SoK: DAG-based Consensus Protocols

***Abstract*—This paper is a Systematization of Knowledge (SoK) that focuses on Directed Acyclic Graph (DAG)-based consensus protocols in Distributed Ledger Technologies (DLTs). Our study evaluates their impact on performance and their tradeoffs concerning consistency, availability, and partition tolerance, as postulated by the CAP theorem.**

We delineate the key functionalities and tradeoffs of DAG-based consensus protocols, highlighting iterative improvements and deviations from foundational models. We analyze aspects such as tip selection, mempool management, transaction finality, and incentive structures, assessing their significance in the performance and security of DAG-based systems. Additionally, we identify research gaps and suggest directions for future work to refine DAG-based consensus mechanisms for diverse applications in distributed computing.

I. Introduction

Distributed Ledger Technology (DLT) plays a pivotal role in technology services and applications, particularly emphasising its profound impact on the cryptocurrency market. DLT empowers the operation of an append-only, immutable ledger meticulously upheld by a network of nodes within a trustless environment. The foremost promise of DLT lies in its ability to furnish transparency and operational efficiency to distributed ledgers, catering to individual and organizational needs. One of the most prominent exemplars of DLT thus far is Bitcoin [1], and its foundational DLT architecture is widely recognised as Blockchain. The remarkable potential exhibited by Bitcoin has generated significant interest across both the industrial and academic sectors. This surge of interest has subsequently given rise to a diverse spectrum of new cryptocurrencies and a myriad of applications built upon the blockchain architecture. These applications harness the potential of blockchain DLT to facilitate secure, dependable, and transparent transaction processing. Despite the numerous benefits of the blockchain to various domains, it is imperative to acknowledge that several limitations and challenges persist in the path toward its adoption in real-world implementations.

The principal impediments inherent to blockchain technology manifest primarily as performance bottlenecks, encompassing constraints such as low throughput, poor scalability and high transaction confirmation latency. Several studies have illuminated different trade-offs in blockchain, especially in speed-security [2] or performance-security [3]. A few studies [4], [5] emphasise that a blockchain system can simultaneously satisfy at most two out of three properties: decentralisation, consistency, and scalability (DCS-satisfiability theorem). Therefore, to tackle the performance bottleneck of blockchain, Directed Acyclic Graph (DAG)-based DLTs have emerged in recent years as a solution. These DLTs have underlying DAG-based consensus protocols for collectively reaching a consensus on the ledger state.

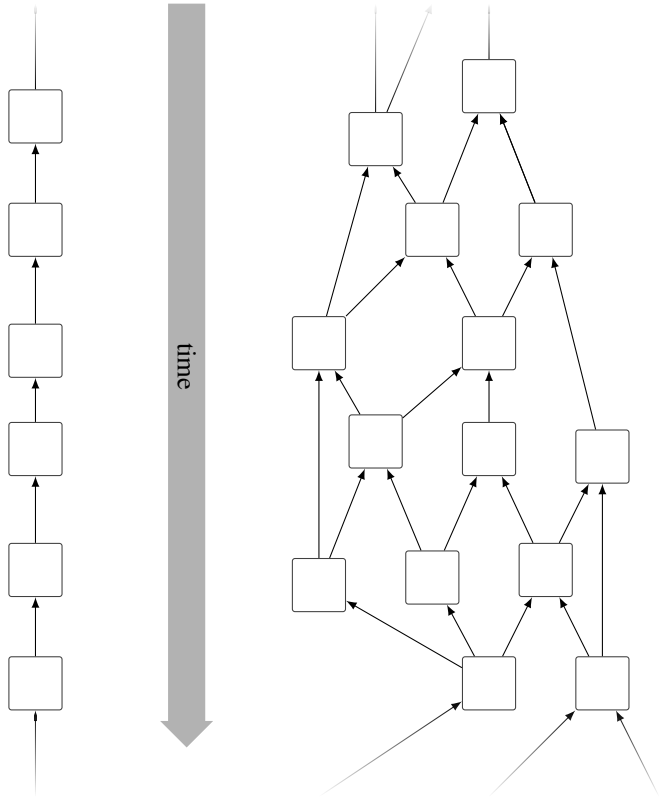

Fig. 1. Blockchain and blockDAG

A. What is a DAG-based Consensus Protocol?

In consensus protocols, a critical aspect is the data structure employed to achieve consensus among participants. A DAG data structure is a type of graph in which edges have directions, and the graph does not contain any directed cycles meaning it is impossible to return to the starting vertex by DAG traversal.

Traditional consensus protocols typically operate in rounds. Within each round, the consensus data, encapsulated in a *block*, references its predecessor, thus forming a linear sequence that extends back to the genesis block. All existing blocks form a directed tree structure, where a leaf stands for a potential agreement among participants. While a directed tree is technically a DAG, with each vertex pointing to its predecessor, its linear nature imposes significant structural limitations.

DAG-based consensus protocols, however, expand beyond this linear framework. In these systems, the referencing mechanism for blocks allows a more intricate interconnection, enabling the concurrent addition of multiple blocks; see Figure 1 for a visualisation. This paper delves into such DAG-based consensus protocols, particularly focusing on protocols where some blocks could reference more than one preceding block, thus diverging from the linear constraints of traditional models.

B. Why DAG-based Consensus Protocol

In the pursuit of transcending the inherent trade-off between security and performance, and in response to the performance bottlenecks, DAG-based consensus protocols were proposed as a solution. These protocols promise high scalability and fast confirmation of transactions, hence effectively addressing the intricate balance between security and performance. The directed tree structure in these protocols allows multiple transactions to be verified in parallel, which leads to a scalable DLT architecture. The scalable DAG-based DLTs achieve higher throughput and scalability than linear blockchain systems and aim to reduce the complexity, latency, and storage overhead. Following, we describe the advantages and challenges of DAG-based consensus protocols compared to linear chain protocols.

Advantages and promises of DAG-based consensus protocols compared to blockchains

1) Scalability: DAG-based protocols can process transactions in parallel rather than sequentially, as in traditional blockchains. This capability significantly increases their scalability and potential to handle a more significant throughput. Moreover, as more participants[1] join the network, the performance can improve rather than degrade, at least to some extent.
2) Latency: DAG-based protocols can reduce the transaction confirmation time without compromising the security and allow the possibility of building a protocol not relying on synchronicity assumption. This ensures faster confirmation, making these protocols favorable options for applications requiring high throughput and fast confirmation.
3) Flexibility: The DAG architecture allows for more flexible consensus mechanisms and can adapt to various network conditions and use cases, potentially making it more versatile than a linear blockchain.
4) Transaction fee: DAG-based protocols have low or no transaction fees, allowing them to include more transactions without raising fees. This makes DAG-based protocols suitable for micropayment systems, e.g., between devices and sensors.

Challenges in DAG-based consensus protocols compared to blockchains

1) Security risks: The relatively complex structure of DAGs introduces new security challenges, including vulnerability to certain types of attacks that do not affect traditional blockchains in the same way.
2) Understanding and adoption: The more complex nature of DAG-based protocols can make them harder to understand and adopt, particularly for developers accustomed to traditional blockchain technology. Protocols that only provide partial ordering may encounter more difficulties in being adopted to existing infrastructures.
3) Consensus mechanism maturity: Consensus mechanisms used within DAG-based DLTs are often newer and less tested than those used with blockchains, creating uncertainty around their long-term stability and resilience against evolving network threats.
4) Reachability challenge: The concept of reachability within a DAG refers to the ability of a new block to reference earlier transactions or blocks, which is central to the structure's integrity and its efficient functioning. This ability has ramifications for functionalities, such as pruning (removing old data to save space) and the operation of light clients (nodes that do not store the full ledger data).

The number of DAG-based consensus protocols has been increasing. However, there have been few attempts to consolidate these protocols and provide a comprehensive analysis of them. As the DAG-based DLT community continues to grow, it is becoming increasingly important to have a holistic understanding of the various architectures and trade-offs involved.

C. Contribution

This SoK contributes to the following:

- It presents a comprehensive overview of the foundational constituents of DAG-based consensus protocols (Sec. II).
- It articulates the DAG-based consensus protocols in two main categories: 1) Availability-focused; 2) Consistency-focused, and examines their attributes and differences (Sec. III).
- It offers a brief security analysis (Sec. IV) and an insightful description of the salient features in the context of DAG-based consensus protocols (Sec. V).
- It explores some important topics in DAG-based consensus and highlights their significance and implication (Sec. VI).
- It provides valuable insights, and key takeaways from the existing works and outlines research gaps throughout the paper to inspire future research (Sec. VII).

Note: This paper focuses on the conceptual/architectural design of the DAG-based consensus protocols. However, the paper does not discuss or present any performance evaluation of the included protocols.

D. Related Work

Recently, there has been a growing interest in DAG-based protocols in both industrial and academic circles. An overview of the development is depicted in Figure 2. There are several lineages in the development. One lineage is GHOST [6], SPECTRE [7], GHOSTDAG [8] and DAG KNIGHT [9]. Another lineage is Aleph [10], DAG-Rider [11], Tusk [12], Bullshark [13], Cordial Miner [14] and BBCA-Chain [15].

Numerous DAG-based consensus protocols have been developed to optimise the complex interplay between security and performance while building additional features. The objective of this paper is to systematize a wide variety of research on consensus protocols that are based on DAG structure.

We acknowledge a concurrent work [16] on this topic and argue that this work presents a complementary and enriched perspective that augments the comprehensive discourse. Furthermore, due to the recent development in the DAG-based DLTs, a need for updated and insightful views on these has emerged. Therefore, we fulfill this need in this work and

[1]Throughout the paper, we use participant and node interchangeably.

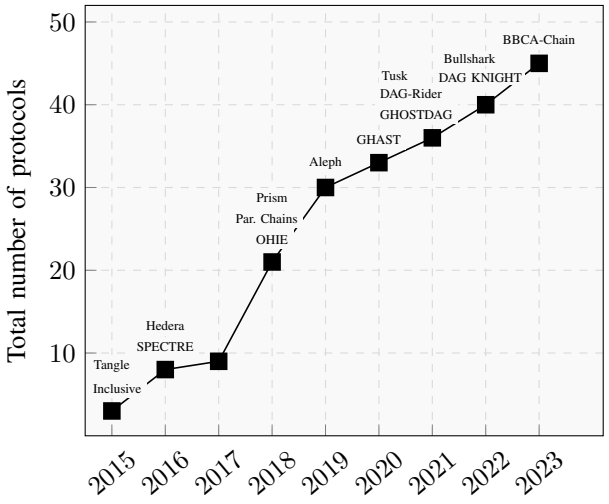

Fig. 2. The evolution of DAG-based consensus protocols over time

present a concise and conceptual systematization of DAG-based consensus protocols. The field of DAG-based DLT is continuously growing, and there is a surge of ongoing research aimed at improving the current state-of-the-art. Many DAG-based consensus protocols have been proposed to improve the existing state-of-the-art and address the main challenges in DAG-based consensus protocols compared to blockchains.

Wang et al. [16] presents a comprehensive analysis of existing DAG-based systems. Their survey gives a formulation of most DAG-based distributed ledgers encompassing aspects such as consensus, security, and performance. The classification is very elaborate and detailed, as is the comprehensive attack analysis. The paper represents a good starting point for acquainting oneself with the DAG-based blockchain systems. On the contrary, it is worth noting that the paper falls short in expounding upon critical insights and research voids within the present landscape of DAG-based consensus protocols. Moreover, considerable advancements in DAG-based DLT space have improved the existing state-of-the-art. These advancements must be put together for consolidated reference to the readers which this SoK does in a succinct manner.

Some other works are focused on the extensive spectrum of DLT. Bellaj et al. [17] surveys DLTs in a layered model, including chained, chainless and hybrid DLTs. Kannengießer et al. [18] studies different characteristics of DLTs and describes many trade-offs between these characteristics. Wu et al. [19] present a literature review on chain-based and DAG-based ledgers. The authors review several consensus protocols, give a detailed taxonomy of different structures in these protocols, and further present a few research issues in current ledgers.

