On forks and fork characteristics in a Bitcoin-like distribution network

Vojislav B. Mišić*, Jelena Mišić*, and Xiaolin Chang†
*Ryerson University, Toronto, ON, Canada
†Beijing Jiaotong University, Beijing, China

***Abstract*—Chain growth in a permissionless blockchain-based ledger is mostly defined by the characteristics of the distribution network and the chosen consensus protocol. In this paper, we investigate the performance of block propagation in a Bitcoin-like peer-to-peer distribution network, and highlight the impact of the Nakamoto consensus protocol on the dynamics of blockchain growth. We use a simulated network with nodes located in different geographic regions, each with its own propagation characteristics; the values of network parameters are chosen to match available data for the Bitcoin peer-to-peer network. We show that the latency of block propagation is mainly affected by the mean round-trip time; that forks occur more often when mean round-trip time is longer, and that the ratio of the number of nodes opting for one or the other of competing blocks as the main chain tip can occur in almost any ratio; finally, that the mean time to resolve a fork is approximately equal to block inter-arrival time.**

***Index Terms*—blockchain; peer-to-peer network;**

I. Introduction

Haber and Stornetta were the first to describe the concept of blockchain [10], a data structure suitable for implementing systems that require ‘a decentralized, replicated, immutable and tamper-evident log’ [1] of a series of transactions or documents. Possibly the best known application of blockchain is Bitcoin [18], a digital cryptocurrency that uses blockchain as the replicated ledger with information about monetary transactions. User identity is protected through a public-private key cryptosystem but the actual contents of transactions are kept public. Bitcoin blocks are generated using the Proof-of-Work (PoW) approach [24] that involves solving a cryptographic puzzle, the difficulty of which is periodically adjusted to maintain a constant (and, perhaps, artificially low) average rate of block mining.

A number of problems are inherent to the structure of Bitcoin blockchain and its consensus protocol. Perhaps the best known such problem is forking which refers to the scenario, shown in Fig. 1, in which two (or more) apparently valid blocks propagate through the network in a short period of time, and different nodes install one or the other as the main tip of their individual blockchains. A fork may even be extended if new blocks are mined that build on different fork branches. In this manner, the distributed ledger is temporarily rendered inconsistent, and the consensus protocol does not offer an easy way to determine which of these is valid and which is not. As the result, there is a possibility for a number of different security attacks [8]. The fork is eventually resolved by accepting the ‘longest chain, which has the greatest proof-of-work effort invested in it’ as the correct one [18]; alternatively, the chain with the most accumulated PoW effort (which may or may not be the longest sub-chain) may be chosen [23].

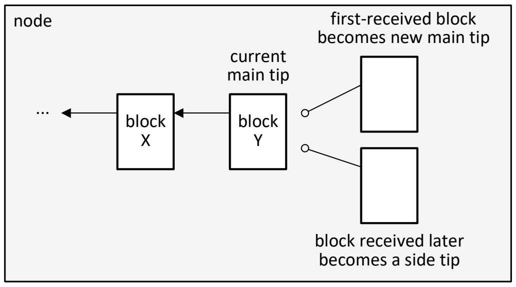

node
first-received block becomes new main tip
current main tip
...
block X
block Y
block received later becomes a side tip

Fig. 1. A fork occurs when different nodes receive different but otherwise valid blocks at nearly the same time.

In this paper, we examine the process of generation of forks in a Bitcoin-like P2P delivery network and their characteristics. The analysis is performed on a purpose-built simulator of Bitcoin-like delivery network with 5,000 nodes. Simulation was chosen over measurement for reasons of coverage and flexibility, as we can observe all nodes at all times which is impossible in the real network; moreover, measurement-based approaches cause a disruption in network operation, the impact of which cannot be fully known. Our results show that (a) the latency of block propagation is mainly affected by the mean round-trip time; (b) the frequency of fork occurrence is correlated with data round-trip time – forks occur more often when mean round-trip time is longer; (c) when a fork occurs, partitioning of the node set according to the block chosen as the main chain tip can occur in almost any ratio; and (d) mean time to resolve a fork is close to block inter-arrival time.

The rest of the paper is organized as follows. Section II presents most relevant related work. Sections III and IV describe the operation of the Bitcoin blockchain and block management in more detail, including processing of confirmed blocks and the mechanism of fork generation. Section V presents our simulation setup, while Section VI presents and discusses our main findings. Finally, Section VII concludes the work and summarizes some avenues for further research.

II. Related work

A number of reports related to measurements of various aspects of the Bitcoin network have appeared in past several years. Combinations of data regarding network size, geographical distribution of nodes, number of connections per node, and block and transaction statistics were reported in [2], [5], [7], [16], [22]. Measurements of round trip times (RTT), coupled with node distribution in the Bitcoin (BTC) network with vantage point in Vienna, Austria, were reported in [2]. It is worth noting that all reported works used measurement tools which had interfered to some extent with genuine activities in the BTC network.

One of the first and most influential works in measuring performance of data distribution algorithms for BTC network was reported in [4]. Results suggest that block distribution time is exponentially distributed and that forking probability was about 1.8% under 100kbps of TCP connection throughput but with unclear values of node connectivity, RTT distribution, and network size. Also, [14] provides a wealth of technical as well as non-technical data.

