Bitcoin P2P Network Measurements: A testbed study of the effect of peer selection on transaction propagation and confirmation times

Befekadu G. Gebraselase, Bjarne E. Helvik, Yuming Jiang
Department of Information Security and Communication Technology
NTNU, Norwegian University of Science and Technology, Trondheim, Norway
{befekadu.gebraselase, bjarne, yuming.jiang}@ntnu.no

***Abstract*—Bitcoin is the first and the most extensive decentralized electronic cryptocurrency system that uses blockchain technology. It uses a peer-to-peer (P2P) network to operate without a central authority and propagate system information such as transactions or blockchain updates. The communication between participating nodes is highly relying on the underlying network infrastructure to facilitate a platform. Understanding the impact of peer formation strategies, peer list, and delay is vital in understanding node to node communication and the system performance. Therefore, we performed an extensive study on the transaction characteristic of Bitcoin through a testbed. The analysis shows that peer selection strategies affect the transactions propagation and confirmation times. In particular, better performance, in terms of smaller transaction confirmation time and lower number of temporary forks, may be achieved by adjusting the default nearby-based peer selection strategy.**

***Index Terms*—Bitcoin, P2P, Peer selection strategies, Transaction characteristics**

I. Introduction

Bitcoin is becoming the leading cryptocurrency system today, with its value rising dramatically since its launch in 2009 [7, 30]. Satoshi Nakamoto, the pseudonym of Bitcoin's creator, stated that Bitcoin is an electronic payment system based on cryptographic proof instead of trust [30]. Bitcoin is the first well-known decentralized electronic peer-to-peer (P2P) system that uses blockchain technology [7, 30]. It adapts a cryptographic proof of work (PoW) mechanism that allows anonymous peers to create and validate transactions through the underlying P2P network [30]. The P2P network is vital to the communications of the blockchain system [10][15]. The nodes send and receive messages via the underlying network infrastructure while the P2P topology is formed at the application layer [25]. The way nodes form an overlay topology affects the overall performance, such as transactions confirmation time [40], block and transaction propagation delay [7][16], fork rate [7], and stability of the ledger. In this regard, we prepared a testbed to analyze the impact of P2P topology formation, end-to-end delay, and bandwidth limitation on the performance of Bitcoin.

Bitcoin operates to distribute the ledger among all the participants in a flooding P2P network [9]. When a node tries to join the Bitcoin network, it uses a hardcoded seed to reach out to the nodes nearby. Through getaddress and node discovery, each node updates/creates eight peers by default (outgoing connection), but it can have up to 124 inbound connections. The logical connections between participating nodes create a dense P2P overlay topology, a mesh network [10][11]. This P2P topology is responsible for broadcasting new updates to peers by which they learn and inform each other about transactions and blocks [10]. The reachability of these messages affects the ability of the system to process more transactions and secure the interactions [10][15].

In Bitcoin, the average inter-block generation time is 10 minutes. This enables all the newly generated blocks to reach the maximum number of nodes in the network. Shortening the inter-block generation time brings higher block propagation delay, which increases the temporary forks rate [37][39], which wastes the miner's resource and makes the transactions wait longer. Alternatively, increasing the inter-transaction generation also increases propagation delay, affecting the confirmation waiting time of the transaction [7]. Likewise, the peer formation strategies also impact the reachability of the transactions and blocks [2][38]. The nodes forward new updates to the peer nodes, in which the number of peer nodes and the delay in between impact the amount of time needed to forward a message. The network element´s delay, processing delay, and peer formation strategies affect the block's number of minutes to reach the maximum number of nodes.

This paper investigates network-related parameters' impact on the technology's overall performance. However, it is challenging to conduct such analysis because the nodes are independent and anonymous, making it challenging to collect measurement data from the unknown nodes. In addition, measuring a network fragment will bring results not representative of the overall performance. Several methods have been proposed in the literature to examine network condition effects on the performance. One is to use analytical models, which, however, are built on much simplified assumptions or approximations to allow tractable analysis, e.g., [28, 39], unable to reflect the actual Bitcoin P2P network environments. Another is to use simulation tools, e.g., [1, 2], which, however, have also made

much simplification and do not exactly implement / reflect the set of mechanisms used by the Bitcoin P2P network. The third is to develop an emulator that behaves like an actual Bitcoin [20]. For this reason, we choose to develop an emulator.

A testbed emulator has been prepared to perform a measurement-based study. As a highlight, the testbed includes 104 Raspberry Pis, six switches, and two-blade racks. Each blade rack can hold up to 40 Raspberry Pis. Each raspberry device has Bitcoin Core 0.21.0 [4] installation with additional scripts to automate transactions and block generation events. Through this testbed, a dataset has been gathered containing primary information about the chain, i.e., the ledger, and information that is not available from the ledger but measured from the local mining pool (mempool). Based on the collected dataset, an explorative study on the transaction characteristics of Bitcoin has been conducted.

This paper investigates the effect of peer selection on transaction propagation and confirmation times. In particular, the aim is to provide valuable insights into the impact of network conditions on transactions confirmation time. The paper's main contributions are the following:

- The paper presents a testbed development that can be used to examine the transaction characteristics of Bitcoin, including transaction propagation and confirmation times.
- The investigation shows that peer selection has substantial impact on the performance. In particular, adding random peer selection can improve the transaction propagation and confirmation times.
- It is also found that some transactions, particularly when the load is high, need to wait much longer than the expected 3600 seconds to get confirmed, and the occurrence of temporary forks, in addition to load, also contributes to this.

The rest of the paper is organized as follows. The current state of the art is covered in Section II. Next, Section III illustrates the testebd setup and what kind of parameters considered. Then, Section IV illustrates the workflow of transaction handling in Bitcoin. Following that, how P2P topology formation and the strategies proposed are discussed in Section VI. Next, Section IX and X reports results gained from the analysis. Following that, Section XI presents the impact of fork occurrence over the transaction confirmation time. Section XII opens up a discussion on what has been observed in the analysis. Finally, Section XIII concludes the paper.

II. Related Work

There are several works related to studying the impact of bitcoin P2P on the security and performance of the technology. Eisenbarth et al. [10] analyzed examining the resilience of bitcoin networks from churn, detection of Sybil nodes, dynamicity, and popularity of peers. Based on one month of observation, the study showed little churn in the network, no Sybil attack, and recent updates on tackling these issues had become effective. Wang et al. [42] developed an Ethereum network analyzer, Ethna, to analyze the P2P network. The analysis showed that the average degree of an Etherium node is 47, and the P2P network of blockchain such as Bitcoin degree of distribution follows a power-law. The network has the characteristics of a scale-free network.

Fadhil et al. [15] proposed locality-based approaches to improve the propagation delay on the P2P network. This study considered clustering nodes in the exact geographical location, where the distance between is used as key on choosing which nodes to add as a peer. They showed that providing a less distance threshold would improve the transaction propagation delay with a high proportion. However, clustering with known deterministic distance may reduce the security of the network. Essaid et al. [11] proposed a Bitcoin P2P topology discovery framework that tracks the information exchange to discover network topologies. Based on 45 days' observation, the node distribution between the USA and China matches closely, while other parts of the world have fewer active public nodes to discover. Sudhan et al. [41] developed a model to simulate the Bitcoin network and studied the impact of the outgoing connection limit over the transaction propagation time. In addition, the study considered two peer selection strategies proximity and random. They showed that peer selection strategies impact the transaction propagation delay.

Shahsavari et al. [39] proposed an analytical model to study the network delay and traffic delay in Bitcoin. The study considered the effect of the default number of connections and the block size on the performance of the Bitcoin network. Deshpande et al. [8] developed a fast and efficient framework named BTCmap to discover and map the Bitcoin network topology. The analysis indicates that the online peers' list remains valid (less than 1% of changes) at 56 minutes 40 seconds. Otsuki et al. [33] showed that a relay network improves the propagation time of a block. In addition, the work showed that relay network decrease in the orphan block rate and the 50th percentile of block propagation time. However, the relay network's improvement of the orphan block rate became smaller as the Internet speed increased. Regarding the mining success rate, it was demonstrated that the relay network did not significantly influence the differences between utilizing and non-utilizing nodes were below 0.1 at any utilization rate.

The authors in [22] proposed KadRTT, an approach that tries to reduce the lookup latency and hop count. The study shows that proximity and uniform ID arrangement methods enable the proposal to improve performance. This improves P2P applications for efficient content lookup mechanisms.

Most of the research work outlined above analyzes the discovery of Bitcoin P2P topology or develops a framework to crawl the live Bitcoin network to discover the structure and security bridges of the technology. However, little has been investigated about the impact of peer selection, topology formation, and end-to-end delay on the transaction characteristics of Bitcoin. Therefore, we developed a testbed that mimics the real Bitcoin network, enabling us to experiment with collected data and make further analyses.

