Modeling, Identification and Control, Vol. 39, No. 3, 2018, pp. 151–156, ISSN 1890–1328

Tail Removal Block Validation: Implementation and Analysis

J. Kučera [1] G. Hovland [2]

[1] *Bismuth Foundation, Lead Developer, District Ostrava-City, Czech Republic*

[2] *Department of Engineering Sciences, University of Agder, N-4898 Grimstad, Norway*

Abstract

In this paper a solution for the removal of long tail blocktimes in a proof-of-work blockchain is proposed, implemented and analysed. Results from the mainnet of the Bismuth blockchain demonstrate that the variances in the key variables, difficulty level and blocktime, were more than halved after the tail removal code was enabled. Low variances in difficulty and blocktimes are desirable for timely execution of transactions in the network as well as reduction of unwanted oscillations in the feedback control problem.

Keywords: Blockchain, block times, long tail removal, feedback control

1 Introduction

The first blockchain-based cryptocurrency was Bitcoin and the original publication was Nakamoto (2009). Since then hundreds of new cryptocurrency projects have emerged. Bitcoin and many other blockchain projects are based on the concept of Proof-of-Work (POW) where miners compete to solve a difficult cryptographic puzzle linking the blocks together. Bitcoin aims to keep the average time between blocks (the blocktime) constant at 10 minutes by adjusting the so-called difficulty level, but because of the stochastic nature of the problem there are large variations in the times it takes to generate a new block, even when the difficulty level is perfectly adjusted to the computational power of the miners. While the average blocktime in Bitcoin is 10 minutes, individual blocks can take anything from a few seconds to more than 100 minutes to generate.

The probability density function (PDF) describing the distribution of blocktimes in Bitcoin has a long tail, meaning that there is a small, but nonzero, probability that it can take a very long time to generate a new block, even when the computational power of the miners is constant or increasing. Such long blocktimes is a problem for two reasons: 1) Long processing times of transactions is undesirable. Processing times which are many factors larger than the desired average blocktime are seen as negative, 2) a blockchain feedback control algorithm can typically not distinguish between a long tail blocktime and the situation where the computational power of the miners has dropped. Hence, a long tail blocktime will normally cause a fast-responding controller to lower the difficulty when it should not do so, and this behavior can cause unwanted oscillations or unstability in the process.

Stone (2017) stated the following when refering to the development of a feedback controller to handle the long tail problem: *When confronted with a "very hard to impossible" problem, the best choice is often to redefine the problem rather than search for a marginal solution.* Such an approach is not uncommon in feedback control of physical systems either, for example the case study of unmanned aerial vehicles in Magnussen et al. (2014). In that paper the actuator dynamics were modified to remove a nonlinearity and as a result make the control problem easier.

In this paper a solution for preventing long blocktimes in an open source, public blockchain project is

doi:10.4173/mic.2018.3.1

proposed, implemented and the performance analysed. The selected blockchain for the implementation is Bismuth programmed in Python, Kučera (2018). For background information and a description of how the blockchain and the POW mining process in Bismuth work, see Hovland and Kučera (2017).

2 Blocktime Probability Density Functions

In Grunspan and Pérez-Marco (2017) it is stated: *At each new hash the work is started from scratch, therefore the random variable T measuring the time it takes to mine a block is memoryless.* T is an exponentially distributed random variable:

$$f_T(t) = \alpha e^{-\alpha t} \tag{1}$$

for some parameter $\alpha > 0$, the mining speed, with the expected value of T equal to $1/\alpha$. This function is plotted in Fig. 1 (red) against the probability distribution of blocktimes in the Bitcoin blocks 514,456-530,336 (blue). As can be seen from Fig. 1 the PDF given by eq. (1) matches well with the data for the Bitcoin blocks 514,456-530,336, period March-July 2018.

