The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication

Marko Vukolić

The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication

Marko Vukolić

IBM Research - Zurich
mvu@zurich.ibm.com

Abstract. Bitcoin cryptocurrency demonstrated the utility of global consensus across thousands of nodes, changing the world of digital transactions forever. In the early days of Bitcoin, the performance of its probabilistic *proof-of-work* (PoW) based consensus fabric, also known as *blockchain*, was not a major issue. Bitcoin became a success story, despite its consensus latencies on the order of an hour and the theoretical peak throughput of only up to 7 transactions per second.
The situation today is radically different and the poor performance scalability of early PoW blockchains no longer makes sense. Specifically, the trend of modern cryptocurrency platforms, such as Ethereum, is to support execution of arbitrary distributed applications on blockchain fabric, needing much better performance. This approach, however, makes cryptocurrency platforms step away from their original purpose and enter the domain of database-replication protocols, notably, the classical *state-machine replication*, and in particular its Byzantine fault-tolerant (BFT) variants.
In this paper, we contrast PoW-based blockchains to those based on BFT state machine replication, focusing on their scalability limits. We also discuss recent proposals to overcoming these scalability limits and outline key outstanding open problems in the quest for the "ultimate" blockchain fabric(s).

Keywords: Bitcoin, blockchain, Byzantine fault tolerance, consensus, proof-of-work, scalability, state machine replication

1 Introduction

Distributed consensus, infamous for its limited scalability, was for decades perceived as a synchronization primitive that is to be used only in applications in desperate need of consistency and only among few nodes (see e.g., [8, 28]). However, Nakamoto's Bitcoin cryptocurrency [48] demonstrated the utility of decentralized consensus across thousands of nodes, changing the world of digital transactions forever.

Although the Bitcoin protocol does not actually implement consensus in the traditional distributed computing sense, it comes very close to consensus with probabilistic agreement [26]. In a nutshell, the goal of a cryptocurrency such as Bitcoin, is to totally order transactions on a distributed ledger, also called

a *blockchain*. The Bitcoin blockchain consists of a hashchain of blocks: every block contains an ordered set of transactions and a hash of the preceding block (starting from the initial, the so-called "genesis" block). The key part is the *Proof-of-Work* (PoW) aspect of the hashchain [22]: a Bitcoin block contains nonces that a Bitcon *miner* (i.e., a node attempting to add a block to the chain) must set in such a way that the hash of the entire block is smaller than a known *target*, which is typically a very small number. In fact, in Bitcoin, the *difficulty* of mining, inversely proportional to the target, is adjusted dynamically throughout the lifetime of the system. The adjustment is made with respect to the block-mining rate and, indirectly, with respect to the computational power of nodes participating in the system, to maintain the expected block-mining rate at roughly one block every 10 min [48]. This latency of 10 minutes (per block) is often referred to as the *block frequency* (see e.g., [23]) and is one of the two critical "magic numbers" in Bitcoin, the other being the *block size*, which is set in Bitcoin to 1 MB.

In the early days of Bitcoin, the *performance scalability* of its probabilistic PoW-based blockchain was not a major issue. Even today, Bitcoin works with a consensus latency of about an hour (for the recommended 6-block transaction confirmation), and with up to 7 (seven) transactions per second peak throughput (with smallest 200-250 byte transactions). On top of this, the Bitcoin network uses a lot of power, which, in 2014, was roughly estimated to be in the ballpark of 0.1-10 GW [49].

However, blockchain requirements change rapidly, with high latency and low throughput of Bitcoin-like blockchain becoming a major challenge [6]. As a comparison, leading global credit-card payment companies serve roughly 2000 transactions per second on average [59], with a peak capacity designed to sustain more than 10000 transactions per second. Moreover, the trend of modern cryptocurrency platforms, such as Ethereum [58], is to support execution of Turing-complete code on blockchain fabric in the form of *smart contracts*, which are, roughly speaking, custom, self-executing programs (distributed applications) that automatically enforce properties of a digital contract. In fact, smart-contract blockchain is seen as a candidate technology for distributed ledgers in many industries. Clearly, in many of the intended smart-contract use cases, distributed applications require much better performance than that offered by Bitcoin. The banking industry is one prominent example, where potential blockchain use cases go well beyond digital payments [46] to, e.g., securities trade settlements and trade finance.

