Real-Time Block Rate Targeting

Thomas M. Harding
tom@chain2.org

January 24, 2020

Abstract

A proof-of-work blockchain uses a retargeting algorithm, also termed a difficulty adjustment algorithm, to manage the rate of block production in the presence of changing hashrate. To derive the parameters that guide the search for the next block, nearly all such algorithms rely on averages of past inter-block time observations, as measured by on-chain timestamps. We are motivated to seek better responsiveness to changing hashrate, while improving stability of the block production rate and retaining the progress-free property of mining. We describe a class of retargeting algorithms for which the sole inter-block time input is that of the block being searched for, and whose response is nonlinear in that time. We discuss how these algorithms allow the other consensus rules that govern allowable timestamps to be tightened, which may improve the blockchain's effectiveness as a time-stamping machine.

1 Introduction

1.1 Traditional retargeting algorithm

A traditional retargeting algorithm sets a hash target G_n that defines a valid cryptographic hash of block n. The hash is required to be no larger than the target:

$$H(\text{block } n) \leq G_n \tag{1}$$

which gives rise to the block's proof-of-work [1]. G_n is a function of data found in earlier blocks and is committed to as part of the header of block n. It is set so as to expect future inter-block times of a desired value T, given the hashrate estimate derived from past inter-block times. In the case of Bitcoin, it is recalculated every 2016 blocks, and unchanged otherwise. Omitting some details inessential to our examination:

$$G_n = \begin{cases} G_{n-1} \frac{s_{(n-1)} - s_{(n-2016)}}{2016T}, & n \bmod 2016 = 0 \\ G_{n-1}, & \text{otherwise} \end{cases} \tag{2}$$

where s_n is the committed timestamp of block n and T (= 600 seconds) is the desired inter-block time. G_n is known as soon as block $n-1$ is published.

Equation 2 makes use of the scaling property of the exponential distribution [3]. Post-retarget, assuming hashrate continues at its average, a valid block n will ideally be found after inter-block time t_n that is exponentially distributed with rate

$$\lambda = \frac{1}{T} \tag{3}$$

$$t_n \sim Exp(\lambda) \tag{4}$$

which describes a homogeneous Poisson process. Of course, hashrate will not be constant on the blockchain of interest and, as a random variable, t_n exhibits sample variance, so retargeting continues *ad infinitum*. Further, the act of retargeting itself introduces inhomogeneity.[4]

Equation 2 contains a well-known error due to Satoshi which does not affect our analysis, and which we will not refer to again: the term $s_{(n-2016)}$ should have been $s_{(n-2017)}$.

1.2 Unresponsiveness of the traditional algorithm

The observed inter-block time t_n has a standard deviation as large as its mean, which likely drove the design decision to aggregate 2016 consecutive samples before retargeting Bitcoin.

That decision reduced the coefficient of variation from 100% to 2.2%, but has resulted in a >5% overshoot versus target of the long term block production rate in the environment of ever-increasing hashrate.

More recently, Bitcoin has shared the pool of global SHA256 hashrate with other blockchains such as Bitcoin Cash, which have been assigned value in the eyes of the market. As miners are free to allocate their hashrate, a dynamic situation has emerged wherein miners follow diverse strategies to deploy hashrate among the blockchains where it can be applied.

In this context, the Bitcoin retargeting algorithm has a weakness. It reacts to hashrate changes only once every two weeks, while relative changes in the prices of the tokens (BTC, BCH, etc.) often occur in seconds. Such changes directly influence relative profitability and therefore miners' allocation decisions.

We do not attempt to survey the altcoin landscape or its myriad retargeting algorithms, but the phenomenon of coin-hopping is nothing new to its inhabitants. Swings in the varied types of altcoin hashrate have been dramatic enough to end the existence of the altcoin as its miners suddenly depart and blocks are produced so slowly that users lose patience altogether [7, 10].

Using a dynamic economic model, Noda et al. [6] find that the traditional retargeting algorithim does not converge after a severe price shock when hashrate is allowed to vary depending on expected dollar reward value.

The Bitcoin Cash algorithm, with its moving 144-block window, reacts more quickly to relative price changes, adjusts its own block production, and mitigates the profitability disparity, to the benefit of both blockchains. It is found to converge slowly, but exhibits a worrisome resonant hashrate oscillation with a

period of 144 blocks. Nevertheless, we see the advantage of responsiveness and are prompted to ask how we can increase it.

