Optimizing Trustless Consensus Algorithms Utilizing Variable Blocktime

Rohan Mathur & Drew Ripberger

Abstract

Consensus represents an important concept and challenge in computing. We believe that with an improved method of reaching total agreement, the internet and, to a greater extent, distributed systems as a whole, could be greatly improved. Attempts at solving this problem have used a variation on a globally-replicated state machine to reach general consensus; this system commonly being referred to as blockchain. Its method of reaching agreement, however, limits the use cases of the technology due to its inherent time requirement to reach this point. In each respective implementation, we plan to test an elapsed time for every successful transaction for a range of blocktimes; each implementation getting its own range, but the number of tests remains constant, along with the nodes used for the testing and the Bitcoin implementation. We expect that decreasing the blocktimes will not only increase the efficiency of each algorithm, but also maintain reliability and stability of the algorithms. By the end of our testing, we hope to be able to determine the optimal blocktime for use with a trustless consensus algorithm. This will allow for the fastest possible confirmation times for use with cryptocurrencies and the most rapid updates in other blockchain systems.

1 Introduction

In 2008, as a result of the financial crisis, Satoshi Nakamoto proposed a completely decentralized, fully electronic currency that he called Bitcoin [1]. Today Bitcoin is still going strong, gaining popularity both conceptually and practically, revolutionizing the financial sector. Nakamoto started a movement of sweeping decentralization, spawning a myriad of crypto currencies similar to Bitcoin. Many of these projects attempt to improve on his original idea by altering the consensus algorithm. These algorithms lay at the core of the technology, making sure that everyone agrees on events and more specifically transactions that happen in the network. Bitcoin utilizes an algorithm called Proof Of Work, which uses computation to provide both financial and cryptographic confirmation of transactions. This, however, can be very costly, environmentally harmful, inefficient and centralized; these problems inspire continuous innovation, and improvement in consensus algorithms as a whole. Transactions in this decentralized network, referred to as a blockchain, are grouped together and are confirmed in blocks after a period of time called the blocktime. We are testing what effect this blocktime has on the capacity of blockchains that use various consensus algorithms.

2 Consensus & Blocktime

The concept of the blocktime originates from one of the core concepts of distributed systems, using time to schedule and interpret actions [2]. Time is a crucial part of both traditional and trustless consensus algorithms. Concepts like the replicated state machine utilize time to maintain consensus [3]. It revolves around the idea of iteratively updating the state of all the machines in the network, so they are all in sync after a certain period of time. In a trustless consensus algorithm, this is defined as blocktime. Blocktime, however, manifests itself in different ways, in different types of consensus algorithms. In the Paxos algorithm, a popular consensus protocol for trusted systems, time is used to evaluate the liveness of the nodes. Though in a trustless system in which peers cannot trust each other, algorithms such as Proof Of Work can use it to organize state changes across the network.

In the Bitcoin network, the Proof Of Work algorithm is used to reach consensus. Essentially, it allows transactions to be backed by a monetary prize and cost [1]. While the blocktime used with Bitcoin is set to be 10 minutes, it in actuality never a constant time. This is because blocktime is representative of how long it should take a node in the network (these nodes are commonly referred to as *miners*) to solve a

complex, computationally intensive problem. However, the network is periodically recalculating the difficulty of this problem in an attempt to keep it close to 10 minutes. The miner that solves this problem is then rewarded in a few ways:

1) They are awarded a sum of Bitcoin.

2) They are allowed to commit the next block of transactions to the blockchain.

To the average user of Bitcoin the first reward is of greatest concern as this is how the money supply is distributed, but the second is essential for the operation of the network.

The main purpose of the computational puzzle is to provide a barrier of access to generating and committing to the blockchain. It is a way for the network to do both: pick a single person in to generate the block that is adopted by the network, but also pick a person who has a stake in the network. Picking a person with stake in the network is essentially a way of verifying that the person trying to commit the block to the chain has more to gain by acting in the good faith of the network, than trying to exploit it because of the vast computational, and therefore monetary, investment required to accomplish it. It would be impossible to have a trustless system in which everyone had equal say in blocks being confirmed to be legitimate, because of course in a computer there is not a way to confidently tie identity to the computer or visa versa.