Smart-contract use cases take the blockchain well beyond its original cryptocurrency purpose, back to the domain of database replication protocols, notably, the classical *state-machine replication* [54]. Indeed, a smart contract can be modeled as a state machine, and its consistent execution across multiple nodes in a distributed environment can be achieved using state machine replication. A family of state-machine replication protocols particularly interesting for blockchain is the family of *Byzantine fault-tolerant* (BFT) [38] state-machine replication protocols, which promise consensus despite participation of mali-

cious (Byzantine) nodes. In more than three decades of research, BFT protocol prototypes have been shown to be practical [10], reaching practically minimal latencies allowed by the network, and supporting tens of thousands transactions per second (see e.g., [35, 3]). However, BFT and state-machine replication protocols in general are often challenged for their scalability in terms of number of nodes (replicas) [8], and have not been throughly tested in this aspect critical to blockchain.

In summary, blockchain consensus technologies of today, PoW and BFT, sit at the two opposite ends of the scalability spectrum. Roughly speaking, PoW-based blockchain offers good node scalability with poor performance, whereas BFT-based blockchain offers good performance for small numbers of replicas, with not-well explored and intuitively very limited scalability. This current state of blockchain scalability is sketched in Figure 1. Given seemingly inherent trade-offs between the number of replicas and performance, it is not clear today what the optimal blockchain solution is for the sweet spot relevant for many use cases in which the number of nodes n ranges from a few tens to 1000 (or perhaps few thousands).

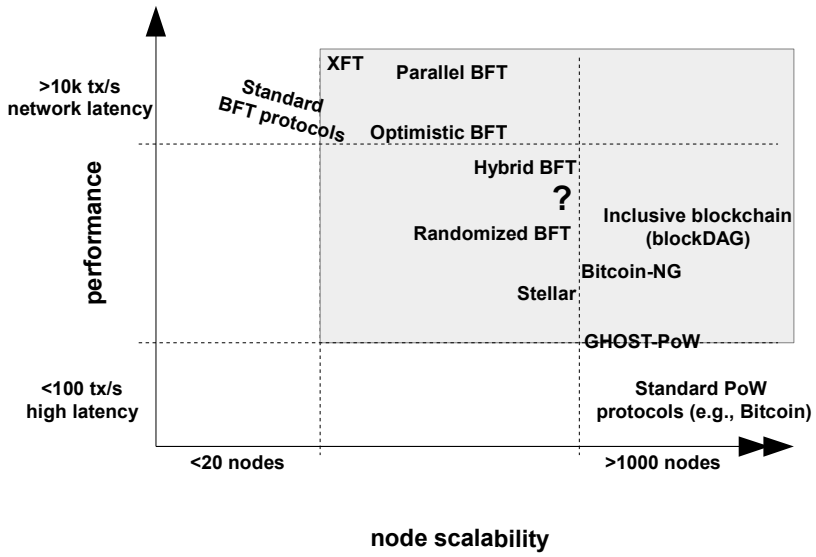


Fig. 1. Illustration of performance and scalability of different families of PoW and BFT protocols discussed in this paper. The actual, real-world performance of systems that touch upon the grey area is subject to further research. Hence, their positioning within the grey area is at the moment entirely speculative and for motivational purposes only.

In this paper, we overview recent efforts towards improving scalability on both sides of the spectrum and highlight interesting directions and open problems

in the quest for the "ultimate" blockchain fabric. First, in Section 2 we compare PoW-based blockchains to those based on BFT state-machine replication. Then, in Section 3, we overview novel promising approaches to scaling PoW and BFT protocols. We conclude in Section 4 with several open questions that will be interesting to tackle in the very near future.

2 PoW vs. BFT blockchains

Table 1 gives a high-level comparison between PoW consensus and BFT consensus for a set of important blockchain properties. These properties include node identity management, consensus finality (or, dually, the possibility of temporary forks in the blockchain), scalability in terms of number of consensus nodes and clients, performance (latency, throughput, power consumption), tolerated power of adversary, network synchrony assumptions, and, last but not least, existence of correctness proofs of protocols underlying blockchain. This set of properties is certainly not exhaustive, but we believe it is representative for comparing two blockchain families. In the rest of this section, we discuss Table 1 in more detail.

