TL;DR

In the difficulty adjustment algorithm for Homestead the scalar represented as max(1 - (block_timestamp - parent_timestamp) // 10, -99)

scales linearly based on the blocktime when the probability that a certain blocktime occurs is based off of an exponential function. This is a mismatch and has interested me for some time. Recently, returning to the problem and after learning they are represented best by a Poisson Point Process I have a theory you can use this fact to build a different, simpler, algorithm that also better fits the probability distribution. Hopefully this means better responsiveness and less unnecessary adjustments. I am unsure, but I am at the point where I could use help simulating different cases. Anyone interested in helping me out on this? A deeper explanation of the discovery and possible solutions below.

Intro

Previously at the Status Hackathon before Devcon last year, Jay Rush and I researched the effects of the time bomb on the up and coming fork. During this time I also noticed that the distribution of observed Blocktimes wasn't normal. It followed some kind of exponential distribution but not exactly. Lucky for me Vitalik pointed out that this is because it is a Poisson Point Process + a Normal Distribution based on block propagation in the network. I view it as each miner is a racehorse, but the horse only runs after he hears the starting pistol. For each miner the starting pistol is the previous block and it takes time for that "sound" to get around.

Mining is Poisson Point Process because each blocktime is completely independent from the previous block time. This leads to a Poisson Distribution which wikipedia has a better explanation than I can write.

In probability theory and statistics, the Poisson distribution

named after French mathematician Siméon Denis Poisson, is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant rate and independently of the time since the last event

Data

This is the Distribution of just under 10000 blocks spanning from Sept 7th to Sept 8th 2018. Thank you Jay Rush for this! I still have the data from when we worked on the Time-bomb together. It plus the charts below are available on the Google Sheet I used to prepare for this post.

You can see after 3 seconds there is some kind of exponential curve. This in isolation isn't very interesting, but if you look at this and at how the difficulty adjustment algorithm adjusts to block times it gets a lot more interesting.

The Formula

block_diff = parent_diff + parent_diff // 2048 * max(1 - (block_timestamp - parent_timestamp) // 10, -99) + int(2** ((block.number // 100000) - 2))

Here you can see that if the blocktime

(hereafter bt

) is less than ten it adds parent_diff // 2048 * 1

. The 1

in this changes for every bucket of 10. If the previous blocktime is >= 40 and < 50 then the scalar is -3

. Or, is subtracts parent_diff // 2048 * -3

. At a bt > 1000

this scalar is capped at -99

. A great stackexchange post that goes into this further

This is clearly a linear relationship.

Now lets look at the distribution of blocks within each of these buckets. Here I also add an exponential line of best fit. I understand that this isn't enough data to get the real values of these probabilities, but it is fairly good estimation. And, the important relationship is shown where the linear adjustment doesn't match the exponential distribution.

It is easier to see this side by side.

So is there another approach?

First I want to say that this isn't a proposal expecting to change Ethereum's Difficulty Adjustment Formula. I want to understand it fully and then publish my findings. That is all. And, perhaps write an EIP for the experience of writing one. I don't know the real world affects of this mismatch and if it is a real problem or not for the current chain. I also understand most of the focus here is on Serenity which doesn't have any of these problems. That being said, this is where I have gotten so far.

Start from the Poisson Point Processes you want to see.

[

image

1065×436 86.5 KB

](https://ethresear.ch/uploads/default/original/2X/6/69a8ab33fd43b60f45b538735ba448b00b7075d9.jpeg)

A homogeneous Poisson Point Process is completely characterized by a single number λ

[1 page 23]

λ

is the mean density. In the case of block times it would be the average number of blocks per some unit of time. What is nice about this distribution is you can calculate the probability of occurrences in a given time period.

e = Euler's number λ = average number of events per interval k = number of events occurred

In our collected data above block time was an average 14.5 seconds per block. Some example calculations from this number.

Within 200 seconds what is the probability we would see n

blocks.

