

Analysis of Difficulty Control in Bitcoin and Proof-of-Work Blockchains

Daniel Fullmer

A. Stephen Morse

Abstract—This paper presents a stochastic model for block arrival times based on the difficulty retargeting rule used in Bitcoin, as well as other proof-of-work blockchains. Unlike some previous work, this paper explicitly models the difficulty target as a random variable which is a function of the previous block arrival times and affecting the block times in the next retargeting period. An explicit marginal distribution is derived for the time between successive blocks (the blocktime), while allowing for randomly changing difficulty. This paper also aims to serve as an introduction to Bitcoin and proof-of-work blockchains for the controls community, focusing on the difficulty retargeting procedure used in Bitcoin.

I. INTRODUCTION

Bitcoin is a decentralized digital currency (or *cryptocurrency*) operated by an ad-hoc network of computers. It enables peer-to-peer payments without requiring a trusted third party. Bitcoin’s original “whitepaper” [1] was released late 2008, and the currency was launched in 2009. There has been a significant amount of current interest in Bitcoin and alternative cryptocurrencies, as well as the technology underlying Bitcoin, the “blockchain.”

This paper focuses on one aspect of Bitcoin and blockchain-based systems, specifically *difficulty retargeting*, also called *difficulty readjustment* or *difficulty control*. Some existing published analysis of difficulty control in Bitcoin may be found in [2] and [3]. In [3], the authors note that the block arrival times do not follow a Poisson distribution and present a variety of modeling alternatives, testing them against real data from the Bitcoin blockchain. In [2] the Bitcoin mining process is treated as a nonhomogeneous Poisson process with a deterministic intensity function $\lambda(t)$. Their analysis principally focuses on the design of a difficulty retargeting algorithm under the assumption of an exponentially increasing hashrate. However, as noted in the paper, [2] does not account for the fact that $\lambda(t)$ is itself a stochastic process depending on the time of the arrivals of the process.¹ This paper explicitly considers this case, and furthermore derives a marginal distribution for the time between successive blocks as well as their expected value and variance. In order to derive these results, a stochastic model for block arrival times is developed as well.

This work was supported by National Science Foundation grant no. 16907101.00 and US Air Force grant no. FA9550-16-1-0290. Daniel Fullmer and A. Stephen Morse are with the Department of Electrical Engineering, Yale University, {daniel.fullmer, as.morse}@yale.edu.

¹Poisson processes whose intensity functions are themselves stochastic processes are sometimes called “Cox processes” [4] or “doubly stochastic Poisson processes.” The authors of this paper are not aware of the study of general Poisson processes whose intensity function depends on previous arrivals in the same way as considered in this paper.

While this paper specifically focuses on the difficulty retargeting rule used in Bitcoin, the analysis applies to a number of related cryptocurrencies relying on proof-of-work which use a similar retargeting rule.

In [section II](#), Bitcoin and blockchains are described and motivation is given for the difficulty adjustment mechanism. In [section III](#), the stochastic model for block arrival times is developed. In [section IV](#), this model is analyzed and the main result of a marginal distribution, expected value, and variance for block times is presented. Finally, in [section V](#), simulations of the block arrival process are presented and compared with the analytical results given in the previous section.

II. BACKGROUND

The following description of Bitcoin and blockchains omits certain details which are not relevant to the specific problem considered in this paper. With that said, it is intended to describe and motivate the purpose of blockchains for cryptocurrencies, the relevant property proof-of-work blockchains ensure (immutability), and the purpose of difficulty retargeting. Readers already familiar these concepts may skip to [section III](#). Readers desiring additional details are encouraged to read [1], [5].

Bitcoin is a cryptocurrency which relies on a public ledger of transactions. All transactions are recorded on this public ledger, called the “blockchain”. This ledger may be thought of as an ordered list of transactions. Each transaction includes the address of the sender, the recipient, the amount, and a digital signature from the sender. Senders of the currency can create, sign, and submit transactions to be included on the public ledger. Recipients can check that transactions are valid and included on the ledger (confirmed). A transaction is valid if it meets a number of criteria, including, if digital signature is valid and the payer has enough currency (as determined by the history of valid transactions on the ledger before that transaction.) There also is a special type of transaction for creating new currency in the system.

