary logic) and the mention of a weight system based on powers of 3 (reflecting the ternary logic on which the original paper's proposal was supposed to be based on).

The focus on quantum cryptography and computation of the original white paper proposal, justified by the early and promising results in the field around the year in which the white paper was published, soon suffered a setback in the following years as quantum cryptography still represents a theoretical result as of today and as of version 2.0 of the IOTA protocol.
It is still worth mentioning that the IOTA foundation designed the newest "Coordicide" approach for IOTA 2.0 with quantum computation in mind, in order to make a transition to quantum cryptography as soon as new and concrete results in the field will be available.

The design principles of the IOTA 1.0 network was the "feeless" nature of its protocol, its speed and scalability, and the freedom of usage of the cryptocurrency and its associated distributed ledger technology and system.

As mentioned in the introduction of this review, the IOTA network was also proposed as an alternative to the existing Blockchain-based technologies, in order to provide a fast and continuous service, opposed to the slow and periodic service of blockchain-based ledgers, a parallel instead of sequential writing and processing of transactions, and finally a miners/validators-free architecture.

The implementation of the IOTA 1.0 protocol as it was originally envisioned soon led to problems which were in need to be solved:

- The reliance on the PoW needed to issue transactions led to an inevitable tradeoff between security and access control (imposing a low PoW requirement for transaction issuance would lead to less Sybil protection and no guarantee about a possible "honest majority" of nodes, while high PoW requirements would make the network harder to access for honest nodes with limited computational power)
- Conflicts spamming can drastically lower the network's transaction throughput (when conflicts are introduced in the system, branches are created, thus the abundance of branches leads to higher times for the convergence of the network to a single branch, and transactions contained in orphaned branches will never get approved)

- Performance issues were introduced along with protocols (random walk algorithms are slow because of its reliance on the cumulative weight, which depends on the future state of the ledger and which requires nodes of the network to update information about this value upon the issuance of new transactions; furthermore, the quantum cryptography was not yet ready-to-use in DLTs when the IOTA 1.0 was first introduced, and the ternary nature of the system made some of the operations required for the protocol slow, and also led to scalability issues because of the the WOTS which required a new "one time" signature at each use)

Version 1.0 of the IOTA protocol had also been the target of phishing, scamming, and hacking attempts by malicious users, and presented vulnerabilities that led to legal issues: the most prominent examples are the seed-generator scam of January 2018 (a fraud in which more than US$10 million worth of IOTA tokens were stolen from users using a malicious online seed-generator for their IOTA tokens' ownership password) and the security flaws of the "Curl-P-27" hash function used by the IOTA 1.0 protocol (which, once discovered and disclosed to the public, led to the IOTA Foundation receiving considerable backlash in their handling of the incident).

6.1.2 IOTA 1.5

The drawbacks of the 1.0 version of the IOTA protocol, along with the discovery and exploit of its vulnerabilities, led to the creation of a new version of the protocol which temporarily fixed most of the problems by introducing a coordinator node managed by the IOTA Foundation.
The coordinator would issue new transactions, called "milestones", at certain time intervals, using the same protocol and rules as regular nodes of the network: milestones would be issued as "final", in the sense that every transaction in a block which is part of the past cone of a milestone would be considered confirmed.
The coordinator made it possible to solve the PoW problem by allowing a low PoW requirement for transactions and blocks issuance, while the security of the network would be granted by the coordinator.
In particular, the later introduction of the so called "Chrysalis Coordinator", consisting of an enhanced coordinator node using a different algorithm for tip selection than regular nodes, made it possible to "totally order" transactions of the Tangle using milestones, thus allowing nodes of the networks to attach their transactions to tips without verifying their correctness (i.e. regardless of conflicting transactions).
This in turn solved the "conflicts spamming" problem of version 1.0 of the IOTA protocol.
The "balance keeping" method used by the protocol also switched from an "account based" approach to a "UTXO" approach (in order to make conflicts more easily detectable).
The performance problems of the 1.0 version of the protocol were solved by removing the reliance on ternary logic, WOTS, and the random walk algorithm (instead, the coordinator takes care of the problems that were originally solved with these approaches).

