Database and Distributed Computing Fundamentals for Scalable, Fault-tolerant, and Consistent Maintenance of Blockchains

Sujaya Maiyya Victor Zakhary Divyakant Agrawal Amr El Abbadi
UC Santa Barbara
Santa Barbara, CA 93106
{sujaya_maiyya, victorzakhary, divyagrawal, elabbadi}@ucsb.edu

ABSTRACT

Bitcoin is a successful and interesting example of a global scale peer-to-peer cryptocurrency that integrates many techniques and protocols from cryptography, distributed systems, and databases. The main underlying data structure is blockchain, a scalable fully replicated structure that is shared among all participants and guarantees a consistent view of all user transactions by all participants in the cryptocurrency system. In this tutorial, we discuss the basic protocols used in blockchain, and elaborate on its main advantages and limitations. To overcome these limitations, we provide the necessary distributed systems background in managing large scale fully replicated ledgers, using Byzantine Agreement protocols to solve the consensus problem. Finally, we expound on some of the most recent proposals to design scalable and efficient blockchains. The focus of the tutorial is on the distributed systems and database technical aspects of the recent innovations in blockchains.

PVLDB Reference Format:
Sujaya Maiyya, Victor Zakhary, Divyakant Agrawal, Amr El Abbadi. Database and Distributed Computing Fundamentals for Scalable, Fault-tolerant, and Consistent Maintenance of Blockchains. *PVLDB*, 11 (12): 2098 - 2101, 2018.
DOI: https://doi.org/10.14778/3229863.3229877

Keywords

Blockchain, Distributed Consensus, Byzantine Faults

1. INTRODUCTION

Bitcoin [16] is considered the first successful global scale peer-to-peer cryptocurrency. The Bitcoin protocol explained by the *mysterious Nakamoto* allows financial transactions to be transacted among participants without the need for a trusted third party, e.g., banks, credit card companies, or PayPal. Bitcoin eliminates the need for such a trusted third party by replacing it with a distributed ledger that is fully replicated among all participants in the cryptocurrency system. This distributed ledger is referred to as *blockchain*.

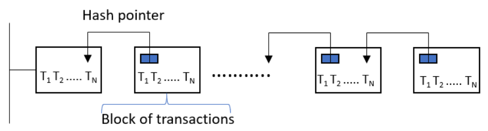

Figure 1: Blockchain is a chain of blocks of transactions linked by hash pointers.

Blockchain, as shown in Figure 1, is a secure linked list of blocks containing financial transactions that occur in the system and linked by hash pointers. The main challenge that Bitcoin addresses is to maintain a consistent view of this replicated blockchain in a secure and fault-tolerant manner in a *permission-less* setting and in the presence of malicious participants. Unlike *permissioned* settings where all the participants in the system are known *a priori*, a permission-less setting allows participants to freely join and leave the system without maintaining any global knowledge of the number of participants. To address these challenges, Bitcoin builds on foundations developed over the last few decades from diverse fields [17], but primarily from the fields of **cryptography** [1, 18], **distributed systems** [4, 11, 12] and **data management** [2, 15, 20] as illustrated in Figure 2. Figure 2 shows the space of techniques that are used in Bitcoin to implement a permission-less decentralized payments. Digital signatures and hashing are the cryptographic foundations that are used in Bitcoin to support distributed transactions stored in a ledger that is replicated across globally distributed sites in the presence of malicious faults.

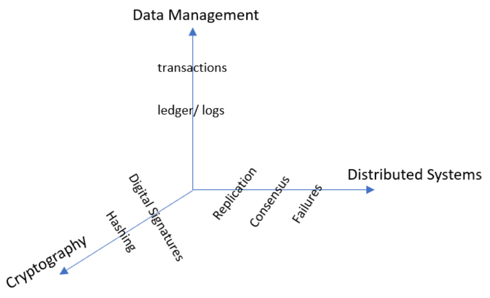

Figure 2: The space of techniques that are used to implement permission-less decentralized cryptosystems.

In spite of its current success, Bitcoin suffers from scalability performance limitations represented by the number of

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3229877

transactions executed per second, especially when compared with commercially successful On Line Transaction Processing (OLTP) financial systems, such as credit card companies. Bitcoin uses a notion of *miners* who need to perform a computationally challenging *Proof of Work (PoW)* puzzle before they can add any block of transactions to the replicated blockchain. Since the PoW puzzle is computationally hard, very few miners can successfully solve the puzzle, and hence a successful miner can add a block to the blockchain and be guaranteed, with very high probability, to be unique. Many concerns have been raised about the wasted massive energy requirements to *mine* one Bitcoin block. In addition, the difficulty of Bitcoin's Proof of Work (PoW) discourages *miners* from mining independently and pushes them to form few powerful *mining cartels*. Concentrating most of the mining power in few hands, or pools, risks to derail the whole idea of decentralization.