Previous digital currency systems required a trusted third party intermediary to maintain and publish the ledger. However, a third party maintainer of the public ledger must always be online and available. It is a centralized, single point of failure. Although such a potential maintainer is a trusted third party, it is not able to forge the digital signatures required for valid transactions. As a result, it cannot arbitrarily transfer funds from one user to another. It can, however, add, remove, or reorder previous transactions on the ledger, as well as censor transactions from particular users. The ability to add, remove, or reorder transactions

in the past may invalidate later transactions. As a simple example, if one user transfers some amount of currency to another user to another exchange for some good, and the maintainer later removes this transaction, the original user then has both the good as well as his or her original currency, and the other user has nothing. So, recipients in the system would want to ensure that previous transactions are unchangeable (*immutable*). The fundamental innovation of Bitcoin was to create a distributed public ledger which could ensure immutability for past transactions. This distributed public ledger is the *blockchain*.

In a blockchain, transactions are grouped into blocks. Each block contains a reference to the previous block, creating a chain. Every user of the system keeps a copy of the blockchain². New blocks are created by users who decide to participate in *mining*. These users are called *miners*. They collect new transactions, attempt to create (*mine*) a new block including those transactions, and publish the newly created block to all other users. However, mining a new block is intentionally difficult. It requires a *proof-of-work* [6], [7]. This proof-of-work can be thought of as a solution to a difficult mathematical puzzle which depends on the data in the candidate block. The purpose of the proof-of-work is to enable immutability of the blockchain, as described below.

For our purposes, a proof-of-work function is a function $W : \mathcal{B} \times \mathbb{Z} \rightarrow [0, 1]$, where \mathcal{B} is the set of valid blocks, and \mathbb{Z} is the set of integers. The goal is to find, given a block $b \in \mathcal{B}$, an integer “nonce” η satisfying the following:

$$W(b, \eta) < \frac{1}{d} \quad (1)$$

for some (large) difficulty target $d > 0$. The function W is assumed to be a random oracle, which means that each unique evaluation of W produces a random number uniformly in $[0, 1]$. Each miner repeatedly evaluates W with different nonces until they find one satisfying the difficulty target. Attempts to find a solution are successful with some small probability as determined by d . If they find one, their block with the included nonce is considered valid and will be accepted by others in the network, and we say that the miner has successfully found a block, or that the miner has *mined* a block. A block and nonce with a low $W(\cdot)$ value is proof that the miner has done a significant amount of work. For Bitcoin, this proof-of-work function is based on the SHA-256 hash function [8], but the details of this are not necessary for this paper, and the model in (1) will suffice.

Successfully mining a block includes a block reward, which is a special transaction creating a predetermined amount of new currency which is allocated to the miner who successfully mined the block. As a result, there is significant incentive for each miner to dedicate computational resources to the task of mining.

Since various miners may produce multiple blocks based on the same previous block, multiple versions of the blockchain may exist simultaneously on the network, but

²This is not strictly true, but is one of the details which is not relevant for our discussion.

there is incentive for miners and users to come to a consensus on one version. The rule which leads to consensus is this: the longest³ valid blockchain is the canonical one. A block is valid if all transactions in the block are valid, the previous block is valid, and the proof-of-work is satisfied. Because of this rule, miners are incentivized to mine new blocks which based on the existing longest blockchain, so that their rewards are accepted by all other users.

As previously mentioned, the proof-of-work system contributes to the immutability of the blockchain. If, for instance, an adversarial miner wants to remove some transaction in the past, he or she could create a new version of the blockchain based on the block immediately preceding the targeted transactions with that specific transaction omitted. However, in order for this new blockchain to be accepted as canonical, it would have to become longer than the existing blockchain. If the adversary controls less than half of the mining processing power, it will mine blocks less frequently (on average) than the miners mining on top of the existing blockchain. It is unlikely the adversary’s blockchain could surpass the existing blockchain, and for this reason, blocks far in the past are treated as immutable by the users in the system. For more details, see [1].

