The Bitcoin Peers Network

Giuseppe Pappalardo*[1,2], Guido Caldarelli[2], and Tomaso Aste[1,3]

[1]Department of Computer Science, UCL, London, UK
[2]IMT School for Advanced Studies, Lucca, IT
[3]UCL Centre for Blockchain Technologies, UCL, London, UK

June 30, 2016

Abstract

We monitor the state of the Bitcoin network investigating how fast peers share blocks and transactions among each other, which kind of services they offer and which version of the protocol they used. This is done in order to understand the mechanisms miners put in place to gain an advantage with respect to the others. The approach used makes also us able to study how data are propagated among the network, investigate forks events and better understand collaboration/competition behavior among peers.

1 Introduction

Behind Bitcoin[1], the most popular cryptographic currency, there are users distributed all over the world who, in a voluntary way or for profit, participate to the network. At the beginning the network was designed keeping in mind several rules in order to guarantee "one CPU, one vote"[1]. When Bitcoin mining profitability started to increase, other people such as developers and manufactors become attracted to the idea to push up the mining process and "win" the reward. Therefore today there is an heterogeneous environment where "peers" run their customized mining client, using Application Specific Integrated Circuit (ASIC) trying to have an advantage on the other clients. The most promising technology introduced within Bitcoin is not just the currency itself but the Blockchain, a distributed public ledger which, using the proof-of-work algorithm, can guarantee trust and reliability in an untrustworthy environment. In this work we measure how the Bitcoin network works using the data exchanged among peers. This will help to understand how nodes interact with each others, and how information propagate through the network.

1.1 Blockchain

The Blockchain is a distributed database which keeps track of all payments made using the Bitcoin currency. A payment is called "transaction" and involves one

*Correspondence author: g.pappalardo@ucl.ac.uk or more “input” Bitcoin wallets who are sending some funds to one or more “output” wallets. Transactions are included inside blocks by special peers called “Miners” which participate to the solution of a criptographic puzzle, the proof-of-work. After a miner create a block, it will try to seal it cryptographically with a hash produced from the block and a random part. If the number is by chances smaller then a threshold imposed by the proof-of-work then it is considered “valid” and it can start to be spread among the network. When a peer receives a new block, it should verify if the block is valid. In order to do that, it has to verify whether the hash of the block fulfills the proof-of-work requirements. After that, the peer has to verify also each transaction included inside the block. If the whole block and all the transactions are verified, it accepts the new block as valid and starts propagating it through the network (and if the peer is a miner, also it will start to discover the next block on top of it). If the block is not valid or at least one transaction inside the block is invalid, the block will be discarded.

1.2 Bitcoin

Bitcoin consist of all the technologies used by the criptocurrency (such as the Blockchain) and the currency itself. The owner of the funds can access or claim them through his own private key, stored in a Bitcoin wallet, which contains only the set of keys (public and private) which allow users to use funds for transactions. It means that each amount of Bitcoin is not included inside a wallet but the available funds of a wallet is recorded on the Blockchain. Since all the Bitcoin transactions are recorded on the Blockchain, it is possible to keep track of all movements among wallets.

1.3 Communication protocol

The Bitcoin Protocol [2] consists of a set of messages used by clients to enable communication among peers. There are several customization of the client but all of them have to respect the rules provided by the protocol. We developed a client implementing these messages:

- getaddr - Used to request a list of known peers from a node. This message will issue an ”addr” message as response.
- addr - Used to send a single peer to the neighbors once discovered or a list of known peers when requested.
- ping - message used to check if the connection is still active.
- pong - message used as reply to a ping message.
- inv - Inventory message sent by a client in order to let peers known about new blocks/peers/transactions.

2 Related Work

In the last few years there as been some interest in the study of the Bitcoin network with notable contributions from Decker[3], Coinscope[4] ad Bitnodes[5].

