MODELING AND ANALYSIS OF BLOCK ARRIVAL TIMES IN THE BITCOIN BLOCKCHAIN: ONLINE APPENDIX CONCERNING DATA, MEASUREMENTS AND PROCESSING

R. BOWDEN[1,*], H.P. KEELER[1], A.E. KRZESINSKI[2], AND P.G. TAYLOR[1]

1. The block arrival process: the data

Blocks arrive, that is, they are mined, published to the bitcoin network and are added to the blockchain according to a random process. Examining the block arrival process can serve as a means of diagnosing the health of the blockchain, as well as detecting the actions of malicious entities.

There are several ways in which we can define when a block "arrives". The two that are easiest to measure are:

(1) the time when the block arrives at a measurement node (or nodes), and
(2) the timestamp encoded in the block header.

There are advantages and disadvantages associated with each of these methods. We have forthcoming work on estimating the distribution of errors in the arrival times from the combination of both sources of data.

1.1. Block arrival times.

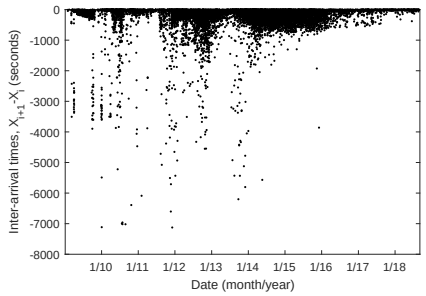
The time when a block arrives at any given node in the Bitcoin peer-to-peer (p2p) network depends upon when the block was mined and how the block propagates through the p2p network. While propagation adds a random delay component to the time when the block was mined, it also represents when a block can truly be said to have arrived at the node. Once a block has propagated through the network (and to measurement nodes), then the information contained within the block is visible to the miners and the users of Bitcoin, and it is much less likely that any miner will mine on top of a previous block and cause a conflict. However, a substantial problem with using block arrival times at measurement nodes is that it is impossible to get historical information, or information from periods when the measurement nodes were not functioning, or not connected to the network.

1.2. Block timestamps.

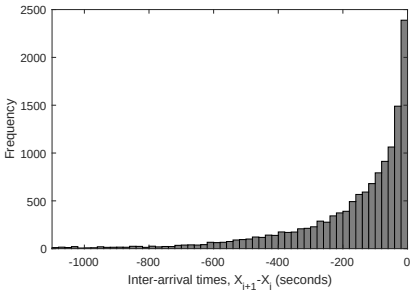
Using the timestamp in the block header as the authoritative time has its own disadvantages. This time is determined from the miner's clock when the mining rig creates the block template. If the miner's clock is incorrect then this will introduce an error. Timestamp errors are limited in magnitude by the Bitcoin protocol: blocks with timestamps prior to the median timestamp of the previous 11 blocks, or more than 2 hours into the future are rejected as being invalid and will not be propagated by the network.

Nevertheless, block timestamp errors can be severe. For example, of the 540,000 blocks mined up to September 2018, 13,807 have a timestamp prior to that of the previous block and some 349 of them have timestamps more than 20 minutes prior

[1]School of Mathematics and Statistics, University of Melbourne, Vic 3010, Australia
[2]Department of Mathematical Sciences, Stellenbosch University, 7600 Stellenbosch, South Africa
(*Corresponding author)
E-mail addresses: rhys.bowden@unimelb.edu.au*, hkeeler@unimelb.edu.au, aek1@cs.sun.ac.za, taylorpg@unimelb.edu.au.



(A) Scatter plot of the sizes of negative inter-arrival times *vs.* when they occurred. Out-of-order timestamps are prevalent throughout the history of Bitcoin, but the process that determines their magnitude is clearly not stationary.



(B) Histogram of those differences between adjacent interarrival times which are negative. There are 394 pairs $(i+1, i)$ that are not shown on the histogram because $X_{i+1} - X_i < -1100$.

FIGURE 1. Known timestamp errors: blocks for which $X_{i+1} - X_i < 0$.

to the previous block timestamp. If one considers that a miner must have access to the previous block header (and hence the timestamp of the previous block) before mining the next block, these errors seem particularly anomalous[1]. This suggests that some miners intentionally use inaccurate timestamps. One potential reason for this is mining software that varies the timestamp to use it as an additional nonce.[2]

Figure 1(a) shows those pairs of blocks $(i+1, i)$ for which $X_{i+1} - X_i < 0$. The difference in timestamp $X_{i+1} - X_i$ is plotted against X_i. The horizontal axis covers the entire history of Bitcoin. These errors are not localized to specific times in history although the magnitude of the errors changes over time. Inconsistent pairs of timestamps remain frequent even recently in the blockchain.

