Scaling Byzantine Consensus: A Broad Analysis

Christian Berger
University of Passau
Passau, Germany
cb@sec.uni-passau.de

Hans P. Reiser
University of Passau
Passau, Germany
hr@sec.uni-passau.de

ABSTRACT

Blockchains and distributed ledger technology (DLT) that rely on Proof-of-Work (PoW) typically show limited performance. Several recent approaches incorporate Byzantine fault-tolerant (BFT) consensus protocols in their DLT design as Byzantine consensus allows for increased performance and energy efficiency, as well as it offers proven liveness and safety properties. While there has been a broad variety of research on BFT consensus protocols over the last decades, those protocols were originally not intended to scale for a large number of nodes. Thus, the quest for scalable BFT consensus was initiated with the emerging research interest in DLT. In this paper, we first provide a broad analysis of various optimization techniques and approaches used in recent protocols to scale Byzantine consensus for large environments such as BFT blockchain infrastructures. We then present an overview of both efforts and assumptions made by existing protocols and compare their solutions.

CCS CONCEPTS

• **General and reference** → **Surveys and overviews**; • **Computing methodologies** → *Distributed algorithms*; • **Computer systems organization** → Fault-tolerant network topologies;

KEYWORDS

Scalability, Byzantine Fault Tolerance, Consensus, Blockchain, Distributed Ledger Technologies

ACM Reference Format:
Christian Berger and Hans P. Reiser. 2018. Scaling Byzantine Consensus: A Broad Analysis. In *2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL'18), December 10–14, 2018, Rennes, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3284764.3284767

1 INTRODUCTION

In the past years, distributed ledger technology (DLT) has gained a tremendous growth in popularity. Originally, the Bitcoin [26] blockchain and its Proof-of-Work (PoW) mechanism were invented with the ambition to create the foundation of a secure and decentralized financial accounting system of confirming transactions in a peer-to-peer fashion and more use cases have followed since then. DLT infrastructures share the need to achieve an agreement about which block they append to the ledger. For this purpose, they employ a mechanism that ideally should work even in a large-scale and widely spread environment like the Internet. Proof-of-Work is very successful in providing open membership by securing the network against Sybil attacks [12] as it couples participation – to be more concrete the probability of the node to decide the next block – to resource consumption, making it very expensive for any attacker to harm the infrastructure. Also, PoW scales well for a huge number of nodes. However, the PoW scheme has *inherent* flaws, to be precise PoW (i) amounts for a vast waste of energy and resources (ii) typically performs very poorly in terms of latency and throughput and (iii) does not guarantee consensus finality [32], meaning a decided block is still possibly subject to future change.

In order to circumvent the inherent drawbacks of the PoW mechanism, Byzantine fault-tolerant (BFT) consensus – used in well understood protocols like PBFT [10] – can be employed as an alternative for ordering transactions in DLTs [4]. BFT algorithms can be used as a variant of *Proof-of-Stake* [7], in which nodes can be randomly assigned the right to propose blocks and the agreement about which block is being appended to the blockchain is canonical among all nodes [7] thus guaranteeing consensus finality.

Furthermore, the weak spots of PoW are exactly the strong points of traditional BFT consensus protocols as they (i) work energy efficiently (ii) can achieve high performance in the range of tens of thousands transactions per second with a latency of only some seconds (or even less) and (iii) provide proven liveness and safety properties. On the downside, traditional BFT protocols lack PoW's support for open membership and scalability for a large number of up to thousands of nodes. We can argue that there is quite justification for the existence of *permissioned* blockchains or DLTs may explore other methods to ensure membership, however scalability remains a big challenge for large blockchain infrastructures.

1.1 Related Work and Contribution

A few systematization of knowledge papers have analyzed blockchain protocols: A 2015 comparison between existing PoW and BFT based blockchain protocols investigates on overcoming scalability limits [32] and a perspective that analyzes design spaces for increasing performance of PoW blockchains has been presented [11]. Also, a systematic and comprehensive study of consensus in blockchain protocols [2] evaluates performance, security and design properties, also addressing the scalability aspect. In this paper, we present a detailed survey that focuses on *techniques* recent approaches incorporate to improve scalability. Thus, this paper summarizes contemporary research and provides a systematic analysis of ongoing efforts. In Section 2, we identify and categorize various techniques for scaling Byzantine consensus. Subsequently, in Section 3, we compare recent BFT blockchain protocols in terms of their assumptions and goals. Finally, Section 4 concludes our work.