While seemingly solving all of the problems introduced by the original IOTA protocol, this new solution's most obvious flaw is its centralization: the "coordinator node" (and

in particular, the IOTA Foundation which operates it) has control over the entire network.
Furthermore, attacks on the IOTA 1.5 protocol were still being performed, among which the most severe was represented by the vulnerability exploit of the third-party payment service "MoonPay" integrated in the wallet application managed by the IOTA Foundation in November 2019 (which resulted in the theft of US$2 Million worth in IOTA tokens).

6.1.3 IOTA 2.0 and Coordicide

The aforementioned drawbacks and vulnerabilities of the IOTA 1.5 protocol, along with the fact that the solution it implements was originally intended to be a temporary solution to version 1.0's problems, led to the development of the newest version of the IOTA protocol: version 2.0.
While most of its functionalities are still being tested and gradually updated, an introduction of the IOTA 2.0 protocol and its specifications is provided in the working paper "The Coordicide", by the researchers of the IOTA Foundation.
The "Coordicide" approach (as the name implies) aims to remove the coordinator from the IOTA network by introducing and implementing novel protocols and technologies which would supposedly lead the IOTA cryptocurrency to be the first to solve the so called "Blockchain Trilemma" (mentioned in the introduction of this review).

The IOTA 2.0 protocol consists of three main layers:

1. An underlying "Network Layer" (which maintains connections and direct communication between nodes)

2. A "Communication Layer" (which maintains the Tangle DAG structure)

3. A decentralized "Application Layer" (which consists of the core applications ran by nodes, like consensus applications or dRNGs, and of the third party applications which can be integrated in the IOTA protocol, like smart contracts)

The "Coordicide" approach introduces the concept of "Mana" as a sybil protection measure, similar to the concept of "weight" of a node of the analyzed second paper's protocol.
In each transaction, the token holder, i.e. the node issuing the transaction, pledges their "mana" to other nodes of the network, and the amount of pledged mana is equal to the amount of funds (IOTA tokens) moved in the transactions.
In order to gain mana, nodes must convince other token holders to pledge them their tokens (e.g. by buying them or by providing rewarded services): in this sense, the "mana" mechanism can be considered as a "Delegated Proof of Token Ownership".
There are two types of "mana":

- Access Mana (which determines the level of "access" a certain node has to the Tangle, i.e. its "throughput", thus how many messages it can issue relative to the total network throughput; it can also be "rented" by nodes of the network from other Access Mana token holders);

- Consensus Mana (which determines the "voting power" of a node, i.e. the "Approval Weight" of transactions and their finality in the network; it cannot be sold on the open market unlike the Access Mana).

A new structure for messages to be sent over the network is also introduced, reflecting the changes of the protocol (e.g. the introduction of a timestamp mechanism in place of the time reference represented by milestones of the 1.5 version) and the aforementioned protocol's 3 layer structure: in this sense, the "network layer" consists of the message bytes, the "communication layer" consists of the message itself, while the "application layer" determines the message's payload (different payloads can be defined for different third party applications built on top of the IOTA network, allowing for an easy decentralized applications integration).

An "autopeering" mechanism for nodes is also introduced, with which nodes can select their potential neighbors of the network to whom sending all of the generated or received messages. The "neighbor selection protocol" of the "autopeering" mechanism uses both a screening process (called "Mana Rank") and a score function (which takes into account both a private and a verifiable public randomness) to select neighbors based on their Consensus Mana (only nodes with a similar Consensus Mana can become mutual neighbors).

A new tip selection algorithm is also defined, based on the idea of the R-URTS algorithm presented in the second paper (and analyzed in the previous sections this review). This new algorithm introduces the concept of "approval switches", which separates approvals into two categories: strong approvals (require transactions not to have conflicts in their past cones) and weak approvals (allow the content of a message to be approved without checking for its past history, allowing messages to participate in the Tangle without creating unmergeable branches).

Another novel concept introduced by the "Coordicide" protocol is the "Adaptive Proof of Work": in order to issue a new message, nodes need to compute a very small Proof of Work, but when consecutive messages are issued in a short time interval, nodes need to solve progressively more difficult puzzles (making it impossible to create a burst of spam transactions).