(λ = 200/14.5 = 13.8)

[

image

236×511 4.18 KB

](https://ethresear.ch/uploads/default/original/2X/2/2fe8dc1e7bb46e3c5465ea0dbd5ab8dbd5a4be07.png)

We can also ask:

- What is the probability there will be between 10 and 16 blocks within a 200 second period?
- 65.4%
- 65.4%

Now given this information we can also ask this in reverse. Given a set of data from a Poisson point process what is the probability that the mean is λ

? Or in Ethereum Terms given the number of blocks within a fixed time period what is the probability the average blocktime is 14.5?

I will note this is due to it being a Marked Poisson Point Process and the Marking Theorem described briefly below.

An illustration of a marked point process, where the unmarked point process is defined on the positive real line, which often represents time. The random marks take on values in the state space

known as the mark space

. Any such marked point process can be interpreted as an unmarked point process on the space

. The marking theorem says that if the original unmarked point process is a Poisson point process and the marks are stochastically independent, then the marked point process is also a Poisson point process on

. If the Poisson point process is homogeneous, then the gaps in the diagram are drawn from an exponential distribution[2] - Marked Poisson point process

The graph above is similar to plotting blocks over time. The following is a chart of the blocks over constant time (seconds) starting with the block found on Sep-08-2018 01:26:20 PM

[

image

841×126 6.02 KB

](https://ethresear.ch/uploads/default/original/2X/5/5b131e812c97e82ee86056d00544522afb64f207.png)

Given any block we can look backwards at the last 200 seconds and observe the probability that currently λ

is 14.5. If it isn't then that means the difficulty is not tuned properly. In the graph above three sections are conveniently segmented. Below I counted the number of blocks and noted the probability

of this many blocks being found based on λ = 14.5

- 48400 - 48600 : 20 Blocks - 2.61%
- 48600 - 48800 : 18 Blocks - 5.22%
- 48800 - 48900 : 17 Blocks - 6.81%

Not particularly useful, as a 6.81% likelihood is not enough confidence to assert λ

is in fact not 14.5. In practical terms asserting that the current block difficulty should be changed. But, If in case you use a range like the before mentioned between 10 and 16 blocks there is a 65.4% probability. Stated inversely, if there is less than 10 or greater than 16 blocks, there is a 65.4% likely hood λ

isn't 14.5. This means when adjusting the difficulty outside of these ranges you are, more often then not, correct to do so. A sufficient answer to the question "Should the algorithm change the current difficulty?".

My theory is there is an ideal length of time to check such that

new_difficulty = parent_difficulty + parent_difficulty*scalar

Where the scalar is defined by the probability an observed number of blocks within a previous x

seconds has a λ

equal to a Target Blocktime.

This is where I believe simulation and testing would give good enough observable results to decide on ideal values for a target λ

. This and as well as more rigorous maths narrowing testing ranges for:

- Different period lengths
- Different adjustment thresholds
- Magnitudes of adjustment

A more specific example.

In the case greater than 17

is selected as the upper adjustment threshold. The scalar can be defined to increase in magnitude proportional to the following equation.

The difference of the scalar between 18 and 20 would be nearly a 2x increase. This would much more closely resemble the exponential distribution in observed blocktimes.

Using this method I hope to document a practical alternative to targeting a blocktime for a PoW chain. One that is simple to calculate as well as follows more closely the actual distribution found in blocktimes.

Please any feedback from mathematicians, statisticians, computer scientists, or interested parties would be appreciated. As well as anyone willing to look at testing and simulation implementations. I also have no idea if this bares anything on the beacon chain found in Serenity. Which if it does, I would love to know.

Cheers,

James

References and Sheets

- [1] Page 23 - Homogeneous Poisson Point Processes

www2.eng.ox.ac.uk

[

](https://www2.eng.ox.ac.uk/sen/files/course2016/lec5.pdf)

lec5.pdf

- [2] Poisson Point Process - Marked Poisson point processhttps://en.wikipedia.org/wiki/Poisson_point_process
- A great stackexchange post that describing the current adjustment algorithm https://ethereum.stackexchange.com/questions/5913/how-does-the-ethereum-homestead-difficulty-adjustment-algorithm-work
- Google Sheet and Charts on Ethereum Block Times
- Worksheet Used to Calculate Poisson Process Distribution Values