SERIAL'18, December 10–14, 2018, Rennes, France
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6110-1/18/12…$15.00
https://doi.org/10.1145/3284764.3284767

2 SCALING BYZANTINE CONSENSUS

Scaling Byzantine consensus is an ongoing effort that requires exploration, advancement and combination of several methods and approaches. We identified the following questions targeted to increase the scalability (and efficiency) of Byzantine consensus:

- Who needs to communicate? Does the entire consortium actively take part in achieving agreement or does a (or multiple) flexibly selected *representative committee(s)* decide?
- How does the communication flow? Increasing the efficiency of communication might require a suitable *communication topology* e.g. hierarchical or overlay / gossip-based.
- How can transactions be ordered and committed *in parallel*?
- Which new ways can be paved by employing *suitable cryptographic primitives*?
- How can we move mission-critical steps of consensus towards hardware using *trusted hardware components*?

Our categorization of the ongoing efforts is shown in Figure 1.

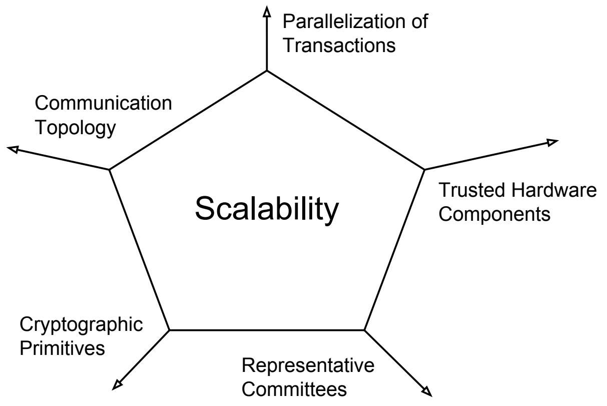
Parallelization of Transactions
Communication Topology
Scalability
Trusted Hardware Components
Cryptographic Primitives
Representative Committees

Figure 1: The different directions of approaches for scaling Byzantine consensus.

2.1 Communication Topology

Improving the efficiency of the communication flow means avoiding the emergence of bottlenecks and thus to distribute the communication load as evenly as possible. To serve as an example, the well-known PBFT protocol does not scale well because of its all-to-all broadcast phases and the high burden that it puts on the leader who needs to propose messages of large size to all nodes.

Flat Communication. A first and naive step to reduce the communication complexity e.g. of a PBFT-like protocol is to let the leader collect and distribute messages [16, 33] which requires the use of signatures instead of message authentication codes. By abstaining from all-to-all direct communication, the number of messages in the network decreases from $O(n^2)$ to $O(n)$. However, this still imposes a huge burden on the leader, making him the bottleneck as he still has to communicate directly with all other nodes and verify the signatures of their messages. A subsequent step is thus to efficiently aggregate multiple signatures into a single one, which we will discuss in Section 2.2.

Tree Communication. A speculative optimization technique is to rearrange the communication flow in a balanced tree where the leader is at the root position, non-leaf nodes forward messages to their children top-down and leaf nodes initiate a reply phase bottom-up ideally with the leader obtaining the collection of all replies. This strategy is adopted by ByzCoin [16] which employs such a communication tree in combination with *collective signatures* [31] (see Section 2.2). Scalability arises especially because every node only receives the aggregated $O(1)$-size rather than $O(n)$-size message and needs only $O(1)$ rather than $O(n)$ computation effort by verifying only the collective signature instead of n individual signatures in the commit phase[1] [16]. However, this approach alone can not guarantee liveness, as an adversary might have malicious nodes at the higher positions in the tree, thus cutting of (subtrees of) correct nodes and breaks the liveness property with less than f malicious nodes. ByzCoin circumvents this issue by making the leader capable of detecting such an attack and providing a costly, less-efficient fall-back mechanism to a flat communication scheme in which he becomes the bottleneck.

