Using Tangly Statistics to Truncate Blockchain Growth

Nathan Stouffer advised by Dr. Mike Wittie

Abstract

Blockchains applications in cryptocurrencies, supply chain tracking, and providing data integrity. The essential service provided by blockchain is a keeping an immutable, decentralized ledger. To provide immutability, some blockchains use Proof of Work to construct an ever growing chain of blocks in a peer to peer network. Miners are expected to expend computational work to create a new block. Since blocks are intentionally difficult to create, users of a blockchain can trust the information held in the longest blockchain.

By design, miners have collective control over a blockchain. When miners are sufficiently decentralized, it is infeasible for a coalition of miners to gain explicit control of a blockchain. Explicit control would allow a coalition to decide which blocks are added to the chain, reducing the integrity of the system. Thus a blockchain perform better when control of the chain is decentralized. Decentralization can be difficult to achieve when blockchains grow to an unmanageable length. To begin mining, one must download and process the entire chain. This is not a problem when the chain is relatively small, but a larger chain (such as Bitcoin) can take days to process.

Excessively long chains limit decentralization in two ways. First, lightweight devices are prevented from becoming miners. Soon, many desktops and laptops will not be able to become a Bitcoin miner. Second, incredibly long bootstrapping time deters participation from users who do have sufficient space. Together, these issues decrease decentralization which reduces the effectiveness of a blockchain.

Over the summer, I worked with collaborators to devise a protocol that prunes blocks while preserving the chain's integrity. We ended the summer with a solution sketch that has potential but still needs significant work. At a high level, our solution has miners of recent blocks vote for a blockchain summary. If sufficiently many votes agree, bootstrapping nodes can trust the blockchain summary, which drastically reduces onboarding times.

For my capstone project, I will extend the solution sketch from this summer. There are still issues to resolve and I must provide formal security proofs. Following this, I will implement a model of our solution. Using the model, I will run simulations and extract experimental results.

1 Introduction

Blockchain was introduced in order to provide distributed consensus without relying on a central party [1]. Avoiding a central party provides two benefits. First, there is no central location that an attacker can access to compromise a system. A centralized blockchain is much easier to compromise than a decentralized blockchain. Second, there is no single party with complete control over a system. An example of where decentralized control is useful is in international currency exchange. Banks have the power to charge high transaction fees for international monetary transfers and customers are forced to pay the fees because there is no alternative. Legitimate cryptocurrencies have enough competition between parties (miners) that there is no room for a monopoly, giving customers a cheaper option.

Bitcoin was released to the general public in 2009 as the first implementation of a blockchain. Bitcoin garnered a lot of support and quickly became one of the most prominent cryptocurrencies. Since Bitcoin's inception, blockchains have found other applications. They are used in other cryptocurrencies, verify supply chains, and increase device autonomy in the Internet of Things [2]. Blockchains are even being used to provide security and privacy in the medical field [3]. Blockchain technology has the potential to enhance data security in a wide range of diverse applications.

Blockchain's services are best realized when control of the chain is decentralized. Decentralized control can become difficult to achieve when blockchains grow to an unmanageable size. I am writing this proposal to request funding to work on a method to truncate blockchains. My work on this project will be a continuation from the REU research I collaborated on this summer. We have a solution sketch and a prototype implementation, but more work needs to be done to prove the correctness of the solution and to translate the prototype to a practical implementation. If successful, this project has the possibility of making blockchains more effective, extending data reliability and user privacy [2] [3].

The remaining sections provide information regarding background information, my plan, an estimated time line, and other logistical details.

Nathan Stouffer

email: nathanstouffer1999@gmail.com
phone: 208.293.5883
github: github.com/nathanstouffer

A motivated student graduating in Spring 2021 with a double major in computer science and mathematics. Hoping to join a fast-paced and exciting work environment where I can combine both my majors to develop software.

EDUCATION

Computer Science Major and Data Science Minor — *Bachelor of Science* GRADUATING SPRING 2021

Mathematics Major — *Bachelor of Science* GPA: 3.94

Montana State University (MSU) – Bozeman, MT

Relevant coursework: advanced algorithms, networks, software engineering, computer security, computer graphics, machine learning, computer science theory, topology, dynamical systems

EMPLOYMENT

Math Tutor — *MSU* FALL 2018 – PRESENT

Working as a tutor in the Math Learning Center at MSU
Identifying difficult areas for students and explaining problem solving techniques
Assisting students with algebra, pre-calculus, calculus I/II, and linear algebra

Research Experiences for Undergraduates — *MSU* SUMMER 2020

Participated in a Research Experiences for Undergraduates (REU) program
Communicated with collaborators about complex technical problems and ideas
Worked towards truncating a blockchain network's ever-growing chain

