Blockchain: Research and Applications 3 (2022) 100048

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

# Security and performance evaluation of master node protocol based reputation blockchain in the bitcoin network

Muntadher Sallal [a,*], Gareth Owenson [b,c], Dawood Salman [d], Mo Adda [c]

[a] Bournemouth University, Poole, Dorset, BH12 5BB, UK
[b] Cyber Espion Ltd., Port Solent, Hampshire, PO6 4TY, UK
[c] University of Portsmouth, Portsmouth, Hampshire, PO1 2UP, UK
[d] University of Kufa, Kufa, Najaf Governorate, Iraq

ARTICLE INFO

*Keywords:*
Information propagation delay
Network clustering
Bitcoin network
Reputation blockchain

ABSTRACT

Bitcoin is a digital currency based on a peer-to-peer network to propagate and verify transactions. Bitcoin is gaining wider adoption than any previous crypto-currency. However, the mechanism of peers randomly choosing logical neighbours without any knowledge about the underlying physical topology can cause a delay overhead in information propagation which makes the system vulnerable to double spend attacks. Aiming at alleviating the propagation delay problem, this paper introduces a proximity-aware extension to the current Bitcoin protocol, named Master Node Based Clustering (MNBC). The ultimate purpose of the proposed protocol, which is based on how clusters are formulated and how nodes can define their membership, is to improve the information propagation delay in the Bitcoin network. In the MNBC protocol, physical internet connectivity increases as well as the number of hops between nodes decreases through assigning nodes to be responsible for maintaining clusters based on physical Internet proximity. Furthermore, a reputation-based blockchain protocol is integrated with MNBC protocol in order to securely assign a master node for every cluster. We validate our proposed methods through a set of simulation experiments and the findings show how the proposed methods run and their impact in optimising the transaction propagation delay.

## 1. Introduction

Bitcoin is the first digital currency to attract mainstream, businesses, and people's attention. Bitcoin is a virtual, decentralised cryptography that no one is in charge of it and it is not tracked by any hard asset or government. Instead, Bitcoin relies on a peer-to-peer (P2P) network that protects the Bitcoin's value by means of cryptography that is performed by peers mining Bitcoins through brute-forcing double SHA-256 hash function. The concept of Bitcoin was proposed by an anonymous programmer using the pseudonymous Satoshi Nakamoto [1]. Due to the Bitcoin advantages such as the absence of intermediates and its decentralised architecture, Bitcoin is now deployed as a currency by many businesses and companies. Bitcoin performs global transactions which allow non-refundable, reasonably fast money transfers to any part of the world [2].

Based on the publicly distributed ledger that is shared by the entire Bitcoin network nodes, a distributed trust mechanism is achieved [3]. This mechanism is considered as a monitoring technique by which the amount of available bitcoins[1] in Bitcoin will be tracked. To achieve this mechanism, two main requirements need to be fulfilled: (i) transactions verification process has to be achieved in a distributed manner to ensure the validity of transactions; and (ii) successfully processed transactions have to be quickly announced to everyone in order to guarantee the consistent state of the blockchain [4]. As transactions are validated

* Corresponding author.
*E-mail addresses:* msallal@bournemouth.ac.uk (M. Sallal), gareth.owenson@port.ac.uk (G. Owenson), Dawood.jasim@uokufa.edu.iq (D. Salman), mo.adda@port.ac.uk (M. Adda).

Production and Hosting by Elsevier on behalf of KeAi

[1] The term "bitcoin" refers to the actual currency, while "Bitcoin" indicates the whole Bitcoin system.

https://doi.org/10.1016/j.bcra.2021.100048
Received 19 December 2020; Received in revised form 15 September 2021; Accepted 6 December 2021

against the blockchain, reaching a consistent state over the blockchain is considered as a fundamental requirement towards achieving the distributed transactions verification process. Upon achieving the transaction verification process, a Bitcoin transaction has to be broadcasted to all nodes within the network with the aim of reaching a consensus about which transactions are valid. Eventually, this consensus will be recorded in the blockchain. As the probability of reaching a global state in the blockchain is mainly affected by how quickly the Bitcoin information (transactions/blocks) are announced to every node in the Bitcoin network, the main goal of the Bitcoin network is to propagate Bitcoin information to the entire nodes as quickly as possible. Unfortunately, a delay in information propagation is experienced during the transaction verification process which results in the inconsistent blockchain. This makes Bitcoin vulnerable to certain classes of attacks.

As the Bitcoin network topology is not proximity defined, connecting to other peers is maintained randomly without considering any proximity criteria. In other words, long-distance links are not taken into consideration when the Bitcoin physical network topology is built. This increases the non-compulsory hops that the information passes through. In addition, the sheer distance between the origin of a transaction or block and other nodes causes delays in the transaction verification process [3,5]. This delay increases the possibility of performing successful double spending attacks, which are more difficult to detect in slow networks, due to the conflict between nodes regarding the transactions history. Uncertainty regarding the validity of a given transaction reduces the chances of achieving a consensus on the same blockchain header, which would cause blockchain forks.

Forks are created when two blocks are possible to be created simultaneously, each one as a possible addition to the same sub-chain. According to Refs. [1,6], a transaction can appear in two different branches of the blockchain. Though, as discussed in Ref. [7], in the special case where the Bitcoin is subject to the blockchain forks, attackers might be able to impose their own transactions history, possibly to reverse transactions they sent so as to successfully perform double spending attacks [8]. Specifically, attackers can secretly mine a branch that includes the transaction that returns the payment to themselves, while disseminating the merchant's transaction. However, blockchain forks are caused by the delay overhead of information propagation [3]. Therefore, the propagation delay between nodes in the Bitcoin network is critical even though the probability of reaching an agreement about transactions history is high [9,10].

Aiming to find the optimal solution to such problems, we proposed, implemented, and evaluated in our previous work [11] a new network clustering protocol, named Master Node Based Clustering (MNBC). In order to provide a comprehensive implementation and evaluation, in this paper we extend our previous work [11] by integrating the MNBC protocol with a new proposed reputation scheme-based blockchain. The reputation scheme aims to increase the security level of the master peer selection process in which trusted nodes are elected to lead several proximity clusters.

The rest of the paper is organized as follows. In Section 2, related work in speeding up information propagation in the Bitcoin network and in modelling approaches to avoid double spending attacks will be critically discussed. Section 3 presents the problem statement and the contributions of this paper. In Section 4, we briefly describe the Bitcoin networking aspects as well as discuss in detail the information propagation in the Bitcoin network. Section 5 details the proposed clustering protocol and discusses its main components. In Section 6, we explain the system design and how the main components of the proposed protocol work. In Section 7, we explain the experimental setup in relation to the performance evaluation of the proposed protocols. In addition, the performance evaluation results of the proposed protocols are performed. Security evaluation of the proposed protocols is conducted in Section 8. Furthermore, security evaluation results with reference to partition attacks are provided. We conclude the work in Section 9.

## 2. Related work

Previous works that are proposed with the aim of mitigating the information propagation delay in the Bitcoin network will be critically discussed in this section under three categories: minimise verification, pipelining information propagation, and connectivity increase. Moreover, existing methods to tackle double spending attacks in the Bitcoin network will be highlighted in this section.

### 2.1. Minimise verification

There have been several investigations that aim to reduce the information propagation delay throughout minimising the time of information (transactions/blocks) verification. In the Bitcoin network, when a node receives a transaction/block, it verifies whether it is valid or not. If the transaction/block is valid, the node forwards it to its neighbours. Otherwise, invalid transactions/blocks are discarded. The idea of reducing the information verification time, in particular, block verification time has been adopted in Ref. [3] where minimise verification protocol has been proposed as a way to speed up information propagation. The protocol has stated some changes in the behaviour of Bitcoin nodes, which make every node fulfills only the first part of the block verification process. Specifically, when a node receives a block, it checks the proof of work difficulty and forwards the block to its neighbours, rather than suspends the relay until the validation of all transactions in the block is completed. This would minimise the block propagation delay in the Bitcoin network. However, the change in the nodes' behaviour mentioned above is more likely to bring a security risk as discarding transactions validation would give a great chance to an attacker to flood the network with invalid transactions which, on the other hand, results in a distributed denial of service attack. In addition, the change in the nodes' behaviour does not take into account the transaction propagation delay which means that transactions would be propagated following the original information broadcasting scenario. As a result, the change does not have a large impact on the overall information propagation delay.

Another theory has been proposed in Ref. [12] which focuses on the blockchain as a main factor in reducing the transaction verification time. As transactions are validated against the blockchain that contains a history of all transactions, and it still grows in size with each new transaction, it has been claimed that reducing transactions history at each node plays an important role towards achieving an optimal transaction verification time. Precisely, a new algorithm, known as BASELINE, has been proposed in Ref. [12], in which the blockchain is divided at each node in the Bitcoin network into several parts $n$. These parts are distributed at each node on several local computers. As all parts represent the same user, public/private keys are the same for all parts. On the other hand, each part has a different portion of the public ledger. Evaluation results in Ref. [12] have shown that the verification time could be enhanced by 71.42% if the blockchain is divided at a given node on five computers. This means that an improvement in the information propagation delay could be achieved when the number of divisions at each node is greater. However, the proposed BASELINE algorithm is less likely to be adopted as a realistic solution due to the expensive requirements where every node in the network should maintain several local computers. In the same context where some research focused on speeding up information propagation in conjunction with minimising the blockchain size [6], proposed a new approach that improves the scalability of the blockchain by performing more security for off-chain blocks through miners. More precisely, miners would have the responsibility to keep track and protect the soft forks that are linked to the main blockchain. In principle, this approach considers miners as a trusted third-party and gives them more control over the Bitcoin network. Therefore, this approach stands against the decentralisation concept of Bitcoin, resulting in minimising security awareness. Furthermore, these soft forks are subject to the 51% attack due to the less hash rate.

On the other hand, the Blinkchain approach is introduced in Ref. [13] which focuses on minimising the transaction verification time with the aim of decreasing the consensus latency. The Blinkchain approach is based on splitting the blockchain into localised shards, one blockchain per geographical location. Each blockchain is associated with a number of nearby validators. This reduces the transactions history on each blockchain which results in speeding up the transaction verification process. However, this approach reduces the resistance of blockchain against 51% attacks as these blockchains offer less hash rate. Furthermore, this approach doesn't offer any interoperability technique that allows shard blockchains to interact with each other.

In the same context, a sharding approach is also introduced in Rapidchain [14] to scale up the blockchain. In Rapidchain, the blockchain network is divided into random shards where each shard randomly selects a leader node. However, shards in Rapidchain are not proximity defined and network information still need to travel long distances. Furthermore, Rapidchain selects a leader for every shard without forcing those leaders to fulfill specific requirements, this is very crucial from a security point of view.

### 2.2. Pipelining information propagation

As introduced in Ref. [5], faster information propagation can be achieved by pipelining information dissemination with the aim of minimising the round-trip times between nodes and their neighbours. Specifically, this solution claims that an incoming INV message which includes a list of hashes of the available transactions, can be immediately forwarded to the rest of the network nodes instead of waiting to receive transactions. Therefore, nodes can ask for a transaction even though it has not arrived yet. On receiving the transaction, it will be forwarded immediately to the nodes that have asked for it, considering that a GETDATA message has already been received from those nodes. By doing this, the idle time in which nodes are normally waiting for the GETDATA message to arrive would be utilized. The key problem with the pipelining propagation protocol is that the global state of the Bitcoin network might become inconsistent when nodes request a transaction that is not available. As a result, an inconsistent network results in increasing the chances of performing successful double spending attacks. Furthermore, the pipelining propagation protocol requires unlimited memory at every node with the aim of keeping either transactions until a GETDATA message arrives, or GETDATA messages until transactions arrive. Moreover, it is believed that this theory is able to reduce the information propagation delay with a very low rate as transactions still need to pass through random and unlocalized connections to visit most of the Bitcoin network nodes.

In Ref. [15], a new pipeline method, named compact block relaying (CBR), is introduced to mitigate the propagation delay problem. Specifically, a compact block that includes only hashes of transactions in the block is announced to other nodes. Upon receiving the compact block, only missing transactions will be transmitted to the receiver rather than the whole block. Even though the CBR improves the propagation delay, the nodes still need to have the compact block in hand before forwarding it further. Therefore, CBR can cause large latency, especially when transmitting CBR of large size.

Falcon, a new propagation protocol proposed in Ref. [16], minimizes the propagation delay by following the cut and forwarding strategy in which reception and forwarding of the compact block are handled in parallel. However, Falcon does not rely on the existing Bitcoin nodes, instead, it deploys relay nodes to implement the cut-through forwarding protocol. In addition, Falcon is a commercial protocol that lacks in-depth analysis.