The Bitcoin data distribution protocol actually resembles randomized rumor spreading [11], but without full network connectivity and random choice of peers for each transmission by the source. Available literature about data distribution paradigm uses different yet related protocols. For example, [21] considers broadcast and flooding, [4], [6] use a gossip (i.e., controlled flooding) based algorithm, and [20] considers both kinds of data distribution protocols. However, pure broadcast protocol tends to be non-scalable to large networks under ever increasing transaction traffic.

A queuing theoretic analysis has been undertaken in [21], although their simulation results have been obtained for extremely limited scale scenarios with 2 and 5 nodes.

The importance of understanding the network layer of the Bitcoin blockchain in order to prevent security attacks is discussed in [6], [20].

III. Operation of the Bitcoin blockchain

Blockchain is a data structure composed of blocks with arbitrary content. Each block is identified by its hash, which is used to identify the previous block in the chain so that the blocks effectively form a backward linked list. The first (oldest) block is oftentimes referred to as 'genesis' block and it is created by the software package that manages the chain [18].

Tamper-detection capability, often incorrectly referred to as 'immutability,' is provided by adding the Merkle or hash tree [15], a binary tree in which leaves are actual transactions, and nodes higher up in the tree are formed by hashing the hashes of the two nodes below. The root of the Merkle tree is added to the header of each block, thus providing an easy way to check whether the transactions have been tampered with.

In addition to the Merkle tree root, block header contains the previous block hash, timestamp of block creation, as well as current difficulty level and nonce – parameters related to the mining process. Blocks are then linked in a list interconnected through hashes that point to the previous block.

The node that has generated a transaction or mined a block distributes it to its peers, who distribute it to their peers, and so on, until the transaction or block reach the entire network. The distribution process is performed as a series of unicast communications in which the source node queries its peers, one by one, with inventory (INV) messages to check if they have the item in question. Peers that know about the item remain silent, while those that don't know it respond with a GETDATA message requesting the data item. The source node then sends the data item in the appropriate message (TRANSACTION or BLOCK). Newly received transactions are checked and, if found to be valid, added to the *mempool* of a node, a pool of validated transactions waiting to be confirmed through inclusion in a block. Newly received blocks and the transactions they contain are checked and, if found to be valid, added to the local blockchain.

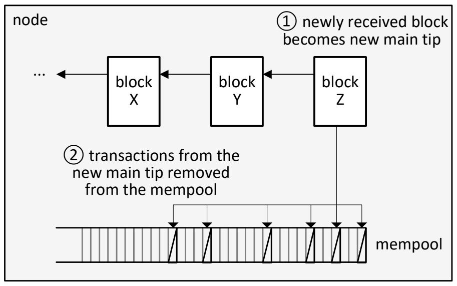
Synchronization of ledger replicas is achieved through a Proof-of-Work-based decentralized consensus protocol, often referred to as the Nakamoto consensus [1]. In this protocol, nodes known as 'miners' choose from their mempools a set of transactions that fit into a single block and then try to solve a cryptographic puzzle based on that block. Once the desired solution is found, the miner node announces the new block to other nodes. When a node learns about the new block, they obtain it, verify its contents and the solution of the puzzle and, if confirmed, append the newly mined block to their blockchains. This process may take some time depending on the number of nodes – there is no central authority to confirm a block, and participation in a Bitcoin network is effectively open to everyone. This approach is known as 'permissionless,' as opposed to 'permissioned' in which only the nodes explicitly authorized by a trusted authority may take part in blockchain operation and maintenance.

IV. Blockchain management and forks

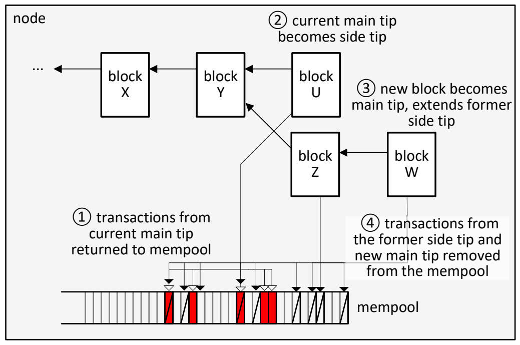
A. *On forks in the blockchain*

Block propagation takes precedence over mining, as an ongoing process of mining a block will be abandoned if the node receives a new block. If a node manages to mine a block, say, Z, and begins distributing it before another recently mined block, say, U, which has been propagated to all the nodes. In this case, some nodes will append U as the main tip and Z as the side tip while others will do the opposite – the situation known as *fork*.

