The Road to Scalable Blockchain Designs

SHEHAR BANO, MUSTAFA AL-BASSAM, AND GEORGE DANEZIS

Shehar Bano is a Postdoctoral Researcher at University College London. Her research interests center on networked systems, particularly in the context of security and measurement. She received her PhD from the University of Cambridge in 2017 where she was an Honorary Cambridge Trust Scholar, and was awarded the Mary Bradburn Scholarship for her research work. s.bano@ucl.ac.uk

Mustafa Al-Bassam is a PhD student at University College London, working on scalable distributed ledger technology and peer-to-peer systems. He received a BSc in computer science from King's College London, where his final-year project focused on public-key infrastructure implemented with smart contracts. mustafa.al-bassam.16@ucl.ac.uk

George Danezis is a Professor of Security and Privacy Engineering at University College London and a Turing Faculty Fellow, where he heads the Information Security Research group. He researches privacy-enhancing technologies, decentralization, and infrastructure security and privacy. In the past he worked at Microsoft Research, KU Leuven, and the University of Cambridge, where he also studied. g.danezis@ucl.ac.uk

Bitcoin has become centralized and slow due to the inherent limitations of its blockchain. A number of alternative blockchain designs have been proposed to address these issues. *Off-chain* solutions allow for small and frequent transactions to take place over low-tier blockchain instances, parallel to and backed by the main blockchain. *On-chain* solutions directly modify the blockchain design to support high performance. We focus on the latter and summarize and discuss recent approaches to on-chain scaling of blockchains.

Despite being founded on the ideals of openness and freedom of information, the Internet has become increasingly centralized, enabling a small number of big players to control who can access information. A number of proposals have emerged to counterbalance this trend such that information storage and processing is not concentrated in any single entity. Of these, blockchains are particularly promising, capturing the attention of popular media, research, and policy communities alike.

A blockchain is an immutable and decentralized database that facilitates transparent and auditable management of data. It first gained traction as the underlying technology of Bitcoin [8] proposed in 2009, but it has since independently evolved thanks to its properties of resiliency, integrity, and transparency. Yet Bitcoin suffers from scalability issues that impede its wider adoption. Over the past year, major divisions have emerged in the Bitcoin community over how Bitcoin should scale, as blocks of transactions have reached capacity, resulting in transaction fees skyrocketing. At the core of the debate is a simple tradeoff between scalability and centralization: the bigger the blockchain, the fewer devices will have the capacity to store and audit the full blockchain, leading to the network becoming more centralized.

Some argue for using the Bitcoin blockchain as a settlement network for large transactions only—with smaller transactions being handled by payment hubs off the blockchain (off-chain scaling), while others argue for increasing the capacity of the blockchain itself for all types of transactions (on-chain scaling). As a result of the fallout, proponents of on-chain scaling recently forked the Bitcoin blockchain to create their own network called Bitcoin Cash. In this article, we provide an overview of key themes and options for on-chain scaling of blockchains.

Functional Components of a Blockchain

A blockchain serves as a decentralized database—or a distributed *ledger*—representing a consensus of synchronized, distributed, and replicated data (called *blocks*, representing sets of *transactions*). It is internally implemented as a linked list in which pointers to previous blocks have been replaced with cryptographic hash pointers (Figure 1). A pointer is simply the hash of some information (e.g., the previous block in this case) and serves to identify the information, as well as to verify its integrity. Each block in the blockchain contains a hash of the previous block and information specific to the current block. The resulting hash chain ensures each block implicitly verifies integrity of the entire blockchain before it. Thus,

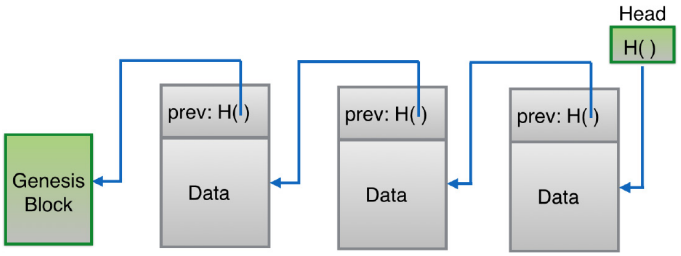

Figure 1: A blockchain is implemented as a linked list of hash pointers.