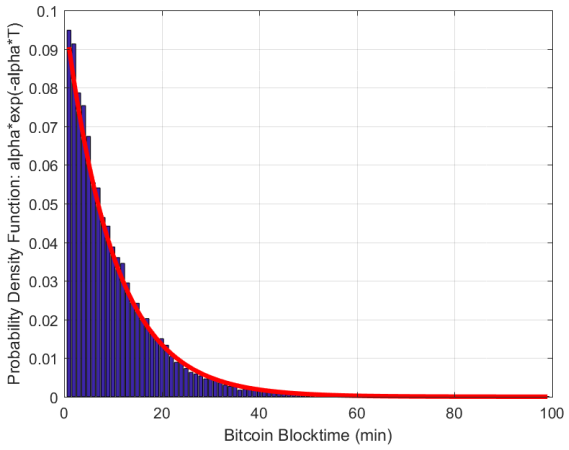

Figure 1: *Theoretical probability density function (red) for the bitcoin blocktime compared against data in blocks 514,456-530,336 (blue).*

The Bitcoin blockchain will, in a year, have approximately 52,560 new blocks since the average blocktime is 10 minutes. The probability of an event which occurs for one block per year is therefore $1/52560 = 1.9\times10^{-5}$.

$$\alpha = \frac{1}{10} \tag{2}$$

$$\int_{108}^{\infty} f_T(t)dt = 1.9 \times 10^{-5} \tag{3}$$

Hence, statistically once per year a blocktime of 108 minutes or larger should occur in the bitcoin network when the mining hashpower stays constant as seen by eq. (3). This matches quite well with data obtained from the Bitcoin blockchain for the period 2011-2016 shown in Table 1, even though there was a significant growth in computational power (hashrate) by the miners during this period. As Table 1 shows there was approximately one block per year which took 100 minutes or more in the Bitcoin network. From Nov. 2011 to Oct. 2016 the Bitcoin hashrate grew exponentially from 9TH/s to 2,000,000 TH/s. The difficulty in Bitcoin is updated only every 2016 blocks (14 days) and not continuously in every block. In periods of rapid growth in hashrate, this will cause the blocktime average to become less than the target which is 10 minutes for Bitcoin and there will also be fewer of the long tail blocktimes compared with the expected number calculated from eq. (3). If the Bitcoin hashrate would stabilize or even decrease in the future the occurrence of long tail blocktimes would increase compared with the historical data.

Blocks	Date	Time (min)
152,217 - 152,218	2011-11-07	100
270,193 - 270,194	2013-11-17	129
334,486 - 334,487	2014-12-15	102
338,697 - 338,698	2015-01-12	109
425,379 - 425,380	2016-08-15	108
435,031 - 435,032	2016-10-19	102

Table 1: *Examples of bitcoin blocks which took 100 minutes or longer (long tail).*

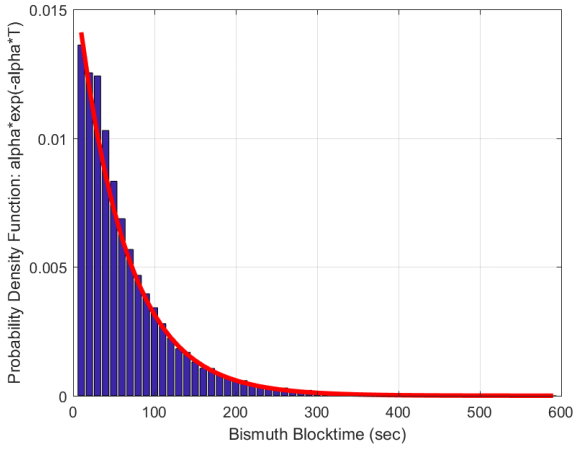

Figure 2: *Theoretical probability density function (red) for the Bismuth blocktime compared against data in blocks 675,000-700,000 (blue).*

```
def difficulty(c):
 execute(c, "SELECT block_height, timestamp FROM transactions WHERE reward != 0 ORDER BY block_height DESC LIMIT 2")
 block_height, timestamp_last = c.fetchone()
 timestamp_before_last = c.fetchone()[1]

 execute_param(c, "SELECT timestamp FROM transactions WHERE block_height > ? AND reward != 0 ORDER BY timestamp ASC LIMIT 2", (block_height - 1441,))
 timestamp_1441 = c.fetchone()[0]
 block_time_prev = (timestamp_before_last - timestamp_1441) / 1440
 timestamp_1440 = c.fetchone()[0]
 block_time = (timestamp_last - timestamp_1440) / 1440
 execute(c, "SELECT difficulty FROM misc ORDER BY block_height DESC LIMIT 1")
 diff_block_previous = c.fetchone()[0]

 time_to_generate = timestamp_last - timestamp_before_last

 hashrate = pow(2, diff_block_previous / 2.0) / (block_time * math.ceil(28 - diff_block_previous / 16.0))
 # Calculate new difficulty for desired blocktime of 60 seconds
 target = 60.0
 difficulty_new = (2 / math.log(2)) * math.log(hashrate * target * math.ceil(28 - diff_block_previous / 16.0))
 # Feedback controller
 Kd = 10
 difficulty_new = difficulty_new - Kd * (block_time - block_time_prev)
 diff_adjustment = (difficulty_new - diff_block_previous) / 720

 if diff_adjustment > 1:
  diff_adjustment = 1

 difficulty_new_adjusted = diff_block_previous + diff_adjustment
 difficulty = difficulty_new_adjusted

 # Tail Removal Code
 diff_drop_time = 180

 if time.time() > timestamp_last + diff_drop_time:
  time_difference = time.time() - timestamp_last
  diff_dropped = difficulty - time_difference / diff_drop_time
 else:
  diff_dropped = difficulty
 # End Tail Removal Code

 if difficulty < 50:
  difficulty = 50
 if diff_dropped < 50:
  diff_dropped = 50

 return (difficulty, diff_dropped, time_to_generate, diff_block_previous, block_time, hashrate, diff_adjustment, block_height)
```