Forks are problematic since the distributed architecture of the Bitcoin blockchain makes it impossible for a node to figure out which of the two otherwise legitimate blocks should be used to extend the chain. Asking one's peers does not help as some of them will have one block as their main tip while the others will have the second one. (Remember that the node has received the new block from one of its neighbors, unless it has been mined locally.) Hence the node must retain both blocks until the fork is eventually resolved. Moreover, competing blocks may contain contradicting transactions, including double spending ones (i.e., those that use the same input) [18], As the result, the ledger becomes inconsistent – yet individual nodes are not aware of it! Of course, the set of transactions included in two blocks mined within a short time of one another may actually overlap. In fact, the only transaction guaranteed to differ is the coinbase transaction through which the miners claim the mining fee.


node
① newly received block becomes new main tip
...
block X
block Y
block Z
② transactions from the new main tip removed from the mempool
mempool

(a) Normally, a new block extends the current main tip and its transactions are removed from the local mempool.


node
② current main tip becomes side tip
...
block X
block Y
block U
③ new block becomes main tip, extends former side tip
block Z
block W
① transactions from current main tip returned to mempool
④ transactions from the former side tip and new main tip removed from the mempool
mempool

(b) A new block may extend a side tip and, if its height exceeds that of the current main tip, switch places with the main tip.

Fig. 2. Pertaining to fork processing.

The only way to resolve a fork is by receiving the next block and appending it to the chain. This block will connect to the main tip or one of the side tips, depending on whether the node that mined it had one or the other block as its main tip. We note that the chain may contain one or more side tips at the same height as the main tip, if three or more blocks had been mined within a short time period and sent out before any of them has been propagated through the network.

Furthermore, a fork may extend for more than a single block, as subsequent blocks may extend the main or the side tip – forks of length of up to four have been recorded in the Bitcoin blockchain [16].

B. Processing of a newly arrived block

To explain in detail the process of adding blocks to the blockchain, we will use the following notation:

- B is the newly arrived block, and $B.previous$ is its previous block, i.e., the block pointed to by the appropriate hash in the header of block B;
- $MainTip$ refers to the current main tip – the most recently added block at the tip of the main blockchain stem;
- $MainStem$ refers to the main blockchain stem, from current $MainTip$ to the genesis block;
- $SideTips$ refers to the collection of blocks that lead sideways from the main blockchain stem;
- $Orphans$ refers to a list of orphan blocks that point to a block not in the main blockchain stem; and
- $height(\cdot)$ is a function that returns the height of a block, i.e., the distance from the genesis block (which is at height zero).

The actual block confirmation protocol operates as shown in Algorithm 1 below.

Algorithm 1: Block processing.

```
Data: confirmed block $B$
Result: updated blockchain, $MainTip$, $SideTips$
if $B.previous \notin MainStem$ then
    add $B$ to $Orphans$;
    request $B.previous$ from $B$'s sender;
else
    validate $B$;
    if $B.previous == MainTip$ then
        append $B$ to $MainTip$;
        set $B$ as $MainTip$;
    else if $B.previous \in sideTips$ then
        append $B$ to $B.previous$;
        if $height(B.previous) == height(MainTip)$
         then
            remove $B.previous$ from $SideTips$;
            switch $MainTip$ with $B$;
        end
    else
        add $B$ to $SideTips$;
        for $o \in Orphans$ do
            if $B == o.previous$ then
                remove $o$ from $Orphans$;
                process $o$;
            end
        endfor
    end
    for $p \in Peers$ do
        send $B$ to peer $p$;
    endfor
end
```

The algorithm first checks if the new block points to any of the blocks in the main stem of the blockchain. If not, the new block is an orphan block and it cannot be processed at this time, as its predecessors and the transactions they contain are unknown to the node and, consequently, cannot be verified. Instead, it is recorded in the $Orphans$ list and a request for its previous block is sent to the source of the original block.

In other cases, the new block is validated, together with its transactions. If the new block points to the current main tip, it is appended to the blockchain at the main tip. The new block thus becomes the new main tip of the blockchain and the transactions it contains are removed from the local mempool, as shown in Fig. 2(a).

The new block may also point to a block from the $SideTips$ list. In this case, the new block is appended to its previous block and its height is checked against that of the current main tip. If the height of the new block is greater, the current main tip is added to the $SideTips$ list while the new block becomes the new main tip and its previous block is removed from the $SideTips$ list. Furthermore, the transactions from the current main tip and all of its previous blocks down to the point of branching of the new block, are returned to the mempool. Then, all the transactions from the new block and its previous blocks down to the branching point are removed from the mempool. This is schematically shown in Fig. 2(b) for a branching point immediately below the current main tip.

Finally, the new block may simply point to a block below the current main tip. In this case, the new block is added to the list of $SideTips$ but without any processing of the transactions it contains.

Ultimately, processing of a non-orphan block results in one of the following outcomes: extension of the main tip; extension of a side tip and, possibly, switching places with the main tip; or creation of a new side tip.

V. Experiment setup

While the mechanism of fork creation is known, little has been done to address the questions such as when do forks appear, how long do they last, what are the sizes of node partitions created by a fork, and the like. This is why we have designed and performed the experiments described below.

Unfortunately, the Bitcoin system and the associated network do not lend themselves easily to a comprehensive analysis. Due to their decentralized and permissionless character, realistic network parameters can be learned only by monitoring the network which can be achieved by using either the original Bitcoin daemon or a functionally equivalent piece of software. However, this approach fails in two important aspects, both of which are well known in the Bitcoin research community.