### 2.3. Connectivity increase

As it is mentioned earlier, the sheer distance between the initiator of a block or transaction and nodes is considered the most causative factor of the information propagation delay in the Bitcoin network. Studies in Ref. [3] claimed that increasing the network connectivity throughout minimising the distance between any two nodes, which can be done by creating a star sub-graph topology that is able to form a central communication hub, has a large effect on the reduction of the information propagation delay. Specifically, a new network topology is proposed in Ref. [3] in which each node maintains a connection pool that is able to keep up to 4000 open connections. As a result, the node mostly connects to every single advertised address. Therefore, information would visit a fewer number of hops which reflects faster information propagation. However, the Bitcoin protocol allows nodes to maintain up to 8 outgoing connections in order to prevent controlling the network by malicious nodes [17]. Though, the proposed network topology raises severe security risks due to the fact that nodes are permitted to maintain many connections to other nodes. Therefore, malicious nodes might be able to control and easily disturb the network.

On the basis of maximising the proximity of connectivity, another change in the Bitcoin network topology has been proposed in Ref. [5]. This change increases the geographical connectivity in the Bitcoin network through several coordinator nodes, known as CDN Bitcoin client, which is distributed strategically around the globe. These CDN clients are able to search and suggest Bitcoin network nodes to each other based on the geographical location. Specifically, a CDN client is able to calculate the geographical distance between the discovered nodes and other CDN clients. By doing this, the CDN client is able to recommend the geographically closest nodes to other CDN clients. Compared to the protocol that is proposed in Ref. [3], CDN clients are allowed to maintain many (up to 100) outgoing connections to the nodes that are geographically close.

The main downside of this solution is that any node can be a CDN client which makes the Bitcoin network more vulnerable to some classes of attacks. Specifically, malicious nodes can easily impersonate the role of CDN clients and maintain connections to many nodes in the network. This results in malicious nodes being able to control a large portion of the network. As a result, the Bitcoin network would be vulnerable to distributed denial-of-service (DDOS) attacks and partition attacks. Another concern raised in relation to the CDN client protocol is that it is relatively centralised as any CDN client can be used as a coordinator node without meeting any requirements or achieving an agreement over the network nodes. Furthermore, the idea of recommending closer nodes to other nodes will not have a high impact on the overall network connectivity if it is implemented by limited nodes that are not well connected.

A new transport protocol layer, known as FIBER (Fast Internet Bitcoin Relay Engine), is introduced in Ref. [18] to reduce the information propagation delay. FIBER focuses on the reduction of the delay caused by the packet loss throughout using UDP with forward error correction. FIBER also reduces the network traffic by using data compression. However, our work introduces a new Bitcoin network protocol that can be easily integrated with FIBER.

### 2.4. Mitigating double spending attacks

Some research has been done towards analysing and mitigating double spending attacks with respect to both scenarios, 0-confirmations[2] and N-confirmations[3]. Regarding N-confirmation double spending attacks, the probability of performing successful double spending attacks on the Bitcoin network has been provided in Ref. [3] throughout developing an analytical model of Bitcoin. Furthermore, a strong correlation between the size of a message and the propagation delay has been

[2] Double spending attack where the blockchain accidental forks do not play any role as the blockchain is not checked at all.

[3] Double spending attack where an attacker requires to control 50% of the computing available in the network.

observed. As an adversarial fork of the blockchain still causes a possibility of double spending, some research has admitted that reducing the possibility of accidental forks would help in double spending attack avoidance [6,7].

In the context of 0-confirmation, a model which considers some modifications in the transaction dissemination protocol has been presented in Refs. [7,19]. The main intuition behind these modifications is to mitigate double spending attacks in fast payments. In Ref. [7], a new model was proposed which allows the vendor to receive $T_K$ (a conflicting transaction) and $T_v$ (an honest transaction that is sent to the vendor) almost at the same time. This would help the vendor to discover double spending attacks at the right time before delivering the products. Specifically, the core idea of this model is that when a transaction is received by a node that has not been seen before, the node adds the transaction to its pool and forwards it to the other nodes. Otherwise, it directly forwards the transaction to other neighbours without adding it to its pool. This scenario allows the conflicting transaction $T_k$ to be received by the vendor before delivering the products. Though, the vendor would immediately detect the attempt of a double spending attack when the conflicting transaction $T_K$ is received. The most serious disadvantage of this method is that a large volume of nonessential traffic would flood the network which results in an inefficient performance of the Bitcoin network.

As a realistic solution, a prototype system which is applied in vending machines was proposed in Ref. [19]. This system has performed a fast payment with 0.088 as a probability of double spending attacks through setting up a server that observes transactions. This server gives a signal, which indicates that a transaction has been confirmed to the blockchain when the transaction is propagated and reached over 40 nodes. Unfortunately, this solution is limited because an attacker's transaction could still be propagated to the majority of nodes. That disproves the claim of considering a transaction is approved if it is received by 40 nodes.

### 2.5. Reputation based blockchain

In P2P networks, reputation is a fundamental feature that increases the level of trust. Recently, research has focused on using blockchain to achieve a secure and efficient reputation scheme in P2P networks. Since it is common to leverage reputation as an incentive mechanism, CertChain [20] proposed a reputation mechanism that ranks peers based on consensus and incentive mechanism, which takes economic benefits and misbehaviour into consideration. However, this work aims to certify authorities (CA), rather than address any scalability issue. In PoT [21], the reputation is maintained based on trust reported by every node in the network. However, reputation in PoT relies on specific identified trusted nodes in the network. RepuCoin [22] proposes a reputation scheme that is based on weighting consensus. However, this scheme relies on the amount of work that is done from the very beginning of the chain to calculate the reputation score. This increases the probability of double spending attack occurrence if high reputation peers collude. In Repchain [23], a reputation-based sharding is proposed to increase the transactions throughput. However, this scheme divides the network into shards following the same mechanism as in Rapidchain [14], where shards are not proximity defined.

## 3. Problem statement and summary of contributions

As we highlighted above, the information propagation delay is a serious problem facing the Bitcoin network nowadays and several methods have been proposed in order to fix this issue. However, previous attempts of updating the network topology have not taken into account any clustering approach. Instead, these attempts have considered either increasing the network connectivity by maintaining a mesh network topology [3], or relying on several coordinator nodes to support proximity of connectivity in the network without paying attention to the security risks [5]. Furthermore, several sharding approaches were introduced to scale up the Bitcoin blockchain without focusing on the propagation delay issue caused by the share distance between Bitcoin network nodes [14,23]. We believe that there is plenty of room for improvement in terms of speeding up information propagation in the Bitcoin network. In this respect, we aim to evaluate the impact of a novel network clustering approach based reputation scheme on improving the propagation delay in the Bitcoin network.

The main aim of this research is to determine '*Can clustering-based master node in the Bitcoin network improve the information propagation delay without compromising security?*'.

With the above context in mind, we can summarise the main contributions of this paper as follows:

- **MNBC protocol**: In this paper, we propose a new model that integrates a proximity based clustering approach with a newly developed reputation score based blockchain. The MNBC protocol relies on several nodes, known as master nodes, to achieve fully connected clusters based on the physical Internet proximity and random peers selection. To increase the security level, master nodes are selected based on a reputation score which is calculated by the reputation scheme proposed in this paper. The main aim of the proposed model is to mitigate the propagation delay problem in the Bitcoin network without compromising security.
- **Performance Evaluation**: The other contribution of this paper is to evaluate the performance and effectiveness of the proposed model against the average latencies of the information delivery between peers in the Bitcoin network without compromising security.
- **Security Evaluation**: As undertaking clustering in the Bitcoin network is different from clustering within other classes of the P2P network due to the strict requirements of security, this research examines whether the proposed clustering protocol can be done safely without increasing the likelihood of certain classes of attacks, in particular, partitioning attacks and Observe-Act attack.
- **Simulations**: To enable the evaluation of the proposed clustering protocols, several simulations were developed using the simulation model that was developed [24]. To parameterise the simulation model, large-scale measurements of the real Bitcoin network parameters that have a direct impact on client behaviour and information propagation in the real Bitcoin network, are performed. Furthermore, measurements of the transaction propagation delay in the Bitcoin network are presented in this paper. These measurements are collected using a methodology by which the transaction propagation delays are accurately measured. These measurements offered an opportunity to validate the developed simulator against the real Bitcoin network.

## 4. Background knowledge

### 4.1. Bitcoin network structure

The Bitcoin network refers to a group of nodes that handle the Bitcoin protocol. Bitcoin is built on a decentralised structure which is considered one of the key features of Bitcoin. Though, there is no centralised server that the Bitcoin architecture relies on. Instead, a distributed protocol has been maintained to support the system [25]. In this network, each peer runs the Bitcoin protocol and connects with other peers over a TCP channel [26]. As the Bitcoin network topology is not proximity defined, connecting to other peers is maintained randomly. In addition, every node should maintain a maximum of 8 outgoing connections to peers and accept up to 117 connections [27]. Nodes can join and leave the network at any time and when a node re-joins, it asks other nodes for new blocks to complete its local copy of the blockchain [28]. For the purpose of making denial of service impractical, just the valid information (transactions and blocks) are propagated, whereas invalid transactions and blocks are discarded. Bitcoin network nodes are classified into two groups, servers which can accept incoming connections and those which cannot (clients), because they are behind NAT or firewall [29].

The Bitcoin nodes take different roles in the network based on the functionality that those nodes support such as wallet services, routing, etc. As Bitcoin relies on distributed validation, an essential role in which transactions are validated in a distributed manner, is covered by all nodes in the network [27]. In order to participate in the Bitcoin network, all nodes have to perform the routing function. This function includes validating and propagating transactions, and maintaining connections to other nodes.

### 4.2. Bitcoin network discovery

When a node *N* joins the Bitcoin network for the first time, a discovery mechanism that does not consider any proximity criteria is adopted to find other nodes in the network. As a first step, at least one existing Bitcoin node needs to be discovered by node *N* in order to discover more nodes [30]. After that, more connections will be established between node *N* and the nodes that are discovered. Establishing connections to other nodes is done without taking into account any proximity priority as the Bitcoin network topology is not proximity defined [3,5]. To establish a TCP connection, a handshake with a known peer is handled by sending a version message which contains basic identifying information. A peer responds back to the version message by sending a verack message. Each peer holds a list of IPs of peers that are connected to it. To stop peers misbehaving, each node handles a penalty score mechanism for each node connected to it. The score is increased when unreliable behaviour is announced. When the score reaches 100, the misbehaving IP is banned by the node that handles the penalty score. Furthermore, a transactions pool is maintained by each node, which includes transactions that wait to be verified and relayed to the neighbouring nodes [26].

On the question of how a new node discovers the first node in the network, there are some stable nodes that behave as seed nodes listed in the Bitcoin client that could suggest to the new node some other nodes in the network [27]. Specifically, bootstrapping that needs to be handled by the new node, requires at least one IP address of a Bitcoin network node which is known as DNS seed node. After maintaining a connection with the seed node, further introductions to other nodes will be handled. Then, more connections to other nodes will be established and the new node will disconnect from the seed node. However, connecting to other nodes would help the new joining node in discovering more nodes. This can be done by sending an Addr message which includes the IP address of the sender node. Precisely, the newly connected node can advertise its own IP to other nodes by sending an Addr message to its neighbours. This helps the new node to be found by other nodes. On the other hand, the new node can get to know other nodes by sending a Getaddr message to its neighbours. As illustrated in Fig. 1, neighbours respond to the Getaddr message by sending a list of IP addresses of the other nodes in the network [30].

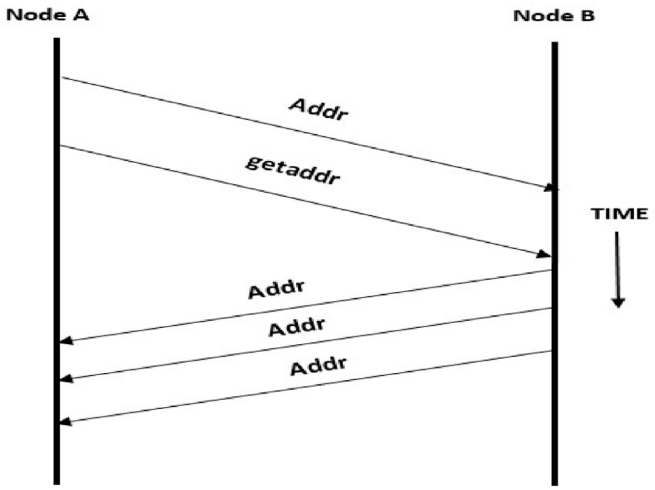


**Fig. 1.** Dissemination of Addr message between two peers.

Even though each node establishes connections to other nodes, the node should continue discovering more nodes and advertise its existence to the newly joined nodes [26]. This is due to unreliable paths as nodes come and go in the network in a random way. Therefore, a node that connects to other nodes does not guarantee that these connections will not be lost. However, discovering other nodes continues to operate with the aim of offering diverse paths into the Bitcoin network. When a node reboots, it can re-join the network without needing to bootstrap the network again as the node can still remember the most recent successful nodes connections, so the node tries to reestablish connections to those nodes by sending connection requests. While there are no responses to these requests, the node starts bootstrapping the network again. In terms of a connection that has no traffic going through for more than 90 min, the connection will be dropped off [30].

