A Survey of Blockchain Storage Requirement Mitigation Techniques

Nicholas Gordon
University of Pittsburgh
nick.gordon@pitt.edu

April 20, 2020

1 Introduction

Since the release of Bitcoin [18] blockchains have grown in popularity and people are applying the concept to a wide variety of problems, as well as continuing fundamental research. Some applications include "digital gold" [18], a "global computer" [24], and a gross-settlement system [3] mainly used by banks. Fundamental research is very open, combining aspects from databases, distributed systems, networks, and security research. Perhaps the best-known problem is the enormous energy consumption of blockchains like Bitcoin, consuming more energy than the entire country of Austria at the time of writing. [2] However, less public problems like the various attack vectors, transaction throughput, and the storage consumption of blockchains impact the community just as much.

Perhaps because of the relative abundance of disk space, most of the current research overlooks the storage requirements of blockchain. This survey examines research in this space. This paper is not intended to be a complete, exhaustive review of the literature; there are undoubtedly countless papers at workshops and conferences that state never-fulfilled positions or implement incremental improvements over existing work. This paper is intended to provide a thorough look at the existing "anchor points" in this field and orient the reader in what has been done and what remains to be done.

In conducting the survey three major groups of techniques were identified:

- *Summarization* – attempting to summarize, abbreviate, or otherwise compress the information in the blockchain to reduce storage requirements
- *Pruning* – proposing how to remove unnecessary or unwanted information from the blockchain to reduce storage requirements
- *Structural* – proposing a fundamental, structural rework of the blockchain itself in an attempt to overcome the need for such storage requirements

These areas are not black-and-white and some papers fall into multiple categories or somewhere in-between. However, in the following sections, work is placed in the category it most fits according to my interpretation. Finally, as virtually all of these works are based on either Bitcoin, Ethereum, or HyperLedger Fabric, familiarity with these three systems is assumed.

2 Problem

The problem that each of these works aim to solve, in some way, is the storage requirement of the blockchain. It is slightly different from chain to chain, but I will use Bitcoin as the model.

Blocks are cryptographically linked to the previous block by a hash. This hash forms the proof-of-work that a miner found it legitimately. This serves doubly as a guarantee that the ledger has not been tampered with, as anyone possessing the entire blockchain can recursively verify hashes until reaching the genesis block, which *de facto* is assumed trustworthy.

However, verifying just the block headers is usually not useful, as the ledger is composed of the transactions within the block bodies. To verify a transaction is part of the blockchain, you must know the transaction's hash and its containing block's hash. Since each participant will want to verify different transactions, in general this means the chain in its entirety is kept by full nodes. For Bitcoin, as of April 2020, the Bitcoin blockchain was 270GB in size. [1]. Various works have stated that it can take days or even weeks to bootstrap a Bitcoin full node.

Furthermore, in the period from April 2019 to April 2020 Bitcoin grew from 211GB to 270GB.[1] The growth rate is not fixed, but a function of block size and the block discovery rate. Unless computing resources grow at least as fast, Bitcoin will eventually exclude more casual and low-powered participants, leading to increased centralization of the blockchain.

3 Summarization

At first glance the reader may confuse these techniques with pruning techniques. These techniques aim to retain all or most of the information in the blockchain, but represent it more efficiently. In almost all cases this results in a change in the properties of the blockchain, but each work emphasises different advantages to offset their costs.

3.1 The mini-blockchain scheme [8]

The earliest paper in this section mini-blockchain scheme, proposed by Bruce in 2014. The observation by Bruce, later made by most others in this survey, is that the blockchain simultaneously accomplishes three things:

- Transaction coordination and ordering a la Paxos
- Proof-of-work that secures transactions

- Record ownership of assets on the chain

His observation is that the blockchain can simultaneously, but inefficiently, accomplish all three as it is. By separating the three parts into a mini-blockchain, a proof-chain, and an account tree, which provide each of those functions respectively.

3.1.1 The mini-blockchain

The mini-blockchain is an ordinary blockchain, except it is truncated after a certain number of blocks from the tip of the chain. It functions to order blocks by the same mechanism that Bitcoin and others do. This means mining proceeds as in Bitcoin. However, after the truncation point we cannot guarantee trust, so the proof-chain is needed.

