An In-Depth Look of BFT Consensus in Blockchain: Challenges and Opportunities

Suyash Gupta
Jelle Hellings
Sajjad Rahnama
Mohammad Sadoghi
Exploratory Systems Lab
Department of Computer Science
University of California, Davis

Abstract

Since the introduction of Bitcoin—the first wide-spread application driven by blockchains—the interest of the public and private sector in blockchains has skyrocketed. At the core of this interest are the ways in which blockchains can be used to improve data management, e.g., by enabling federated data management via decentralization, resilience against failure and malicious actors via replication and consensus, and strong data provenance via a secured immutable ledger.

In practice, high-performance blockchains for data management are usually built in permissioned environments in which the participants are vetted and can be identified. In this setting, blockchains are typically powered by Byzantine fault-tolerant consensus protocols. These consensus protocols are used to provide full replication among all honest blockchain participants by enforcing an unique order of processing incoming requests among the participants.

In this tutorial, we take an in-depth look at Byzantine fault-tolerant consensus protocols. First, we take a look at the theory behind replicated computing and consensus. Then, we delve into how common consensus protocols operate. Finally, we take a look at current developments and briefly look at our vision moving forward.

ACM Reference Format:

1 Introduction

Since the introduction of Bitcoin—the first wide-spread application driven by blockchain—the interest of the public and private sector in blockchain has skyrocketed. Recently, this interest has cumulated in the introduction of several blockchain-inspired database systems and blockchain fabrics [2, 3, 25, 26, 52, 53]. Blockchain-based systems have also been demonstrated to address challenges in various other fields such as the trade of valuable commodities [11, 15, 61], food production [29], managing land property rights [59], managing identities [4, 11, 59], supporting transparent aid delivery [4, 59], managing health care data [7, 31, 43], insurance fraud prevention [42], energy production and energy trading [61], and managing compliance with the GDPR [16].

In each of these systems and use cases, blockchain technology is used to improve data management in one way or another. Blockchain technology—which is at the intersection of database systems, distributed systems, and cryptography—provides promises for new directions in data management by utilizing and combining underused techniques from distributed systems and cryptography in novel and powerful ways [10, 19, 40, 51].

The role of Bitcoin, Ethereum, and other cryptocurrencies in these developments cannot be understated: at their core, Bitcoin and Ethereum provided proof of the viability of blockchain techniques in non-trivial large-scale settings. At the core of these cryptocurrencies is the maintenance of a blockchain among a group of replicas. This blockchain holds an ordered record of all transactions and this record is secured against changes using cryptographic primitives. To assure that all replicas agree on the same set of transactions and maintain the same blockchain, new transactions are agreed upon via a consensus protocol (which are fault-tolerant counterparts of the classical two-phase and three-phase commit protocols used in distributed database systems [32, 37, 63]).

From our perspective, these technologies can strengthen data management in three vital directions:
1. The blockchain structure in Bitcoin and Ethereum utilizes cryptographic primitives to harden against unwanted changes (providing immutability). These techniques can also be used to secure a traditional database journal against tampering, which is a major step towards providing systems that provide reliable data provenance. Such tamper-resistant data storage can be used to irrefutably proof any claims about the current state of the data.

2. Fault-tolerant consensus—used to replicate the transactions that are part of the blockchain—is designed to deal with malicious behavior of some of the replicas. The same techniques can also be used to deal with malicious behavior caused by the compromise (e.g., due to a cyberattack) of some of the replicas in a system. In this way, blockchain technology promises one way to harden against cyberattacks on data-based services, which can reduce the huge societal and economic impacts of such attacks [33, 54–56, 64].

3. Finally, fault-tolerance consensus can also be used as the technique for supporting federated data management, the collective management of a single database among various stakeholders. Federated data management is in itself a major step towards dealing with data quality issues arising from the non-federated interchange of information between various stakeholders and, as such, can reduce the huge negative economic impact of bad data [24, 41, 62]. Moreover, in this federated setting, the immutable and irrefutable structure of the blockchain can further help in policing disputes in cases where some stakeholders do not trust each other.

The explosion of blockchain based applications, products, and proof-of-concepts (see, e.g., [3, 10, 50, 69]), has led to the development of several different approaches towards consensus. We can roughly categorize these approaches into permissionless blockchains, as used in public settings by Bitcoin and other cryptocurrencies, and permissioned blockchains, which are better suited for managed private environments.

In permissionless blockchains such as Bitcoin, Proof-of-Work-inspired consensus algorithms are used to replicate data [36, 50, 69]. These algorithms require limited communication between replicas and can operate in unstructured peer-to-peer networks in which independent parties can join and leave at any time [60]. Proof-of-Work uses computationally complex puzzles to limit the influence any malicious party has on the evolution of the blockchain. At the same time, these puzzles incur a high computational costs on all parties, which has raised questions about the sustainability of the energy consumption of permissionless blockchain systems [17, 68]. Additionally, the complexity of Proof-of-Work puzzles causes relative long transaction processing times (minutes to hours) and significantly limits the number of transactions a permissionless blockchain can handle: in 2017, it was reported that Bitcoin can only process 7 transactions per second, whereas Visa already processes 2000 transactions per second on average [59]. Finally, the design of Proof-of-Work prevents scalability, as adding more computational power to the network will only increase the cost of Proof-of-Work puzzles. Other permissionless consensus algorithms based on Proof-of-Work, such as Proof-of-Space and Proof-of-Stake have similar limitations with respect to resource usage and transaction throughput.

