Churn in the Bitcoin Network: Characterization and Impact

Muhammad Anas Imtiaz
ECE Department
Boston University
Boston, MA 02215, USA
maimtiaz@bu.edu

David Starobinski
ECE Department
Boston University
Boston, MA 02215, USA
staro@bu.edu

Ari Trachtenberg
ECE Department
Boston University
Boston, MA 02215, USA
trachen@bu.edu

Nabeel Younis
ECE Department
Boston University
Boston, MA 02215, USA
nyounis@bu.edu

***Abstract*—Efficient and reliable propagation of blocks is vital for ensuring the scalability of the Bitcoin network. As a result, several schemes have been proposed over the last few years to speed up the block propagation, most notably the compact block protocol (BIP 152). Despite this, we show experimental evidence that (i) the vast majority (97%) of Bitcoin nodes exhibit intermittent network connectivity (churn), and (ii) this churn results in significant numbers of unsuccessful compact blocks, roughly twice the figure for continuously connected nodes. Specifically, we conduct experiments on the Bitcoin network that show that churn results in a 135% average increase in block propagation time (i.e., 336.57 ms vs 142.62 ms), and can lead to as high as an 800-fold increase in the worst case. To effect our analysis, we develop a statistical model for churn based on empirical network data, and use this model to actuate the live test nodes on the Bitcoin network. The performance of the system is measured by means of a novel framework that we develop for logging the internal behavior of a Bitcoin node and share for public use.**

***Index Terms*—Bitcoin, blockchain, churn, propagation delay, distribution fitting**

I. Introduction

The Bitcoin cryptocurrency, originally introduced by Satoshi Nakamoto in 2008 [1] as a peer-to-peer electronic payment system, is currently used for buying and selling a wide variety of goods in different markets across the globe. Combined with hundreds of derivative cryptocurrencies, the total market capitalization of these electronic payment systems is roughly 200 billion dollars [2].

Bitcoin's public ledger system, also known as *blockchain*, records all transactions that take place in the Bitcoin network [3]. Each new transaction is broadcast over the network, and thereafter recorded by every node in its local memory pool (known as a *mempool*) for subsequent consensus-based validation. By design, a new block containing transactions is created (by a *mining* node) and propagated over the network's nodes roughly once every ten minutes [4].

A key challenge in this context lies in reducing the propagation time of blocks. The consequences of slower block propagation times include an increase of *forks*, wherein several blocks are mined independently and distributed before the network nodes accept one of the blocks as the head of the blockchain while the other blocks become orphan. This issue leads to periods of ambiguity, during which different nodes in the network have different views of the blockchain. An adversary may leverage such ambiguities for certain attacks, such as a double-spending attack [3]. Orphan blocks also lead to a waste of computational resources for nodes that have mined them and nodes that have mined on top of them.

To address this challenge, the *compact block relay* protocol [5], described in greater detail in Section II-A, has been proposed and is currently implemented on the standard Bitcoin Core reference implementation. This protocol aims to decrease the propagation time to the broader network by reducing the amount of data propagated between nodes.

Yet, like any peer-to-peer network, it is also important that the Bitcoin network be able to support a high rate of *churn* [6], that being the rate at which nodes independently enter and leave the network. In fact, Satoshi's white paper envisions that Bitcoin nodes "can leave and rejoin the network at will" [1]. In other words, the network should be able to quickly propagate blocks to all current nodes, even when some of these nodes frequently enter and leave the network.

Churn in different peer-to-peer networks has been widely studied, characterized and modeled [6]–[12], though it has received little attention in relation to the Bitcoin network. While some previous works on Bitcoin do consider churn in their models [13]–[15], they do not seek a full characterization and evaluation of its impact (cf. discussion in Section II-B). Undeniably, questions remain about the extent of churn in the Bitcoin network and its effect on block propagation.

Our first key contribution in this work is to systematically characterize churn in the Bitcoin network. Our characterization is based on measurements of the duration of time that nodes in the Bitcoin network are continuously reachable, i.e., *up session lengths*, and continuously unreachable, i.e., *down session lengths*. Our data show that out of more than $40{,}000$ unique nodes on the network, over 97% leave and rejoin the network multiple times over a time span of about two months. In fact, the daily average churn rate in the Bitcoin network (i.e., the rate of oscillations between up and down sessions) exceeds 4 times per node. Our statistical analysis in Section III-B2 points out that, among several possible distributions, the *log-logistic*

978-1-7281-1328-9/19/$31.00 ©2019 IEEE

distribution and the *Weibull distribution* are the best fits for up session lengths and down session lengths, respectively.

Our second key contribution is an experimental evaluation of the behavior of the compact block protocol under realistic node churning behavior. We leverage our statistical characterization of churn to generate samples from the above distributions. We then use these samples to emulate churn on nodes of ours running in the *live Bitcoin network*, taking these nodes off the network and bringing them back on according to the sampled session lengths over a two week period. Our analysis, compared against a control group of nodes that are continuously connected to the network, shows that the performance of the compact block protocol significantly degrades in the presence of churn. Specifically, the churning nodes see roughly twice as many incomplete blocks as the control nodes, that is, on average 26.78% versus 13.16% of incomplete blocks (i.e., unsuccessful compact blocks). This is due to an absence of about 312 transactions on average for the churning nodes, versus slightly less than 3 transactions for the control nodes. The end result is that, on average, churning nodes require over twice as much time to propagate a block than their continuously connected counterparts (i.e., 336.57 ms vs. 142.62 ms, which represents an increase of about 135% in the delay). When comparing incomplete blocks from churning nodes to complete blocks from control nodes, the former have an average delay about 17 times larger than the latter (the worst case is about an 800-fold difference). These results show that churn has indeed a major impact on block propagation in Bitcoin.

