IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 1, 2019, accepted April 26, 2019, date of publication April 30, 2019, date of current version May 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2914098

Nodes in the Bitcoin Network: Comparative Measurement Study and Survey

SEHYUN PARK, SEONGWON IM, YOUHWAN SEOL, AND JEONGYEUP PAEK, (Senior Member, IEEE)
School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Jeongyeup Paek (jpaek@cau.ac.kr)

This work was supported in part by the Chung-Ang University Research Grants and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2017R1D1A1B03031348.

ABSTRACT Bitcoin is a decentralized digital currency that has gained significant attention and growth in recent years. Unlike traditional currencies, Bitcoin does not rely on a centralized authority to control the supply, distribution, and verification of the validity of transactions. Instead, Bitcoin relies on a peer-to-peer (P2P) network of volunteers to distribute pending transactions and confirmed blocks, to verify transactions, and to collectively implement a replicated ledger that everyone agrees on. This P2P network is at the heart of Bitcoin and many other blockchain technologies. In this paper, we present a comparative measurement study of nodes in the Bitcoin network. We measure and analyze how many the so-called ''volunteers'' are in the Bitcoin P2P network by scanning the live Bitcoin network for 37 days in 2018 and compare them with the data reported by prior work in 2013∼2016. This paper is motivated by the fact that Bitcoin has experienced explosive growth in terms of a number of users, transactions, value, and interest over a recent couple of years. Our investigation includes the IP addresses of Bitcoin nodes, size of the network, power law in the geographic distribution, protocol, and client versions, and network latencies and shows how today's network is different from early days. In addition, based on the observations made from the measurement study, we propose a simple distance-based peer selection rule for improved connectivity and faster data propagation. The evaluation results show that our proposed lightweight and backward-compatible peer selection rule has the potential to reduce data dissemination latency.

INDEX TERMS Bitcoin, peer-to-peer network, blockchain, peer selection, network measurement.

I. INTRODUCTION

Bitcoin is a decentralized digital currency proposed in 2008 in a paper authored by someone behind the Satoshi Nakamoto pseudonym [1]. Since then, it has reached a level of adoption unrealized by decades of previously proposed digital currencies [2]–[4]. Unlike most previous proposals, Bitcoin does not distribute digital monetary units to users. Instead, a public ledger (called the blockchain) maintains a list of every transaction made by all Bitcoin users since the deployment of the currency in January 2009 when it became fully functional.

Bitcoin network is an overlay network built upon peer-to-peer broadcasts[1] which carries all information in the network. Through broadcasts, network achieves eventual consensus and information consistency. Generated transactions are continually propagated to neighbor nodes to induce updates, and the block which contains several confirmed transactions are also created and propagated to the entire network. There are a number of nodes distributed all over the world (as we will show in Section IV) who, in a voluntary way or for profit, participate to the network.

At the beginning, Bitcoin network was designed and has evolved with rules based on the ''one CPU, one vote'' philosophy which proposed a democratic world-view. Therefore individual users often connected directly to the Bitcoin P2P network. However, as Bitcoin became popular and financially significant, block creating has become the domain of specialized miners. Some miners may run their customized mining client or use special-purpose hardware (e.g. ASIC) in an attempt to have an advantage over other clients [5]. Miners around the world may also organize into ''mining pools'', often hiding behind small number of gateway nodes.

The associate editor coordinating the review of this manuscript and approving it for publication was Kaiping Xue.

[1]Peer-to-peer broadcasts are conceptual at the application layer, and is actually implemented as multiple TCP unicasts in the underlying Internet.

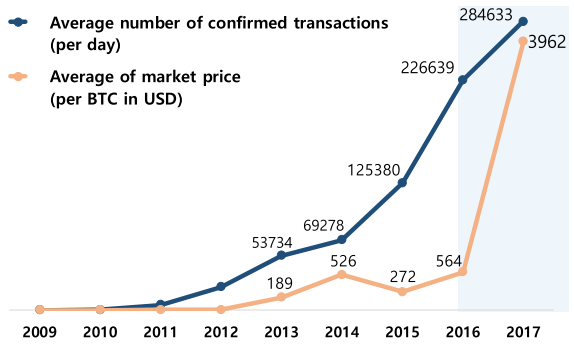


FIGURE 1. Average number of confirmed transactions per day, and average market price of 1 BTC (in USD) for each year. There has been an explosive growth in recent 2 years (Data from *https://blockchain.com/explorer*).

In addition, exponential growth of transactions is leading to overload of the network because the block size that needs to be processed has reached the originally anticipated limit. These changes inevitably leads to changes in the network topology.

Since the problems that these changes may bring are real-time and serious (in particular, financially), a lot of research and discussions among the community are underway (Section VI). For example, the 'decloaking' technique revealed the topology changes and its problems [6], and Decker et al. investigated the impact of information propagation delay on ledger consistency [7]. The work in [8] presented quantitative aspects of various information in the network, and Donet et al. took snapshots of a Bitcoin network once a day for 37 days to present the number of participants and their geographical distribution [9]. These prior work in 2013∼2015 has allowed better understanding of the Bitcoin network.

However, although unsure whether it is simply mere speculation or due to the true potential of blockchain technology, Bitcoin has experienced explosive growth in terms of number of users, transactions, value, and interest over the recent couple years. Figure 1 shows the increase in both the average number of confirmed transactions per day and average market price of 1 BTC, averaged over each year. For instance, the average market value of 1 BTC in 2015 was $271.83 USD (max $458.95), but has increased to $3962.38 in 2017, an order of magnitude (14.57×) greater. If we compare the starting value of 1 BTC in 2013 ($13.48) with the maximum market value of $19289.79 achieved in 2017, it is a 1430× increase. It can surely be said as an 'explosive' increase.

Our work is motivated by this explosive growth over the recent couple years. In this work, we present a comparative measurement study of nodes in the Bitcoin network. We measure and analyze how many so-called 'volunteers' are in Bitcoin P2P network by scanning the live Bitcoin network for 37 days in 2018, and compare them with data reported by prior works in 2013∼2016. Our investigation includes the IP addresses of Bitcoin nodes, size of the network, geographic distribution, protocol and client versions, network latencies, and more to show how today's network is different from early days. In addition, based on the observations made from the measurement study, we propose a simple distance-based peer selection rule for improved connectivity and faster data propagation. Simulation results show that our proposed light-weight and backward-compatible peer selection rule has potential to reduce data dissemination latency.

The remainder of this paper is structured as follows. We first provide a brief overview of Bitcoin and its P2P network in Section II. Then in Section III, we describe the design and implementation of our BITCOIN-NODE-SCANNER which we use to collect data in the live Bitcoin network. Section IV presents our measurement results and comparative analysis, and we propose a simple distance-based peer selection rule for improved connectivity and faster data propagation in Section V. Section VI discusses the related work, and we conclude the paper in Section VII.

II. BACKGROUND - BITCOIN

In this section, we provide a brief background of Bitcoin and its blockchain [1]. We present a high-level overview, along with some details only on those that are relevant to our work. Since Bitcoin clients are open source and each node may customize their software for their own purposes, our descriptions are that of the reference/default implementation (Satoshi client). For more comprehensive description, please refer to the original Bitcoin paper [1], or other related work in Section VI [4], [7]–[14].

A. BLOCKCHAIN

Bitcoin aims to achieve complete decentralization, and the trust relationship between participants rely on the blockchain technology which is an open ledger containing all current and historical transactions in the system. To prevent alterations of previous transactions and maintain the integrity of the ledger, the ledger is organized as a hash tree, and the system requires participants to use their computational power to generate proof of work (PoW) by solving cryptographic puzzles. The PoW is used as a distributed random function to implement leader election that is resilient to Sybil attacks.

A PoW is required to generate a block and add transactions to the blockchain. This process is essential for maintaining the integrity, correctness, and consensus, but on the other hand it is very expensive. Therefore, the nodes who create a new block earn a certain amount of Bitcoin (newly minted) in addition to transaction fees (bidded on each transaction) as an incentive. The process of block generation is called *mining*, and those carrying out this activity are called *miners*.

Once someone generates a new block by solving a PoW puzzle, and this solution is disseminated into the Bitcoin network, all nodes begin to solve a new cryptographic puzzle on top of this newly created block. The blocks are organized in a directed tree where each block contains a reference to a previously found block as depicted in Figure 2. The root block of the tree is called the *genesis block*, an ancestor of all blocks by definition, and is hardcoded into the client software. The

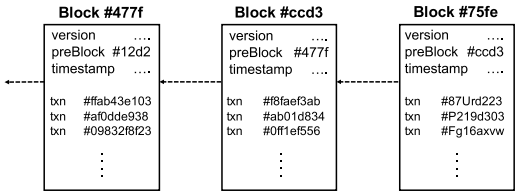


FIGURE 2. Blockchain: Each block contains the information (hash) of the parent block, and is linked in chronological order. Only the longest branch is accepted, and other shorter branches are deemed invalid – the consensus mechanism.

distance between a block b and the *genesis block* is referred to as its *block height*, and the block that is furthest away from the *genesis block* is referred to as *blockchain head*. Then, the blockchain is defined as the longest path from any block to the *genesis block* – any other shorter branch is called a 'fork' and deemed invalid by the nodes in the network. This is how consensus is maintained.