### 4.3. DNS seed nodes in the Bitcoin network

As it is mentioned earlier, a Bitcoin DNS seeder is a server that assists nodes in discovering active peers in the Bitcoin network. Therefore, the DNS seeder responds to the DNS query by initiating a message that contains a list of IPs. The maximum number of IPs that can be attached to the message is limited by constraints on DNS, around 4000 messages can be returned by a single DNS query [27]. In the Bitcoin network, there are six DNS seeds that periodically crawl the entire network in order to obtain active IP addresses. However, there are two scenarios where DNS seeders are queried by other nodes. The first scenario happens when a node joins the network for the first time and tries to connect to the active IPs. While in the second scenario, the DNS seeder is queried by a node that restarts and attempts to reconnect to new peers. In this case, the DNS query is initialised after 11 s since the node attempted to reconnect and has less than two outgoing connections [27].

### 4.4. Bitcoin protocol & information propagation

The Bitcoin protocol achieves the distributed validation based on a replicated ledger that is collectively implemented by network volunteries. This ledger tracks the address balances of all users. An arbitrary number of addresses can be created by each user to send and receive bitcoins. An ECDSA key pair is used to prove the ownership of bitcoins associated with that address. Each entry in the public ledger represents a transaction which is a signed data structure that is created by a Bitcoin user who intends to send a specific bitcoin to one or more destination accounts [26]. Transactions are responsible for claiming some bitcoins that are associated with the address of the sending party and reassigning them to the address of the receiving part/parties. Transactions are represented by a hash of the previous transaction that has been sent before as well as the public key of the future owner. Each transaction includes an input and output. For combining or splitting bitcoins, transactions can handle multiple inputs and outputs. Inputs reference the funds from other previous transactions, whereas outputs indicate the transferred bitcoins. A transaction output also indicates the new owner of the transferred bitcoins when it is referenced as an input in a future transaction. Though, the balance of an account is the sum of all the values of all unspent outputs owned by that account. The sum of all outputs should be equal to or less than the sum of all inputs [27].

By propagating transactions and blocks, nodes synchronise their replicas of the public ledger. To avoid sending a transaction to a node that already received it from other nodes, the transaction is not forwarded directly. Instead, the transaction availability is announced first to nodes once the transaction has been verified as shown in Fig. 2. This can be done by propagating an INV message that contains the hash of the transaction [31]. On receiving an INV message, a node checks whether the transaction has been received before. If it has not been seen before, the node requests the transaction by sending a GETDATA message.

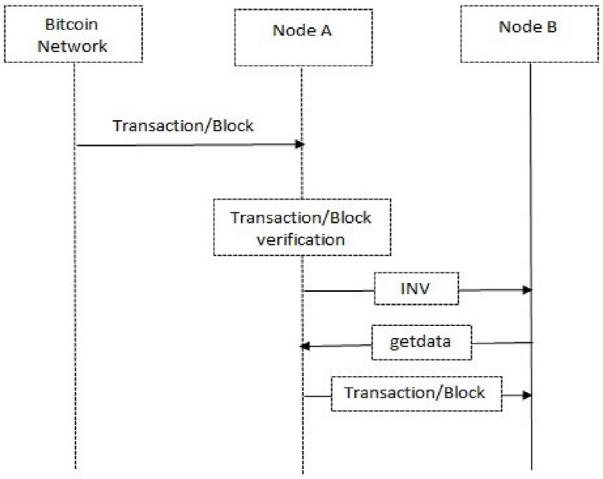


**Fig. 2.** Information propagation mechanism between Nodes A and B.

Responding to the received GETDATA message, a node sends the transaction's data. Valid received transactions will be collected and included in a block by a node that generates blocks. A block availability will be announced to other nodes, as explained in Fig. 2, following the same mechanism of transactions availability announcement. However, a delay in transactions propagation occurs which is caused by the information broadcasting scenario [5].

## 5. Master Node Based Clustering (MNBC)

The MNBC protocol extends the Bitcoin Clustering Based Super Node (BCBSN) protocol that was proposed in our previous work [32], with the aim of addressing the security and performance limitations of the BCBSN protocol. As it is mentioned in Ref. [32], the BCBSN protocol aims to generate a set of geographically diverse clusters in the Bitcoin network by exploiting super peers technology. Within each cluster, the BCBSN protocol assigns one node to be a super peer that is responsible for maintaining the cluster and broadcasting information on the Bitcoin network. In the BCBSN protocol, clusters are fully connected via super peers only. Due to this, the information flow between clusters in the BCBSN protocol is only fulfilled through super peers. Furthermore, super peers in the BCBSN protocol group peers based on their geographical location in order to increase the proximity of connectivity in the network. However, the long-link distance might be applied between any two peers even though they are in the same geographical location. The node selection in the BCBSN protocol is not random, instead, the node is forced to connect to the list of nodes that are supplied by the super peer that the node connects to. From a security point of view, the level of security awareness in the BCBSN protocol can be improved if more nodes between clusters are maintained as well as random selections of peers which are important in the Bitcoin network is preserved. This improves the network resistance against the partitioning attack as well as the eclipse attack.

The limitations of the BCBSN protocol mentioned above have motivated the development of a new protocol that overcomes the lack of connection channels between clusters as well as considers random selections of peers based on the physical Internet distance rather than purely based on the geographical location. Specifically, the new protocol, named MNBC, relies on several nodes, known as master nodes, to achieve fully connected clusters based on the physical Internet proximity and random peers selection, where information can be exchanged between clusters via master nodes as well as normal nodes. The idea of the MNBC protocol is inspired by the master node technology that was originally adopted in Ref. [33]. However, selecting master nodes in Darkcoin does not require conditions to be fulfilled in order to preserve security. Whereas master nodes in the MNBC protocol are selected through applying a selection phase that requires a reputation score to be achieved in order to cover the role of master nodes.

### 5.1. System model

We assume that the Bitcoin network starts with one cluster which includes normal nodes as well as $n$ number of validators $V$ where $V_n = V1, V2, \ldots, V_n$. After that, the network will be divided into clusters $Cs$ using a clustering approach based on physical proximity proposed in Ref. [34]. This results in $s$ number of proximity clusters $C$ where $Cs = C1, C2, \ldots, Cs$. Within an epoch, MNBC randomly distributes validators over the generated proximity clusters as well as runs the master peer selection process to elect the master peer for each cluster. This results in $Cs = C1, C2, \ldots, Cs$ with $r = n/s$ validators in each cluster, including one master node and $r-1$ members. In MNBC, an epoch $e$ denotes the time interval between events of validator assignment to clusters and master node selection. MNBC relies on fresh randomness to randomly assign validators to proximity clusters. The fresh randomness is generated in each epoch using a bias-resistant random generation protocol adopted by RapidChain [14]. At the end of an epoch, the master node is elected in each cluster based on a calculated reputation score. The reputation score of each validator is calculated within every epoch following a new reputation scheme calculation based on the blockchain proposed in this work. Clusters in MNBC are fully connected via master nodes. Giving the possibility of a better improvement in relation to information propagation as well as security awareness, clusters are also connected by several nodes, known as edge nodes, that represent the closest nodes belonging to different clusters. Master nodes are normal Bitcoin full nodes that can offer a level of additional functions as follows. Supporting a propagation scenario, by which messages are propagated to a list of all known master nodes across the network as well as validators and nodes that belong to the master node's cluster. In addition, information can also be propagated outside a cluster by edge nodes that are connected to other nodes in different clusters. Nodes can join the network by solving a computationally hard puzzle following Sybil-resistant identity leveraging techniques in Ref. [14]. Identity can be represented by the node's locally-generated identity which can be verified by all other honest nodes.

### 5.2. Threat model

This paper considers two attacker strategies as follows:

**Partition attack**: We assume that the attack will be performed within three phases. The first phase starts when several malicious nodes which belong to an attacker join the P2P Bitcoin network and connect to many honest nodes. In order to increase the probability of connecting to as many honest nodes as possible, only the IP addresses of attacker nodes are announced by other attacker nodes. Once the attacker guarantees that the satisfied number of connections to honest nodes is maintained and the connectivity graph is thinned out, a proximate snapshot of the network graph layout will be given by launching the second scenario of the attack. This scenario can be achieved through a probabilistic method which has been introduced in Ref. [26]. By this method, the Bitcoin network topology can be learnt within a reasonable probability through indicating whether or not two peers in the network are connected by sending marker addresses and observing the flow of these addresses. By doing so, the attacker will be able to indicate the minimum vertex cut of the network. Minimum vertex cut is defined as minimum honest peers that removing them causes splitting the graph into at least two partitions [35]. When the attacker selects peers for Minimum vertex cut, DDOS attack will be performed on the selected peers.

**Observe-Act Attack**: We assume the adversary is able to control a fixed part of nodes. This would allow the attacker to perform several malicious actions, such as corrupting honest nodes, sending invalid or inconsistent messages, or remaining silent. Furthermore, the adversary may connect to other helper malicious nodes and try to gain a high

reputation score (same distribution as normal nodes). The main objective of this behaviour is to maximise the probability of having all malicious nodes in one group.

### 5.3. Model Overview

The MNBC protocol has the following main components, as pictured in Fig. 3: Proximity Groups, Master node selection, Reputation scheme, Reputation blockchain, and Group maintenance. The details of how these components work are deferred until Section 6.

**Proximity Groups**: The Bitcoin network is divided into several proximity groups based on network link latency. More information about how proximity groups are established will be provided in Section 6.1.

**Master node selection**: Within each proximity group, there are validators and a master node which is elected based on the reputation score.

**Reputation scheme**: Within an epoch, each validator will have a reputation score which is calculated by all members of the group following the reputation scheme. More information about how to calculate the reputation scheme will be provided in Section 6.3.

**Reputation blockchain**: Within each group and over an epoch, a version of the reputation blockchain is maintained. At the end of the epoch, all groups synchronise their copies of the reputation ledger to have consensus across the entire system on the current reputation blockchain.

**Group maintenance**: This component handles the dynamic nature of the network and how nodes can join and leave the group.

## 6. System design

The following subsections explain each phase of the MNBC protocol in more depth.

### 6.1. Groups establishment

Proximity groups in the MNBC protocol are constructed following the proximity based approach proposed in Ref. [34]. We assume that the network starts as one group. Each node independently runs the proximity protocol by information about discovered nodes and local neighbours. Each node is responsible for gathering proximity knowledge regarding the discovered nodes. When a node discovers new Bitcoin nodes, it calculates the physical Internet distance between itself and the Bitcoin nodes that it has discovered. The calculation of the physical Internet

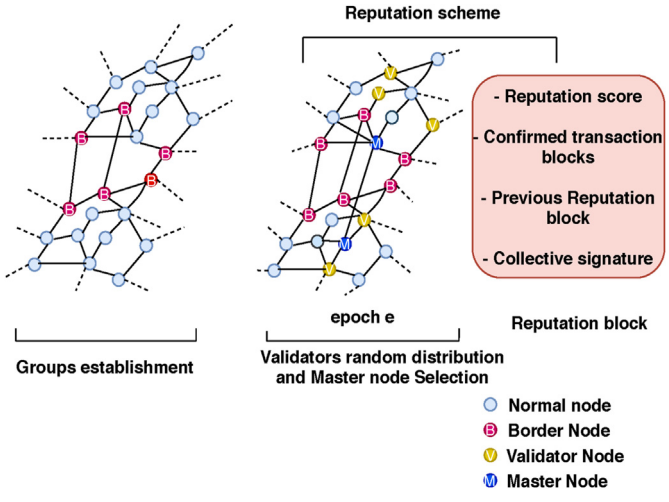


**Fig. 3.** Master Node Based Clustering overview.

distance relies on the round-trip latency between two nodes. As it is mentioned earlier, nodes discover other nodes in the Bitcoin network using either the Bitcoin network discovery mechanism or the Bitcoin DNS service. Two nodes $N_i$ and $N_j$ are considered close on the physical Internet if

$$D_{i,j} < D_{th} \tag{1}$$

where $D_{i,j}$ is the distance between $N_i$ and $N_j$ measured by the round-trip latency, $D_{th}$ is the latency threshold. We introduce a utility function that could calculate the distance between two nodes in the Bitcoin network measured by latency. This function would dramatically change the behaviour of the overlay and help enrich nodes with proximity knowledge. The new utility function is shown in Eq. (2):

$$D_{i,j} = \frac{M_{ping}}{rate(r)} + 2P + q^{'} \tag{2}$$

where $i$ and $j$ are two nodes in the network, $M_{ping}$ is the length of the ping message (Bytes). The term *rate(r)* represents the rate of transmission which is the total amount of data that can be sent from one place to another in a given period of time (around $\approx$ 100 KB/h), while $P$ refers to the propagation speed which is the amount of time it takes for one particular signal to get from one point to another. $P$ is multiplied by Eq. (2) because of the round trip time. The propagation speed is calculated as:

$$P = \frac{D_M}{S} \tag{3}$$

The term $D_M$ denotes the distance between two nodes $i$ and $j$. $D_M$ can be calculated using the geographical distance calculation methodology introduced in Ref. [36]. $S$ is the speed of the signal which is equal to $3\times10^8$ m/s when dealing with Wi-Fi internet, while it is equal to $2/3\times3\times10^8$ m/s in terms of copper cable [37]. $q'$ represents the queuing time (average). Queuing time can be calculated as:

$$q^{'} = \frac{M_{ping}}{rate(r) - \lambda} \times M_{ping} \tag{4}$$

where $\lambda$ represents the arrival rate (how many pings are arriving to the node $j$). This function will allow the calculation of the average waiting time of transactions/blocks happening at each node due to receiving this information relatively at the same time.