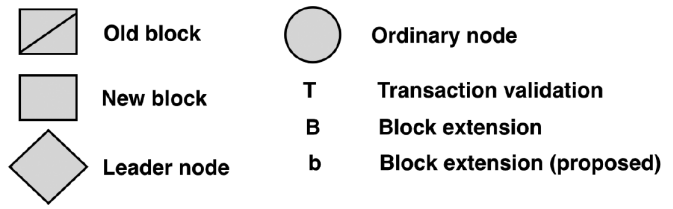

Figure 2: Legend used in the figures

a blockchain acts as a tamper-evident log where data can be appended to the end of the log, and tampering with previous data in the log is detectable. A blockchain has two key functional components: *transaction validation* and *extending the blockchain*, which we discuss individually.

Transaction Validation

A *transaction* specifies some transformation on the state of the ledger. These transactions, subject to passing validity and verification checks, are included in a candidate block (a set of transactions) to be appended to the blockchain. As a concrete example, we describe a typical (simplified) Bitcoin transaction involving transfer of money from payer(s) to payee(s). The payers and payees are identified by their public keys, and the payer digitally signs the transaction, involving value they control encoded in previous blocks of the blockchain. Nodes in the Bitcoin network must perform a set of checks before accepting a transaction as valid according to the rules of the network. First, they must check that the transaction is well-formed. Second, they must verify that the payer is authorized to conduct this transaction by checking that its digital signature corresponds to the public key of the payer. Third, the nodes must verify that the sum of outputs is lower than the sum of inputs—payers cannot pay out more than they own, but a transaction fee can be included in the payment. Finally, nodes must ensure that none of the inputs is being double-spent. This can be verified by traversing back in the blockchain to when the input value was created, and then traversing forward all the way to the current transaction—ensuring along the way that the input has not been previously spent.

Extending the Blockchain

In reality, transaction outputs (e.g., *X bitcoins*) have no physical existence: the fact that Bob owns a transaction output corresponds to the fact that a majority of the nodes believe this to be the case. Agreement between nodes on how to extend the blockchain is reached through a collaborative process called *consensus*.

Consensus. The problem of consensus in the presence of faulty or malicious nodes has seen extensive study in the distributed systems community, long before it was revisited in the context of blockchains. In a network with n honest nodes that each receive input values and share them with rest of the network, the consensus protocol enables agreement between all n honest nodes on the set of input values generated by honest nodes. In the Bitcoin context, where nodes broadcast transactions as part of a peer-to-peer (p2p) network, nodes need to reach consensus on exactly which transactions took place and in what order—that is, the nodes must agree on the state of the blockchain.

Forks. Consensus is challenging because nodes might have different views of the blockchain (*forks*) due to latency in propagation of transactions over the p2p network, nodes randomly failing, and malicious nodes trying to suppress valid transactions and push invalid transactions to the blockchain. Forks defy consensus, so a mechanism is needed to resolve conflicts and get a majority of the nodes to agree on the state of the blockchain.

Leader. Consensus protocols typically rely on a *leader*. The leader is responsible for coordinating with other nodes to reach consensus, and for appending a final, committed value to the blockchain. The leader is usually effective only for a period of time called an *epoch*, after which or upon a fault, a new leader is elected. A crucial property of a leader is that it should behave honestly. This is important because even though a blockchain is tamper-evident by design, appending bad blocks to the blockchain will likely result in forks leading to wasted system resources in getting nodes to re-converge to a previous valid blockchain view. Honest leader behavior is usually enforced via incentivization and auditability.