III. Measurement Setup and Node Configuration

For the measurement study, a testbed has been implemented to record information about Bitcoin transactions. The testbed includes 104 Raspberry Pis, six switches, two-blade rack (each holding 40 Raspberry Pis). Each Raspberry Pi has an installation of a full Bitcoin core.

A. Node configuration

Every Raspberry Pi is used as a full node that participates in addition, validation, and generating valid logs. These devices boot from an SD card. The SD card has Ubuntu Server Version 20.10 for the ARM architecture. In addition, the SD cards contain the scripts necessary to run the setup, for instance, scripts to start Bitcoin daemon, adding topology and delay and generating transactions and blocks.

1) Network configuration: Each node interface is configured with an IP address 192.168.xx.1/24. Subnetting with /24 may not be necessary to have a single node, but we plan to increase nodes per subnet for the future use case. Assigning such an IP address also mimics an actual Bitcoin node with its public address. Since each node becomes part of the network, we used VPP (Version 21.6) to perform routing between the nodes. It is an open-source software that provides high-performance switching and routing features for commodity hardware [43].

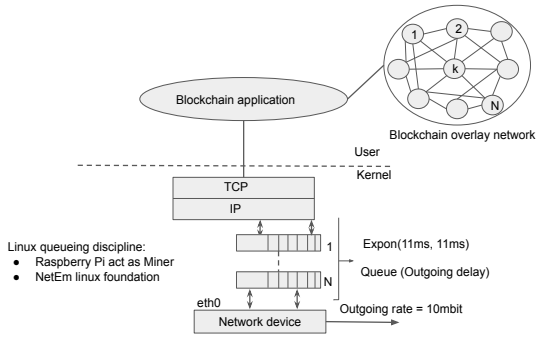


Figure 1. Bitcoin node configuration

The basic architecture of Linux queuing disciplines is shown in Fig. 1. The queuing disciplines exist between the protocol output and the network device, and the default queuing discipline is a simple packet FIFO queue. A queuing discipline is a simple object with two key interfaces. One queues packets to be sent, and the other releases packets to the network device. The queuing discipline makes the policy decision of which packets to send based on the current settings. As shown in Fig. 1, the packet leaving each node adds delay to each packet which follows an exponential distribution. Since each node has an N peer list, we can also see N queues. In addition, the bandwidth is limited to 10 Mbps capacity. These configurations mimic the real Bitcoin network's peer list, and delay arises from the node and network capacity limitations. To simulate a network of the whole Bitcoin network, we used NetEM. It provides Network Emulation functionality for testing protocols by emulating the properties of wide-area networks [19].

To emulate network traffic, the NetEm emulator provides Normal and Pareto distributions [19, 21]. However, the literature study, e.g., [17], has revealed that the inter-packet delay in Bitcoin follows more closely an exponential distribution. This is another challenge since the NetEm does not provide this distribution but allows users to add their distribution. There are different ways to prepare a user-defined distribution. For instance, extracting the RTT values from ping statistics gives the mean and standard deviation, then using it in the NetEm command when activating the distribution table produced. This is easy to do between a few nodes. Our setup mimics the actual Bitcoin network of 5670-7279 active full nodes [9][11]. The Bitcoin documentation states that a node chooses a peer within shorter latency. We generated random variables by inverse transform sampling of exponential distribution based on this fact and then used iproute2 marketable to create an exponential distribution. We set the delay (d) between 11 ms, and it is a shorter end-to-end delay to add nodes. This 11 ms is extracted from an independent full Bitcoin node [16], where we calculated the delay between the eight peers from this node and took the minimum delay between the node and its peer which was 11 ms.

2) Node to node delay: In the previous subsection, we discussed why NetEM is used to add delay and bandwidth limitation to emulate the underlying wide area network (WAN) of Bitcoin. This section shows how independent nodes communicate with each other through an open-source software router Vector Packet Processing (VPP) [43]. Nodes add delay d to each outgoing packet. The outgoing packet passes through the router and reaches the destination. Fig. 2 illustrates node to node communication delay between Node$_i$ and Node$_k$ while the Dell computer is used as a router. The VPP open source software router is configured in Dell OPTIPLEX 9020, with a specification of Intel® 4th generation Core™ i7/i5 Quad Core, Ubuntu 20.04, 32GB memory, Integrated Intel® I217LM Ethernet LAN 10/100/1000, and 256GB storage capacity.

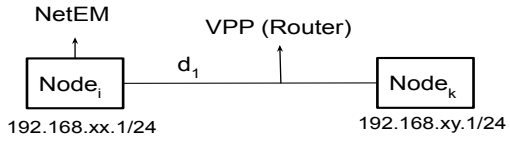


Figure 2. Node adding delay

B. Time synchronization

The devices have to be time-synchronized to enable accurate time stamping by each node in the network. For this reason, we used a well-known time synchronization application called Network Time Protocol (NTP). NTP is an application that allows computers to coordinate their system time [26, 36]. The implementation is in userspace rather than in kernel mode; however, its performance is much better than the other network time protocols [36]. Usually, it is available for most

Linux distributions, which makes it easier to integrate with applications. We have 104 nodes that generate events that require accurate timing and synchronization. Therefore, we used NTP in our setup, where, node 1 acts as an NTP server, while the rest 103 nodes act as a client. The nodes synchronize time means to set them to agree at a particular epoch with respect to coordinated universal time (UTC) [26]. Fig. 3 shows how NTP is added to the setup. As we can see from Fig. 3, node 1 is the NTP server, while the rest 103 nodes are the NTP clients.

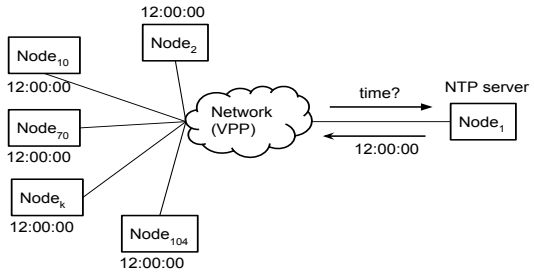


Figure 3. Time synchronization

C. Raspberry Pi specification

The Raspberry Pi devices are running the Bitcoin protocol through Bitcoin Core 0.21.0. To identify them, each device is given a unique number from 1 to 104. These devices act as full nodes, and a single device will be referred to as *node n* where n is the given number. As we see from Table I, in total, the setup has 93 "Raspberry Pi 3´´ devices and 11 "Raspberry Pi 4´´ devices. There are some differences between Raspberry Pi 3 and Pi 4 that are relevant for the setup. Raspberry Pi 4 Plus has a CPU clock speed of 1.5 GHz, 0.1 GHz more than Raspberry Pi 3, which has a clock speed of 1.4 GHz. Additionally, while Raspberry Pi 3 has an Ethernet port with a maximum throughput of 300 mbps, Raspberry Pi 4 has Gigabit Ethernet.

Table I
RASPBERRY PI MODELS

	Raspberry Pi 3 Model B+	Raspberry Pi 4
Processor	1.4 GHz	1.5 GHz, 64 bit CPU
Memory	1GB RAM	1-4GB RAM
WiFi	2.4GHz Wireless LAN	2.4Ghz and 5Ghz Wireless
Ethernet	300Mbps	Gigabit Ethernet
SD card	8-16 GB	8-16 GB
# nodes	93	11

IV. THE WORK FLOW OF BITCOIN

This section gives essential background on how Bitcoin handles transactions, in addition to how the nodes communicate and discover each other.

1) Workflow: Fig.4 illustrates the workflow of transaction arrival, block formation, propagation, and validation in Blockchain. Briefly, after transactions are generated by the users, they are sent to all full (validation) nodes. Upon the

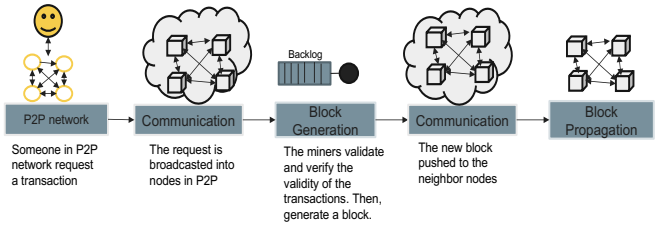


Figure 4. Blockchain process flow

arrival of a transaction at a full node, the node stores the transaction in its backlog (memory pool), waiting for confirmation. Besides, the node may choose unconfirmed transactions in the backlog to pack into a new transaction block. If the puzzle finding is successful, this newly generated block is added to the Blockchain. This information is sent to all the nodes. At each node, the validity of the newly generated block is checked. If the validity is confirmed with consensus, the updated Blockchain is accepted, and the new block of transactions are validated. Such validated transactions are removed from the mempool at each full node that then repeats the above process.