On top of giving time for the race deciding the miner of the block, the blocktime also decides the degree to which you should trust that a transaction set is valid. Just because a transaction has been included in a block does not guarantee that it will forever be valid. Coming to consensus also includes dealing with multiple people coming up with correct solutions to the block puzzle at nearly the same time, and how to bring the network to a univalent state that has a common history of transactions. So at times, there may temporarily be chains that are all considered valid. Whether or not the chain that is agreed upon in the end contained your transaction is unknown.

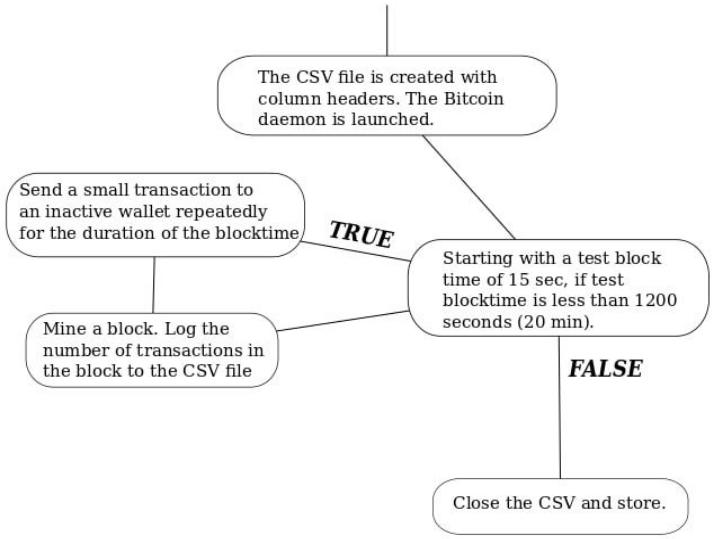
All of these fundamental principles of blockchain need to be accounted for to accurately weigh the pros and cons of various blocktimes.

3 Hypothesis

In the long term a longer blocktime will make for a less robust and efficient algorithm. This is generally speaking, as certain scenarios may lend themselves to a different blocktime.

4 Testing

To test a range of blocktimes, a mode included in the Bitcoin Core software called Regtest is used. This allows users to manually and near instantly generate blocks when needed [5]. We were able to use a public JSON-RPC wrapper for the daemon to allow us to interact with the Bitcoin node using C++ [6]. Then using a bash script to wrap the interface with our regtest program we set up the iterative testing environment using the different blocktimes [7]. The algorithm used to test these different blocktimes is as follows:



5 Efficiency Analysis

Our final set of tests generated around 200 data points; they are shown graphed in figure 1). This data represents the number of successful transactions in one unit of blocktime. This could convey that an increase of blocktime increases the efficiency of the transaction. That, however, is not the case. The graph is slightly misleading in that while it does show an increase in transactions made, that increase exists not because of an increased efficiency, but just more time to complete transactions. The real efficiency of each

blocktime is best shown in figure 2. Figure 2 shows the number of transactions per second for every unit blocktime. From this, one can derive that the actual efficiency of confirming transactions decreases as the blocktime increases.

As shown, the efficiency based on unit block time makes a negative logarithmic curve defined as:

$$f(x) = -1.08ln(x) + 10.383$$

Using this function, one can tell that although the efficiency is always decreasing, the rate at which it decreases slows. This is best represented as the derivative of efficiency function:

$$f'(x) = \frac{-27}{(25x)}$$

While defining the derivative may seem, at first, trivial and unnecessary to understand the trend of the graph, it is very useful for many important practical calculations.