As distance measurements are subject to the network congestion and therefore dynamic, within some variance, multiple messages between pairs of nodes, are repeatedly sent over time in order to determine the variance. In terms of a discovered node being close to another node, the node establishes a connection with the discovered node by sending a version message as a handshake. In contrast, these two nodes would have very little chance to get directly connected and stay in the same cluster if they are so far away from each other. Therefore, clusters in the overlay network become more proximity-aware and nodes make a better decision in terms of limiting the cost of communication.

As mentioned before, clusters are fully connected by their edge nodes. Therefore, edge nodes will be selected between every pair of clusters. Edge nodes will be selected to be the closest pair of nodes that belong to two clusters. This ensures efficient information dissemination between clusters as many transmission channels will be available for information to be exchanged among clusters. Furthermore, increasing the number of edge nodes between clusters results in maximising the level of difficulty in partitioning the network (e.g., partitioning attack). More clearly, let $S=\{s_1, s_2, \ldots, s_m\}$ and $R=\{r_1, r_2\ldots\ldots, r_n\}$ represent two clusters, and let $[s_b,r_b]$ denote their border nodes, where $s_b \in S$ and $r_b \in R$, then for all other pairs of clusters (such that $s_i \neq s_b$, $r_j \neq r_b$, $s_i \in S$, $r_j \in R$), $distance(s_i,r_j) \geqslant distance(s_b,r_b)$. Note that $distance(x,y)$ represents the distance between the two nodes $x$ and $y$ measured by link latencies.

### 6.2. Master node selection

When a new epoch $e$ begins, all the validators will be distributed over different groups. Specifically, validators distribution in MNBC relies on random numbers generated by random $seed_e$. A seed can be generated by the secure distributed bias-resistant randomness generation protocol which is adopted by RapidChain [14]. By following this strategy, each validator will be assigned to a different group. Within each group, a validator is elected to act as a master node based on a reputation score which is calculated following the reputation scheme described in Section 6.3. The reputation score helps in electing master nodes that are better suited for that role. To encourage nodes to compete towards winning the master node's role, as it has been proven in Ref. [38], a reward is given to a master node when it propagates a valid transaction and behaves honestly. At the end of an epoch $e$, the validator with the highest score is elected as a master node, as it is illustrated in Algorithm 1. We consider the following three aspects in the process of leader selection.

**Security**: We believe that nodes with higher reputation values are more willing to be responsible for system security.

**Incentive**: The master node will get more rewards in a round of consensus. Thus, we assume that every node wants to be a master node.

**Randomness**: It is hard to predict the results of the clustering and leader selection.

**Algorithm 1**. Master node selection algorithm

**Algorithm 1:** Master node selection algorithm

```
Let M as: validator nodes set in the group
Let z as : highest reputation score to achieve
while M ≠ 0 do
    for validator node in M do
        n ← validator.ReputationScore()
        if n > z then
            z = n
            winning − ValidatorNode ←
             masternode
            Exit()
        end
    end
end
```

### 6.3. Reputation scheme

The reputation scheme is run independently by all validators and members of each group to finalise the reputation score for each validator at the end of every epoch. The reputation score is calculated based on several reputation parameters:

**Number of transactions**: This parameter indicates the number of transactions that the validator has collected and included in a block. The number of transactions is an important scope factor for comparing the behaviour in terms of the degree of satisfaction among different validators.

**The reputation scaling factor**: This parameter indicates the scaling factor of the validator behaviour converted to a reputation score of -1, 0, 1 for correct, unknown, and incorrect decisions. This would help the reputation scheme to distinguish between different behaviours and assess them differently with the aim of making the punishment for dishonest behaviours larger than the reward for honest ones.

**Amount of stake**: This parameter shows how many bitcoins that the validator has burned to get a reputation score based on age. Specifically, the amount of bitcoin that the validator is considered as a stake will be converted to one of the three reputation values of the node by considering the age. The formulation of converting bitcoin stakes into reputation values is as follows:

$$RS(Si,t) = \alpha log(S_i k) \tag{5}$$

where $S_i$ denotes the number of bitcoin owned by the validator $i$ ($S_i$>1), $t$ represents the time when the validator holds the bitcoin, ($S_i k$) denotes the currency age of the currency $S_i$, and $\alpha$ is the conversion factor indicating the proportion of the bitcoin age converted to the reputation value ($0<\alpha<1$). We believe that long-term holders of a large number of bitcoins are more likely to be credible and are less motivated to misbehave, compared with short-term holders. However, the large stake validators who have kept their stake for a longer period will not be able to keep the validation role while they are misbehaving as other reputation factors will be affected. This means the reputation score of the misbehaving validator will be degraded regardless of the large stake.

Given a recent time epoch, following PeerTrust [39], the reputation score $r_i$ of validator $i$ is calculated as follows:

$$r_i = \sum_{n=1}^{l} S(J)*T(J)*RS(Si,t) \tag{6}$$

where $l$ is the number of transactions generated after the previous reputation block. $T(J)$ is the number of transactions collected by the validator $i$. $S(J)$ is the reputation scaling factor of the validator $i$. $RS(Si, t)$ is the reputation value of the validator $i$ based on the amount of stake (bitcoins) the validator $i$ used.

### 6.4. Reputation blockchain

After the reputation score calculation, a Reputation Block (RB) will be generated by validators via a Byzantine fault tolerance consensus. The reputation block includes the reputation score the validator owns in this epoch, the confirmed transaction blocks, the previous reputation block, and the collective signature. Within each group, the reputation block is generated by a validator and signed by other validators within the group. After that, the signed reputation block is sent to other validators across groups to be validated and agreed upon. Our reputation blockchain follows the consensus from RapidChain [14] which can achieve 1/2 resilience within the group. Specifically, the master node sends the reputation block to validators within the group, the validators forward the RB to other nodes within the group with the tag echo if the block is correct. An honest validator will accept and sign the RB if it receives $f$+1 echo of the same and only RB. Validators in other groups will accept the RB if more than half of the validators have signed it.

### 6.5. Group maintenance

Turning now to the second phase of the MNBC protocol which is the group maintenance protocol. We deliver notations to be used in this phase before presenting the peer joining algorithm. In order to increase the security level, peer selection in MNBC reserves the idea of random selections of peers which is important in the Bitcoin network. Specifically, peers in the MNBC protocol select other peers based on a combination of factors of physical proximity (link latency) and random selection. Let $R\{n_0, n_1, \ldots, n_{i-1}\}$ be a set of peers in the Bitcoin network, where $i$ is the number of total peers. Let $M\{mp_0, mp_1, \ldots, mp_{j-1}\}$ be a set of master nodes, where $j$ is the number of master nodes and $M \subseteq R$, Let $mp_l\{mp_l, b_0, b_1, \ldots, b_{k-1}\}$, ($l$ =0, 1, …, $j-1$) and $k$ is the number of peers in the cluster, $mp_l$ be a set of peers in the $l$th cluster. Therefore, we have $mp_l \subseteq R$ and $R = mp_0 \cup mp_1 \cup \ldots \cup mp_{j-1}$. When a node $z$ wants to join the Bitcoin network, it first learns about the available master nodes by contacting an arbitrary node $T$ which already has been learnt from the DNS service. The node $T$ responds with a list of the master nodes it knows about in the network. According to the peer joining Algorithm 2, the node $z$ selects a master node $mp_i$ such that $\forall mp_j \in M$, $distance(z,mp_i) \leq distance(z,mp_j)$. Then, the node $z$ sends a JoiningRequest message to the selected master node. Note that the distance is also calculated based on the link latency, following the same methodology that has been adopted in Section 6.1.

**Algorithm 2.** Peer joining algorithm

**Algorithm 2:** Peer joining algorithm

**Let** $M$ **as:** Master nodes set
**Let** $z$ **as :** new peer to join the network
**while** $M \neq 0$ **do**
  $d \leftarrow distance(z, mp_i)$where $\forall mp_i \in M$
  $d_1 \leftarrow distance(z, mp_j)$ where $\forall mp_j \in M$
  **if** $d < d_1$ **then**
    $z \leftarrow connectTo\ mp_i$
  **else**
    $z \leftarrow connectTo\ mp_j$
  **end**
**end**

## 7. Performance evaluation

As the main goal of the proposed protocol is to perform faster information propagation in the Bitcoin network, the information propagation delay will be considered as the main performance metric in the evaluation of the proposed protocol. However, any improvement on the transaction propagation delay can be generalised to other information dissemination in the Bitcoin network. A security evaluation is carried out in this paper to examine whether the MNBC protocol reduces the level of security in the Bitcoin network. In order to evaluate the proposed protocol, we developed several simulations based on an event-based simulator that has been built in Ref. [32].

### 7.1. Simulation structure

The presented model is a lightweight, event-based simulation which is abstracted from cryptography aspects of Bitcoin. Instead, it focuses on the Bitcoin overlay network and transaction round-trip time delay. The simulation model is developed in Java for object-oriented structure and modularity. Based on the concept of discrete event simulation, the behaviour of the Bitcoin client is modelled as an ordered sequence of well-defined events. These events, which take place at discrete points in simulation time, comprise a specific change in the system's state. In the developed discrete event simulator, two notions of time are taken into account: simulation time and run time. Simulation time reflects the virtual time or logical time in the simulation world, whereas the run time refers to the time of the processor that is consumed by a particular thread. However, simulation time has a direct impact on how the simulation events are organised and accurate results are gained. Specifically, when an event $E_1$ is executed by a thread $A$, $E_1$ should schedule another event $E_{1,Return}$ which represents a successful return from $E_1$. The $E_{1,Return}$ must be scheduled at a specific point in the simulation time which is calculated after adding an appropriate delay. During the time between $E_1$ and $E_{1,Return}$, the simulator can execute any number of events of the same or another client.

The simulator centres around a priority queue that includes all events which are ranked based on their expected time of schedule (ETS) (See Fig. 4). ETS is calculated and referred to each event based on the time distributions which are measured in the real Bitcoin network and attached to the simulator. Based on ETS, the foremost event will be scheduled and removed from the queue. An individual node behaviour such as joining or leaving the network, creating transactions, forwarding transactions, is implemented by inheriting from given generic Java classes.

To make our simulations realistic, we maintained all the required conditions to simulate the reality of the Bitcoin network with respect to the information propagation. Specifically, different measurements of the most influential parameters that have a direct impact on a client's behaviour and information propagation in the real Bitcoin network are attached to the developed simulator. These measurements include the number of reachable nodes, link latencies, nodes' session lengths, and

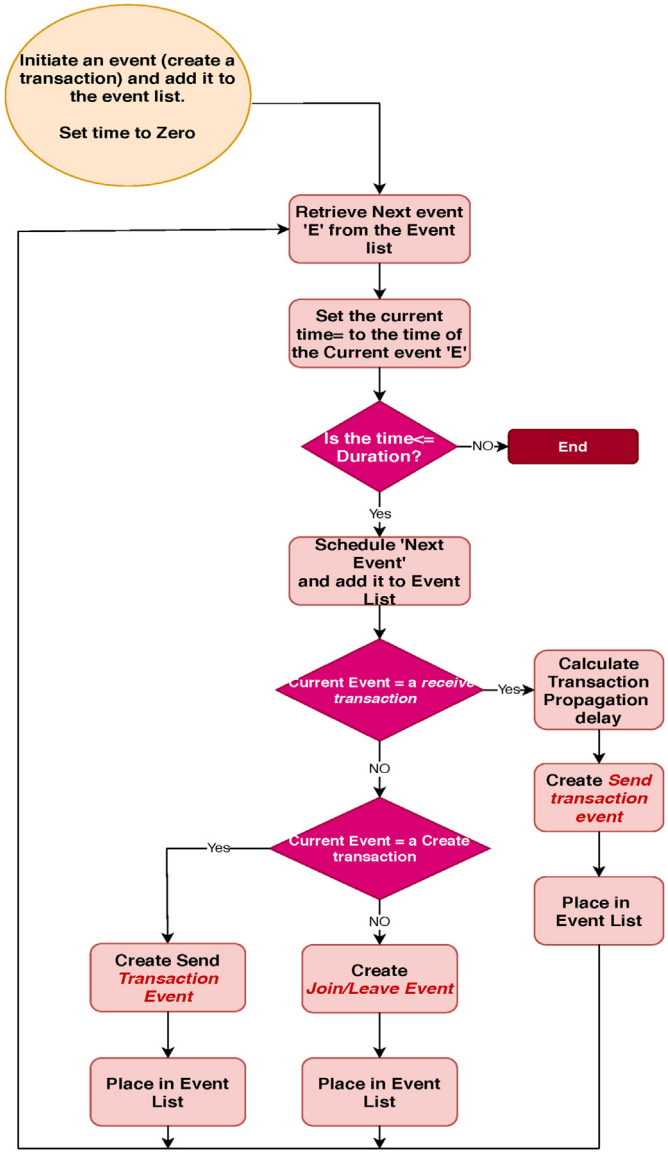


**Fig. 4.** Bitcoin simulator structure.

block confirmation time [32]. The block and transaction processing time is calculated based on the link latency measurements collected from the real Bitcoin network (Section 7.1.2).