The aforementioned works fall short of addressing several important aspects of DAG-based consensus protocols. These works not only fail to evaluate the protocols through the CAP theorem lens but also neglect to delineate crucial components and desirable properties of a DAG-based protocol. Therefore, this SoK presents a comprehensive investigation that delves into these facets offering insights into key topics, and further outlines current open research problems.

II. DAG-based Consensus in a Nutshell

A DAG-based consensus protocol arranges blocks or transactions in a DAG instead of a linear chain, allowing nodes to reach a consensus on the inclusion of these blocks. The directed edges in the DAG establish a causal order linked by cryptographic means, providing evidence of communication between the nodes. This concept is already present in Nakamoto's proposal, but DAG-based consensus protocols use it more explicitly, especially those that do not use an additional broadcast primitive like Hashgraph [20]. Moreover, each block typically has multiple references, allowing for the inclusion of more parts of the tree that are not in the longest or heaviest chain, unlike blockchain, where only the blocks in the longest chain are selected. This inclusion rule leads to processing more blocks or transactions, resulting in increased throughput.

DAG-based DLT systems arrange transactions/blocks in the form of a DAG topology. This graph consists of a set of vertices (or nodes) and directed edges between them. Each vertex represents a transaction or block, while the directed edges represent the relationships between these units. Specifically, an edge between two vertices represents a partial order relationship, where one unit verifies, confirms, or witnesses the other unit. An edge can also represent a hash reference from one unit to another. A directed graph is called a DAG if it does not contain any directed cycles. We also refer to Figure 1 for an illustrative comparison between a blockchain and a (block)DAG.

In a traditional linear blockchain, the longest chain rule serves two purposes: 1) agreeing on the included blocks and 2) establishing their order. However, when transitioning from linear chains to DAGs, these two functions need to be replaced. To address this, two primary classes of protocols have emerged: those that perform these tasks directly on the DAG itself and Byzantine Fault Tolerant (BFT) protocols that utilise supplementary broadcast primitives.

A DAG-based consensus protocol consists of the following components. To design a DAG-based consensus protocol, one needs to choose one option from the given options in each component. Following, we describe these components in detail.

A. Ordering

Distributed ledgers are systems that arrange blocks or transactions in a specific sequence. In traditional blockchain networks, this order is linear. However, in DAG-based systems, the arrangement is more complex and depends on factors such as the type of DAG and the consensus mechanism used.

Ordering of blocks in DAG-based DLTs can be categorized into two types: partial and total. Partial ordering arranges blocks based on their causal relationships within the DAG using topological sorting. Conversely, total ordering arranges blocks in a single, linear sequence using a parameter-based sorting approach. Both approaches cater to different system requirements, balancing efficiency and dependency accuracy.

B. Ledger Model

Most consensus protocols are agnostic to the underlying ledger models as they rely on a total ordering of transac-

tions. However, some protocols, like those in SPECTRE [7], IOTA [21], and Avalanche [22], deviate by adopting a partial order, challenging the idea that total ordering is indispensable.

Two prevalent ledger models serve as the backbone for transaction management:

1) *UTXO* (Unspent Transaction Output) In a UTXO model, transactions are transfers of value from previous transaction outputs to new unspent outputs in an inductive way. New unspent transaction outputs are called UTXO, used as inputs for new transactions. Bitcoin [1], and Cardano [23], epitomises this approach. It was adapted to the DAG-setting by IOTA [21] and Avalanche [22] since it sidesteps the need for total ordering by structuring the UTXOs as a DAG, promising higher parallelism in transaction processing.
2) *Account-based* In an account-based model, each public address is considered an account. Each account has a balance associated with it. Transactions signify direct value transfers between these accounts, enabling a straightforward balance update mechanism exemplified by Ethereum [24].

Object-based and message-based models offer a more general description of possible ledger models. In object-based ledgers, such as the UTXO model, transactions modify objects and thus only affect local states. Due to its localised change impact, this model naturally lends itself to sharding solutions. The causality in object-based models forms a DAG, suggesting that many scenarios do not require a total ordering of transactions thanks to their innate parallelism, as explored in the reality-based ledger model [25]. Conversely, message-based ledgers conceptualise transactions as messages that induce changes across a global state, highlighting a fundamental distinction in how transactions are processed and effectuated.

A mention of owned and shared objects, and how they interplay in contemporary platforms like SUI [26] further refines this classification, showcasing a transition towards more granular and flexible state management within DLT systems. Such differentiation could enhance understanding and stimulate the development of more scalable ledger solutions.

C. System Openness Model

Consensus protocols have two main classes of openness:

1) *Permissioned* In a permissioned system, participation is restricted. It requires potential participants (especially validators) to obtain permission from either the network starter or a set of rules governed by the network protocol. Typically, every participant is aware of any other participant.
2) *Permissionless* In a permissionless system, anyone can join and participate. In other words, anyone can become a node, validate transactions, and participate in the consensus process without needing permission from a central authority or governance. In recent work, Lewis-Pye and Roughgarden [27] present a thorough study of permissionless consensus protocols. They give a hierarchy of four settings according to the "degree of permissionlessness" in consensus protocols. This degree refers to how much knowledge a protocol has about its current participation and is often used explicitly in the protocol.

D. Network Model

Consensus protocols are designed based on different network models that describe how communication occurs between the participants. These models are often defined by the potential power of an adversary to control message delays. Three commonly used communication models are synchronous, asynchronous, and partially synchronous. To comprehend these models, it is essential to have a formal understanding of key concepts in network systems.

Delay parameter Δ defines the maximum delay (in time steps) a message suffers in a synchronous phase. The parameter Δ is known upfront to the nodes in the protocol.

Global Stabilization Time (GST) dictates when the underlying communication network switches from asynchronous to synchronous. Unlike Δ, GST is not known upfront in the protocol and can be chosen by an adversary.

1) *Synchronous* In a synchronous model, the messages between nodes are delivered within known and predictable time frames. Nodes follow a common global clock or synchronized time intervals to operate in lockstep. In short, for any message sent, an adversary can delay its delivery by at most Δ. This model is used in time-sensitive systems, making it suitable for some DLT-based real-time systems.
2) *Asynchronous* In an asynchronous model, the messages between nodes experience varying delays. An adversary can delay the delivery of any message sent by any finite amount of time, but eventually, the message gets delivered. This model offers the most flexibility, as message delivery times are unpredictable. These models are standard and used in scenarios where global synchronization is impractical.
3) *Partially Synchronous* In a partially synchronous model, messages are delivered within an unknown finite time. In the equivalent *eventually synchrony* model, the GST event will occur after a certain, unspecified time has passed. Moreover, any message sent at time t must be delivered by time $\Delta + max(t, \mathrm{GST})$. This model offers a balance between flexibility and accuracy. Nodes in this model aim to operate with timers that can measure time Δ after an event and there is no guarantee about the exact timing of message deliveries. These models are common in network environments where transient conditions may cause disruptions. After GST, the network guarantees that messages are delivered within the delay parameter Δ, providing a window of predictability which can be very useful for consensus algorithms, especially in fault-tolerant systems that must operate under the assumption of periodic network instability. These models are particularly relevant for DLT systems that aim to achieve consensus despite unpredictable network conditions.

E. Dynamic Availability

In certain consensus protocols, as highlighted by Neu et al. (2022) [28], nodes operate within a fluid participation framework that aligns well with consensus algorithms built on the "sleepy" model, put forth by Pass and Shi (2017) [29]. In these consensus protocols, nodes can exhibit dynamic availability, seamlessly transitioning between periods of activity and

dormancy up to a specified *Global Awake Time* (GAT). After this moment, an assumption is made that all honest nodes will be consistently online, facilitating consensus in an environment reflective of real-world participation and connectivity patterns. Thus, dynamic availability is the assurance of transaction being processed and finalized definitively despite any transient fault.

F. Additional Broadcast Primitive

DAG-based consensus protocols adopt different strategies to ensure the reliable propagation of blocks. These strategies can be broadly classified based on their reliance on additional reliable broadcast primitives.

Many DAG-based systems lean on established Reliable Broadcast (RB) protocols [30] to distribute blocks. These systems implement an RB protocol as an additional layer, ensuring consistent and reliably disseminated information among nodes, even in the face of network challenges or adversarial behaviour.

Conversely, some protocols employ the blockDAG structure itself to replicate the functionality of a reliable broadcast mechanism. The DAG's architecture inherently distributes data across the network so that blocks are organically verified and propagated without needing a specialized RB protocol.

Both approaches aim to maintain a secure and coherent state across distributed participants. However, they differ fundamentally in whether they view the blockDAG as sufficient for reliable broadcasting or require an additional overlay of RB protocol to strengthen information dissemination and security.

G. Adversarial Model

In analysing distributed network models, it is crucial to consider the adversarial elements that may aim to disrupt or undermine the network's integrity. These adversarial models conceptualise potential threats by defining the capabilities and limitations of an adversary's influence over the network and also provide theoretical limitations.

1) *Adaptive vs. Static* An adaptive adversary dynamically corrupts a fraction f (less than 50%) out of total n nodes during the protocol execution to compromise the protocol, thereby undermining security. However, a static adversary corrupts a certain number of nodes before the protocol execution and exercises complete control over them.
2) *Common Threshold vs. Resource-based Threshold* A common threshold adversary can corrupt up to f nodes from a fixed set of n nodes. This is the most common adversary model followed in a permissioned distributed system. Nevertheless, a resource-based threshold adversary can corrupt a certain number of nodes in the network bounded by the adversary's resources. A resource can be computational power or a finite financial resource such as a stake.

H. Leader-based V/s Leaderless Consensus

Consensus mechanisms within DLTs typically adopt either a leader-based or a leaderless structure. Leader-based consensus protocols designate a leader to propose an orderly view of the ledger transactions. This view becomes committed when a sufficient consensus, or votes, affirming the view is reached. Leaders may have a fixed position or be chosen through various mechanisms, such as Proof-of-Work (PoW), Proof-of-Stake (PoS) lottery, or a round-robin selection.

Conversely, leaderless consensus forgoes a single point of proposal power, relying instead on a collective decision-making process amongst the DLT nodes. It is important to note that proposing a global view comprises two distinct steps: suggesting new blocks and resolving potential conflicts. While these two functions typically coincide within most systems, they are not intrinsically linked, paving the way for innovative consensus designs that separate these roles, as discussed in the subsequent section.

I. Writing Access

Control over the writing access to the ledger is a critical aspect of distributed ledger technology (DLT), which defines who can contribute new information to the network. This access can be regulated in several ways, each with implications for security, decentralization, and efficiency.

1) *Lottery-based* There is a PoW/PoS lottery mechanism.
2) *Permissioned* There is a pre-defined set of validators that are allowed to write new blocks.
3) *Committee-based* A rotating or varying subset of validators is given the authority to write new blocks.
4) *Open Writing Access* Any node can propose the next block.

These models further diverge based on the presence of a leader. In a leader-based model, a single validator is exclusively appointed to write at a given time. Conversely, a leaderless model allows for simultaneous writing by multiple validators.

J. Mempool

For protocols where writing access is probabilistic or requires selection from a group, such as lottery-based or committee-based systems, a mempool is essential.

A mempool, short for memory pool, functions as a form of temporary storage for transactions that have been broadcast to the network but not yet included in a block. The mempool serves several important roles:

1) *Transaction Queue* It acts as a queue for pending transactions, maintaining them in an accessible state until they are selected and confirmed by a node with writing access.
2) *Prioritization* The mempool can prioritize transactions based on specific criteria, such as fee rates, ensuring that transactions with higher fees are confirmed more quickly.
3) *Availability* It ensures that transactions remain available for selection and confirmation, even when uncertain which node will write the next transaction or block into the ledger.
4) *Synchronization* In systems with multiple potential writers or validators, the mempool is crucial for synchronizing the state across these entities, allowing them to view and validate the same set of unconfirmed transactions.