One of these important calculations is the point or set of points at which the change in efficiency becomes so minute, for all practical purposes those points are the same. We decided that a reasonable slope at which all change is rendered insignificant is 0.0001. Therefore the point at which the graph reaches a practical "flatline," can be defined as:

$$0.0001 = |\frac{-27}{25x}|$$
$$x = 10,080$$

Therefore, once x reaches around 10,080, the preceding difference in efficiency is so small, it would make little, if any, difference in practical use.

6 Security Considerations

The total possible load of the network is not the only factor to be considered when evaluating the overall effectiveness of a blocktime. Maintaining the security of the network is also a major aspect of optimizing the algorithm. Without assurance that a sent transaction cannot be tampered with, there is little point in trusting the blockchain with the very vital transfer of value.

If a merchant wanted to accept Bitcoin in exchange for their goods or services, they have an important decision to make: how many transaction confirmations should they require until they consider a payment valid? For small purchases where only a negligible amount of Bitcoin is being exchanged, a low minimum number of confirmations is safe. However, when a large exchange of value is required, the need to be safe from attacks from malicious nodes is of course greater, in turn needing a greater number of confirmations.

These attacks from malicious nodes are committed in an attempt to edit the blockchain to a state that fits the attackers agenda. This can be double spending of Bitcoin, general disruption or a variety of other malicious purposes. To accomplish this, the attacker has to mine valid blocks faster than the totality of the honest nodes in the network. Due to the nature of the Bitcoin Proof of Work consensus algorithm this is very rare. These attacks are very hard to accomplish, and require an enormous amount of computing power. In addition, Proof of Work provides important incentive for honest behavior. The monetary gain of contributing your hashing power to the honest section of the network would generally be greater than trying to act in bad faith.

The effect the change of blocktime has on the network can be shown as follows. B is the set of blocktimes being tested.

$$B = \{x \in \mathbb{N} \mid x = 15n,\ n \in \mathbb{N},\ x \leq 1200\}$$

We can use the equations provided in the Bitcoin whitepaper to find the probability of an attacker overtaking the honest network from z blocks behind [1]. This z blocks behind is also equivalent to the number of confirmations that a merchant is requiring. We also use p as the probability an honest node finds the next block, q as the probability an attacker finds the next block, and λ is the Poisson distribution of the attackers potential progress [9].

$$\lambda = z\frac{q}{p}$$

Then using this occurences per time interval, we can get the probability that the malicious node will wage a successful attack on the network using the cumulative distribution equation defined by Nakamoto in the white paper with a quick adaption assuming $p + q = 1$

$$f(z,\ q) = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (q/(1-q))^{(z-k)})$$

Assuming a hypothetical merchant wanted to require 60 minutes (3600 seconds) worth of blocks to pass before accepting a transaction as valid, we can find the number of blocks required to pass for our differing blocktimes. This gives us the z value for each point, and we can set the probability of a successful attack to an arbitrary number between 0.5 and 0. For our analysis we will be using 0.1 or a 10% chance of generating the next block. This number is representative of the part hashrate the malicious miners have out of the honest network. If their probability of mining the next block, essentially their hashrate, is greater than or equal to 50% of the honest nodes, their attack will always be successful at some point, so for all intents and purposes we can assume any attack will be less than 50% of the honest network. Here we define M as the set of pairs of the blocktime and the corresponding probability of a successful attack within an hour worth of blocks.

$$M = \{(t \in B, p) \mid p = f(3600/t,\ 0.1)\}$$

Figure 3 is demonstrating this graph of set M, generated by using an adaptation [7] of the code provided by Satoshi Nakamoto in the Bitcoin whitepaper [1], written in C++ for easier data collection. With a constant of an hour of confirmation time set by our theoretical merchant, we see that the probability of a successful attack on the network varies greatly. As the blocktime increases the probability that an attacker with a constant share of the network can successfully alter the network, increases as well.

7 Conclusion

As we found through analyzing our data, a lower blocktime is consistent with increased efficiency. This being said, however, the efficiency is not the only consideration that needs to be made when choosing a blocktime for a blockchain.