3.1.2 The proof-chain

The proof-chain, in short, is the block headers since the genesis block, but no block bodies. Each block header includes the hash of the previous block, a nonce, the account master hash, and the block root hash as in Bitcoin. This brings the security to as strong as Bitcoin's security with regards to 51% attacks, or "hidden chain attacks" as he calls them.

3.1.3 The account tree

The missing information in the proof-chain is retained in the account tree, implemented approximately as a Merkle tree. Any structure that supports a securing master hash could be used, instead.

3.1.4 Node bootstrapping

When a node wants to join the network it will acquire the proof-chain from other nodes in the network until it reaches the mini-blockchain. At this point, "slices" of the account tree are acquired and combined and the rest is similar to Bitcoin, where blocks are requested and applied to the current state.

3.1.5 Analysis

Bruce does not provide any empirical analysis of the scheme's performance. Since the block headers are simply a few hashes, we can assume they will be on the order of hundreds of bytes, a significant savings over Bitcoin's approximately 1MB block size.[1]

Further, as the mini-blockchain inherits most of its properties from Bitcoin, the same analyses apply. The final storage overheads come from the account tree, which Bruce does quantify. If we assume we are using 160-bit addresses and 64-bit account values, then Bruce's estimate of 100 billion addresses takes up constant space on the order of $160 + 64 * 10^{10} =$ 2.5TB to store the entire account tree, not counting data structure overhead.

3.2 Blockchain Abbreviation [4]

This paper was published by Amelchenko and Dolev in 2017. Briefly, the paper discusses how to condense the Ethereum blockchain into a new genesis block, as well as a description of how to accomplish the same thing in a shared-memory setting on a filesystem.

3.2.1 New Genesis Block

Ethereum blocks contain a state description that was current when the block was the tip of the blockchain, so they propose to have the network agree to replace the current chain with the abbreviated chain. That is, the abbreviated chain should occur near the tip of the chain. Specifically, this new genesis block summarizes the account balances in the blockchain.

Since probabilistic blockchains follow the chain with the highest work, they permit this new genesis block's difficulty to be the sum of the blocks it abbreviates. As other blocks receive this abbreviation, they can verify that difficulty as they will still have the entire chain.

In order for abbreviation to be successful, honest nodes must promptly adopt a valid abbreviation, otherwise an adversary with enough mining power would be able to exert outsize influence on the network.

3.2.2 Abbreviation Time

They discuss two different network types, one with uniform network latency and non-uniform network latency with respect to how much time the network has to abbreviate the chain. Their results are that a non-uniform network, that is, we cannot speculate what percentage of nodes have received a message until they all have, is that abbreviation must take less time than the block mining time minus the network latency. In a uniform network, where we can calculate the percentage of nodes that have received it, we have more time, that is, abbreviation must be faster than the block mining time.

3.2.3 Shared Memory Scheme

They also propose a model wherein nodes remotely log in to other nodes to communicate blocks, by writing them to that node's blockchain filesystem. All of their previous results apply also to this system.

3.2.4 Analysis

This approach has an obvious, unanswered question, which is how to reconcile multiple abbreviations. It is not clear how susceptible this abbreviation process is to 51% attacks, hidden-chain attacks, or other vulnerabilities that the whole chain secures against.

As for storage requirement, it is very low as the network can decide to abbreviate whenever it wants, giving a lower storage bound of the size of the

Ethereum blockchain's state representation plus the number of blocks between abbreviation.

Notably, abbreviating, i.e. deleting old block bodies makes interlocking smart contracts impossible. However, this could be countered by nodes re-publishing old smart contracts in a new block that is based off the abbreviated block. This re-publication will decrease the effectiveness of the abbreviation since the data we expected to be abbreviated is brought forward past the abbreviation. The authors do not discuss this trade-off.

3.3 Recycling Smart Contracts [21]

Published by Pontiveros, Norvill, and State in 2018, this method proposes adding a new pseudo-opcode to the Ethereum VM to compress smart contracts by replacing segments of a smart contract's code with pointers to other existing smart contracts, like a compression algorithm.