Recalling (1), if the difficulty target d is too high or too low, solutions will be found by miners too frequently or infrequently. The desired goal is to have new solutions (and therefore new blocks) found every $\beta = 10$ minutes, on average. This parameter β was chosen as a tradeoff between ensuring blocks have sufficient time to propagate to all users in the network, and ensuring that transactions do not take too long to be confirmed (included on the blockchain). Since miners continue to dedicate additional computational resources to the task of mining, absent any accommodating factor, blocks would be mined too frequently. So, there is a difficulty retargeting algorithm as part of the Bitcoin consensus rules which adjusts the difficulty upward if blocks are found too frequently, or adjusts the difficulty downward if blocks are found too infrequently. This may be thought of as a *difficulty control problem* integral to blockchains which rely on proof-of-work.

A. Notation

For a random variable X , $X \sim \text{Dist}(\cdot)$ is denotes that X is distributed according to some distribution $\text{Dist}(\cdot)$. For a continuous random variable X , $f_X(x; \theta)$ represent the probability density function of X parameterized by θ . The families of probability distributions used in this paper are $\text{Exp}(\lambda)$, $\text{Erlang}(N, \lambda)$, and $\text{Lomax}(N, \lambda)$ which are the exponential, Erlang, and Lomax distributions with rate parameter λ and shape parameter N .

III. PROBLEM FORMULATION

Suppose blocks are found at the times given by the random variables $0 \leq t_1 \leq t_2 \leq \dots$ with the initial block time $t_0 =$

³More precisely, the canonical blockchain is the one with the most accumulated proof-of-work.

0. The time between blocks is denoted by $X_k = t_k - t_{k-1}$ for $k \geq 1$, and is called the *blocktime* for block k .

Recall that we treat the proof-of-work function W as a random oracle, meaning each unique evaluation samples uniformly a real value between 0 and 1. As a result, the process of repeatedly evaluating $W(\cdot)$ until a nonce is found which satisfies the difficulty target may be thought of as Bernoulli trials. As such, the number of evaluations needed until a success is found follows a geometric distribution. The continuous analogue of a geometric distribution is the *exponential distribution*. The limiting behaviour of such a geometric distribution as the number of parallel evaluations and difficulty increases to infinity follows an exponential distribution. See section 2.2.5 of [9].

In a similar way as in [2], for each $k \geq 1$, the random variable X_k is assumed to be distributed according to an exponential distribution with a rate λ_k given by

$$\lambda_k = \frac{r_k}{d_k} \quad (2)$$

where d_k and r_k are two positive real (random) variables, called the *difficulty* and the *hashrate* respectively. The hashrate may be thought of as representing the sum of the computational resources dedicated toward mining at that time. This is determined exogenously by the miners. The difficulty, however, is updated automatically according to the Bitcoin consensus rules. Recall that the expected value of an exponentially distributed random variable is equal to the inverse of its rate. So, given a known λ_k , $\mathbb{E}[X_k|\lambda_k] = 1/\lambda_k$.

The design of Bitcoin includes a ‘‘difficulty retargeting’’ process which periodically updates the difficulty as the hashrate increases or decreases. The goal is to have a new block found according to a desired blocktime $\beta = 10$ minutes (in expectation). The difficulty is adjusted according to⁴

$$d_{k+1} = \begin{cases} \frac{N\beta}{\sum_{i=1}^N X_{k-N+i}} d_k & \text{if } k \bmod N = 0 \\ d_k & \text{otherwise} \end{cases} \quad (3)$$

where $N = 2016$ is the number of blocks in each *difficulty retargeting period* and d_1 is assumed to be initialized arbitrarily. Note that the difficulty is constant between difficulty readjustments. Intuitively, if the time to mine the previous N blocks took longer than $N\beta$, then the difficulty is decreased. Likewise, if the time to mine the previous N blocks was shorter than $N\beta$, then the difficulty is increased.