First, it is good to keep in mind what creating more blocks does to the average user of the chain. Every single node that wants to run as a part of the Bitcoin network is required to have an instance of the complete blockchain. Even now, with a blocktime of 10 minutes the total space required to store the Bitcoin blockchain has surpassed 156 GB as of March 7, 2018 [8]. With a similar volume of transactions, but up to 40 times as many blocks for a blocktime of 15 seconds, the headers (or the necessary identification information) of the blocks could easily begin to raise the necessary storage required to download the blockchain to even more manageable levels.

Practically speaking though, a shorter blocktime in an algorithm would be ideal. This is because as we have shown they can withstand a greater number of transactions per second, but also due to the inherent nature of having more blocks, they are able to includes one's transactions in more blocks. If a new transaction is included in a block, it is said to have a confirmation. Then, for every block after that, the transaction gains a confirmation. These confirmations are used by merchants and those who use Bitcoin to determine the validity of a transactions. Due to the fact that multiple chains can exist at the same time because of two valid blocks being mined almost simultaneously, people have to be careful of what transactions they consider to be valid. If the transaction were in actuality just mined to one of the less supported chains, their entire transactions could be thrown out until mined on the chain most popularly supported. Therefore using an algorithm with a faster blocktime would also allow people to confirm their transactions faster.

Ultimately, while there may be some benefits to a blocktime as long as 10 minutes, in today's fast paced economic world, it is a hindrance to efficiency and day to day practicality of a trustless consensus algorithm. As the data expresses, a 15 second blocktime would greatly increase the efficiency necessary to support an increasingly globalized world, while at the same time supporting transactions that could settle in the duration of a conversation.