3.3.1 Pseudo-Opcode and Compression

They propose modifying the Ethereum VM to introduce a new pseudo-opcode, `COMPRETH`, that specifies where to find the substituted source code, how long the source is, which transaction its found in, and where it starts in the transaction.

They claim this "pointer" is different from the existing calls such as `DELEGATECALL` in that it does not get executed as is; to decompress, a textual replacement is done and then the source is executed after.

Chain-pointers are resolved by having a distance limit of 256 blocks, as well as resolving pointers sequentially so that it is impossible for a pointer to refer to source outside that 256 block limit.

Compression is done using a simple longest common subsequence algorithm, but instead of trying to solve for the most efficient solution, they pick the longest strings first and accept the sub-optimal packing of pointers in exchange for quick compression.

They describe that segments of code would not be replaced if the `COMPRETH` instruction would be greater. Finally they permit a hard-cap on the number of pointers in a contract to tune between compressed size and decompression time.

3.3.2 Analysis

Their results show that a window size of 256 blocks yields an average compressibility of 65%, but quadrupling the window size only yields a 69% compression ratio, a finding consistent with increasing dictionary sizes in compression algorithms.

However, at this 256 block depth it takes 13 seconds to compress a block's transactions, which is just shorter than the 14 seconds between blocks.

The authors neglect to describe how this pseudo-opcode would be tolerated by the network. It is unclear whether they intend for it to have a comparable gas price to the existing delegating opcode referenced above. They even admit

that the `DELEGATECALL` opcode was previously in high use, but with its increased gas price it has fallen out of favor, and it is unclear how they would avoid this problem.

3.4 Sidecoin [16]

Published in 2015 by Krug and Peterson, this work does not directly suggest its method as a solution for an existing blockchain's size, but how a snapshotting mechanism can create a new genesis block for a new blockchain, allowing "free buy-in" for Bitcoin users.

3.4.1 Snapshotting and Token Claiming

They stipulate that a snapsnot is all non-trivial balances currently in the Bitcoin network, and use this information as the genesis block for the new chain. The Bitcoin addresses and their balances will be embedded.

Then, roughly, a user can claim their coins on the new network by signing a transaction with both their Bitcoin key and a new sidechain key.

3.4.2 Analysis

This method is very similar to the work by Amelchenko and Dolev, though Krug and Peterson are more focused the technique's use for easing adoption of new blockchains and facilitating innovation.

3.5 Empowering Light Nodes in Blockchain with Block Summarization

[19] Published in 2018 by Palai, Vora, and Shah, which they position as a node option in between simplified payment verification nodes and full nodes, achieved by recursive summarization of blocks in the chain.

3.5.1 Summary Blocks

Simply, they replace certain blocks in the chain with summary blocks instead, where the summary block will condense transactions such as $A- > B$, $B- > C$, and $C- > D$ to $A- > D$. To avoid computational overheads associated with forking, a certain number of blocks, n, near the tip are not summarized. Then, after those spared blocks, every m unsummarized blocks are replaced by a summary block as the chain grows. That is, eventually the chain will be mostly summary blocks. To further improve this, they propose recursively summarizing existing summary blocks in the chain, creating a "summary tree" of a tuneable depth.

3.5.2 Analysis

They show that with varying tree depths and varying values of m they achieve Bitcoin compression ratios from 0.53 to 0.60.

This method allows a node to actually verify transactions given the spared blocks near the tip, something impossible with SPV. However, they are not required to store the whole chain, thus saving about half of the disk space, as shown above.

As with other compression methods, not well specified is the computational overhead of this procedure and what kinds of nodes are capable of this compression while keeping up with the network. Additionally, summarization excludes Bitcoin scripts, as the block bodies are not retained in summary blocks. Naturally, the summarization process also obscures transaction histories, perhaps enabling things such as money laundering if all nodes summarize given transactions.

3.6 Block Summarization and Compression in Bitcoin Blockchain [17]

Published in 2018 by Nadiya, Mutijarsa, Yuwono, this work is a straightforward application of summarization and compression to Bitcoin.

They apply the recursive compression technique described by Palai et al in their 2018 work, and additionally apply LZ77 and Huffman coding to compress blocks.