Table 1. High-level comparison between PoW and BFT blockchain consensus families for a set of important blockchain properties. Entries in bold suggest desirable features and highlight advantages of one consensus family over the other.

	PoW consensus	BFT consensus
Node identity management	**open, entirely decentralized**	permissioned, nodes need to know IDs of all other nodes
Consensus finality	no	**yes**
Scalability (no. of nodes)	**excellent (thousands of nodes)**	limited, not well explored (tested only up to $n \leq 20$ nodes)
Scalability (no. of clients)	**excellent (thousands of clients)**	**excellent (thousands of clients)**
Performance (throughput)	limited (due to possible of chain forks)	**excellent (tens of thousands tx/sec)**
Performance (latency)	high latency (due to multi-block confirmations)	**excellent (matches network latency)**
Power consumption	very poor (PoW wastes energy)	**good**
Tolerated power of an adversary	$\leq 25\%$ computing power	$\leq 33\%$ voting power
Network synchrony assumptions	physical clock timestamps (e.g., for block validity)	**none for consensus safety** (synchrony needed for liveness)
Correctness proofs	no	**yes**

Node identity management. How node identities are managed in PoW and BFT protocols is possibly their most fundamental difference. PoW blockchains feature an entirely decentralized identity management — for example, anybody can download the code for Bitcoin miner, and start participating in the protocol, knowing basically only a single peer to start with. This is a very powerful feature of PoW blockchains and the main reason why they are the blockchain family of choice when it comes to so-called "public" blockchains in which anybody is allowed to participate. Such public blockchains are sometimes also called "permissionless" blockchains — permissionless participation is made possible by PoW, as PoW inherently addresses the Sybil attack [18], infamous in anonymous networks. Specifically, in PoW-based blockchains, the ability of a node (resp., a pool of nodes) to influence the outcome of PoW consensus depends on computational power of a node (resp., a pool).

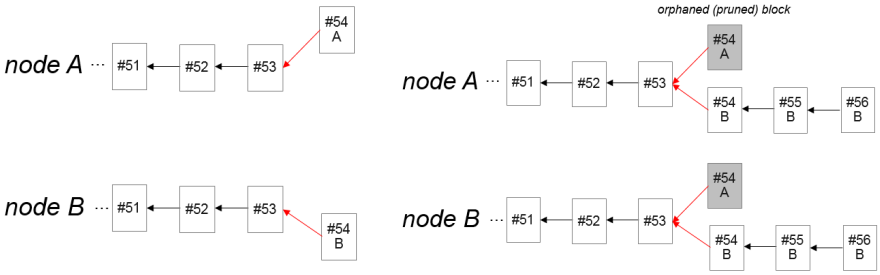
In contrast, the BFT approach to consensus typically requires every node to know the entire set of its peer nodes participating in consensus. This in turn calls for a (logically) centralized identity management in which a trusted party issues identities and cryptographic certificates to nodes.[1] Intuitively, this aspect of BFT-based blockchains puts it at a disadvantage with respect to PoW blockchains. That said, in a number of emerging blockchain applications (e.g., banking, finance, land and real-estate ownership ledgers) the requirement for known identity of nodes might anyway be imposed for legal and compliance reasons. This explains why BFT consensus protocols are the technology of choice for so-called "permissioned" blockchains, which require blockchain participants identity to be known.

Consensus finality. Roughly speaking, what is often informally referred to as "consensus finality" (and sometimes as "forward security" [15]) is a property that mandates that a valid block, appended to the blockchain at some point in time, be never removed from the blockchain. In the standard distributed computing terminology, "consensus finality" follows from a combination of the *total order* and *agreement* properties of total order (atomic) broadcast [17], which is the primitive all state-machine replication protocols are built upon (total order broadcast is, in turn, equivalent to consensus). Translated to blockchain terminology, this property can be phrased as follows:

Definition 1 (Consensus Finality). *If a correct node p appends block b to its copy of the blockchain before appending block b', then no correct node q appends block b' before b to its copy of the blockchain.*

Consensus finality is not satisfied by PoW-based blockchains. To see why, note that, besides obviating the need for identity management, PoW acts as a randomized concurrency control mechanism, in which the block frequency is adjusted such that block collisions (i.e., concurrent appends of different blocks to

[1] Here, it is important to note that after an initial bootstrap of a BFT-based blockchain, the nodes already on the blockchain could themselves act together as a distributed trusted party and help reconfigure the system [53, 5].