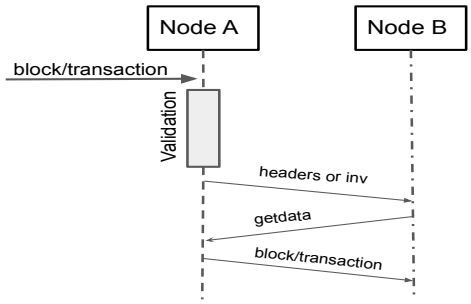


Figure 5. Legacy relaying

2) Node to node interaction: Bitcoin nodes form a P2P network, while each node by default can have eight peer list. It is a logical link that allows peers to push/pull new updates to the neighbors. Fig. 5 shows node to node message exchange sequence. The new arrival block or transaction picked up by Node A. Then, a block/transaction is validated (the grey bar) by Node A, which then sends an inv message to Node B requesting permission to send the block. Node B replies with a request (getdata) for the block/transaction, and Node A sends it.

3) Network discovery: A Bitcoin node is allowed to maintain up to 132 connections (maxconnections) as default, of which 8 are outgoing connections and the rest are incoming connections. Peers listen on port 8333 for inbound connections. When a node wants to join the network, as it is a public blockchain, the node uses DNS names (called DNS seeds) hardcoded in the Bitcoin Core. From this point, the new node updates its peer list by discovering nodes nearby. In this way, new nodes select peers that are part of the network. This peer formation is called nearby-based since it highly depends

on adding nearby nodes. The peers randomly choose logical neighbors without knowledge about the underlying physical topology.

The peer list used as a reference list to send an inventory or receive messages from the neighbor nodes. After the node joins the network, it can take part in propagation, consensus, and block generation. These nodes act as a full node, which means the users/owner can create a transaction and create a block, and forward the new updates to the network. Each block created by the nodes that are valid enough to be included in the chain will contain the hash of previous records of the blocks. Blocks that are created but ignored by the network become orphan blocks. Mostly these blocks become fragments that will never be used but waste all the computation cost and resources.

4) Peer list: Nodes can have up to 132 connection lists. This is the combination of incoming and outgoing peers. When a node initiates the connection, it is called outgoing, or if the connection initialization comes from other nodes, it is incoming bound. The number of peers (P) represents the number of outgoing peers of each node. The total connection list is the sum of P outgoing peers plus incoming peers (Q). In this work, the peer list length (p_l) is set to be $2P$, i.e. $Q = P$.

V. Setup of Input Parameters

This section describes the input parameters such as inter-transaction generation time, inter-block generation time, and node to node delay added to the network.

The transaction and block generation events must also include similar characteristics to mimic the Bitcoin network. The transaction inter-arrival time to a node follows an exponential distribution, based on the literature investigation [17][40]. Similarly, the inter-block generation time also follows an exponential distribution [17][18].

Algorithm 1 Generate transaction

```
1: procedure POISSON(λ(t), T_d)
2:     Initialisation: T_t = timenow() + T_d
3:     Condition: T_d ≤ T_t
4:     while True do
5:         w_t ∼ negExp(λ(t))
6:         if timenow() + w_t < T_t then
7:             time.sleep(w_t)
8:             generateTransaction()
```

1) Transaction inter-generation time: Each node acts as a full Bitcoin node that creates, validates, and propagates transactions and blocks. Therefore, nodes have a script that generates transactions and blocks following an exponential distribution. The script accepts duration and the inter-generation interval in terms of seconds as an input parameter, as illustrated in algorithm 1. $1/\lambda(t)$ is the mean inter-generation time (t_{gi+1} - t_{gi}) in seconds for each node. Furthermore, T_d is the total duration of running time in seconds. The result of the inter-generation time distribution follows an exponential distribution.

2) Block inter-generation time: Bitcoin network generates a block on average 10 minutes. This makes the recent block propagate to the network before the next generation. Bitcoin adjusts the difficulty after 2016 blocks are generated to control the average inter-block generation time. Although this is true for live Bitcoin nodes, the Bitcoin core regtest mode has difficulty close to zero, which means there is no difficulty generating a block. However, to mimic the real Bitcoin network, we developed a script that produces a block on average ten minutes. Overall, we have 104 nodes, which means a block is generated in 103*600 second (61800), the remaining 1 node is measurement node. Similar to the previous transaction generation case, here the Algorithm 2 takes the generation rate and duration of the simulation in seconds as an input.

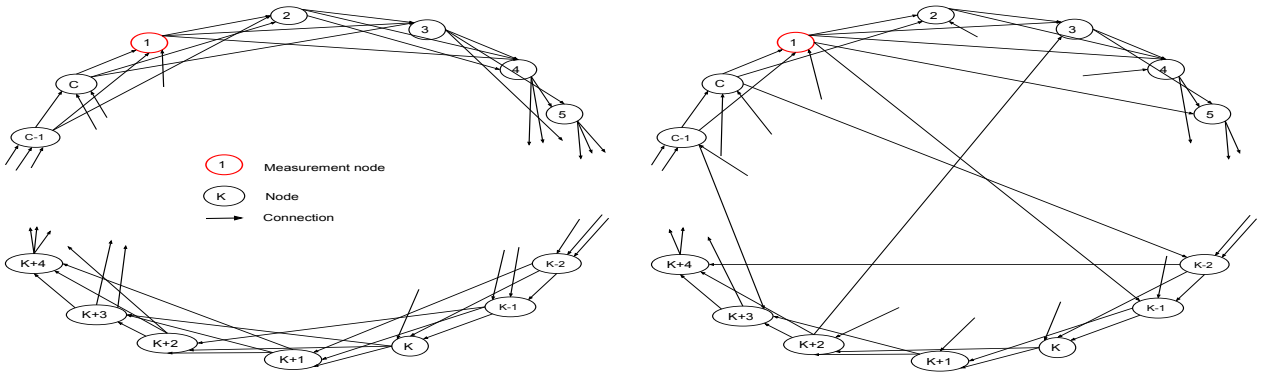
Algorithm 2 Generate Block

```
1: procedure POISSON(λ(t), T_d)
2:     Initialisation: T_t = timenow() + T_d
3:     Condition: T_d ≤ T_t
4:     while True do
5:         w_t ∼ negExp(λ(t))
6:         if timenow() + w_t < T_t then
7:             time.sleep(w_t)
8:             generateBlock()
```

3) Node-to-node delay: In the actual Bitcoin network, nodes are distributed across the globe, which are geographically and domain-wise isolated from each other. Since the underlying network infrastructure is providing the communication platform and the actual network traffic is unpredictable, it is common to consider a distribution that captures the network delay between two participating ends. To mimic the delay that arises from the network element and distance between the participating nodes we introduced a delay (d) that follows an exponential distribution with the shorter mean of 11 ms.

VI. Network Topology

The Bitcoin research community states that peer formation starts from looking at DNS seed nodes. Some of those DNS seeds provide a static list of IP addresses of stable Bitcoin listening nodes. Once a peer receives a full Bitcoin node IP address list, the peer performs up to eight outgoing connection attempts. These eight nodes that the peer attempts connection with are called entry nodes. A node can request from its neighbors the IP addresses of peers they are aware of using the addr P2P network message and increasing the nodes' awareness nearby. The distance between nodes, such as delay, the number of peers, and how to select the peer affect the overall performance. We consider three peer formation cases to investigate this impact: nearby, random, and mixed. The nearby-based approach is a method that adds neighbor nodes, as stated in the Bitcoin documentation [5]. The other way is randomly selecting peers, as expressed by other authors [7, 41]. Finally, to mix these two approaches to investigate the effect, this method is a mixed approach. The following section introduces these approaches.



(a) Nearby-based approach

(b) Mixed-based approach

Figure 6. Bitcoin overlay network example (P=3), while considering only outgoing links

A. Nearby-based peer selection

Nearby-based peer selection approach enables peers to form close by neighbor peer creating P2P topology. A full Bitcoin node can have eight peers by default, but it can have up to 132 connection link points. The nearby metric depends on adding nodes close by.

Algorithm 3 Nearby-based

1: **procedure** NEARBY(P, k, C)
2: p_l= {l—l= (k+i) mod C, i=1, ...,P}

Algorithm 3 illustrates the nearby-based peer selection method. The procedure takes the number of peers to add (P), the current node (k), and the total number of nodes (C). The algorithm adds peers that are closeby.