1.3 The memoryless property vs. progress-freedom

The exponential is the unique *memoryless* statistical distribution [2, chapter 4]. Being memoryless is the property that the likelihood of observing any particular wall-clock time interval from now until the appearance of block n is independent of the time passed since the appearance of block $n-1$, whether that time is zero, 48 seconds, or any other value.

Progress-freedom is distinct from being memoryless. We define progress-freedom to be the property that a miner's chance of finding a block at any moment is independent of how much work he has already done since the appearance of block $n-1$.

Progress-freedom is a critical property for a proof-of-work blockchain because it results in a miner's chance to find the next block being proportional to his hashrate. If a miner were to somehow get credit for progressive work done toward a block solution, larger miners would increase their chances of finding the block with each passing second, at the expense of smaller miners, creating an incentive to centralize mining.

Progress-freedom is a property of the iterated cryptographic hash search, whose trials are independent of *each other*, even if the hash target has a time dependency. We describe a system that remains progress-free, but aims to be more responsive to hashrate changes, to exhibit more stable inter-block times, and be more resistant to target and timestamp manipulation attacks, while necessarily discarding the memoryless property.

2 Engineering the block production rate

Consider a more general block production rate function

$$\lambda(t) = at^{k-1} \tag{5}$$

Equation 5 contemplates a system where, during the search for block n, the instantaneous block production rate is not constant, but rather a monomial function of the time t since block $n-1$. Equation 3 is a special case ($a = 1/T$, $k = 1$) of Equation 5.

By "time t" we refer to idealized global wall-clock time, and we assume for the moment that timestamps are always recorded as the global time of the respective event. The accuracy of this assumption depends on the incentive framework we create. This is discussed in Section 5.

For the simple reason that it depends only on time, and not on accumulated work, Equation 5 is progress-free. Time-dependent *rate functions* are well studied in statistics and are also known as *hazard rate functions* (the hazard is often something like a machine failing).

We set $k > 1$, so that with each passing second, the instantaneous block production rate increases. However, this even more strongly affects the likelihood that the block has already been found, which is expressed by the cumulative distribution function $F(t)$. These factors are illustrated in Figure 1.

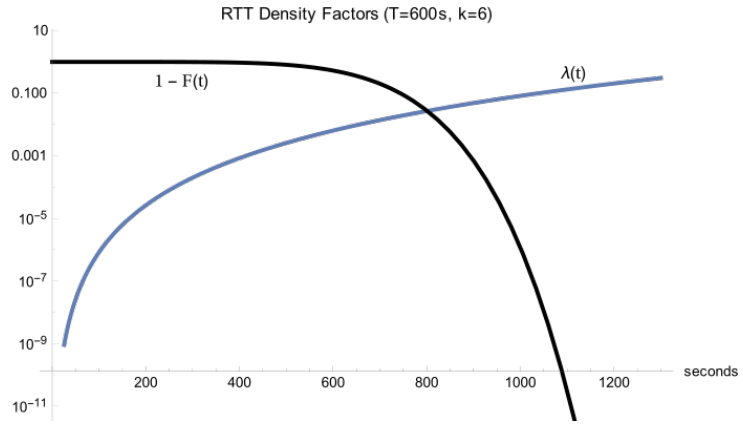

Figure 1: RTT density factors (log scale)

The net result is a quasi-symmetrical distribution of inter-block times [Figure 2], whose shape is narrower with higher values of k. We are free to choose k, but we also require the mean inter-block time to be T. Therefore, we are not free to choose a; we derive it.