First, due to the decentralized character of the network, one cannot possibly observe the evolution of blockchains across all peers. At best, a limited number of nodes can be monitored, depending on the connectivity of the monitoring peer or peers. If a single monitoring node can monitor only a few hundred nodes out of about 10,000 that are connected at any given time, the accuracy of measurements will not be high.

Second, any measurement setup that connects to the real network will inevitably disrupt its operation. In some cases, the monitoring nodes only use up some of the available connections and the associated bandwidth [2], [7], [16]. In this case, measurement slightly degrades the capacity of the network but does not affect the legitimate traffic carried therein. In other cases, the behavior of the network is probed through insertion of fake transactions [5]. Unfortunately, this approach uses up the capacity but it also creates quasi-legitimate traffic which is probably more disruptive as it interferes with the block confirmation process, although the extent of the disruption is not actually known.

Moreover, the information we are looking for cannot be obtained by scraping data from the existing blockchains, be it from a real network trace as done in [4] or using a publicly available information such as the one found at https://www.blockchain.com/explorer. In both cases, one can obtain just a list of confirmed blocks and transactions, but the evolution of individual chains remains unknown.

All of those shortcomings can be addressed through a simulation model, in which we can observe all nodes and the evolution of their blockchains at all times. Moreover, we can modify the network and protocol parameters at will. Of course, whether it is better to use measurement on the real system, or to simulate it in a 'laboratory' environment, is an age-old question in system analysis [13]. But in this case, the coverage and flexibility offered by the simulation model far outweigh its downsides. It is worth noting that the simulation approach seems rather unpopular amongst Bitcoin researchers as only a few papers [19] used it to analyze the behavior of the Bitcoin system.

We have built a simulator of the Bitcoin peer-to-peer network and the associated protocol, using Anylogic 8.3.3 by Anylogic, Inc., Oakbrook Terrace, IL. The simulated network had 5,000 nodes that implement the distributed consensus protocol outlined above. The network was run for 605,000 seconds which corresponds to 7 days of simulated time. Parameter values were chosen according to available data from research papers and public sites such as https://www.blockchain.com/ mentioned above.

Transactions are generated at a rate of 1.25 per second and sent to a random node in the network; the target is chosen with equal probability from all nodes. All nodes maintain their individual mempools and blockchains, but only 20% of them actually mine blocks – others simply collect and disseminate transactions and blocks. For simplicity, in this experiment all miners were assumed to have identical hashpower. Blocks contain a random number of transactions, uniformly distributed in the range between 500 and 2,000.

TABLE I
Minimum round-trip time t_{min}, in milliseconds.

from a node in	to a node in Americas	Europe	Asia
Americas	10.0	100.0	200.0
Europe	100.0	20.0	200.0
Asia	200.0	200.0	100.0

TABLE II
Mean value of the variable part of round-trip time, t_{mean}, in milliseconds.

from a node in	to a node in Americas	Europe	Asia
Americas	20	40	66.7
Europe	40	20	66.7
Asia	66.7	66.7	40

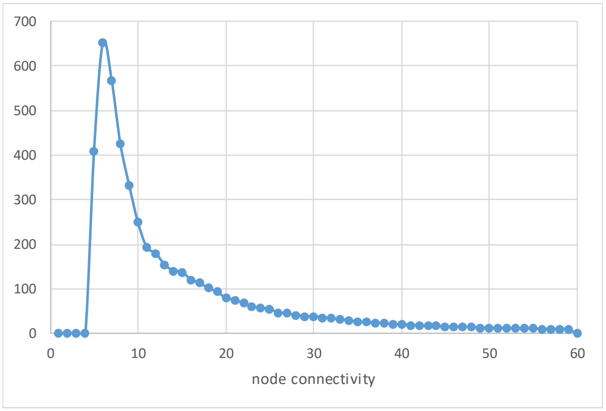


Fig. 3. Distribution of node connectivity.

According to the measurements in [2], we have assumed that 50%, 20%, and 30% of the nodes are located in Europe, Asia, and the Americas, respectively. The distribution of peer connectivity was determined as follows. Bitcoin node can have up to 8 outbound peers and up to 125 inbound ones but these numbers have nothing to do with the directionality of the link. Namely, the protocol[1] involves message transmission in both directions, and TCP protocol is inherently bidirectional [12]. Rather, the 'outgoing' and 'inbound' labels denote the manner in which peers learn of each other.

Consequently, our network had randomly generated connectivity with the probability distribution obtained by combining a binomial distribution and a long-tail with each node connected to at least 5 peers, with an average of 12.01, and graph diameter of five.

Regarding the block arrival process, it is widely assumed that the time to complete PoW for a new block is exponentially distributed, which means that the arrival of confirmed blocks to the network can be modeled with a homogeneous Poisson process [4], [8], [9], [18]. This is the assumption we have used in our work, with the mining rate of 1 in 10 minutes (i.e., 600 seconds) which is the target for the actual Bitcoin network. It is worth noting that a recent analysis indicates that a non-homogenous Poisson process might be a better match due to periodic adjustment of the difficulty to compute the nonce in the block header [3]. However, our experiment run time was too short to incorporate this effect into our analysis.