Figure 2 shows visual representations of some of these concepts, which are later used to explain various design themes.

Bitcoin and Its Scalability Issues

The advent of Bitcoin in 2009 sparked interest in blockchains, the technology that lies at its foundation. Being the predecessor of the myriad blockchain variations that subsequently emerged, it is useful to understand the blockchain scalability problem in the context of Bitcoin.

The Bitcoin Blockchain. Bitcoin is a p2p network where any node can join and become part of the network. If a node receives

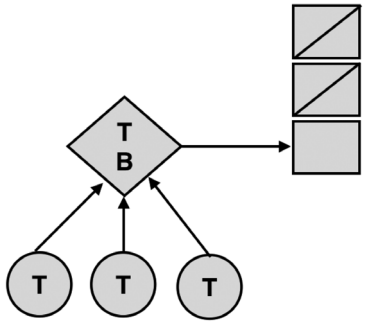

Figure 3: The Bitcoin blockchain model

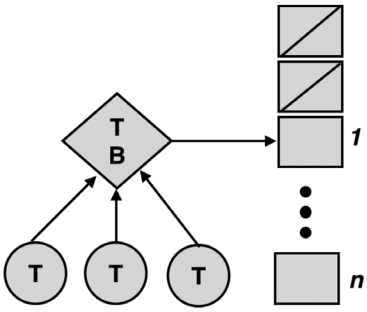

Figure 4: Multiple blocks per leader

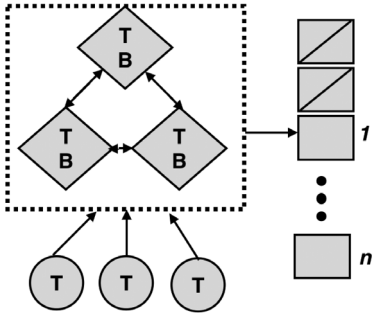

Figure 5: Collective leaders

a new block, it broadcasts it to rest of the network (Figure 3). While all nodes listen to and broadcast blocks, only leader nodes can append information to the blockchain. To stop dishonest leaders from bringing the system to a stall—for example, by creating frequent forks—the leader for each epoch is chosen randomly via proof-of-work. This involves solving a hash puzzle—also called *mining*, which is why Bitcoin leaders are referred to as *miners*. If a miner gets lucky by finding a solution to the hash puzzle, it proposes the next block to append to the blockchain. To incentivize miners to solve hash puzzles and propose next blocks, successful miners are rewarded by allowing them to pay some amount to themselves (a *block reward*, which diminishes over time) or by keeping some part of the transaction output amount as the *transaction fee*.

Blockchain Scalability. Two metrics are directly related to blockchain scalability: transaction throughput (the maximum rate at which the blockchain can process transactions) and latency (time to confirm that a transaction has been included in the blockchain). While previous work has identified additional metrics [4], throughput and latency are bottleneck issues and more challenging to address from a research perspective. Bitcoin's transaction throughput is a function of its block size and inter-block interval. With its current block size of 1 MB and 10 minute inter-block interval, the maximum throughput is capped at about seven transactions per second; and a client that creates a transaction has to wait for at least 10 minutes on average to be sure that the transaction is included in the blockchain. In contrast, mainstream payment-processing companies like Visa confirm transactions within a few seconds and have a high throughput of up to 24,000 transactions per second [9].

Current research is focused on developing solutions to significantly improve blockchain performance while retaining its decentralized nature. Reparametrization of Bitcoin's block size and inter-block interval can improve performance to a limited extent—estimated by a recent study [4] at 27 transactions per second and 12 seconds, respectively. However, significant improvement in performance requires fundamental redesign of the blockchain paradigm.