Overlay Networks and Gossip. Since leader-based BFT consensus protocols tend to suffer from the high communication burden of broadcasts, for instance when the leader proposes a block of ordered transactions to all other nodes, an interesting approach insists on relieving the communication efforts of the leader by disseminating messages using gossip over an application layer overlay network – an idea recently incorporated by Algorand [13] or Gosig [19]. While the leader initiates the sending of a propose message m to k other, randomly chosen nodes, at every hop a node that receives m will forward it to randomly chosen k of its connected neighbors, hence spreading m like a rumor with high reliability in the network. The propagation speed depends on the fanout parameter k and the overall communications burden is moved away from the leader to a more fairly distribution with nodes communicating only with their $O(1)$ neighbors, thus improving scalability. However, since gossip is a technique that makes use of randomness, protocols that rely on it tend to provide probabilistic guarantees.

Leader-less Communication. Another approach to avoid the leader bottleneck is introduced by *Snowflake to Avalanche* a novel, leader-less consensus protocol family [27]. Avalanche's core idea relies in a *metastable mechanism* which is inspired by gossip algorithms: In the protocol, each node queries k randomly chosen other nodes in a loop while adopting to the value that is being replied by an adjusted majority of nodes so that eventually, correct nodes are being guided towards the same consensus value. In this *probabilistic* approach, nodes will – even in a large network – quickly converge to an irreversible state, but without guarantee. Note, that also *deterministic* leader-free BFT consensus protocols exist [5].

Federated Byzantine Agreement. While in traditional BFT protocols like PBFT [10] consensus is defined by the assumption of *equitable and evenly trusted and system-wide accepted* nodes with the property that *any* $2f + 1$ nodes out of $3f + 1$ can convince *any* correct node to commit to a proposed ordering of transactions, the concept of consensus is generalized in the Federated Byzantine Agreement (FBA) model [22], where any node v_i accepts transactions based

[1]Nodes need to verify that a super majority, typically $2f + 1$, signed the prepare phase. We provide more details of the CoSi cryptographic primitive in section 2.2.1.

on its individual trust decisions. These decisions are refined and expressed by *quorum slices*. A quorum slice is a subset of a quorum which will convince at least one particular node (e.g. v_i) to commit to a specific value for a given slot. Nodes can be in multiple quorum slices at the same time and they need to carefully choose their quorum slices: To ensure *agreement*, the FBA system must guarantee *quorum intersection* [22], a property that states that any two of its quorums share at least one node. To ensure progress in presence of faulty nodes, quorum slices must overlap in a way that quorum intersection still holds even if certain groups of nodes fail (they are characterized by *dispensable sets* [22]). In FBA, scalability depends on the concrete structure of the dependencies and trust relations formed by the consensus slices and quorum intersections. It is advised [22] to build a tiered, hierarchical consensus structure, in which nodes form a consensus slice containing itself and a specific number of arbitrary nodes out of the upper adjacent layer. While such an approach reduces communication efforts compared to traditional BFT protocols, fault tolerance does not linearly scale with the total number of system nodes anymore – at least from the perspective of an individual node – thus yielding a scalable and open system, in which however a few bad trust decisions can block a single correct node from reaching agreement with the federation.

2.2 Cryptographic Primitives

Cryptographic primitives play a key role in scaling BFT protocols as they can be used for several scaling approaches, presented below.

Collective Signatures. CoSi [31] is a cosigning protocol which efficiently supports large-scale groups of witnesses where an authoritative statement (e.g. leader propose message) is validated and publicly logged by the witnesses as well as signed by the authoritative and all or most of witnesses. The CoSi protocol achieves scalability by constructing a Schnorr multisignature [29] over a spanning tree in four communication phases (which are essentially two top-down-then-bottom-up runs over the tree) [31].

The CoSi protocol can serve as building block for scalable BFT blockchain protocols as it can efficiently compress hundreds of individual signatures into a single signature of almost the same size. However, in the Byzantine setting, the *use of this protocol is speculative* and will only improve optimistic performance as faulty nodes may not forward messages to their children. As ByzCoin [16] shows, the leader only collects a $O(1)$-size rather than $O(n)$-size message and has less verification effort since he only needs to verify the single collective signature instead of n individual signatures, hence sparing both computation and network resources.