B. Random-based peer selection

Unlike the nearby-based approach, the random-based method does not depend on the proximity of nodes, instead on the random selection of the peer to add. Even-though Bitcoin is a distributed P2P technology where each node acts as an independent node, it has little knowledge on the global distribution of the nodes. For the random-peer selection method, we consider that nodes know the number of full active nodes in the network they are participating in. Similar to the nearby-based approach, Algorithm 4 illustrates the random peer selection method. The procedure takes the number of peers to add (P), the current node (k), and the total number of nodes (C). The method adds randomly selected nodes as its peer list.

Algorithm 4 Random-based

1: **procedure** RANDOM(P, k, C)
2: Initialisation: $p_l = \{\}, p_c = \{1, \ldots, C\} \setminus \{k\}$
3: **for** $i = 1$ step 1 until P **do**
4: $p_l \leftarrow p_l \cup (\text{RANDOM}(p_c \setminus p_l))$

C. Nearby + Random (Mixed)-based peer selection

The third case is to combine nearby-based and random-based approaches. In these combinations, the nearby-based method adds $n-1$ peers and the random-based approach adds the last node by choosing randomly. This is to introduce a random link to the nearby-based peer list. Similar to the previous two approaches, Algorithm 5 illustrates the mixed peer selection method. The procedure takes the number of peers to add (P), the current node (k), the total number of nodes (C). As discussed in the previous subsections, the method adds the $n-1$ nodes based on the nearby-based approach. The random-based approach adds the last node.

Algorithm 5 Mixed-approach

1: **procedure** RANDOM(P, k, C)
2: $p_l = \{l | l = (k+i)\ mod\ C, i = 1, \ldots, P-1\}$
3: $p_c = \{1, \ldots, C\} \setminus \{k\} \setminus p_l$
4: $p_l \leftarrow p_l \cup$RANDOM(p_c)

From this point on forwarding, we use random to represent the random-based approach, normal for the nearby-based default approach, and mixed for the approach that mixes the two approaches.

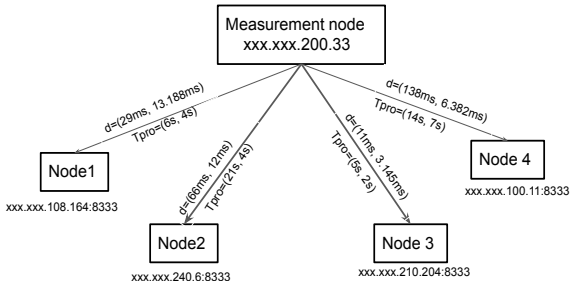
VII. Setup Validation

This section relates the timings in the testbed with those in the live Bitcoin network.

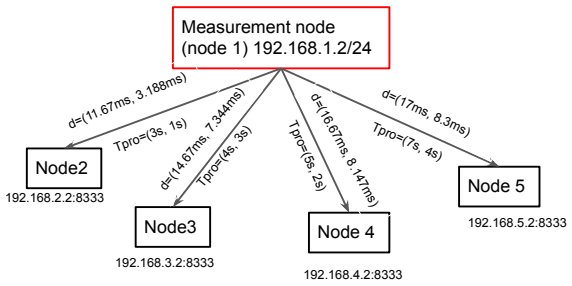
A. Node to node delay

In our previous work [16], an independent Bitcoin full node was deployed to collect transactions and block related feature sets. We used observations from this node to validate some of the input parameters and results. For instance, our nodes have 132 connected nodes. Eight of these nodes are peer nodes, while the rest are incoming bound nodes. The average ping delay between these nodes from the Bitcoin application is 156.20 ms with a standard deviation of 152.23 ms. This ping is handled in a queue with other commands in the application layer to include the processing backlog. However, we also conducted further analysis to ping these nodes from outside of the Bitcoin core, which resulted in an average of 80 ms second in deference. This 80 ms accounts for processing backlog.

As previously mentioned, the eight peers are more important than the others. These peer nodes synchronize more often



(a) Bitcoin live full node



(b) Testbed

Figure 7. Transaction propagation delay between active full nodes, where $\lambda = 3$ transactions per second per network, eight outgoing peers per node

than the other 124 incoming bound nodes. For this purpose, we conducted an independent investigation to see the delay between our node to eight peer nodes. Our analysis shows that the minimum delay between our node and the other nodes is 11ms with a standard deviation of 7ms. This 11 ms delay is used in our setup as a minimum delay guarantee between nodes.

B. Information propagation

This subsection discusses how fast a transaction propagates in the Bitcoin network and how the number of nodes impacts this. We considered four publicly available nodes to collect mempool state and compare it with our node. Fig. 7(a) shows the delay between our measurement node in [16] with four peer nodes that provide their state of the mempool in the Bitcoin network. The figure shows only four out of eight nodes because the remaining four nodes were unreachable. This is because some of them are hidden behind firewall and NAT. As we can see from the figure, transaction propagation between nodes can be up to about 20 seconds [14]. This is mainly because the P2P communication protocol makes processing check the validation of each transaction before forwarding an Inv message to its peers. At the same time, nodes that received the Inv message have to check if the transaction is at the mempool or seen before inside a block. The node sends a getdata message and gets the new transaction when the check is completed. Even though the delay between nodes may be less than 100 ms, processing a transaction takes longer.

We tested out the testbed based on the live Bitcoin full node observation to see if similar transaction characteristics occurred. As we can see from Fig. 7(b), the transaction propagation delay between the measurement node, Node 1, and its four peers also varies in the same order. This demonstrates that the timings in the testbed are similar to those in the real Bitcoin network.

A closer check at the mempool status of the four nodes on the Bitcoin network shows that the number of transactions waiting in their mempool varies between peers in an instant of time. A similar observation is found in our emulated network. For instance, each node shown in Fig. 7(b) respectively has 1566, 3976, 3000, 2244, and 2300 transactions waiting at the mempool at one checking time instant

C. Inter-block generation and inter-transaction arrival time

The average inter-block generation time is close to 10 minutes in the actual Bitcoin network. After 2016 blocks are generated, the difficulty of solving the puzzles increases to make sure nodes generate on an average of 10 minutes so that the new block reaches the maximum number of nodes in the network. Similarly, the transaction inter-arrival time to the mempool also follows exponential distribution [16][17]. These parameters are considered in our setup as input parameters.

VIII. Measurement Data Collection

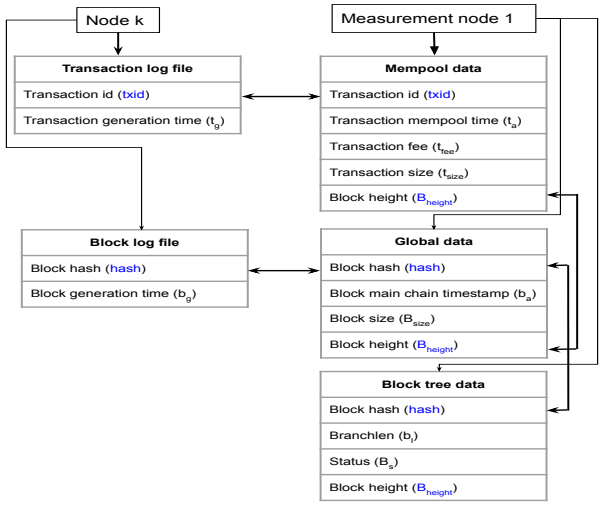


Figure 8. Data collection

A dataset consisting of four parts has been collected by the testbed, as show in Fig. 8. One part of the dataset records each node transaction and block generation events. When a node k, where $k \in [2, 104]$, generates transactions, it records a log about the transaction generation time (t_g) and transactions id ($txid$). Similarly, when a node generates a block, it records the block generation time(b_g) and block hash ($hash$). The second part of the data contains information about each transaction's arrival time at mempool (t_a), transaction size (t_{size}), transaction fee (t_{fee}), transaction id ($txid$), and block height (B_{height}). The block height (B_{height}) in which the transaction belongs can be empty or a number depending on if the transaction is added to the block or just a new arrival. The third part collects information about the blocks from the main chain, such as block hash ($hash$), block size (b_{size}),

block time (b_t), and block height (B_{height}). The fourth part of the data collection contains extracted details about the block tree of the chain, such as Block hash ($hash$), Block height (B_{height}), Branchlen (B_l), and Status (B_s). The Branchlen is the length of the branch in the block tree. It holds 0 for the main chain or a number, indicating the length of the soft fork in terms of the number of blocks in the side chain. The Status (B_s) indicates the Status of the block, whether it is active, part of the main chain, or valid-fork meaning a block is a fork or invalid-block meaning the block is not valid enough to be a candidate.

The second, third and fourth parts of the data are collected from a single node. This node is considered as a measurement node, and in our case, Node 1 is the measurement node. Node 1 is part of the network invalidation and processing transaction at the mempool, but it does not generate transactions or blocks. On the contrary, it collects information about the transactions from its mempool (Mempool data). When the emulation times are over, it also extract information about valid blocks from the main chain (Global data, Block tree data).