Figure 3: *Difficulty calculation function (feedback controller) with the single block difficulty drop included.*

```
1  mining_hash = bin_convert(hashlib.sha224((miner_address + nonce + db_block_hash).encode
       ("utf-8")).hexdigest())
2  diff_drop_time = 180
3  mining_condition = bin_convert(db_block_hash)[:diff]
4
5  if mining_condition in mining_hash:  # simplified comparison, no backwards mining
6   app_log.info("Difficulty requirement satisfied for block {} from {}".format(
        block_height_new, peer_ip))
7   diff_save = diff
8
9  # Tail Removal Condition Check Starts Here
10 elif received_timestamp > db_timestamp_last + diff_drop_time:
11  time_difference = received_timestamp - db_timestamp_last
12  diff_dropped = diff - time_difference / diff_drop_time
13  if diff_dropped < 50:
14   diff_dropped = 50
15
16  mining_condition = bin_convert(db_block_hash)[:diff_dropped]
17
18  if mining_condition in mining_hash:
19   app_log.info("Readjusted difficulty requirement satisfied for block {} from {}".
        format(block_height_new, peer_ip))
20   diff_save = diff
21  else:
22   raise ValueError("Readjusted difficulty too low for block {} from {}, should be at
        least {}".format(block_height_new, peer_ip, diff_dropped))
23 else:
24  raise ValueError("Difficulty too low for block {} from {}, should be at least {}".
       format(block_height_new, peer_ip, diff))
```

Figure 4: *Python code for calculating mining condition.*

Bismuth will, in a year, have 525,600 new blocks since the average blocktime is 60 seconds. A probability of an event which occurs for one block per year is therefore $1/525600 = 1.9 \times 10^{-6}$.

$$\alpha = \frac{1}{60} \tag{4}$$

$$\int_{790}^{\infty} f_T(t)dt = 1.9 \times 10^{-6} \tag{5}$$

Hence, statistically once per year a blocktime of 790 seconds (13 minutes and 10 seconds) or larger could have occured in the Bismuth network before the hard-fork at height 700,000, as seen by eq.(5).

3 Tail Removal Implementation

Tail removal means to prevent blocks from taking a long time to generate. In this paper the tail is defined as blocks taking 180 seconds or longer to generate. In Fig. 2 this accounts for 6.2% of the blocks which translates to approximately 90 blocks per day.

The tail removal implementation is only a few lines of code, shown in Fig. 3, lines 34-38, see also Kučera (2018) (file node.py). The code before line 34 contains the difficulty feedback controller developed in Hovland and Kučera (2017) without modification. Note that the modified difficulty variable `diff_dropped` is the second index in the returned array from the function `difficulty(c)`, whereas the unmodified difficulty is in the first index. As seen in lines 34-38 in Fig. 3 the difficulty is reduced linearly by the factor `time_difference / 180`, where `time_difference` is the difference between the current system time and the time when the previous block was validated.

In Fig.4 the modified code for calculating the mining condition is shown. For a more detailed explanation of the mining condition in Bismuth, see Hovland and Kučera (2017). Line 15 in Fig. 4 uses the tail removal variable `diff_dropped` to calculate a new mining condition. Note also that in line 19 the unmodified difficulty (first variable index returned from the function `difficulty(c)` in Fig. 3 is saved instead of `diff_dropped` in order not to affect the feedback control algorithm for future blocks. By design, the feedback controller should respond to external changes in hashrate by the miners, and less to the occurrence of

long tail blocktimes.