The rest of this paper is organized as follows. In Section II, we cover background and related work. In Section III, we describe our methodology for obtaining and processing data on churn in the Bitcoin network, and conduct a statistical analysis of the data. In Section IV, we detail the experimental setup for evaluating the impact of churn on block propagation, and present the results. Section V concludes the paper and discusses potential areas for future work.

II. Background and Related Work

In this section, we provide relevant background material on the Bitcoin network followed by a discussion of related work.

A. Bitcoin Background

1) Blocks and the mempool: Bitcoin's primary record-keeping mechanism is the *block*. It is a data structure that contains metadata about the block's position in the blockchain together with a number of associated transactions (typically a couple thousands [16]). A block is generated roughly every ten minutes through the *mining* process, and, once generated, the block and its transactions become a part of the Bitcoin blockchain. There is a probability that different nodes will incorporate different blocks in their version of the blockchain (a process known as a *fork*). These differences are reconciled over time in a competitive process.

In the interim time between when a transaction is announced and when it is included in a block, transactions are stored locally in the *mempool*. The mempool is a constantly changing data set that stores all the unconfirmed transactions waiting to be included in future blocks. It typically contains anywhere between 10^4 to 10^5 transactions, depending on network activity. Currently the mempool experiences as low as 1 and as high as 17 insertions per second [17]; the arrival of a new block also instigates many deletions from the mempool, between $1{,}300$ to $2{,}400$ transactions [18] per block.

2) Block propagation: Block Propagation is the process of communicating a newly mined block to the network. It is the backbone of Bitcoin's ability to maintain consensus on the current balances of address (wallets). When a new block is discovered, each Bitcoin node advertises the block to all of its neighboring peers.

There are currently two main block protocols in Bitcoin: the original protocol developed for the first implementation of Bitcoin and the *Compact Block Relay* Protocol (BIP 152) [5]. The original protocol is adequate for block propagation but may require significant network resources, typically close to 1 MB per block [19]. Since neighboring peers in the Bitcoin networks can be geographically distant, this approach is susceptible to large delays [20].

The *compact block relay* was developed in an effort to reduce the total bandwidth required for block propagation. As the name implies, the compact block is able to communicate all the necessary data for a node to reconstruct and validate one standard block. The compact block contains the same metadata as the normal block, but instead of sending a full copy of each transaction, it sends only a hash of the transaction. Depending on the number of inputs and outputs, a transaction is between 500 and 800 bytes [21], whereas the hashes used for the compact block are only 6 bytes per transaction [5], a significant bandwidth saving that relies on the assumption that the receiver already has the relevant transactions in its mempool and just needs to know in which blocks they belong. This trade-off makes a compact block much smaller in size than the original block at the cost of potentially needing extra round-trip communications for transactions whose hashes the receiver does not recognize (using the `getblocktxn/blocktxn` messages [22]).

If a receiver's mempool contains all the transactions whose hashes are contained in a compact block that it received, then it will be able to successfully reconstruct the original block. However, if not all transactions are already in the node's mempool then it will fail to reconstruct the block. When the compact block protocol fails, the extra round trips slow down block propagation and increases the risks of a fork in the blockchain. Fig. 1 illustrates the process.

B. Related Work

Stutzbach and Rejaie [6] study churn in several peer-to-peer networks, specifically Gnutella, BitTorrent, and Kad. By inserting crawlers into each network, they characterize various metrics, such as peer inter-arrival time, session lengths, peer up time, peer down time etc., and fit distributions to the respective metrics. The authors state that "one of the most

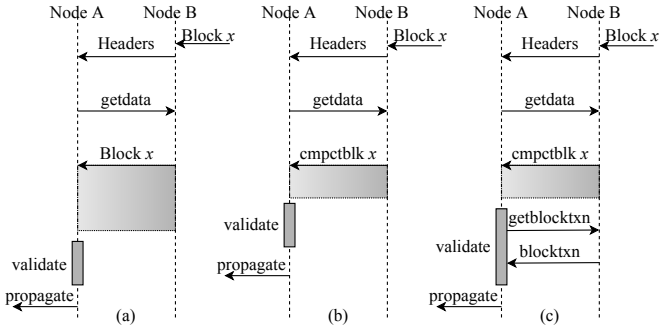

Fig. 1: Block propagation: (a) an *Original Satoshi block*, (b) a *Successful compact block*, (c) and an *Unsuccessful compact block*. In the last case, additional communications recover missing transactions from peers.

basic properties of churn is the session length distribution, which captures how long peers remain in the system each time they appear". Our work characterizes the statistics of session lengths and churn in the Bitcoin network, which to our knowledge have not been studied so far. Furthermore, our work is not limited to statistical characterization of churn, but also evaluates the impact of churn on the behavior of the Bitcoin network with respect to the efficacy of the compact block protocol.

Apostolaki et al. [14] simulate partitioning attacks. The authors incorporate churn in their simulations and assume that session lengths follow exponential distributions. We show in our work that, on an aggregate level, session lengths are better modeled by heavy-tailed distributions.

Decker and Wattenhofer [3] measure the time it takes for a block to propagate in the Bitcoin network. They show that delay in propagation of blocks in the network results in forks in the Bitcoin blockchain. Since only one branch of the fork becomes part of the blockchain, nodes who create blocks in the other branch(es) essentially waste their power. Forks in the blockchain also lead to a phenomenon called *information eclipsing* which an adversary can leverage to perform a *double spending* attack. However, the work in [3] was published before the compact block protocol was implemented, i.e., each block contained full transactions and no reconstruction was needed. Therefore, their work does not capture the current behavior of block propagation, including additional delay incurred due to missing transactions in a compact block received by a peer.