Figure 1(b) shows a truncated histogram of the sizes of the negative inter-arrival times. The most negative inter-arrival time is $-7,125$ seconds but most are between $-1,000$ and 0 seconds.

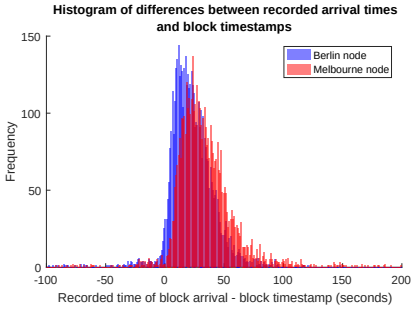
We will later argue that while data for individual block arrivals are unreliable, our estimation of the hash rate is insensitive to errors in the individual arrival times (especially if those errors are independent) since it only relies upon the average rate of arrival of large numbers of blocks.

1.3. **Differences between measured block arrivals and the timestamp data.** We do not know with certainty when any given block was mined, or even when they start propagating through the Bitcoin p2p network. However, we can take multiple measurements of the same block as it arrives at different nodes, and compare these to the timestamp in the block itself. To this end, from November 2016 to January 2017 we measured arrival times for 4700 blocks at two measurement nodes: one in Melbourne, Australia, and one in Berlin, Germany.

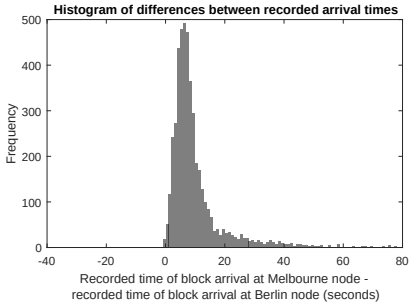
Let X_i denote the timestamp of block i. Let $\tilde{X}_{i,1}$ and $\tilde{X}_{i,2}$ denote the arrival time of block i as measured at Berlin and Melbourne respectively. Figure 2(a) presents histograms of the differences $\tilde{X}_{i,\bullet} - X_i$ between the measured arrival time of block i and the timestamp for that block, one histogram for each measurement node. The figure shows that on average the Berlin node measured a smaller delay

[1]Out-of-order timestamps are often caused by a miner using a timestamp from the future, and then miners for later blocks using a correct timestamp that falls before the incorrect (future) one.

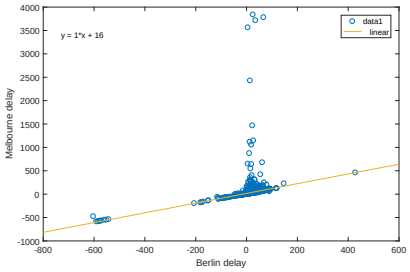
[2]Miners can change both the nonce in the header and another nonce in the coinbase transaction. These two nonces combined allow miners to cycle through 2^{96} hashes before modifying the timestamp.



(A) Histogram of $\tilde{X}_{i,\cdot} - X_i$, one histogram for each measurement node.



(B) Histogram of the difference between the block arrival times at the two measurement nodes.



(C) Scatter plot: block delay Melbourne *vs.* Berlin measurement.

FIGURE 2. Block arrival and block timestamp data.

than the Melbourne node, and that the histograms are similar in shape. The vast majority of the differences are positive, as would be expected if the timestamp was usually close to the true mining time, and the measurement nodes are observing the block arriving after a random delay.

Figure 2(b) presents a histogram of the differences $\tilde{X}_{i,2} - \tilde{X}_{i,1}$ between the measurements for the arrival time of block i at the two measurement nodes.

Figures 2(c) presents a scatter plot of $\tilde{X}_{i,2} - X_i$ vs $\tilde{X}_{i,1} - X_i$. The result of a linear regression on the data is also displayed, showing that there is in general an additive relationship between the two measurements, the Melbourne delay being some 16 seconds larger than the Berlin delay on average when outliers are included. On the other hand, Figure 2(b) shows that difference is mostly concentrated around the mode of 6 seconds.