#### 7.1.1. Session length

Session length in the real Bitcoin network was calculated by implementing a Bitcoin client which is used to crawl the entire Bitcoin network through establishing connections to all reachable peers in the network. Periodically, the client attempts to discover Bitcoin network peers with the aim of maintaining connections to the majority of them. This is done by sending an Addr message to the client's neighbours. By getting a list of IPs from neighbours, the client starts connecting to each IP in the received list of IPs. As crawlers require time to capture a complete snapshot that accurately reflects the topological properties and dynamics of unstructured P2P networks [40], the developed client crawls the Bitcoin network over a week. During this week, snapshots of IP addresses of reachable peers were published every 3 h to avoid a situation where the captured snapshots became more distorted due to a gap between consecutive snapshots. By using the data which was gathered by running the developed crawler for one week, points in time in which peers left or joined the network were indicated.

The captured snapshots require to be continuous (in time) over a reasonable period of time to avoid a challenge that appears in the data collection for studying churn. More clearly, an incident that might happen during snapshots gathering, such as losing the network connectivity or the observation software crashes, results in a gap in the overall gathering time. During this gap, important data will be missing. To overcome the challenge of data missing, measurements are composed of a series of snapshots maintained by the crawler, each snapshot includes the start time of the crawl. Therefore, it has been possible to identify whether or not some data got missing through examining the series of times in which snapshots started to be captured. By doing this, it has been discovered that significant gaps in the collected data have not been experienced.

The distributions of session length in the real Bitcoin network are shown in Fig. 5. Even though the distributions of session length reveal a considerable churn in the data, 1400 peers did not leave the network during the observation time. Taking these distributions into account, the stability of the network fluctuates. This might lead to change the topology substantially.

### 7.1.2. Link latencies

Measurements of the network latency between peers on the Internet play a significant role in the development of any P2P network model as these measurements control the accuracy of conclusions produced by network models [41]. Thereby, the quality of the developed model relies on the Bitcoin network latency information that requires the acquisition of large-scale measurements to be provided at each node. On top of that, the aim of our research that lies in the area of information propagation in the Bitcoin network, makes the measurements of link latencies between peers a considerable requirement to be performed in the developed model.

In this work, measurements of link latencies between peers were collected by setting up a Bitcoin client that crawls the entire Bitcoin network. Specifically, the developed client utilises a list of IP addresses that can be obtained by following the Bitcoin network discovery mechanism to connect to the majority of peers in the network. Also, the client considers the advantage of ping/pong messages to measure the round trip latency between the discovered peers and the developed client. More precisely, the client attempts to maintain connections to several peers. After that, the client begins an iterative process of sending ping messages to each peer of the connected peers. The link latency between the client and a particular connected peer is calculated when the client hears back from the peer (receiving a pong message). Specifically, the link latency is measured by calculating the time difference between sending a ping message to the peer and receiving a pong message by the client. In order to maintain large-scale and distributed measurements, the client periodically scans the network and applies the same scenario of measuring the link latencies.

The distribution of latencies between the developed client that was located in Portsmouth, UK, and peers in the real Bitcoin network is shown in Fig. 6. These distributions were collected by running the developed crawler which was connected to around 7000 network peers and observing a total of 27000 ping/pong messages. The distribution of latencies reveals that around 75% of the collected latencies are below 800 ms, while 25% of the distributions are over 1000 ms and reach up to 2500 ms. It should be taken into account that these measured distributions indicate the latency between the developed crawler and other peers in the network. However, the obtained link latencies reflect an empirical distribution that is close to the normal distributions.

Although the link latency between two peers relies on the location of the host from which the latency is measured, a similar distribution of latencies over the entire peers might be obtained from two different given hosts, each host in a different location. To prove that the developed crawler was run in a different location. Fig. 7 shows the distribution of the round-trip latencies between peers that are collected by running the developed crawler in Los Angeles, USA. It can be seen that the distributions in Fig. 7 are relatively similar (not identical) to the previous distributions in Fig. 6. However, attaching the obtained link latencies distribution to the developed simulation model would give an accurate estimate of the time delay that is taken by a transaction to reach different peers in the network.

### 7.1.3. The size of the Bitcoin network

As the developed model simulates the information propagation in the Bitcoin network, the size of the network matters due to the fact that the number of nodes has a direct impact on the range of propagation delays. Therefore, attaching an accurate measurement of the number of nodes in the network to the developed model assists in drawing appropriate conclusions from it.

The size of the Bitcoin network was measured in this work by using the same developed crawler in Section 7.1.1. The crawler was able to measure the size of the network by discovering the available IPs in the network and trying to connect to them. Presently, the size of the Bitcoin

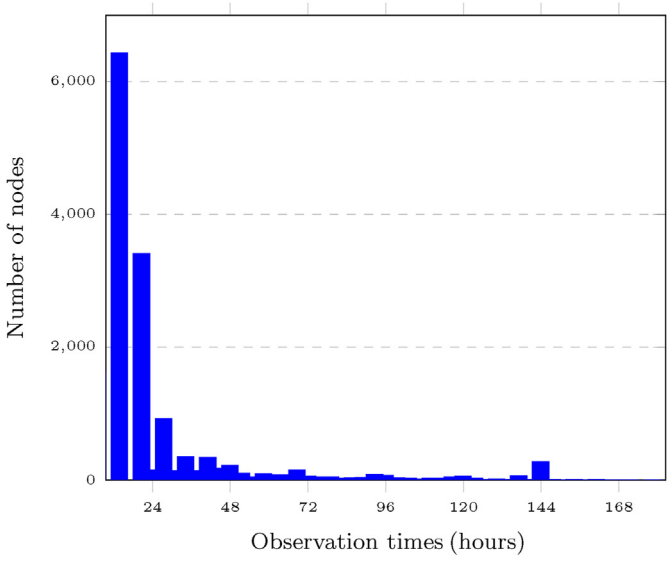


**Fig. 5.** Session lengths of peers in the Bitcoin network.

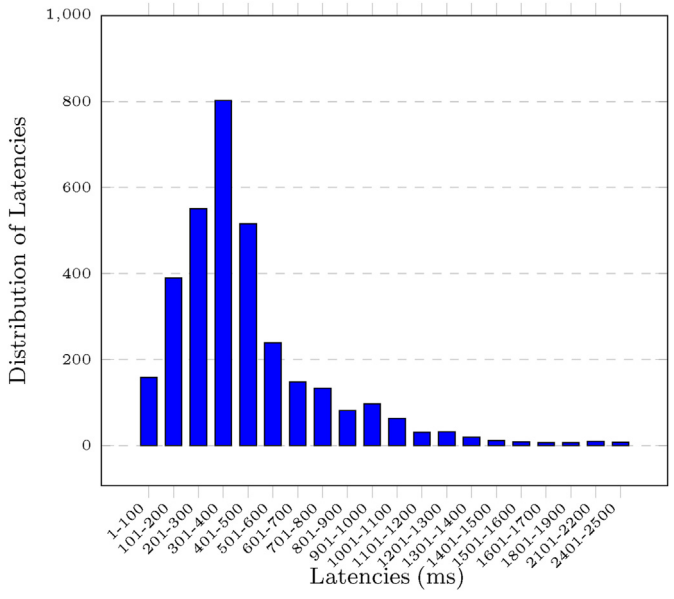


**Fig. 6.** Link latencies between the measurement node (located in Portssmouth, UK) and other peers in the Bitcoin network.

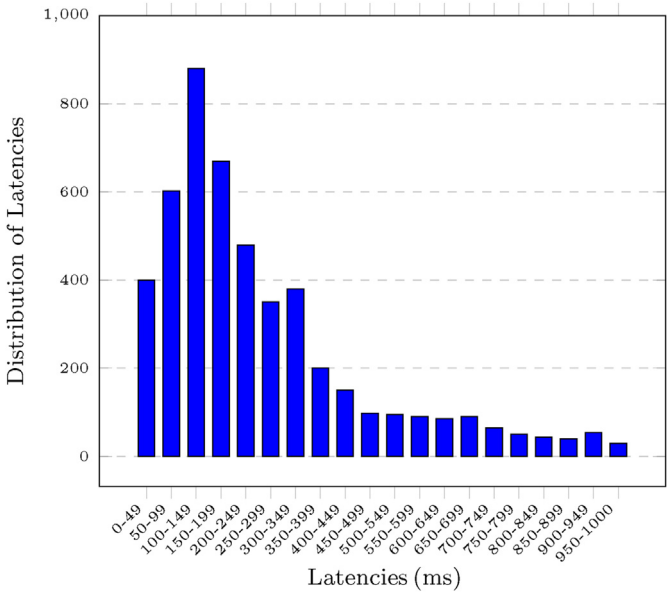


**Fig. 7.** Link latencies between the measurement node (located in Los Angeles, USA) and other peers in the Bitcoin network.

network is around 8000 nodes as the crawler learned 313676 IPs but was only able to connect to 7834 peers.

### 7.1.4. The model validation

In this section, the developed model is validated against the real Bitcoin network based on the transaction propagation delay. As several aspects of the real Bitcoin network, such as client's behaviour, processing delay, and network topology have a direct impact on transaction propagation, the transaction propagation delay measurements are important to test whether the presented model behaves as close as possible to the real network. In the prior research, transaction propagation delay measurements were presented in the real Bitcoin network based on the propagation of INV messages. Specifically, the transaction propagation delay was measured in Refs. [3,41] by setting up a Bitcoin client that keeps listening for INV messages. More clearly, the client calculates the time difference between the first reception of an INV message and subsequent receptions of INV messages, where all the received INV messages belong to the same announcement of a transaction. However, the collected measurements are not indicated when transactions are received, so these measurements do not represent the actual transaction propagation delay. Therefore, measurements of the transaction propagation delay in the real Bitcoin network are performed in this work using a novel methodology by which the transaction propagation delay is accurately measured as these delays are indicated when peers receive transactions.

To measure how fast a transaction is propagated in the Bitcoin network, the Bitcoin protocol was implemented and used to establish connections to many points in the network, in order to measure the time that a transaction takes to reach each point. Clearly, a measuring node is implemented, which behaves exactly like a normal node with the following functionalities. The measuring node connects to 10 reachable peers in the Bitcoin network. Furthermore, it is capable of creating a valid transaction and propagating it to one peer of its connections, and then it tracks the transaction in order to record the time by which each peer of its connections announces the transaction. Specifically, suppose a client $c$ has connections (1,2,3, …., n), $c$ propagates a transaction at time $T$, and it is received by its connected nodes at different times ($T_1$, $T_2$, $T_3$, …, $T_n$) as illustrated in Fig. 8. The time differences between the first transaction propagation and subsequent receptions of the transaction by connected nodes were calculated ($\Delta t_{c,1}$, …, $\Delta t_{c,n}$) according to Eq. (7):

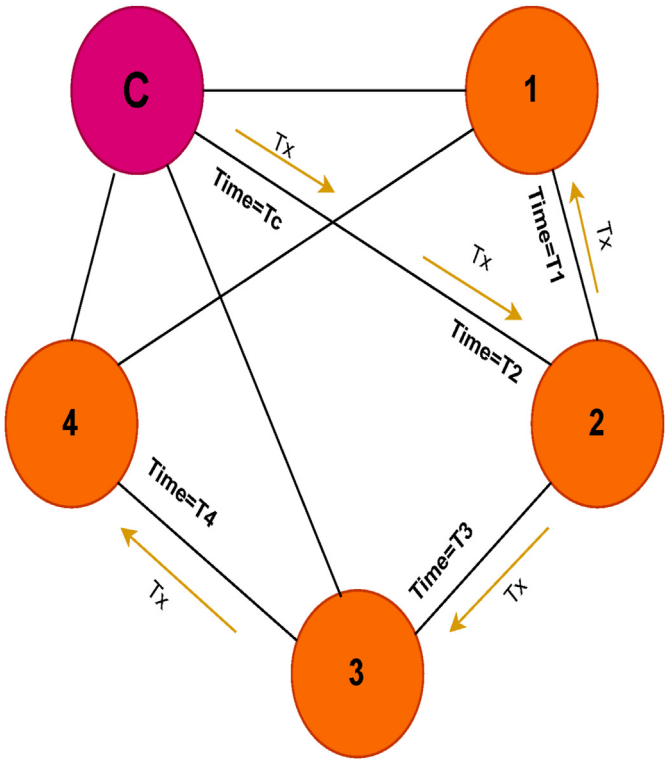


**Fig. 8.** Illustration of propagation experimental setup.

$$\Delta t_{c,n} = T_n - T_c \tag{7}$$

where $T_n > T_{n-1} >$, ……, $T_2 > T_1$. By running the measuring node, the time in which the transaction is propagated by the measuring node and reached each node of the measuring node's connections was calculated. Specifically, the timing information is collected by running the experiment 1000 times as one–off style events, networking delays, etc., are average out. At each run, the measuring node is randomly connected to 10 nodes. The number of connected nodes represents the sequence of the random nodes that the measuring node connects with at each run.