To include a reference to the parent block, that parent block's identity (its hash) has to be known in advance, and thus the child block must have been found after the parent. This chaining is used to assign a chronological order to the transactions: transactions in lower height blocks must have been verified before transactions in higher blocks. As only the blocks appearing in the blockchain (i.e. currently longest branch) will be rewarded with newly minted coins that are accepted by other users, miners will always attempt to find a block that builds on the current blockchain head. Building on a shorter branch, a fork, would be a waste of computation on what is likely be to invalidated by the consensus mechanism.

However, unfortunately, blockchain forks do occur in Bitcoin, more often than one may expect [7]. Blockchain forks may occur intended (e.g. selfish mining [14]–[18], double spending [13], [19], etc.), but also unintended due to propagation delay in the network [7]. A fork implies that the ledger replicas in the network may no longer be consistent; that is, the network is not at a consensus about the balances of the accounts[2] and which transactions are valid. For this reason, forks should be avoided as much as possible. There is one large thread of research on protecting against the intended attacks, and our work approaches the problem from a networking perspective on the unintended forks due to propagation delay. That is, we focus on the peer-to-peer nodes, connections, and message dissemination latency to propose a simple peer-selection rule that can make the information propagation faster and mitigate the problem.

B. NETWORK

All Bitcoin nodes (a.k.a clients) are connected to each other in a peer-to-peer network, and thus there are no central servers or authorities. A node wanting to join to the network for the first time thus needs to know who the peers are, which peers to connect to, and the addresses of those peers. For this purpose, a set of special peers called '*seed*' nodes, also known as 'Bitcoin DNS' nodes are used. Such seed nodes maintain a list of peers known them, and returns a subset of the list (in a *'addr'* message) upon request (*'getaddr'* message), where the nodes in the returned subset are chosen randomly and can contain up to 1000 nodes. The set of seed nodes are hardcoded into the Bitcoin client software for initial bootstrapping, and a subset known as of January 2018 are listed below.

- seed.bitcoin.sipa.be
- dnsseed.bluematt.me
- dnsseed.bitcoin.dashjr.org
- seed.bitcoinstats.com
- seed.bitcoin.jonasschnelli.ch
- seed.btc.petertodd.org

Note that, once a node knows of peers that it can connect to other than the seed nodes, it may ask any other peer for additional peers (address) list in the same way as asking the seed nodes. This is how we perform measurements.

After retrieving the peers list, a node *'randomly'* selects an address from its set of known addresses and attempt to establish a connection. A node repeats this process until it successfully connects to peers and the number of outgoing connections reach 8, the default maximum. If one of 8 outgoing connection is disconnected, the node will try to maintain 8 outgoing connections by trying to connect to another peer. At the same time, a node may accept incoming connections from other peers, up to 117, the default maximum number of incoming connections. A node will reject connection requests if this limit is reached. Note that, although 8 and 117 (total of 125) are the default maximum, the number of connections may vary according to the configuration and network settings of the Bitcoin client used [6].

It is well known that Bitcoin relies on peer-to-peer broadcast to distribute pending transactions and confirmed blocks. However, one of the misconception about Bitcoin's network by non-technical general public is that, this broadcast is a real (physical) broadcast to all nodes in the network. It is not. Bitcoin network is an overlay network built on top of peer-to-peer connections, and broadcasts are conceptual at the application layer. It is actually implemented as multiple TCP unicasts between connected peers determined by the connection establishment rule described above. Thus, (by default) up to only 125 peers will receive a message sent by a node. Then, *recursive flooding* is used to eventually propagate the message throughout the whole network.

Although Bitcoin's network formation procedure described above is intended to induce a random graph topology, however, due to reasons such as the use of a few designated seed nodes and uneven geographic distribution of nodes, Bitcoin network is not purely random. And the topology over which this flooding is executed affects the latency of information propagation. One node may receive new information earlier than others, and the time required to disseminate a message to 'all' nodes may take significantly more than it takes to deliver to connected nodes. If a data item does not spread throughout

[2]To be precise, we mean the value of the UTXOs for their corresponding keys.

the network quickly, then the system risks reaching an inconsistent state, a fork. Thus, the connection establishment rule used in Bitcoin nodes is of significant importance.

Finally, each node connected to the network may individually decide how to contribute to the network by choosing which service to provide. For example, by relaying transactions, by storing a copy of the blockchain or by using their own computational power for mining. We call the node who stores and serves the full blockchain as '*full node*'. A full node will (usually) accepts incoming connections, verify transactions and blocks, and contribute to data propagation. A full node may also mine for new bitcoins. On the other hand, a '*light-weight client node*' (e.g. Bitcoin wallet on your smartphone) does not accept incoming connections, does not store the blockchain in its storage, does not verify blocks, and (usually) does not participate in the message propagation. Instead, light-weight clients can make transactions (e.g. purchases, money transfer) and send that information to known full-nodes so that the full-nodes can propagate those transactions throughout the Bitcoin network. Intuitively, it would not be feasible to store hundreds of gigabytes of blockchain data on your smartphone and perform mining. Thus, most of casual users use only the light-weight clients on their personal PCs and smartphones. Of course, there could be nodes with other variations of client software that support partial subset of full node's functionalities.

Thus, although it is difficult to confirm whether a node is a full node or not by simple message exchanges, we can generally regard a node which accept incoming connection as a full node. As we will show later in Section IV, although there are over a million client nodes in the Bitcoin network, the number of full nodes are only in the order of ∼10000, another piece of information that non-technical general public are unaware of. Note that, in general, a peer-to-peer node or the software used by a node is often called a 'client' (e.g. Satoshi Client), and this term includes both full client nodes and light-weight client nodes. As such, this paper (and many other Bitcoin-related papers) use the term 'client' and 'node' interchangeably, and use specific terms 'full node' and 'light-weight client node' when necessary.

C. PROTOCOL DIVERSITY

Once connected to the network, a node can send and receive messages such as blocks and transactions to and from the connected peers. All these messages, and the actions taken upon the transmission or reception of these messages, have to follow the rules settled up by the Bitcoin protocol. However, the protocol can be updated or changed according to the need for improvements (e.g. block size increase, attack robustness), and each client may have different customizations for their own purposes. Table 1 shows that there has been a continuous update of the protocol.

None of those protocol versions or clients are enforced by a specific company, organization, or authority. Since Bitcoin is open source and decentralized, anyone can suggest improvements and participate in the development. Participants can actively apply the changes to their own clients, or adopt the clients published by others to meet their needs. Users may not want to apply some changes intentionally, and this is also allowable. If the protocol changes are insignificant and backward compatible, then the network will not be much different. In contrast, if there is a significant change in the protocol (e.g., changes in the structure of the block/transaction, change in the method of transmission, etc.), it will require an agreement between the participants. This agreement is also done in a distributed way through voting (within the protocol) by participants who support or oppose it. In some cases, the network might divide into several different protocols, clients, even blockchains.

TABLE 1. History of Bitcoin protocol (BIP: Bitcoin Improvement Proposal).

Version	Initial Release	Major Changes
70015	Jan 2017	New banning behavior for invalid compact blocks.
70014	Aug 2016	BIP 152 : Added *sendcmpct*, *cmpctblock*, *getblocktxn*, *blocktxn* messages.
70013	Aug 2016	BIP 133 : Added *feefilter* message Removed *alert* message system[3].
70012	Feb 2016	BIP 130 : Added *sendheaders* messages
70011	Feb 2016	BIP 111 : *filter* related messages are disabled without NODE_BLOOM.
70002	Mar 2014	BIP 61 : Added *reject* message. Send multiple *inv* messages in response to a *mempool* message if necessary.
70001	Feb 2013	BIP 37 : Added *filter* related messages.
60002	Sep 2012	BIP 35 : Added *mempool* messages. Extended *getdata* message to allow download of memory pool transactions.
60001	May 2012	BIP 31 : Added nonce filed to *ping* message. Added *pong* message.
60000	Mar 2012	BIP 14 : Seperated protocol version from Bitcoin Core version.
31800	Dec 2010	Added *getheaders* and *headers* message.
31402	Oct 2010	Added time field to *addr* message.
311	Aug 2010	Added *alert* message.
209	May 2010	Added checksum field to message headers, added *verack* message, and added starting height field to *version* message.
106	Oct 2009	Added transmitter IP address fields, nonce, and User Agent(subVer) to *version* message.

