DRAFT
A Stochastic Model for Proof of Work Block Production

Owen Delahoy* and Josiah Evans*

December 6, 2018

Abstract

In Proof of Work blockchain systems, block production time can be modeled as an exponential distribution, which provides a natural method for adjusting to changes in network hash power via a cumulative distribution function. The cumulative distribution function can be adjusted by modeling network hash power via two exponential moving averages, one for block production time and the other for block work, and by choosing a desired quantile for a target time to produce a block. The bias in the exponential moving averages toward present information can be tuned such that the variance in the moving averages is tolerable. Improvements to the model include accounting for latency in effective hash power 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.

Bitcoin's original algorithm calculates the average block time over a 2016 block period, approximately once every two weeks, and produces a new target based on the average block time over the period [1]. However, it is vulnerable to extreme changes in hash power. In Bitcoin, a reduction of network hash power 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. Consequently transaction processing throughput is reduced 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.

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

Bitcoin Cash uses a simple moving average (SMA) of time over the last 144 blocks to calculate the difficulty for the next block [2]. An SMA allows difficulty to be more responsive to extreme changes in hash power, such as in the early days of the Bitcoin Cash fork. Bitcoin Cash was subject to regular severe hash power fluctuation due to a combination of the limited number of options ASIC-based miners have in choosing which currency to mine and the relatively unresponsive nature of Bitcoin's difficulty adjustment [3]. The network hash power fluctuated by as much as 300% [4] over the span of 12 hours on multiple occasions.

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 Ethereum's 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. We propose a method of controlling block production time that is derived from a statistical model, representing the distribution of possible block times, capable of adapting even in the event of drastic changes to network conditions

1.1 Difficulty in Proof of Work

In a Proof of Work based blockchain system, the hash h of a block is required to satisfy certain conditions [7]. A hash that satisfies such conditions can be said to belong to the set of usable hashes H_u. If the condition is such that the value of h must be below some target value D, then it can be said:

$$\forall\ h \in H_u,\ h \leq D \tag{1}$$

The probability that a randomly chosen hash value h is usable ($h \in H \wedge h \in H_u$) can be represented by proportion of the cardinality $|H_u|$ divided by the cardinality of the set of possible hash outputs $|H|$ of the hash function:

$$P(h \in H_u | h \in H) = \frac{|H_u|}{|H|} \tag{2}$$

The reciprocal of this probability represents the expected or mean number of randomly chosen hashes, to produce a usable hash $h \in H_u$, more commonly known as the block difficulty [8]. Thus, a lower target value D results in a higher difficulty, and allows the network to control the mean amount of work required to produce a block. Combined with an estimate of the total hashrate of the network, the protocol may adjust difficulty to minimize the difference between the target mean time T and actual mean block time.

2 Properties of Cryptographically Secure Hashing Functions

For a given hashing function to be considered cryptographically secure, the following three properties [9] 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, and Grøstl, all of which are widely considered to be cryptographically secure [10].

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

Using a cryptographically secure hashing algorithm means that there is no practical strategy for choosing an input which gives a greater chance of producing a desired output, given the random nature of the resulting hash values; in our model 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 the cumulative distribution function, it is possible to solve for p for the desired distribution for a given k and desired quantile Q.

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

Using p, a new D value can be selected via:

$$D = \frac{1}{p} \tag{5}$$

3 Estimating Network Hashrate

To choose a new difficulty generating parameter p, it is important to find an appropriate value for k, the expected number of trials, commonly referred to as work. 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.

$$\hat{k_i} = \frac{1}{p_i} \tag{6}$$

As hash power is prone to fluctuation due to market forces, energy availability, etc., recent observations of network hash power are more relevant than older observations for the purpose of adjusting difficulty. We propose an exponential moving average to efficiently capture this distinction. Since hashrate is a measure of hashes over a period, two observations are needed to form an estimate: the miner reported block time β_i and the past values of $\hat{k}_i$. As β_i is subject to manipulation, it must be bounded by some maximum value β_{max}.

$$EMA(k_i) = \alpha\hat{k_i} + (1 - \alpha)\hat{k}_{i-1} \tag{7}$$

$$EMA(\beta_i) = \alpha\beta_i + (1 - \alpha)\beta_{i-1} \tag{8}$$

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

3.1 Choosing an α value

As observed data is subject to luck, manipulation, or poor clock synchronization, the bias α toward present data must be appropriately selected. An α value that is too high is prone to manipulation and attack, as a small number of inputs can significantly change the average. However, a high α is also capable of reacting to severe changes in network conditions more quickly. Conversely, a low α is significantly harder to manipulate, but is unable to respond to severe changes in network conditions, leading to potentially undesirable block production rates.