Neudecker et al. [13] study churn in the Bitcoin network from an attacker's perspective. They vary the session length of an attacking node in the Bitcoin network and through simulations, show that a network partitioning "attack is sensitive to churn of the attacking node". However, they do not characterize churn in the network, and thus, it is unclear what is the basis for the parameters used in the simulations.

Karame et al. [15] study the security of using Bitcoin in fast payments, such as paying for a meal at a fast-food restaurant. They theorize that because of churn in the Bitcoin network, the connectivity of a victim node with the rest of the network varies with time. This gives an adversarial node considerable opportunities to connect with a victim node and perform a double spend attack. However, the authors neither characterize churn nor take it into account when performing analysis, measurements and experiments.

Ozisik et al. [23] propose the Graphene protocol, which couples an Invertible Bloom Lookup Table (IBLT) [24] with a Bloom filter in order to send transactions in a package smaller than a compact block. According to the authors in [23], the size of a Graphene block can be a fifth of the size of a compact block, and they provide simulations (but not an actual implementation) demonstrating their system. Yet, similar to the compact block protocol, the Graphene protocol also assumes a large degree of synchronization between mempools of sending and receiving peers. In case of missing transactions, the receiving peer requests larger IBLTs from the sending peer, thus significantly adding to the propagation delay.

III. Churn Characterization

Nodes on the Bitcoin network may leave and rejoin the network independently. Characterizing churn requires observation of the node activity on the network. In this section, we first detail our methodology to obtain and process data on churn. Next, we present our statistical analysis. In Section IV, we leverage this characterization to run experiments of the compact block protocol with churning nodes.

A. Obtaining and processing data

Bitnodes [25] continuously crawls the Bitcoin network and provides a list of all up nodes with an approximate 5 minute resolution. Each network snapshot is available for roughly 60 days [26]. The website provides a rich API interface that can be used to download each snapshot as a JavaScript Object Notation (JSON) file. Each JSON file contains information, such as IP address, version of Bitcoin running, geographic location etc., of the nodes on the network that are up. Listing 1 shows an example from a JSON file representing a snapshot taken by the crawler at the UNIX timestamp 1526742217.

```
"220.75.229.130:3927": [
    70015,                  Protocol Version
    "/Satoshi:0.13.2/",     User Agent
    1526337217,             Connected Since
    13,                     Services
    165277,                 Height
    "220.75.229.130",       Hostname
    "Seoul",                City
    "KR",                   Country Code
    37.5985,                Latitude
    126.9783,               Longitude
    "Asia/Seoul",           Timezone
    "AS4766",               ASN
    "Korea Telecom"         Organization Name
]
```

Listing 1: Part of a JSON file transmitted to a Bitcoin node.

We download all available JSON files from Saturday, May 19, 2018 11:03:37 AM EST (UNIX timestamp: 1526742217) to Tuesday, July 17, 2018 04:06:14 PM EST (UNIX timestamp: 1531857974) with a total of 14,674 snapshots ordered according to unique UNIX timestamps.

Next, we parse each JSON file and generate a data set of all IP addresses that appear at any point on the Bitcoin network during the aforementioned time period. We find a total of 47,702 distinct IP addresses.

Given the list of IP addresses, we run a script that looks for each IP address through each consecutive network snapshot. If an IP address is found in two consecutive network snapshots, we can conclude that the IP address was up, and thus online, for 10 minutes (recall Bitnodes' 5 minute resolution). Similarly, if the IP address is found in only one of the two consecutive network snapshots, we can infer that the node either left or rejoined the network. Finally, if the IP address is not found in any of the two consecutive network snapshots, we deduce that the IP address was down, and thus offline, for 10 minutes. This allows us to record the behavior of a node, i.e., the duration of time it is on and off the Bitcoin network over the 14,674 snapshots.

B. Statistical Analysis

1) Churn Rate: We discover that out of 47,702 distinct IP addresses observed on the network during the aforementioned time period, only 1,154 are always up, i.e., only about 2.42% of the nodes are online at all times. Nodes corresponding to the remaining IP addresses contribute to churn in the Bitcoin network.

Next, we evaluate the churn rate, namely the rate at which nodes oscillate between up and down sessions. For this purpose, let T be a random variable corresponding to the sum of the duration of an up session and its subsequent down session. We then define the *churn rate* as $R = 1/T$. Fig. 2 shows the CCDF of the churn rate R as measured across all the observed Bitcoin nodes. With probability greater than 45%, R exceeds one churn per node per day. In fact, there is a 10% probability that $R \geq 9$ churns per node per day. The average churn rate per node is $\bar{R} = 4.16$ per day.

Note that nodes that are always up do not contribute to churn in the Bitcoin network. Therefore, we filter out data related to these nodes from our data sets on the session lengths of a node's up and down time on the network. In addition, we filter out the first and the last session, whether up or down, of each node. This is because we do not know how long a node was up or down before we started and after we finished our measurement.