3.6.1 Analysis

The authors do not specify exactly which blocks are compressed, i.e. the unsummarized blocks near the tips, the summarized block tree, or if all blocks are compressed.

This results show that compressing the blocks yields a large space savings, approximately 78% in most cases. They do not quantify the computational overhead of this compression scheme, so it is unclear how much savings can be achieved if a node wants to stay up to date with the network.

Their work is mainly an incremental improvement over the summarization scheme.

4 Pruning

All of the techniques in this section in some way choose to remove information from the blockchain, either whole blocks or individual transactions. Inevitably this means losing a degree of security or traceability in exchange for compactness.

4.1 Downsampling Blockchain Algorithm [23]

Published by Quan, et al in 2019, this work applies a technique from digital signal processing to identify high-information blocks to retain, while discarding other blocks.

4.1.1 Block Information

They identify that the important information in a block is its UTXOs, and without that information the blockchain cannot process transactions. They describe the "survival block" of UTXO as the number of blocks between when it is created and spent; the lifetime of the UTXO.

Knowing this, they show that the lifetime of UTXOs fits an exponential distribution, compute a probability density function for UTXOs given a block. Then this is applied to find a block's information entropy, where approximately the higher likelihood it is to have UTXOs the higher its information entropy is.

4.1.2 Downsampling

Armed with a block's information entropy, their strategy is to pick the M blocks with the highest information entropy, and discard the rest of the block bodies, but keep all headers. Given the behavior of Bitcoin nodes to gossip transactions they receive, this downsampling introduces four cases in the products of a transaction's validity and whether the node broadcasts it.

A node could fail to broadcast a valid transaction because it has downsampled the needed block away, correctly discard an invalid transaction, or the inverse of either of those two. They give a statistical treatment that shows the probabilities at $M = 1024$ to be valid broadcasting at 1.000 and invalid discarding to be 0.999. This is better than a trivial comparison downsampling where blocks are randomly selected for downsampling. Further, they show that the probability of requiring a full node to verify transactions is relatively stable at 0.07.

4.1.3 Analysis

As with other strategies, because the block headers are retained we keep a higher level of security.

This method can be seen as a more sophisticated version of simple pruning. As with other headers-only methods, it ensures growth very slowly, as the number of retained full blocks is only this parameter M.

An obvious problem with this method is that it relies on the behavior of the network – that is, it is not necessary that Bitcoin or another UTXO-based blockchain system have an exponential curve for UTXO lifetimes. However, the statistical analysis can be repeated when the error rates become too high.

4.2 Temporal Blockchain: A Formal Analysis [12]

Published in 2016 by Dennis, Owenson, and Aziz, this work primarily focuses on designing a blockchain using formal methods. They attempt to tackle the problem of a "rolling" blockchain by designing it in the B formal language to prove correctness and other desirable properties.

4.2.1 Rolling Blockchain

Their blockchain continually deletes all blocks older than, say, 30 days. They mention the use of a checkpointing method so that the deletion is safe but do not specify what this checkpointing method could be. Other methods mentioned in this survey could be used, such as Blockchain Abbreviation or Sidecoin to achieve this.

They are not retaining block headers, as the disk requirements for their blockchain are static, which is impossible if all block headers are retained.

4.2.2 Analysis

Orthogonal to the quality of their formal analysis is their solution to the disk storage requirement. In that respect, they take the extreme position of completely losing history. At this point it becomes less clear what blockchain can solve that a conventional distributed ledger cannot.

4.3 Selective Blockchain Transaction Pruning and State Derivability [20]

Published in 2018 by Palm, Schelen, and Bodin, the they propose allowing nodes to select transactions to prune using an arbitrary predicate. This permits nodes in the network to make their own decisions about what transactions are important to them, and what to store. However, to retain the derivability of the current state extra discrimination of transactions is necessary.

4.3.1 Blockchain Anatomy and Pruning Process

The authors note that not all blocks can be pruned; for probabilistic blockchains some "guard blocks" must be left un-pruned near the tip of the chain to ensure forks and rollbacks can occur correctly. Further, transactions cannot be pruned without changing the hash of a block, so they require that the hash of a block be saved before the pruning occurs.