In some DAG-based protocols, the blockDAG acts as a distributed mempool, accommodating pending transactions. This integrated approach negates the necessity for an independent mempool structure, thereby simplifying the transaction selection and validation processes. Additionally, as the blockDAG

maintains a real-time ledger of unconfirmed transactions, it enhances the system's transparency, rendering the mempool more verifiable and publicly accessible.

K. Finality

In DLTs, finality refers to the point at which a transaction or a block of transactions is considered irreversible and permanently part of the ledger. Finality comes in two forms: probabilistic and deterministic (or absolute).

1) *Probabilistic Finality* implies a non-zero risk of transaction reversal, which diminishes as the transaction becomes more deeply embedded in the blockchain or blockDAG over time. In systems that use a PoW mechanism, finality is probabilistic; with each additional block referring to a given block, the probability of the block being eventually included increases and becomes closer to 1.
2) *Deterministic Finality* provides absolute assurance of a transaction's permanence once it is recorded. Once a transaction is included in a block and that block is added to the chain by honest nodes, the transaction is considered final. Protocols using Byzantine Agreement (BA) or similar consensus mechanisms, often found in PoS systems, offer deterministic finality. They typically require a block to be validated by a supermajority, ensuring it is irrevocably part of the ledger.

Additionally, some DLTs with DAG structures may initially only offer probabilistic finality. However, they can obtain deterministic finality by designating special roles; for instance, witness nodes in Byteball [31] or nodes creating snapshot chains in Vite [32] and IOTA 2.0 [33].

L. Overview

Table I presents a diverse range of DAG-based consensus protocols, their underlying models, technical specifications/features, and key innovation ideas. The column "Innovation/Relation" highlights the novel contributions of each protocol or delineates its evolution from preexisting technologies.

It is important to acknowledge that the features and their implementation might vary or evolve over time, and the table is to be updated accordingly with the most recent resources.

III. DAG-based Consensus protocols

Consensus in distributed systems is a critical process through which all participants agree on a unified version of the system's state. Within DAG-based DLTs, a blockDAG is collaboratively built by the entire network, with each node maintaining a replica. Nodes must agree on this DAG and achieve consensus regarding the ledger's state.

To ensure a shared understanding of a ledger's history, nodes must consistently interpret their locally held DAGs. This is done by agreeing on a subset of interconnected blocks unambiguously determined by the DAG's inherent causal order. This subset is known as a prefix[2] of the DAG. Once nodes agree on this prefix, they can establish a total ordering for the included transactions. In certain protocols, they might only ascertain a partial ordering, which can nevertheless be adequate for executing transactions depending on the chosen ledger model, see Section II-B.

As we delve into the specifics of consensus protocols within DAG-based DLTs, it is important to remember the constraints outlined by the CAP theorem [44], [45], which posits that a distributed system can only simultaneously satisfy two out of the three following properties: consistency, availability, and partition tolerance, where

- **Consistency (Safety)**: Every read on the distributed system gives the latest write on each node. This means that all nodes see the same data at the same time. Consistency is equivalent to having a single up-to-date copy of the data.
- **Availability (Liveness)**: Every client request receives a (non-error) response irrespective of whether the read is the latest write, even if one or more nodes are down. This means the system is always operational, and every request is met with a response, but that response might not be the most recent data.
- **Partition Tolerance**: The system continues to operate despite any number of communication breakdowns between the nodes (network partition) in the system. This means the system can sustain any amount of network failure that does not result in a failure of the entire network.

Given these constraints, DAG-based consensus protocols face trade-offs and are classified into two primary classes.

1) **Availability-focused Protocols:**
   Focusing primarily on maintaining the continuous progress of the ledger, most of these protocols use variations of a weighted[3] lottery, where nodes are randomly selected to propose new blocks to the DAG and vote on the existing ones. Such protocols are typically suited for permissionless or dynamically available settings [27]. In this setting, a potentially large number of entities participate in the consensus process, and the list of active participants is unknown and changing over time. Canonical examples of availability-focused consensus protocols among linear blockchains are Nakamoto consensus [1] and Ouroboros [46]. When applied in a DAG-based architecture, this approach manifests in different forms, including structured and unstructured DAGs. The DAG structure aids in occupying the entire bandwidth effectively while ensuring faster finalization. It is worth noting that, if guaranteed, finalization in availability-focused protocols is often probabilistic.
   a) *Structured DAGs:* These protocols enforce strict rules for adding blocks, keeping an organized DAG topology.
   b) *Unstructured DAGs:* These protocols represent a more flexible approach to DAG construction, where nodes have greater liberty in attaching new blocks without stringent adherence to predefined rules. However, the trade-off often lies in increased complexity for achieving consensus and validating transactions.

[2] A prefix in this context is a set of blocks in a DAG which is closed by causal order relations imposed by the DAG.

[3] For instance, the weight could be computing power, stake, or storage.

TABLE I
OVERVIEW OF DAG-BASED CONSENSUS PROTOCOLS

Protocol	Writing Access	Additional Broadcast Primitive	Additional Mempool	Leader-based	Dynamic Availability	Ordering	Finality	Openness	Innovation/Relation
Nakamoto [1]	PoW	No	Yes	Yes	Yes	Total	Probabilistic	Permissionless	Genesis of blockchain and cryptocurrencies
GHOST [6]	PoW	No	Yes	Yes	Yes	Total	Probabilistic	Permissionless	Improved upon Nakamotos's chain structure for higher throughput using heaviest chain selection
Inclusive [34]	PoW	No	Yes	Yes	Yes	Total	Probabilistic	Permissionless	Evolved from GHOST, presents a game-theoretic model for reward among miners.
Byteball [31]	Open	No	Yes	No	Yes	Total	Deterministic	Permissionless	Witness nodes for syncing and ordering transactions using main chain index (MCI)
SPECTRE [7]	PoW	No	Yes	No	Yes	Partial	Probabilistic	Permissionless	Further evolved from GHOST, prioritizing fast confirmations using pairwise voting
PHANTOM [8]	PoW	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Introduced k-cluster for better structure over SPECTRE's DAG
GHOSTDAG [8]	PoW	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Focused on reducing latency and refined k-cluster approach
DAG KNIGHT [9]	PoW	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Parameterless expansion from GHOSTDAG without latency bounding
Prism [35]	PoW	No	Yes	Yes	Yes	Total	Probabilistic	Permissionless	Separated consensus functions into multiple parallel chains
GHAST [36]	PoW	No	Yes	Yes	Yes	Total	Probabilistic	Permissionless	Tree-graph approach and adaptive mechanisms for performance and security
Hashgraph [20]	Permissioned	No	No	No	No	Total	Deterministic	Permissioned	Gossip about gossip mechanism for events, not just transactions with virtual voting
Jointgraph [37]	Permissioned	No	No	Yes	No	Total	Deterministic	Permissioned	Improves throughput and latency of Hashgraph using a supervisor node with additional events
Aleph [10]	Permissioned	Yes	Yes	No	No	Total	Deterministic	Permissioned	Utilized round-based DAG with randomized leader election
DAG-Rider [11]	Permissioned	Yes	Yes	No	No	Total	Deterministic	Permissioned	Post-quantum safe, structured round-based protocol with optimal resilience and complexity
Avalanche [22]	Open	No	No	No	No	Partial	Probabilistic	Permissionless	Introduced leaderless consensus using repeated sub-sampled querying
OHIE [38]	PoW	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Parallel chain architecture, where all chains have equal roles and are utilized uniformly
Parallel Chains [39]	PoW / PoS	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Parallel chain architecture, where all chains have equal roles and are utilized uniformly
Expected consensus [40]	PoS	No	Yes	No	Yes	Total	Probabilistic	Permissionless	Increased throughput and adapted consensus for storage resource
Nano [41]	Open	No	Yes	No	Yes	Partial	Deterministic	Permissionless	Block-lattice structure allowing asynchronous updates
Chainweb [42]	PoW	No	Yes	No	No	Partial	Probabilistic	Permissionless	Employs simple payment verification (SPV) for token transfer
Graphchain [43]	PoW	No	Yes	No	Yes	Partial	Probabilistic	Permissionless	Cross-verifying transactions, employing fair reward mechanism for its participants
Meshcash [43]	PoW	No	Yes	No	Yes	Total	Deterministic	Permissionless	Ebb-and-flow type protocol
Tusk [12]	Permissioned	Yes	Yes	No	No	Total	Deterministic	Permissioned	Separated mempool management from consensus with Narwhal and Tusk
Bullshark [13]	Permissioned	Yes	Yes	Both	No	Total	Deterministic	Permissioned	Zero overhead BFT protocol providing practical garbage collection and timely fairness after GST
Cordial Miner [14]	Permissioned	No	No	Both	No	Total	Deterministic	Permissioned	Reduce the latency of DAG-based consensus protocol using best effort broadcast
IOTA 2.0 [33]	Open	No	No	No	Yes	Partial	Deterministic	Permissionless	Conflict resolution using the DAG and reality-based ledger state
BBCA-Chain [15]	Permissioned	Yes	Yes	Yes	No	Total	Deterministic	Permissioned	Latency reduction by several network trips by parallel block broadcast using BBCA primitive

2) **Consistency-focused Protocols:**
The protocols of this type prioritise ledger consistency by drawing from classical Byzantine Fault Tolerant (BFT) protocols [30], [47], [48]. Consistency is mostly achieved in permissioned and quasi-permissionless environments, e.g. see [27]. These protocols can be further categorized based on how participants disseminate their blocks.

a) *Best-effort broadcast* In these protocols, block dissemination is done through gossiping. Thereby, such protocols require mechanisms to handle equivocation.
b) *Reliable/consistent broadcast:* These protocols employ broadcast primitives to block dissemination which ensures participants to maintain a consistent DAG view.

We depict Table II to show which consensus protocols fall in which category. The subsequent subsections will explore these categories, analyzing key protocols and their innovations.

A. Availability-focused Protocols with Structured DAG

1) Parallel instances of Nakamoto consensus: The most straightforward approach to improve Bitcoin's scalability is to execute m concurrent instances of the Nakamoto consensus protocol as proposed in *Parallel Chain*s [39] and *OHIE* [38]. Mining simultaneously and uniformly for all m chains is achieved by a similar technique in both protocols. A newly mined block must include the digests of the latest blocks from all chains, and a miner remains unaware of which chain a new block will extend until it solves the puzzle. After successful mining, block's hash value decides which chain it belongs to.

Establishing total ordering on parallel chain architectures can be achieved differently. In OHIE, a ranking system for blocks enables participants to find the order. Each block is equipped with a *current rank* and a *next rank*; the current rank mirrors the next rank of its referenced predecessor, while the next rank represents the highest current rank observed across all chains. Parallel Chains employs a different strategy, where one specific chain is designated for the purpose of synchronization. Each block in the parallel architecture references a block from this synchronization chain. The chronological order of the entire network is then inferred based on the sequence of blocks in this special chain.

Parallel Chains and OHIE guarantee the same safety properties as the Nakamato Consensus, [1], and improve either the throughput or latency, as their scalability does not offer both desired features simultaneously. Multiple other protocols can also fall into this category, e.g. Chainweb [42], but they degrade either security or latency compared to Nakamoto consensus. Nano [41], Vite [32] protocols also follow a similar concept where participants maintain their parallel chains.

2) Decoupling block's functionalities: A more powerful approach suggested in *Prism* [35] to augment scalability in parallel chain architectures involves deconstructing the multi-faceted functionalities of blocks, as seen in Nakamoto's consensus. Traditionally, blocks in Bitcoin simultaneously fulfil three key roles: 1) leader election, 2) transaction proposing, and 3) voting on the causal history through parent links. Prism entails a clear segregation of these functions into distinct types of blocks, each dedicated to a specific purpose, such as transaction processing, block proposal, and voting.

