A Stochastic Model for Proof of Work Block Production

Owen Delahoy* and Josiah Evans*

November 19, 2018

Abstract

In Proof of Work blockchain systems under theoretical conditions, block production time can be modeled as an exponential distribution. This model provides a natural method for adjusting to changes in network hashpower via a cumulative distribution function. The cumulative distribution function can be adjusted by modeling network hashpower via two exponential moving averages, one for block production time and the other for block difficulty, and by choosing a desired quantile for a target time to produce a block. Improvements to this model include accounting for latency in hashpower consolidation and block propagation.

1 Introduction

Methods of difficulty adjustment in proof of work blockchains have gone through several iterations since the introduction of Bitcoin in 2009.

The original algorithm takes the average block time over a 2016 block period, ideally every two weeks, and produces a new target based on the average block time over the period [1]. This, however, is vulnerable to extreme changes in hashpower. In Bitcoin, a reduction of network hashpower of 50% at the beginning of a 2016 block period would take nearly one month to correct to an appropriate difficulty and resume normal 10-minute block times. This in turn would lead to a reduction in transaction processing throughput for an extended period, leading to higher transaction fees.

In response to the drawbacks of Bitcoin's algorithm, other projects have taken to calculating a new difficulty target after every block.

Bitcoin Cash uses a simple moving average (SMA) of time over the last 144 blocks to calculate the difficulty for the next block [2]. This allows the difficulty

*Owen Delahoy and Josiah Evans contributed equally to this work. This paper was sponsored by the Hycon Foundation

to converge on an appropriate value quickly in the event of extreme changes in hashpower, as were experienced in the early days of the Bitcoin Cash fork. Bitcoin Cash was especially vulnerable to this due to the limited number of options ASIC-based miners have in choosing which currency to mine [3]. This caused the hashpower to fluctuate by as much as 300% [4] over the span of 12 hours.

Ethereum uses a piece-wise function to adjust its difficulty. If the block time of the last block is above 20 seconds, the difficulty is adjusted downward; if the block time is less than 10 seconds, the difficulty is adjusted upward [5]. In addition to this calculation, the presence of uncle blocks (due to Ethereums implementation of GHOST) adds a modifier to how the difficulty is adjusted [6].

In practice, these methods and others work well enough given the high degree of variance and noise in order to maintain a specified average time but exert little control over the precise shape of the distributions of observable block times, owing to the lack of a general, parameterized block production model.

2 Difficulty in Proof of Work

In a Proof of Work based blockchain system, the hash H of the block is required to satisfy certain conditions [7]. In Bitcoin [8] and Ethereum [9] [10], a hash with a numerical representation that is less than a given target value is eligible for inclusion into the blockchain. This can be represented algebraically by the expression:

$$H < T \tag{1}$$

As hash values in use in most PoW blockchain systems have specific bounds, the probability p that a randomly generated hash value is less than the target value T can be represented by proportion of T over the range R of the hash function:

$$p = \frac{T}{R} \tag{2}$$

The inverse of this probability represents the expected number of hashes, or the mean number of hashes, to produce a block with the supplied target. This inverse is more commonly known as the block difficulty. Thus, a lower target value results in a higher difficulty. Combining this feature with an estimate of the total hashrate of the network allows the protocol to adjust difficulty such that some target time t is maintained on average between blocks.

3 Properties of Cryptographically Secure Hashing Functions

For a given hashing function to be considered cryptographically secure, the following three properties must be present:

- Pre-image resistance
- Second pre-image resistance
- Collision resistance

For a hash function to be pre-image resistant, it must be sufficiently difficult to resolve the input provided to the hash function when provided only the output, a property shared by one-way functions.

For a hash function to be second pre-image resistant, it must be difficult to create a second input from the first input which creates the same output to the hash function.

For a hash function to be collision resistant, it must be difficult to create a separate input that creates an identical output.

These properties are required for Proof of Work blockchain systems to ensure that the block production process is stochastic. Some examples of popular hashing algorithms used for Proof of Work include SHA-256, SHA-3, Blake2b, Skein, Grstl, Scrypt and JH, all of which are widely considered to be cryptographically secure [11].

4 Implications of Cryptographically Secure Hashing Functions in Proof of Work Blockchains

Using a cryptographically secure hashing algorithm means that there is no strategy for choosing an input which gives a greater chance of producing a desired output, given the random nature of the resulting hash values; they are represented by a discrete random variable with a uniform distribution.