2) Statistical fitting of session lengths: Prior work [27]–[30] showed that session lengths exhibit a behavior similar to a heavy-tailed distribution. Therefore, in our statistical fitting, we focus on fitting heavy-tailed distributions to the data, namely the *generalized Pareto distribution*, *the log-normal distribution*, the *Weibull distribution* [31] and the *log-logistic distribution*. Nolan [32] shows that maximum likelihood estimation (MLE) of heavy-tailed distribution parameters

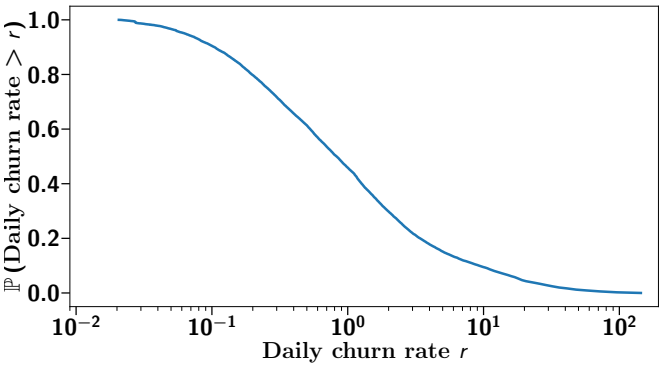

Fig. 2: Daily churn rate on the Bitcoin network.

is feasible. Hence, we use MATLAB's distribution fitting capabilities [33] to fit distributions based on the MLE criterion. Finally, we also consider the exponential distribution, as a basis for comparison.

Up sessions: We first fit a distribution to the data representing up session lengths. Our fitting applies to the first 25,000 minutes, which roughly translates to 2.5 weeks. We use the following criteria to determine the goodness-of-fit of the various distributions to the actual data: 1) The R-squared $\left(\mathrm{R}^2\right)$, which is given by

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})}$$

where y is the actual value, $\hat{y}$ is the calculated value, and $\bar{y}$ is the mean of y [34]. An R^2 value of 1 would indicate a perfect model [35]. 2) The root mean squared error (RMSE), which is given by

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$$

where y is the actual value and $\hat{y}$ is the calculated value [34]. An RMSE score of 0 indicates that all of the calculate values lie on the line formed by the actual values [36]. 3) Visual inspection of the data.

We set the parameter values generated by MATLAB as a base and perform an exhaustive search within $\pm 10\%$ of the base parameters. The final results for each distribution are the highest R^2 and lowest RMSE in that range.

The results can be seen in Fig. 3. The R^2 and RMSE scores for the fits are detailed in TABLE I. A key observation is that the *exponential distribution* is a very poor fit for the session lengths. While the *log-normal distribution* performs the best in terms of R^2 and RMSE scores, Fig. 3 indicates that the *log-logistic distribution* fits best the CDF of the empirical data, at least in the initial portion where most of the data lies. Therefore, judging from the combination of Fig. 3 and TABLE I, we conclude that the *log-logistic distribution*, given by

$$F_{(\alpha,\beta)}(x) = \frac{1}{1 + (x/\alpha)^{-\beta}}, \tag{1}$$

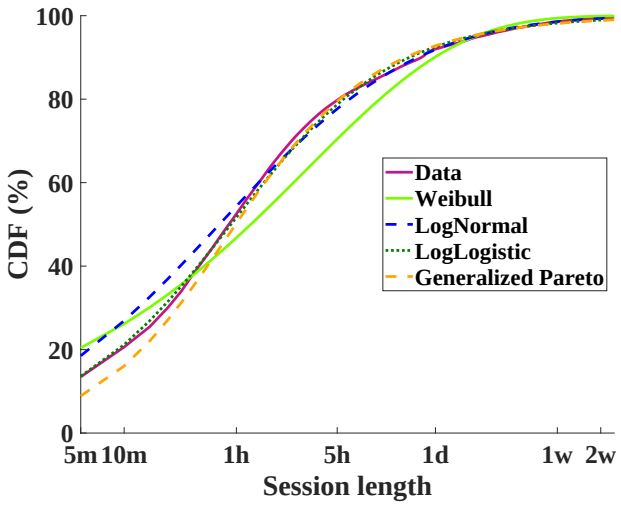

Fig. 3: Distribution fits for "up session" lengths.

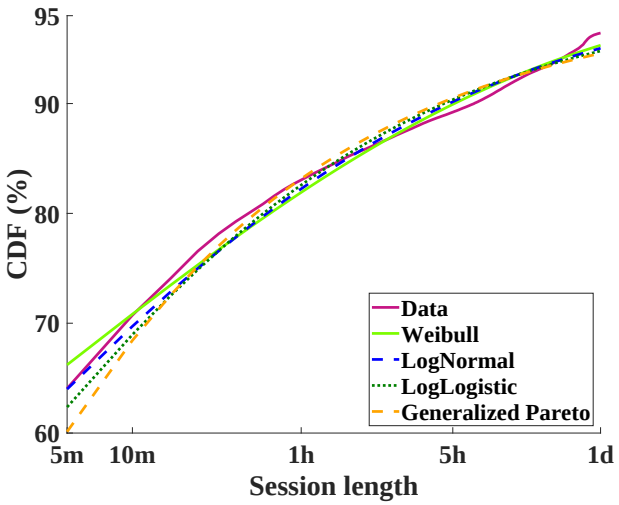

Fig. 4: Distribution fits for "down session" lengths.

Distribution	R^2	RMSE
Weibull	0.9002	2.60e−03
Log-normal	0.9939	1.29e−06
Log-logistic	0.9907	1.80e−03
Generalized Pareto	0.9856	2.20e−03
Exponential	0.4904	21.70e−03

TABLE I: R^2 and RMSE scores of distribution fits for "up session" lengths.

Distribution	R^2	RMSE
Weibull	0.9777	3.28e−04
Log-normal	0.9694	5.36e−04
Log-logistic	0.9575	7.72e−04
Generalized Pareto	0.9429	9.55e−04
Exponential	0	1

TABLE II: R^2 and RMSE scores of distribution fits for "down session" lengths.

where $\alpha > 0$ is the scale parameter, and $\beta > 0$ is the shape parameter, is the best fit for the up sessions. The parameters for the *log-logistic distribution* fit in Fig. 3 are $\alpha = 11.000$ and $\beta = 0.771$.