Due to these unrestrictedness of division, two kinds of fork may occur in Bitcoin due to protocol updates; *soft-fork*, and *hardfork*. The former just changes previous valid blocks/transactions to invalid. It is forward-compatible since old nodes will recognize the new blocks as valid. On the other hand, in *hardfork*, old nodes can not validate blocks created by upgraded nodes that follow newer protocol, so it makes a permanent divergence in the blockchain. In this way, *Bitcoin Cash* (Forked at Block 478558, 1 August 2017) and *Bitcoin Gold* (Forked at block 491407, 24 October 2017) are hardforked from Bitcoin. This coin split is also a major difference between the current network and the past network.

D. MESSAGES

There are several message types and formats defined in the Bitcoin protocol. Most frequently used ones are the following:

• **version**: This is the first message that a client sends to connect to a peer (at the app-layer, given successful TCP connection). It contains various information to connect such as client's protocol version, services, IP addresses, etc.

• **verack**: 'Version ACK' message is a confirmation message in response to the version message. Verack message can be omitted in recent protocol versions.

• **getaddr**: 'Get address' message requests for peer's known address list.

• **addr**: 'Address' message contains a list of peer addresses that a node knows of. This may be sent in response to a *getaddr* message, or periodically to update the information of connected peers. It may contain up to 1000 addresses.

• **inv**: 'Inventory' message is used to announce the indexes of block and transaction data which the sender has. Note that an 'inv' message contains only indexes but not the actual block / transaction data.

• **getdata**: This message is used to request for the actual block / transaction data that has been announced in the *inv* message. Upon receiving an *inv* message, a node compares the index included in the *inv* message with what it already has, and requests for new data that it is currently missing.

• **tx** :'Transaction' message contains the actual transaction data (version, input, output, amount, etc.), and it is sent in response to *getdata* message.

• **block**: This message contains the actual block, and it is also sent in response to a *getdata* message.

When a Bitcoin node wishes to connect to a peer, the sequence of message exchanges for connection establishment is as follows. Let the client who initiates (outgoing) connection be A, and the client who accepts (incoming) connection be B. First, A establishes a TCP connection to B. If successful, A sends a *version* message to B. Then B check the validity of the received message and sends a *version* message to A. Upon reception of B's *version* message, A sends *verack* to B, then B send *verack* to A. Through this handshake procedure, two peers knows of each others protocol version, client software, services, etc., and regard each other as connected. Once connected, two nodes exchange other messages such as *inv*, *block*, *tx*, etc. for data propagation and relaying. They may also exchange *getaddr* and *addr* messages to obtain addresses of more peers. Figure 3 depicts this message exchange process.

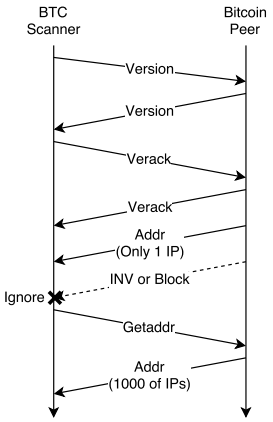


FIGURE 3. Message exchange process between BITCOIN-NODE-SCANNER and a connected bitcoin peer.

III. BITCOIN-NODE-SCANNER

In this section, we describe the design and implementation of our own "BITCOIN-NODE-SCANNER" software for analyzing the Bitcoin network.

BITCOIN-NODE-SCANNER is based on, and adheres to the Bitcoin protocol,[4] and implements a subset of its functionality to act as a client. At a high level, it scans through the Bitcoin network to obtain as much known IP addresses of Bitcoin nodes as possible (which may include full-node peers and light-weight clients), and verifies the connectability to each node. If connected, BITCOIN-NODE-SCANNER obtains information such as the protocol version, client agent type, round-trip-time from the scanner, etc. from the connected peer using the Bitcoin protocol. Several prior work has used similar approach [5]–[7], [10], [20]–[22]. However, some of these are outdated (with older protocol version), not publicly available, and did not collect the information that we needed. Thus, we adopted the ideas from prior work but implemented our own up-to-date scanner for collecting data needed for our analysis. We have also made our up-to-date scanner publicly available.[5]

When the BITCOIN-NODE-SCANNER starts, it does not know of any peer's IP addresses. By issuing a *getaddr* command to the seed nodes (Section II-B), BITCOIN-NODE-SCANNER obtains a list of nodes (IP address and port number pairs) that are known to the seed nodes. In our experiments, seed nodes usually returned about ∼400 IP addresses including both IPv4 and IPv6 addresses. Then by recursively applying the same procedure to all reachable nodes that we discover and accept connection, that is, by sending *getaddr* commands to all connectable nodes we find, we discover more nodes.

Figure 4 depicts the main functional blocks of our BITCOIN-NODE-SCANNER implementation. Node addresses returned in response to *getaddr* requests are enqueued into the *IP Queue* which stores all the unique IP addresses (together with the port number) that needs to be connected to. In other words, *IP Queue* stores all nodes that are pending reachability check and attempt to connect, after filtering out the duplicates and already processed addresses. At the same time, all new unique IP addresses are stored to the *Connection DB* (part of the '*Data Repository DB*') which is implemented using a hashmap with IP address as the key. It stores various information obtained from each node, as well as a flag indicating whether a connection attempt has already been made to the IP address or not to prevent connecting again to the same node.

[4]Bitcoin protocol wiki: https://en.bitcoin.it/wiki/Protocol_documentation

[5]https://github.com/Crepepepe/JBS

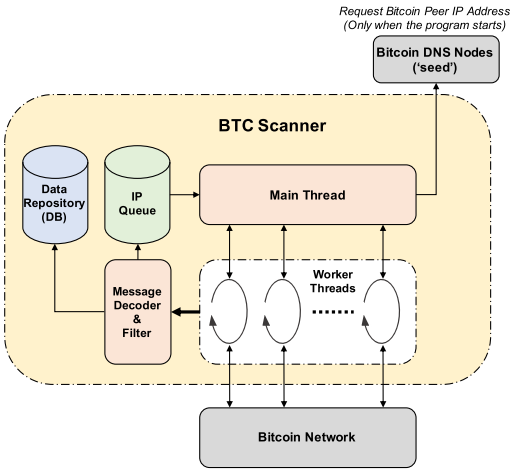


FIGURE 4. Functional block diagram of BITCOIN-NODE-SCANNER's implementation. *Data repository (DB)* includes *Connection DB* and other DBs.

Then, BITCOIN-NODE-SCANNER dequeues one node at a time from the *IP Queue* and attempts to connect to that node. If BITCOIN-NODE-SCANNER fails to connect to a node, it stores the result to *Connection DB* and continues with the next node in the *IP Queue*. If it succeeds to connect to a peer, scanner exchanges *version* and *verack* messages with that peer. After receiving *verack* message, scanner sends *getaddr* message to the peer to request for an *addr* message which usually contains approximately 1000 IP addresses along with port numbers and timestamps. This message exchange process is depicted in Figure 3. During this procedure, BITCOIN-NODE-SCANNER also obtains other information such as the protocol version, client agent type, round-trip-time from the scanner, etc. It can also listen to messages such as *inv*, transactions, and blocks broadcasted by the connected peer. After filtering out duplicate IP addresses based on *Connection DB*, only those IP addresses that we have not yet attempted to connect will be enqueued to the *IP Queue*. The process ends when *IP Queue* is empty; that is, there are no more new nodes pending to be connected and queried.

If BITCOIN-NODE-SCANNER succeeds to connect to a peer but does not receive any messages from that peer, it waits for 10 seconds to timeout, disconnect that peer, and continues to try to connect to other peers. Due to the large number of unreachable nodes (TCP connection failures) and nodes that do not respond to messages even after successful connection, timeout duration till aborting the attempt could prolong the scanning process significantly. Thus, to expedite the whole process, BITCOIN-NODE-SCANNER uses maximum of 128 threads simultaneously where each thread handles a connection attempt to one node at a time, resulting in up to 128 simultaneous connections.

Since peers of the Bitcoin network connect to each other over a TCP channel, we use *Netty*[6] library to handle TCP sockets. When trying to connect to a peer, scanner makes a *StateBundle* which contains *version* and *ping* message from peer, error messages which occurred during communication, number of IP addresses in *addr* message and number of newly discovered IP addresses. Messages are received in raw bytes, which are collected and passes to the *ChannelInboundHandler* until the *ByteToMessageDecoder* matches the checksum in the message header. The Handler sends the following message to the peer depending on the message type in the header. When scanning finishes, data manager saves the *Connection DB* and *StateBundle DB* into log files in JSON format by using *Jackson*[7] library.