Thus, the distribution that describes the relationship between several trials and the chance of success of a given number of trials k can be described by the cumulative distribution function of a geometric distribution, where p represents the chance of success of any individual hashing attempt.

$$CDF(p, k) = 1 - (1 - p)^k \tag{3}$$

From this cumulative distribution function, it is possible to solve for p for the desired distribution for a given k and desired quantile Q.

$$p = 1 - (1 - Q)^{\frac{1}{k}} \tag{4}$$

5 Estimating Network Hashrate

To choose a new difficulty generating parameter p, it is important to find an appropriate value for k, the number of trials. However, in a decentralized adversarial environment, k cannot be measured directly. Instead, k must be estimated from observations of past data. One method of estimating k is to use the mean number of expected trials $\hat{k_i}$ from previous p_i values.

$$\hat{k_i} = \frac{1}{p_i} \tag{5}$$

As hashpower is prone to fluctuation due to market forces, energy availability, etc., recent observations of network hashpower are more relevant than older observations for the purpose of adjusting difficulty. An exponential moving average may be used to efficiently capture this distinction. Since hashrate is a measure of hashes over a period, two observations are needed to form an estimate: the rate of block production β_i and the past values of $\hat{k}_i$.

$$EMA(k_i) = \alpha\hat{k_i} + (1 - \alpha)\hat{k}_{i-1} \tag{6}$$

$$EMA(\beta_i) = \alpha\beta_i + (1 - \alpha)\beta_{i-1} \tag{7}$$

$$\text{where } 0 < \alpha < 1,\ \beta_i \leq \beta_{max}$$

These parameters may be tuned by a sensitivity parameter α. A helpful metric for tuning the α parameter with respect to security is the security margin ξ.

$$\xi = \frac{\sum_1^x \alpha^x}{\sum_x^\infty \alpha^x} \tag{8}$$

Which can be simplified to:

$$\xi = (1 - \alpha)^x \tag{9}$$

For a given security margin ξ, an α may be chosen from the required number of observations x:

$$\alpha = 1 - \xi^{\frac{1}{x}} \tag{10}$$

6 Choosing a Difficulty Parameter

The goal of the difficulty parameter p is to control the block production rate β such that β converges to some target t. An estimated network hashrate $\hat{\lambda}$ can be defined as:

$$\hat{\lambda} = \frac{EMA(\hat{k})}{EMA(\beta)} \tag{11}$$

To adjust p such that $\hat{\lambda}$ converges toward t, a new mean k value may be decided via:

$$k = t \cdot \hat{\lambda} \tag{12}$$

Thus, a new difficulty parameter p may be determined by substituting for k:

$$p = 1 - (1 - Q)^{\frac{1}{t\hat{\lambda}}} \tag{13}$$

The choice of quantile Q should be determined by the distribution of block rates such that the target time describes the chosen quantile.

7 Implications of the Stochastic Model

Under theoretical, ideal network conditions where block propagation is instantaneous and all sources of hashpower can immediately respond to mining a new block with a discrete number of hashes, the distribution of block times in this scheme follows a geometric distribution. However, due to the high number of hashes typically required to produce a block, and since multiple hashing attempts may overlap from different parties, we suggest that an exponential model more appropriately describes the distribution of block times:

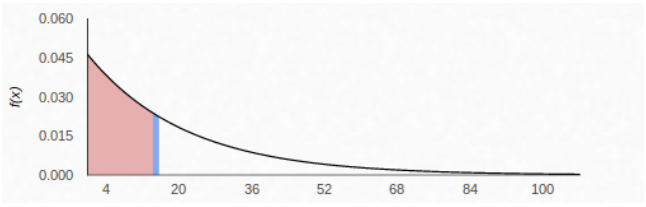

Figure 1: The theoretical block time distribution in the Hycon Network [12]. The red portion represents the lower half of the distribution. [13]

8 Future Work

In most real circumstances network latency in propagating new blocks makes the total hashpower increase over a delay period until all miners are mining on the

new block. This introduces a skew in the otherwise monotonically decreasing character of the exponential distribution. As the skew is no longer constant, but variable, a better model for the distribution of block times follows a gamma distribution [14], where shape and rate may be computed from the mean block time and the variance of the observed block times.

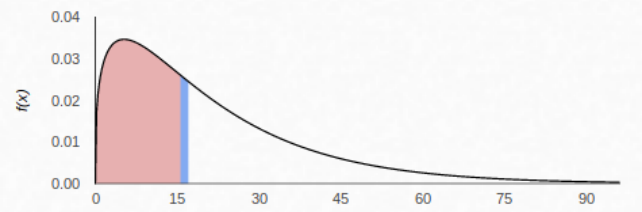

Figure 2: A better approximation of the real block time distribution in the Hycon Network. The red portion represents the lower half of the distribution. [13]

A difficulty adjustment algorithm exists for the approximate gamma distribution, although as alpha and beta may vary over time, a more complex analysis is needed to maintain the security thresholds and prevent manipulation of inputs by malicious parties. One approach may be to model the shape and rate parameters with exponential moving averages with a sensitivity appropriate to the desired security margin.