The pruning process is three steps: preparation, marking, and sweeping.

- *Preparation* – Collecting needed data structures, perhaps off-chain, as well as identifying blocks that might contain transactions to prune.
- *Marking* – Once blocks are identified, the predicate function is applied to each transaction in the block and marked if it passes.
- *Sweeping* – Marked transactions are deleted from the block.

4.3.2 Types of Transactions

The authors identify three classes of transactions based on their effect on system state: significant, universally insignificant, and retroactively significant. The first class is the "typical" transaction, which must be retained in order to derive system state. An example might be a transaction where an asset is issued. The second class is the trivial case, say a transaction that transfers zero tokens from one address to another. The third class is where the authors aim to find the most savings.

Transactions are considered "retroactively" insignificant if they no longer contribute to the system state in a way relevant to the user. That is, if a newer transaction overwrites a value set by an older transaction, the user can prune that transaction if they don't care about deriving the state associated with the transactions in that older block.

4.3.3 Analysis

In their implementation they modify Hyperledger Fabric and model a supply-chain scenario. This ideal scenario allows for very good results. Anytime an asset is successfully delivered, the transactions associated with that asset are considered insignificant and can be pruned. This leads to a 93% prunability and an 85% size reduction of the chain.

However, they do not investigate or thoroughly discuss the use in public, probabilistic blockchains. It is also unclear the consequences on the network if all nodes are using this scheme and have distinct pruning functions. Block availability may be impacted, which can lead to transaction verification problems if it is discovered nobody in the network possesses the transaction.

4.4 OmniLedger [15]

Published in 2018 by Kokoris-Kogias et al, this paper describes a complete blockchain system designed to employ sharding to improve scalability. This work is not primarily concerned with the storage requirements, though they do describe an optimization to their system to permit pruning.

In their optimization, the shard nodes are not required to store the entire blockchain, as in their system blocks contain a description of the system state, like Ethereum. They permit shard nodes to delete old blocks and move the responsibility of transaction verification to clients. This resembles the arbitrary pruning predicates described by Palm.[20] In general, this will not constrain the size of the blockchain.

4.5 Pruneable sharding-based blockchain protocol [14]

Published in 2018 by Feng et al, this paper positions itself as a refinement and improvement of Rollerchain. They propose something called the "PSRB" protocol. They claim that Rollerchain does not avoid the blockchain size growth problem because the whole blockchain is retained, and they propose that nodes

should only retain recent blocks and the block headers. In this way it resembles a fusion of the mini-blockchain and Rollerchain, while adding sharding to the mix.

5 Structural

All of the techniques in this section aim to solve this problem by re-thinking some property of the blockchain or adding new mechanisms. Many of the works in this category have elements from the other two categories, but because of their more radical changes to the concept they are placed here. Included in this section are blockchain-like concepts which dispense with the linear block*chain* as well as proposals to modify the proof-of-work functions to require storage or an indication of transaction validity.

5.1 Rollerchain [9]

Published in 2016 by Chepurnoy, Larangeira, and Ojiganov, Rollerchain's main insight is that full nodes are not incentivized in any way, only mining is. Rollerchain re-works the typical proof-of-work to include a "proof-of-storage" such that miners must also retain snapshots of the system state.

5.1.1 State Representation and Proof of storage

Rollerchain is based off a previous, formal definition of a blockchain system called GKL. In their view, part of the reason block storage is difficult to solve is that there is no state representation required by Bitcoin. Thus, any cumulative UTXO state is only conventional, not specific.

They propose to define a formal state that includes an account dictionary, reminiscent of Ethereum's per-block state or mini-blockchain's account tree.

Then, by extending Bitcoin Backbone's proof-of-work to include a "ticket" system which enforces the retention of snapshots, blocks are mined.

5.1.2 Analysis

The perform an analysis which shows that their POW could be interchanged invisibly with Bitcoin's POW, and thus the security of Rollerchain is at least as good as Bitcoin's.

The protocol does not reduce state size in any way, but by incentivizing miner's to retain state for, as they say, 10,000 blocks, the role of an altruistic full node can be eliminated.