In this paper, for simplicity, we additionally suppose that r_k remains constant during each retargeting period.

We concerned with computing the marginal distribution of the blocktimes X_k , $k \geq 1$, along with the expected value and variance of block times while accounting for randomly varying difficulty according to (3). These results may be found in the sequel as [Theorem 1](#) and [Corollary 1](#).

⁴The update rule used in Bitcoin additionally restricts d_{k+1} to only change by a factor of 4 in either direction. Moreover, the Bitcoin code includes a well-known bug which excludes the final X_k in the sum.

A. Derivation of adjustment algorithm

Below is a description of how such a rule (3) might be derived. Specifically, it’s designed so that λ_k^{-1} (the expected time to mine the k th block) is approximately β , assuming the hashrate is unchanging from the previous to the next period. To derive this update rule, we first attempt to estimate the hashrate in the previous period, r_k , knowing only d_k and the previous X_{k-N+i} , $1 \leq i \leq N$. Toward this end, we estimate the λ_k , and call it $\hat{\lambda}_k$, by setting the expected time to mine N blocks equal to the actual time to mine N blocks.

$$N\mathbb{E}[X_k|\hat{\lambda}_k] = \frac{N}{\hat{\lambda}_k} = \sum_{i=1}^N X_{k-N+i} \quad (4)$$

Let $\hat{\lambda}_k = \hat{r}_k/d_k$, where \hat{r}_k is the estimate of the hashrate in the previous period.

$$\hat{r}_k = \hat{\lambda}_k d_k = \frac{Nd_k}{\sum_{i=1}^N X_{k-N+i}} \quad (5)$$

With this estimate of the hashrate, the goal is to set d_{k+1} such that the expected blocktime of the next block X_{k+1} is equal to β , with r_{k+1} assumed to be equal to \hat{r}_k .

$$\beta = \mathbb{E}[X_{k+1}|\lambda_{k+1}] = \frac{1}{\lambda_{k+1}} = \frac{d_{k+1}}{r_{k+1}} = \frac{d_{k+1}}{\hat{r}_k} \quad (6)$$

From this and (5),

$$d_{k+1} = \frac{N\beta}{\sum_{i=1}^N X_{k-N+i}} d_k \quad (7)$$

which matches the update rule in (3).

IV. ANALYSIS

Since d_k and r_k are assumed to be constant during each retargeting period, it proves convenient to introduce the following notation.

$$\bar{d}_n = d_{(n-1)N+1} = d_{(n-1)N+2} = \dots = d_{nN} \quad (8)$$

$$\bar{r}_n = r_{(n-1)N+1} = r_{(n-1)N+2} = \dots = r_{nN} \quad (9)$$

$$\bar{\lambda}_n = \lambda_{(n-1)N+1} = \lambda_{(n-1)N+2} = \dots = \lambda_{nN} \quad (10)$$

$$T_n = \sum_{k=1}^N X_{(n-1)N+k} \quad (11)$$

for each $n \geq 1$. From this and (3) it follows that

$$\bar{d}_{n+1} = \frac{N\beta}{T_n} \bar{d}_n, \quad n \geq 1 \quad (12)$$

From this and (2), for each $n \geq 1$

$$\bar{\lambda}_{n+1} = \frac{\bar{r}_{n+1}}{\bar{d}_{n+1}} = \frac{\bar{r}_{n+1}T_n}{N\beta\bar{d}_n} = \frac{\bar{r}_{n+1}\bar{r}_n T_n}{\bar{r}_n N\beta\bar{d}_n} = \delta_{n+1} \frac{T_n}{N\beta} \bar{\lambda}_n \quad (13)$$

where $\delta_n = \bar{r}_n/\bar{r}_{n-1}$ for $n > 1$. It proves convenient to define $\theta_n = \frac{N\beta}{\delta_n}$ for $n \geq 1$. So,

$$\bar{\lambda}_{n+1} = \frac{T_n}{\theta_{n+1}} \bar{\lambda}_n \quad (14)$$

for each $n \geq 1$.