(a) Consensus finality violation resulting in a fork.

(b) Eventually, one of the blocks must be pruned by a conflict resolution rule (e.g., Bitcoin's longest chain rule).

Fig. 2. Illustration of a violation of consensus finality, fork and conflict resolution.

the blockchain) are rare. However, as concurrency control is only probabilistic and as block propagation over a network can take some time [16], collisions do happen, resulting in temporary forks on the blockchain that PoW-based blockchains are prone to even if all nodes are honest. These temporary forks (see Fig. 2 for an illustration) are resolved by rules such as Bitcoin's longest (most difficult) fork rule [48], or the GHOST rule [55], a variant of which is used in Ethereum. However, the very presence of temporary forks implies no consensus finality. As we discuss in more detail below, absence of consensus finality directly impacts the consensus latency of PoW blockchains as transactions need to be followed by several blocks to increase the probability that a transaction will not end up being pruned and removed from the blockchain (we speak of multi-block confirmation).

In contrast, consensus finality is satisfied by all BFT and state-machine replication protocols.[2] This gives BFT-based blockchains a clear advantage over PoW, as applications, users and smart contracts can have immediate confirmation of the final inclusion of a transaction into the blockchain.

Scalability. Although decoupling the issue of blockchain scalability (with the number of nodes and clients in the system) from that of blockchain performance (latency and throughput) is not entirely possible, we nevertheless first focus on the number of nodes and clients for which PoW and BFT technologies have been proven to work in practice.

On the one hand, the Bitcoin network features thousands of mining nodes, demonstrating node scalability of PoW-based blockchains in practice. That said, it is worth mentioning that grouping of miners into mining pools (with the goal of splitting mining rewards and making mining a financially more predictable endeavour) plagues Bitcoin, effectively centralizing the cryptocurrency [27]. We note that mining pool centralization is not a specific trait of Bitcoin, but more a

[2] Provided the assumptions about the power of the adversary hold.

consequence of the popularity of a PoW blockchain, affecting also many *altcoins* (alternative Bitcoin-like cryptocurrencies) as well as popular blockchains, such as Ethereum.

On the other hand. BFT and state-machine replication are, in general, perceived as protocols with poor scalability (see, e.g., Brewer's CAP theorem [8]). However, having been invented in the context of replicating traditional applications, such as databases, for fault-tolerance, BFT protocols were never really tested thoroughly for their scalability beyond, say, $n = 10$ or $n = 20$ nodes, in particular in the light of the fairly modest performance targets of many blockchain applications. Intuitively, because of their intensive network communication which often involves as many as $O(n^2)$ messages per block [10], BFT protocols are seen in the database and systems communities as not scalable (see also [45]).[3] This is true even for their crash-tolerant counterparts, i.e., replication protocols such as Paxos [37], Zab [31] and Raft [50], which are used in many large scale systems but practically never across more than a handful of replicas (see e.g., [13]).

Finally, when it comes to scalability with the number of clients, both PoW and BFT protocols support thousands of clients and scale well.

Performance. Beyond the very limited performance of Bitcoin of up to 7 transactions per second (with the current block size) and 1-hour latency with 6-block confirmation, PoW-based blockchains face inherent performance challenges. As we already discussed, the two main performance-related parameters of a PoW blockchain are *block size* and *block frequency*. Increasing the block size with the goal of boosting throughput comes at the cost of increasing the latency, because of longer propagation delays of larger blocks across the Internet. These longer delays, in turn, have negative implications on blockchain security: longer delays may increase the number of forks and the possibilities for mounting double-spending attacks [34], because of the possibility of temporary chain forks and absence of consensus finality in PoW blockchains. Similar security challenges apply when the block frequency is increased, with the goal of reducing the latency of multi-block confirmation. The exact security implications of tuning the block frequency and the block size in PoW-based blockchain are in general rather involved (see e.g., [55] for an analysis) and should be handled with care. With this in mind, limited performance is seemingly inherent to PoW blockchains and not an artifact of a particular implementation.