Down sessions: Next, we fit distributions to the data representing down session lengths. We employ an approach similar to that in the previous section. We focus on performing a statistical fitting for sessions that are down for up to one day (which represents over 93% of the cases). Note that the mempool of a node that is continuously off the network for a duration exceeding one day will largely be out of synchronization with the rest of the network.

The fitting results are shown in Fig. 4. The corresponding R^2 and RMSE scores are listed in TABLE II. Notice that the *exponential distribution* is a very poor fit and is never able to achieve an R^2 value above 0. Observing the combination of Fig. 4 and TABLE II, we conclude that in this case the *Weibull distribution*, given by

$$F_{(\lambda,k)}(x) = \begin{cases} 1 - e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0, \end{cases} \quad (2)$$

where $\lambda > 0$ is the scale parameter, and $k > 0$ is the shape parameter, is the best fit for the down sessions. The parameters for the *Weibull distribution* fit in Fig. 4 are $\lambda = 0.640$ and $k = 0.183$.

3) Geographical Analysis: Fig. 5 shows the geographical location of the 47,702 individual nodes discovered in the Bitcoin network during the time period mentioned in Section III-A. Nodes that are always up during this time period are marked white to make them distinguishable from the remaining nodes that contribute to churn in the network. We observe that the majority of the Bitcoin nodes are located in the North America and Europe. South America, North Asia, the Far East and Oceania show a moderate presence while Africa and Central and South Asia show a very little presence of Bitcoin nodes. We note that the nodes that are always connected are not co-located in one region but rather spread out over the entire world map.

TABLE III indicates that North America and Europe are the continents with the highest percentage of nodes that are always up. On the other hand, Africa has a very small percentage of nodes that are always up. These results may be due to the more intermittent nature of Internet access in that continent.

IV. Experimental Analysis of Compact Block Performance with Churn

In this section, we evaluate the performance of block propagation, and especially the compact block protocol, in the presence of churning nodes, to realistically reflect the behavior of the Bitcoin P2P network. The section details the mechanism that we developed to log events on the Bitcoin network, the experimental setup, the method for emulating churning nodes based on the distribution fits performed in Section III-B2, and finally, the results obtained.

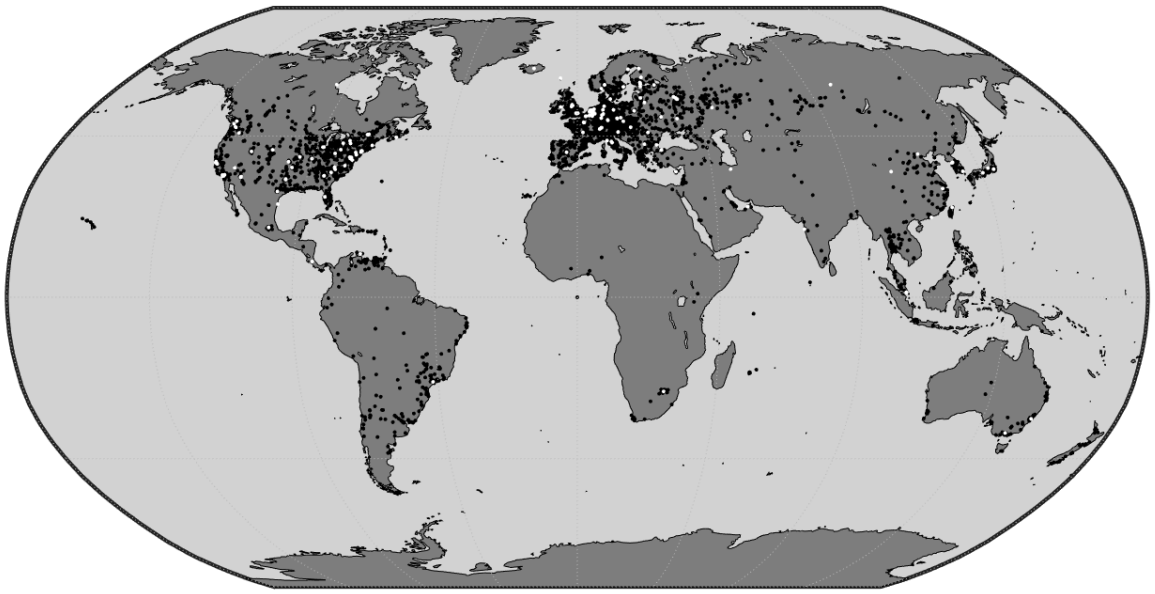

Fig. 5: Geographic location of individual nodes on the Bitcoin network. Nodes that are always up are marked white. Remaining (black) nodes contribute to churn in the network.

Continent	Percentage
Africa	0.051
Asia	1.138
Europe	3.239
Oceania	0.692
North America	3.414
South America	0.289

TABLE III: Percentage of continuously connected nodes in each continent.

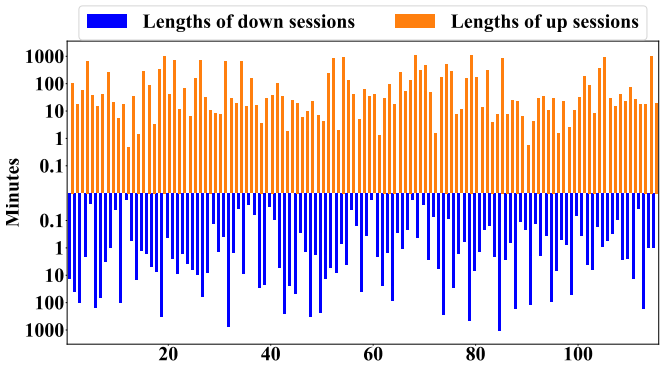

Fig. 6: Sampled up and down session lengths.