Threshold Signatures. In recent work [1, 14, 33] it also has been shown that the use of threshold signatures [30] can reduce communication costs of some BFT protocols by a factor of $O(n)$ if applied correctly. Like multi-signatures, threshold signatures compress the size of multiple signatures to that of a single one with the predicate that for a (t, n)-*threshold signature* at least t participants out of n are required to create the valid signature. For instance, PBFT [10] uses commit certificates of size $O(n)$ to prove that at least $2f + 1$ nodes support a commit decision. Using $t = 2f + 1$ threshold signatures, these certificates can be aggregated into a single commit certificate of size $O(1)$, making the protocol more scalable.

Threshold Encryption. In a (t, n) threshold cryptography scheme, at least t out of n total parties need to work together in order to decrypt an encrypted message. In the BFT adversary model, setting $t = f + 1$ implies that an attacker can not read encrypted messages as long as no correct node reveals its decryption share.

In HoneyBadgerBFT [24], threshold encryption is used to optimize the asynchronous common subset (ACS) primitive[2]. In particular, as the costs of ACS depend on the size of the individual nodes' transaction sets, an improved batching strategy relies in letting the nodes propose mostly disjoint sets of transactions using randomly chosen samples from their queues [24]. Ideally, each transaction is only proposed by a single node, hence improving communication efficiency. However, as ACS allows for an adversary to choose which nodes' proposals are being included, he can easily censor transactions to his liking [24]. This is being prevented using threshold encryption to encrypt transactions before the agreement – the agreement phase is being run on cipher texts, thus the adversary can not figure out which node proposes which transactions until after agreement [24]. With this improved batching strategy, HoneyBadgerBFT achieves an optimized communication complexity of $O(n)$ bits per transaction for *asynchronous* atomic broadcast which is better by the factor of $O(n)$ than existing approaches [3, 8], hence greatly improving scalability. On a side-note, in HoneyBadgerBFT's design, ACS is then reduced to reliable broadcast (RBC) and the efficiency of RBC is improved using *erasure codes* [9].

Verifiable Random Functions. The ambition of verifiable random functions (VRFs) [23] is to choose a selection of all system actors in a random and secret way without the actors needing to interact with each other. VRFs can be used to implement *cryptographic sortition* [13], an algorithm for choosing such a subset of actors in a way that considers weights, e.g. in a system with n users, for user i with weight w_i the probability for him being chosen is proportional to $\frac{w_i}{\sum_{j=0}^{n-1} w_j}$. The seed for generating the randomness can be derived from a public source provided by the blockchain. Algorand [13] uses VRFs so that every user in the system can check if he is chosen to participate in a special committee only by taking his private key and the seed as input while generating proof of his membership and role as output. Scalability arises from the fact that (i) the committee is chosen by VRFs in a non-interacting way without any excessive communication, even independent from the total size of the system and (ii) only committee members actively participate in the consensus protocol (which requires exchanging messages) while the others only learn the agreed value. Furthermore, the privacy, randomness and non-interactiveness of the selection process protects committee members from being the target of attacks [13].

2.3 Representative Committees

A key idea for achieving scalable consensus in large-scale systems is to announce a committee of delegates with active roles (e.g. proposers and acceptors) while a large portion of the nodes stay passive e.g. they only learn about the agreed value. This committee can be best thought of as a representative and random selection of actors that communicate with each other to reach agreement. Because the

[2]The ACS primitive is then used to build an atomic broadcast primitive by letting nodes propose a subset of the transactions stored in their queue and output the union of transactions as the agreed-upon vector [24].