Redesigning Blockchains for Scalability

We now take a look at key design schemes that have been developed to improve blockchain scalability. Our scope is restricted to approaches targeting the blockchain core design (on-chain solutions) rather than techniques that delegate trust to parallel off-path blockchain instances such as sidechains [1] (off-chain solutions). The list of themes and example systems is not meant to be comprehensive but, rather, indicative of some major ways in which this subject has been approached, and to provide a high-level roadmap for future research efforts. Figure 2 shows the basic building blocks of the design themes that we discuss in the following sections.

Multiple Blocks per Leader

Bitcoin-NG [5] shares Bitcoin's trust model but decouples leader election (performed randomly and infrequently via proof-of-work) from transaction serialization (Figure 4). However, unlike Bitcoin where the leader can only propose one block to append to the blockchain, Bitcoin-NG divides time into epochs, and a leader can unilaterally append multiple transactions to the blockchain for the duration of its epoch, which ends when a new leader is elected. There are two kind of blocks in Bitcoin-NG: keyblocks and microblocks. Keyblocks contain a solution to the puzzle and are used for leader election. Keyblocks contain a public key that is used to sign subsequent microblocks generated by the leader. Every block contains a reference to the previous microblock and keyblock. A fee is distributed between the current leader (40%) and the next leader (60%).

Similar to Bitcoin, forks are resolved by extending the longest branch aggregated over all keyblocks. Note that microblocks do not contribute to the length of a branch since these do not contain proof-of-work. To penalize a leader that creates forks in the generated microblocks, a subsequent leader can insert a special poison transaction after its keyblock that contains the header of the first block in the pruned branch as a proof-of-fraud. This invalidates the malicious leader's reward, a fraction of which is paid to the reporting leader. Forks can also occur when a new leader has been elected but the previous leader has not yet heard

about it and continues to generate microblocks. However, such forks are resolved as soon as the announcement of the new leader election reaches all the nodes.

Collective Leaders

This scheme employs multiple leaders to collectively and quickly decide if a block should be added to the blockchain (Figure 5). ByzCoin [6] replaces Bitcoin's probabilistic transaction consistency guarantees with strong consistency by extending Bitcoin-NG (see preceding section) to achieve high transaction throughput. This has the advantage that a transaction submitted by a client will be added to the blockchain, and the blockchain remains fork-free since all leaders instantly agree on block validity. ByzCoin modifies how Bitcoin-NG generates keyblocks: a group of leaders, rather than a single leader, generates a keyblock followed by microblocks. The leader group is dynamically formed by a window of recent miners. Each miner has voting power proportional to the number of mining blocks it has in the current window, which is its hash power. When a new miner solves the puzzle, it becomes a member of the current leader group, which moves one step forward, ejecting the oldest miner. ByzCoin uses the same incentive model as Bitcoin, but the remuneration is shared between members of the leader group in proportion to their shares.

The leader group is organized into a communication tree where the most recent miner (the leader) is at the root. The leader runs a modified version of the Practical Byzantine Fault Tolerance (PBFT) protocol [3] with linear messaging complexity to generate a collective signature that proves that at least two-thirds of the consensus group members witnessed and attested the microblock. A node in the network can verify in $O(1)$ that a microblock has been validated by the consensus group. This design addresses a limitation of Bitcoin-NG where a malicious leader can create microblock forks: in ByzCoin this would require a two-thirds majority of leader group members to be malicious. Moreover, Bitcoin-NG suffers from a race condition where an old leader who has not yet heard about the new leader may continue to incorrectly mine on top of older microblocks. In ByzCoin, leader group members ensure that a new leader builds on top of the most recent microblock.