This mining approach to determine the process eligible to add a new block to the block chain is in contrast to the distributed systems approach, that has been promoting the use of Byzantine Agreement or consensus, which is efficient and more egalitarian. In fact, consensus protocols such as Paxos have been quite successful in recent years in laying the foundations of large global scale data management system. Unfortunately, Paxos has many limitations, especially from a global cryptocurrency point of view, including the requirement of a permissioned setting, and that participants can only fail by crashing. An alternative to Paxos that tolerates malicious failures is Practical Byzantine Fault-Tolerance (PBFT) [4]. Although it tolerates malicious failures, PBFT still requires a permissioned setting, and requires a large number of message exchanges, hence does not scale to the large number of participants expected in modern day permission-less cryptocurrencies. Recently, the security and distributed systems communities have been aggressively exploring alternative scalable solutions. These systems, including Byzcoin [10], Elastico [13], Algorand [9], etc., attempt to solve the Bitcoin protocol shortcomings in permission-less settings using efficient and practical solutions that integrate cryptographic and consensus mechanisms.

In this tutorial, our goal is to present to the database community an in-depth understanding of state-of-the-art solutions for efficient scalable blockchains. We progress towards this goal by starting from a detailed description of the protocols and techniques underlying the design of Bitcoin. Since most recent innovations in blockchain design depend critically on consensus protocols in malicious settings, we outline the basic foundations of distributed fault-tolerant consensus protocols. This is followed by a discussion of the most recent proposals to solve the blockchain design problem using scalable fault-tolerant solutions.

2. TUTORIAL OUTLINE

2.1 Blockchain and Nakamoto's Consensus

The problem of *double spending* is one of the main challenges of the Bitcoin protocol. Double spending happens when a coin owner signs *two* concurrent transactions trying to spend the same coin twice. This *concurrency* anomaly can easily be prevented if transactions are serialized [2]. Bitcoin relies on a network of *miners* to achieve serializability. Every financial transaction is broadcast to all sites/miners in the system. Each miner receives these digitally signed transactions and groups them into a new block. A miner "mines" to add this block to the blockchain but before doing that, these transactions need to be verified. The verification process ensures that the transactions in a new block are neither conflicting with each other nor conflicting with other transactions in any preceding block in the current blockchain.

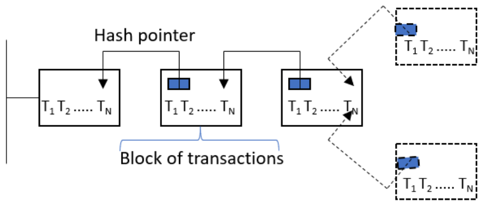

Figure 3: A fork in the blockchain.

To add a block to the blockchain, miners need to perform a computationally challenging *Proof of Work (PoW)* puzzle before they can add their block of transactions to the replicated blockchain. Since the PoW puzzle is computationally hard, very few miners can successfully solve the puzzle, hence a successful miner can add a block to the blockchain and be guaranteed, with very high probability to be unique. Serializability is ensured by adding verified blocks one at a time. After a block is added, the miner who "mined" this block broadcasts it to all other miners. Miners are incentivized to accept the first received block, add it to their copy of the blockchain, and immediately start mining for the next block.

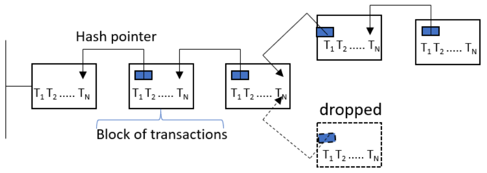

Figure 4: Miners join the longest chain to resolve forks.