	ByzCoin	**FastBFT**	**Stellar**	**HoneyBadgerBFT**	**Algorand**	**Gosig**	**OmniLedger**
Scalability (evaluated with)	1004	199	currently running ca 100	104	**up to 500k**	up to 10k	1800
Throughput (transactions/s)	700 (n=1004)	370 (n=199)	1000 (n ca. 100)	1200 (n=104)	<1000	4000 (n=140)	**≥ 4000 (n=1800)**
Latency	30s	**< 1s (1 Gbps LAN)**	few seconds	100s	1 minute	<1 minute	< 2s
Synchrony	weakly synchronous	weakly synchronous	asynchronous, but progress depends on synchrony	**asynchronous**	weakly synchronous	asynchronous, but provable liveness only under weak synchrony	synchronous
Consensus determinism	deterministic	deterministic	deterministic	probabilistic	probabilistic	probabilistic	probabilistic
Approaches for scaling consensus	communication tree + collective signatures	hardware-based TEE + secret sharing, tree topology	federal Byzantine agreement with hierarchical structure	novel ACS reduction with threshold encryption, efficient RBC with erasure codes	committee (cryptographic sortition) + gossip	multi-signatures + gossip	communication tree, collective signatures, parallelizing transactions

Table 1: Comparison of existing, scalable BFT protocols designated for large-scale blockchain infrastructures.

decision of the delegates will be binding for all, the overall system reaches agreement as the decision is being propagated through the network. Since the committee has a fixed size, this procedure can scale very well, in fact being even almost independent from the total number of nodes and work safely as long as some important conditions are satisfied [13]: (i) The selection process should not require coordination among the nodes and (ii) delegates are chosen randomly, but in a representative way, e.g. by using stake-based weighted probabilities so that an attacker can not launch a Sybil attack and (iii) delegates are chosen privately to avoid being the target of an attacker. Algorand [13] solves this puzzle by introducing the *cryptographic sortition algorithm*, which employs verifiable random functions (VRFs) to achieve all goals above.

2.4 Parallelizing Transactions

The idea of committees can be taken up to parallelize the validation and ordering process of transactions. *Sharding* [21] is an approach in which the overall system is being partitioned into smaller, (almost) equally-sized committees. It is important that an adversary should not be able to influence the distribution e.g. faulty nodes spread evenly over all committees with overwhelming probability. Byzantine consensus is then individually run in every committee thus simultaneously processing disjoint sets of transactions (*shards*). Ideally, the overall approach allows for scaling throughput almost linearly [21] with an increasing total number of nodes.

Conceptually, equally sized committee can either mean possessing equal hash power, equal stake or just equal number of nodes with proven identity. The denotation depends on the way identities are being managed and how voting power is being distributed by the protocol. In order to ensure security, the evenly distribution of nodes across committees should be random and the required seed can be derived e.g. by generating epoch randomness using Proof-of-Work [21] or employing a distributed-randomness generation protocol [17]. Furthermore, the search for members of the same committee – and thus the creation of an overlay network – should be handled efficiently and not rely on all-to-all broadcasts, e.g. *ELASTICO* [21] introduces a special committee called *directory committee*, which serves as directory service, in which joining nodes can register their identity and get the identities of their co-members announced once a new fully-sized committee can be established.

Two transactions can be committed in different blocks by different committees and thus parallelized if they don't conflict with each other or depend on each other, e.g. if they do not try to double-spend (spend the same unspent transaction output UTXO twice) or if one transaction creates an UTXO as output that is input to the other one [17]. Efficiently analyzing conflicting transactions and dependencies between transactions (such as A must happen before B) can be done using a block-based directed acyclic graph [18].

2.5 Trusted Hardware Components

An interesting approach [15] to increase efficiency is to remove the costs of achieving consensus, in particular the atomic broadcast primitive, from the critical path of the protocol and move it to hardware. In their crash fault-tolerant solution, field programmable gate arrays were used to provide consensus as a service to applications, e.g. an instance of Zookeeper atomic broadcast with a corresponding key-value store. Although requiring specialized hardware might be a constraint for openness and decentralization of DLTs, we still think the general idea of consensus in hardware deserves further exploration by research e.g. for BFT blockchain protocol variants.