A. Data Collection Mechanism

To aid in understanding Bitcoin Core's behavior, we have developed a new log-to-file system that produces human-friendly, easy-to-read text files. This logging system is open-source and we have made it available to the research community (`[37]/src/logFile.*`). This new logging mechanism allows one to isolate specific behaviors through select calls anywhere within the Bitcoin core's source code, most notably information about different protocols such as the compact block. The logging system writes core data to a log file, and also can record various events and the information associated with those events. For instance, when a compact block arrives, the system logs this event and saves the transaction hashes included in the compact block in a separate file with a unique identifier tying it to a log entry. We have used this system as our primary data collection mechanism for all of our experiments.

B. Experimental setup

The aim of the experiment is to determine the efficiency of the compact block protocol in the presence of churn. We achieve this by running five nodes in the Bitcoin network. The nodes are Dell Inspiron 3670 desktops, each equipped with an 8^{th} Generation Intel® Core i5$-$8400 processor (9 mB cache, up to 4.0 GHz), 1TB HDD and 16GB RAM. The nodes are each running the Linux Ubuntu 18.04.1 LTS distribution.

We run experiments over a period of two weeks. Three nodes use sampled session lengths to emulate churn on the network. Specifically, we generate samples of the best fit distributions given by Eq. (1) and Eq. (2), such that the aggregate sum of the up and down sessions is at least two weeks for each node. We limit both the up and down session lengths from a minimum of 1 second to a maximum of 1 day. The remaining two nodes act as *control nodes* that are continuously connected to the network. Fig. 6 shows the sampled up and down session lengths used in the experiments. In order to avoid any bias, we use the Bitcoin RPC API `setban` [38], [39] to ensure that the five nodes are not connected to each other as peers in the Bitcoin network. This way, our nodes do not directly influence each other. Our experiment runs without interruption from Friday, November

Nodes	Blocks Received	Successful Compact Blocks (%)	Unsuccessful Compact Blocks (%)
S_1	1826	87.84	12.16
S_2	1815	85.84	14.16
C_1	1430	72.03	27.97
C_2	1455	72.44	27.56
C_3	1454	75.17	24.83

TABLE IV: Block reception statistics for control nodes S_1 and S_2 and churning nodes C_1, C_2, and C_3.

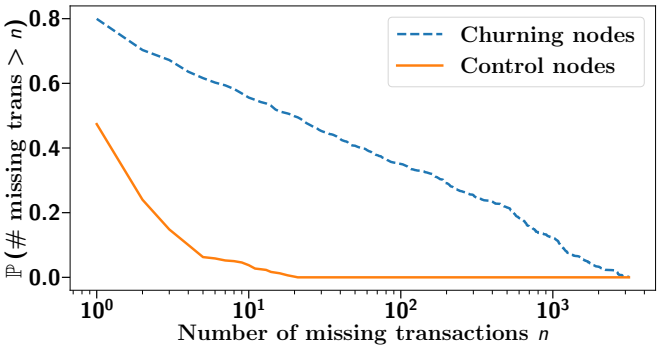

Fig. 7: CCDF of number of missing transactions in churning and control nodes.

09, 2018 18:18:41 to Sunday, November 25, 2018 23:30:24.

C. Experimental results

1) Statistics on compact blocks: We compare the number of compact blocks that the churning nodes (denoted by C_1, C_2, C_3) and the (stable) control nodes (denoted by S_1, S_2) fail to reconstruct. TABLE IV shows the results. The churning nodes are unable to reconstruct roughly twice as many blocks as the control nodes (i.e., 26.78% vs. 13.16%). Indeed, of the blocks they receive, the control nodes are able to reconstruct successfully on average 1,581.00 blocks out of 1,820.50 blocks (i.e., 86.84% of the blocks), while the churning nodes are able to reconstruct successfully on average only 1,059.00 blocks out of 1,446.30 blocks (i.e., 73.22% of the blocks). The results are quite consistent across both the control and churning nodes.

2) Statistics on missing transactions: Churning nodes are generally missing far more transactions in blocks they are unable to reconstruct than the control nodes. We find that on average a churning node misses 312.02 transactions from a block with a standard deviation of 591.70 transactions, whereas a control node misses on average 2.91 transactions with a standard deviation of 7.97 transactions. Fig. 7 shows the CCDF of the number of missing transactions. From the figure, we observe that churning nodes may be missing up to thousands of transactions, while control nodes may miss at most a few hundred transactions. Indeed, in 12% of the cases, churning nodes may miss between 1,000 transactions and a maximum of 3,167 transactions, while control nodes have a probability of 12% to miss between 3 and a maximum of 145 transactions. Therefore, our results clearly indicate that churning nodes need to request a high number of transactions from their peers to successfully reconstruct a block.

Nodes	Mean (ms)	Std Dev (ms)
S_1	120.62	686.05
S_2	164.75	789.59
C_1	299.99	1445.80
C_2	288.37	1279.13
C_3	450.76	2631.69

TABLE V: Propagation delay statistics of received blocks for control nodes S_1 and S_2 and churning nodes C_1, C_2, and C_3.