A helpful metric for choosing an α parameter is the proportion of influence of new inputs to old inputs ξ:

$$\xi = \frac{\sum_1^x \alpha^x}{\sum_x^\infty \alpha^x} \tag{9}$$

Which can be simplified to:

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

For a given ξ, an α may be chosen from the required number of observations x to have ξ influence over the average:

$$\alpha = 1 - \xi^{\frac{1}{x}} \tag{11}$$

In the context of security in a blockchain, x is the number of maliciously crafted blocks required to have artificial inputs to have ξ influence over the EMA calculations. In the context of sensitivity, x is the number of blocks needed to respond with ξ effectiveness over extreme changes in network configuration.

We suggest that α be chosen such that the expected cost of acquiring ξ influence by any one party be greater than or equal to the cost of attacking the network via other means.

3.2 Choosing a Difficulty Parameter

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

$$\hat{\lambda} = \frac{EMA(\hat{k})}{EMA(\beta)} \tag{12}$$

To adjust p such that $\hat{\lambda}$ approaches T, a new expected mean k value may be chosen via:

$$k = T \cdot \hat{\lambda} \tag{13}$$

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

$$p = 1 - (1 - Q)^{\frac{1}{T\lambda}} \tag{14}$$

The choice of quantile Q should be determined by the distribution of block rates such that the target time describes the chosen quantile; i.e. $Q = \frac{1}{2}$ suggests that 50% of blocks will be produced prior to T, and 50% after T making T the median block time.

3.3 Implications of the Stochastic Model

Under theoretical, ideal network conditions where block propagation is instantaneous and all sources of hash power 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 we suggest that an exponential model more appropriately describes the distribution of block times:

$$PDF(t) = ke^{-kt} \tag{15}$$

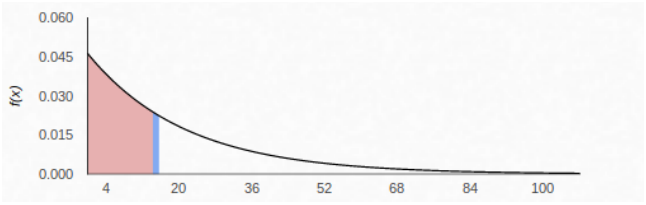

Figure 1: The theoretical block time distribution in the Hycon Network [11]. The red portion represents the lower 50% of the distribution. [12]

4 Future Work

In most real circumstances network latency in propagating new blocks makes the total hash power increase over a delay period until all miners are mining on the new block, introducing a higher variance and skew in the otherwise monotonically decreasing character of the exponential distribution. As the skew is a function of typical block propagation latency, a better model for the distribution of block times follows a gamma distribution [13], where shape A and rate B may be computed from the mean and variance of the observed block times.

$$PDF(t) = \frac{1}{\Gamma(A)} t^{A-1} e^{-Bt} \tag{16}$$

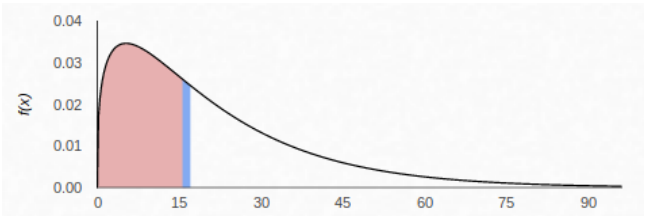

Figure 2: A better approximation of the real block time distribution in the Hycon Network. The red portion represents the lower 50% of the distribution. [12]

A difficulty adjustment algorithm exists for the approximate gamma distribution, although as A and B may vary over time, a more complex analysis is needed to maintain the security thresholds and prevent manipulation of inputs by malicious parties. An approach similar to our prior parameter estimation may be to model A and B with exponential moving averages with a sensitivity appropriate with respect to security and performance.