We also see potential in making use of trusted execution environments (TEEs) e.g. based on Intel SGX. Hardware components could prove useful to prevent against Sybil attacks by proving unforgeable uniqueness within a network [28]. Recent work proposes Proof-of-Luck (PoL) [25] consensus that utilizes Intel SGX (but can generally be adopted for other TEEs). Conceptually, PoL combines the ideas of TEE based proof-of-time and TEE based proof-of-ownership to make mining energy and time efficient, thus addressing major problems that exist in PoW.

TEEs are also used in FastBFT [20], which is an approach for achieving scalable Byzantine consensus using hardware-assisted secret sharing. Combining TEEs with lightweight secret sharing allows FastBFT to introduce an efficient message aggregation technique – it also incorporates some other optimizations like optimistic execution and a tree topology to achieve better performance and scalability, making it an appealing choice for integration in DLTs.

3 COMPARISON OF EXISTING PROTOCOLS

State-of-the-art BFT protocols employ different assumptions and pursue different goals. In this section we present a brief comparison (also see Table 1) of existing approaches and analyze their efforts.

3.1 Proof-of-Work vs. Proof-of-Stake

ByzCoin. An interesting approach insists in combining different consensus mechanisms i.e. to solve the problem of determining membership without the protocol being target to Sybil attacks using the Proof-of-Work mechanism while at the same time using Byzantine consensus for transactions processing in order to achieve better performance. ByzCoin [16] proposes this idea using a Proof-of-Work mechanism on an separate *identity chain*. Miners obtain hash-power proportionate consensus shares which determine the frequency of being leader of an epoch on the *keyblock chain*, hence having the right to order transactions using BFT consensus. Notable scalability improvements over traditional BFT protocols (like PBFT) such as an hierarchical (or pessimistically flat) communication and collective signatures reduce the communication complexity from quadratical to logarithmic (linear in pessimistic case) and signature verification costs from linear to constant time [16].

Tendermint. Tendermint [6] is a BFT state machine replication protocol that uses *Proof-of-Stake* (PoS), a mechanism in which voting power is being distributed based on a notation of owning portions (*stake*) of a native cryptocurrency on the protocol and *validators* (nodes that run consensus) deposit stake that they risk to loose if they are not honest. This way, the protocol can safely enjoy open membership as long as more than 2/3 of active, stake-weighted validators behave correctly. The Tendermint core consensus algorithm is based on multi-round voting like PBFT, but has some significant improvements for better scalability and decentralization [6]: (i) a novel termination mechanism which efficiently utilizes gossip based communication and (ii) a stake weighted round-robin based leader rotation scheme. However, the leader rotation works deterministically and thus is predictable for an adversary.

3.2 Federated Consensus

Compared to ByzCoin or Tendermint, the Stellar [22] consensus protocol (SCP) enjoys *flexible trust* and constructs a system for *federated Byzantine agreement*. SCP neither relies on PoW nor PoS but can still provide open membership as joining nodes are not automatically trusted by others. While SCP scales well for reasons we explained in Section 2.1, safety depends on nodes to choose adequate quorum slices (trust decisions) and as a possible problem for decentralization and network growth, *there exists no monetary initiative* to run a node and participate in consensus. At time of writing the Stellar network consists of roughly hundred nodes, processing up to thousand transactions per second. However, nodes are mostly run by the Stellar Foundation or businesses like IBM that have inherent interest in operating a secure network.

3.3 Randomized BFT Protocols

HoneyBadger. A practical *asynchronous* BFT protocol that solves the problem of atomic broadcast and thus consensus, is HoneyBadgerBFT (HBBFT) [24]. The HBBFT paper justifies the thought that timing assumptions are harmful by providing an adversarial network scheduler that delays messages and creates an environment that renders weakly synchronous BFT protocols like PBFT useless. Moreover, HBBFT introduces a novel atomic broadcast protocol which is implemented by an efficient reduction to asynchronous common subset (ACS) using a new batching strategy in combination with threshold encryption as we broached in Section 2.2. In experiments, HBBFT shows that it can process thousands of transactions per second while scaling up to hundred nodes and outperform PBFT even in a weakly synchronous network [24].