Computer Science Course Assistant — *MSU* FALL 2019

Instructed a lab section of 24 students in an introductory computer science course
Encouraging students to think through problems and own their solution
Hosted weekly office hours
Graded lab assignments and larger programs

PROJECTS

Senior Capstone — *MSU* FALL 2020 – PRESENT

Implementing and extending the research performed in my REU program
Working with collaborators to produce a new protocol that prunes blockchains
Building and interpreting results from a large-scale model of our solution

Emergent Behavior — *MSU* SUMMER 2020 – PRESENT

Exploring the emergent behavior of agents acting on local rules
Building an agent-based computer simulation
Analyzing bifurcations in emergent behavior based on initial conditions
Comparing results with a partial differential equation model

Directed Reading Program — *MSU* SPRING 2020

Independently studied a book about abstract algebra
Discussed thoughts and questions with two graduate students once per week

SKILLS

Java, Python, C++, C, OpenGL, MATLAB, LaTeX, Git

HONORS AND AWARDS

MSU Math Department's Outstanding Scholar Award

MSU's Achievement Scholarship

Milton F. Chauner Mathematics Excellence Scholarship

Mary C. Griffin Scholarship

President's List (MSU): S18, F18, F19, S20

Dean's List (MSU): F17, S19

1st place team at MSU's 2019 programming contest

2nd place team at MSU's 2018 programming contest

Ran a marathon

Voted "Most Influential (2018–2019)" by MSU's National Residence Hall Honorary

ADDITIONAL EMPLOYMENT

Resident Assistant — MSU
Construction Laborer — ID

INTERESTS

skiing (downhill/xc)
ultimate frisbee
running
cribbage

2 Background

2.1 Blockchain

At a high level, a blockchain is a sequence of blocks that is immutable in practical applications. Immutability in blockchains is often provided through Proof of Work. Proof of Work was introduced in [4] and first used for blockchain in the Bitcoin whitepaper [1]. In a Proof of Work blockchain, a block is valid if its cryptographic hash is below a preset threshold. Each block primarily consists of data that cannot change (financial transactions, medical records, etc). However, every block contains a field for a fixed-length string of bits that make the output of the hash function below the threshold. The fixed-length string is called a nonce. If someone finds a nonce, they have "mined" a block. Since the hash function is cryptographic, it is difficult to reverse. The best strategy for someone mining a block is to guess and check nonces until a sufficiently low output is found. Figure 1 displays the typical structure of a blockchain.

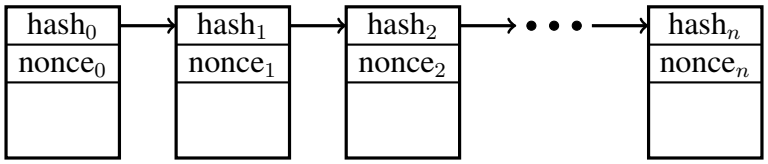

Figure 1: Prototypical Proof of Work Blockchain

A broader perspective of a blockchain is an implementation of a finite state machine. In the state machine, blocks are transitions between the states of applications running on top of a blockchain. Abstractly, the k^{th} block B_k is a transition from state S_{k-1} to state S_k with certain validity requirements. See Figure 2 for a visual. A tangible example is a cryptocurrency. Cryptocurrencies realize state as a record of the amount of currency each public key controls. Blocks are a set of transactions that transfer currency between public keys. In state S_0, no one controls any currency. Then block B_1 moves the blockchain to S_1 where its miner controls some newly created currency. Then block B_2 grants cryptocurrency to its miner and contains currency transfers between users of the currency. This process repeats until state S_n is reached, which contains the current account-balance pairs of all the current users of a cryptocurrency.

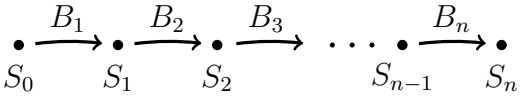

Figure 2: Blocks as transitions between states

2.2 Technical Problem

To understand the technical problem, we must understand two properties of blockchains. First, we observe that miners must have every block in the chain to contribute new blocks. And, second, blockchains grow in perpetuity. These properties mean that new miners must download and process an ever increasing amount of data as time passes. Even now, when Bitcoin is just over a decade old, it can take days or weeks to become a miner. Such a barrier prevents some nodes from mining, limiting decentralization. Less decentralization reduces the effectiveness of a blockchain.

When viewing a blockchain as a finite state machine, we can change the requirements of what a miner needs to know. All a miner must know is a state S_k and every block following that state. Then a bootstrapping miner could compute the current state. If S_k is close to the end of the chain, then the miner requires significantly less bootstrapping time by only requesting blocks following S_k.