As a final note, we are not claiming that our BITCOIN-NODE-SCANNER is superior than those used in prior work in terms of identifying more number of unique IP addresses. The comparison cannot be fair if they are not performed at the same time. The main point of the comparison is that, since Bitcoin had significant growth (in terms of users/transactions/value) during the past few years, we are interested in seeing how the network statistics (e.g. number of nodes, country distribution, version/protocol, etc.) changed over time. In other words, we are not comparing the tools for collecting data, but comparing the data itself with the goal of investigating the evolution of Bitcoin nodes. Regarding the data collection method, all we are claiming is that we have done our best effort to collect the data with due diligence.

IV. MEASUREMENT

This section presents a comparative measurement study and analysis between our collected data and the data reported in prior work.

A. METHODOLOGY

Using BITCOIN-NODE-SCANNER, we conducted one scan every day at 12:30 PM KST from January 16th, 2018 to February 21st, 2018 for 37 days. Each scan took around 8 to 10 hours to complete using 128 threads and 10s timeout. The reason that we selected 37 days for data collection period is to do a comparative analysis with the work by Donet *et al.* [9] which also performed data collection for 37 days. We had one caveat; the network on which we run experiments does not support IPv6 addresses. For this reason, although we collected both of IPv4 and IPv6 nodes addresses, connection attempts were made to only IPv4 addresses.

In this work, we identify Bitcoin nodes by their IP addresses. Among all the IP addresses that we collect and attempt to connect to, some nodes do not accept connection, and some nodes do accept connection but do not respond to messages. For the conciseness of analysis and presentation, we classify nodes into two categories from here on. We call a node *'full node'* if it accepts connection and responds to messages. For all other nodes, we regard them as *'client node'*. *'All IP addresses'* include both full nodes and client nodes. This definition and approach has some potential

[6]Netty library: https://netty.io

[7]Jackson library: https://github.com/FasterXML/jackson

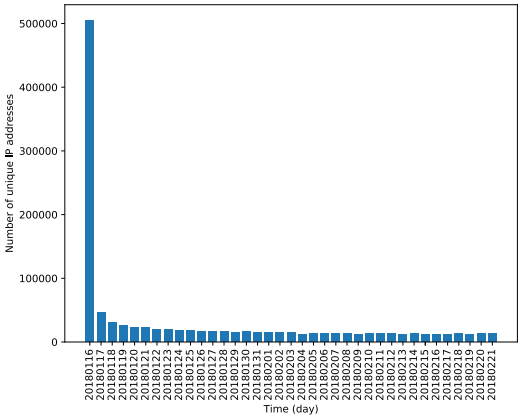


FIGURE 5. Newly discovered unique IP addresses (collected using *getaddr* request to bitcoin peers, and known to run Bitcoin node) per day.

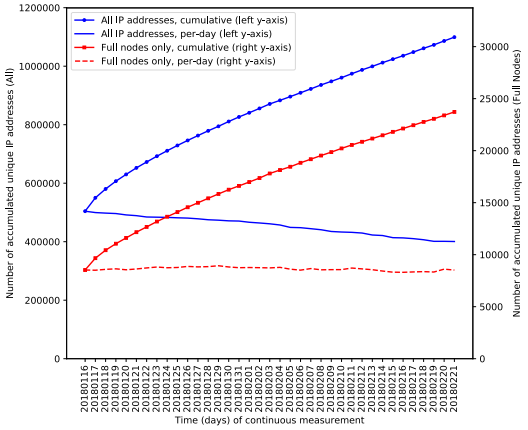


FIGURE 6. Total number of discovered unique IP addresses (collected using *getaddr* request to bitcoin peers, and known to run bitcoin node) per day, as well as its cumulative unique count, for full nodes and all nodes (full+client).

limitations and may not be precise since nodes may use dynamic IP addresses, be behind network address translation (NAT), be behind a firewall, or be hidden in a mining pool. Furthermore, some full nodes may not respond to messages (for whatever reason the user of that node/agent has), and nodes do not return all addresses they know in response to *getaddr* requests. For these reasons, we acknowledge the fact that the IP addresses that we have collected may not be the complete picture of the Bitcoin network, and the distinction between *'full node'* and *'client node'* may be an approximation.

B. NUMBER OF NODES AND IP ADDRESSES

After the 37 days of data collection, we have discovered total of 1,099,322 unique IP addresses corresponding to machines running Bitcoin nodes. Among those discovered IP addresses, we found 504,283 (45.8%) on the first scan on the first day, which implies that there are significant overlap between nodes found on each day. Interpreting the same data from an opposite perspective, we see that the IP addresses obtained on each day has quite a bit of differences. Figure 5 plots, starting from the first scan, the number of newly discovered unique IP addresses per day, where x-axis is the actual date of the scan. It can be seen that the number of newly found IP addresses gradually decreased to approximately 12,000 per day. Among those 1,099,322 unique IP addresses, 23,725 were full nodes (i.e. were connectable and responded to messages) out of which 8527 were found on the first day. On average, we found approximately ∼8500 full nodes every day, where the number of newly discovered full nodes were around 1000 ∼ 300 per day. Figure 6 plots the number of all discovered unique IP addresses (known to run Bitcoin node) per day, as well as the cumulative number of unique IP address (left y-axis). It also plots the number of discovered full nodes per day, as well as its cumulative count (right y-axis).

Now, let's compare our numbers with prior work. Decker and Wattenhofer [7] identified ∼16000 unique IP addresses among which ∼3500 was reachable on one day in 2013. Thus for a one day scan, we see 31.5× times greater number of Bitcoin nodes in 2018, a significant increase in 5 years, while the number of full nodes has increased by only 2.4× times.

In 2014, Donet *et al.* [9] identified a total of 872,648 unique IP addresses during 37 days, where the first scan on day-1 found 111,475 nodes. Thus, our measurement shows only 25% increase for the 37-day data, but approximately ∼5× times increase for the day-1 count in 2018 compared to Donet's data in 2014. Also in 2014, Biryukov *et al.* [12], reported ∼100,000 IP addresses (in one day) among which ∼8000 where reachable full nodes. The number of all IP addresses is consistent with Donet's work, a ∼5× times increase, while the number of full nodes has increased by only ∼7%.

In 2016, Pappalardo *et al.* [5], [10] observed 12,425 unique peers (among which 8969 on IPv4 network) during 7 days in May 2016, with an average of ∼6000 peers participating to the Bitcoin network at the same time. Also in 2016, Fadhil *et al.* [21] conducted measurement for one week and found 6430 peers and 313,676 client IP addresses. Compared to Pappalardo's work, our data has approximately 20∼30% increase in the number of full nodes. For the Fadhil's work, our measurement has roughly 2× more numbers and we are guessing that the methodology of measurement could have been slightly different.

Overall, we observe that the number of Bitcoin nodes (in terms of unique IP addresses) has increased significantly (31.5×) over the past 5 years. However, the number of *full nodes* has increased at a much lower rate (2.4×) than the number of all nodes. This implies that, although the number of Bitcoin "users" has increased significantly, majority are passive users using *light-weight clients* only, presumably general public that are interested in the financial aspect of Bitcoin. On the other hand, the number of "peers", the active users who participate in the peer-to-peer network and contribute in maintaining the blockchain by providing a *full node*, has not increased much. This trend may not be what the original 'one CPU one vote' philosophy intended, and may change the direction of how the technology will evolve in the future.

TABLE 2. Summary of number of nodes (IP addresses) in related work.

Work by	Year	One day		Multi-day	
		All node	Full-node	All node	Full-node
[7]	2013	∼16,000	∼3,500		
[9]	2014	111,475		872,648	
[12]	2014	∼100,000	∼8,000		
[5], [10]	2016		∼6,000		∼12,425
[21]	2016			313,676	6,430
Our work	2018	504,283	8,527	1,099,322	23,725

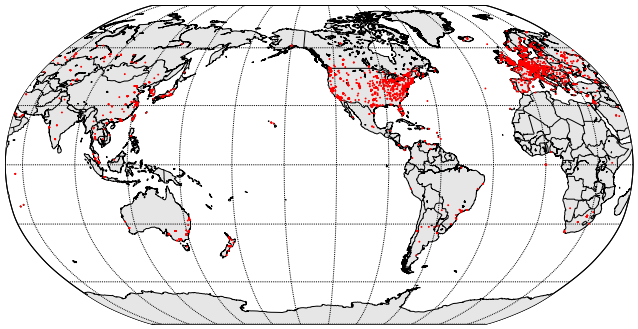


FIGURE 7. Map of full-node locations based on IP addresses.