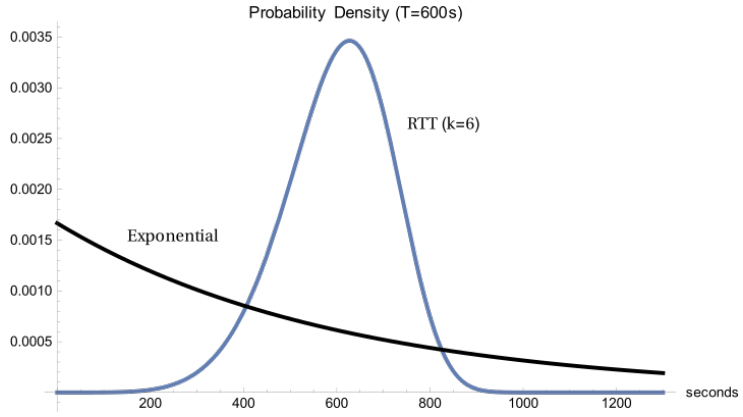

Figure 2: RTT density (compare exponential)

It turns out that the hazard rate function uniquely defines the distribution of observed times by the relation:

$$F(t) = 1 - \exp\left[-\int_0^t \lambda(t)dt\right] \qquad [2, \text{chapter } 4.1] \qquad (6)$$

Substituting Equation 5 into Equation 6 and solving the integral gives the cumulative distribution function

$$F(t) = 1 - e^{-\frac{at^k}{k}} \tag{7}$$

and probability density function ($= F'(t)$)

$$f(t) = at^{k-1}e^{-\frac{at^k}{k}} \tag{8}$$

which is a Weibull distribution [2, chapter 5.2] (notably, Weibull reduces to exponential when $k = 1$). The expected value, or mean, inter-block time t_n for block n is

$$\begin{aligned} \mathbb{E}[f(t)] = \mathbb{E}[t_n] &= \int_0^\infty at^k e^{-\frac{at^k}{k}}\, dt \\ &= \Gamma(1 + \frac{1}{k})(\frac{k}{a})^{\frac{1}{k}} \end{aligned} \tag{9}$$

Setting Equation 9 equal to T and solving for a gives the value of a in terms of the constants k and T:

$$a = k\left[\frac{\Gamma(1 + \frac{1}{k})}{T}\right]^k \tag{10}$$

3 Real-Time Targeting

To keep the blockchain verifiable, t_n must be the difference in committed timestamps between the block n being searched for and the previous block:

$$t_n = s_n - s_{(n-1)}$$

The hash target $G_{(n-1)}$ of the previous block, starting with the genesis block, is known. This committed target continues to be interpreted according to Equation 1, namely as the maximum hash that would be valid *if* the search for the block n were carried out using the traditional algorithm.

But we introduce changes to the *actual* search algorithm, described in the next three subsections.

3.1 Subtargeting

To conduct the block search using the increasing block production rate of Equation 5, we define a *subtarget* at time t since block $n - 1$, such that

$$g(t) = G_{(n-1)}\frac{\lambda(t)}{\lambda} \tag{11}$$

whereby we have applied the desired block production rate, and corrected for the traditional production rate λ. Combining Equations 3, 5, 10 and 11 gives

$$\begin{aligned} g(t) &= G_{(n-1)}k\frac{\Gamma(1 + \frac{1}{k})^k}{T^{k-1}}t^{k-1} \\ &= G_{(n-1)} \cdot \mathit{constant} \cdot t^{k-1} \end{aligned}$$

The proof-of-work validity constraint / mining target depends on t and is given by

$$H(\text{block } n) \leq g(t)$$

3.2 Modified retargeting

The subtargeting process results in a block found at time t_n which is distributed as shown in Figure 2. The quasi-symmetry and low (81% lower for $k = 6$) variance compared to exponential are key. To a chosen degree, and before the next block's base target is even computed, t_n is forced toward its mean by subtargeting. In other words, the observed inter-block time reacts less strongly to changed hashrate than it would under the traditional algorithm.

While it reduces the incidence of overly short or long inter-block times, the less reactive inter-block time works against us when retargeting for the next block. For that operation, we need to amplify the deviation from the mean back to its traditional equivalent.

To achieve this transformation is straightforward given the form of the function we chose for Equation 5. At each moment, we compute $F(t)$ from Equation 7, and plug it into the inverse CDF of the exponential distribution, giving an equivalent time u in that space:

$$u = -\frac{\ln(1 - F(t))}{\lambda} \tag{12}$$

We can see on Figure 2 that stretching the RTT distribution of t out to the equivalent exponentially distributed time u will make a short inter-block time shorter, and a long inter-block time longer. The amplified reaction leads to noisier—but appropriately reactive—difficulty.