Bitnodes provides a snapshot of all reachable peers on the networks and some statistics related to the type of the client (i.e. protocol version used, last block stored and ip-geolocalization). Since all the data are provided as a list of online clients it is not possible to understand how the peers are connected to each other or how data are propagated among them. The approach used by Bitnodes to discover the peers is to send recursively to a "getaddr" message to each node reachable in order to get back part of their known nodes list. Coinscope uses the same approach of Bitnodes in order to discover clients, but they also discover edges between nodes, introducing an algorithm they named "AddressProbe". Guessing the connections among clients was made possible by the Bitcoin protocol itself (until Bitcoin Core 0.10.1[6]), indeed each node keeps a list of known nodes coupled with a timestamp information. If the node exchanges some messages with a peer it keeps its own timestamp on the database updated. If, instead, a node discovers some new nodes through another peer, it applied a 2 hours penalties on the timestamp before storing the address into its own peer database. According to that it was possible to guess the connections[7] of a peer just retrieving several time the known peers list and sorting all the records in chronological order. This kind of network topology inference makes use of behavior specific to Bitcoin Core prior to version 0.10.1. Biryukov[7] [6] showed that reconstruct the peers network could be used in order to make an attack on Bitcoin Core clients. To avoid the possibility of such attack, the software was modified and now each client does not update the timestamp of a connected client every time they send or receive data. Decker[3] did not studied the topology of the network but the data propagation rate. His idea was to establish a connection with each node and provide a timestamp for each block or transaction received. In this way, without knowing how the peers are connected, he was able to measure how long a block or a transactions takes to propagate on the network.

3 Methods

The Bitcoin network groups on average five thousands heterogeneous reachable peers with different computational capabilities and distributed around the world, connected by "random" links. In this work we use the data propagated through the peers and reconstruct all the information related to the network in order to understand:

- How the peers are characterized
- How they interact within each other

Data exchanged by peers consist of coordinating signals (i.e. announcing new blocks or transactions) and data messages (blocks, addresses and transactions). Following the path of [4] and [3], we first decided to monitor our own Bitcoin client through the Bitcoin-Core API using the "getpeerinfo" command and requesting the "getaddr" message in order to understand, whether it is still possible to exploit the "getaddr" messages to reconstruct the peers network. After the paper by Biryukov[7], the Bitcoin core client was fixed. Indeed we noticed that if two clients are connected the timestamp on their known peer database is not updated anymore. We observed for an active connection that the timestamp is updated only when the connection drops or each 24 hours (in case the connection is still alive). Other cases are the same as described on [4]. Data were collected joining on the network as a normal node and trying to establish a connection within each peer address discovered and waiting for "inv" messages for both, blocks and transactions. During the listening period of 10 days, we found more than 12 thousands unique peers, 8969 belonging to ipv4 network, 3332 belonging to ipv6 network and 124 belonging to Tor network. This amount of peers is consistent with the amount reported by Bitnodes[5]. Surprisingly we received more than 126 thousands different blocks (instead of about 1200), some of them valid but "old", where the oldest of them was included into the Blockchain more than 3 years ago.

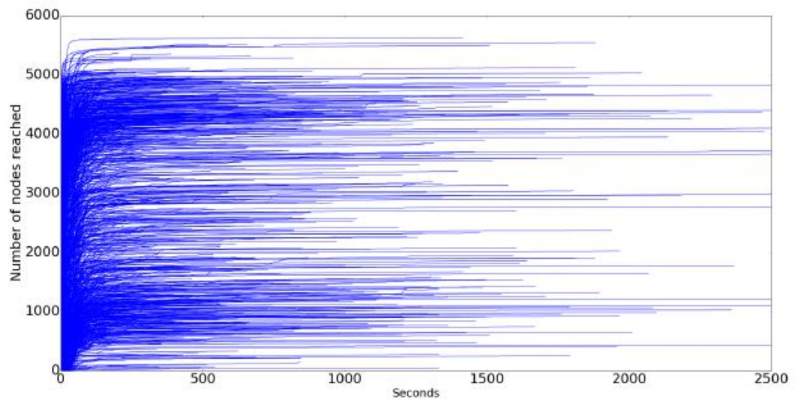

Figure 1: The figure shows how long time requires each block to reach all the nodes of the networks. Since in the network each connection can be dropped by a node without alert the peers, some blocks could be received from a partition of the total network. As end time for each wave is selected the first propagated announce of the following blocks.