How does a bootstrapping miner obtain the state S_k? They can't just ask a blockchain node, because they could receive a faulty state. This is the problem we would like to solve: Can we provide a mechanism so that a bootstrapping node can trust a recent state?

2.3 Related Work

Our solution gains inspiration from a paper written by Matzutt et al. that provides a way to verify Bitcoin states. They suggest that recent miners be allowed to vote for a state and then bootstrapping nodes trust the majority of votes. Each vote is recorded in a block and each block has the capacity to store a single vote [5]. Storing votes in blocks gives them immutability, so a bootstrapping node knows that they are counting every vote.

There are two problems with this idea. First, a vote can only be generated as quickly as new blocks. In Bitcoin, this averages to about 10 minutes per block, meaning a bootstrapping node might have to wait a week for there to be sufficiently many votes to trust a state. Second, this solution relies on implementation details in the Bitcoin protocol and may not apply to blockchains in general.

3 Goal

3.1 Idea

This project will continue my research from this summer as part of a Research Experiences for Undergraduates (REU) at Montana State University. My collaborators and I came up with the following idea.

Like Matzutt and his collaborators, we allow recent miners to vote [5]. However, we choose to store votes in a distributed hash table. Storing votes in a distributed hash table is convenient because it allows recent miners to vote immediately, but it also gives attackers an opportunity to delete votes. To combat vote deletion, we drew on the idea of a tangle [6]. A tangle is a directed acyclic graph

where the nodes are added to tangle by selecting the two most recently added nodes as parents. In this context, nodes are realized as votes. A tangle manager receives votes from recent miners and adds each of them to the tangle by attaching them to parent votes.

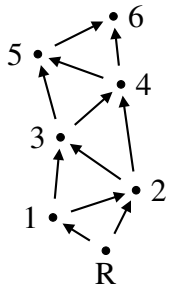

Figure 3: Example of a tangle

A vote is invalid if one if its parents is invalid or deleted. Deleting a vote causes a chain reaction in the tangle that deletes votes that support the attacker, preventing the attacker from swinging the election, by disproportionately deleting votes on one side of the election. A bootstrapping node can request the vote tangle and then make a decision based on valid votes.

While we do have a sketch of a solution idea, more work needs to be done to prove its correctness and security as well as to demonstrate these properties through a practical implementation. Preliminary results indicate that it is resistant to some types of attacks but we need to consider smarter and more powerful attackers.

3.2 Implementation

4 Methodology

The methods of this project come in three different styles. First, we will focus on generating correctness and security proofs. For this, I will spend a lot of time thinking about the properties of our solution and how they can be formalized. Second, there comes the implementation. Over the summer, my collaborators and I built a proof of concept system using python. I plan to expand the capacity of this implementation. This task will involve writing and testing code as well as integrating the system with a distributed hash table. Finally, there will be our methods while analyzing results. After the system is built, we can simulate attackers with varying intelligence to see how our solution responds. I will need to be able to write scripts that analyze and visualize the output data. At the end of all of this, our goal is to publish a paper. This will involve drafts and proof-reading.

4.1 Design

5 Timeline

I plan to stick to the following general time frame, which comes in 4 phases.

1. Proofs. In the first phase of the year, I will focus on writing formal security proofs about our solution. This will provide verification that our idea is resistant to attacks, increasing legitimacy. I expect this phase to last about two months.

2. Software Development. Beginning around November, I will return to the implementation that I worked on over the summer. We only implemented a proof of concept, so there is a lot of functionality that remains to be implemented. Most importantly, I must port the code from running on a single computer to running on a distributed system with a functional network. I estimate that this will last until February 2021.

3. Analyze Results. After the model has been created, I can use it to generate experimental verification that our solution is resistant to attacks. I will run simulations and analyze the results. This should take around one month.

4. Prepare Publication. I will spend the remainder of the year working towards a publication.

6 Results

7 Discussion

8 Conclusion

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[2] Wei Cai, Zehua Wang, Jason B Ernst, Zhen Hong, Chen Feng, and Victor CM Leung. Decentralized applications: The blockchain-empowered software system. *IEEE Access*, 6:53019–53033, 2018.

[3] Asad Ali Siyal, Aisha Zahid Junejo, Muhammad Zawish, Kainat Ahmed, Aiman Khalil, and Georgia Soursou. Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. *Cryptography*, 3(1):3, 2019.

[4] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[5] Roman Matzutt, Benedikt Kalde, Jan Pennekamp, Arthur Drichel, Martin Henze, and Klaus Wehrle. How to securely prune bitcoin's blockchain. *ArXiv*, abs/2004.06911, 2020.

[6] Serguei Popov. The tangle. *cit. on*, page 131, 2016.