The difficulty of PoW aims to minimize the probability that two miners solve the puzzle at the same time. However, with low probability, more than one miner can concurrently reach a solution to the puzzle causing a fork in the blockchain as shown in Figure 3. Forks are a violation of serializability and cause double spending as transactions in the two added blocks can have conflicts with each other. Forks divide the miners network into two groups and each group independently mines for the next block. Once a block is added to either of the fork branches, miners in both groups join the longest chain and drop the other branch of the fork as shown in Figure 4. Transactions in the dropped block are considered *aborted* and need to be resubmitted to the network again. As blocks can be dropped after being added to the blockchain, transactions should not be considered *committed* unless their blocks are buried deep in the blockchain, typically a depth of 6 blocks. A block that is buried in the chain for more than 6 blocks is guaranteed with very high probability not to be dropped, and hence transactions in that block will probably not be aborted. However, if 51% of the mining power maliciously collude, they can redo the whole blockchain risking the safety of the network and causing committed transactions to be dropped even if they were deeply buried in the blockchain. This is widely known as the *51% attack* [6].

2.2 Traditional Consensus and PBFT

Consensus and *Byzatine Agreement (BA)* are well studied problems that were first proposed by Lamport, Shostak and Pease in 1982 [12]. Any solution to the BA problem tries to reach agreement among a well defined set of processes on a single value. The distributed systems community has extensively explored this problem in both synchronous and asynchronous systems. In an asynchronous system, the Fisher Lynch and Patterson (FLP) impossibility result states that consensus is not guaranteed to terminate in the presence of even a single crash failure [8]. This led to many BA protocols for synchronous systems, where the lower bound requires that the number of maliciously faulty processes is at most one third of the total number of processes. Synchronous BA protocols require multiple rounds of communication and extensive message passing. On the other hand, several efficient asynchronous BA protocols have been developed based on Lamport's Paxos protocol [11]. The main challenge such asynchronous protocols face is that they do not guarantee termination. However, many systems have been designed that depend on Paxos, and have been practically successful [3, 5]. Paxos, however, assumes that processes may only fail by crashing. In a cryptocurrency setting, processes may act in a malicious manner. In 1999, Castro and Liskov proposed the Practical Byzantine Fault Tolerance (PBFT) algorithm [4], which is similar to Paxos in that it uses a small number of message rounds and assumes an asynchronous system. However, it tolerates malicious faults. During the tutorial, we will provide a general overview of the consensus problem and a high level description of various BA protocols. In particular, we will provide a detailed description of PBFT, given its particular relevance to many recent cryptocurrency proposals, as discussed in the next section. We will also highlight the main advantages and limitation of these distributed BA protocols.

2.3 Permission-less Blockchains

This section discusses some recent state-of-the-art proposals in permission-less crytocurrency, *i.e.*, any server can attach (or detach) itself from the network of servers that are mining the currency. Most recent work is focused on tackling the performance limitation posed by Bitcoin. Bitcoin executes 7 transactions per second [7] whereas trusted, centralized systems such as Visa execute thousands of transactions per second [7]. One of the main reasons contributing to the low transaction rate is *the possibility forking* of the blockchain in Bitcoin; since the system allows concurrent miners to add blocks, there may be two (or more) blockchain forks with the same length. Bitcoin mitigates this by waiting for the chain to grow by a certain length (6 blocks) before a block is considered to be committed. To mitigate this bottleneck, many attempts have focused on devising novel ways to avoid forking of blockchains.

ByzCoin [10] is a Bitcoin-like cryptocurrency that uses traditional PBFT [4] consensus protocol to obtain better performance compared to Bitcoin. ByzCoin incorporates PBFT along with collective signing (CoSi) [19] in order to dynamically form a small consensus group to agree upon the next block to be added to the blockchain. Consensus groups are formed based on a moving window of the most recent miners based on their hash power. CoSi is a 4 phase protocol which uses a tree-based structure to collectively validate transactions. ByzCoin integrates phases of CoSi, which validate transactions, along with PBFT consensus rounds, which reach an agreement on a block, thus reducing the overall number of messages. Thus, by building a smaller consensus group that runs a modified PBFT-like protocol to obtain agreement on the next block to be added, ByzCoin achieves confirmation latency of 15-20 seconds; a significant improvement in comparison to Bitcoin.

Elastico [13] is another distributed agreement protocol that strives to achieve linear performance increase proportional to the total computation power of the system. The key idea presented in Elastico is to split all the servers in the system into smaller sized groups called *committees*, each of which is responsible for processing a subset or a *shard* of transactions. Every committee runs classical PBFT to agree on a set of transactions; these transactions are sent to a special committee called the *final committee*, which then aggregates the transactions obtained from different committees and runs another round of PBFT to make a global, final decision on the next block that is appended to the blockchain. Committee assignment has to be random to avoid targeted attacks by malicious nodes. Elastico uses *identities* that are created based on PoW and epoch-randomness to guarantee the required randomness. Since transaction verification is parallelized across different committees, Elastico is able to obtain higher throughput than Bitcoin.