5.2 Proof-of-property [13]

Published in 2018 by Ehmke, Wessling, and Friedrich, this work modifies Ethereum to include a proof that all transactions will complete successfully by providing the state information needed to verify it with the transaction.

5.2.1 Validation Paths

The proof-of-property is manifested in the validation path, which is a partial state tree, specifically they suggest a Merkle Patricia tree because it allows for a smaller path, that has all of the information needed to verify that the addresses involved in the transaction can complete.

Specifically, this is a partial state tree, including all public key addresses and their balances needed to cryptographically validate an account's balance. For an account involved in the transaction, all of that account's parent nodes in the tree and each parent's siblings must be included in this partial state tree.

The state of the blockchain will change each block, so nodes are required to upkeep their validation paths.

5.2.2 Analysis

This can be viewed as an inversion of the concept of Palm[20] and OmniLedger[15] in that instead of clients choosing what state to receive to validate transactions, clients must retain enough state to prove to others that their transactions are valid.

The authors claim that, in general, this means no nodes are required to store all blocks, as transaction submitters are required to remember the state needed; nodes merely validate this partial state tree. Because most nodes can discard most block bodies, most nodes will save up to 90% space, they claim.

Security is unaffected since block headers are not compromised, but this exchange of responsibility warrants a more thorough examination of security and correctness, as it is not clear that old results apply to this new model.

5.3 Enabling Blockchain Innovations with Pegged Sidechains [5]

Published in 2014 by Back et al, this paper does not directly address the storage requirements, rather it describes a mechanism by which two blockchains can be made to interact. This mechanism could be used to ease the adoption of technological innovations on existing blockchains rather than requiring hard forks as is currently required to adopt a major change.

5.4 On Scaling Decentralized Blockchains: A Position Paper [10]

Published in 2016 by Croman et al, this paper is an examination of the bottlenecks in current blockchain systems. Their discussion provides the insight that blockchains may need fundamental, structural changes that make them dissimilar to current blockchain models to scale to the throughput of current transaction processing services like Visa.

The paper discusses many other aspects of blockchain scalability, but their specific contributions to the storage plane of blockchains include an identifica-

tion that Bitcoin's implementation of ledger storage is inefficient. They additionally suggest that UTXO could be stored in a data structure that is "sharded" between nodes to reduce its per-node footprint. They suggest a distributed hash table, but provide no further discussion.

5.5 A Low Storage Room Requirement Framework for Distributed Ledger in Blockchain [11]

Published in 2018 by Dai et al, they propose distributing the workload of storing blocks by using error-correcting codes to encode blocks and distribute the blocks.

5.5.1 Network Codes and Distributed Storage

The authors describe "network codes" which other readers may know better as error-correcting codes, such as their specific example of a Reed-Solomon code. These codes can be grouped by their (n, k) pair, which states that if a unit of data is encoded into n packet fragments you will be able to recover it by obtaining any k of the n encoded fragments.

Then, because each block is encoded in this way, a block is not replicated everywhere, but instead the work is distributed, so that a block under this scheme will take up $\frac{1}{k}$ of the space it does in a conventional blockchain, where k is the same as above.

Further, they suggest two ways to determine the exact values of k and n as the number of participating nodes changes, as well as how to shift the computational cost of this scheme from decoding to encoding. Decoding such packets requires a solving of linear equations and then back-substitution to combine fragments. They propose restraining the coding to the binary field and at encoding time intentionally shifting packets by a random number of bits.

5.5.2 Analysis

Their "binary field random shift encoding" is not well-explained, so its correctness cannot be evaluated. They make several claims about scalability that follow from an assumed understanding of this scheme, so whether the computational overhead of such a scheme will dominate the advantage we can gain from it is unknown.

The bandwidth consumed by this scheme is lower since a block is not propagated in whole, but rather in pieces. They do not, however, explain how a node should select which packets to retain. Such a system could be readily created, such as by taking the modulo of a node's public key to determine which packets it should store, but the authors do not say.

They do not describe how to ensure that an honest majority will have enough encoded packets to reconstruct the original block. They admit a weakness of their work is packet pollution, which enables attackers to waste node computation power by providing bogus packets that force an honest node to attempt to combine them, which is computationally intensive.