4 Preliminary Results and Discussion

Here we report results concerning only the "new" blocks mined during the listening time. Our client established a connection with each reachable peer into the the network and waited for "inv" messages sent by them. The client for collecting the data was written in Go programming languages[8], in order to exploit its multithreading native management. We established only one connection with each reachable node in order to not interfere with the network behavior. This is because most clients accepts only 8 connections from peers and it is not possible anymore to measure or estimate the number of active connections held by each client. Also, each client have the possibility of dropping the connection at any time without advice the peer. This means that while a peer is transmitting a block (on average every 10 minutes), if the connection is lost before the peer is propagating the new block, the node will not send anymore the block after that the connection was recovered. We collected 592GB of data in a period of 1208

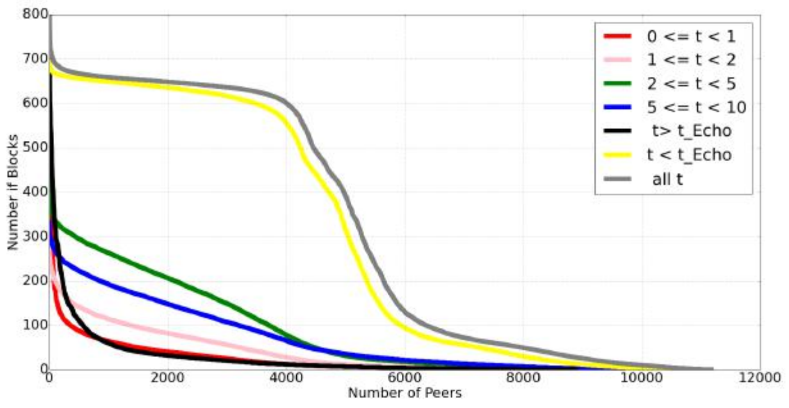

Figure 2: This figure shows the number of Blocks received from Peers in a defined time window. The red line groups how many Blocks (y axis) are received by nodes (x axis) in 1 second after the first propagation (t¡1 second).

valid blocks (from block height 410119 to 411327) mined during the listening time window. The most part of data regard transactions "inv" messages (589 GB) while the remaining is related to blocks "inv" messages. During the investigation we received a large amount of blocks and transactions, and we decided to classify them. The classification will be discussed in the following parts.

	Listening Time	Blockchain Time
Minimum Time	-5.48 seconds	-558 seconds
Maximum Time	4650.09 seconds	4642 seconds
Medium Time	550.05 seconds	550.05 seconds
Variance	550.11 seconds	550.30 seconds
Percentile 50%	383.25 seconds	384 seconds

Table 1: This table show some time statistics related to Mined During Listening Blocks set, comparing timestamp wrote on each block within the time reported inside the Blockchain. The time on the Blockchain can be wrong since a miner could vary the timestamp if the nonce don't converge to a valid proof-of-work block. The minimum time is negative due to a Fork event. During the monitored period we observed that the minimum time required to a block to be mined is about 2 minutes, while the maximum time is 77 minutes. Also, the medium time for discovering a block is about 9 minutes and the 50% percentile is about 6 minutes.

4.1 Blocks

Each Block received from a client is recorded within the receiving time and labeled using one of the following definition:

- Mined During Listening Blocks (MDLB): This set identify all the blocks which were included on the Blockchain during the listening period and propagated by the peers before the next block was discovered.

- Echo Blocks (EB): This set identifies all the blocks included in the Blockchain propagated with some delay.
- Fork Blocks (FB): This set identifies all the blocks not included in the Blockchain, propagated by the peers with a valid hash (below the proof-of-work threshold).
- Invalid Blocks (IB): This set identifies all the blocks not included in the Blockchain, propagated by the peers and having a hash above the proof-of-work threshold (so they should be discarded by the peers).

All of the preliminary results shown refer to MDLB set (if not specified otherwise). In figure 1 is possible to see a cumulative block propagation. The curves have different length due to the fact that each block is discovered after a varying time. Also the number of nodes depends on the number of connections established during the block propagation time. Table 1 shows some time statistics related to time required from miners in order to discover a new block. Despite data collected for each block are propagated by a different number of peers, preliminary results show that the propagation time seems to be quite stable on the network as showed in figure 2, depending only on the Source of the block and by the Bitcoin client used, as showed in figure 3.