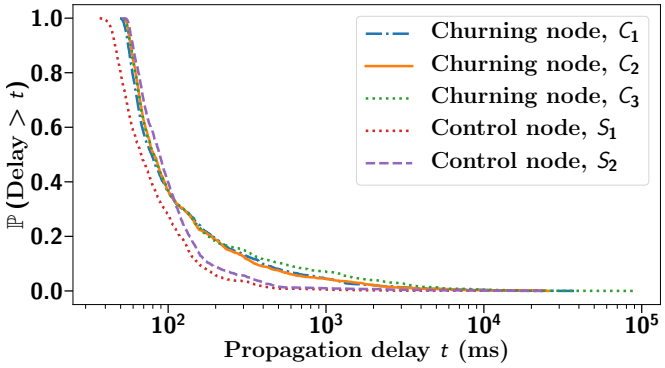

Fig. 8: Propagation delay across all blocks for both churning and control nodes.

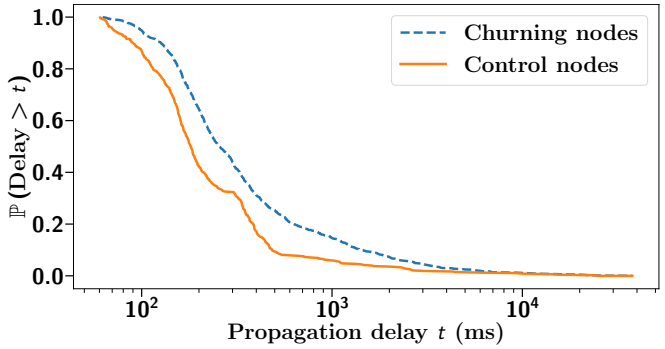

Fig. 9: Propagation delay for blocks that are unsuccessful in both churning and control nodes.

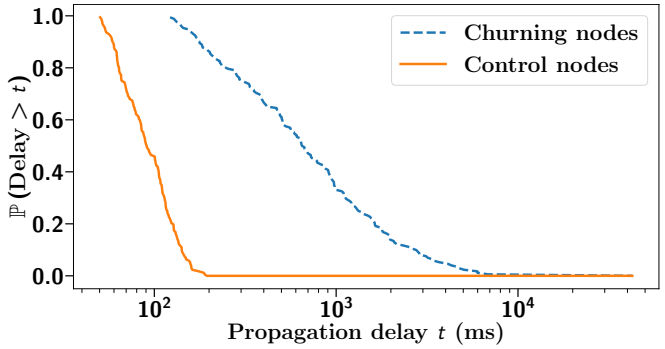

Fig. 10: Propagation delay for blocks that are successful in control nodes but unsuccessful in churning nodes.

3) Statistics on propagation delay: TABLE V shows the propagation delay statistics for each of the churning and control nodes, across all received blocks. Blocks received by the control nodes experience an average propagation delay of 142.62 ms whereas blocks received by the churning nodes experience an average propagation delay of 336.57 ms (i.e., an increase of about 135%). Fig. 8 shows that, for each churning node, the CCDF of the propagation delay for each churning node is almost always above the CCDF of the propagation delay for each control node.

Fig. 9 shows the propagation delays of all unsuccessful blocks in both churning and control nodes. We see that overall, the propagation delay of such blocks in churning nodes is higher. In fact, unsuccessful blocks in control nodes have a propagation delay of 525.31 ms with a standard deviation of $1{,}995.18$ ms, and the unsuccessful blocks in the churning nodes have a propagation delay of 796.45 ms with a standard deviation of $2{,}336.13$ ms.

In Fig. 10, we show the CCDF of the propagation delay for the blocks that were successful in all control nodes but unsuccessful in all churning nodes. When a block cannot be reconstructed successfully, a recipient has to request missing transactions from its peers, thus increasing the propagation delay. Successful blocks in the control nodes have a propagation delay of 96.98 ms on average with a standard deviation of 32.78 ms, whereas unsuccessful blocks in churning nodes experience a propagation delay of $1{,}331.43$ ms on average with a standard deviation of $3{,}373.47$ ms. In 33% of the cases, the propagation delay of blocks received by churning nodes exceeds 1000 ms up to a maximum of $42{,}539.53$ ms, while the propagation delay in 33% of blocks received by the control nodes exceeds 110 ms to a maximum of 194.13 ms.

V. Conclusion

In this paper, we identified and empirically demonstrated the heretofore undocumented effect of churn on the Bitcoin network. We performed a thorough characterization of churn, including the daily churn rate and statistical fitting of the distributions of the lengths of up and down sessions. This statistical characterization should prove useful to other researchers, for the purpose of analyzing, simulating, and emulating the behavior of the Bitcoin network.

We used the statistical characterization to evaluate the impact of churn on the propagation delay of blocks in the live Bitcoin network. In the process of this research, we developed a logging mechanism for tracing events in Bitcoin nodes, which we have released for public use [37]. Our experiments showed that churn produces a marked degradation in the performance of the delay-optimized compact block protocol. This is because unsuccessful compact blocks are much more prevalent in churning nodes, and the associated incomplete blocks often miss a large number of transactions (312.02 on average). As a result, the propagation delay of blocks processed by churning nodes is substantially larger on average than that of nodes that are always connected. In fact, occurrences of propagation delays that exceed one second are common. Our measurements show that about 14.51% of the blocks processed by the churning nodes have a propagation delay exceeding one second, compared to 5.72% of the blocks processed by the control nodes. Note that this corresponds to the delay over a single hop on the Bitcoin network, and hence the end-to-end delay would be even larger.

As an outcome of this work, it is evident that there should be significant benefit in implementing efficient synchronization of the mempools of Bitcoin nodes, thus keeping them up-to-date with transactions that they might have missed while being disconnected. We believe that it should be possible to engineer transaction synchronization based on prioritization metrics by utilizing difference-finding algorithms and reconciliation-optimized data-structures from the synchronization literature. Potential candidates include IBLT [24], due to its high tolerance for differences, and CPISync [40]–[42], due to its near-optimal communication complexity.