4 Results

Fig. 5 shows the original theoretical PDF (red) vs. the new PDF with tail removal (blue) as implemented on the Bismuth mainnet starting from block 700,000. Blocktimes starting from block 701,441 are chosen since the feedback controller in Bismuth considers the previous 1440 blocks, see Fig. 3, Line 10.

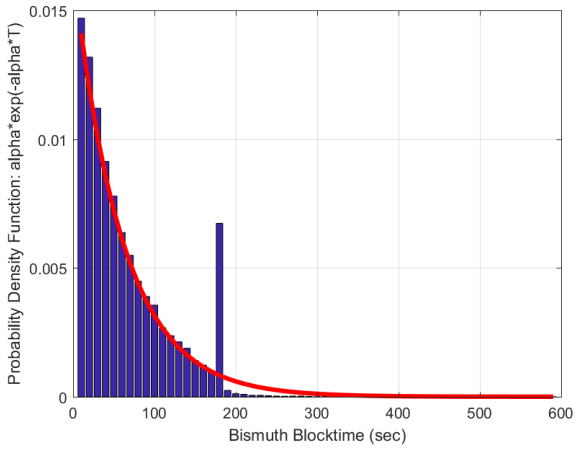

Figure 5: *Theoretical probability density function (red) for the bismuth blocktime compared against data in blocks 701,441-726,440 (blue) which have tail removal activated.*

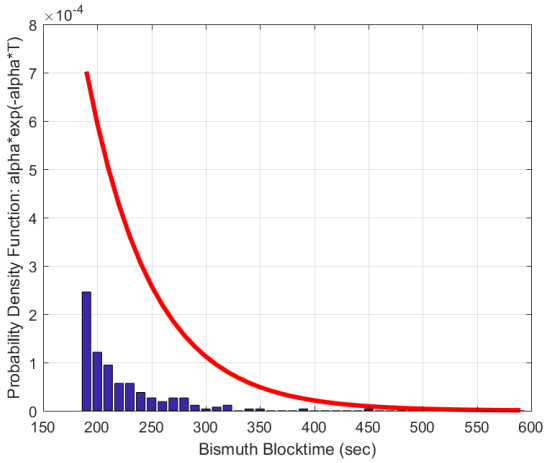

Figure 6: *Theoretical curve (red) vs. Bismuth blocktimes larger than 180 seconds in blocks 701,441-726,440 (blue).*

As can be seen from the plot the tail after blocktime 180 sec has been almost completely removed and there

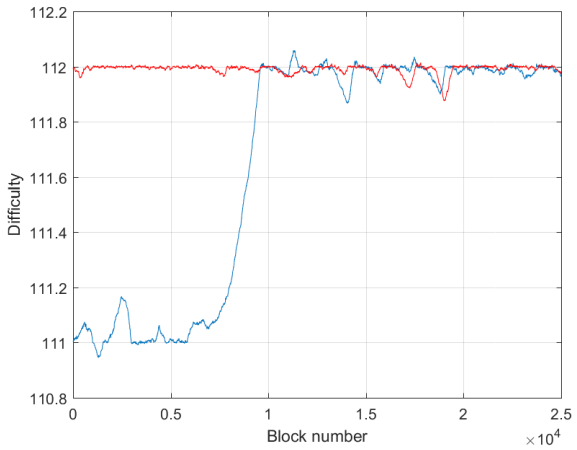

Figure 7: *Mining difficulty: Blue=blocks 675,001-700,000, Red=blocks 701,441-726,440.*

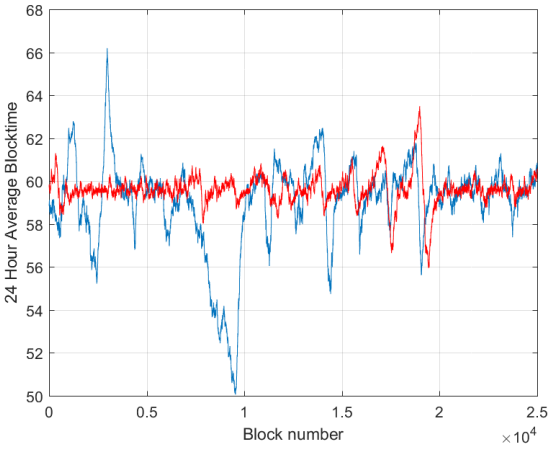

Figure 8: *24 hour average blocktime: Blue=blocks 675,001-700,000, Red=blocks 701,441-726,440.*