Parallel Blockchain Extension

As shown in Figure 6, in this approach multiple leaders extend in parallel different parts of the blockchain (e.g., represented as a graph of transactions). Bitcoin has a linear process of extending the blockchain: miners try to solve the puzzle, and the one that finds a solution appends the next block. The framework proposed by Boyen, Carr, and Haines [2] parallelizes this process by forgoing the concepts of "blocks" and "chain" in favor of a graph of cross-verifying transactions. Each transaction validates two

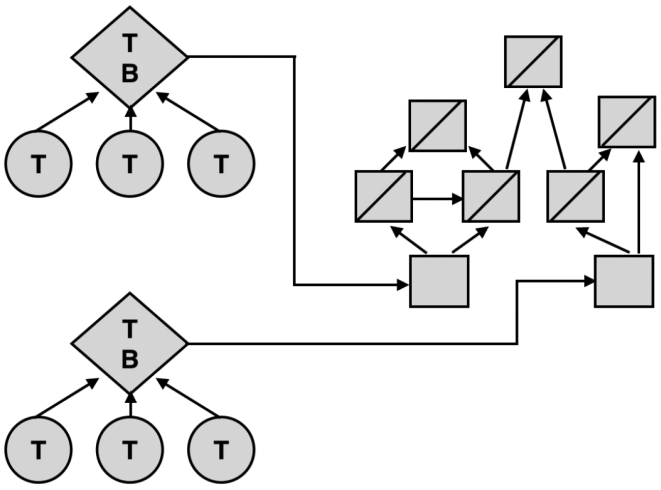

Figure 6: Parallel blockchain extension

previous transactions (its parents) and contains some payload (e.g., cryptocurrency) and proof-of-work.

A transaction can be potentially validated by multiple children nodes. Additionally, each transaction also carries a reward to be collected by the transaction that validates it. The value of the reward decreases as more nodes directly or indirectly validate it, so new nodes have more incentive to validate recent transactions. The system has been shown to converge, meaning that at some point there is a transaction that connects to (and thus implicitly verifies) all transactions before it. As a result of this graph structure, miners can extend different branches of the transactions graph in parallel. Normal (non-miner) nodes in the system verify transactions as they receive them. In addition to standard checks on the correctness of proof-of-work and structural validity of the transaction and its parents, the node also checks that the transaction is not a double-spend by accepting as valid the well-formed transaction that has the largest amount of work attached to it.

Sharding Transactions

Elastico [7] partitions nodes into groups called committees, and each committee manages a subset (shard) of transactions. In Figure 7, the top shard handles the first 10 transactions, while the bottom shard handles the next 10. Within a committee, nodes run a Byzantine consensus protocol (e.g., PBFT) to agree on a block of transactions. If the block has been signed by enough nodes, the committee sends it to a final committee. The final committee collates sets of transactions received from committees into a final block, runs a Byzantine consensus protocol between its members to get agreement on extending the blockchain, and broadcasts the appended block to other committees.