In contrast, modern BFT protocols have been confirmed to sustain tens of thousands of transactions with practically network-speed latencies, not only as prototypes (e.g., [35, 12, 3]) but also as practical systems [5].

Adversary. PoW and BFT consider different adversaries. In PoW blockchains, what matters is the total computational (hashing) power controlled by the adversary. Initially, Bitcoin was thought to be invulnerable so long as the adversary

[3] That said, it is worth noting that there are *optimistic* BFT protocols with $O(n)$ common-case (expected) message-complexity (see, e.g., [52, 3]) — we discuss these later in more detail.

controls less than 50% of hashing power. Years later, it was shown that Bitcoin mining is actually vulnerable even if only 25% of the computing power is controlled by an adversary [24]. In contrast, BFT voting schemes are known to tolerate at most $n/3$ corrupted nodes [20]. This bound holds only when the network is allowed to be (from time to time) fully asynchronous — strengthening synchrony assumptions makes it possible to raise this threshold. The classical $n/3$ threshold bound for BFT consensus can be generalized to general adversary structures, where an adversary can control different subsets of nodes [29, 57].

Network synchrony. Bitcoin relies on the local time of a node to timestamp a block. Roughly speaking, a block is accepted as valid if its timestamp is greater than the median of the last 11 blocks. Additionally, timestamps play a major role in calculating the difficulty of mining and maintaining block frequency. Therefore, loose clock synchrony is needed for liveness. However, timestamp manipulation attacks that may also compromise the consistency of the blockchain are conceivable (see the "zeitgeist attack" [1]). Although such attacks are difficult to stage against major PoW blockchains such as Bitcoin, they have been successfully performed in the context of some PoW altcoins.

BFT protocols typically do not rely on any physical clock.[4] However, eventually synchronous communication is needed to ensure liveness, owing to the FLP consensus impossibility result, which states that consensus is impossible to achieve deterministically with potentially faulty nodes in a purely asynchronous system [25]. The safety properties of consensus, including consensus finality, are maintained despite global communication outages and arbitrarily long asynchrony periods [20].

Correctness proofs. Historically, state-machine replication protocols, and in particular their BFT variants, have been recognized as very challenging to design and implement [11, 5, 3]. Consequently, new protocols are subject to detailed academic scrutiny and therefore come with (more or less) detailed proofs, sometimes even with formal proofs that take an entire PhD thesis (see [14, 41]). Even if it may be understandable why Bitcoin was originally deployed without having been subjected to similar scrutiny, it is rather surprising that novel PoW blockchains are rarely accompanied by a detailed security and distributed protocol and security analysis.

3 Improving blockchain scalability

In this section we overview and discuss several recent efforts that focus on improving the scalability aspects of both PoW and BFT blockchains.

[4] Some state-machine replication protocols do use physical clock timestamps, but only to improve performance [19].

Improving the performance of PoW blockchains. Sompolinski and Zohar recently proposed the GHOST (Greedy Heaviest-Observed Sub-Tree) rule [55], which basically resolves conflicts in a PoW blockchain by weighing the subtrees rooted in blocks rather than the longest (sub)chain rooted in given blocks. Although GHOST is essentially a conflict-resolution strategy, it offers performance benefits over the standard longest (heaviest) chain rule of Bitcoin, as it provides more secure means of increasing the block frequency and the block size [55]. A variant of the GHOST rule is actually implemented in the Ethereum blockchain [58], although the GHOST-PoW performance has not yet been adequately stress-tested with high loads (in 2016, typical Ethereum throughput is fewer than 20,000 transactions per day, i.e., about 0.2 tx/s on average).[5]

Bitcoin-NG is a novel proposal by Eyal et al. [23] that uses standard PoW for leader election, declaring a node which mines a block with standard difficulty (called a key block) to become a leader until a new key block is mined. In the meantime, the leader can append microblocks to the chain, which are not subject to PoW mining but are merely hashchained together. As such, microblocks considerably increase the throughput of the whole system and decrease the latency (that said, Bitcoin-NG is still to be stress-tested in practice). In a sense, Bitcoin-NG mixes leader election, often seen in BFT protocols, with a leader-centric protocol in between leader-election epochs. However, what is different in Bitcoin-NG from BFT protocols is that leader election is PoW-based. Consequently, forks are still possible in Bitcoin-NG and consensus finality is not ensured, which may lead to security implications such as asset double-spending, as discussed earlier.