Block delivery time is assumed to follow exponential distribution [4], but few works have dealt with node-to-node delivery of blocks; a recent work has shown that this time can be described with a sub-exponential distribution [17]. Recent measurements have shown that the parameters of this distribution depend on the geographical locations of the sender and receiver peers [2]. We have modeled the delay between a message and the ensuing response with $t_d = t_m + \delta RTT(t_{min}, t_{mean}$, where t_m is the actual message propagation time, $RTT(t_{min}, t_{mean})$ is a random value obtained from an exponential distribution with a mean of t_{mean}, shifted to the right by t_{min}; both values are dependent on the locations of the nodes, as described in Tables I and II, respectively. The value δ is a parameter which we used to scale the delay for the entire network; our experiments used values of $\delta = 1.5$ and 0.75 to model slower and faster networks, respectively.

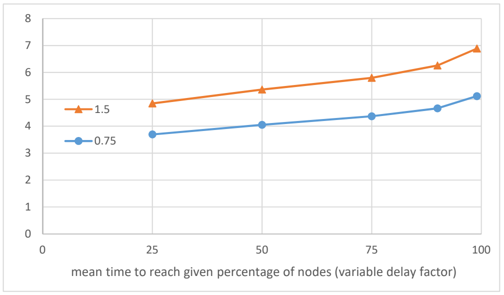
VI. Experimental results

A. Block propagation

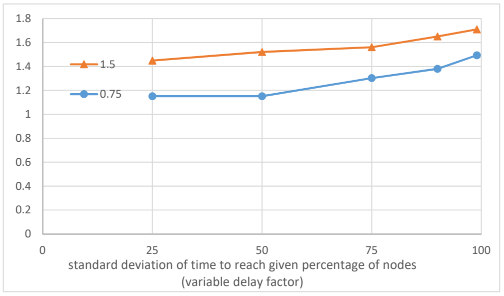
We have recorded the time needed for a block to reach a predefined percentage of nodes; the results are shown in Fig. 4 where the top diagram shows mean value while the bottom one shows standard deviation of the respective variable. All times are expressed in seconds.

As can be seen from Fig. 4(a), a major part of block propagation time is used to reach the first 25% of the nodes. Once that coverage is reached, the block will propagate fairly quickly to the subsequent milestones of 50, 75, and 90 percent

[1]https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_4):_P2P_Network, last accessed February 4, 2019



(a) Mean time to reach a given percentage of nodes.



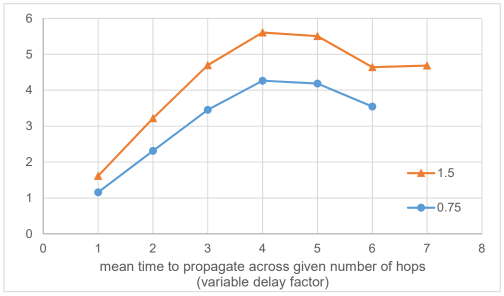
(b) Standard deviation of time to reach a given percentage of nodes.

Fig. 4. Propagation time (in seconds) for a block vs. percentage of nodes reached, for two values of delay factor δ.

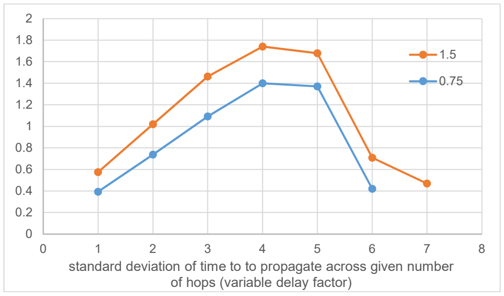
of the nodes. This last observation complies well with the nature of the controlled flooding protocol used in Bitcoin. Only when the time comes to reach the last ten percent of the nodes, i.e., to get to the 99% mark, does the slope of the curve increase slightly.

Regarding standard deviation of the block propagation time, Fig. 4(b), its growth is similar to that of the mean value, except that values are lower, and the amount of change in relative terms is smaller. It is worth noting that standard deviation of about 1.2s is not too high compared to the mean values of about four seconds, which shows that block propagation time has a sub-exponential distribution. More importantly, it shows that actual propagation times do not vary too much from one block to another, regardless of the location of the peer that mined them.

We have also measured the time it takes a block to propagate over a given number of hops from the source; the resulting mean value and standard deviation are shown in top and bottom diagrams of Fig. 5, respectively. The time to propagate for a given number of hops increases in an almost linear fashion up to about 4 hops, stays about the same for five hops, and drops rapidly for 6, in case of lower network delay factor, and 6 and 7 hops, in case of higher network delay factor. It may seem strange that some blocks actually have to pass six or seven hops to reach all nodes when the network has the diameter of 5. But one should keep in mind that the controlled flooding data propagation protocol in Bitcoin proceeds as a series of unicasts with random delays at each hop. As the result, some nodes may get a block sooner over a roundabout route with a higher number of hops but with lower delay. When the network delay factor is lower, all nodes in the network are



(a) Mean time to reach a given number of hops.



(b) Standard deviation of time to reach a given number of hops.