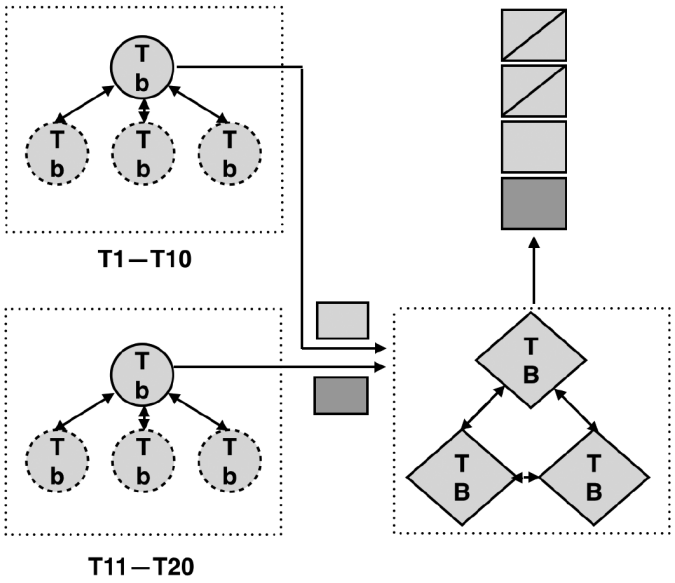

Figure 7: Sharding transactions

The system operates in epochs: the assignment of nodes to committees is valid only for duration of the epoch. At the end of the epoch, the nodes solve a puzzle seeded by a random string generated by the current final committee, and sends the solution to its next final committee. As a result, in each epoch a node is paired with different nodes in a committee and manages a different set of transactions. The number of committees scales linearly to the amount of computational power available in the system, but the number of nodes within a committee is fixed. Consequently, as more nodes join the network, the transaction throughput increases without adding to latency, since messages needed for consensus are decoupled from computation and broadcast of the final block to be added to the blockchain.

Conclusion

We framed the blockchain scalability problem and presented an overview of key approaches for on-chain scalability of blockchains. This revealed design patterns that can be used to compose scalable blockchains. Indeed, some patterns have already been used in this way: ByzCoin builds *collective leadership* on top of Bitcoin-NG's *multiple-blocks-per-leader* design as discussed. *Collective leadership* is a useful primitive to enforce honest behavior (and to avoid forks) by spreading out accountability and stakes across multiple leaders. *Sharding* speeds up transaction throughput by partly delegating consensus to smaller groups where classical BFT protocols can be effectively run, and making a leader group (that potentially also runs a BFT consensus protocol among leaders) responsible for extending the blockchain. It might be possible to replace consensus in the leader group with *collective leadership,* which has lower messaging complexity than the original PBFT protocol and a higher degree of trust. The idea of *parallel blockchain extension* can be combined with *sharding* such that the blockchain exists as partially connected trees on separate shards. Blocks that are part of separate trees are connected only when there is a transaction that consumes blocks managed by different shards.

Mining centralization is a well-known problem in Bitcoin: the biggest miners have built up a large advantage in how the blockchain grows. Systems like Bitcoin-NG and ByzCoin that inherit Bitcoin's mining-based consensus suffer from the same problem of centralization and favoring the biggest miners. Broadly, there has been a shift from Bitcoin's slow mining-based leader election to novel compositions or variations of classical consensus protocols. The latter cannot be directly employed in blockchains as these were originally written for a LAN setting, and their throughput decreases with the number of nodes. It will be interesting to see what new designs emerge and how existing consensus protocols are repurposed to operate in a decentralized WAN setting and in various threat models. This research direction revitalizes the field of Byzantine consensus and has the potential to make it relevant to widely deployed peer-to-peer systems.

Acknowledgments

The authors are supported in part by EPSRC Grant EP/M013286/1 and the EU H2020 DECODE project under grant agreement number 732546, as well as The Alan Turing Institute.

References

[1] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," Blockstream.com, 2014: https://www.blockstream.com/sidechains.pdf.

[2] X. Boyen, C. Carr, and T. Haines, "Blockchain-Free Cryptocurrencies: A Rational Framework for Truly Decentralised Fast Transactions," Cryptology ePrint Archive, Report 2016/871, 2016: https://eprint.iacr.org/2016/871.

[3] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)*, USENIX Association, 1999, pp. 173–186: http://bit.ly/2fjh6dN.

[4] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. G. Sirer, D. Song, R. Wattenhofer, "On Scaling Decentralized Blockchains," 3rd Workshop on Bitcoin and Blockchain Research, 2016: http://bit.ly/2xfz5Jl.

[5] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol," in *Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation (NSDI '16)*, pp. 45–59: http://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf.

[6] 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 *Proceedings of the 25th USENIX Security Symposium (USENIX Security '16)*, pp. 279–296: http://bit.ly/2wziEbl.

[7] 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 (CCS '16)*, pp. 17–30: https://www.comp.nus.edu.sg/~loiluu/papers/elastico.pdf.

[8] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," December 2008: https://bitcoin.org/bitcoin.pdf.

[9] https://usa.visa.com/run-your-business/small-business-tools/retail.html).