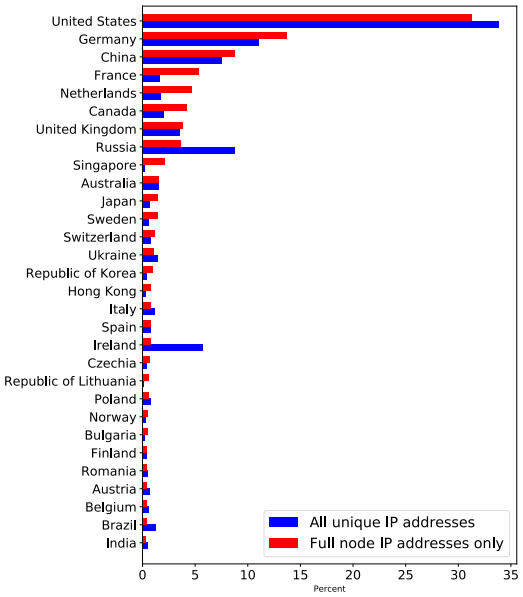


FIGURE 8. Percentage ratio (%) of all IP addresses and full nodes count for each country, ordered by the number of full nodes.

C. PORT NUMBER

The default TCP port number used by the Bitcoin network is 8333. However, some nodes may use different port numbers for various reasons such as hardfork, firewall/security, testing, or to be in a mining pool. For example, different cryptocurrencies branched by bitcoin hardfork use same protocol and network while use different default port number such as 8335 for Bitcoin Cash and 8338 for Bitcoin Gold. Furthermore, light client nodes may use dynamic port number (0), and Bitcoin test network uses 18333. In our collected dataset, 96.8% of the full nodes and 93.7% of all nodes used the default port number 8333, a dominant proportion which implies that not many users change the default port number. Other (relatively) popular port numbers are 8555 (3.7%), 9145 (1.5%), 8338 (0.41%), 8334 (0.22%), 8838 (0.16%), 18333 (0.04%), 0, etc., but their proportions are small, and those not listed here are almost negligible.

D. GEOGRAPHICAL LOCATION

We identify Bitcoin nodes by their IP addresses, and we can infer the geographical distribution by locating those IP addresses. We used *geolite2* IP geolocating package[8] to determine the country of a peer using the IP address. Figure 7 depicts the geo-location of each Bitcoin node's IP address on the globe (better seen in color), and Figure 8 presents the percentage distribution of how many unique IP addresses of Bitcoin nodes are located in each country. We counted both 'all unique IP addresses' of nodes as well as the unique IP address of 'full-nodes' only. The table is sorted by the number of full-node IP addresses, and lists the top 30 countries that show the highest number of Bitcoin nodes during the 37-days of our data collection.

The figure shows that United States is clearly at the top with over 30% for both 'all node' and 'full node' IP addresses. Germany is at the second place, and then follows China. It is interesting to see that France, Netherlands, and Canada has relatively higher proportion of full-nodes in comparison to all-node IP addresses, where as Russia and Ireland are the opposite which means that there are many 'users' but less 'volunteers', relatively.

These results are somewhat similar and different from the data reported by Donet *et al.* in 2014 [9]. Note that they had data for IP addresses of all Bitcoin nodes without distinction for full-nodes, and thus we compare the rank ordering of all-node IP addresses only. In [9], it is identical that United States was at the top. However, China was at the 2^{nd} place (14.98%), followed by Germany (6.90%), United Kingdom (6.14%), and Russia (6.14%). Ireland was not even in top 25 countries. The proportion of United States went up from 22.08% to 37.63%, the gap between United States and the second country became larger, and China's rank went down to 4^{th} (3^{rd} for full nodes) with notable reduction in the distribution share. This may apparently be because relatively more users are joining in United States, or because more nodes are hiding behind NAT or mining pools in other major countries such as China.

More interesting observation from the country distribution data can be seen in Figures 9(a) and 9(b), which plot the rank of each country versus the percentage ratio of those countries for their IP addresses and full-node count in linear and log-log scale, respectively. The data shows *power-law distribution*, with their trend and coefficients similar to Zipf's law. Specifically, for the 37-days full-node IP address data, the percentage

[8]geolite2 package: https://github.com/rr2do2/maxminddb-geolite2

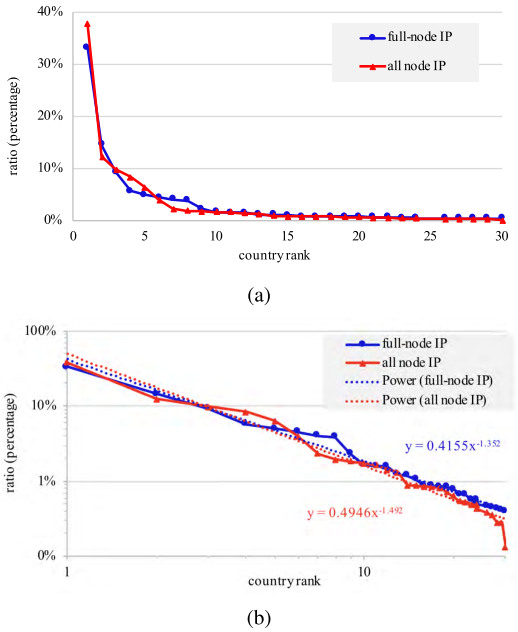


FIGURE 9. Rank vs. ratio (percentage) of countries for their IP addresses and full node count. The figures show power-law distribution. (a) Linear scale. (b) Log-log scale.

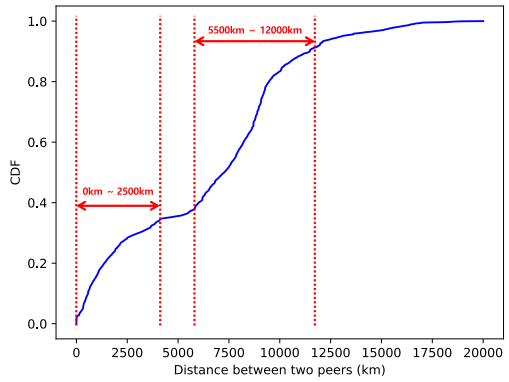


FIGURE 10. CDF of the distances between any two peers, for all pairs of full-nodes, based on geo-locations of their IP-addresses.

ratio of a country can be approximated/estimated based on their rank using the relationship,

$$ratio_f(rank_f) = \frac{0.4155}{rank_f{}^{-1.352}} \quad (1)$$

Similar distribution holds for all node IP addresses as well as for 1-day data, with their power-coefficients ranging between $-1.352 \sim -1.506$ and the numerator coefficients ranging between $0.4155 \sim 0.5071$. For example, the rank-ratio relationship for the 37-days all-node IP addresses can be approximated as,

$$ratio_a(rank_a) = \frac{0.4946}{rank_a{}^{-1.492}} \quad (2)$$

TABLE 3. Comparison of Bitcoin Nodes' Protocol Version Distribution in 2018 with Data From 2016 [10].

Protocol	Our data(2018)		From 2016 [10]	
	number	percent	number	percent
70015	21336	88.671		
80002	501	2.082	1	0.008
70012	433	1.800	6655	55.770
70001	427	1.775		
70002	403	1.675	3013	25.249
70014	396	1.646	2	0.017
80003	271	1.126		
70016	127	0.528		
70013	76	0.316	88	0.737
60002	27	0.112	1	0.008
99999	17	0.071	4	0.034
70010	5	0.021	78	0.654
70011	4	0.017	68	0.570
7000			1153	9.662
0			771	6.461
80001			71	0.595
80000			24	0.201
50400			2	0.017
70003			1	0.008
60000			1	0.008
Other	39	0.162		
Total	24062		11933	

Finally, to get a sense of geographical distribution of full-node peers, we plotted the cumulative distribution function (CDF) of the distances between any two peers, for all possible pairs of full-nodes that we found, based on the geo-locations of their IP-addresses. This is shown in Figure 10. The purpose of this figure was to get an idea on how distant (physically) are the peers and how they are clustered. We have plotted for full-node peers only since only the full nodes participate in the information dissemination, and physical distance indirectly correlate to propagation latency in the network. Figure 10 shows that there are steep increases at distances below 2500km and between 5500km $\sim$ 12000km, but a plateau around 4000km $\sim$ 6000km and also after 12500km. This is because the distance between North America and Europe over the North Atlantic ocean is approximately 6000$\sim$8000km, and the distance between Asia and North America over the Pacific ocean is approximately 9500$\sim$11000km. Centered around the majority of nodes within United States (32.03%), there are many node pairs that are within continent (<2500km) and also pairs that are inter-continent (6000km$\sim$12500km). This distance distribution is one of the basis which inspired us to design our simple distance-based peer-selection rule in Section V.