This segregation results in the formation of different types of chains within the architecture. Primarily, it creates one *proposer* chain, where each block references a preceding block within the proposer chain and several *transaction* blocks. In parallel, m *vote* chains emerge, with each chain functioning to cast votes determining the total order of blocks within the

TABLE II
CATEGORIZATION OF CONSENSUS PROTOCOLS IN DAG-BASED DLTS

Consensus in DAG-based DLTs			
Availability-Focused		Consistency-Focused	
Structured DAGs	Unstructured DAGs	Best-effort broadcast	Reliable/consistent broadcast
Expected consensus [40]	SPECTRE [7]	Hashgraph [20], [49]	Aleph [10]
Inclusive [34]	PHANTOM [50]	Jointgraph [37]	DAG-Rider [11]
GHAST/Conflux [36], [51]	GHOSTDAG [8]	Cordial Miners [14]	Tusk [12]
OHIE [38]	DAG KNIGHT [9]	IOTA 2.0 [33]	Bullshark [13]
Chainweb [42]	IOTA 2.0 [33]		BBCA-Ledger [52]
Parallel chains [39]	Byteball [31]		BBCA-Chain [15]
Prism [35]	Meshcash [43]		Mysticeti [53]
	Graphchain [54]		
	Avalanche [55]		

proposer chain. Such a structural reorganization clarifies the roles within the blockchain and allows to achieve simultaneously high throughput, low (constant) latency, and the same safety guarantees as in the Nakamoto consensus.

Let us note that there are other predecessors of Prism where functionalities of blocks are decoupled, e.g. FruitChain [56], Bitcoin-NG [57], but which have not improved some other aspects, e.g. latency.

3) Tree-structure in a DAG: In *GHAST* (Greedy Heaviest Adaptive SubTree) [36], [51], each block except genesis has precisely one outgoing parent edge and can have multiple outgoing reference edges. The parent edges construct a tree within the broader DAG, and the reference edges establish temporal precedence, indicating that the referenced block predates the block containing the reference. Once a winning chain within the tree is selected, a deterministic order over all referenced blocks can be established. The selection of this winning chain is influenced by the weights assigned to each block, akin to the GHOST protocol [6]. Here, the fork choice rule systematically favours the heaviest subtree at every fork, typically equating block weight with the extent of computational work done.

However, GHAST diverges from traditional protocols in its adaptive response to potential liveness attacks. During such scenarios, it modifies the weight distribution among blocks. While normally all blocks carry equal weight, under attack detection, the protocol randomly selects only a few blocks to assign non-zero weights and keeps the expected block weight at the same level.

Inclusive [34]: The motive of Inclusive blockchain protocols was to build the DAG structure for blocks - the blockDAG that provides increased throughput while keeping a similar security guarantee as linear-structured blockchain protocols. The main idea of these protocols was to allow transactions from all the blocks to be included in the ledger. An inclusive rule is defined to select a main chain from the DAG, and further non-conflicting blocks from the outside DAG are appended to the final block structure. The Inclusive protocol also rewards miners whose blocks are not in the main chain but are contained within the DAG.

4) Round-based chain of tipsets: Another attempt to improve the latency of Nakamoto's consensus was suggested in *Expected Consensus* [40]. This heaviest-chain style protocol operates in rounds. Each round involves a cryptographic sortition on the weighted list of participants. It is parametrized such that, on average, a given number of m participants may be eligible to propose a block every round. Every block must reference a *tipset*, a set of blocks sharing the same round and a parent tipset. Every block adds its weight (determined by the sortition) to the chain of tipsets, and the consensus is achieved by following the heaviest chain of tipsets.

B. Availability-focused Protocols with Unstructured DAG

1) Unstructured Block-DAG protocols: This includes GHOST-lineage protocols and a few more protocols where blocks (consisting of transactions) form an unstructured DAG.

The *GHOST* protocol [6] represents a divergence from Nakamoto's longest chain rule. Designed to increase throughput, GHOST selects the Greedy Heaviest-Observed Sub-Tree, enabling shorter intervals between block creations without compromising security — a concept explored in [58] in further detail. GHOST capitalizes on the proof of work in off-chain blocks, traversing the tree-like structure that emerges from chain forks. This alternate selection method for the main chain is specifically tailored to mitigate issues associated with network latency, presenting an advancement in the performance of blockchain protocols.

SPECTRE [7] protocol focuses on building a blockDAG structure to allow concurrent and parallel block creation, with separate mechanisms for mining and consensus, leading to a notion of weak liveness for greater scalability. SPECTRE uses a recursive weighted voting technique where each new block in the DAG submits votes (preference ordering) over every pair of blocks based on which block they believe occurred first. The final ordering in SPECTRE is based on the majority vote on pairwise ordering across all the blocks. However, this ordering is extendable to a total order due to Condorcet paradox [59]. While SPECTRE only achieves partial ordering, it does not need a synchronicity assumption but is working under partial synchronicity.

PHANTOM [50] protocol is an extension of the SPECTRE protocol. It aims to achieve a total ordering but under a synchronous network model. PHANTOM works by separating block creation or mining from the consensus mechanisms. It builds on the intuition that blocks created by honest miners are well-connected, while adversarial blocks are not well-connected and thus should be excluded. This well-connectedness is expressed by the notion of k-cluster. Once

this k-cluster is determined, a total block order is established via a topological ordering. The synchronicity assumptions become apparent in the choice of the parameter k, which depends on the latency and the desired throughput. Moreover, the identification of such k-cluster is NP-hard, GHOSTDAG, [8], is proposed as a practical alternative; it also addressed possible attack vectors, pointed out by [60].

DAG KNIGHT [9] is an evolution of GHOSTDAG. Unlike GHOSTDAG, DAG KNIGHT assumes no upper bound on network latency, leveraging a dual min-max optimization approach to dynamically find the largest k-cluster for each k, selecting the minimal k covering at least 50% of the DAG. This design accommodates latency variations for safety and provides a responsive protocol to actual network latency rather than a hard-coded latency bound as in GHOSTDAG.

Meshcash [43] is a framework containing two layered protocols to reach a consensus on the blocks added to the DAG using PoW. The blocks are arranged in layers; each block belongs to a layer (includes a layer number in its header) and refers to blocks in the previous layer. The consensus protocol is an ebb-and-flow style consensus protocol [61] with a slower protocol, called the "tortoise", that favours safety and an interchangeable faster one, the "hare" protocol, that favours liveness. Meshcash presents its security guarantees with formal proofs in an asynchronous network model.

IOTA 2.0 [33] consensus protocol is an ebb-and-flow style consensus protocol [61] based on delegated PoS that allows permissionless writing to the DAG by all accounts. Every block in IOTA 2.0 references at least two randomly selected tips, which results in an unstructured DAG. Unlike the common approach in which the total order on the log of transactions is found after a prefix of a blockDAG is finalized, IOTA 2.0 decouples these two processes - agreement on conflicting transactions is done in parallel to block dissemination. Based on the perception of a locally maintained DAG and online validators, each node generates commitments for past slots, i.e., time intervals of fixed duration. A slot commitment represents a hash digest of the accepted data (e.g., blocks and transactions) issued within the slot. Each commitment is linked to the commitment of the previous slot, resulting in a slot commitment chain that represents the prefix of a DAG. This allows the network to progress the DAG and the ledger even when the network experiences partitions. Once the network recovers from its divided state and a supermajority of validators is back online, nodes switch to the heaviest chain, and finalization on the slot commitment level recommences.

2) Unstructured Transaction-DAG protocols: This class includes protocols that append transactions directly in their DAGs without the block as a wrapper, e.g., Graphchain [54].

Byteball [31] protocol allows users to add transaction units to the DAG by signing them and paying a fee based on the data size using the internal currency, 'bytes'. These units are linked in a DAG, containing hashes of prior units to confirm their validity and create a partial order. In Byteball, there are trustworthy nodes called witnesses chosen by users based on the reputation of the nodes. These witness nodes periodically generate transaction units that help compute a main chain in the DAG. Consensus is then found by the total ordering of the transaction with respect to the main chain.

Avalanche [55] protocol allows nodes to repeatedly query a random group of nodes about the validity of a transaction. Further, to be able to vote on several transactions at the same time, the nodes employ a DAG structure. When a node promptly accumulates sufficient positive responses for its query, it solidifies its decision. Critiques have emerged regarding the security (especially liveness) of the protocol [62], [63], and the Avalanche project stopped working on the DAG-version of the consensus and maintains a chain version of it.

C. Consistency-focused Protocols with Best-effort broadcast

The protocols in this category rely on validators, a special subset of nodes with a non-zero weight in the voting process. Blocks are propagated using best-effort broadcast or gossiping protocol. During this propagation, the nodes need to ensure that the recipient of the new block also has knowledge of the causal history of the block. In optimistic cases, this can result in a small number of network trips to achieve finality for blocks. Typically, these protocols can tolerate adversarial validators holding less than $1/3$ of the total weight. It will be convenient to refer to validators holding more than $2/3$ of the total weight as a *supermajority*, and the main assumption in the following protocols is there exists an honest supermajority of validators.

One main problem of reliable or consistent broadcasting blocks among the validators is the problem of equivocations. This describes the situation where a validator presents different blocks to different parts of the network. In the following protocols, the DAG structure is used to avoid equivocations using a mechanism of double approvals. Specifically, a block X is *approved* by a block Y if X is reachable from Y in the DAG and the block creator of X does not equivocate in the causal history of block Y. A block X is *doubly approved* by a block Y if a supermajority of blocks exists in the causal history of block Y. Each validator can be shown to have at most one block in a given round which is doubly approved by a supermajority of nodes. The idea was introduced in *Hashgraph* [20], [49], and its variation was used in other protocols.

Hashgraph suggests constructing a DAG by gossiping over gossiping events[4]. In this protocol, each node maintains its chain of events. When receiving an event block from another node, it randomly selects at least one peer to disseminate information about its current knowledge about blocks, e.g., a new block within the maintained chain is created which in addition references the received block. A node advances to the next round once its block doubly approves a supermajority of blocks from the previous round.

By employing this idea of double approval in the resulting DAG structure, the Hashgraph protocol decides when the network reliably receives each event. The *received* timestamp is then assigned, and the total ordering is achieved by sorting events over the received timestamps.

Jointgraph [37] reduces the number of voting rounds in Hashgraph by introducing a *supervisor node*. An event in

[4]In Hashgraph, the term an *event* is used instead of a block

Jointgraph is final if the event receives more than $2/3$ votes from the nodes in which one of the votes is from the supervisor node. The supervisor node also creates snapshot and storage events which are used to finalize the events from the ordinary nodes and provide a total order. When the supervisor node is offline, Jointgraph switches to the Hashgraph process.

Cordial Miners [14] is a family of simple and efficient Byzantine Atomic Broadcast protocols [64] with instances for the asynchronous and eventual synchronous models. They improve the latency of comparable protocols such as DAG-Rider [11] and Bullshark [13] (described in Section III-D) and offer a faster commitment of new blocks. Unlike these aforementioned protocols, which rely on Reliable Broadcast (RB) [30] for disseminating vertices in the DAG, Cordial Miners protocols diverge from this approach. As RB incurs costs in terms of latency and message complexity, cordial miners protocols establish a partially-ordered structure called the *blocklace*, which serves as a generalized form of a blockchain data structure. The core concept behind Cordial Miners is the utilization of this blocklace for tasks such as block distribution, equivocation-exclusion, and block ordering. Notably, while the blocklace may contain equivocating blocks, a miner removes these blocks in its local DAG during the ordering without incurring any additional communication or latency costs.

An asynchronous payment system, *Flash* [65], is built by encoding payment transactions into the blocks of blocklace. The resulting ordering is only partial and hence weaker than the total ordering achieved by Cordial Miners.

D. Consistency-focused Protocols with Reliable/consistent broadcast

These consensus protocols are based on classical BFT algorithms. These protocols are either leader-based, similar to most BFT algorithms, or leaderless.