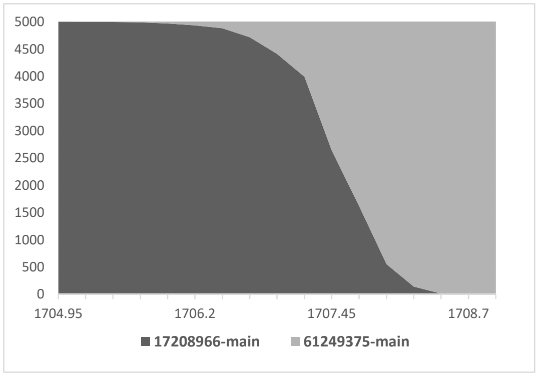
Fig. 5. Time (in seconds) for a block to reach a given number of hops for two values of delay factor δ.

reached in six hops, while seven-hop routes appear only in the network with higher delay factor. However, the number of such long routes is much smaller than the number of those with up to 5 hops, and the 7-hop route actually occurs in a single-digit number of cases only.

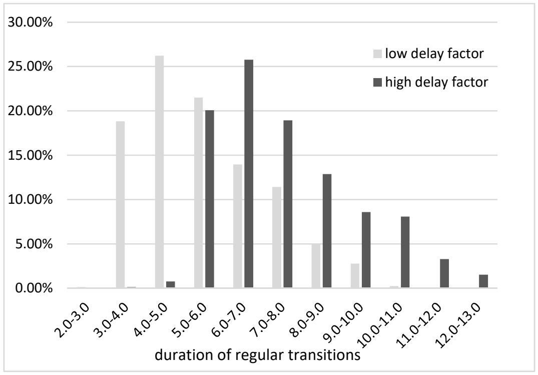
It is interesting to note that aggregate mean values (not shown for reasons of clarity) are about 5.32 and 4.01 seconds for the higher and lower delay factor values, respectively; both values correspond well to measured values. Those values are only slightly lower than the greatest per-hop mean which corresponds to four hops from the source node, for both values of the delay factor. This is due to the distribution of node connectivity, as the number of nodes that can be reached in four or five hops vastly outnumbers all others. This is also the reason for the sharp drop in standard deviation of block propagation time beyond five hops, as observed in Fig. 5(b).

B. Extending the blockchain

In the block propagation measurements discussed above, we didn't distinguish between blocks that eventually become part of the individual node blockchain and those that do not. Namely, a new block may reach a node, but that doesn't mean that it will automatically be added to the blockchain as a new main tip. Instead, the block may end up as a side tip in case of a fork, as shown above, or it may be discarded if it's not confirmed. To see how fast the blocks that eventually become the main tip of the blockchain get there, we have also kept a log of the main tips for all nodes and observed tip transition times. Tip transition time is measured from the moment a new block appears in a single blockchain – which would be that of the node that has mined the block – to the moment the new



(a) Block 61249375 replaces block 17208966 as the main tip. Following an identical pattern (not shown), block 17208966 replaces block 98971925 as the previous, i.e., second-highest block in all blockchains.



(b) Duration of regular main tip transitions, for low (gray) and high (black) values of delay factor δ.

Fig. 6. Pertaining to regular tip transitions. All times in seconds.

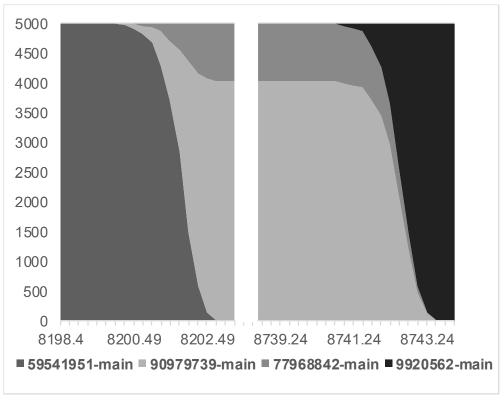
block appears as the main tip in the blockchains of all nodes in the network; it does not include the time needed to mine the block.

Normally, a new block appears first in the miner node and propagates through the network, replacing the former main tip. As more nodes confirm the new block and append it to their blockchains as the main tip, the former main tip becomes the previous block, as shown in Fig. 6(a).

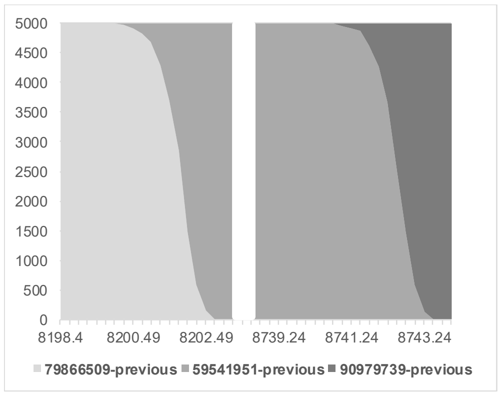
The distribution of actual time to execute the transition is shown in Figs. 6(b), for low and high values of the delay factor value represented with gray and black columns, respectively. As can be seen, more than half of regular transitions last less than six or eight seconds, for the low and high delay factor, respectively; in fact, about 98.5% of all transitions last up to 10 or 12 seconds, for the low and high delay factor case, respectively. These numbers correspond well to the measured values of block propagation times shown in Figs. 4 and 5 above.