In terms of measuring the transaction propagation delay in the simulation world, the aforementioned measuring method in the real Bitcoin network was used in the simulation. By doing this, the simulation model was validated by comparing the propagation delay measurements that have been collected from the Bitcoin simulator to the same measurements that have been collected from the real Bitcoin network. As the measurements are indicated when peers receive transactions, the distribution of these measured time differences $\Delta t_{c,1}$ represents the real transaction propagation delay.

The average distribution of $\Delta t_{c,n}$ for the real Bitcoin network and the simulated network is shown in Fig. 9. Results reveal that during the first 13 s the transaction has been propagated faster and 6 nodes received it with low a variance of delays. It should be noted that the transaction propagation delays are dramatically increased over nodes (9,10) which means that the transaction has been received by these nodes with significantly larger variances of delays. Obviously, these results reveal that the propagation delay negatively corresponds with the number of nodes, as the total duration of subsequent announcements of the transaction by the remaining nodes increases with larger numbers of connected nodes. This happened due to each node being connected to large segments of the network, while the connected nodes were not geographically localized. On the other hand, transaction verification at each node affects trickling transactions to the remaining nodes. However, we discovered that our simulation model approximately behaves as the real Bitcoin network.

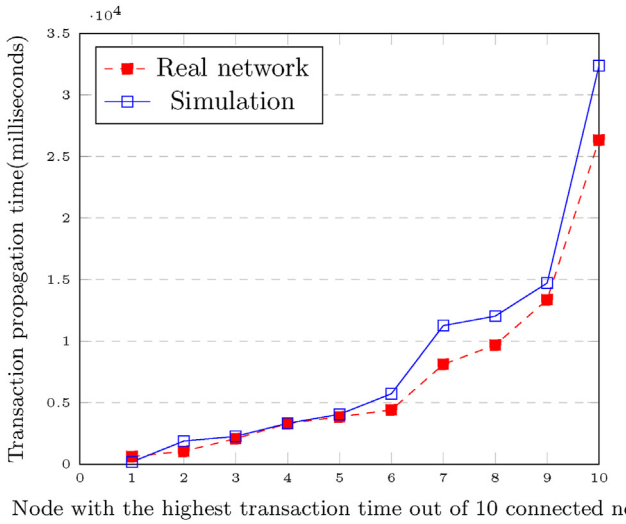


**Fig. 9.** Comparison of the transaction distribution of $\Delta t_{c,n}$ as measured in the real Bitcoin network with simulation results.

To collect the block propagation delay distribution in the real Bitcoin network, a measuring node was implemented to measure the block propagation delay. Specifically, the measuring node $c$ propagates a block at time $T$, and it is received by its connected nodes at different times ($T_1$, $T_2$, …, $T_n$). According to Eq. (7), the block propagation delay in the real Bitcoin network was measured based on the time differences between the first block propagation and subsequent receptions of the block by connected nodes. The average distribution of $\Delta t_{c,n}$ for the block propagation delay in the real Bitcoin network and the simulated network is shown in Fig. 10. These results confirm that the block propagation delay in the simulation model relatively reflects the same measurements in the real Bitcoin network.

### 7.2. Experiments setup

In this section, the experiment setup that is related to the performance evaluation of the MNBC protocol will be explained. In the experiment, the size of the network matters as we based our evaluation on the transaction propagation delay. Therefore, the size of the network in each simulation is 7000 nodes which relatively matches the size of the real Bitcoin network which was measured in Section 7.1.3. Each node in the overlay is allowed to discover new nodes every 100 ms. Several proximity-based clusters will be generated at certain times. During the consensus epoch, nodes within every cluster start running the master node selection algorithm to select a master node for each group. As we based our performance evaluation on measuring how fast a transaction and block are propagated in the network after applying our clustering approaches, we measure the transaction and block propagation delays using the same methodology which was used in our previous work to measure the transaction propagation delays in the real and simulated Bitcoin network. By doing this, evaluation of the proposed protocol against the real Bitcoin network can be done by comparing the measurements of the transaction propagation delay and block propagation delay that have been collected in the simulated Bitcoin protocol to the same measurements that have been collected in the simulated proposed protocol.

Fig. 11 gives a simple diagram of how the simulation experiment works with regard to the MNBC protocol. Before applying the cluster generation algorithms of the proposed protocol, we assume that the network nodes belong to one cluster. Based on the MNBC protocol, proximity groups will be generated and validators will be randomly assigned to these groups. After that, master nodes will be selected at certain times by running the master node selection algorithm.

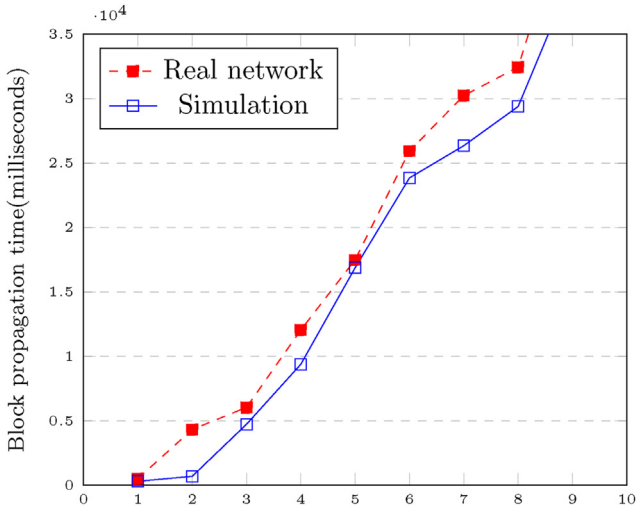


**Fig. 10.** Comparison of the block distribution of $\Delta t_{c,n}$ as measured in the real Bitcoin network with simulation results.

### 7.3. Transaction and block latency

A measuring node $c$ is implemented which is able to create a valid transaction $Tx$ and send it to one node of its connected nodes. It then tracks the transaction in order to record the time by which each node of its connections announces the transaction. Suppose the client $c$ has proximity-based connections (1,2,3, …., $n$), $c$ propagates a transaction at time $T$, and it is received by its connected nodes at different times ($T_1$, $T_2$, $T_3$, …, $T_n$). The time differences between the first transaction propagation and subsequent receptions of the transaction by connected nodes were calculated ($\Delta t_{c,1}$, …, $\Delta t_{c,n}$).

However, the latency is determined by an average of approximately 1000 runs in order to increase the accuracy of the collected latencies which might be affected by several factors such as data corruption and loss of connection.

Similarly, the measuring node $c$ was implemented to create a block with valid transactions and then send it to its connected nodes. Following

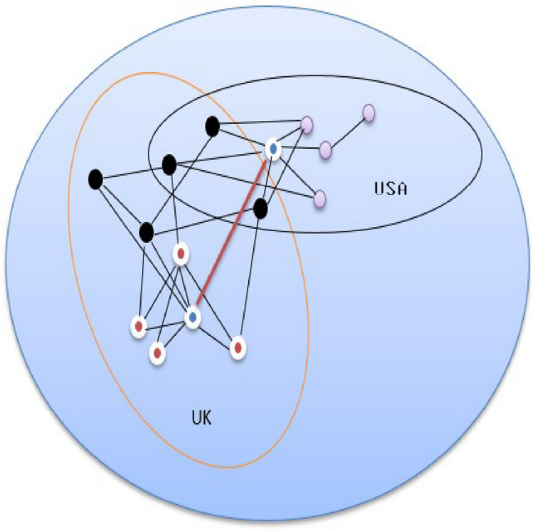


**Fig. 11.** Master Node Based Clustering simulation setup. The black circles represent the border nodes between clusters, while grey and red circles represent nodes in different clusters. Blue circles represents master nodes in each cluster.

the same developed scenario to measure the transaction latency, the node $c$ was able to record the time by which the block was delivered to its connected nodes using the Eq. (7).

### 7.4. Results and discussions

The simulation results show that the proposed protocol offers an improvement in information propagation delay compared to the Bitcoin protocol. Fig. 12 and Fig. 13 compare the distributions of $\Delta t_{c,n}$ for transaction and block propagation delays in the simulated Bitcoin protocol against the same distributions that have been measured in the simulated proposed protocol MNBC. In the figures, the number of connected nodes represents the sequence of the random nodes that the measuring node connects with at each run. Regarding the comparison between the MNBC and Bitcoin protocol, the Bitcoin protocol performs variances of delays, which have been collected in Section 7.1.4, that grow linearly with the number of connected nodes, whereas MNBC maintains lower variances of delays regardless of the number of connected nodes. The reduction of the transaction and block propagation time variances in the proposed protocol has to do with the fact that the Bitcoin network layout in which nodes connect to other nodes without taking advantage of any proximity correlations results in a long communication link cost measured by the distance between nodes. Consequently, the average delay to get transactions and blocks delivered is also increased which, on the other hand, would affect the consistency of the public ledger. On the other hand, maintaining clusters, which are fully connected via master nodes and edge nodes, based on physical internet proximity implies faster information propagation in the MNBC protocol. In fact, contrary to what was previously thought in this area, we found that reconstructing the Bitcoin network layout on proximity bases implies faster transmissions.

Turning now to the comparison between the MNBC protocol and the BCBSN protocol. As shown in Fig. 12, both proposed protocols show relatively the same variances of delays over nodes 1, 2, 3, 4, 5, and 6. From node 7, variances of delays in the BCBSN protocol started climbing steadily and reached a peak over node 10 recording transaction propagation delays of nearly 18000 ms. On the other hand, the variances of delays were totally improved in the MNBC protocol over Bitcoin and the BCBSN protocol, especially over nodes 8, 9, and 10. The most likely cause of the higher variances of delays in the BCBSN protocol is the fact that the information flow between clusters in the BCBSN protocol can only be

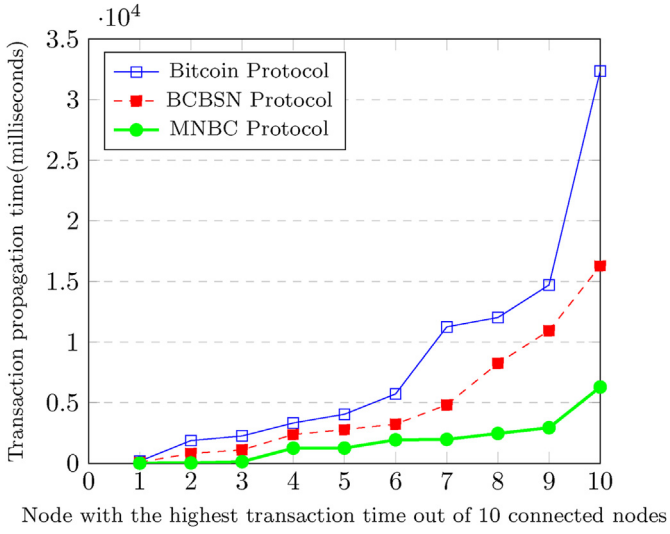


**Fig. 12.** Comparison of the transaction propagation distribution of $\Delta t_{c,n}$ measured in the simulated Bitcoin protocol with Master Node Based Clustering (MNBC) protocol and Bitcoin Clustering Based Super Node (BCBSN) protocol simulation results.($d_t$ in MNBC = 25 ms).

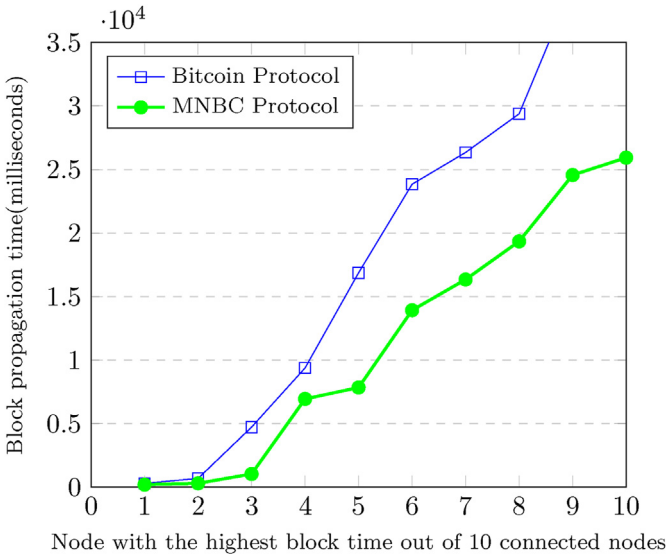


**Fig. 13.** Comparison of the block propagation distribution of $\Delta t_{c,n}$ measured in the simulated Bitcoin protocol with Master Node Based Clustering (MNBC) protocol simulation results.($d_t$ in MNBC = 25 ms).

maintained through supers peers. This causes a lack of transformation channels between clusters which results in inefficient information distribution over the network. The lack of connections between clusters in the BCBSN protocol has been tackled in the MNBC protocol by considering the edge nodes technology which adds an extra connection channel between clusters. Therefore, faster information propagation has been achieved in MNBC compared to BCBSN.