Most data management use cases do not require the flexibility of unstructured peer-to-peer networks in which participants can join and leave at any time: data management systems are usually employed in a managed environment in which all participants are known, can be identified, and are vetted. This is exactly the setting for which permissioned blockchains are designed. In these blockchains, traditional Byzantine fault-tolerant consensus algorithms based on consensus algorithms such as PBFT are employed to accept, order, and execute client transactions among all replicas [5, 6, 13, 45, 46, 57, 66, 67]. The benefit of these consensus algorithms, compared to Proof-of-Work-based algorithms, is that they have low computational costs, low transaction processing times, and high transaction throughput. This is already exemplified in 2002 by BFS, a fault-tolerant version of the networked file system [38], which could already handle hundreds of transactions per second [13, 14].

In this tutorial, we will provide a deep dive into consensus protocols with a focus on data management. To do so, we take an in-depth look at Byzantine fault-tolerant consensus protocols, the main technique powering permissioned blockchains.

Our tutorial will focus on three avenues. First, we look at the theoretical framework in which permissioned blockchains operate. Then, we look at practical high-performance consensus protocols and at current developments. Finally, we look at the challenges in the design of future-proof high-performance permissioned blockchain systems that can deal with huge amounts of data, and provide our vision on future developments. These avenues are detailed in the following three sections. Finally, in Section 5, we provide a summary of the topics addressed during the tutorial.

2 Fault-tolerant distributed computing

Blockchains are, at their basis, fully replicated distributed systems that aim to maintain data consistency. The well-known CAP Theorem puts restrictions on the types of failures these blockchains can deal with while guaranteeing continued services [8, 9, 30]. The CAP Theorem puts rather general limitations on the design of blockchains, however. More specific limitations are also known, as the Byzantine consensus problem and other related problems, such as the Byzantine
agreement problem and the interactive consistency problem, have received considerable attention.

It is well-known that the Byzantine agreement problem can only be solved when using synchronous communication [28, 49, 65]. In a synchronous environment with n replicas of which f are Byzantine (e.g., malicious), Byzantine agreement requires that n > 3f [20, 21]. When strong cryptographic primitives are available, this can be improved to n > f [23, 47, 58] (although practical systems will still require n > 2f). Additionally, bounds on the amount of communication and the quality of the network are known [18, 20–23, 27].

3 Practical consensus protocols

Having provided a theoretical background, we make the step toward detailing practical consensus protocols. We do so by a full coverage of the Practical Byzantine Fault Tolerant (PBFT) consensus protocol of Castro et al. [12–14]. Next, we also look at the lineage of consensus protocols that refine and improve PBFT. This detailed overview will cover many of the practical consensus protocols currently in use and, simultaneously, also covers recent developments. Our coverage will include protocols such as PBFT, HotStuff [70], Zyzzyva [5, 44, 45], FaB [48], SynBFT [1], RBFT [6], PoE, and MultiBFT [34].

4 Challenges and our Vision

As outlined above, the approaches taken by permissionless and permissioned blockchains towards fault-tolerant replication have benefits and practical use cases. Unfortunately, fault-tolerant replication is challenged by the scalability and performance required by many modern big-data-driven applications. In specific, we see that there is no obvious way to scale up fault-tolerant replication: adding more replicas will only increase the cost of replication and decrease the throughput of the system, even when using the most efficient consensus protocols. We will close our tutorial by discussing recent steps toward the design of new fault-tolerant architectures that step away from the full-replicated nature of blockchains, this to increase scalability and the ability to serve big-data-driven applications. In specific, we will look at two low-level techniques, cluster-sending [39] and delayed-replication, and at a high-performance sharded and geo-scale aware architecture that is enabled by these techniques.

5 Outline of the Tutorial

The tutorial starts with a general-purpose introduction to blockchains from the perspective of data management. Next, each of three avenues discussed in Section 2–4 will receive attention. In specific, we plan to explore the following outline:

**Blockchains and Data Management**
1. What are blockchain and why should I care about them.
2. Blockchains from the perspective of data management.
3. The connection between blockchains and consensus.
4. Permissioned and permissionless blockchain systems.

**Overview of Resilient Distributed Systems**
1. What are distributed and resilient systems.
2. General limitations of distributed systems: the CAP Theorem.
3. Failure models: crashes versus Byzantine failures.
5. Consensus, Byzantine agreement, and interactive consistency.
6. Theoretical results on resilient distributed systems.

**Byzantine Fault-Tolerance in Practice**
1. The Practical Byzantine Fault Tolerant consensus algorithm in detail.
2. Improving on PBFT: digital signatures versus symmetric encryption, multi-round reasoning, optimistic execution, randomization, threshold signatures, trusted components, parallelization, and other techniques.

**Challenges and our Vision**
1. Towards sharding and geo-scale designs.
2. Specialized designs via cluster-sending.
3. Specialized designs via Byzantine learners.

This tutorial is based on the outline of our upcoming book on fault-tolerant transaction processing on blockchains [35].

**References**


Suyash Gupta, Jelle Hellings, Sajjad Rahnama, and Mohammad Sadoghi