C. Fork transitions

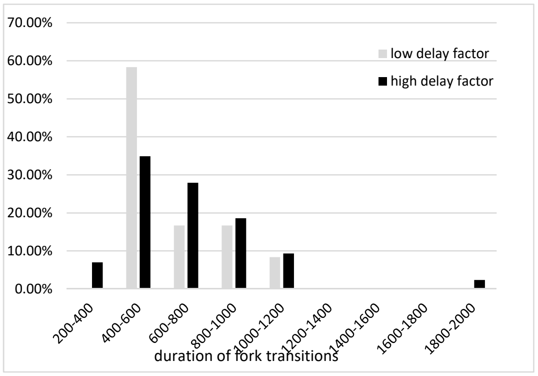
We have also noticed a number of tip transitions resulting from forks, during which the network has been partitioned into two or even more node subsets that have chosen different blocks as their new main tip. However, such transitions last



(a) Fork: block 59541951 is replaced with blocks 90979739 by some nodes, and block 77968842 by others, as the main tip of individual node blockchains. Fork is resolved by the arrival of block 9920562 which all nodes append as the new main tip. Vertical axis shows the number of nodes in respective partitions.



(b) Block 79866509 is replaced by block 59541951 as the second-highest block in all blockchains; the latter is subsequently replaced with block 90979739. Vertical axis shows the number of nodes in respective partitions.



(c) Duration of fork transitions for low (gray) and high (black) values of delay factor δ.

Fig. 7. Pertaining to fork transitions. All times in seconds.

much longer than the regular ones – about two magnitudes of order longer.

A sample fork transition with two competing main tips is shown in Fig. 7(a); the delay factor was $\delta = 0.75$. As can be seen, partitioning of the network in terms of main tips is achieved rather quickly, in about five seconds or so, which is close to the mean time needed to reach 99% of the nodes in Fig. 4(a). Once the steady state is reached, the partitions remain stable for a long time, more than 500 seconds in the example shown.

Interestingly enough, the previous node transitioning shown in Fig. 7(b) still behaves in a way that's very similar, or nearly identical, to that of the regular transition case. This is to be expected since both competing blocks refer to the former main tip as their previous block.

As expected, competition is resolved only when another newly mined block is received, because it's only at that moment that Algorithm 1 can then choose the branch with the largest height, i.e., distance from the genesis block, as the valid chain, and its tip as the new main tip. Alternatively, the branch with the greatest accumulated work can be chosen [23]. In the example shown, once the new block arrives, it actually propagates to all the nodes rather quickly; the corresponding time of about four seconds is close to the 50% propagation time from Fig. 4(a). Note that, due to a large number of nodes in the network, partitions with small sizes cannot be seen on the diagram.

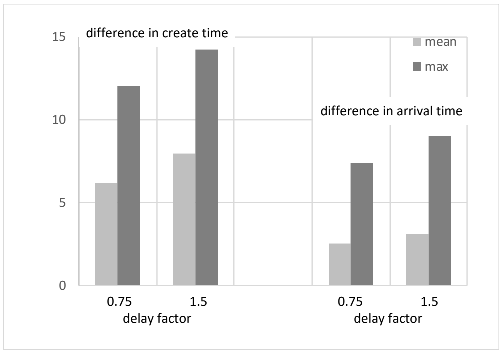
However, the time to resolve a fork is much longer than the regular transition time, as can be seen from Fig. 7(c). Roughly a half of the forks observed in our experiments were resolved in less than 800 seconds, but a non-negligible portion lasted for more than 1000 seconds.

D. On conditions that lead to a fork

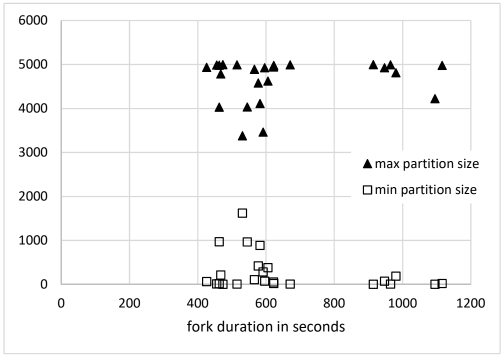
Forks occur when two or more blocks are propagated through the network at about the same time. To establish the conditions for a fork to occur, we have measured the time difference between the competing blocks, both between their creation time as recorded in their timestamps and their arrival times as recorded by the receiving node.

Mean and maximum values of time difference between block creation and block reception are shown in Fig. 8(a). As can be seen, blocks that are created (i.e., confirmed mined and sent out to the network) within about 6 to 8 seconds, on the average, result in a fork. Maximum time difference between block creation, i.e., mining, is about 12 and 14 seconds, for the low and high delay factor, respectively. Note that the information about block creation is readily available in our simulation model in which clocks of all the nodes were synchronized. However, this information cannot be reliably obtained in the real Bitcoin network due to its decentralized structure and, possibly, unsynchronized clocks. In fact, the Bitcoin protocol allows blocks with timestamps of up to two hours in the future to be accepted as valid.