As MNBC is based on the suggested threshold, it is worth investigating the optimal latency that can speed up information propagation. For this purpose, we experiment with MNBC based on several suggested latency $d_t$. In MNBC, the comparison among three variances of delays was done based on three different latency suggested thresholds 30 ms, 60 ms, and 90 ms. Results that are shown in Fig. 14 reveal that the less latency distance threshold in MNBC performs less variance of delays. Judging from that, there is a negative correlation between propagation delay and the latency threshold, as the total duration of subsequent announcements of the transaction by the remaining nodes increases with a larger latency threshold. The key reason variances of delays have been declined when the threshold value is reduced is that the number of nodes in each cluster

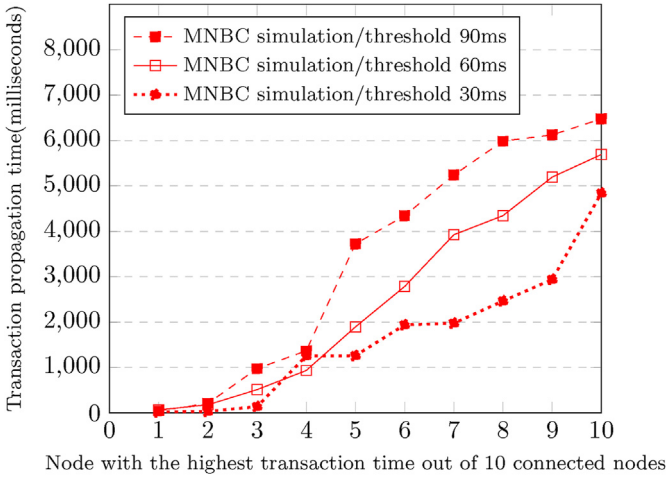


**Fig. 14.** Comparison of the transaction propagation distribution of $\Delta t_{c,n}$ as measured in the simulated Master Node Based Clustering (MNBC) protocol with three thresholds ($d_t$ = 30 ms, 60 ms, 90 ms ).

is minimised due to the limited coverage of the physical topology which is offered by $d_t$. However, reducing the latency threshold would decrease the size of clusters which leads to the creation of many clusters in the network. This might result in a side effect which is represented by an extra traffic overhead. This will be investigated in our future work.

## 8. Security evaluation

### 8.1. Partition attack

The potential of partition attacks on the proposed protocols as well as the Bitcoin network will be evaluated in this section using the designed simulator. Generally, the main goal of partition attacks is to partition the network into two or more partitions as well as prevent information flow between partitions [42]. In terms of the Bitcoin network, the main target for an attacker launching partition attacks is to disturb the normal Bitcoin's main functions which would affect the users' trust in the system. This might be an incentive for an attacker due to the influence of users' trust in the system on the Bitcoin exchange rate.

#### 8.1.1. Experiment setup

The experiment setup of the partition attack evaluation is explained in this section. In this experiment, two platforms were used, the developed simulator (See Section 7.1) and the metis graph partition toolkits [43]. The first phase of the attack will be started when the network topology is restructured according to each MNBC protocol. Specifically, several attacker nodes join the network and start establishing connections with many honest nodes. As the partition attack evaluation in this work is based on minimum vertex cut as a cost metric, the minimum vertex cut of the network topology was determined at regular intervals using metis graph partition toolkits. Metis algorithm can achieve a balanced partitioning that minimises either the communication volume or the number of edge cuts [43]. However, the attack's aim in this work is to get partitions of non-negligible size without taking into consideration whether or not the partitions are imbalanced. The minimum vertex cut is determined by an average of approximately 1000 runs in the simulated Bitcoin protocol and the proposed protocols.

We developed four experiment scenarios with different network sizes (2000, 4000, 6000, and 8000). The size of the attacking botnet was chosen to match the number of honest peers in each scenario. The first phase of the attack starts when the network topology is restructured according to each protocol of the proposed protocols. Specifically, several attacker nodes join the network and start establishing connections with many honest nodes. As we based our partition attack evaluation on minimum vertex cut as a cost metric, the minimum vertex cut of the network topology is determined at regular intervals using metis graph partition toolkits [43]. Metis algorithm can achieve a balanced partitioning that minimises either the communication volume or the number of edge cuts.

### 8.2. Results and discussions

Fig. 15 shows the results of three simulated attacks on a model of the real Bitcoin network, MNBC, and BCBSN protocols. Each attack was launched based on different network sizes (2000, 4000, 6000, and 8000). The configuration of the desired imbalance factor was done in a way that the largest partitions did not include more than 60% of all nodes. In the small scenarios with a number of nodes (2000 and 4000), the number of honest peers in the minimum vertex cut in all protocols after launching the partition attack stayed below 500 which reveals that all protocols are relatively similar in terms of resistance against partition attacks. While in the large scenarios with 6000 and 8000 peers, the level of resistance against partition attacks increased in all protocols as the number of nodes increased. The highest level is experienced in the Bitcoin protocol, while the lowest level appears in the BCBSN protocol. Precisely, the minimum vertex cut in the Bitcoin protocol increased from around 500 to 3800

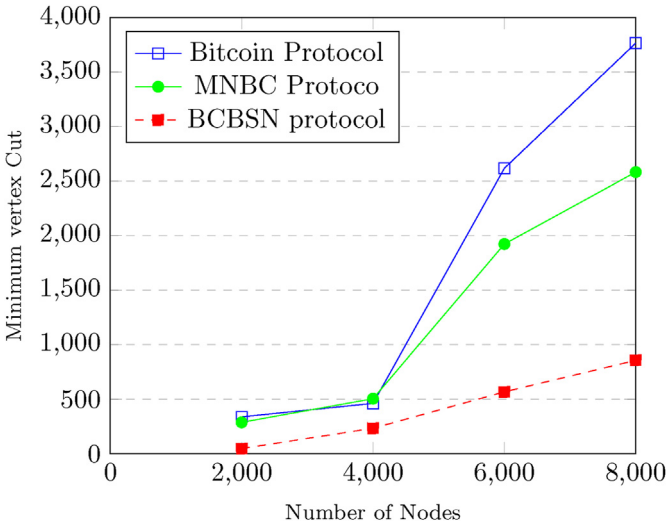


**Fig. 15.** Number of honest peers on the minimum vertex cut. MNBC: Master Node Based Clustering; BCBSN: Bitcoin Clustering Based Super Node.

with the scenario of 8000 peers resulting in a notable gap in the minimum vertex cut between the Bitcoin protocol and other protocols. Whereas, the MNBC protocol shows higher resistance than the BCBSN protocol, where the number of honest nodes in the minimum vertex cut goes above 2500 in the scenario of 8000 nodes. The BCBSN protocol is considered the worst of the proposed protocols in terms of how easy to perform partition attacks as it experienced the lowest minimum vertex cut in both large and small scenarios. Although the proposed protocols experienced less minimum vertex cut compared to the Bitcoin protocol, the number of honest nodes required to cut in the proposed protocols is still high which needs significant resources to be considered. As expected, clusters in MNBC that are fully connected via master nodes and edge nodes reflect less number of honest nodes in minimum vertex cut. While clusters in BCBSN that are connected via super peers result in a number of nodes in the area of minimum vertex cut goes down. However, results from large scenarios in all protocols illustrate that for a higher number of peers, more effort needs to be spent by an attacker to split the network.

On the question of whether or not an attacker's session length affects the resistance of the network to partition attacks, the resistance of the MNBC protocol will be tested against several session lengths of attack. Fig. 16 shows the results of the simulated partition attacks on a model of the real Bitcoin network, MNBC, and BCBSN protocol, including different session lengths.

The results shown in Fig. 16 illustrate the impact of the attacker's session length ($S_A$) on the success of the attack. Within 24 h of attack, the number of nodes in the minimum vertex cut declined in the simulated MNBC protocol as well as the Bitcoin and BCBSN protocols as follows: the minimum vertex cut declined from around 3700 to 1500 in the real Bitcoin network. The same scenario happened in the MNBC protocol where the minimum vertex cut decreased from around 2500 to 1150, and from 850 to 290 in the BCBSN protocol. However, it can be concluded from the obtained results that the more patience from the attackers with a higher number of peers, the better chances of success in splitting the network.

Moving on now to evaluate the impact of the number of clusters on the difficulty of performing partitioning attacks in the proposed approaches. Fig. 17 shows the minimum vertex cut in the proposed approaches with respect to different numbers of clusters.

According to the results that are shown in Fig. 17, increasing the number of clusters results in more nodes in the minimum vertex cut. It can be noticed that the minimum vertex cut in all protocols positively corresponds with the number of clusters, more proximity clusters result in increasing the difficulty to perform partition attacks. This suggests a

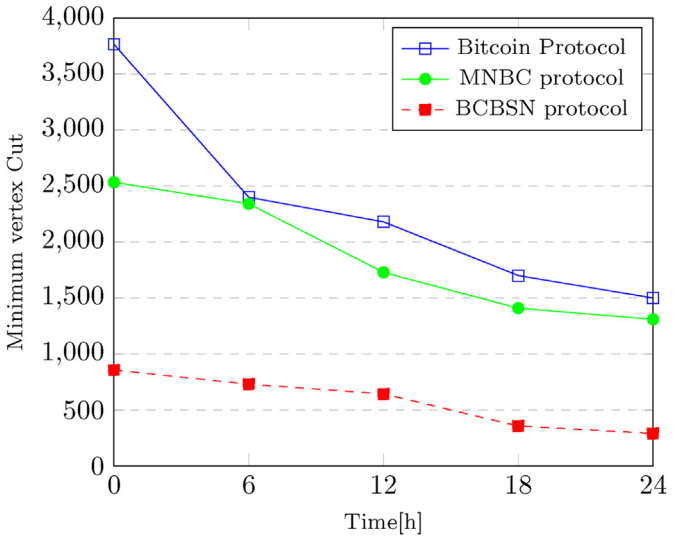


**Fig. 16.** Number of non-attacker peers on the minimum vertex cut during an attack with 7000 honest peers parametrized as in the real-world network, attacker's session length $S_A$=6 h. MNBC: Master Node Based Clustering; BCBSN: Bitcoin Clustering Based Super Node.

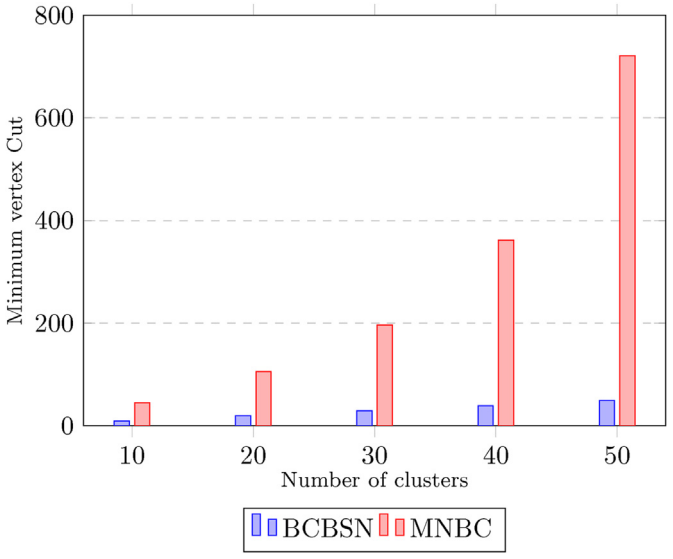


**Fig. 17.** Number of honest peers on the minimum vertex cut based on number of partitions in Master Node Based Clustering (MNBC) and Bitcoin Clustering Based Super Node (BCBSN).

strong link may exist between the number of clusters and the improvement of the minimum vertex cut. This improvement can be translated based on the number of master nodes and edge nodes in the MNBC approach, and super peers in the BCBSN protocol. Specifically, increasing the number of clusters in MNBC offers more master nodes and edge nodes that are located in the minimum vertex cut. In respect to the BCBSN protocol, more clusters reflect more super peers that are classified as nodes in the minimum vertex cut.