Fig. 8 demonstrates the collected feature set from the nodes. As we can see from the figure, measurement Node 1 collects information about the state of the mempool and keeps track of the status of the main chain. It also illustrates the primary key used to link the data set from each device with Node 1. By using the datasets, we performed analysis on transaction propagation ($t_a - t_g$) time and confirmation time ($b_{g(i+6)}$ - $t_g(x)$), where transaction x goes into block i and $i + 6$, representing when the transaction is six-block deep into the main chain.

In addition to the above-collected information, we also extracted the state of the block tree. This information includes which block is fork ($hash$), at which height this event happened (B_{height}), and the number of blocks within the same branch ($B_{branchlen}$). We used these extracted feature sets to count the number of forks that happened while considering different peer formation strategies and how they impact the confirmation time of transactions inside a fork block. These datasets are downloaded and post-processed after the emulation period is completed.

IX. Transaction Propagation time

The transaction arrival intensity affects the number of transactions waiting at mempool and the number of transactions to be validated and pushed to the network. This section reports results and observations from examining the impact of arrival intensity variation while illustrating the effect of peer list per node.

Bitcoin uses a gossip-like protocol to broadcast updates throughout the network [11]. When a node receives new transactions, it validates and verifies the validity of the transactions, then sends an Inv message to peer nodes to notify them if the peer nodes want these new transactions, before pushing the transaction to the peers. Due to this continuous process, a delay in transaction propagation occurs. The delay combines validation time and the time it takes to disseminate

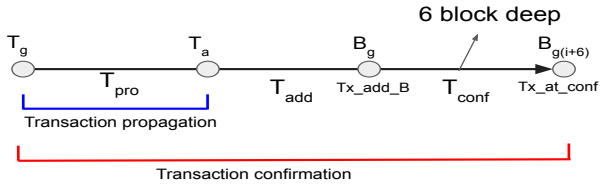
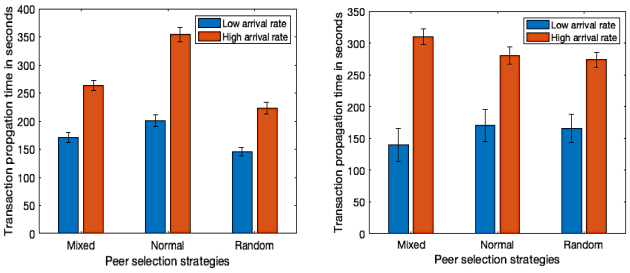


Figure 9. Transaction propagation and confirmation time sequence

the transaction. Fig. 9 shows a time sequence of the life cycle of a transaction. In this section, we focus on the transaction's propagation, and this is the typical time when the transaction is generated (t_g) until it reaches the memppol of a node. Specifically, in our case, the time difference between t_g and t_a is the propagation time, where t_a is the time transaction arrived at the memppol of the measurement node Node 1, and t_g is the time of the transaction generated by one of the nodes (2-104). As illustrated in Fig. 9, the blue line indicates the time length of transaction propagation time. The transaction propagation delay is less than 8 seconds for the default and low-intensity eight peer node case, (see Fig. 7(a)).

1) Average transaction propagation time: Fig. 10 shows the average transaction propagation time in seconds while considering different peer formation strategies. The x-axis represents peer selection strategies, the y-axis represents the propagation delay in seconds, and the legend shows the arrival intensity.



(a) $P = 8$ (b) $P = 4$

Figure 10. Average transaction propagation times for the various peer formation strategies, number of peers P and low (3 t/min) and high (6 t/min) intensity generation rate λ. Error bars indicate 95% confidence intervals from 10 independent runs

Fig. 10(a) and 10(b) illustrate that when the arrival rate is high, which means each node generates on average six transactions per second, in respective of the number of peers per node, the transaction propagation increases. However, with a low arrival rate, three transactions per minute per node, the transaction propagation is less than 170 seconds. In addition, when the number of peers is higher, the normal approach tends to perform worse overall, while random-based peer selection better than the other two.

2) Distribution of transaction propagation time: Fig. 11 and 12 illustrate the distribution of transaction propagation time under a low and a high arrival rate respectively, while the number of peers is fixed to eight. The x-axis represents the propagation time in seconds. The y-axis is the log result of

the distribution $P(t_a - t_g > t)$, while the three peer formation strategies are used.

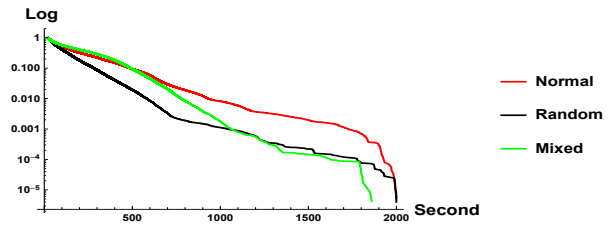


Figure 11. Transaction propagation delay, where $\lambda = 3$ transactions per minute per each node (low intensity)

Fig. 11 shows that, in most cases (80%), transaction propagation in random peer selection has less than 300 seconds propagation time, whereas it has 400 seconds during the mixed approach, while for normal peer selection transactions observe close to 500 seconds propagation time. In all three peer selection approaches, the transaction propagation time can grow more than 1000 seconds in 1% of the cases. Relatively, 90% of the transactions observe propagation time less than 500 seconds for mixed and normal approaches, while random-based peer formation brings less than 450 seconds of propagation time.

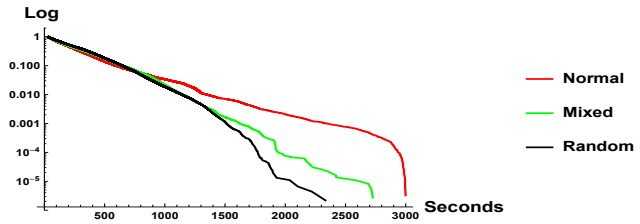


Figure 12. Transaction propagation delay, where $\lambda = 6$ transactions per minute per each node (high intensity)

Fig. 12 also illustrates the transaction propagation delay distribution where the three peer selection approaches are considered. In most cases (80%), the figure reports that transaction propagation in random peer selection has less than 400 seconds propagation time, whereas it has 500 seconds with the mixed and normal peer selection approaches. In all three approaches, the transaction propagation time can grow more than 1500 seconds in 1% of the cases. Relatively, 90% of the transactions see propagation time less than 700 seconds for mixed and normal approaches, while random-based peer formation brings less than 600 seconds of propagation time.

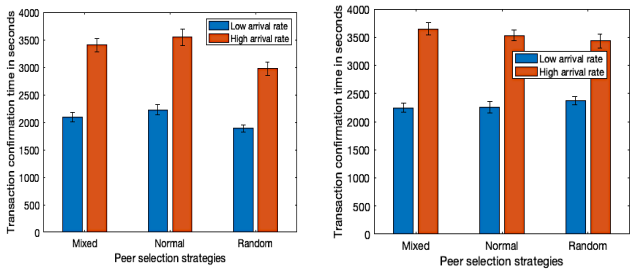
In summary, when the arrival intensity is low, the random-based peer selection strategy performs better, but when the intensity is high, all three strategies produce a comparable propagation delay. This shows arrival intensity has more significant impact on the propagation delay than the type of strategies used or the number of peers.

X. Transaction Confirmation Time

In Bitcoin, the transaction is considered confirmed six blocks deep in the main chain. This tries to ensure no double-spending while maintaining security: By linking the previous block with the other six blocks, it requires more computational effort to modify the confirmed transactions. Thus, to improve security, Bitcoin reduces its performance. This section examines how the arrival intensity, peer list, and end-to-end delay affect performance. In the 'regtest' setup, the transaction is considered valid when it is 101 blocks deep. However, our analysis used six blocks deep for confirmation to obtain results representative of the live Bitcoin blockchain.

Fig. 9 also demonstrates the time sequence of transaction confirmation time. The transaction confirmation time is the difference of the t_g and the t_{conf}. The t_{conf} is the amount of time for the Bitcoin network to generate six valid blocks. Similar to the previous case, t_g is the time a transaction is generated by one of the nodes, and t_{conf} is the time between the blocks from the main chain extracted at node 1. As shown in Fig. 9 the red line indicates the time sequence of the transaction confirmation time. A transaction has to wait until it is six blocks deep. Since a new block is generated every 10 minutes or 60 seconds, this means that the expected transaction confirmation time is 3600 seconds.