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

$$H(t) = H_{max} \cdot (1 - \Lambda^{\frac{t}{\tau}}) \tag{17}$$

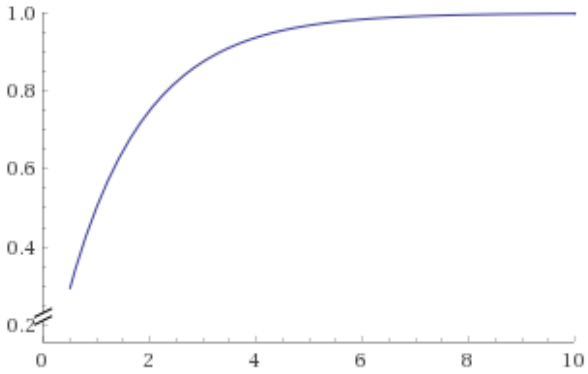

Figure 3: How hashrate might reach full efficiency over time [14]

Where Λ describes the proportion of the network not updated for each increase in t, and τ is the time for $1 - \Lambda$ of the network to be updated. The probability density function of such a distribution would be of the form:

$$PDF(t) = 2 \cdot t\hat{\lambda}p(1 - \Lambda^t) \cdot e^{-t^2\hat{\lambda}p(1-\Lambda^{\frac{t}{\tau}})} \tag{18}$$

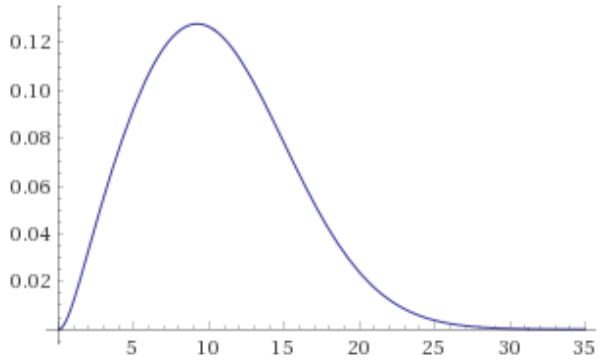

Figure 4: An example PDF from the advanced model [15]

Comparing the theoretical models to observed data, it is clear that the distribution of block times follows an exponential character following a sharp rise as effective hash rate increases.

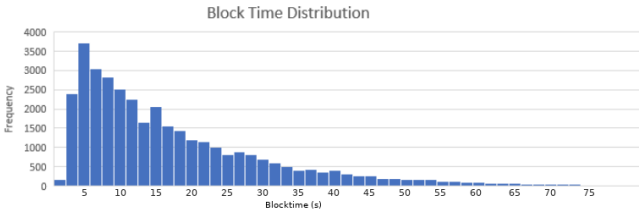

Figure 5: The observed distribution of block times from the Hycon network

5 Conclusion

Using an exponential distribution as a model of block time distribution and an exponential moving average of network hash power, it is possible to select a difficulty that under theoretical conditions maintains a desired time at the specified quantile. Tentative empirical data from the Hycon network indicates that this maintains a reasonable block time despite large fluctuations in effective hash power. 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 effective hash power.

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, *Bitcoin Developer Reference*. Available at `https://bitcoin.org/en/developer-reference#block-headers`.

[9] P. Rogaway and T. Shrimpton, *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*. Available at `http://web.cs.ucdavis.edu/~rogaway/papers/relates.pdf`.

[10] 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`.

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

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

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

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

[15] Wolfram Alpha, *Computational Intelligence*. Available at https://www.wolframalpha.com/input/?i=y+\%3D+2+*+t+*+1000+*+.00001+*+(1+-+.4\%5Et)+*+e\%5E(-t\%5E2+*+1000+*+.00001+*+(1+-+.4\%5E(t+\%2F+15)))+from+0+to+35.