The Ledger of the IOTA 2.0 protocol is maintained in a similar way to what was described in the second paper we analyzed: a Ledger DAG is maintained along with a Tangle DAG, and a "reality-based UTXO ledger" is used to allow nodes to keep track of their reality ledger state.

Finally, the Coordicide approach also provides a new consensus mechanism named "Fast Probabilistic Consensus" (FPC), i.e. a leaderless consensus protocol based on voting and mana, as well as a dRNG.
A node's vote is weighted according to the amount of consensus mana it holds, and the finality of transactions of the Ledger is defined by a confirmation threshold for the Approval Weight of a node that is randomized.
As clearly noticeable, this approach is very similar to the second "synchronized" layer of the leaderless Nakamoto consensus protocol provided by the second paper we analyzed in previous sections of this review.
Randomizing the threshold for majority makes it extremely difficult for adversaries to manipulate the consensus by making it converge to a specific value or prolonging consensus time.

6.2 Additional Notes and Comments

We conclude this review by providing some general comments on not-in-depth analyzed concepts of the first and second paper.

Both the first and the second paper we analyzed did not provide an in-depth analysis of two performance metrics that are crucial for a distributed ledger technology: throughput and energy consumption.

6.2.1 Throughput

The throughput of the system is related to the number of transactions per second the system can process, and despite related concepts being discussed in the performance-related section of this review (namely the number of issued transactions per second), neither the first nor the second paper evaluate the quantitative aspect of the throughput of the network.
We will however point out that DAG based systems allow for the throughput to increase as size of the network increases: this happens because the addition of new nodes inside the network would lead to more transactions being issued, and therefore to more unapproved transactions being approved by the newly issued ones, resulting in a faster confirmation times.
The scalability of the network is therefore directly proportional to its transaction throughput, as opposed to blockchain-based technologies in which throughput must decrease when new nodes are added to the network because of the sequentiality of the network and the need to wait for longer times for the propagation of a single transaction.

6.2.2 Energy Consumption

The papers also don't provide a quantitative analysis of the energy consumption of the protocol, while still being briefly mentioned in the second paper.
In general, the energy consumption of blockchain-based distributed systems is usually artificially inflated in order to provide protection against "sybil attacks" in the form of "Proof of Work".
The proposed protocol of the second paper allows for a sybil protection mechanism based on node's weights instead of PoW, which in turn allows to obtain a lower energy consumption, in a similar way to "Proof of Stake" based blockchains.
Furthermore, energy consumption can also be lower than PoS based mechanisms (such as Ethereum, which is based on an "account-based" model for balance keeping) because of the "reality based UTXO ledger", which allows blocks to avoid considering the entire ledger history for their computations in order to validate transactions, and has a lower communication overhead because of the merge of blockchain-based roles of validator and issuer into a single role.
In IOTA 2.0, the energy consumption is estimated at about 0.00011 KWh, which, compared to Ethereum's estimate of 0.002601 KWh and Bitcoin's astounding 707 KWh, is very low: this makes sense when considering that the IOTA protocol was designed for the IoT industry, where computing powers of devices is generally limited.

7 Conclusions

Our review focused on the main concepts of the two analyzed papers, by starting with a brief introduction of distributed ledgers and the drawbacks of the most widely adopted blockchain-based technology.

The approaches analyzed in the reviewed papers propose solutions to mitigate the investigated problems: their main contribution is the introduction of a distributed ledger based on the structure of a directed acyclic graph, called the "Tangle".
We described the main features and functioning of the "Tangle" and highlighted the differences between the two analyzed papers, while then describing the related proposed algorithms and protocols used as consensus mechanisms.
We then discussed the analysis of the main performance metrics of the Tangle-based DLT, and provided an explanation of possible attacks to the papers' proposed solutions, along with related possible mitigation strategies and security guarantees offered by the protocols.

We concluded our analysis by providing a brief history of the IOTA cryptocurrency, to which the papers refer to when describing their proposed solutions, along with an additional introduction to a novel Tangle-based protocol called "Coordicide" and further considerations about important performance metrics of the solutions proposed by the analyzed papers.