Fig. 13 shows the average transaction confirmation time for different peer formation strategies and number of peers. The figure shows that peer formation strategy impacts the overall confirmation time. The x-axis represents the peer selection strategies while the y-axis indicates the confirmation time in seconds. Specifically, Fig. 13(a) and 13(b) show that in respect of the number of peers per node, the arrival rate has a higher impact on the confirmation time. It is worth highlighting that peer formation strategies bring less effect when the arrival rate is lower.



(a) $P = 8$ (b) $P = 4$

Figure 13. Average transaction confirmation times for the various peer formation strategies, number of peers P and low (3 t/min) and high (6 t/min) intensity generation rate λ. Error bars indicate 95% confidence intervals from 10 independent runs

1) Distribution of the confirmation time: Fig. 14 and 15 show the distribution of the transaction confirmation times for low and high arrival rates while the number of peers is fixed to eight. The x-axis represents the confirmation time in seconds. The y-axis is the log of the distribution $P(t_g - b_{gi+6} > t)$, when the three peer formation strategies are used.

Specifically, Fig. 14 illustrates the transaction confirmation time in seconds under a low transaction intensity. The three peer selection strategies are compared. In almost 80% of the cases, random and mixed peer formation strategies produce

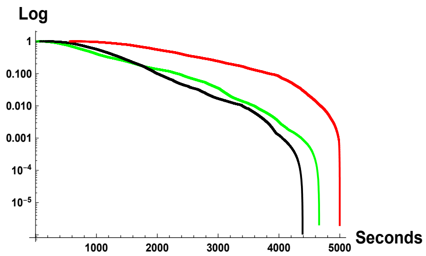


Figure 14. Transaction confirmation time, where $\lambda = 3$ transactions per minute per each node (low intensity)

transaction confirmation time less than 1654 seconds, while the normal approach introduces twice the confirmation time. 1% of the time, mixed and random strategies give confirmation time greater than 2000 seconds, while the normal approach doubles this amount.

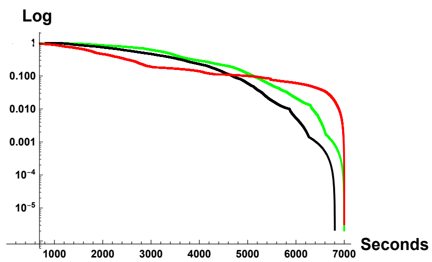


Figure 15. Transaction confirmation time, where $\lambda = 6$ transactions per minute per each node (high intensity)

Fig. 15 reports the transaction confirmation time in seconds under a high transaction intensity. In almost 80% of the cases, random and mixed peer formation strategies produce transaction confirmation time less than 4000 seconds, while the normal approach introduces 1000 seconds less confirmation time. 1% of the time, all the strategies give confirmation time greater than 5000 seconds.

Overall, for low transaction intensity, random peer selection performs better than the other two approaches. However, when we doubled the intensity, it was seen that all strategies yielded more similar distributions. Doubling the arrival intensity also affected the confirmation time. More transactions observe higher confirmation time. This reflects how the P2P protocol fails to propagate the transactions faster but spends significant time validating and processing transactions. It also means the P2P protocol is not good enough to handle high traffic, which has caused a doubt if Bitcoin will be able to catch up with the increasing user demand [24].

XI. Temporary Forking

A temporary fork occurs when two miners independently find and publish a new block referencing the same previous block. In such events, one block becomes an orphan block, where all the transactions not part of the accepted block (valid block) are pulled back to mempool for pickup again, and the miner who generated this block earns nothing for the effort. This affects the performance. The main cause of forking is propagation delay: Without such delay, the notification of a new block would be instantaneously received by all nodes avoiding them to continue working on generating new blocks. Since propagation delay depends also on the network topology that synchronizes between nodes as investigated in the previous sections, the present section is devoted to studying how peer selection strategies may affect the fork generation rate and how forking affects the transaction confirmation time.

A. *Introduction to temporary fork*

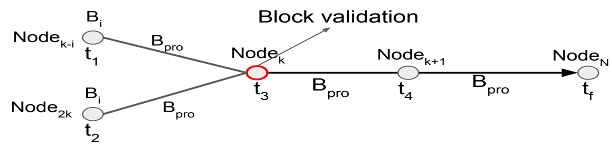
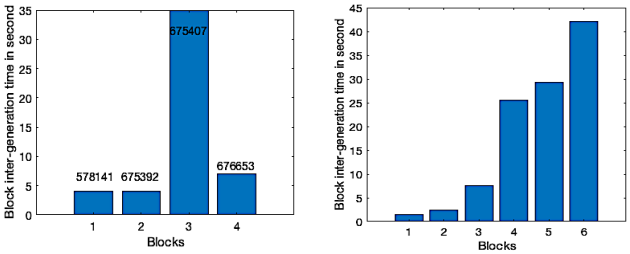


Figure 16. Block propagation time sequence

Fig. 16 illustrates the time sequence of block propagation. Two blocks are generated at time t_1 and t_2 then pushed to the neighbor nodes with some B_{pro} delay. When Node$_k$ receives these two blocks simultaneously, it validates both of them. Suppose both blocks point to the same previous hash of the block. Then the node compares the number of confirmations and an earlier timestamp. It selects one block based on these criteria, increases the confirmation, and forwards it to the neighbor nodes. Similarly, Node$_{k+1}$ will do the same operations, and this will increase the number of confirmation numbers of the valid block that will lead the orphan (fork) block to become less important with time. Once all the N nodes see these two blocks, the network ignores the orphan block while the valid block is added to the main chain [32]. In this way, the Bitcoin network maintains the ledger's consistency and security. However, this temporary fork impacts the overall performance of the technology. The validated transactions in the orphan block which are not part of the valid block are sent back to mempool to wait for pick-up again, increasing the average confirmation time. In addition, miners who created the ignored block (orphan block) wasted considerable resources for little gain.

1) Example: Based on our full independent Bitcoin live node [16], we were able to see four valid forks in the main chain from 578141 to 678853 block height. These four blocks hold from 1200 to 2400 transactions within. The average generation time between two blocks forming a fork is 12.5 seconds, which is much less than the 10-minute average block generation interval. Fig. 17(a) reports the inter-block generation time between fork and valid block in the Bitcoin network. The x-axis represents the blocks where the fork happened, and the y-axis indicates the inter-block generation time in seconds between fork and valid block. As we can see from the figure, the maximum inter-block generation time between valid and fork block is 35 seconds, which happened in the 675407 block height.

Fig. 17(b) shows the block inter-generation time between valid and fork blocks observed in our testbed, under the normal



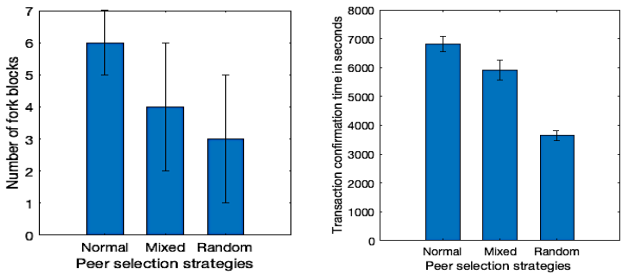
(a) $P = 8$ (b) $P = 8$

Figure 17. Fork vs. valid block example from live bitcoin full node

peer selection approach, high arrival rate, and 8 peers. The figure illustrates that block inter-generated time greater than 40 seconds might increases the high probability of creating a fork event. This plot is to demonstrate what we see from Fig. 17(a), which is from live Bitcoin node, is also seen from our setup.

B. Impact of peer selection strategy on the fork rate

The peer selection strategy impacts the performance of the system, particularly in terms of transaction propagation and confirmation time as discussed in the previous sections, in this subsection, we demonstrate its impact on the occurrence of forks.



(a) Number of forks (b) Confirmation time

Figure 18. Number of forks and their impact on transaction confirmation time

Fig. 18(a) reports the number of fork block happenings under different peer selection strategies, high arrival intensity, and the number of peers per node is 8. As we can see from the figure, the normal peer selection strategy brings a higher number of temporary forks. The mixed and random-based peer selection strategies produce lower numbers of fork blocks.

C. Impact of forking on transaction confirmation time

In the event of forking, transactions inside a fork block return to the mempool for being picked up again. This makes these transactions wait a longer time before confirmation. Fig. 18(b) reports the average transaction confirmation time seen by transactions inside the fork block. When the network ignores the fork block, all the transactions that are not part of the valid block are returned back to the mempool for pickup. The main issue with this is that the fork block may wait for more than one block to be ignored by the network, depending on the length of the pruned branch, which leads to the transaction frozen and waiting for a longer time. Fig. 18(b) also demonstrates this phenomenon. As we can see from the figure, the normal peer selection strategy produces a high number of fork blocks, which leads to transactions waiting longer than 6000 seconds.