There is also a cluster of points in Figure 2(c) with $(\tilde{X}_{i,2} - X_i, \tilde{X}_{i,1} - X_i)$ around $(-550, -550)$. These nine blocks at heights 439551, 439965, 440004, 440304, 441396, 442539, 442948, 443922, 444425 have erroneous timestamps. Many of the timestamps for these blocks are later than those of blocks following them in the blockchain. All of these blocks have only one transaction: a coinbase payment to the miner (mining pool) with address `1F1xcRt8H8Wa623KqmkEontwAAVqDSAWCV`. This suggests that while most blocks have timestamps accurate to within a few seconds, there is a small minority of miners who are consistently producing blocks with incorrect timestamps.

To summarize, if we assume that errors in the timestamps (confining ourselves to errors that do not cause the block to become invalid) are approximately independent of the propagation delay through the p2p network, then we can propose a simple

formal model for these data sources that is consistent with the measured data namely:

- P1 Timestamps are accurate to within a few seconds (or even more accurate) for the most part. Occasionally a timestamp has a much larger additive error.
- P2 Arrival time measurements include a random propagation delay according to some distribution. This random propagation delay is usually large compared to the timestamp errors. These delays are correlated between different measurement nodes (Figure 2(c)).

The combination of P1 and P2 means that Fig. 2(a) is predominantly showing values from the distribution of propagation delay for each node, with occasional outliers for the (rare, large) timestamp errors, or other rare errors that greatly delay propagation. The high variance in the measured block arrival times means that we will focus on using block timestamps for the rest of the paper, and attempt to mitigate the rare timestamp errors by cleaning the data.

1.4. **Cleaning the timestamp data.** Blockchain timestamps are recorded in whole numbers of seconds since the 1st of January 1970, with the first block being mined on the 3rd of January 2009. The first step we took was to disregard all data prior to the 30th of December 2009. Owing to Bitcoin being newly adopted over 2009, the block arrival process in this interval had a large number of anomalies, including one day in which no blocks were mined.

The next step was to deal with obvious errors. As discussed in Section 0.2, it is possible to have errors in the timestamps. We know the order that the blocks *must* have been mined in, because each block header contains the hash of the previous block header. Let the timestamp of the i^{th} block be X_i. Due to errors, sometimes the timestamps are out of order: $X_i > X_j$ for some $i < j$. We call $T_i := X_i - X_{i-1}$ the i^{th} *inter-arrival* time, and if T_i is negative then there must have been an error. Some of these errors are both substantial and obvious: for instance if $X_i - X_{i-1} = 7162$ and $X_{i+1} - X_i = -6603$ while other nearby inter-arrival times are unremarkable, then it would be sensible to assume that X_i is an error. However, there are other cases that are more complicated to deal with. There are several approaches that we considered.

(1) **Ignore**: Ignore the fact that the data are out of order. If we then look at the block inter-arrival times, some of these will be negative. Also, some will be very large. If we are studying the variance of the inter-arrival times then this will be inflated relative to any of the methods discussed below. Furthermore, the empirical distribution function for inter-arrival times will not make physical sense, and will certainly not be exponentially distributed.

(2) **Reorder**: Sort the timestamps into ascending order. This results in non-negative inter-arrival times, but we are essentially removing random points and inserting them elsewhere in the process. This has a far from obvious effect, and the effect is dependent on the distribution of these already unreliable timestamps.

(3) **Resample**: Remove those timestamps which we deem unreliable (by some criterion), then resample them uniformly in the interval between the adjacent timestamps that we consider reliable. For instance, say $X_{i+1}, X_{i+2}, X_{i+3}$ are deemed unreliable but X_i and X_{i+4} are not. Then we would independently uniformly sample three timestamps in the interval (X_i, X_{i+4}), sort those new timestamps, and replace the previous values of $X_{i+1}, X_{i+2}, X_{i+3}$ by the three new timestamps. This is essentially replacing the unreliable timestamps with timestamps sampled from a Poisson process on (X_i, X_{i+4}), conditional on there being three timestamps in this interval. There are a few ways we can determine which timestamps are unreliable:

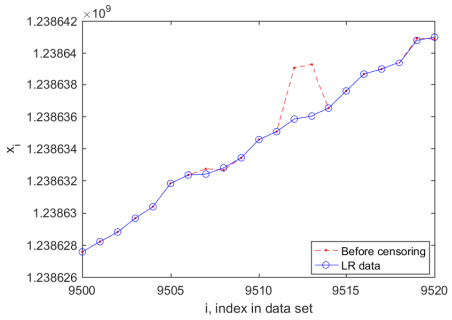


FIGURE 3. An example of the effect of the resampling scheme (3d) from Section 0.4. The index i is the number of blocks since 1 January 2009.