Aleph [10] protocol improves the Hashgraph complexity by building a round-based structured DAG and employing an efficient binary agreement protocol [66]. Like the Hashgraph protocol, Aleph separates the network layer (communication DAG) from the protocol logic (virtual voting and ordering). The execution of protocol happens in terms of virtual rounds. In each round, every node should create a block after learning a large enough portion (at least $2f + 1$) of blocks created by other nodes in the previous round. The protocol works under the general synchronicity and hence utilizes a trustless ABFT Randomness Beacon to circumvent the FLP-impossibility result, see [67], to reach a total ordering.

DAG-Rider [11] is an asynchronous Byzantine Atomic Broadcast (BAB) protocol with optimal resilience and optimal complexity. DAG-Rider creates the round-based structured DAG similar to Aleph. DAG-Rider improves upon Aleph and Hashgraph protocols by providing amortized complexity and making the protocol post-quantum safe. DAG-Rider is constructed in two layers: communication and ordering layer. In the communication layer, nodes reliably broadcast their proposals (messages) and form a DAG of the messages they deliver. In each round of DAG-Rider, each node broadcasts at most one message, which should contain references (strong edges) to messages (i.e., at least $2f + 1$) of previous rounds. The message can also have references (weak edges) to messages of rounds before the previous round. The weak edges ensure the validity property of BAB. Furthermore, in the ordering layer of DAG-Rider, each node observes its local DAG and locally orders all the messages in DAG by employing randomization. This randomization is achieved through a global perfect coin, implemented using threshold signatures that circumvent the FLP-impossibility result.

Narwhal & Tusk [12]: Narwhal is a DAG-based, structured mempool incorporating concepts from reliable broadcast [68] and reliable storage [69], with the addition of a byzantine fault-tolerant threshold clock [70] for round advancement. To achieve asynchronous consensus with minimal latency, the Tusk protocol is built atop Narwhal. This approach empowers each node to reach a consensus on agreed-upon values by examining its local DAG without any additional messages.

Tusk refines DAG-Rider, transforming it from theory into an implementable system. This is achieved through three pivotal steps: Firstly, Tusk adopts a Quorum-based reliable broadcast instead of the conventional reliable broadcast utilized in DAG-Rider. Secondly, refining the commit rules enhances the latency in common cases. Lastly, Tusk removes the weak links of DAG-Rider, enabling efficient garbage collection.

Bullshark [13] is a zero-overhead BFT protocol optimized for the synchronous case and achieves substantially reduced latency compared to both Tusk and DAG-Rider. In the partially synchronous version, Bullshark is the most performant and robust compared to existing protocols. Bullshark upholds all the desired properties of DAG-Rider while reducing the common-case latency during the period of synchrony. The main feature of the Bullshark protocol is to exploit the synchronous periods and remove the complex processes of view-change and view-synchronization. Bullshark attains amortized complexity and addresses the imperative facet of fairness.

BBCA-LEDGER [52] is a single broadcast consensus protocol with reduced latency by several network trips compared to above-described protocols such as DAG-Rider, Tusk, and Bullshark. The consensus logic is built over a novel broadcast primitive called Byzantine Broadcast With Commit-Adopt (BBCA). BBCA-LEDGER allows parallel block broadcasts such that each broadcast block is finalized in the log independently, thereby exhibiting high throughput and low latency. BBCA-LEDGER decouples the block finalization and sequencing, which results in fast committing of transactions with low latency during common cases and maintaining high throughput during network partitions or faults.

Apart from the above, Fino [71] and *Mysticeti* [53] are consistency-focused protocols based on broadcast primitives.

IV. Security

To assess the correctness of a DAG-based consensus protocol, the protocol must possess safety and liveness properties. Informally, safety asserts that certain undesirable never happen, while liveness asserts that something good eventually happens and the system keeps making progress.

A. Properties

- Safety: It refers to the guarantee that once a block (transaction) is accepted by an honest node, the probability of getting the block reverted is negligible. This includes properties like double-spending prevention, ensuring no transaction can be spent more than once; and data integrity, ensuring all transactions are valid and have not been tampered with. In a DAG-based consensus protocol, safety is maintained by validating and confirming transactions using protocol rules.
- Liveness: It is property of a protocol ensuring its uninterrupted operation despite adversarial behavior or network failures. This entails the continuous processing and confirmation of transactions by consensus participants, facilitating their ongoing addition to the ledger. Liveness encompasses property like network availability, ensuring the network is always available and accessible to the nodes.

Bitcoin Backbone-inspired Properties Garay et al. [72] analyzed the core of the Bitcoin protocol and presented an elegant backbone protocol. They defined and proved the two fundamental properties: common prefix and chain quality in Bitcoin under a static setting and a synchronous network. Many subsequent Bitcoin-inspired blockchain systems followed the backbone protocol and analyzed their security. Later, Pass et al. [73] presented the analysis of the backbone properties of blockchain in an asynchronous network setting. While there's limited work replicating Bitcoin's backbone properties for DAG-based consensus, Meshcash [43] presents security properties for its DAG-based consensus protocols in the influence of backbone protocol. Another paper [74] shows a game-theoretic analysis of the DAG-ledger backbone with a brief introduction to different security properties of DAG-based distributed ledgers. Following, we present backbone properties for unstructured DAG-based consensus protocols, where K refers to the last K blocks added in the DAG.

- *Common prefix*: With overwhelming probability (in K), at any point, the valid DAGs DAG_i, DAG_j of two honest players i, j respectively can differ only in the last K blocks.
- *DAG uniformity:* With overwhelming probability (in K), in the absence of block appending, for the valid DAGs DAG_i, DAG_j of two honest players i, j, eventually, $DAG_i = DAG_j$.
- *p-DAG growth*: With overwhelming probability (in K), at any point in the execution, the valid DAGs of honest players grew by at least K blocks in the last p rounds, where p is called the DAG-growth of the protocol.
- *q-DAG quality*: With overwhelming probability (in K), for any K consecutive blocks in any valid DAG DAG_i of an honest player i, the fraction of blocks that were contributed by honest players is at least q.

B. Security-Performance Trade-off

The primary objective of DAG-based consensus protocols is to surpass the traditional security-performance trade-off seen in linear ledgers, pushing network capabilities further. However, while boosting throughput, their security measures often fall short of the robustness offered by Nakamoto consensus (NC).

Numerous blockDAG protocols have sought to mitigate the security-performance trade-off. Nevertheless, only a select few, such as Prism [35], OHIE [38], and DAG KNIGHT [9], furnish security guarantees on par with NC and show the ability to withstand corruption of up to 50% of computational power. These protocols effectively disentangle security and performance from their respective proofs. Moreover, DAG KNIGHT assumes no upper bound on network latency.

Wu et al. [3] unveil the Congestible Blockchain Model (CBM), offering insights into the security-performance balance within DAG-based protocols. Unlike uniform upper bounds, CBM sets case-specific limits on block propagation delay, crucial in high-throughput scenarios where simultaneous blocks increase propagation delays. Prism and OHIE protocols, analyzed through CBM, show compromised security and performance in high-throughput environments.

The majority of DAG-based consensus protocols prioritize enhancing throughput but struggle with ensuring robust security. Zhang et al. [75] introduce NC-Max, diverging from the trend of adjusting NC's backbone. NC-Max enhances performance by allowing full network-supported throughput while strengthening defense against block-withholding attacks compared to NC. This strategy, adaptable to existing or new DAG-based DLTs, breaks the security-performance trade-off without modifying the backbone protocol.

C. Attack Analysis

We categorise the attacks in the following two main classes.

1) *Liveness-oriented attacks* aim to hinder honest nodes from contributing blocks to the ledger and achieving consensus. Adversaries execute these by corrupting nodes, exhausting their resources, or disrupting consensus message transmission. These attacks in DAG-based protocols encompass strategies like basic liveness attacks, balance attacks, transaction flooding, censorship, and Sybil attacks.
2) *Safety-oriented attacks* aim to invalidate blocks of honest nodes, preventing their addition or causing their removal from the ledger. Adversaries use strategies to invalidate honest blocks, ensuring their own blocks are accepted to gain more profit in incentive-enabled consensus. Examples include incentive attacks, parasite-chain attacks, long-range attacks, double-spend attacks, and equivocation attacks, where adversaries manipulate subsets of nodes to commit conflicting blocks, violating consensus safety.

We present these attacks, including their assumptions, main targets, instances, and potential countermeasures in Table III within the context of various DAG-based consensus protocols. Some protocols address specific attacks; for instance, Avalanche [55] discusses employing Sybil protection mechanisms, while Nano [41] remains unaffected by Sybil attacks.

V. Desirable Properties

A. Ordering

Partial order can be found in protocols such as IOTA [84], and Graphchain [54] where from every arbitrary block, there is a path to the genesis block. Moreover, protocols such as

TABLE III
ATTACK ANALYSIS OF DAG-BASED CONSENSUS PROTOCOLS

Attack	Description	Assumptions	Possible Countermeasures	Instances
Basic liveness attack	Adversary tries to disrupt the progress of consensus for a certain period of time	Block generation rate is significantly higher than block propagation rate	Encoding a conservative strategy to ensure the consensus progress [51], [61], [76]	GHOST [6] Inclusive [34] PHANTOM [50]
Balance attack	Adversary partitions the network into balanced subDAGs, and employs a dynamic strategy to traverse the subDAGs and select a favorable one to maximize his profit	Adversary has enough computation power to partition the network and to devise a strategy for traversal in different subDAGs	By reducing the block generation (or mining) rate, for example in GHOST [6], or in paraller-chain DAG protocols [35], [38]	Conflux [36], [51] Inclusive [34]
Transaction flooding attack	Adversary generates multiple transactions and floods the network and protocol data structure, consuming storage	Either creating a transaction does not incur sufficient cost or the adversary has enough resources to create multiple transactions	1. By introducing transaction rate limiting methods, e.g., using transaction fee [77], 2. By employing network layer protection, 3. By employing client puzzle schemes [78]	Nano [41] Vite [32] Jointgraph [37]
Censorship attack	Adversary proposes blocks faster and censors certain blocks (honest proposer block), hoping to convince nodes to accept his block and to invalidate certain blocks	Adversary controls a significant part of the ledger and dominates the proposer block sequence	By employing censorship resistant techniques, for example in Aleph [10] or techniques described in [79], [80]	Prism [35]
Sybil attack	Adversary creates a large number of pseudonymous identities to conduct malicious activities	Either creating identities is not expensive or the adversary has enough power to create identities	1. By making the identity creation expensive, 2. By employing sybil protection mechanisms, e.g., PoW/PoS puzzles, 3. By a randomized membership selection in committee-based systems	Nano [41] Hashgraph [20] Avalanche [55]
Incentive attack	Adversary deviates from the Random Transaction Selection (RTS) strategy and either tries to earn greater reward than honest nodes or harms the transaction throughput [81]	Incentive scheme relies on the transaction fees and the block creation rate is bounded by the network propagation delay	1. By enforcing the randomness in RTS at the consensus protocol level, for example, Sycomore [82] uses a sequence of transaction hashes to extend a chain in the DAG, 2. By having a fixed transaction fee in the network	Inclusive [34] SPECTRE [7] PHANTOM [50] Prism [35] GHOSTDAG [8]
Double-spend attack	Adversary tries to double-spend his transaction on two balanced branches of the DAG	Adversary controls or influences over a substantial portion of the network's computational power	By employing a conflict resolution mechanism or by the aid of a centralized authority, e.g., in Byteball [31], IOTA [83]	IOTA [84] SPECTRE [85]
Parasite-chain attack	Adversary tries to race and replace the honest subDAG by creating a secret subDAG to increase his profit	Adversary has enough computing power to generate its secret subDAG	Incentivise users for validating most recent tips; for example, Biased Random Walk (BRW) tip selection on IOTA Tangle	IOTA [84] SPECTRE [7]
Long range attack	Adversary creates an alternative DAG history starting from the genesis (e.g., by acquiring old private keys)	Adversary has enough computation power in PoW-based protocols or enough stake in PoS-based protocols	1. By employing time-dependent cryptographic primitive, e.g., Verifiable Delay Function (VDF) [86], 2. By having checkpoints in the consensus protocols	Possible in PoS-based Protocols
Equivocation attack	Adversary sends conflicting information to the nodes, e.g., sending conflicting messages m, m' for same sequence number in PBFT-EA [87]	Adversary has enough voting power to replicate different information (state) across nodes, there can also be a temporary network partition among nodes	1. By using trusted components (e.g., Intel SGX, AWS) at each node, 2. By employing slashing in the consensus protocol [88]	Hashgraph [20] Jointgraph [37] Cordial Miner [14]

SPECTRE [7], and Nano [41] also have partial order due to their pairwise ordering. Hence, from every arbitrary block, there is always a path to some blocks in the graph but not always to the genesis. Due to this partial order, these DAG-based consensus protocols are limited to cryptocurrency payments and don't support smart contract functionality.