Of more interest, thus, is the difference in block arrival time which is recorded at the receiving node; after all, a fork occurs when a block is received at a node, not when it is sent out to the network. As can be seen from Fig. 8(a), mean time difference between the blocks is about 2.5 and 3 seconds for the lower and higher delay factor, respectively, each of which is less than half of the corresponding mean time difference for creation time. More importantly, the maximum time difference between block arrivals that result in a fork is about 7.5 and 9



(a) Mean and maximum values of creation and arrival time differences (in seconds) between competing blocks.



(b) Minimum and maximum partition size vs. fork duration, $\delta = 0.75$.

Fig. 8. Characteristics of forks.

seconds for the lower and higher delay factor, respectively. Both values are smaller than the corresponding maximum difference of block creation times.

In our experiments most forks were of size two, i.e., one main tip and one side tip, and of length one; there were a couple forks of size three and four, and even one fork of size five! However, apart from the fact that multi-way forks are indeed possible, we cannot draw any definite conclusion due to the limited size of our sample. In particular, there is no discernible correlation between the size of the partitions created by the fork and the duration of the fork, as can be seen from the diagram of the sizes of minimum and maximum node partitions vs. fork duration shown in Fig. 8(b).

VII. Conclusion and directions for further research

In this paper we have examined some characteristics of forks in the Bitcoin blockchain and the mechanism of fork creation. We have shown that regular transitions typically finish within a short time, up to 10 or 12 seconds at most, depending on the network delays. Fork transitions last much longer, with some samples lasting as much as 1000 seconds or more, as they only get resolved by the arrival of a new block. We found no discernible correlation between the duration of a fork transition and the number and individual sizes of network partitions caused by the fork.

Our future work will focus on investigating the impact of network connectivity and topology, as well as of the geographical distribution of the nodes and their hashing power, on block propagation and frequency and characteristics of forks. We will also analyze some of the attacks such as double spending and reorganization attacks and ways to prevent them.

References

[1] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "SoK: Consensus in the age of blockchains," *arXiv preprint arXiv:1711.03936*, 2017.

[2] S. Ben Mariem, P. Casas, and B. Donnet, "Vivisecting blockchain P2P networks: Unveiling the Bitcoin IP network," in *ACM CoNEXT Student Workshop*, 2018.

[3] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Block arrivals in the Bitcoin blockchain," *arXiv preprint arXiv:1801.07447*, 2018.

[4] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *Proc. 13th IEEE Int. Conf. Peer-to-Peer Computing (P2P'13)*, vol. 26, 2013.

[5] S. Delgado-Segura, S. Bakshi, C. Pérez-Solà, J. Litton, A. Pachulski, A. Miller, and B. Bhattacharjee, "TxProbe: Discovering Bitcoin's network topology using orphan transactions," *arXiv preprint arXiv:1812.00942*, 2018.

[6] S. Delgado-Segura, C. Pérez-Solà, J. Herrera-Joancomartí, G. Navarro-Arribas, and J. Borrell, "Cryptocurrency networks: A new P2P paradigm," *Mobile Information Systems*, 2018.

[7] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí, "The Bitcoin P2P network," in *Int. Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 87–102.

[8] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *arXiv preprint arXiv:1311.0243*, 2013.

[9] J. Göbel, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, vol. 104, pp. 23–41, 2016.

[10] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conf. Theory Appl. of Cryptography*. Springer, 1990, pp. 437–455.

[11] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA, 2000, pp. 565–574.

[12] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring The Internet*, 6th ed. Boston, MA: Addison-Wesley Longman, 2016.

[13] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. McGraw-Hill, 2015.

[14] M. Lischke and B. Fabian, "Analyzing the Bitcoin network: The first four years," *Future Internet*, vol. 8, no. 1, p. 7, 2016.

[15] R. C. Merkle, "Protocols for public key cryptosystems," in *Security and Privacy, 1980 IEEE Symposium on*. IEEE, 1980, pp. 122–122.

[16] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering Bitcoin's public topology and influential nodes," report, 2015.

[17] J. Mišić, V. B. Mišić, X. Chang, S. G. Motlagh, and M. Z. Ali, "Block delivery time in Bitcoin distribution network," in *IEEE Int. Conf. Communications ICC 2019*, Shanghai, China, May 2019.

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

[19] T. Neudecker, P. Andelfinger, and H. Hartenstein, "A simulation model for analysis of attacks on the Bitcoin peer-to-peer network," in *IFIP/IEEE Int. Symp. Integrated Network Management (IM)*, 2015, pp. 1327–1332.

[20] T. Neudecker and H. Hartenstein, "Network layer aspects of permissionless blockchains," *IEEE Communications Surveys Tutorials*, 2018, 10.1109/COMST.2018.2852480.

[21] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *IEEE INFOCOM*, 2018, pp. 2546–2554.

[22] G. Pappalardo, G. Caldarelli, and T. Aste, "The Bitcoin peers network," in *2nd Int. Workshop P2P Financial Systems*, London, UK, Sep. 2016.

[23] Y. Sompolinsky and A. Zohar, "Accelerating Bitcoin's transaction processing. fast money grows on trees, not chains." *IACR Cryptology ePrint Archive*, vol. 2013, no. 881, 2013, https://ia.cr/2013/881.

[24] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125.