Fig. 1

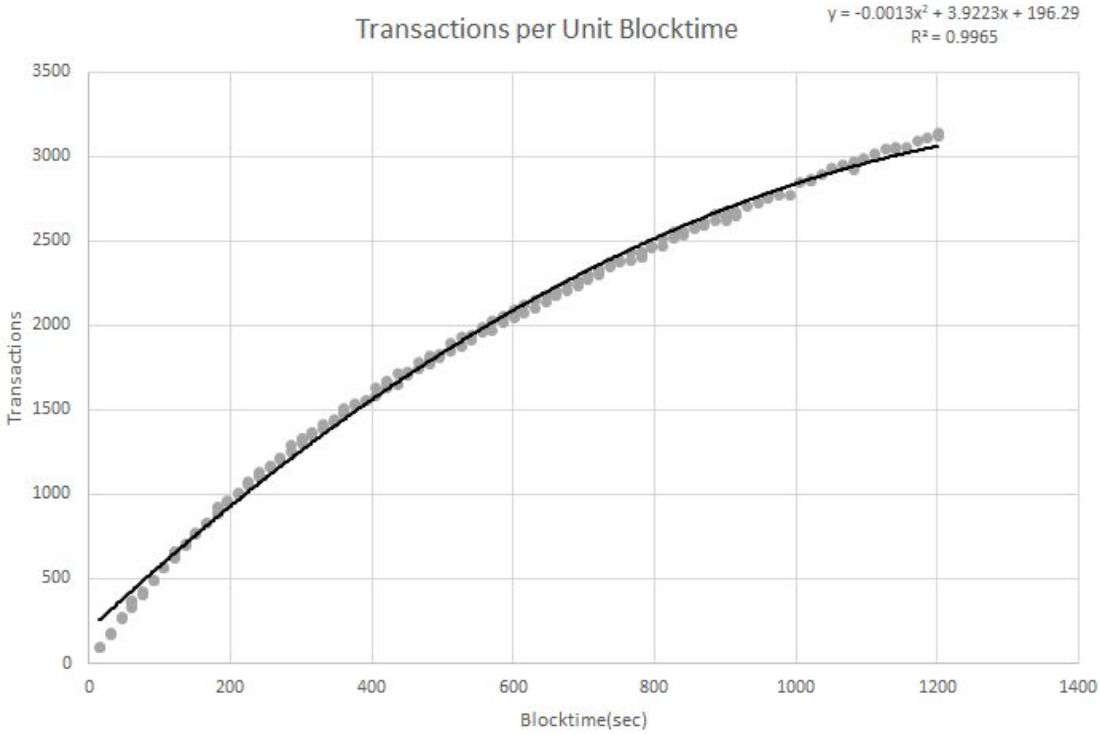

Transactions per Unit Blocktime
y = -0.0013x² + 3.9223x + 196.29
R² = 0.9965
Transactions
Blocktime(sec)
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400

Fig. 2

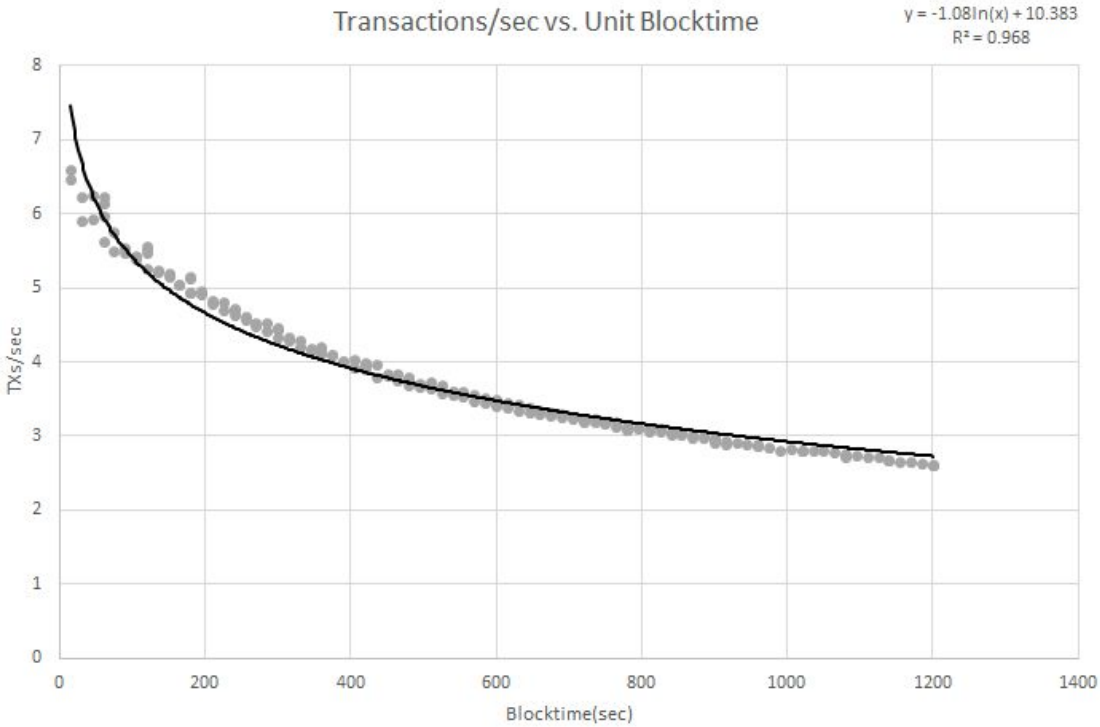
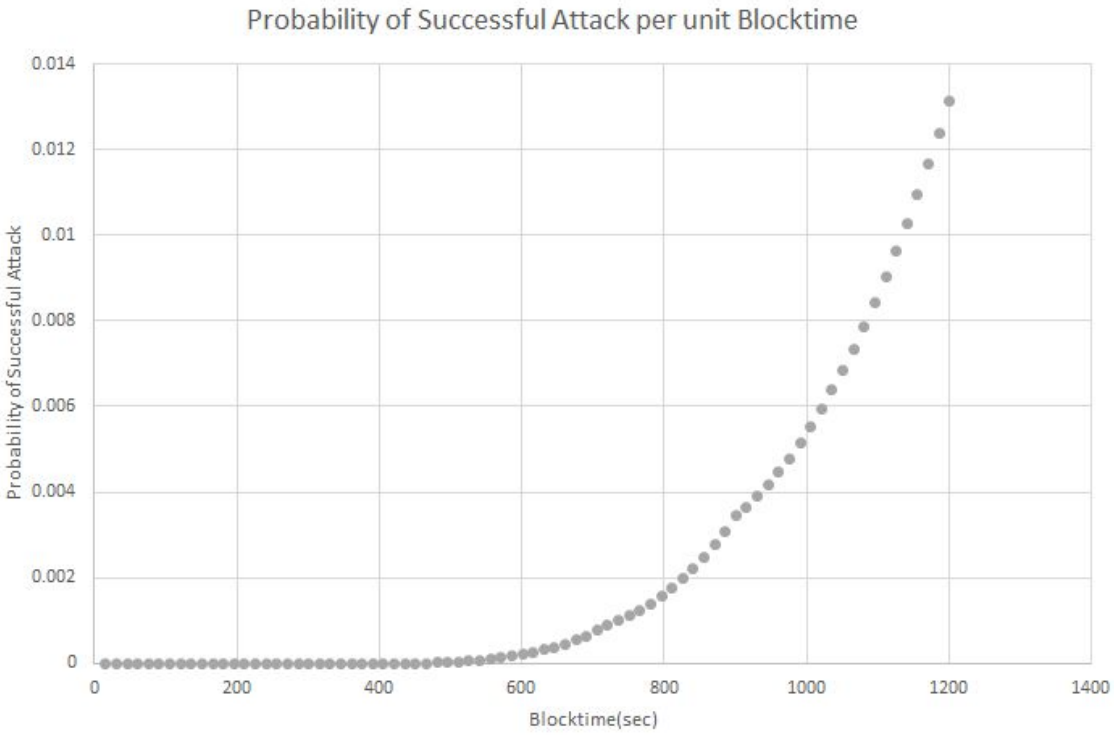