(a) Guess which timestamps are unreliable based on the surrounding timestamps. It is not clear how to do this, especially if we wish to do it empirically.
(b) Declare any timestamp that is adjacent to a negative inter-arrival time to be unreliable. That is, if $X_{i+1} - X_i < 0$, both X_i and X_{i+1} are unreliable.
(c) Sort the timestamps and consider any timestamp that changes position when sorted to be unreliable. This addresses the case of a single wrong timestamp the same way as in (3b) and provides a consistent approach for more complicated scenarios.
(d) Find the longest increasing subsequence of the data (if $X_1, X_2, \ldots$ is treated as a sequence). Such a subsequence might not be unique. If there is more than one longest increasing subsequence, then take the intersection of all longest increasing subsequences. Any timestamps not in this intersection are considered unreliable.

Once the unreliable timestamps are determined, resample them between the adjacent reliable timestamps.

To see why (3d) is reasonable, consider the case of having two adjacent timestamps a long way out of order, see for instance Figure 3: $X_1 < X_2 < X_5 < X_6 < X_7 < X_8 < X_3 < X_4$. This is a scenario that we have observed in the blockchain. Here it is clear that X_3 and X_4 should be the timestamps under suspicion, but (3b) will replace X_4 and X_5, and (3c) will replace all of $X_3, \ldots, X_8$, whereas (3d) replaces exactly X_3 and X_4. The algorithm for computing this is available at [1].

These are the steps we followed to clean the block arrival time data:

- Disregarded all data prior to the 30th of December 2009.
- Performed (3d) for the rest of the data.

We refer to the data after these two steps as LR (later, resampled). Of the 464,372 blocks after the 30th of December 2009, 441,225 or 95% of them are in every longest increasing subsequence. The remaining 5% were resampled. While the longest increasing subsequences span the whole length of the blockchain, the errors that are corrected by taking the intersection of all longest increasing subsequences are all of a local nature. Most of those timestamps that were resampled were in consecutive groups of 1, 2 or 3. Two consecutive timestamps being resampled was the most common: this occurs when one timestamp is incorrect but with a small enough error to be only one place out of order. The other timestamp that it is transposed with will also be considered unreliable and resampled. An isolated timestamp being resampled is caused by that timestamp having a particularly large error. If a timestamp has a large error it can have two or more timestamps between

its observed value and its correct value. In this case, if none of the other nearby timestamps are out of order the incorrect timestamp will be the only one resampled.

We provide this data (along with annotation as to which points have been resampled) at [1]. This sequence could serve as a reference version of cleaned blockchain timestamp data for other research on this topic.

Acknowledgments

The work of A.E. Krzesinski is supported by Telkom SA Limited.

The work of P.G. Taylor and R. Bowden is supported by the Australian Research Council Laureate Fellowship FL130100039 and the ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

The work of H.P. Keeler is supported by a Discovery Early Career Researcher Award, DE180100463.

References

[1] Rhys Bowden. Blocktimes. https://github.com/rhysbowden/blocktimes, December 2019.

[2] Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100(469):36–50, 2005.

[3] Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the Lambert W function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

[4] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[5] Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes, vol. 1.* Springer, New York, 2003.

[6] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. IEEE, 2013.

[7] Tim Dodd. University of Melbourne first in Australia to use blockchain for student records. *Australian Financial Review*, April 30, 2017, 2017.

[8] Vassiliy A Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[9] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[10] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[11] Pierre-Olivier Goffard. Fraud risk assessment within blockchain transactions. *To appear in Advances in Applied Probability*, 2019.

[12] Song-Hee Kim and Ward Whitt. Choosing arrival process models for service systems: Tests of a nonhomogeneous Poisson process. *Naval Research Logistics (NRL)*, 61(1):66–90, 2014.

[13] John Frank Charles Kingman. *Poisson processes*, volume 3. Oxford University Press, 1992.

[14] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous*

Agents and Multiagent Systems, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[15] PAW Lewis and GS Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Res. Logistics Quart*, 26:403–413, 1979.

[16] Hubert W Lilliefors. On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown. *Journal of the American Statistical Association*, 64(325):387–389, 1969.

[17] Andrew Miller and Joseph J LaViola Jr. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. *Available on line: http://nakamotoinstitute. org/research/anonymous-byzantine-consensus*, 2014.

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

[19] Meni Rosenfeld. Analysis of Bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[20] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[21] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[22] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. Chapman and Hall/CRC, 2018.

[23] Siamak Solat and Maria Potop-Butucaru. ZeroBlock: Preventing Selfish Mining in Bitcoin. *arXiv preprint arXiv:1605.02435*, 2016.