Algorand. A novel approach to scale a blockchain protocol for *billions* of users is Algorand [13]. For every block that needs to be decided, Algorand forms a representative committee of randomly and secretly chosen users. The committee members actively participate in the Byzantine consensus protocol as they possess the roles of either *proposers* or *verifiers* while others will only learn about the decision of the committee. The selection process uses *cryptographic sortition* [13], an algorithm that relies on the use of verifiable random functions VRFs (see Section 2.2). As the selection process is secret, the committee members can not be targeted by an attacker. The VRFs allow the users to check for themselves if they are elected as proposers or verifiers and also generate an unforgeable proof of the rightful election. This process can be best though of as a secret lottery with each user having a ticket as proof. The selection process yields multiple *proposers*, however only the user that is online and has the ticket with the smallest hash becomes the actual *leader*. *Verifiers* need to reach agreement about the block proposed by the leader. Furthermore, messages are propagated using gossip to all users to learn about the agreed block. Thus, Algorand can generate a new block at the same speed as messages can be propagated through the network and it scales independently of the total number of users in the network [13]. The stake-weighted election process assigns probabilities proportional to the amount of stake a user has – hence the protocol is safe with high probability as long as the attacker controls less than 1/3 of the stake [13].

Gosig. Gosig [19] has the ambition to scale Byzantine consensus on adversarial wide area networks. Like in Algorand, cryptographic sortition is used to randomly and secretly select a leader, however all nodes *actively* participate in reaching agreement in multi-round signature-based voting which allows it to be safe in an asynchronous setting. Gosig adapts scalability improvements on implementation level, e.g. it reduces communication overhead by aggregating signatures into multi-signature form. Furthermore, it efficiently implements proposal broadcast and signature collection phases by utilizing gossip communication. Compared to HoneyBadgerBFT, Gosig has less performance overhead but scarifies provable liveness in the *asynchronous* setting, however implements techniques to increase the probability of liveness on best-effort basis such as failure discovery and asynchronous signature verification [19].

3.4 Sharding BFT Protocols

ELASTICO. Ideally, sharding protocols can increase the throughput of a growing system by partitioning the processing of transactions into shards and processing them in parallel by disjunctive sets of validators (committees) [21]. ELASTICO [21] is a permissionless sharding protocol, which (i) uses Proof-of-Work to establish identities (ii) securely partitions the system into smaller committees, thus parallelizing transaction processing and (iii) scales almost linearly with available computation power (which corresponds to available proven identities). In a way, sharding is an orthogonal approach that can be constructed using any Byzantine consensus protocol, e.g. ELASTICO uses PBFT to run intra-committee consensus. In experiments [21], ELASTICO can increase the number of blocks per epoch from 1 to 16 with increasing number of total nodes (starting with 100 up to $1,600$). But latency slightly increases from $600s$ to $711s$ for finding the proofs of work while the time of reaching consensus within the committees remains at around $100s$.

OmniLedger. Compared to ELASTICO, OmniLedger [17] uses a *bias-resistant distributed randomness generation protocol* instead of PoW to generate a seed for sharding securely. It also introduces an efficient crossshard commit protocol called *AtomiX* [17] that can atomically handle transactions which affect multiple shards. Over ELASTICO, OmniLedger makes significant performance improvements by using optimizations such as: (i) using ByzCoinX [17] instead of PBFT to more efficiently process transactions within shards (e.g. by using better communication patterns such as tree topology), (ii) resolving dependencies on the transaction level to achieve better block parallelization and (iii) introducing a "trust-but-verify" validation architecture for optimistic real-time transaction confirmations. In evaluation results, OmniLedger manages to scale throughput nearly linearly with the number of available validators and achieves a throughput in the range of thousands of transactions per second while it can maintain latencies of below two seconds [17].

4 CONCLUSIONS

In this paper we analyzed different approaches and created a survey of existing techniques and designs for scaling Byzantine consensus. We conclude that great potential lies in the combination of efficient communication strategies such as gossip and cryptographic primitives like threshold-signatures as well as the use of randomness e.g. for cryptographic sortition to elect a committee or leader. Orthogonal to these techniques, the parallelization of consensus in multiple instances can help to scale throughput with a growing number of participants if needed. Additionally, we see a vast yet mostly unexplored research space in the incorporation of hardware components such as trusted execution environments left open for future work. Scalable BFT protocols differ in their ambitions. For instance, Algorand is optimized for scaling to a tremendous number of nodes, FastBFT employs hardware to perform extremely fast, HoneyBadgerBFT does not break any performance records but can withstand adversarial network conditions and OmniLedger shows how to construct a secure scale-out blockchain protocol which can increase the overall system throughput by sharding transactions.