Scaling blockchain through parallelization. Scaling blockchain by making it a blockDAG (directed acyclic graph) rather than a linear chain of blocks, was recently proposed by Lewenberg et al. in the context of PoW [39]. The idea is to allow non-conflicting transactions (e.g., those transactions that do not constitute double-spending attempts) to be initially on different forks, but to eventually merge the forks by mining a block that would include them both in the ledger.[6] The BFT and state-machine replication communities have also been intensively exploring the idea of *parallel replication* for a few years now, leveraging parallelization of execution of independent requests (transactions) (see, e.g., [33, 43]).

Eliminating communication and resource overhead in BFT protocols. As we have already discussed, the major challenge for BFT protocols that prevents their wider adoption in blockchain is their scalability in terms of the number of nodes. Stellar [44] is an ongoing effort aimed at removing unanimously accepted membership lists from BFT protocols, while maintaining the other BFT advantages over PoW. Other approaches target the BFT scalability without changing

[5] https://etherchain.org/statistics/basic.

[6] BlockDAGs are conceptually similar to the notion of parallel sharded chains (sidechains) combined with merge mining.

membership assumptions. These include *optimistic BFT* protocols [52, 3] which feature linear communication complexity in the "common case" and resort to expensive $O(n^2)$ communication among nodes featured by classical protocols such as PBFT [10] only if the network and the process fault pattern are particularly infavorable. However, even optimistic BFT have a resource and communication overhead when compared to crash-tolerant replication protocols (e.g., [37, 31, 50]), which are better proven in practice and may serve as a baseline for BFT.

To rectify this, Liu et al. recently proposed a novel network and node fault model called XFT [40] that allows one to tolerate up to $n/2$ Byzantine nodes. At the same time, XFT features message patterns characteristic to crash-tolerant replication protocols, i.e., without the overhead pertaining to typical BFT message patterns. To this end, XFT ("cross" fault tolerance) challenges the established ability of a BFT adversary to control the network and Byzantine nodes simultaneously, decoupling network faults from Byzantine-node faults, treating them as largely independent. As such, XFT goes in the direction of a more realistic adversary model that resembles the one of PoW blockchains, which are not very concerned with the ability of the adversary to control the entire communication network.

Finally, another appealing direction for future BFT-based blockchain is BFT protocols that leverage small pieces of trusted hardware (e.g., [32]) to improve communication and reduce resource cost.

Randomized BFT. Randomized BFT protocols (e.g., [7, 56, 9]) are appealing alternative to standard, eventually synchronous [21] BFT protocols such as PBFT. Specifically, randomized BFT protocols circumvent the FLP consensus impossibility result [25] by guaranteeing correctness with very high probability (i.e., always, except with negligible probability), rather than deterministically. This allows randomized BFT protocols to be completely asynchronous [4].

For many years, an issue with randomized BFT protocols has been their performance. Specifically, classical randomized BFT (e.g., [4, 7, 56, 9]) are very inefficient compared to eventually synchronous, deterministic BFT protocols mostly due to overhead of cryptographic tools they use. However, this may be changing soon with novel randomized BFT protocols such as HoneyBadger [47] showing promise for good practical performance (i.e., reasonably high throughput) with up to about 100 nodes, through cherry-picking best available cryptographic tools for randomization as well as processing requests in very large batches. Clearly, large batches negatively impact latency, but this could be addressed by *Hybrid BFT* protocols [2] that may combine very efficient optimistic and deterministic BFT protocols (e.g., those described in [3]) with practical randomized protocols such as HoneyBadger. Early examples of such Hybrid BFT protocols can be found in [2, 36, 52], but the development of future Hybrid BFT protocols can be facilitated by using the modular BFT design framework described in [3].