E. BITCOIN PROTOCOL VERSION / AGENT TYPE

The work by Pappalardo *et al.* [10] presented the distribution of Bitcoin peer's protocol version and client agent type using their data collected in 2016. To understand the changes between the past study (in 2016) and our measurement (in 2018), we present Table 3 and Table 4 which compares the distribution of protocol version and client agent type respectively. For both tables, we have listed the items that are above 0.01%, and grouped the remaining into 'other's

TABLE 4. Comparison of Bitcoin Nodes' Client Agent Distribution in 2018 With Data From 2016 [10].

Agent	Our data (2018)		From 2016 [10]	
	number	percent	number	percent
Satoshi:0.15.1	13670	56.812		
Satoshi:0.15.0.1	1866	7.755		
Satoshi:0.14.2	1753	7.285		
Bitcoin ABC:0.16.1	898	3.732		
Satoshi:0.15.0	893	3.711		
BitcoinUnlimited:1.0.3	447	1.858		
Satoshi:0.14.1	378	1.571		
Satoshi:0.15.99	373	1.550		
Satoshi:0.12.1	369	1.534	1790	16.812
Satoshi:0.13.1	319	1.326		
Satoshi:0.13.2	291	1.209		
BUCash:1.1.2	249	1.035		
Satoshi:0.16.0	227	0.943		
Satoshi:0.14.0	180	0.748		
BitCore:0.14.1.6	178	0.740		
Bitcoin ABC:0.16.0	170	0.707		
Satoshi:0.8.6	141	0.586	72	0.676
Satoshi:0.8.1	134	0.557	20	0.188
Satoshi:0.16.99	134	0.557		
Satoshi:0.8.5	109	0.453	42	0.394
Satoshi:0.9.1	67	0.278	71	0.667
Satoshi:0.9.3	66	0.274	122	1.146
Satoshi:0.11.2	54	0.224	1465	13.760
Satoshi:0.11.0	46	0.191	368	3.456
Satoshi:0.9.2.1	44	0.183	56	0.526
Satoshi:0.10.2	31	0.129	233	2.188
Satoshi:0.12.0	30	0.125	1691	15.882
Satoshi:0.10.0	17	0.071	116	1.090
Satoshi:0.12.99	9	0.037	144	1.352
Satoshi:0.11.1	7	0.029	226	2.123
Satoshi:0.10.1	7	0.029	67	0.629
Classic:0.11.2	7	0.029	184	1.728
Classic:0.12.0	6	0.025	2969	27.886
Bitcoin XT:0.11.0	3	0.012	26	0.244
Bitcoin XT:0.11.0E	2	0.008	22	0.207
N/A			771	7.241
BTCC:0.12.1			93	0.873
BitcoinUnlimited:0.12.0			70	0.657
Bitcoin XT:0.11.0D			29	0.272
Other	887	3.686		
Total	24062		10647	

category. However, we did not omit the current items that overlap with the past data.

We can observe from Table 3 that the protocol version has indeed evolved over the past 2 years. The dominant protocol version in 2018 is '70015' (88.67%), where as it used to be '70012' (55.77%→1.8%) and '70002' (25.24%→1.67%) in 2016. At the same time, we can also see that many older and less popular versions are still being used in the current network although their proportions have reduced.

For the client agent, Table 4 shows that '*Satoshi:0.15.1*' (56.81%) has the majority share currently while all other agent types each have less than 8%. This distribution characteristics is quite different from that reported in 2016 where there used to be multiple popular agents; '*Classic:0.12.0*' (27.88%), '*Satoshi:0.12.1*' (16.81%), '*Satoshi:0.12.0*' (15.88%), and '*Satoshi:0.11.2*' (13.76%); each having over 10% popularity. This implies that the users are converging onto a common type of client agent due either to keep up-to-date with the protocol updates (and willingness to do so) or for easy of installation and configuration. It is interesting to see that very old versions of client agents are still being used by some of the users today.

V. PEER SELECTION RULE

In this section, we propose a simple and backward-compatible distance-based peer-selection rule that can reduce the information propagation delay in the Bitcoin network, and explore the alternatives. As we have discussed earlier, network latency is one of the major reasons for blockchain forks, and forks should be avoided as much as possible since it implies inconsistency in the consensus mechanism. We have also mentioned that information propagation in Bitcoin's peer-to-peer network is in fact a recursive flooding over an overlay network, the number of direct connections between peers are limited, and thus the selection of which peers to connect to (directly) is an important design decision. Finally, the default behavior in Bitcoin is to choose peers randomly. To this end, our goal is to modify this 'random' choice with some *informed selection* to improve the connectivity without any disruption or changes to the protocol. Details are as follows.

A. PROPOSAL

Selecting which peer to connect to, among all the peer addresses that a node has learned or obtained through *getaddr*/*addr* messages, can play a critical role in the P2P connectivity graph and message dissemination latency. Default behavior of Bitcoin client software defines up to 8 outgoing connections and 117 incoming connections. Thus, once a node obtains the list of peer addresses, it must choose 8 peers that it wishes to connect to. Again, the default behavior of Bitcoin client software is to choose randomly. That is, among all the peer addresses it knows of, it selects 8 random peers and attempts to connect to them. Of course, the selected peer may not accept the connection request for several reasons such as not being a full node, does not wish to accept incoming connections, firewall/NAT, or because it already has full number of connections that it can support (e.g. 117). In that case, the algorithm should move on to the next possible peer for connection.

Our proposal is to modify this 'random' choice with some *informed selection* to improve the connectivity, and the basis of our approach is the geographical distribution of full-node peers (Figure 10). Based on the observation made from the distances between any possible pair of full-nodes, our idea is to choose x nodes that are close by, y nodes that are in mid range, and z nodes that are far way, among the 8 possible choices ($x+y+z=8$). And the distinction of near/mid/far would be probabilistic based on the boundaries at which Figure 10 shows steep increases. Intuitively, the physical distance indirectly correlates to propagation latency determined by the network topology, and our proposal is inspired by prior work in complex networks which have shown that networks with small-world topology can spread information faster than lattice networks [23].

For comparison purposes, we implemented five rules:

- **Rule 1 – [Default]**: Random 8 nodes from the known peers list, the behavior of default Bitcoin client.

• **Rule 2 – [Nearest]**: Nearest 8 nodes from the known peers list, an extreme case of locality based approach.

• **Rule 3 – [Clustering]**: Random 7 nodes from the nearer half, and 1 node from the farther half. In other words, we sort the known peer-addresses list based on distance from this node, divide that list into half, and randomly select 7 nodes from the first half and 1 nodes from the second half.

• **Rule 4 – [Half&Half]**: Random 4 nodes from the nearer half, and 4 nodes from the farther half. Similar to 'Rule 3', but 4-4 instead of 7-1.

• **Rule 5 – [Near-Mid-Far]**: Random 3 nodes from the nearer $\frac{1}{3}$, 3 nodes from the mid range $\frac{1}{3}$, and 2 nodes from the farther $\frac{1}{3}$. Said differently, we sort the known peer-addresses list based on distance from this node, divide that list into three, and randomly select 3/3/2 nodes from each one thirds.

Among these rules, 'Rule 1' is the default behavior of Bitcoin clients, and 'Rule 5' is our proposed approach where we have used x=3, y=3, and z=2. Rules 2∼4 are implemented for comparison purposes, where Rules 2 and 3 may be considered similar to what was proposed by Fadhil *et al.* [21], [24]. However, their work either had an unclear notion of 'long' link, or had a strict cut-off threshold of 100km which would not work if there were no peers within 100km or outside 100km. We have also tried several other variants of Rule 5 with different values for x, y, and z, and found the 3-3-2 to be the best combination with the current list of IP addresses.

B. EVALUATION METHODOLOGY

To evaluate our proposed approach, we implemented a custom-designed event-based simulator in JAVA which simulates the information propagation latency of recursive flooding in the Bitcoin's multihop overlay network. The simulator takes as input the IP address list of all full-nodes. In addition, since we need to simulate the propagation delay between any two connected peers, we use the round-trip-time (RTT) measurements that we collected using ping to all unique IP addresses known to run Bitcoin nodes. This approach is similar to that of Sallal *et al.* [25]. Results are plotted in Figure 11 where the straight line represent a linear fit on the measurements. We use this fit, divided by 2, to simulate the one-way propagation latency between any pair of nodes given the distance. Then, since transfer of a Bitcoin data message requires at least a 3-way handshake of *inv–getdata–tx/block*, we multiply the one-way propagation latency by 3 to calculate the 1-hop transmission latency. Note that, since Internet latency does not satisfy the triangle inequality and we have no way of knowing the actual network topology without having access to all Bitcoin full nodes, our approach is a best-effort approximation.