### 8.3. Observe-Act Attack

We have implemented the MNBC protocol using Go language [44] to evaluate the impact of the Observe-Act Attack on the throughput performance. We consider that the adversary is able to corrupt a fixed

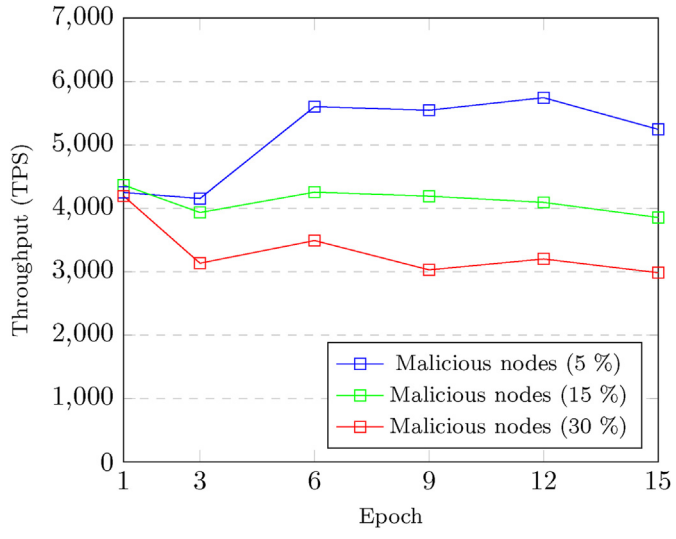


**Fig. 18.** The effect of malicious nodes on overall Master Node Based Clustering performance.

number of honest nodes $t$, where $t<n/3$. The corrupted nodes may collude with each other to disturb the protocol in any arbitrary manner, such as disturbing the transaction propagation or sending invalid messages. We tested the Observe-Act attack model over several epochs with 4000 nodes and a group size of 225. Within the experiment, nodes were categorized into two types—legitimate nodes and malicious nodes. The legitimate nodes took part in the transaction/block forwarding process, master peers selection, and consensus protocol. Whereas the malicious nodes do not participate in the gossip protocol (i.e., remain silent).

Fig. 18 depicts the effect of malicious nodes on the MNBC protocol over several epochs. Our protocol runs poorly within the first two epochs, but the throughput increased after epoch 3 where malicious nodes cannot significantly degrade the performance of MNBC due to lower reputation. However, the performance of MNBC decreased as the percentage of malicious nodes increased from 5% to 30%.

## 9. Conclusion

In this paper, a brief background of the Bitcoin system as well as analysing the information propagation in the real Bitcoin network were presented. In addition, how propagation delay in the Bitcoin network could affect the security by offering an opportunity to double spend the same coins; thereby abusing the consistency of the public ledger was discussed in this paper. The MNBC, a novel clustering protocol that incorporates master node based physical proximity and reputation scheme based blockchain into the existing Bitcoin protocol, was presented in this paper. By conducting extensive simulations, MNBC evaluation results indicate an improvement in the transaction propagation delay over the Bitcoin network protocol. However, MNBC maintains a lower variance of delays than the BCBSN protocol. Furthermore, experiments with different latency thresholds have been conducted to identify the distance threshold that would give a better improvement in the transaction propagation delay. We discovered that providing a less latency distance threshold would improve the transaction propagation delay with a high proportion. Furthermore, evaluation of partitioning attacks in the Bitcoin network as well as the MNBC and BCBSN protocols were presented in this paper. The results revealed that attackers still need more resources to split the network in the proposed protocol, especially with a higher number of nodes. Furthermore, the effect of malicious nodes on the MNBC protocol was validated in this paper. The results proved that MNBC throughput had not been significantly affected by the malicious behaviour.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, Available online: http://www.bitcoin.org/bitcoin.pdf, 2008. (Accessed 18 December 2020).

[2] D. Ron, A. Shamir, Quantitative analysis of the full bitcoin transaction graph, in: Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, Germany, 2013, pp. 6–24.

[3] C. Decker, R. Wattenhofer, Information propagation in the bitcoin network, in: 2013 IEEE P2P 2013 Proceedings; 9–11 Sep 2013; Trento, Italy, IEEE, Piscataway, NJ, USA, 2013, pp. 1–10.

[4] M. Conti, E.S. Kumar, C. Lal, et al., A survey on security and privacy issues of bitcoin, IEEE Commun. Surv. Tutorials 20 (4) (2018) 3416–3452.

[5] C. Stathakopoulou, C. Decker, A. Wattenhofer, A Faster Bitcoin Network, Semester Thesis, 2015. ETH Zürich, Zürich, Sweden.

[6] Y. Sompolinsky, A. Zohar, Accelerating bitcoin's transaction processing: fast money grows on trees, not chains, Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.433.6590&rep=rep1&type=pdf, 2008. (Accessed 18 December 2020).

[7] G.O. Karame, E. Androulaki, M. Roeschlin, et al., Misbehavior in bitcoin: a study of double-spending and accountability, ACM Trans. Inf. Syst. Secur. 18 (1) (2015) 1–32, https://doi.org/10.1145/2732196.

[8] M. Rosenfeld, Analysis of Hashrate-Based Double Spending, arXiv, preprint (2009) arXiv: 1402.2009.

[9] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: analysis and applications, in: Advances in Cryptology—EUROCRYPT 2015, Springer, Berlin, Heidelberg, Germany, 2015, pp. 281–310.

[10] A. Miller, J.J. LaViola, Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin, Available online: https://nakamotoinstitute.org/static/docs/anonymous-byzantine-consensus.pdf, 2014. (Accessed 18 December 2020).

[11] M. Sallal, G. Owenson, M. Adda, Security and performance evaluation of master node protocol in the bitcoin peer-to-peer network, in: 2020 IEEE Symposium on Computers and Communications (ISCC); 7–10 July 2020; Rennes, France, IEEE, Piscataway, NJ, USA: IEEE, 2020, pp. 1–6.

[12] J.E. Pazmiño, C.K. da Silva Rodrigues, Simply dividing a bitcoin network node may reduce transaction verification time, in: The SIJ Transactions on Computer Networks & Communication Engineering (CNCE), 4, 2015, pp. 17–21 (2).

[13] C. Basescu, E. Kokoris-Kogias, B.A. Ford, Poster: Low-latency blockchain consensus, 2017. Available online: http://www.ieee-security.org/TC/SP2017/poster-abstracts/IEEE-SP17_Posters_paper_31.pdf. (Accessed 18 December 2020).

[14] M. Zamani, M. Movahedi, M. Raykova, Rapidchain: scaling blockchain via full sharding, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security; 15–19 Oct 2018; Toronto, ON, Canada, ACM, NewYork, NY, USA, 2018, pp. 931–948.

[15] M. Corallo, Compact block relay, BIP: 152 (2017). Available online: https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki. (Accessed 18 December 2020).

[16] F. falcon, Fast bitcoin backbone falcon, URL, https://www.falcon-net.org, 2015.

[17] A. Biryukov, I. Pustogarov, Bitcoin over Tor isn't a good idea, in: 36th IEEE Symposium on Security and Privacy; 18–20 May 2015; San Jose, CA, USA, IEEE, Piscataway, NJ, USA, 2015, pp. 122–134.

[18] M. Corallo, Fibre: Fast Internet Bitcoin Relay Engine, 2017. Available online: https://github.com/bitcoinfibre/bitcoinfibre. (Accessed 18 December 2020).

[19] T. Bamert, C. Decker, L. Elsen, et al., Have a snack, pay with bitcoins, in: 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P); 9–11 Sep 2013; Trento, Italy, IEEE, Piscataway, NJ, USA, 2013, pp. 1–5.

[20] J. Chen, SX. Yao, Q. Yuan, et al., Certchain: public and efficient certificate audit based on blockchain for tls connections, in: IEEE INFOCOM 2018—IEEE Conference on Computer Communications; 16–19 Apr 2018; Honolulu, HI, USA, IEEE, Piscataway, NJ, USA, 2018, pp. 2060–2068.

[21] L. Bahri, S. Girdzijauskas, Trust mends blockchains: living up to expectations, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS); 7–10 Jul 2019; Dallas, TX, USA, IEEE, Piscataway, NJ, USA, 2019, pp. 1358–1368.

[22] JS. Yu, D. Kozhaya, J. Decouchant, et al., RepuCoin: your reputation is your power, IEEE Trans. Comput. 68 (8) (2019) 1225–1237, https://doi.org/10.1109/TC.2019.2900648.

[23] CY. Huang, ZY. Wang, HX. Chen, et al., Repchain: a reputation based secure, fast and high incentive blockchain system via sharding, IEEE Internet Things J. 8 (6) (2020) 4291–4304.

[24] M. Fadhil, G. Owenson, M. Adda, A bitcoin model for evaluation of clustering to improve propagation delay in bitcoin network, in: 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES); 26–26 Aug 2016; Paris, France, IEEE, Piscataway, NJ, USA, 2016, pp. 468–475.

[25] J.A. Donet, C. Pérez-Solà, J. Herrera-Joancomartí, The bitcoin p2p network, in: R. Böhme, M. Brenner, T. Moore (Eds.), FC 2014: Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, Germany, 2014, pp. 87–102.

[26] A. Biryukov, D. Khovratovich, I. Pustogarov, Deanonymisation of clients in bitcoin p2p network, in: CCS '14: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security; 3–7 Nov 2014; Scottsdale, AZ, USA, ACM, New York, NY, USA, 2014, pp. 15–29.

[27] E. Heilman, A. Kendler, A. Zohar, et al., Eclipse attacks on bitcoin's peer-to-peer network, in: 24th USENIX Security Symposium (USENIX Security 15); 12–14 Aug 2015; Washington, DC, USA, USENIX Association, Berkeley, CA, USA, 2015, pp. 129–144.

[28] D. Kondor, M. Pósfai, I. Csabai, et al., Do the rich get richer? an empirical analysis of the bitcoin transaction network, PLoS One 9 (2) (2014), e86197, https://doi.org/10.1371/journal.pone.0086197.

[29] S. Feld, M. Schönfeld, M. Werner, Analyzing the deployment of bitcoin's p2p network under an as-level perspective, Procedia Comput. Sci. 32 (2014) 1121–1126, https://doi.org/10.1016/j.procs.2014.05.542.

[30] A.M. Antonopoulos, Mastering Bitcoin: Unlocking Digital Cryptocurrencies, O'Reilly Media, Inc., Sebastopol, CA, USA, 2015.

[31] G. Karame, E. Androulaki, S. Capkun, Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin, IACR Cryptology ePrint Archive 248 (2012).

[32] M. Fadhil, G. Owen, M. Adda, Bitcoin network measurements for simulation validation and parameterisation, in: S. Alekseev, P. Dowland, B. Ghita (Eds.), Proceedings of the Eleventh International Network Conference (INC 2016), University of Plymouth, Plymouth, UK, 2016, pp. 109–114.

[33] E. Duffield, H. Schinzel, F. Gutierrez, Transaction Locking and Masternode Consensus: A Mechanism for Mitigating Double Spending Attacks, 2014. Available online: https://cryptopapers.info/assets/pdf/instasend.pdf. (Accessed 18 December 2020).

[34] M. sallal, G. Owenson, M. Adda, Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network, in: 2017 IEEE 37th International Conference on Distributed Computing Systems; 5–8 Jun 2017; Atlanta, GA, USA, IEEE, Piscataway, NJ, USA, 2017, pp. 2411–2416.

[35] O. Ugurlu, M. Berberler, G. Kızılates, et al., New algorithm for finding minimum vertex cut set, in: 2012 IV International Conference "Problems of Cybernetics and Informatics (PCI)"; 12–14 Sep 2012; Baku, Azerbaijan, IEEE, Piscataway, NJ, USA, 2012, pp. 1–4.

[36] M. Fadhil, G. Owenson, M. 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); 8–12 May 2017; Lisbon, Portugal, IEEE, Piscataway, NJ, USA, 2017, pp. 556–559.

[37] E. Bogatin, Signal and Power Integrity—Simplified, 2nd, Pearson Education, Inc, Upper Saddle River, NJ, USA, 2010.

[38] M. Babaioff, S. Dobzinski, S. Oren, et al., On bitcoin and red balloons, in: EC'12: Proceedings of the 13th ACM conference on electronic commerce; 4–8 Jun 2012; Valencia, Spain, ACM, New York, NY, USA, 2012, pp. 56–73.

[39] L. Xiong, L. Liu, Peertrust: supporting reputation-based trust for peer-to-peer electronic communities, IEEE Trans. Knowl. Data Eng. 16 (7) (2004) 843–857, https://doi.org/10.1109/TKDE.2004.1318566.

[40] D. Stutzbach, R. Rejaie, S. Sen, Characterizing unstructured overlay topologies in modern p2p file-sharing systems, IEEE/ACM Trans. Netw. 16 (2) (2008) 267–280, https://doi.org/10.1109/TNET.2007.900406.

[41] T. Neudecker, P. Andelfinger, H. Hartenstein, A simulation model for analysis of attacks on the bitcoin peer-to-peer network, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM); 11–15 May 2015; Ottawa, ON, Canada, IEEE, Piscataway, NJ, USA, 2015, pp. 1327–1332.

[42] M. Apostolaki, A. Zohar, L. Vanbever, Hijacking bitcoin: routing attacks on cryptocurrencies, in: 2017 IEEE Symposium on Security and Privacy (SP); 22–26 May 2017; San Jose, CA, USA, IEEE, Piscataway, NJ, USA, 2017, pp. 375–392.

[43] P. Miettinen, M. Honkala, J. Roos, Using METIS and hMETIS Algorithms in Circuit Partitioning, Helsinki University of Technology, Espoo, Finland, 2006.

[44] A.A.A. Donovan, B.W. Kernighan, The Go Programming Language, Addison-Wesley Professional, Old Tappan, NJ, USA, 2015.