5.6 Section- and Segment-Blockchain [25][26]

Published in 2018 and 2020 by Xu et al, Section-Blockchain and Segment-Blockchain respectively overlap in many ways with regards to the storage aspect and are grouped together here. They propose dividing the blockchain into "segments" of consecutive blocks and extending the proof-of-work puzzle to include a proof-of-storage that a node actually retains their segment. In this way it resembles Rollerchain. As with Rollerchain, nodes must still store all block headers.

5.6.1 Node occupations and proof-of-storage

They re-use an argument from sharding that states that an attacker must control more than half of the nodes (or *jurors*) in each shard in order to control a blockchain. They appropriate this occupation analogy to distribute the segments of the blockchain. They explain that the number of blocks per segment can change as the number of nodes in the system changes.

Specifically, the proof of storage uses the hash of the current block to select a transaction from within the segment that the claiming block should be hosting. The challenged node must provide the transaction, proof that the transaction is contained within that block, and a proof that the block is part of the chain, specifically a Merkle branch.

A reward is also specified for the proof-of-storage, which is split evenly among all miners. I believe this is to incentivize nodes to select underfilled occupations, as described in the next section.

5.6.2 Balancing the number of nodes per occupation

A particular innovation is the self-balancing property. Before a node can begin mining it must "apply" to host a specific segment. This queue is stored in each block, and a node must continually solve proof-of-work puzzles to retain its slot in the queue. Once a node joins the occupation, whenever a block is mined, it must also provide a proof-of-storage to keep its occupation. If it does not, it will be fired from its occupation and cannot mine.

5.6.3 Analysis

The authors go on to explain how to combine this system with sharding, which is in large part orthogonal to the concerns of this survey. This segmentation scheme does not reduce the size of the blockchain, but instead reduces the footprint of the blockchain on individual nodes; each node will only have block bodies for their segment, effectively limiting the number of block bodies to just the current number of blocks per segment.

5.7 Corda, IOTA, and Swirlds

The following three schemes are grouped together because each one uses a fundamentally different graph structure to represent the state of the system, making comparisons to existing blockchains difficult, especially translating existing summarization and pruning techniques. These will not be discussed in depth, but given for completeness and a suggestion of directions for future work.

5.7.1 Corda [7]

Published in 2019 by Hearn and Brown, instead of a proof-of-work and mining, transactions in Corda are controlled by distributing trust among *notaries*, essentially a controller or guarantor of assets in the network. A notary guarantees it will not validate a transaction unless all input assets are unspent. Corda thus resembles more PBTF than a blockchain, and the research into storage requirements and efficient state representation are outside the scope of this survey.

5.7.2 IOTA [22]

Published in 2018 by Popov, this system is also referred to as "the tangle." The fundamental difference is that instead of a blockchain, the tangle is a dircted acyclic graph. The tangle positions itself as a generalization and next evolutionary step of the blockchain. Transaction processing is similar to Bitcoin, where a proof-of-work puzzle must be solved, but additionally a node must validate two other transactions before it can issue its own transaction.

5.7.3 Swirlds [6]

Published in 2016 by Baird, Swirlds uses a Byzantine agreement protocol and a *hashgraph* to optimize information transfer and voting procedures in such algorithms. It explicitly is not a blockchain, rejecting proofs of work and blocks completely, falling back to well-establihed consensus protocol concepts.

6 Conclusion

As mentioned in the introduction, this survey does not and cannot cover all of the research in this field. However, the papers presented here demonstrate a clear trichotomy for how to reduce the storage size of a blockchain. Specifically, through this trichotomy we can see that there is an apparently unresolvable tension between data storage, ledger completeness, availability, reliability, and computational requirements.

Summarization techniques sacrifice some ledger completeness and computational efficiency in exchange for higher availability and reliability, while pruning techniques are more efficient but can in some cases dramatically affect availability of a given block or transaction, and may fully discard ledger completeness. On the other hand, structural techniques appear to strike a good balance by

in some cases incentivizing the problem or shifting responsibilities, but in these cases there may be significant economic ramifications that are not present in other works which are merely technical improvements on the existing system. The more radical structural approaches such as IOTA require re-evaluation of the the techniques presented here to discover how they apply to the tangle.