So for each $n \geq 1$ and $1 \leq k \leq N$, the block time $X_{(n-1)N+k}$ is exponentially distributed according to $\bar{\lambda}_n$.

However, while $\bar{\lambda}_1$ is a fixed value, each $\bar{\lambda}_n$, $n > 1$ is a random variable. In other words:

$$X_k \sim \text{Exp}(\bar{\lambda}_1) \quad 1 \leq k \leq N \quad (15)$$

$$X_{(n-1)N+k} | \bar{\lambda}_n \sim \text{Exp}(\bar{\lambda}_n) \quad n > 1, 1 \leq k \leq N \quad (16)$$

So, the (conditional) probability density functions are as follows:

$$f_{X_k}(x) = \bar{\lambda}_1 e^{-\bar{\lambda}_1 x} \quad 1 \leq k \leq N \quad (17)$$

$$f_{X_{(n-1)N+k} | \bar{\lambda}_n}(x, \lambda) = \lambda e^{-\lambda x} \quad n > 1, 1 \leq k \leq N \quad (18)$$

Here the distribution of each $X_{(n-1)N+k}$, $n > 1$, $1 \leq k \leq N$ is conditional on the value of $\bar{\lambda}_n$. Since each T_n is the sum of N i.i.d, exponentially distributed random variables whose parameter is $\bar{\lambda}_n$, T_n follows an Erlang distribution with parameters N and $\bar{\lambda}_n$. Similarly,

$$T_1 \sim \text{Erlang}(N, \bar{\lambda}_1) \quad (19)$$

$$T_n | \bar{\lambda}_n \sim \text{Erlang}(N, \bar{\lambda}_n) \quad n > 1 \quad (20)$$

$$f_{T_1}(t) = \frac{\bar{\lambda}_1^N t^{N-1} e^{-\bar{\lambda}_1 t}}{(N-1)!} \quad (21)$$

$$f_{T_n | \bar{\lambda}_n}(t, \lambda) = \frac{\lambda^N t^{N-1} e^{-\lambda t}}{(N-1)!} \quad n > 1 \quad (22)$$

We next derive the conditional density function for $\bar{\lambda}_{n+1} | \bar{\lambda}_n$, $n \geq 1$. Since $\bar{\lambda}_{n+1}$ is monotonically increasing in T_n , we can perform a change of variables from $\bar{\lambda}_{n+1}$ to T_n to determine the p.d.f. of $\bar{\lambda}_{n+1}$ conditioned on $\bar{\lambda}_n$. From (14),

$$f_{\bar{\lambda}_{n+1} | \bar{\lambda}_n}(\lambda' | \lambda) = \left(\frac{d}{d\lambda'} \frac{\theta_{n+1} \lambda'}{\lambda} \right) f_{T_n | \bar{\lambda}_n} \left(\frac{\theta_{n+1} \lambda'}{\lambda}, \lambda \right)$$

From this and (22),

$$\begin{aligned} f_{\bar{\lambda}_{n+1} | \bar{\lambda}_n}(\lambda' | \lambda) &= \frac{\theta_{n+1}}{\lambda} \frac{\lambda^N \left(\frac{\theta_{n+1} \lambda'}{\lambda} \right)^{N-1} e^{-\theta_{n+1} \lambda'}}{(N-1)!} \\ &= \frac{\theta_{n+1}^N (\lambda')^{N-1} e^{-\theta_{n+1} \lambda'}}{(N-1)!} \end{aligned}$$

Note two things: First, this is an Erlang distribution with parameters N and θ_n . Second, this expression is independent of λ . Writing this more succinctly,

$$\bar{\lambda}_n \sim \text{Erlang}(N, \theta_n) \quad \text{where} \quad \theta_n = \frac{N\beta}{\delta_n}, \quad n > 1. \quad (23)$$

Knowing this, the expected value can be easily calculated as

$$\mathbb{E}[\bar{\lambda}_n] = \frac{N}{\theta_n} = \frac{\delta_n}{\beta} \quad (24)$$

for $n > 1$.