Similarly, the mixed peer selection strategy produces a closer number of fork blocks to the normal peer selection approach, and the impact on the transaction confirmation time is more than 5500 seconds. However, the random peer selection strategy performs better than the other two approaches regarding the number of forks and confirmation time, with a transaction confirmation time of fewer than 3670 seconds.

D. Valid vs fork block overlap

When two blocks arrive within a shorter time difference window having the same hash pointing to the previous block, we call it a temporary fork. When this event happens, one of the blocks will become part of the chain, and the other will become an orphan block. Since the comparison is based on the previous block's hash, we further analyze the extent to which valid and fork blocks share the same transactions. Table II shows the overlap in percentage between the valid and fork blocks while considering different peer selection strategies.

Table II
OVERLAP BETWEEN VALID AND FORK BLOCK

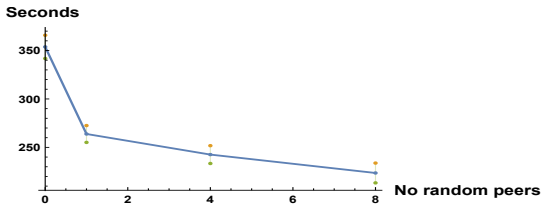
Peer selection strategies	Overlap valid vs fork block ((μ, σ))
Normal	(88%, 6%)
Mixed	(90%, 5%)
Random	(90%, 3%)

The mixed and normal peer selection strategies produce four to six fork blocks, where 90% of the transactions are the same, but the rest 10% are unique transactions which will be forced to return to the backlog for more waiting time. Similarly, for the random-based strategy, the valid and fork blocks share 90% of the transactions, but the remaining wait more time to be added to the chain.

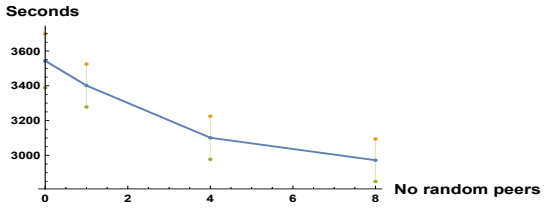
Overall, the peer selection strategy impacts the number of fork occurrences, mainly due to that different strategies give different propagation delays. This section has also showed that fork occurrence can affect the performance significantly. For instance, some of the transactions have had to wait more than 6600 seconds, which is 3000 more seconds of waiting time. This means some transactions have had to wait, on average, 11 valid block generation times.

XII. DISCUSSION

1) Proposed approaches: The P2P formation strategies are essential in propagating information between participating nodes. In this work, we showed that peer selection strategies affect the overall performance of Bitcoin. There have been some research works proposing schemes and methods to reduce the propagation delay in Bitcoin. These proposals focus on either introducing a compact block [23][27] or having some relay nodes [33][34] in the middle to provide a pipeline to push



(a) Propagation time



(b) Confirmation time

Figure 19. Average transaction propagation and confirmation times : number of peers is eight with high (6 t/min) intensity generation rate λ.

more updates to the other nodes. The compact block may introduce better performance in propagating the information based on the available bandwidth between participating nodes [27]. However, this method focuses on how to propagate blocks in the network than how to propagate transactions. Based on our observation, nodes may have a different number of arrivals at the backlog waiting. The compact block method has to push more than half of the block content in such cases.

Using relay nodes to reduce propagation delay is another method proposed by researchers. This method relies on the relay nodes having a higher number of peer nodes from the network, enabling pushing more updates in the network. The main challenge in this approach is that the relay nodes become a security bridge or vulnerability point. Attacking these nodes or taking control gives extra incentive to earn more or disrupt the overall activity in the network.

The best strategy to improve the propagation time is perhaps to improve the communication protocol. The protocol spends significant time validating and updating the same transaction. Furthermore, reducing the peer to peer network diameter by having peers other than the nearest may improve. The random strategies investigated in this paper are simple examples. For instance, Fig. 19 shows a specific use case, where the impact of adding more random peers on the performance is provided. The x-axis shows the number of random peers selected, and the y-axis indicates the propagation and confirmation times. Specifically, Fig. 19(a) and Fig. 19(b) show that adding random peers generally improves transaction propagation and confirmation times. In particular, Fig. 19(a) shows that adding one random peer significantly improves the propagation delay with a steep decline. However, this is not comparably visible in confirmation time (see Fig. 19(b)). One reason, as also implied by Fig. 19(a) and Fig. 19(b), is that the Bitcoin P2P protocol spends significant time in validating transactions, e.g.requiring six-block deep in the blockchain to confirm the transactions in the block, dominating the confirmation time.

2) Transaction propagation and confirmation times: The transaction propagation and confirmation times show some values higher than expected. This is because of the impact of the P2P formation strategies and P2P legacy relaying protocol. Some of the transactions have to return to mempool because of fork occurrence. For instance, the normal approach produces more forks than the other two approaches. In such cases, the transactions inside the fork block return to the backlog for pickup, of which some will be added to the new recent block, but others may wait for future block generation events. In addition to this, the processing capacity of the Raspberry Pi devices may contribute to some extent. Although we analyzed to observe the total usage, the Bitcoin, on average, in each device uses 114% CPU and 16% RAM. It is worth highlighting that the Raspberry Pi used has 64 bit quad-core Cortex-A53 and Cortex-A72, which is good enough to handle the traffic generated from Bitcoin and background processing.

3) Impact of temporary fork: The number of fork event occurrences has been reduced recently with the new Bitcoin core release [32]. However, the Bitcoin network is still not tested if it can handle high loads. Based on the current state where 3.3 to 7.2 transactions are processed per second, having arrivals at the mempool from 1700-2600 transactions waiting for pickup [3, 6, 12, 13]. The P2P network may handle processing and propagating updates with some acceptable performance index. However, when we pushed the load to 5500 to 6000 transactions at the backlog, the performance reduced significantly from propagating transactions in 10 seconds into 250-350 seconds. It also impacted the number of fork block occurrences in the network, making some transactions wait more than the expected confirmation time. For instance, for the normal peer formation strategy, the number of prude branches is higher because each node validates new arrivals before propagating to the neighbor nodes. In such cases, more delays happen in the network than having a few random peer links. This shows that the P2P network protocol requires improvement and research to improve its capacity.

XIII. Conclusion

In this paper, we analyzed the impact of peer formation strategies, arrival rate, and the number of peers on the overall performance of the technology. Specifically, we developed a testbed to mimic the Bitcoin P2P network, which enabled us to conduct a comprehensive investigation and gain deep insight into the impact of the underlying P2P network on the performance. The analysis shows that the transaction validation and propagation can take longer than expected, even with a low arrival rate and a high number of connected nodes. In addition, while the peer formation strategy currently adopted by the Bitcoin community is highly reliable in finding peers with low latency response, it does not give the best system performance in terms of propagation and confirmation times and fork rate. Considering a few random nodes in peer selection can improve the performance. These results indicate that the normal peer formation strategy alone may not bring optimal solutions. In addition, these results also imply that

improving the P2P communication protocol, including peer selection and the P2P network topology, has a great potential in improving the performance, including transaction confirmation time.

REFERENCES

[1] Lina Alsahan, Noureddine Lasla, and Mohamed Abdallah. "Local Bitcoin Network Simulator for Performance Evaluation using Lightweight Virtualization". In: *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. 2020, pp. 355–360.

[2] Ryohei Banno and Kazuyuki Shudo. "Simulating a Blockchain Network with SimBlock". In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2019, pp. 3–4.

[3] Bitaps. *Today bitcoin blocks*. URL: https : / / bitaps . com / blocks. (accessed: 01.07.2020).

[4] Bitcoin. *Bitcoin Core release 0.21.0*. URL: https://bitcoincore.org/en/releases/0.21.0/. (accessed: 01.07.2020).

[5] Bitcoin. *Bitcoin developer-guide*. URL: https://btcinformation.org/en/developer-guide#peer-discovery. (accessed: 01.07.2020).

[6] Blockstream.info. *Recent Transactions and blocks*. URL: https : / / blockstream.info/tx/recent. (accessed: 01.07.2020).

[7] Christian Decker and Roger Wattenhofer. "Information propagation in the Bitcoin network". In: *IEEE P2P 2013 Proceedings*. 2013, pp. 1–10.

[8] Varun Deshpande, Hakim Badis, and Laurent George. "BTCmap: Mapping Bitcoin Peer-to-Peer Network Topology". In: *2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*. 2018, pp. 1–6.

[9] Joan Antoni Donet Donet, Cristina Pérez-Solà, and Jordi Herrera-Joancomartí. "The Bitcoin P2P Network". In: *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 87–102.

[10] Jean-Philippe Eisenbarth, Thibault Cholez, and Olivier Perrin. "A Comprehensive Study of the Bitcoin P2P Network". In: *2021 3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*. 2021, pp. 105–112.