There is evidently room for interdisciplinary innovation, as evidenced by the downsampling algorithm work and the network coding work. There is still work to be done, as the very recent work by Xu demonstrates.

Finally, it is unclear if we can have our cake and eat it too, that is, if the original vision of the blockchain in Bitcoin has inherent disadvantages we cannot overcome with technical innovation and if we must re-consider the goals of our system as the position paper by Croman suggests. In any case, this survey demonstrates a wide, open field of research waiting to be done on a problem that certainly will not go away on its own anytime soon.

References

[1] Blockchain charts. `https://www.blockchain.com/charts/blocks-size`. Accessed: 2020-04-20.

[2] Cambridge bitcoin electricity consumption index. `https://www.cbeci.org/cbeci/comparisons`. Accessed: 2020-04-20.

[3] Xrp ledger dev portal. `https://xrpl.org/docs.html`. Accessed: 2020-04-20.

[4] Amelchenko, M., and Dolev, S. Blockchain abbreviation: Implemented by message passing and shared memory (extended abstract). *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)* (2017), 1–7.

[5] Back, S. A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., and Timón, J. Enabling blockchain innovations with pegged.

[6] Baird, L. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds, Inc. Technical Report SWIRLDS-TR-2016 1* (2016).

[7] Brown, R. G. The corda platform : An introduction.

[8] Bruce, J. D. The mini-blockchain scheme.

[9] Chepurnoy, A., Larangeira, M., and Ojiganov, A. Rollerchain, a blockchain with safely pruneable full blocks. *arXiv: Cryptography and Security* (2016).

[10] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A. E., Miller, A., Saxena, P., Shi, E., Sirer, E. G., Song, D. X., and Wattenhofer, R. On scaling decentralized blockchains - (a position paper). In *Financial Cryptography Workshops* (2016).

[11] Dai, M., Zhang, S., Wang, H., and Jin, S. A low storage room requirement framework for distributed ledger in blockchain. *IEEE Access 6* (2018), 22970–22975.

[12] Dennis, R., Owenson, G., and Aziz, B. A temporal blockchain: A formal analysis. *2016 International Conference on Collaboration Technologies and Systems (CTS)* (2016), 430–437.

[13] Ehmke, C., Wessling, F., and Friedrich, C. M. Proof-of-property - a lightweight and scalable blockchain protocol. *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (2018), 48–51.

[14] Feng, X., Ma, J., Miao, Y., Meng, Q., Liu, X., Jiang, Q., and Li, H. Pruneable sharding-based blockchain protocol. *Peer-to-Peer Networking and Applications 12* (2019), 934–950.

[15] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. *2018 IEEE Symposium on Security and Privacy (SP)* (2018), 583–598.

[16] Krug, J., and Peterson, J. Sidecoin: a snapshot mechanism for bootstrapping a blockchain. *ArXiv abs/1501.01039* (2015).

[17] Nadiya, U., Mutijarsa, K., and Rizqi, C. Y. Block summarization and compression in bitcoin blockchain. *2018 International Symposium on Electronics and Smart Devices (ISESD)* (2018), 1–4.

[18] Nakamoto, S., et al. Bitcoin: A peer-to-peer electronic cash system.(2008), 2008.

[19] Palai, A., Vora, M., and Shah, A. Empowering light nodes in blockchains with block summarization. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (2018), 1–5.

[20] Palm, E., Schelén, O., and Bodin, U. Selective blockchain transaction pruning and state derivability. *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (2018), 31–40.

[21] Pontiveros, B. B. F., Norvill, R., and State, R. Recycling smart contracts: Compression of the ethereum blockchain. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (2018), 1–5.

[22] Popov, S. The tangle.

[23] Quan, L., Huang, Q., Zhang, S., and Wang, Z. Downsampling blockchain algorithm. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2019), 342–347.

[24] Wood, D. D. Ethereum: A secure decentralised generalised transaction ledger.

[25] Xu, Y. Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)* (2018), 115–125.

[26] Xu, Y., and Huang, Y. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access 8* (2020), 17434–17441.