Acknowledgment

This research was supported in part by NSF under grant CCF-1563753.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 03 2009.
[2] "Cryptocurrency market capitalizations." `https://coinmarketcap.com/all/views/all/`, 2018. Online; Accessed: May 24, 2018.
[3] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pp. 1–10, IEEE, 2013.
[4] B. Wiki, "Block." `https://en.bitcoin.it/wiki/Block`, 2016. Online; Accessed: November 17, 2018.
[5] M. Corallo, "Compact block relay." `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`, 2016.
[6] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 189–202, ACM, 2006.
[7] D. Stutzbach and R. Rejaie, "Towards a better understanding of churn in peer-to-peer networks," *Univ. of Oregon, Tech. Rep*, 2004.
[8] O. Herrera and T. Znati, "Modeling churn in p2p networks," in *Simulation Symposium, 2007. ANSS'07. 40th Annual*, pp. 33–40, IEEE, 2007.
[9] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured p2p networks," in *icnp*, pp. 32–41, IEEE, 2006.
[10] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE transactions on multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.
[11] D. Yang, Y.-x. Zhang, H.-k. Zhang, T.-Y. Wu, and H.-C. Chao, "Multi-factors oriented study of p2p churn," *International Journal of Communication Systems*, vol. 22, no. 9, pp. 1089–1103, 2009.
[12] F. Lin, C. Chen, and H. Zhang, "Characterizing churn in gnutella network in a new aspect," in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, pp. 305–309, IEEE, 2008.
[13] T. Neudecker, P. Andelfinger, and H. Hartenstein, "A simulation model for analysis of attacks on the bitcoin peer-to-peer network," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 1327–1332, IEEE, 2015.
[14] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 375–392, IEEE, 2017.
[15] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 906–917, ACM, 2012.
[16] "Average number of transactions per block." `https://blockchain.info/charts/n-transactions-per-block?timespan=2years`, 2017. Online; Accessed: November 17, 2018.

[17] "Transaction rate." https://www.blockchain.com/charts/transactions-per-second?timespan=30days, 2017. Online; Accessed: November 17, 2018.

[18] "Average number of transactions per block." https://www.blockchain.com/en/charts/n-transactions-per-block?timespan=60days, 2017. Online; Accessed: November 17, 2018.

[19] "Block size limit controversy." https://en.bitcoin.it/wiki/Block_size_limit_controversy, 2017. Online; Accessed: November 17, 2018.

[20] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste, "Blockchain inefficiency in the bitcoin peers network," *arXiv preprint arXiv:1704.01414*, 2017.

[21] "Analysis of bitcoin transaction size trends." https://tradeblock.com/blog/analysis-of-bitcoin-transaction-size-trends, 2015. Online; Accessed: November 17, 2018.

[22] "Protocol documentation." https://en.bitcoin.it/wiki/Protocol_documentation, 2018. Online; Accessed: May 17, 2018.

[23] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. Levine, "Graphene: A new protocol for block propagation using set reconciliation," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 420–428, Springer, 2017.

[24] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pp. 792–799, IEEE, 2011.

[25] A. Yeow, "Bitnodes." https://bitnodes.earn.com. Online; Accessed: November 15, 2018.

[26] A. Yeow, "Bitnodes api v1.0." https://bitnodes.earn.com/api/. Online; Accessed: December 17, 2018.

[27] F. E. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in p2p protocols," in *Web content caching and distribution*, pp. 233–246, Springer, 2004.

[28] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 314–329, 2003.

[29] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of dht routing tables," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 99–114, USENIX Association, 2005.

[30] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 137–150, ACM, 2002.

[31] S. Foss, D. Korshunov, S. Zachary, *et al.*, *An introduction to heavy-tailed and subexponential distributions*, vol. 6. Springer, 2011.

[32] J. P. Nolan, "Maximum likelihood estimation and diagnostics for stable distributions," in *Lévy processes*, pp. 379–400, Springer, 2001.

[33] T. M. Inc., "Fit probability distribution object to data - MATLAB fitdist." https://www.mathworks.com/help/stats/fitdist.html. Online; Accessed: November 06, 2018.

[34] "Evaluating goodness of fit." https://www.mathworks.com/help/curvefit/evaluating-goodness-of-fit.html#bq_5kwr-7. Online; Accessed: November 18, 2018.

[35] "Coefficient of determination (r-squared) explained." https://towardsdatascience.com/coefficient-of-determination-r-squared-explained-db32700d924e. Online; Accessed: December 2, 2018.

[36] "Rms error." http://statweb.stanford.edu/~susan/courses/s60/split/node60.html. Online; Accessed: December 2, 2018.

[37] N. Younis, "Bitcoin." https://github.com/Nabeelperson/bitcoin, 2017.

[38] "Bitcoin developer reference." https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs, 2017. Online; Accessed: November 17, 2018.

[39] "Bitcoin core :: setban (0.16.0 rpc)." https://bitcoincore.org/en/doc/0.16.0/rpc/network/setban/. Online; Accessed: November 17, 2018.

[40] J. Jin, W. Si, D. Starobinski, and A. Trachtenberg, "Prioritized data synchronization for disruption tolerant networks," in *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pp. 1–8, IEEE, 2012.

[41] D. Starobinski, A. Trachtenberg, and S. Agarwal, "Efficient pda synchronization," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 40–51, 2003.

[42] A. Trachtenberg, D. Starobinski, and S. Agarwal, "Fast pda synchronization using characteristic polynomial interpolation," in *IEEE INFOCOM*, vol. 3, pp. 1510–1519, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2002.