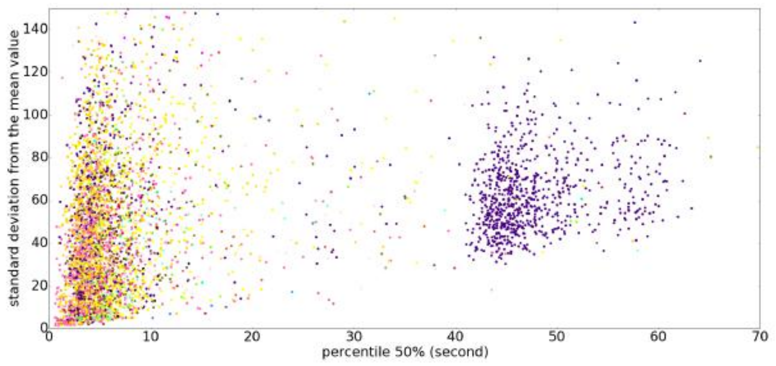

Figure 3: This figure shows the relationship between the 50% percentile of for a peer (x axis) with the standard deviation from the mean time (y axis). Each color represent a different Bitcoin client program. As it is possible to see the nodes who uses the purple client software are quite slower on receiving blocks compared to the other clients.

4.2 Transactions

Each Transaction received from a client is recorded within the receiving time and labeled using one of the following definition:

- Blockchain Transaction (BT): Valid Transactions, included in a Blockchain's Block and propagated before the block they are included is discovered and propagated through the network.
- Echo Transaction (ET): Valid Transactions, included in a Blockchain's Block but propagated in delay.
- Invalid Transaction (IT): Transactions not valid for some reasons.

Figure [?] shows the received transactions rate per hour for IT set (in red) and BT plus ET set in blue. We will proceed the investigation in order to see if peers have a competitive behavior (as seen for Blocks) or if they are more collaborative. During the listening time we received 1820212 Transactions. Among them there are 1722696 which were included in the Blockchain. The total number of Transactions included in the Blockchain during the monitoring time is 1723962. So our client do not received 1266 Transactions which were included in the Blockchain. Our investigation will continue understanding if the time required for include a transaction in the Blockchain depends on some factors, such as the source node who issued it.

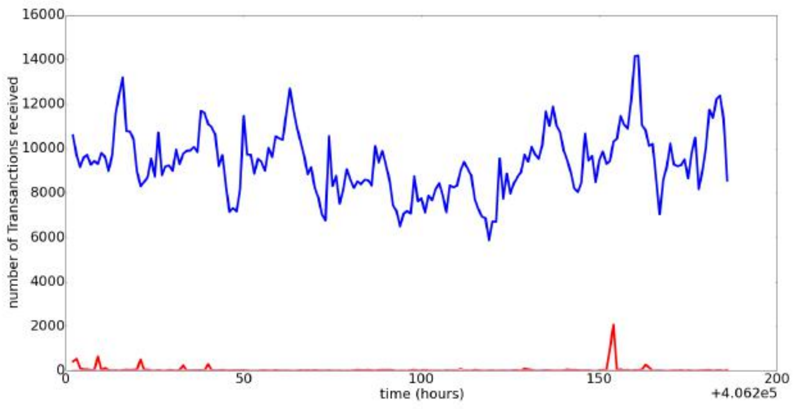

Figure 4: This figure shows the number of Transactions per hour received from Peers.The Blue line represent the transactions included in the Blockchain during or after the listening time. The red line represent all the invalid transactions.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] "Bitcoin protocol documentation." [Online]. Available: https://en.bitcoin.it/wiki/Protocol_documentation

[3] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," in *13th IEEE International Conference on Peer-to-Peer Computing (P2P), Trento, Italy*, September 2013.

[4] M. Andrew, L. James, P. Andrew, G. Neal, L. Dave, S. Neil, and B. Bobby, "Discovering bitcoin's public topology and influential nodes."

[5] Bitnodes is currently being developed to estimate the size of the bitcoin network by finding all the reachable nodes in the network. [Online]. Available: https://bitnodes.21.co/

[6] Guessing bitcoin's p2p connections. [Online]. Available: http://jonasnick.github.io/blog/2015/03/06/guessing-bitcoins-p2p-connections/

[7] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P network," *CoRR*, vol. abs/1405.7418, 2014. [Online]. Available: http://arxiv.org/abs/1405.7418

[8] The go programming languages. [Online]. Available: https://golang.org/