REFERENCES

[1] Ittai Abraham, Guy Gueta, and Dahlia Malkhi. 2018. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil. *arXiv preprint arXiv:1803.05069* (2018).

[2] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. 2017. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936* (2017).

[3] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. 1994. Asynchronous secure computations with optimal resilience. In *Proceedings of the thirteenth Annual ACM Symposium on Principles of Distributed Computing*. ACM, 183–192.

[4] Alysson Bessani, João Sousa, and Marko Vukolić. 2017. A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *Proc. of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. ACM.

[5] Fatemeh Borran and André Schiper. 2010. A leader-free Byzantine consensus algorithm. In *ICDCN 2010*. Springer, 67–78.

[6] Ethan Buchman, Jae Kwon, and Zarko Milosevic. 2018. The latest gossip on BFT consensus. *arXiv preprint arXiv:1807.04938* (2018).

[7] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

[8] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*. Springer, 524–541.

[9] Christian Cachin and Stefano Tessaro. 2005. Asynchronous verifiable information dispersal. In *24th IEEE Symp. on Reliable Distributed Systems*. IEEE, 191–201.

[10] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[11] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. 2016. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*. Springer, 106–125.

[12] John R Douceur. 2002. The sybil attack. In *International Workshop on Peer-to-Peer Systems*. Springer, 251–260.

[13] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 51–68.

[14] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2018. SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains. *arXiv preprint arXiv:1804.01626* (2018).

[15] Zsolt István, David Sidler, Gustavo Alonso, and Marko Vukolic. 2016. Consensus in a Box: Inexpensive Coordination in Hardware. In *NSDI*. 425–438.

[16] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. 2016. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*. 279–296.

[17] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2017. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. https://eprint.iacr.org/2017/406.pdf. (2017).

[18] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*. Springer, 528–547.

[19] Peilun Li, Guosai Wang, Xiaoqi Chen, and Wei Xu. 2018. Gosig: Scalable Byzantine Consensus on Adversarial Wide Area Network for Blockchains. *arXiv preprint arXiv:1802.01315* (2018).

[20] Jian Liu, Wenting Li, G Karame, and N Asokan. 2018. Scalable Byzantine Consensus via Hardware-assisted Secret Sharing. *IEEE Trans. Comput.* (2018).

[21] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*. 17–30.

[22] David Mazieres. 2015. The stellar consensus protocol: A federated model for internet-level consensus. stellar.org/papers/stellar-consensus-protocol.pdf

[23] Silvio Micali, Michael Rabin, and Salil Vadhan. 1999. Verifiable random functions. In *40th Annual Symp. on Foundations of Computer Science*. IEEE, 120–130.

[24] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 31–42.

[25] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. 2016. Proof of luck: An efficient blockchain consensus protocol. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM.

[26] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[27] Team Rocket. 2018. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. https://avalanchelabs.org/avalanche.pdf

[28] Signe Rüsch. 2018. High-Performance Consensus Mechanisms for Blockchains. In *12th EuroSys Doctoral Workshop (EuroDW'18)*. Porto, Portugal. https://www.ibr.cs.tu-bs.de/users/ruesch/papers/eurodw18-ruesch-final.pdf

[29] Claus-Peter Schnorr. 1991. Efficient signature generation by smart cards. *Journal of cryptology* 4, 3 (1991), 161–174.

[30] Victor Shoup. 2000. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 207–220.

[31] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Keeping authorities "honest" or "bust" with decentralized witness cosigning. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 526–545.

[32] Marko Vukolić. 2015. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*. Springer, 112–125.

[33] Yin Yang. 2018. LinBFT: Linear-Communication Byzantine Fault Tolerance for Public Blockchains. *arXiv preprint arXiv:1807.01829* (2018).