In addition, to simulate the transaction/block verification time at each node, we add a data processing latency $T_{\text{processing}}$ at each hop whenever a node needs to forward a message. In reality, this time will vary depending on the block size, processing power of the Bitcoin node, and the workload on that device. We thus have experimented with several values

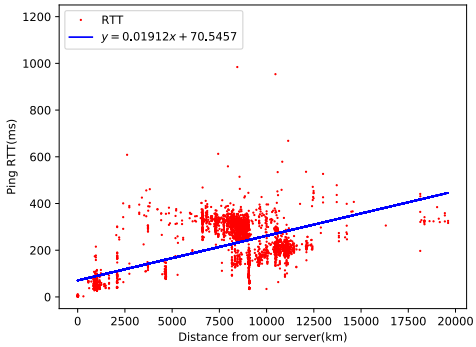


FIGURE 11. Round-trip-time (RTT) measurement using ping to all unique IP addresses known to run bitcoin node. Straight line represent a linear fit on the measurements.

such as 10ms, 45ms, 100ms, 1000ms, and 2700ms based on the discussions found online.[9] We provide only a representative result using $T_{\text{processing}} = 45$ms here for brevity.

For the evaluation metric, 'information propagation latency' is defined as the time required to deliver a message from an originator node to all nodes in the network through recursive flooding. A message will be delivered directly to connected peers in 1-hop from the sender, but otherwise will require forwarding over multiple hops. We transmit 1000 messages with randomly selected originator nodes, and run the simulation 5 times to average the results.

C. EVALUATION RESULTS

Figure 12 plots the information propagation latency to all nodes in the Bitcoin network using various peer selection rules. It shows that our proposed 'Near-Mid-Far' approach achieves reduced propagation latency compared to all other approaches in terms of lower mean, median, max, min, and standard deviation. We have tried extensive combinations of x, y, and z values, and 3-3-2 was one of the best configurations. At the same time, the result shows that 'Nearest' or 'Clustering' approaches are not preferable. Although the improvement achieved by 'Near-Mid-Far' approach is not huge, there was clear and consistent improvement for every run of the experiment. It also shows that the default 'random' selection is a reasonable approximation of a better-informed strategy that trades-off implementation simplicity, and implies that without the full topology information from within the network, there is no substantial room for improvement in peer selection rule.

Figure 13 is the CDF of the same result in Figure 12. It shows more clearly that 'Near-Mid-Far' approach improves on the latency distribution of information propagation, and confirms that 'Nearest' or 'Clustering' ([21], [24]) approaches should not be selected. The reason for such result can be inferred from Figure 14 which plots the CDF of the number of forwarding hops required (by recursive flooding) for information propagation to all nodes in the Bitcoin

[9]e.g. https://bitcoin.stackexchange.com/questions/50349/how-long-does-block-validation-take

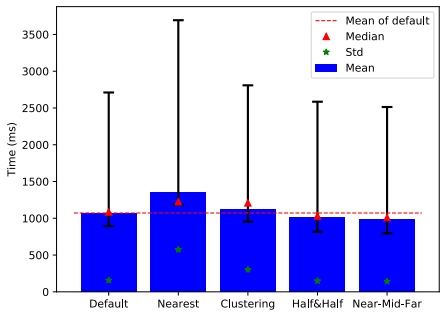


FIGURE 12. Information propagation latency to all nodes in the bitcoin network using various peer selection rules. Error-bars represent min and max.

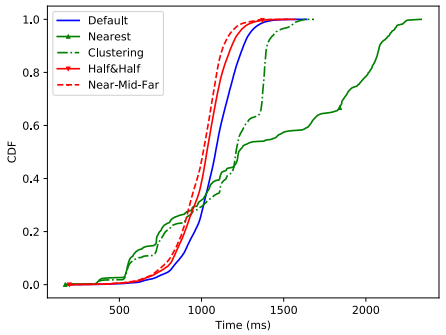


FIGURE 13. CDF of information propagation latency to all nodes in the bitcoin network using various peer selection rules.

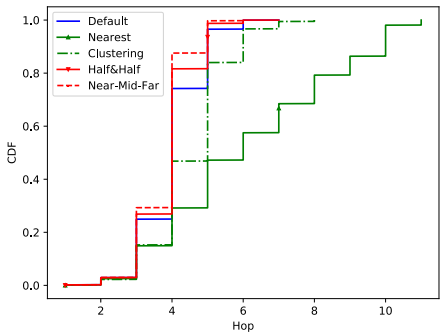


FIGURE 14. CDF of the number of forwarding hops required (by recursive flooding) for information propagation to all nodes in the bitcoin network using various peer selection rules.

network using various peer selection rules. It shows that a better peer selection rule affects the connectivity graph, reduces the network diameter in terms of number of hops, and thus results in lower information propagation latency over the network.

As a final note, our proposed approach is backward-compatible in the sense that it can be deployed incrementally and independently by individual Bitcoin clients without any global agreement or protocol changes.

VI. SURVEY OF RELATED WORK

There are several prior work that investigate and perform measurement study on the Bitcoin network from a networking perspective [5]–[7], [10], [20]–[22]. As we have declared earlier, some part of our measurements are re-doing (in 2018) of what has been done in the past (2013∼2015). Our goal is to present a comparative analysis of how the Bitcoin's peer-to-peer network has evolved over the recent years, especially after the explosive growth of Bitcoin since 2016.

In 2012, Sebicas presented 'Bitcoin P2P Network Sniffer' [20], a light-weight open source custom software written in Python that can sniff the transactions or blocks from a connected node without participating in the Bitcoin peers network. However, this software is no longer functioning because the Bitcoin protocol has been modified since then. Our BITCOIN-NODE-SCANNER implements the updated protocol using Java, and has been designed to fit our purpose of efficiently collecting peers list, version messages, and RTT measurements from all connectable nodes.

In 2013, Decker and Wattenhofer [7] studied the rate of information propagation throughout the Bitcoin network and proposed modifications to accelerate it. By establishing connections with each peer, they measured the time that blocks or transactions received take to propagate into the network. Based on the measurements of information dissemination delay, they introduced an analytical model that explains the existence of forks, matched it with their observations, and analyzed that propagation delay is the primary cause for blockchain forks. To mitigate the problem, they proposed a few changes to the protocol to speed up the propagation; (1) minimize verification, (2) pipelining block propagation, and (3) connectivity increase to reduce the distance between the origin of a transaction or a block and other nodes. This work identified ∼16000 unique IP addresses among which ∼3500 was reachable.

In 2014, Donet *et al.* [9] measured data on Bitcoin network to identify a list of 872,648 different IP addresses known to run Bitcoin node. They used a client called 'BTCdoNET', a modified version of Bitcoin P2P Network Sniffer [20], and presented information on the geographic distribution, network stability, and information propagation latency. Data collection was done in 2014 for 37 days, one scan per day, and 2 hours per scan. Their first scan on day-1 found 111,475 nodes This reference work is the reason why we chose 37 days for our data collection as well; for fair quantitative comparison.

Biryukov *et al.* [12], also in 2014, investigated a method to deanonymize Bitcoin users, which allows to link user pseudonyms to the IP addresses where the transactions are generated. Their technique uses Bitcoin address propagation messages to detect peer links based on the idea that each client can be uniquely identified by a set of nodes it connects to, even if behind NAT. They exploited the fact that, until at that time, it was possible to guess the connections of a peer just by retrieving several times the known peers list and sorting all the records in chronological order. This work reported ∼100,000 IP addresses (clients) among which ∼8000 where reachable (full) nodes.

In 2015, Miller *et al.* [6] developed a system called 'CoinScope' using similar address scanning approach, and proposed a technique called 'AddressProbe' to infer peer-to-peer links in Bitcoin network based on the characteristics of timestamps. Discovering connections was possible because each client kept updated timestamp of a peer. They also analyzed the inferred topology to discover high-degree nodes and influential nodes in the topology. They found that there are nodes that have an order of magnitude more connections than the default maximum 125 connections, implying existence of mining pools or some 'influential' nodes. Data collection was done 2014 for 18 days, but there was no statement regarding the exact number of IP addresses collected. This was a notable work which revealed possible vulnerability in the Bitcoin clients. For this reason, however, the Bitcoin Core client software has been updated to avoid such inference and possible attacks.

In 2016, Pappalardo *et al.* [5], [10] investigated the dynamics of blocks and transactions by monitoring the Bitcoin network using a custom client written in 'Go' language which uses similar address scanning approach. Data collection was done in May 2016 for 7 days, and the authors observed over 12000 unique peers (among which 8969 on IPv4 network) with an average of ∼6000 peers participating to the Bitcoin network at the same time. Based on their analysis of collected transactions and block messages, they report that sizable fraction of transactions is not processed in a timely manner.