After discussing some of the distributed ledger solutions that use PBFT as a way to achieve Byzantine consensus, we now explore Algorand [9] which proposes a novel Byzantine Agreement protocol (BA*). As with the previously discussed approaches, Algorand also uses smaller committees to obtain consensus. However, in contrast, it uses a different committee for each *step* of the BA* protocol. It uses a *cryptographic sortition* method to randomize committee member selection. Each committee member broadcasts its block with some priority and all nodes try to reach consensus on the block with the highest priority by voting for that block. With this new byzantine agreement protocol, Algorand attains throughput 125x of Bitcoin's throughput.

The overall takeaway from the above protocols is that they all strive towards avoiding forks of the blockchain by reaching *finality* in consensus on the new block to be appended. To reach immediate commitment, all these protocols significantly reduce the size of the consensus group. Each protocol uses different cryptographic techniques to guarantee randomness in committee formation, and overall, achieve better throughput than Bitcoin. During the tutorial, we highlight the main advantages and limitations of each of these approaches.

3. TUTORIAL INFORMATION

This is a **three hours** tutorial targeting researchers, designers, and practitioners interested in large scale transaction support in permission-less blockchain consensus-based research. The **target audience** with basic background about distributed consensus should benefit the most from this tutorial. For the general audience and newcomers, the tutorial explains the design space of distributed consensus and permission-less blockchains. This tutorial differs from previous tutorials on the same topic in database conferences, especially C. Mohan [14], where he explicitly states that the scope of his tutorial ”is general in nature without getting into the nitty gritty of, e.g., cryptographic algorithms or the distributed consensus protocols”. Furthermore, Mohan's tutorial ”only discuss(es) permissioned/private blockchains and not permission-less ones”. This tutorial, in contrast, focuses on *permission-less* blockchains and discusses the distributed protocols and their interaction with user transactions in detail. The cryptographic algorithms will be specified and their properties discussed to help understanding the distributed systems and database implications.

4. BIOGRAPHICAL SKETCHES

Sujaya Maiyya is a PhD student at the University of California at Santa Barbara. Her current research work is targeted towards building consensus and commitment protocols in a byzantine environment and to integrate these protocols into large scale data management systems.

Victor Zakhary is a PhD student at the University of California at Santa Barbara. His current research work is in the areas of data placement for geo-replicated databases and for distributed caching services to achieve low access latency and dynamically handle data access skewness.

Divyakant Agrawal is a Professor of Computer Science at the University of California at Santa Barbara. His current interests are in the area of scalable data management and data analysis in cloud computing environments, security and privacy of data in the cloud, and scalable analytics over big data. Prof. Agrawal is an ACM Distinguished Scientist (2010), an ACM Fellow (2012), an IEEE Fellow (2012), and an AAAS Fellow (2016).

Amr El Abbadi is a Professor of Computer Science at the University of California, Santa Barbara. Prof. El Abbadi is an ACM Fellow, AAAS Fellow, and IEEE Fellow. He was Chair of the Computer Science Department at UCSB from 2007 to 2011. He has served as a journal editor for several database journals and has been Program Chair for multiple database and distributed systems conferences. Most recently Prof. El Abbadi was the co-recipient of the Test of Time Award at EDBT/ICDT 2015. He has published over 300 articles in databases and distributed systems and has supervised over 30 PhD students.

5. ACKNOWLEDGEMENT

This work is funded by NSF grants CNS-1703560 and CNS-1649469.

6. REFERENCES

[1] A. Back et al. Hashcash-a denial of service counter-measure. 2002.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency control and recovery in database systems. 1987.

[3] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.

[4] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[6] M. Conti, C. Lal, S. Ruj, et al. A survey on security and privacy issues of bitcoin. *arXiv preprint arXiv:1706.00916*, 2017.

[7] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[8] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[10] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 279–296, 2016.

[11] L. Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[12] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[13] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[14] C. Mohan. Tutorial: blockchains and databases. *PVLDB*, 10(12):2000–2001, 2017.

[15] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz. Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems (TODS)*, 17(1):94–162, 1992.

[16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[17] A. Narayanan and J. Clark. Bitcoin's academic pedigree. *Communications of the ACM*, 60(12):36–45, 2017.

[18] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[19] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities” honest or bust” with decentralized witness cosigning. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 526–545. Ieee, 2016.

[20] G. Weikum and G. Vossen. *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. Elsevier, 2001.