When block n is found (or found prospectively in a block template produced for a miner), we compute u_n directly from t_n by combining Equations 3, 7, 10 and 12:

$$u_n = aT\frac{t_n^k}{k} \tag{13}$$

u_n is the inter-block time that would have been observed if the traditional retargeting algorithm were being used. Each block header commits to a homogeneous-Poisson-equivalent target

$$G_n = G_{(n-1)}\frac{u_n}{T} \tag{14}$$

which is analogous to Equation 2, but which

- references the timestamp in block n itself: $G_n \leftarrow u_n \leftarrow t_n \leftarrow s_n$.
- uses only one inter-block time (t_n) as input.
- is recomputed with every block.
- is not used to validate the hash of block n (although the target itself is validated), but rather block $n + 1$ (per Section 3.1).

Note that the subtarget $g(t)$ does not influence G_n except through its role in the discovery of t_n.

Remarkably, there is no averaging whatsoever of past inter-block times. Instead, the subtargeting process is a kind of auction, which starts at a level difficult for the network to achieve, and continually becomes easier until the network achieves it with the hashrate available. For example, with $k = 4$ and $T = 600s$, producing a block with $t_n = 1s$ is some 80 million times more difficult than in the traditional system.

3.3 Required Adjustment to T

We stated in Section 1.1 that the process of retargeting itself makes the global block production process inhomogeneous [4], even without our nonlinear construction.

Rosenfeld [5] finds that with constant hashrate, the traditional Bitcoin retargeting algorithm produces a mean inter-block time that is $\frac{1}{2015}$ longer than the intended 600 seconds.

Since we fully retarget every block, this effect is much larger and cannot be ignored. We show in Appendix A that to achieve a true mean inter-block time of T, the adjusted value of Equation 18

$$\hat{T} = \frac{T}{\Gamma(1 - \frac{1}{k})}$$

must be used in place of T by software implementing the algorithm, just as though it were the desired inter-block time.

This adjustment is sufficient to produce an observed mean of T under constant hashrate. Variable or trending hashrate will produce a different mean, but this is true of every retargeting algorithm.

4 Parent chainwork

Since a block's chainwork now depends sensitively on its inter-block time t_n, the block comparator is changed to reference the chainwork of the parent, block $n - 1$, rather than the chainwork of the block itself.

This change ensures that blocks with the same parent have the same chainwork, and reduces the opportunity for miners to conduct inexpensive "orphaning" attacks by starting a competing chain with a block whose timestamp is one second earlier than another miner's block.

5 Tighter timestamp validity rules

In Section 3.2, we mentioned that it is very difficult to mine a block with very small t_n. It is evident, too, that it is impossible to mine a block with $t_n = 0$,

as Equation 5 makes plain that the block production rate is zero in that case. $t_n < 0$ is just as nonsensical.

Consequently, it is natural that $t_n > 0$ be enforced as a consensus rule. Furthermore, without negative inter-block timestamp differences, it may not be possible to recover from a situation where timestamps get ahead of the real-world clock. So it is very natural that we also tighten the maximum allowable timestamp to "now" instead of Bitcoin's "now + 2 hours". We doubt anyway that miners will be eager to accept a peer's "block from the future" that was easier to find than allowed right now.

How well the committed timestamps match actual time is an emergent result of the system, but since there is strong incentive to choose the latest possible non-future timestamp, i.e. a timestamp of "now", we have reason to be optimistic.

6 Conclusion

We have described Real-Time Targeting (RTT), a class of algorithms for retargeting the block production rate of a proof-of-work blockchain.

By changing the hash target during the search for a block, it is no longer necessary to mine a block at the old difficulty to change to a new difficulty. Instead, every block immediately resets to a new difficulty that immediately reflects the network hashrate estimate gathered during the search for the latest block itself, with no averaging of past inter-block times.

We found that with this algorithm, tighter timestamp validity rules are necessary, and that miners have incentive to use accurate timestamps.

Our aim in the preceding sections has been to promote understanding of these algorithms when they are seen in operation. We have not attempted to formally specify a particular algorithm, to exhaustively quantify its responsiveness or overall performance, nor to compare its performance to other algorithms. These pursuits remain for the future.

A Appendix - Derivation of Adjustment to T

Retargeting is shown to increase the mean inter-block time by a constant factor, assuming constant hashrate. We find the factor and use it to balance the main algorithm.