Using this, it is possible to derive the p.d.f. for all X_k beyond the initial period. These random variables have distributions whose parameters are themselves random variables, which are referred to as *compound distributions* [10]. Specifically, the distribution for $X_{(n-1)N+k}$, $n > 1$, $1 \leq k \leq N$ is a *Lomax distribution*, which is the result of compounding an exponential distribution (18) with its rate parameter $\bar{\lambda}_n$ set according to an Erlang distribution (23). Computing the

p.d.f. for $X_{(n-1)N+k}$, $n > 1$, $1 \leq k \leq N$ using (18) and (23) gives:

$$\begin{aligned} f_{X_{(n-1)N+k}}(x) &= \int_{\lambda=0}^{\infty} f_{X_{(n-1)N+k} | \bar{\lambda}_n}(x, \lambda) f_{\bar{\lambda}_n}(\lambda) d\lambda \\ &= \int_{\lambda=0}^{\infty} \lambda e^{-\lambda x} \frac{\theta_n^N \lambda^{N-1} e^{-\lambda \theta_n}}{(N-1)!} d\lambda \\ &= \frac{N \theta_n^N}{(x + \theta_n)^{N+1}} \int_{\lambda=0}^{\infty} \frac{(x + \theta_n)^{N+1} \lambda^N e^{-(x + \theta_n) \lambda}}{N!} d\lambda \\ &= \frac{N \theta_n^N}{(x + \theta_n)^{N+1}} \int_{\lambda=0}^{\infty} \text{Erlang}(\lambda; N + 1, x + \theta_n) d\lambda \\ &= \frac{N \theta_n^N}{(x + \theta_n)^{N+1}} \end{aligned}$$

This gives our main result:

Theorem 1: Using the difficulty retargeting rule in (3), for $n > 1$, $1 \leq k \leq N$ the marginal distribution of $X_{(n-1)N+k}$ is the Lomax distribution with parameters N and θ_n . i.e.

$$X_{(n-1)N+k} \sim \text{Lomax}(N, \theta_n), \quad n > 1, 1 \leq k \leq N$$

The expected blocktime and variance are easily computed knowing this distribution.

Corollary 1: Using the difficulty retargeting rule in (3), for $n > 1$, $1 \leq k \leq N$

$$\mathbb{E}[X_{(n-1)N+k}] = \frac{\theta_n}{N-1} = \frac{N}{(N-1)\delta_n} \beta$$

assuming $N > 1$. Additionally,

$$\begin{aligned} \text{Var}[X_{(n-1)N+k}] &= \frac{\theta_n^2 N}{(N-1)^2 (N-2)} \\ &= \frac{N^3}{(N-1)^2 (N-2) \delta_n^2} \beta^2 \end{aligned}$$

assuming $N > 2$.

If, instead, the difficulty d_k was assumed to be constant, each blocktime would indeed be distributed according to an exponential distribution with fixed rate parameter $\lambda = 1/\beta$, whose expected value would be β and variance would be β^2 . It is clear that the difficulty retargeting procedure in (3) leads to slightly higher expected value and variance. So, even in the case of constant hashrate $\delta_n = 1$, the Bitcoin blockchain runs too fast by a factor of $N/(N-1)$. However, for the value of N used in Bitcoin, 2016, $N/(N-1)$ is very close to 1.

One modification to (3) which would provide slightly better results would be to change N to $(N-1)$. With this modification, $\theta_n = (N-1)\beta/\delta_n$. And, supposing $\delta_n = 1$, the expected value of $X_{(n-1)N+k}$, $n > 1$, $1 \leq k \leq N$ is just β , as desired, and its variance is $\frac{N}{N-2}\beta^2$.