Total order is achieved by sorting the blocks in a recursive manner. Sorting can be based on different parameters such as weight, votes, and timestamp. Some protocols, e.g., GHOST [6], PHANTOM [8], and DAG KNIGHT [9] run the total order as soon as a block is added to the DAG. However, some protocols, e.g., Tusk [12], Bullshark [13] wait for a certain time or rounds to compute a leader block and further with the help of the leader block, a total order is calculated. A few DAG-based systems, e.g., Jointgraph [37], Vite [32] employ trusted nodes that compute the total order.

MEV Attacks Ordering in a consensus protocol can be abused by Miner Extractable Value (MEV) attacks, where an adversary tries to include, exclude, or change the order of clients' transactions to maximize his profit. MEV was first introduced as a measure in Flash Boys 2.0 [89], later it was named as Maximal Extractable Value. These attacks are a major bottleneck in DLT applications in various use cases.

In major DLTs like Ethereum, the urgency for MEV protection is clear. There have been a few studies to countermeasure these MEV attacks. Yang et al. [90] present a SoK on MEV attacks and its countermeasures. A recent paper [91] presents important insights about MEV extraction on Ethereum through MEV bot bidding strategies where a bid is the highest bribe per computation. The authors exploited a machine learning model that forecasts the winning bids whilst increasing profitability.

MEV attacks can be readily executed on leader-based DAG protocols, yet they can also occur in unstructured DAG protocols where adversaries can act as block proposers. Fino [71] protocol integrates MEV protection into a BFT-based DAG protocol by employing k-out-of-n secret-sharing technique. Nasrulin et al. [92] present an accountable base layer protocol, Lø, which detects and mitigates transaction manipulations by creating a secure mempool of verifiable transactions.

B. Fairness

Fairness plays a critical role in consensus protocols within DLT systems, encompassing technical, economic, and social aspects. Unfair scenarios could breed discontent among participants, potentially hindering technology adoption. Hence, DAG-based protocols must prioritize fairness. This concept often involves accepting blocks from slower yet honest nodes, defined as those with low computing power in PoW-based protocols, low stake in PoS-based protocols, or significant communication delays. While this definition offers a broad understanding, fairness can be interpret in various ways.

While blockchain protocols struggles with fairness challenges stemming from centralized decision-making, DAG-based consensus protocols involve multiple nodes overseeing ledger state progression, potentially enhancing fairness. Protocols such as Bullshark [13], Coordicide [83] allow weak references to rescue older blocks, contributing to overall fairness.

Raikwar et al.'s recent work [93] explores fairness in DAG-based DLTs. They cover fairness for participants, consensus order, and DLT components, defining these notions and methods to achieve them. The authors also discuss how different DLT components and functions relate to these fairness notions.

C. Garbage Collection

In DAG-based consensus protocols, ensuring validity and fairness demands nodes to have unlimited memory, posing challenges for deploying these protocols. Due to the necessity for boundless memory, nodes can't garbage collect old data, risking the loss of honest but slower nodes' blocks. Garbage collection clashes with fairness, especially within an asynchronous network framework where blocks may experience indefinite delays. This creates a crucial trade-off between ensuring fairness and optimizing performance.

Garbage collection methods vary across different DAG-based consensus protocols, aiming to balance the need for memory optimization. The blocks of the DAG can be garbage collected based on their depth (or round), timestamp (or age), or finality. It can also be performed in DAG-based consensus protocols Vite [32], and Jointgraph [37] which maintain snapshot blocks (or chain). Consequently, older blocks can be garbage collected since their history is stored within the snapshot blocks. Jointgraph uses its snapshot and storage events to release the previous memory.

Narwhal [12] cleans up its DAG by discarding information up to a specific round from the genesis. However, this compromises fairness at the block level, risking disposal of slower nodes' data before proper ordering. In contrast, Bullshark [13] employs a garbage collection mechanism while maintaining fairness across all nodes. Bullshark practically handles nodes with limited memory, ensuring garbage collection and fairness post GST during synchronous periods.

Introducing garbage collection is a recent advancement in DAG-based protocols. Similar mechanisms to Bullshark's can potentially be adopted in other DAG-based protocols. Such mechanisms are vital for high-throughput DLTs, as their foundational data structures require pruning over time.

VI. Discussion

A. Tip Selection

The process of selecting and approving blocks when adding a new one is known as tip selection. This algorithm is crucial for shaping the DAG's structure and expansion, affecting security, consistency, and transaction confirmation time.

- **Uniform Random Tip Selection** To issue a new block, a node chooses tips to approve uniformly at random from all valid tips in the DAG until fixed k references are created, e.g., IOTA [21], [83]. Such tip selection fits in (eventually) synchronous, permissionless consensus protocols.
- **Round-based Tip Selection** To issue a new block at round r, a node chooses at least $n - f$ tips in the DAG from round $r - 1$, e.g., Aleph [10], Tusk [12], Bullshark [13], Cordial Miners [14]. This approach suits in partially synchronous/asynchronous network and permissioned models.
- **Parent + First-known Tip Selection** To issue a new block, a node selects its previous block as a (parent) tip along with a first-known tip. Hashgraph [20], Jointgraph [37], Nano [41], Vite [32], DLattice [94] employ such tip selection.
- **Heaviest Cluster Tip Selection** To issue a new block, a node selects all tips from the heaviest cluster, the largest subset of blocks in the DAG, in which each block is not comparable by the partial order with at most a constant number of other blocks, e.g., PHANTOM [50], DAG KNIGHT [9].
- **Sub-sampled Voting Tip Selection** To issue a new block, a node queries small random samples of other nodes for tips likely to get network approval, e.g., [22] and [83]. This method suits in permissionless DAG-based protocols.

B. Mempool

A Mempool protocol provides a vital abstraction layer, decoupling transaction dissemination from the consensus, and streamlining the ordering process for better efficiency. Notably, in distributed ledgers, the Atomic Broadcast facilitates batching [47], allowing a single ordering operation to handle multiple transactions. This has sparked significant interest and development in DAG-based Mempool protocols, starting from Narwhal mempool [12] to recent protocols [95], [96].

Gai et al. [96] introduced *Stratus*, a robust shared mempool (SMP) protocol designed to separate transaction distribution from consensus in leader-based BFT protocols. Unlike Narwhal, Stratus avoids quadratic message complexity due to reliable broadcast. It uses a distributed load balancing protocol, enabling each node to handle client transactions. Transaction ordering is managed by the leader using transaction identifiers. Stratus can be seamless integrated into existing DLT systems.

Camaioni et al. [95] introduced Chop Chop, a Byzantine atomic broadcast system addressing communication complexity challenges faced by contemporary systems like Narwhal. This discrepancy stems from traditional batching methods that demand each payload within a batch to carry specific data. Chop Chop employs atomic broadcast, leading to amortized costs for message ordering and aggregation using an innovative batching technique "distillation." A distilled batch comprises messages that can be quickly authenticated and duplicated.

C. Broadcast Techniques

Broadcast techniques in DAG-based consensus ensure reliable propagation of blocks, maintaining ledger integrity and consistency. Most protocols rely on underlying Reliable Broadcast (RB) protocols such as Bracha's [97] or Cachin and Tessaro's [64] to disseminate transactions. Recent DAG-based protocols have advanced broadcast techniques, outlined below.

- **Consistent Broadcast (CBC)** guarantees at transport level that a sender cannot send conflicting blocks to the correct nodes (equivocate). It employs a multi-step communication pattern and shares core properties in protocols, e.g., PBFT. In DAG-based consensus protocols [11]–[13], [71], CBC is treated as an abstract entity. Several layers of CBC are required to commit an individual backbone block, where each block in a DAG operates as its own CBC broadcast.
- **Byzantine Reliable Broadcast (BRB)** ensures the reliable dissemination of messages, even when a cetrain number of nodes in the system may be malicious or faulty. BRB requires at least $f+1$ honest nodes to unanimously agree on a value v before it's officially committed across all nodes. While BRB adds an extra step compared to CBC, DAG-based protocols, e.g., BBCA-LEDGER [52] using BRB exhibit low communication complexity and latency.
- **Cooperative Construction** facilitates transaction dissemination without relying on RB. In this approach, when a node intends to share a block, they simply transmit it to all other nodes, requiring only a single round of communication, as opposed to at least two rounds when utilizing RB. This approach is demonstrated in the Cordial Miner protocols [14].

D. Conflict Resolution

In DAG-based consensus, conflict resolution mechanisms determine the order and dependencies among conflicting units like transactions or blocks. These mechanisms decide how to agree on conflicting units, potentially rejecting or overlooking late blocks in favor of previously accepted ones.

The conflicting units can be 1) transactions, e.g., in IOTA [21], Graphchain [54], 2) blocks, e.g., in FANTOM [98], OHIE [38], 3) events, e.g., in Hashgraph [20], Jointgraph [37], 4) branch of transactions, e.g., in GHOST [6], Inclusive [34], 5) cluster, e.g., in GHOSTDAG [8], DAG KNIGHT [9].

Conflict resolution methods in DAG-based consensus vary based on parameters like weight, stake, vote, reputation, rank, and fee. Below, we outline several conflict resolution methods for conflicting units in the DAG.

- **Weight-based** Conflict resolution is performed by computing the cumulative weight and check for the valid unit using the weight, e.g., GHOSTDAG [8], DAG KNIGHT [9].
- **Vote-based** Conflict is resolved through the node casting the votes on the conflicting units and the unit with the most votes is considered valid, e.g.,Tusk [12], Bullshark [13].
- **Hash-based** Conflicts resolve through hashing the units, and the one with the higher (or lower, based on the protocol) hash is considered valid, e.g., Aleph [10], Blockmania [99].
- **First-seen** The first unit seen by a node is deemed valid, but a malicious node can intentionally delay the spread of harmful units, as observed in OHIE [38], Meshcash [43].

E. Latency and Throughput

Network latency and round trip time (RTT) are essential for consensus. Latency is the time for a packet to travel from point A to point B, while RTT encompasses the time for a packet to go from A to B and back, including encoding, queuing, processing, decoding, and propagation delays delays. These factors, typically consistent for a given pair of endpoints, can be affected by network congestion, adding variability to RTT. It's important to note that latency is not necessarily half of RTT due to potential differences in delays between endpoints.

Consistency-focused consensus protocols based on BFT incur high latency in asynchronous networks. Recent works aim to minimize the latency of these protocols. Spiegelman et al. [100] introduced Shoal, a protocol-agnostic framework that improves the latency of the BFT-based consensus protocols. Shoal [100] enhances the latency of Narwhal-based consensus protocols by employing leader reputation mechanism to prevent failures and by introducing pipelining to ensure a well-ordered DAG construction without timeout requirements.

Liu et al. [101] present a novel DAG-based commit rule that effectively reduces transaction latency in the UTXO model. Their novel commit rule accelerates transaction confirmation and reduces the confirmation latency. Another recent work [102] presents a flexible advancement in asynchronous BFT consensus protocols, bridging the ordering and agreement components and reduces the latency of the consensus.