An alternate model of skew based on block propagation may resemble the equation:

$$H(t) = H_{max} \cdot 1 - ar^t \tag{14}$$

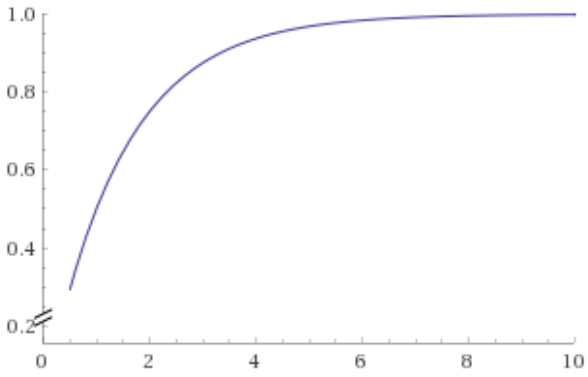

Figure 3: How hashrate might tend to full consolidation over time [15]

Where r describes the rate of block propagation over time and a is a modifying coefficient. The probability density function of such a distribution would be of the form:

$$PDF(t) = p(1-p)^{t(1-ar^t)\hat{\lambda}t} \tag{15}$$

9 Conclusion

Using an exponential distribution as a model of block time distribution and an exponential moving average of network hashpower, it is possible to select a difficulty that under theoretical conditions maintains a desired time at the specified quantile. Real factors, such as network latency and hashrate consolidation introduce a skew that can be modeled more accurately from a gamma distribution, and further still by a distribution that contains a more complex model of hashrate consolidation.

References

[1] Bitcoin Core, *pow.cpp*. Available at `https://github.com/bitcoin/bitcoin/blob/78dae8caccd82cfbfd76557f1fb7d7557c7b5edb/src/pow.cpp#L49`.

[2] Bitcoin ABC, *Difficulty Adjustment Algorithm Update*. Available at `https://www.bitcoinabc.org/2017-11-01-DAA/`.

[3] Aaron van Wirdum, *Miners Are Milking Bcashs Difficulty Adjustments (and Why This Is a Problem)*. Available at `https://bitcoinmagazine.com/articles/miners-are-milking-bcashs-difficulty-adjustments-and-why-problem1/`.

[4] BitInfoCharts, *Bitcoin Cash Difficulty historical chart*. Available at `https://bitinfocharts.com/comparison/bitcoin%20cash-difficulty.html`.

[5] Ajay Singh and Neeraj S Srivastava, *How Difficulty Adjustment Algorithm Works in Ethereum*. Available at `https://dltlabs.com/how-difficulty-adjustment-algorithm-works-in-ethereum/`.

[6] Vitalik Buterin, *EIP 100: Change difficulty adjustment to target mean block time including uncles*. Available at `https://github.com/ethereum/EIPs/issues/100`.

[7] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available at `https://bitcoin.org/bitcoin.pdf`.

[8] Bitcoin Core, *pow.cpp*. Available at `https://github.com/bitcoin/bitcoin/blob/384967f311b4c6b1d6c797f7821d25feb26bafbf/src/pow.cpp#L74`.

[9] Go Ethereum, *sealer.go*. Available at https://github.com/ethereum/go-ethereum/blob/144c1c6c52456808428e2b69dbe5c4ebfc3606ca/consensus/ethash/sealer.go\#L166.

[10] Dr. Gavin Wood, *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Available at https://ethereum.github.io/yellowpaper/paper.pdf.

[11] Shu-jen Chang, Ray Perlner, William E. Burr, Meltem Snmez Turan, John M. Kelsey, Souradyuti Paul, and Lawrence E. Bassham, *Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition*. Available at https://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf.

[12] Hycon Foundation, *Hycon Block Explorer*. Available at https://explorer.hycon.io/.

[13] Matt Bognar, *Gamma Distribution*. Available at https://homepage.divms.uiowa.edu/~mbognar/applets/gamma.html.

[14] Eric W. Weisstein, *Gamma Distribution*. Available at http://mathworld.wolfram.com/GammaDistribution.html.

[15] Wolfram Alpha, *Computational Intelligence*. Available at https://www.wolframalpha.com/input/?i=y+%3D+1+-+.5%5Ex+from+0+to+10.