[11] Meryam Essaid, Sejin Park, and Hongteak Ju. "Visualising Bitcoin's Dynamic P2P Network Topoogy and Performance". In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2019, pp. 141–145.

[12] Explorer. *Blockchain Explorer*. URL: https://www.blockchain.com/explorer. (accessed: 01.07.2020).

[13] Btc Block Explorere. *Block Explorer*. URL: https : / / btc . com/. (accessed: 01.07.2020).

[14] Muntadher Fadhil, Gareth Owen, and Mo Adda. "Bitcoin Network Measurements for Simulation Validation and Parameterisation". In: May 2016.

[15] Muntadher Fadhil, Gareth Owenson, and Mo Adda. "Locality based approach to improve propagation delay on the Bitcoin peer-to-peer network". In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2017, pp. 556–559.

[16] Befekadu G. Gebraselase, Bjarne E. Helvik, and Yuming Jiang. "An Analysis of Transaction Handling in Bitcoin". In: *2021 IEEE International Conference on Smart Data Services (SMDS)*. 2021, pp. 162–172.

[17] Befekadu G. Gebraselase, Bjarne E. Helvik, and Yuming Jiang. "Transaction Characteristics of Bitcoin". In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 2021, pp. 544–550.

[18] Johannes Göbel et al. "Bitcoin Blockchain Dynamics: The Selfish-Mine Strategy in the Presence of Propagation Delay". In: *Performance Evaluation* 104 (May 2015).

[19] Stephen Hemminger et al. "Network emulation with NetEm". In: *Linux conf au*. Vol. 5. Citeseer. 2005, p. 2005.

[20] Muhammad Imran, Abas Md Said, and Halabi Hasbullah. "A survey of simulators, emulators and testbeds for wireless sensor networks". In: *2010 International Symposium on Information Technology*. Vol. 2. 2010, pp. 897–902.

[21] Audrius Jurgelionis et al. "An Empirical Study of NetEm Network Emulation Functionalities". In: *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*. 2011, pp. 1–6.

[22] Hidehiro Kanemitsu and Hidenori Nakazato. "KadRTT: Routing with network proximity and uniform ID arrangement in Kademlia". In: *2021 IFIP Networking Conference (IFIP Networking)*. 2021, pp. 1–6.

[23] Aeri Kim et al. "Analysis of Compact Block Propagation Delay in Bitcoin Network". In: *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2021, pp. 313–318.

[24] Quan-Lin Li, Jing-Yu Ma, and Yan-Xia Chang. "Blockchain Queue Theory". In: *Computational Data and Social Networks*. Ed. by Xuemin Chen et al. Cham: Springer International Publishing, 2018, pp. 25–40.

[25] Andrew K. Miller et al. "Discovering Bitcoin ' s Public Topology and Influential Nodes". In: 2015.

[26] D.L. Mills. "Internet time synchronization: the network time protocol". In: *IEEE Transactions on Communications* 39.10 (1991), pp. 1482–1493.

[27] Jelena Mišić, Vojislav Misic, and Xiaolin Chang. "On the Benefits of Compact Blocks in Bitcoin". In: Feb. 2020.

[28] Jelena Mišić et al. "Modeling of Bitcoin's Blockchain Delivery Network". In: *IEEE Transactions on Network Science and Engineering* 7.3 (2020), pp. 1368–1381. DOI: 10.1109/TNSE.2019.2928716.

[29] Saeideh G. Motlagh, Jelena Mišić, and Vojislav B. Mišić. "Impact of Node Churn in the Bitcoin Network". In: *IEEE Transactions on Network Science and Engineering* 7.3 (2020), pp. 2104–2113.

[30] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[31] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. "Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network". In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*. 2016, pp. 358–367.

[32] Till Neudecker and Hannes Hartenstein. "Short Paper: An Empirical Analysis of Blockchain Forks in Bitcoin". In: Sept. 2019, pp. 84–92.

[33] Kai Otsuki, Ryohei Banno, and Kazuyuki Shudo. "Quantitatively Analyzing Relay Networks in Bitcoin". In: *2020 IEEE International Conference on Blockchain (Blockchain)*. 2020, pp. 214–220.

[34] Kai Otsuki et al. "Effects of a Simple Relay Network on the Bitcoin Network". In: Aug. 2019, pp. 41–46.

[35] N D Patel, B M Mehtre, and Rajeev Wankar. "Simulators, Emulators, and Test-beds for Internet of Things: A Comparison". In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2019, pp. 139–145.

[36] Carsten Rieck. "An Approach to Primary NTP by Using the LINUX Kernel". In: *2007 IEEE International Frequency Control Symposium Joint with the 21st European Frequency and Time Forum*. 2007, pp. 873–876.

[37] Hirotsugu Seike, Yasukazu Aoki, and Noboru Koshizuka. "Fork Rate-Based Analysis of the Longest Chain Growth Time Interval of a PoW Blockchain". In: *2019 IEEE International Conference on Blockchain (Blockchain)*. 2019, pp. 253–260.

[38] Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. "Performance Modeling and Analysis of the Bitcoin Inventory Protocol". In: Apr. 2019. DOI: 10.1109/DAPPCON.2019.00019.

[39] Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. "A Theoretical Model for Block Propagation Analysis in Bitcoin Network". In: *IEEE Transactions on Engineering Management* (2020), pp. 1–18.

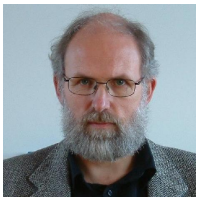
[40] Jun.Kawahara Shoji.Kasahara. "Effect of Bitcoin fee on transaction-confirmation process". In: *Journal of Industrial and Management Optimization* 15.1547 (2019), p. 365.

[41] Amool Sudhan and Manisha J Nene. "Peer Selection Techniques for Enhanced Transaction Propagation in Bitcoin Peer-to-Peer Network". In: *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2018, pp. 679–684.

[42] Taotao Wang et al. "Ethna: Analyzing the Underlying Peer-to-Peer Network of Ethereum Blockchain". In: *IEEE Transactions on Network Science and Engineering* 8.3 (2021), pp. 2131–2146.

[43] *What is VPP?* https://wiki.fd.io/view/VPP/What_is_VPP?. Accessed: 2021-06-10.

Befekadu Gezaheng Gebraselase received a B.Sc. degree in computer science from Addis Ababa University, Ethiopia, in 2012, an M.Sc. degree in computer science from the University of Milan, Italy, in 2018, and currently pursuing a Ph.D. degree in computer science and technology from the Norwegian University of Science and Technology, Trondheim, Norway, since 2018. From 2012 to 2016, he worked with Abyssinia bank, Ethiopia, as a Senior network engineer. He has published seven scientific conferences and journal papers. His research interests include Blockchains, vehicular network, IoT, fog computing, UAV network, IoT, and 5G/6G wireless network, and machine learning.



Bjarne Emil Helvik was born in 1952. He received the Siv.ing. degree (M.Sc. in technology) and the Dr.Techn. degree from the Norwegian Institute of Technology (NTH), Trondheim, Norway, in 1975 and 1982, respectively. Since 1997, he has been a Professor with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU). From 2009 to 2017, he was the Vice Dean with responsibility for research with the Faculty of Information Technology and Electrical Engineering, NTNU, where he was a Principal Investigator with the Norwegian Centre of Excellence Q2S, Centre for Quantifiable Quality of Service in Communication Systems from 2003 to 2012. He has previously held various positions with ELAB and SINTEF Telecom and Informatics. From 1988 to 1997, he was appointed as an Adjunct Professor with the Department of Computer Engineering and Telematics, NTH. His field of interests includes QoS, dependability modeling, measurements, analysis and simulation, fault-tolerant computing systems and survivable networks, as well as related system architectural issues. His current research focus is on ensuring dependability in services provided by multidomain, virtualized ICT systems.

Yuming Jiang (Senior Member, IEEE) received the B.Sc. degree from Peking University and the Ph.D. degree from the National University of Singapore. From 1996 to 1997, he was with Motorola, Beijing, China, and the Institute for Infocomm Research (I2R), Singapore, from 2001 to 2003. He has been a Professor with the Norwegian University of Science and Technology, Trondheim, Norway, since 2005. He has authored the book entitled Stochastic Network Calculus. His research interests are the provision, analysis, and management of quality of service guarantees. He was the Co-Chair of IEEE Globecom 2005—General Conference Symposium, the TPC Co-Chair of 67th IEEE Vehicular Technology Conference (VTC) 2008, the General Chair of IFIP Networking 2014 Conference, the Chair of the 2018 International Workshop on Network Calculus and Applications, and the TPC Co-Chair of the 32nd International Teletraffic Congress (ITC32) in 2020.