Mixing PoW and BFT. Recently, Decker et al. [15] have proposed to enhance PoW blockchain with BFT (concretely, the PBFT protocol [10]), primarily to ensure consensus finality in a PoW blockchain by using BFT. SCP [42] also

proposes a hybrid PoW/BFT protocol, using PoW for identity management and (parallel and hierarchical) BFT consensus for agreement. Clearly, the above discussion on the importance of scaling BFT in terms of the number of nodes is also critical to such approaches that mix PoW and BFT.

4 Conclusion and open problems

We briefly overviewed state of the art as well as emerging directions towards scalable blockchain. We contrasted proof-of-work (PoW) and Byzantine fault-tolerant (BFT) consensus protocols, highlighting their respective advantages.

Future work will be very dynamic and interesting. Making Figure 1 more precise, i.e., placing various protocols at the correct place with respect to their performance versus their node-scalability, entails a fair amount of research, but represents an immediate open problem that needs to be better understood to facilitate future blockchain scalability improvements. Furthermore, a lot of potential lies in synergies between PoW and BFT, both when it comes to combining protocol techniques and when it comes to refining the adversarial and network models.

Finally, for the most demanding blockchain applications, it would be interesting to move computationally expensive parts of BFT protocols (e.g., cryptography) closer to hardware. In general, implementing consensus in hardware is indeed very appealing and may yield impressive performance, as attested by recent proposals that explore this idea in the context of crash fault-tolerance [51, 30].

References

1. The "Zeitgeist attack". http://bitcoin.stackexchange.com/questions/1055/what-is-the-zeitgeist-attack-does-it-affect-all-blockchain-technologies.
2. Marcos Kawazoe Aguilera and Sam Toueg. Failure detection and randomization: A hybrid approach to solve consensus. *SIAM J. Comput.*, 28(3):890–903, 1998.
3. Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 BFT protocols. *ACM Trans. Comput. Syst.*, 32(4):12:1–12:45, January 2015.
4. Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 27–30, 1983.
5. Alysson Neves Bessani, João Sousa, and Eduardo Adílio Pelinson Alchieri. State machine replication for the masses with BFT-SMART. In *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014*, pages 355–362, 2014.
6. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015*, pages 104–121, 2015.

7. Gabriel Bracha. An asynchronou [(n-1)/3]-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 154–162, 1984.
8. Eric A. Brewer. Towards robust distributed systems (abstract). In *ACM Symposium on Principles of Distributed Computing (PODC)*, page 7, 2000.
9. Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 123–132, 2000.
10. Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002.
11. Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 2007.
12. Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, and Mirco Marchetti. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 153–168. USENIX Association, 2009.
13. James C. Corbett, Jeffrey Dean, and Michael Epstein et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):8, 2013.
14. Roberto de Prisco. *On Building Blocks for Distributed Systems*. PhD thesis, Massachussets Institute of Technology, 1999.
15. Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *17th International Conference on Distributed Computing and Networking (ICDCN)*, 2016.
16. Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013*, pages 1–10, 2013.
17. Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4):372–421, 2004.
18. John R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002*, pages 251–260, 2002.
19. Jiaqing Du, Daniele Sciascia, Sameh Elnikety, Willy Zwaenepoel, and Fernando Pedone. Clock-RSM: Low-latency inter-datacenter state machine replication using loosely synchronized physical clocks. In *The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2014.
20. Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35, April 1988.
21. Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988.
22. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference*, pages 139–147, 1992.
23. Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI '16*, 2016.
24. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014*, pages 436–454, 2014.

25. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
26. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310, 2015.
27. Arthur Gervais, Ghassan O. Karame, Vedran Capkun, and Srdjan Capkun. Is Bitcoin a decentralized currency? *IEEE Security & Privacy*, 12(3):54–60, 2014.
28. Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.
29. Rachid Guerraoui and Marko Vukolić. Refined quorum systems. *Distributed Computing*, 23(1):1–42, 2010.
30. Zsolt Istvan, David Sidler, Gustavo Alonso, and Marko Vukolić. Consensus in a box: Inexpensive coordination in hardware. In *Proceedings of the 13th USENIX symposium on Networked systems design and implementation*, NSDI, 2016.
31. Flavio Paiva Junqueira, Benjamin C. Reed, and Marco Serafini. Zab: High-performance broadcast for primary-backup systems. In *Proceedings of the Conference on Dependable Systems and Networks (DSN)*, pages 245–256, 2011.
32. Rüdiger Kapitza, Johannes Behl, Christian Cachin, Tobias Distler, Simon Kuhnle, Seyed Vahid Mohammadi, Wolfgang Schröder-Preikschat, and Klaus Stengel. CheapBFT: resource-efficient Byzantine fault tolerance. In *European Conference on Computer Systems, Proceedings of the Seventh EuroSys Conference 2012*, pages 295–308, 2012.
33. Manos Kapritsos, Yang Wang, Vivien Quema, Allen Clement, Lorenzo Alvisi, and Mike Dahlin. All about Eve: execute-verify replication for multi-core servers. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, OSDI'12, pages 237–250. USENIX Association, 2012.
34. Ghassan O. Karame, Elli Androulaki, Marc Roeschlin, Arthur Gervais, and Srdjan Capkun. Misbehavior in Bitcoin: A study of double-spending and accountability. *ACM Trans. Inf. Syst. Secur.*, 18(1):2, 2015.
35. Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27:7:1–7:39, January 2010.
36. Klaus Kursawe and Victor Shoup. Optimistic asynchronous atomic broadcast. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP*, pages 204–215, 2005.
37. Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16:133–169, May 1998.
38. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, July 1982.
39. Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015*, pages 528–547, 2015.
40. Shengyun Liu, Christian Cachin, Vivien Quéma, and Marko Vukolić. XFT: practical fault tolerance beyond crashes. *CoRR*, abs/1502.05831, 2015.
41. Giuliano Losa. *Modularity in the design of robust distributed algorithms*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2014.
42. Loi Luu, Viswesh Narayanan, Kunal Baweja, Chaodong Zheng, Seth Gilbert, and Prateek Saxena. SCP: A computationally-scalable Byzantine consensus

protocol for blockchains. Cryptology ePrint Archive, Report 2015/1168, 2015. http://eprint.iacr.org/.
43. Parisa Jalili Marandi, Carlos Eduardo Benevides Bezerra, and Fernando Pedone. Rethinking state-machine replication for parallelism. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014*, pages 368–377, 2014.
44. David Maziéres. The Stellar consensus protocol: A federated model for internet-level consensus. https://www.stellar.org/papers/stellar-consensus-protocol.pdf, November 2015.
45. James Mickens. The saddest moment. *;login: The Usenix magazine*, 39(3), 2014.
46. Udo Milkau and Jürgen Bott. Digitalisation in payments: From interoperability to centralised models? *Journal of Payment Strategy & Systems*, 9(3), 2015.
47. Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Cryptology ePrint Archive 2016/199*, 2016.
48. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.
49. Karl J. O'Dwyer and David Malone. Bitcoin mining and its energy footprint. In *Proceedings of the 2014 IET Irish Signals & Systems Conference*, 2014.
50. Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320. USENIX Association, 2014.
51. Marius Poke and Torsten Hoefler. DARE: high-performance state machine replication on RDMA networks. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, HPDC 2015*, pages 107–118, 2015.
52. HariGovind V. Ramasamy and Christian Cachin. Parsimonious asynchronous byzantine-fault-tolerant atomic broadcast. In *Principles of Distributed Systems, 9th International Conference, OPODIS 2005*, pages 88–102, 2005.
53. Rodrigo Rodrigues, Barbara Liskov, Kathryn Chen, Moses Liskov, and David A. Schultz. Automatic reconfiguration for large-scale reliable storage systems. *IEEE Trans. Dependable Sec. Comput.*, 9(2):145–158, 2012.
54. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.
55. Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in Bitcoin. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015*, pages 507–527, 2015.
56. Sam Toueg. Randomized byzantine agreements. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, Vancouver, B. C., Canada, August 27-29, 1984*, pages 163–178, 1984.
57. Marko Vukolić. *Quorum Systems: With Applications to Storage and Consensus*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.
58. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. http://gavwood.com/paper.pdf, 2015.
59. Aviv Zohar. Bitcoin: under the hood. *Commun. ACM*, 58(9):104–113, 2015.