The primary advantage of DAG-based protocols over blockchains is their enhanced throughput. Various DAG-based consensus protocols demonstrate high throughput, prompting ongoing research to further improve this aspect. Amores-Sesar and Cachin [103] present a construction that takes a DAG-based consensus protocol Π as input and outputs a new DAG-based protocol Π' which have superior throughput and latency.

F. Reward Mechanism

In distributed ledgers, the incentive structure plays a crucial role in ensuring fairness among participants by rewarding their contributions to the protocol's progress. Rewards, typically in the form of block rewards or transaction fees, incentivize active and foster participation and honest behavior from all nodes.

Many DAG-based consensus protocols do not have a reward mechanism, finding value in being "feeless" as seen in protocols, e.g., [84]. However, certain protocols employ reward mechanisms, such as Inclusive [34], Sphinx [104] to increase network participation, Graphchain [54] to maintain the DAG.

Various DAG-based consensus protocols employ diverse reward mechanisms tailored to incentivize specific behaviors among participants. G-IOTA [105] implements rewards for honest nodes executing correctly and penalizes adversaries for publishing conflicting transactions. SUI protocol [26] rewards validators based on their performance, as perceived by other validators using a stake distribution rule. Avalanche [55] adopts a multi-faceted approach: validators receive rewards proportional to their network security contribution tied to stake size, enhanced rewards for longer stake lock-ins, and incentives for sustained online activity and correct operations through proof-of-uptime and proof-of-correctness metrics.

G. Mathematical Models and Simulations

A mathematical model can analyze DAG-based consensus protocol metrics, but constructing a generic model to simulate and evaluate these protocols is a challenging task. Some attempts have been made to build such models and simulations.

S. Popov's initial mathematical model for a DAG-based protocol [84] assumed new message arrivals following a unique Poisson process, leading to a conjecture about the stationary rate of unapproved transactions. Simulations supporting this conjecture with homogeneous delays spurred interest in developing new mathematical models for DAG-based protocols, as seen in studies such as [106], [107]. Works like [108] explored non-Poisson message arrival scenarios, often assuming a central node managing ledger records with other nodes accessing the ledger state through it. Penzkofer et al. [109] extended Popov's model, introducing heterogeneous delays instead of homogeneous ones. Zander et al. introduced DAGsim [110], a multi-agent simulator based on a heterogeneous delay model for DAG-based protocols. Recent research [111] employed a discrete-time Markov chain to model the evolution of unapproved transactions under heterogeneous delay. Lin et al. developed TangleSim [112] to implement leaderless Tangle 2.0 in various network scenarios and byzantine environments.

For the simulation of deteministic protocols, Schett and Danezis [113] formalized a blockDAG protocol implementing a reliable point-to-point channel, embedding deterministic BFT protocols while maintaining safety and liveness properties. They utilized deterministic state machines for node communication, affirming properties claimed by Hashgraph [20], Blockmania [99], and Flare [114] within their framework. Attiya et al. [115] extended this work, faithfully simulating non-deterministic (Randomized) BFT protocols on blockDAG, capturing probabilistic guarantees and enabling analysis of protocols like Aleph [10], DAG-Rider [11], and Bullshark [13].

H. Lightweight DAG-based Protocols

DAG-based protocols offer enhanced performance and scalability compared to blockchains but face critical challenges, particularly limited storage scalability due to data redundancy. This necessitates the development of lightweight protocols to unlock diverse applications like Vehicular Social Networks (VSN) and Internet-of-Things (IoT). Yang et al. [116] introduced LDV, a robust and efficient lightweight DAG-based blockchain for resource-constrained VSNs. LDV optimizes data storage by selectively retaining pertinent information and pruning historical data, catering to vehicles with limited resources. Similarly, Cherupally et al. [117] proposed LSDI, a scalable DAG-based ledger for IoT data integrity verification in cloud-based IoT architectures. LSDI strategically prunes outdated segments to enhance its lightweight functionality.

In a recent work, Dai et al. [118] introduced GeckoDAG, a lightweight DAG-based blockchain, targeting data redundancy issues. Addressing account and reference redundancy, GeckoDAG consolidates transactions into compact unions, eliminating redundant account-related data, and introduces *reference override* to manage reference redundancy efficiently. GeckoDAG cuts storage needs without compromising security.

VII. Future Research

The wide variety of DAG-based protocols demands a multifaceted conclusion, underscoring performance, security, and applicability. In the following, we present an overview of emerging discussion points and future research questions.

Performance Evaluation: No standardized benchmarks exist for evaluating DAG-based consensus protocol performance. Research papers often focus on specific metrics, emphasizing protocol benefits. Developing comprehensive, standardized performance metrics is crucial for future advancements, allowing objective evaluation of whether DAG-based approaches mitigate performance bottlenecks in chain-based protocols.

Security Analysis: Rigorous formal security proofs for DAG-based protocols are offered sporadically, and the depth and scope of such proofs vary significantly. A comprehensive and objective framework is imperative for understanding each protocol's theoretical foundations and constraints. Such a framework would enhance our grasp of their relative strengths, weaknesses, and an informed comparison of security features.

Consensus Protocol Maturity: Research on DAG-based consensus should prioritize understanding their real-world performance and deriving lessons from practical deployments. Evaluating the effectiveness of testing protocols and extracting insights from actual implementations across diverse settings is essential. Empirical findings play a critical role in refining these mechanisms and establishing their significance in DLTs.

Modular Trade-offs: Future research on DAG-based consensus should focus on identifying crucial components and modules, understanding their operational framework and associated trade-offs. Emphasizing a modular approach, research should analyze interdependencies and interactions between protocol parts. This analysis will enhance comprehension of how design choices within one module influence overall system performance, security, and scalability, guiding the development of more efficient and robust DAG-based DLTs.

Fairness: Future research on fairness in DAG-based protocols should explore equitable transaction processing dynamics to prevent undue influence by any node. This involves investigating bias prevention in writing access to maintain decentralization. Understanding how the consensus mechanism affects fairness is vital for establishing equitable relationships among protocol users. Additionally, examining the interplay between fairness, decentralization, and network trust is crucial for ensuring the stability and integrity of the underlying DLT.

VIII. Conclusion

In this Systematization of Knowledge (SoK), our comprehensive exploration centred on DAG-based consensus protocols, categorising them by availability and consistency constraints. We provided an overview of the essential models and components for designing these protocols. Our study presented a detailed analysis of various protocols, delineating their fundamental functionalities, security properties, potential attacks, and corresponding countermeasures. The examination encompassed critical components, including tip selection, broadcast techniques, reward mechanisms, garbage collection, mempool management, and fairness considerations.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[2] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," *Cryptology ePrint Archive*, 2015.

[3] S. Wu, P. Wei, R. Zhang, and B. Jiang, "Security-performance tradeoff in dag-based proof-of-work blockchain protocols," *Cryptology ePrint Archive*, 2023.

[4] NeonVest, "The scalability trilemma in blockchain," 2018, accessed: 23.08.2023. [Online]. Available: https://aakash-111.medium.com/the-scalability-trilemma-in-blockchain-75fb57f646df

[5] J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, and D. Gligoroski, "Databases fit for blockchain technology: A complete overview," *Blockchain: Research and Applications*, p. 100116, 2022.

[6] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer, 2015, pp. 507–527.

[7] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *Cryptology ePrint Archive*, 2016.

[8] Y. Sompolinsky, S. Wyborski, and A. Zohar, "Phantom ghostdag: a scalable generalization of nakamoto consensus: September 2, 2021," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, 2021, pp. 57–70.

[9] Y. Sompolinsky and M. Sutton, "The dag knight protocol: A parameterless generalization of nakamoto consensus," *Cryptology ePrint Archive*, 2022.

[10] A. Gągol, D. Leśniak, D. Straszak, and M. Świętek, "Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 214–228.

[11] I. Keidar, E. Kokoris-Kogias, O. Naor, and A. Spiegelman, "All you need is dag," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021, pp. 165–175.

[12] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and tusk: a dag-based mempool and efficient bft consensus," in *Proceedings of the Seventeenth European Conference on Computer Systems*, 2022, pp. 34–50.

[13] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: Dag bft protocols made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2705–2718.

[14] I. Keidar, O. Naor, and E. Shapiro, "Cordial miners: A family of simple, efficient and self-contained consensus protocols for every eventuality," *arXiv preprint arXiv:2205.09174*, 2022.

[15] D. Malkhi, C. Stathakopoulou, and M. Yin, "Bbca-chain: One-message, low latency bft consensus on a dag," *arXiv preprint arXiv:2310.06335*, 2023.

[16] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "Sok: Dag-based blockchain systems," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[17] B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, and A. Mezrioui, "Sok: a comprehensive survey on distributed ledger technologies," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2022, pp. 1–16.

[18] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, "Mind the gap: Trade-offs between distributed ledger technology characteristics," *arXiv preprint arXiv:1906.00861*, 2019.

[19] H. Y. Wu, X. Yang, C. Yue, H.-Y. Paik, and S. S. Kanhere, "Chain or dag? underlying data structures, architectures, topologies and consensus in distributed ledger technology: A review, taxonomy and research issues," *Journal of Systems Architecture*, vol. 131, p. 102720, 2022.

[20] L. Baird and A. Luykx, "The hashgraph protocol: Efficient asynchronous bft for high-throughput distributed ledgers," in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 2020, pp. 1–7.

[21] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, and H. Moog, "Tangle 2.0 leaderless nakamoto consensus on the heaviest dag," *IEEE Access*, vol. 10, pp. 105 807–105 842, 2022.

[22] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless bft consensus through metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[23] M. M. T. Chakravarty, J. Chapman, K. MacKenzie, O. Melkonian, M. Peyton Jones, and P. Wadler, "The Extended UTXO Model," in *Financial Cryptography and Data Security*, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 525–539.

[24] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[25] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, and H. Moog, "Reality-based utxo ledger," *Distrib. Ledger Technol.*, vol. 2, no. 3, sep 2023. [Online]. Available: https://doi.org/10.1145/3616022

[26] The Mysten Labs Team, "The sui smart contracts platform," 2022, accessed: 22.08.2023. [Online]. Available: https://github.com/MystenLabs/sui/blob/main/doc/paper/sui.pdf

[27] A. Lewis-Pye and T. Roughgarden, "Permissionless consensus," *arXiv preprint arXiv:2304.14701*, 2023.

[28] J. Neu, E. N. Tas, and D. Tse, "The availability-accountability dilemma and its resolution via accountability gadgets," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 541–559.

[29] R. Pass and E. Shi, "The sleepy model of consensus," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*. Springer, 2017, pp. 380–409.

[30] G. Bracha, "Asynchronous byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, 1987.

[31] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, p. 11, 2016.

[32] C. Liu, D. Wang, and M. Wu, "Vite: A high performance asynchronous decentralized application platform," *White Paper*, 2018.

[33] IOTA Foundation, "Consensus on a dag," 2023, accessed: 19.11.2023. [Online]. Available: https://wiki.iota.org/learn/protocols/iota2.0/core-concepts/consensus/introduction/

[34] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer, 2015, pp. 528–547.

[35] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 585–602.

[36] C. Li, F. Long, and G. Yang, "Ghast: Breaking confirmation delay barrier in nakamoto consensus via adaptive weighted blocks," 2020.

[37] F. Xiang, W. Huaimin, S. Peichang, O. Xue, and Z. Xunhui, "Jointgraph: A dag-based efficient consensus algorithm for consortium blockchains," *Software: Practice and Experience*, vol. 51, no. 10, pp. 1987–1999, 2021.

[38] H. Yu, I. Nikolić, R. Hou, and P. Saxena, "Ohie: Blockchain scaling made simple," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 90–105.

[39] M. Fitzi, P. Ga, A. Kiayias, and A. Russell, "Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition," *Cryptology ePrint Archive*, 2018.

[40] X. Wang, S. Azouvi, and M. Vukolić, "Security analysis of filecoin's expected consensus in the byzantine vs honest model," *arXiv preprint arXiv:2308.06955*, 2023.