Transactions/sec vs. Unit Blocktime
y = -1.08ln(x) + 10.383
R² = 0.968
TXs/sec
Blocktime(sec)
0
1
2
3
4
5
6
7
8
0
200
400
600
800
1000
1200
1400

Fig. 3


Probability of Successful Attack per unit Blocktime
Probability of Successful Attack
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
200
400
600
800
1000
1200
1400
Blocktime(sec)

7 References

[1] Nakamoto, S. (n.d.). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of ACM, 21*(7), 558-565. doi:10.1145/359545.359563

[3] Lamport, L. (1984). Using Time Instead of Timeout for Fault-Tolerant Distributed Systems. *ACM Transactions on Programming Languages and Systems , 6*(2), 254-280. doi:10.1145/2993.2994

[4] Lamport, L. (1998). The part-time parliament. *ACM Transactions on Programming Languages and Systems, 16*(2), 133-169. doi:10.1145/279227.279229

[5] Regtest, Regression Test Mode. (n.d.). Retrieved March 09, 2018, from https://bitcoin.org/en/glossary/regression-test-mode

[6] Minium, Xloem, & CryptoCurrencyStuff. (2017, September 30). Bitcoin-api-cpp (Version 0.3) [Computer software]. Retrieved March 9, 2018, from https://github.com/minium/bitcoin-api-cpp

[7] Drewrip, & THE-COB. (2018, March 1). Blocktime (Version 1.0) [Computer software]. Retrieved March 9, 2018, from https://github.com/drewrip/blocktime

[8] Blockchain Size. (n.d.). Retrieved March 10, 2018, from https://blockchain.info/charts/blocks-size

[9] Ozisik, P., & Levine, B. N. (n.d.). An Explanation of Nakamoto's Analysis of Double-spend Attacks. Retrieved April 4, 2018, from https://arxiv.org/pdf/1701.03977.pdf.