By Equations 13 and 14,

$$G_n = G_{(n-1)}\frac{u_n}{T} = G_{n-1}\frac{at_n^k}{k}$$

By Equation 11, G_n will be used as the basis for the search for block $n+1$. We may define a retargeted rate function

$$\lambda_{n+1}(t) = \frac{at_n^k}{k}\lambda(t)$$

which absorbs the effect of retargeting into the rate function for block $n+1$. Repeating the analysis of Equations 6 through 9 gives, briefly,

$$F_{n+1}(t) = 1 - e^{-\frac{a^2 t_n^k t^k}{k^2}}$$

$$\begin{aligned}\mathbb{E}[f_{n+1}(t)] = \mathbb{E}[t_{n+1}] &= \mathbb{E}[\int_0^\infty \frac{a^2 t_n^k t^k e^{-\frac{a^2 t_n^k t^k}{k^2}}}{k} dt] \\ &= \mathbb{E}[\frac{a^2 t_n^k \Gamma\left(1+\frac{1}{k}\right)\left(\frac{a^2 t_n^k}{k^2}\right)^{-\frac{k+1}{k}}}{k^2}]\end{aligned}$$

Simplifying and substituting Equation 10,

$$\mathbb{E}[t_{n+1}] = \frac{T^2}{\Gamma\left(1+\frac{1}{k}\right)}\,\mathbb{E}[\frac{1}{t_n}] \tag{15}$$

The expectation of $1/t_n$ follows from the law of the unconscious statistician and equation 8:

$$\mathbb{E}[\frac{1}{t_n}] = \int_0^\infty at^{k-2}e^{-\frac{at^k}{k}}dt = \frac{\Gamma(1+\frac{1}{k})\Gamma(1-\frac{1}{k})}{T} \tag{16}$$

Substituting Equation 16 into Equation 15,

$$\mathbb{E}[t_{n+1}] = T\cdot\Gamma(1-\frac{1}{k}) \tag{17}$$

We correct this error before it occurs by using a modified value $\hat{T}$ as the "desired" inter-block time:

$$\hat{T} = \frac{T}{\Gamma(1-\frac{1}{k})} \tag{18}$$

B Acknowledgements

We are particularly indebted to Scott Roberts [7], for the many insights and results stemming from his work across several altcoins; to jameslee777 [10, 11] for his work on Komodo, and to Andrew Stone [8, 9] for his seminal presentation of some of these ideas.

References

[1] Satoshi Nakamoto: Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Sheldon Ross: A First Course in Probability, Macmillan Publishing Company, New York, 2nd edition, 1984.

[3] Andre Nicolas: Is the family of exponential distributions closed under scaling?, Mathematics Stack Exchange, https://math.stackexchange.com/q/85578, version 2011-11-25.

[4] R. Bowden, H.P. Keeler, A.E. Krzesinski and P.G. Taylor: Block arrivals in the Bitcoin blockchain, https://arxiv.org/pdf/1801.07447, 2018.

[5] Meni Rosenfeld: Predicting Block Halving Party Times, https://arxiv.org/pdf/1708.05185, 2017.

[6] S. Noda, K. Okumura, Y. Hashimoto: An Economic Analysis of Difficulty Adjustment in Proof-of-Work Blockchain Systems, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3410460, page 19, 2019.

[7] Scott Roberts: TSA: Change Difficulty During the Block, https://github.com/zawy12/difficulty-algorithms/issues/36, retrieved Sep. 2019.

[8] Andrew Stone: The Blockchain Difficulty Information Paradox, https://medium.com/@g.andrew.stone/the-blockchain-difficulty-information-paradox-879b0336864f, retrieved Sep. 2019.

[9] Andrew Stone: Tail Removal Block Validation, https://medium.com/@g.andrew.stone/tail-removal-block-validation-ae26fb436524, retrieved Sep. 2019.

[10] jl777: AdaptivePoW: the solution to diff stranding of smaller blockchains, https://medium.com/@jameslee777/adaptivepow-the-solution-to-diff-stranding-of-smaller-blockchains-425609df5563, retrieved Sep. 2019.

[11] jl777: Evolution of AdaptivePoW, https://medium.com/@jameslee777/evolution-of-adaptivepow-dfea220d343f, retrieved Sep. 2019.