V. SIMULATIONS

In this section, we sample a realization of the random variables X_k and d_k for $k \geq 1$ and different values of the parameter N . Figure 1 and Figure 2 are two realizations of the stochastic process X_k , for $N = 2$ and $N = 20$

REFERENCES

- [1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," www.bitcoin.org, p. 9, 2008.
- [2] D. Kraft, "Difficulty control for blockchain-based consensus systems," *Peer-to-Peer Networking and Applications*, vol. 9, no. 2, pp. 397–413, 2016.
- [3] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Block arrivals in the Bitcoin blockchain," *arXiv:1801.07447*, Jan. 2018.
- [4] A. Cox, "Some Statistical Methods Connected with Series of Events," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 17, no. 2, pp. 129–164, 1955.
- [5] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.
- [6] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in *Advances in Cryptology - CRYPTO' 92*, pp. 139–147, Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.
- [7] A. Back, "Hashcash - A Denial of Service Counter-Measure," www.hashcash.org/papers/hashcash.pdf, no. August, pp. 1–10, 2002.
- [8] N. I. o. S. and Technology, "Specifications for the Secure Hash Standard - FIPS PUB 180-2," *Computing*, vol. 2, pp. 1–71, 2002.
- [9] R. Gallager, *Stochastic processes: theory for applications*. 2013.
- [10] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions, Vol. 1*, vol. 2. Wiley-Interscience, 2nd ed., 1994.

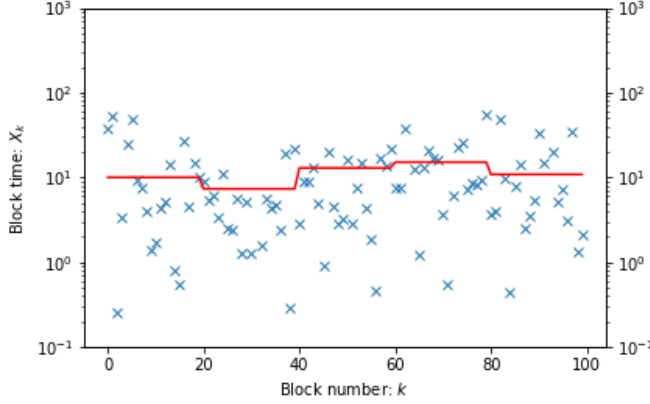


Fig. 1. Sampling with $N = 20$

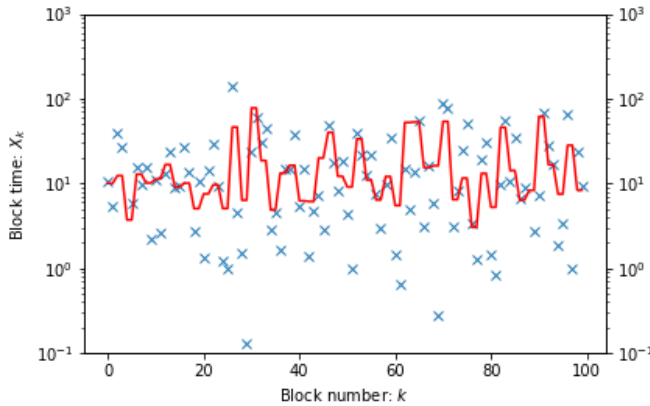


Fig. 2. Sampling with $N = 2$

respectively. The blue 'x's represent individual block times X_k , and the red line represents the value of $1/\lambda_k$, which is the expected value of X_k . Note that the y-axes used in these figures are logarithmic.

In these simulations the parameter β is set to 10 and λ_1 set to $1/10$, so the blocktimes in the initial period have an expected value of 10. For Figure 2, λ_k is adjusted every other unit of time based on the values of X_k for the previous two blocks, which leads to significantly more variation in the value of λ_k , as compared with Figure 1. The quality of the difficulty adjustment algorithm may be intuitively evaluated by how closely the red line stays to the value 10. As can be seen by Corollary 1, the variance of these block times becomes particularly bad for small values of N . In fact, for $N = 2$, the blocktimes have infinite variance, as a result of them being Lomax-distributed.

VI. CONCLUSION

Future work may consider additional difficulty retargeting rules used in other cryptocurrencies, as well as studying the interaction between multiple blockchains which share a common proof-of-work scheme.