There are also a few prior work that attempts to tackle the problem of information propagation delay in the Bitcoin network [7], [21], [24], one of which we have already mentioned above (Decker *et al.* [7])

In 2016, Fadhil *et al.* [21] collected the number of reachable nodes in the Bitcoin network for one week, and found 6430 peers and 313,676 client IP addresses. Based on the collected data, the authors proposed BCBSN (Bitcoin clustering based super node) mechanism in an attempt to improve propagation delay in the Bitcoin network through geographical clustering. In BCBSN, nodes geographically close to each other form a cluster, and one node within the cluster is assigned the role of cluster head, called 'super peer', which maintains the cluster and manages data propagation across clusters. Non-super peer nodes propagate information within the cluster only, and only the super peers propagate information to other distant super peers. The key objective is to reduce the churn of the Bitcoin overlay network. However, it is unclear how the boundaries of proximity is defined and how many super peers (thus clusters) should exist over the globe. Furthermore, BCBSN introduces several new messages and rules for super peer selection and maintenance that are not backward compatible: the proposal cannot be adopted gradually by Bitcoin peers. Moreover, the super peers need to maintain states regarding how long each node has been online and how much Bitcoins are burned by each node. This may introduce high overhead to the super peers, and there is no clear incentive / reward for the role.

To improve upon BCBSN, the same authors proposed another locality based clustering (LBC) protocol [24] which also selects peers based on proximity to form clusters, but instead of using cluster heads, LBC takes a distributed approach where each node uses local rule to determine which cluster to join. However, LBC uses a fixed distance threshold which is not robust to distance and density variations (what if no node within 100km?), and there is no explanation on how to select distant peers. On the contrary, our proposal is extremely simple, light-weight, and backward compatible. Although the idea of attempting to reduce the number of propagation hops is similar, our proposal has no additional messages nor states, and only a simple local rule for selecting connections can be adopted gradually by Bitcoin peers.

We have listed so far the prior work that are most related to ours; investigating nodes participating in the Bitcoin network from a networking perspective by scanning through the network and collecting IP addresses. There are several other threads of research on Bitcoin. For example, there are a body of work on the privacy and anonymity of Bitcoin users [26], [27], analyzing the transaction graphs [8], investigating double spending problem [13], [28], and on the attacks and security of Bitcoin system [14], [29]–[32]. Those line of work are orthogonal to ours.

VII. CONCLUSION

This work presented a comparative measurement study of nodes in the Bitcoin network by scanning the live Bitcoin network for 37 days in 2018 and compare them with data reported by prior work in 2013∼2016. Our measurements have shown that there are approximately 1 million users in the Bitcoin networks, but only around 8500∼23000 are full-node peers that participate in information propagation. The numbers have increased, but at a much slower rate than the increase in number of transactions or value of Bitcoin. United States has the most number of users followed by Germany and China, and the rank–ratio relationship of the country distribution follows a power-law, similar to the Zipf's distribution. Furthermore, the protocol version and client agent type are evolving and converging. In addition, based on the observations made from the measurement study, we proposed a simple distance-based peer selection rule for improved connectivity and faster data propagation. Evaluation results show that our proposed light-weight and backward-compatible peer selection rule has potential to reduce data dissemination latency. For future work, we plan to design and implement an open-source library for network simulator NS-3 that can more accurately simulate the Bitcoin network.

REFERENCES

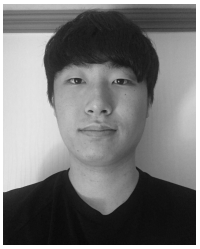
[1] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] W. Dai. (1998). *B-Money*. [Online]. Available: http://www.weidai.com/bmoney.txt

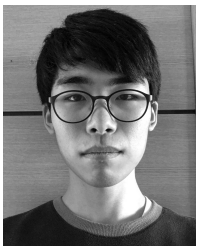
[3] N. Szabo. (Dec. 2005). *Bit Gold*. [Online]. Available: https://unenumerated.blogspot.kr/2005/12/bit-gold.html

[4] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[5] G. Pappalardo, G. Caldarelli, and T. Aste. (Jun. 30, 2016). *The Bitcoin Peers Network*. [Online]. Available: http://blockchain.cs.ucl.ac.uk/wp-content/uploads/2016/11/P2PFISY2016_paper_32.pdf
[6] A. Miller *et al.* (May 2015). *Discovering Bitcoin's Public Topology and Influential Nodes*. [Online]. Available: http://cs.umd.edu/projects/coinscope/coinscope.pdf
[7] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE 13th Int. Conf. Peer-Peer Comput. (P2P)*, Sep. 2013, pp. 1–10.
[8] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2013, pp. 6–24.
[9] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí, "The bitcoin P2P network," in *Proc. Financial Cryptogr. Data Secur. FC Workshops, BITCOIN WAHC*, vol. 16, Mar. 2014, pp. 87–102.
[10] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste, "Blockchain inefficiency in the bitcoin peers network," *CoRR*, vol. abs/1704.01414, pp. 1–15, Apr. 2017.
[11] J. Göbel and A. E. Krzesinski, "Increased block size and bitcoin blockchain dynamics," in *Proc. Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2017, pp. 1–6.
[12] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P Network," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 15–29.
[13] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Čapkun, "Misbehavior in bitcoin: A study of double-spending and accountability," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 1, pp. 2:1–2:32, May 2015.
[14] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (FAW) attacks on bitcoin," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 195–209.
[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *CoRR*, vol. abs/1311.0243, pp. 1–17, Nov. 2013.
[16] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," *CoRR*, vol. abs/1507.06183, pp. 1–31, Jul. 2015.
[17] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *CoRR*, vol. abs/1112.4980, pp. 1–50, Dec. 2011.
[18] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Mar. 2016, pp. 305–320.
[19] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 906–917.
[20] Sebicas. (2013). *Bitcoin p2p Network Sniffer*. [Online]. Available: https://github.com/sebicas/bitcoin-sniffer
[21] M. Fadhil, G. Owenson, and M. Adda, "A bitcoin model for evaluation of clustering to improve propagation delay in bitcoin network," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC), 15th Int. Symp. Distrib. Comput. Appl. Bus. Eng. (DCABES)*, Aug. 2016, pp. 468–475.
[22] S. Feld, M. Schönfeld, and M. Werner, "Analyzing the deployment of bitcoin's P2P network under an as-level perspective," *Procedia Comput. Sci.*, vol. 32, pp. 1121–1126, Jan. 2014.
[23] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998. doi: 10.1038/30918.
[24] M. Fadhil, G. Owenson, and M. Adda, "Locality based approach to improve propagation delay on the bitcoin peer-to-peer network," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 556–559.
[25] M. F. Sallal, G. Owen, and M. Adda, "Bitcoin network measurements for simulation validation and parameterisation," in *Proc. 11th Int. Netw. Conf. (INC)*, 2016, pp. 109–114.
[26] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2013, pp. 397–411.
[27] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and Privacy in Social Networks*. New York, NY, USA: Springer, 2013, pp. 197–223.
[28] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten, "Have a snack, pay with Bitcoins," in *Proc. IEEE P2P*, Sep. 2013, pp. 1–5.
[29] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 313–326.
[30] Y. Kwon, D. Kim, Y. Son, J. Choi, and Y. Kim, "Doppelganger in bitcoin mining pools: An analysis of the duplication share attack," in *Proc. Int. Workshop Inf. Secur. Appl.* Cham, Switzerland: Springer, 2016, pp. 124–135.
[31] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*, 2015, pp. 129–144.
[32] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015, pp. 692–705.



SEHYUN PARK is currently pursuing the bachelor's degree with the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea. He is also a Research Assistant with the Networked Systems Laboratory (NSL) led by Dr. J. Paek.



SEONGWON IM is currently pursuing the bachelor's degree with the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea. He is also a Research Assistant with the Networked Systems Laboratory (NSL) led by Dr. J. Paek.

YOUHWAN SEOL is currently pursuing the bachelor's degree with the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea. He is also a Research Assistant with the Networked Systems Laboratory (NSL) led by Dr. J. Paek.

JEONGYEUP PAEK received the B.S. degree in electrical engineering from Seoul National University, in 2003, and the M.S. degree in electrical engineering and the Ph.D. degree in computer science from the University of Southern California, in 2005 and 2010, respectively, where he was a member of the Networked Systems Laboratory (NSL) led by Dr. R. Govindan. He was with Deutsche Telekom Inc. R&D Labs, USA, as a Research Intern, in 2010, and then joined Cisco Systems Inc., in 2011, where he was a Technical Leader with the Internet of Things Group (IoTG), Connected Energy Networks Business Unit (CENBU, formerly the Smart Grid BU). In 2014, he was with the Department of Computer Information Communication, Hongik University, as an Assistant Professor. He has been an Assistant Professor with the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea, since 2015. He is an IEEE Senior Member.