[41] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," *Nano [Online resource]. URL: https://nano. org/en/whitepaper (date of access: 24.03. 2018)*, vol. 16, p. 17, 2018.

[42] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb: A proof-of-work parallel-chain architecture for massive throughput," *Chainweb whitepaper*, vol. 19, 2018.

[43] I. Bentov, P. Hubáček, T. Moran, and A. Nadler, "Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies," in *Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8–9, 2021, Proceedings 5*. Springer, 2021, pp. 114–127.

[44] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[45] E. A. Brewer, "Towards robust distributed systems," in *PODC*, vol. 7, no. 10.1145. Portland, OR, 2000, pp. 343 477–343 502.

[46] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic*

Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37. Springer, 2018, pp. 66–98.

[47] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[48] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[49] L. Baird, M. Harmon, and P. Madsen, "Hedera: A governing council & public hashgraph network," *The trust layer of the internet, whitepaper*, vol. 1, pp. 1–97, 2018.

[50] Y. Sompolinsky and A. Zohar, "Phantom," *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.

[51] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 515–528.

[52] C. Stathakopoulou, M. Wei, M. Yin, H. Zhang, and D. Malkhi, "Bbca-ledger: High throughput consensus meets low latency," *arXiv preprint arXiv:2306.14757*, 2023.

[53] K. Babel, A. Chursin, G. Danezis, L. Kokoris-Kogias, and A. Sonnino, "Mysticeti: Low-latency dag consensus with fast commit path," 2023.

[54] X. Boyen, C. Carr, and T. Haines, "Graphchain: A blockchain-free scalable decentralised ledger," in *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, 2018, pp. 21–33.

[55] S. Buttolph, A. Moin, K. Sekniqi, and E. G¨un Sirer, "Avalanche native token ($avax) dynamics," 2020, accessed: 14.11.2023. [Online]. Available: https://assets.website-files.com/5d80307810123f5ffbb34d6e/6008d7bc56430d6b8792b8d1_Avalanche%20Native%20Token%20Dynamics.pdf

[56] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM symposium on principles of distributed computing*, 2017, pp. 315–324.

[57] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "{Bitcoin-NG}: A scalable blockchain protocol," in *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, 2016, pp. 45–59.

[58] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," in *Progress in Cryptology–LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20–22, 2017, Revised Selected Papers 5*. Springer, 2019, pp. 327–351.

[59] W. V. Gehrlein, "Condorcet's paradox," *Theory and Decision*, vol. 15, no. 2, pp. 161–197, 1983.

[60] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.

[61] J. Neu, E. N. Tas, and D. Tse, "Ebb-and-flow protocols: A resolution of the availability-finality dilemma," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 446–465.

[62] S. Popov and S. Müller, "Voting-based probabilistic consensuses and their applications in distributed ledgers," *Annals of Telecommunications*, vol. 77, no. 1, pp. 77–99, 2022. [Online]. Available: https://doi.org/10.1007/s12243-021-00875-7

[63] I. Amores-Sesar, C. Cachin, and E. Tedeschi, "When is spring coming? a security analysis of avalanche consensus," 2022.

[64] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Annual International Cryptology Conference*. Springer, 2001, pp. 524–541.

[65] A. Lewis-Pye, O. Naor, and E. Shapiro, "Flash: An asynchronous payment system with good-case linear communication complexity," *arXiv preprint arXiv:2305.03567*, 2023.

[66] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography," in *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, 2000, pp. 123–132.

[67] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, p. 374–382, apr 1985. [Online]. Available: https://doi.org/10.1145/3149.214121

[68] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 824–840, 1985.

[69] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable byzantine fault-tolerant services," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 59–74, 2005.

[70] B. Ford, "Threshold logical clocks for asynchronous distributed coordination and consensus," *arXiv preprint arXiv:1907.07010*, 2019.

[71] D. Malkhi and P. Szalachowski, "Maximal extractable value (mev) protection on a dag," *arXiv preprint arXiv:2208.00940*, 2022.

[72] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 281–310.

[73] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2017, pp. 643–673.

[74] S. Galimberti and M. Potop-Butucaru, "Game theoretical analysis of dag-ledgers backbone," *Cryptology ePrint Archive*, 2023.

[75] R. Zhang, D. Zhang, Q. Wang, S. Wu, J. Xie, and B. Preneel, "Nc-max: Breaking the security-performance tradeoff in nakamoto consensus," *Cryptology ePrint Archive*, 2020.

[76] D. Camargo, A. Penzkofer, S. Müller, and W. Sanders, "Mitigation of liveness attacks in dag-based ledgers," *arXiv preprint arXiv:2305.01207*, 2023.

[77] M. Raikwar and D. Gligoroski, "Dos attacks on blockchain ecosystem," in *European Conference on Parallel Processing*. Springer, 2021, pp. 230–242.

[78] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," *Massachusetts Institute of Technology, Laboratory for Computer Science*, 1996.

[79] S. Khattak, T. Elahi, L. Simon, C. M. Swanson, S. J. Murdoch, and I. Goldberg, "Sok: Making sense of censorship resistance systems," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 37–61, 2016.

[80] K. Kostiainen, S. Gnap, and G. Karame, "Censorship-resilient and confidential collateralized second-layer payments," *Cryptology ePrint Archive*, 2022.

[81] M. Perešíni, I. Homoliak, F. M. Benčić, M. Hrubỳ, and K. Malinka, "Incentive attacks on dag-based blockchains with random transaction selection," *arXiv preprint arXiv:2305.16757*, 2023.

[82] E. Anceaume, A. Guellier, R. Ludinard, and B. Sericola, "Sycomore: A permissionless distributed ledger that self-adapts to transactions demand," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2018, pp. 1–8.

[83] S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer *et al.*, "The coordicide," *Accessed Jan*, pp. 1–30, 2020.

[84] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, p. 30, 2018.

[85] L. Kovalchuk, M. Rodinko, and R. Oliynykov, "Upper bound probability of double spend attack on spectre," in *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 2020, pp. 18–22.

[86] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual international cryptology conference*. Springer, 2018, pp. 757–788.

[87] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, "Attested append-only memory: Making adversaries stick to their word," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 189–204, 2007.

[88] M. Reinhart, "An overview of polkadot slashing," 2021, accessed: 30.11.2023. [Online]. Available: https://blog.unit410.com/polkadot/kusama/slashing/validator/2021/08/05/polkadot-slashing.html

[89] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.

[90] S. Yang, F. Zhang, K. Huang, X. Chen, Y. Yang, and F. Zhu, "Sok: Mev countermeasures: Theory and practice," *arXiv preprint arXiv:2212.05111*, 2022.

[91] C. Raun, B. Estermann, L. Zhou, K. Qin, R. Wattenhofer, A. Gervais, and Y. Wang, "Leveraging machine learning for bidding strategies in miner extractable value (mev) auctions," Cryptology ePrint Archive, Paper 2023/1281, 2023. [Online]. Available: https://eprint.iacr.org/2023/1281

[92] B. Nasrulin, G. Ishmaev, J. Decouchant, and J. Pouwelse, "Lo: An accountable mempool for mev resistance," in *Proceedings of the 24th International Middleware Conference*, ser. Middleware '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 98–110. [Online]. Available: https://doi.org/10.1145/3590140.3629108

[93] M. Raikwar, N. Polyanskii, and S. Müller, "Fairness notions in dag-based dlts," in *2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2023, pp. 1–8.

[94] T. Zhou, X. Li, and H. Zhao, "Dlattice: A permission-less blockchain based on dpos-ba-dag consensus for data tokenization," *IEEE Access*, vol. 7, pp. 39 273–39 287, 2019.

[95] M. Camaioni, R. Guerraoui, M. Monti, P.-L. Roman, M. Vidigueira, and G. Voron, "Chop chop: Byzantine atomic broadcast to the network limit," *arXiv preprint arXiv:2304.07081*, 2023.

[96] F. Gai, J. Niu, I. Beschastnikh, C. Feng, and S. Wang, "Scaling blockchain consensus via a robust shared mempool," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 530–543.

[97] G. Bracha, "An asynchronous [(n-1)/3]-resilient consensus protocol," in *Proceedings of the third annual ACM symposium on Principles of distributed computing*, 1984, pp. 154–162.

[98] S.-M. Choi, J. Park, Q. Nguyen, and A. Cronje, "Fantom: A scalable framework for asynchronous distributed systems," *arXiv preprint arXiv:1810.10360*, 2018.

[99] G. Danezis and D. Hrycyszyn, "Blockmania: from block dags to consensus," *arXiv preprint arXiv:1809.01620*, 2018.

[100] A. Spiegelman, B. Aurn, R. Gelashvili, and Z. Li, "Shoal: Improving dag-bft latency and robustness," *arXiv preprint arXiv:2306.03058*, 2023.

[101] K. Liu, M. Jourenko, and M. Larangeira, "Reducing latency of dag-based consensus in the asynchronous setting via the utxo model," *arXiv preprint arXiv:2307.15269*, 2023.

[102] S. Liu, W. Xu, C. Shan, X. Yan, T. Xu, B. Wang, L. Fan, F. Deng, Y. Yan, and H. Zhang, "Flexible advancement in asynchronous bft consensus," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 264–280.

[103] I. Amores-Sesar and C. Cachin, "We will dag you," *arXiv preprint arXiv:2311.03092*, 2023.

[104] Q. Wang and R. Li, "A weak consensus algorithm and its application to high-performance blockchain," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[105] G. Bu, Ö. Gürcan, and M. Potop-Butucaru, "G-iota: Fair and confidence aware tangle," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 644–649.

[106] B. Kusmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the tangle for uniform random and random walk tip selection," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 228–236.

[107] S. Park, S. Oh, and H. Kim, "Performance analysis of dag-based cryptocurrency," in *2019 IEEE International Conference on Communications workshops (ICC workshops)*. IEEE, 2019, pp. 1–6.

[108] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu, "Direct acyclic graph-based ledger for internet of things: Performance and security analysis," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1643–1656, 2020.

[109] A. Penzkofer, O. Saa, and D. Dziubałtowska, "Impact of delay classes on the data structure in iota," in *International Workshop on Data Privacy Management*. Springer, 2021, pp. 289–300.

[110] M. Zander, T. Waite, and D. Harz, "Dagsim: Simulation of dag-based distributed ledger protocols," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 118–121, 2019.

[111] N. Kumar, A. Reiffers-Masson, I. Amigo, and S. Ruano Rincón, "The effect of network delays on distributed ledgers based on direct acyclic graphs: A mathematical model," *Available at SSRN 4253421*, 2022.

[112] B.-Y. Lin, D. Dziubałtowska, P. Macek, A. Penzkofer, and S. Müller, "Tanglesim: An agent-based, modular simulator for dag-based distributed ledger technologies," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2023, pp. 1–5.

[113] M. A. Schett and G. Danezis, "Embedding a deterministic bft protocol in a block dag," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021, pp. 177–186.

[114] N. S. Rowan and N. Usher, "The flare consensus protocol: Fair fast federated byzantine agreement consensus," Tech. rep, Tech. Rep, Tech. Rep., 2019.

[115] H. Attiya, C. Enea, and S. Nassar, "Faithful simulation of randomized bft protocols on block dags," *Cryptology ePrint Archive*, 2023.

[116] W. Yang, X. Dai, J. Xiao, and H. Jin, "Ldv: A lightweight dag-based blockchain for vehicular social networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5749–5759, 2020.

[117] S. R. Cherupally, S. Boga, P. Podili, and K. Kataoka, "Lightweight and scalable dag based distributed ledger for verifying iot data integrity," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 267–272.

[118] X. Dai, Y. Zhou, J. Xiao, F. Cheng, X. Xie, H. Jin, and B. Li, "Geckodag: Towards a lightweight dag-based blockchain via reducing data redundancy," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 451–462.