is a desired, distinct high probability of bloctime between 180 and 190 seconds. The blocks with blocktime larger than or equal to 190 seconds in Fig. 6 account for less than 1% of the total when the tail removal code is activated, compared to 5.4% before. The largest blocktime in the data was 450 seconds or 7.5 minutes compared to 810 seconds in blocks 675,001-700,000. If more than 180 seconds has passed since the last block was validated, the difficulty is reduced by 1.0, and for every 180 seconds thereafter it is reduced linearly by another 1.0. In the Bismuth blockchain implementation a reduction of difficulty by 1.0 normally results in the miners validating blocks twice as fast as before. Hence, the probability of long tail blocktimes is significantly reduced by the algorithm presented in this

paper.

Variance	685,001-700,000	701,441-726,440
Blocktime	1.3466	0.5333
Difficulty	7.1705×10^{-4}	3.3182×10^{-4}

Table 2: *Variance in 24 hour (1440 blocks) average blocktime and difficulty before and after tail removal.*

Fig. 7 shows the difficulty level in blocks 675,001-700,000 (blue) and in blocks 701,441-726,440 (red). Since the hashrate increased rapidly during blocks 675,001-685,000, the feedback controller caused the difficulty level to increase from 111.0 to 112.0. During this period the 24 hour average blocktime decreased from 60 to 50 seconds, as seen in Fig. 8. Hence, blocks 675,001-685,000 are not included in the results in Table 2. The two periods compared had approximately the same constant hashrate. The table shows that the variance in blocktime and difficulty more than halved when introducing tail removal.

5 Discussion and Conclusions

In this paper a solution for preventing very large blocktimes in a blockchain has been proposed, implemented and analysed based on data stored on Bismuth mainnet. The code was enabled at block height 700,000 and comparison of the network before and after the code was enabled shows that the variance in the key variables, difficulty level and blocktime, have been more than halved when a long tail threshold of 180 seconds, 3 times the desired blocktime of 60 seconds, was chosen.

To avoid creating large number of orphaned blocks at the threshold mark and later, the difficulty is reduced linearly with time instead of a sudden change to a low difficulty level. By using a linear ramp, the risk of a large number of orphans is mitigated. A linear ramp was also proposed in Stone (2017).

One potential side-effect of the long tail removal is a stored shares race by the miners under certain conditions. If such a race turns out to be a problem it could be mitigated with a slight adjustment to the mining condition after the 180 second threshold. Such an adjustment could be made which has no effect upon the performance of the feedback control loop.

In the case where computational power from one or several large miners would suddenly disappear from the network, the proposed long tail removal code would also be beneficial. The feedback controller would adjust the difficulty level relatively slowly down to adapt to the new situation, while the long tail removal code would prevent large blocktimes from occurring during the transient phase until a new steady-state phase has been reached.

Historically blockchain projects such as Bitcoin have experienced rapid (exponential) growth in computational power (hashrate) by the miners. This growth has, to some extent, reduced the occurrences of long tail blocktimes. It is believed by the authors that in the future, when the growth in hashrate levels out or when miners to a larger degree switch frequently between different projects to maximize rewards causing hashrate to decrease in many projects, the focus on stabilizing and performance enhancing functions such as long tail removal is likely to increase.

Acknowledgement

The research presented in this paper has received partial funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

References

Grunspan, C. and Pérez-Marco, R. Double spend races. *CoRR*, 2017. abs/1702.02867. URL http://arxiv.org/abs/1702.02867.

Hovland, G. and Kučera, J. Nonlinear Feedback Control and Stability Analysis of a Proof-of-Work Blockchain. *Modeling, Identification and Control*, 2017. 38(4):157–168. doi:10.4173/mic.2017.4.1.

Kučera, J. Bismuth source code, release 4.2.5.3. https://github.com/hclivess/Bismuth/releases, commit hash 6646704833987a8dd4ecd984cb395c1f27d0de44, 2018.

Magnussen, Ø., Ottestad, M., Hovland, G., and Kirby, S. Experimental study on the influence of controller firmware on multirotor actuator dynamics. In *Proc. IEEE Intl. Symp. on Robotic and Sensors Environments (ROSE 2014)*. 2014. doi:10.1109/ROSE.2014.6952992.

Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, 2009.

Stone, A. Tail removal block validation. https://medium.com/@g.andrew.stone/tail-removal-block-validation-ae26fb436524, 2017.