# An Efficient Peer-to-Peer Bitcoin Protocol with Probabilistic Flooding

Huy Vu(✉) and Hitesh Tewari

Trinity College Dublin, Dublin 2, Ireland
{vuhu,htewari}@tcd.ie

**Abstract.** Bitcoin was launched in 2009, becoming the world's first ever decentralized digital currency. It uses a publicly distributed ledger called the blockchain to record the transaction history of the network. The Bitcoin network is structured as a decentralized peer-to-peer network, where there are no central or supernodes, and all peers are seen as equal. Nodes in the network do not have a complete view of the entire network and are only aware of the nodes that they are directly connected to. In order to propagate information across the network, Bitcoin implements a gossip-based flooding protocol. However, the current flooding protocol is inefficient and wasteful, producing a number of redundant and duplicated messages. In this paper, we present an alternative approach to the current flooding protocol implemented by Bitcoin. We propose a novel protocol that changes the current flooding protocol to a probabilistic flooding approach. Our approach allows nodes to maintain certain probabilities of sending information to their neighbours, based on previous message exchanges between the nodes. Our experimental evaluation shows a reduction in the number of duplicated messages received by each node in the network and the total number of messages exchanged in the network, whilst ensuring that the reliability and resilience of the system were not negatively affected.

**Keywords:** Bitcoin · Peer-to-Peer · Flooding · Cryptocurrencies · Information propagation

## 1 Introduction

Cryptocurrencies, of which Bitcoin is the most popular, have risen greatly in popularity in recent times. With the popularization of cryptocurrencies comes an increase in daily users. As a result of this increase in daily users, countless more transactions are made within the network leading to an increase in network resources and power consumption by the systems in order to maintain the cryptocurrencies. For example, Bitcoin between 2011 and 2012 averaged approximately 7,000 transactions per day, but at the time of

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie).


M. H. Miraz et al. (Eds.): iCETiC 2019, LNICST 285, pp. 29–45, 2019.
https://doi.org/10.1007/978-3-030-23943-5_3

writing, Bitcoin currently averages approximately 270,000 transactions per day, with the daily trading value estimated at approximately $700 million[1].

Bitcoin uses a publicly shared distributed ledger known as the *blockchain* to maintain the transaction history in the network. Transactions in the network are grouped together and placed in *blocks*. Bitcoin uses proof-of-work (PoW) as its consensus mechanism [1]. In PoW, participants in the network are challenged to solve a computationally difficult problem, from which blocks are produced when solved. By successfully completing the PoW puzzle, the newly created block is then added to end of the already existing chain of blocks. The linking of the blocks to create the chain of blocks ensures a serial and chronological ordering of transactions, allowing all the nodes in the network to agree on a common ordering of transactions. The blockchain is maintained in a decentralized manner by all the nodes participating in the network.

As the Bitcoin network is structured as a decentralized peer-to-peer network (P2P), information is disseminated across the network through *gossip-based flooding* [2]. Nodes participating in the network do not have an entire view of the network. Instead of having an entire view of the network topology, nodes are only aware of the other nodes that they are directly connected to, known as their *neighbours*. Due to the decentralized nature of the network, if a node wants to broadcast new information across the network, they must follow Bitcoin's implementation of the flooding protocol. By following the flooding protocol implemented by Bitcoin, nodes will broadcast their desired information to each of their connected neighbours. Once received, the node's neighbours will then in turn broadcast the newly received information to their neighbours, who will then broadcast it to their neighbours until eventually the information is received by all the peers in the network [3]. However, this flooding protocol for information dissemination is wasteful, producing a large number of redundant and duplicated messages.

### 1.1 Our Contributions

In this paper, we present a novel protocol that aims to change the current flooding protocol implemented by Bitcoin. The proposed protocol changes the flooding protocol implemented by Bitcoin to a probabilistic flooding approach. The proposed probabilistic flooding approach is based on the idea that nodes in the Bitcoin network have a wide variance in the number of neighbours they are connected to. Therefore, if a node's neighbour is well-connected in the network, they are likely to already have the transaction that the node was going to transmit to it, making the message redundant. However, if a node's neighbour is less connected in the network, they may likely not have the transaction and may need the transmitting node to broadcast the transaction to them. As a result of the wide variance in the number of neighbours a node may have, we propose a probabilistic flooding approach where a node maintains a "probability of sending" for each of their neighbours based on previous message exchanges between the node and the neighbour. The main objective of the change in protocol is to reduce

[1] https://www.blockchain.com/charts/estimated-transaction-volume-usd?daysAverageString=7timespan=all.

the number of redundant and duplicated messages being generated in the network whilst ensuring that the reliability and resilience of the system is not negatively affected by the change in protocol.

### 1.2 Paper Structure

The remainder of the paper is structured as follows: Sect. 2 presents an overview of Bitcoin, Sect. 3 discusses the Bitcoin network and Information Propagation, Sect. 4 discusses related work, Sect. 5 presents our probabilistic flooding protocol, Sect. 6 evaluates our protocol and Sect. 7 concludes the paper.

## 2 Bitcoin Overview

Bitcoin was proposed in 2008 and launched in 2009, under the pseudonym Satoshi Nakamoto in their paper entitled "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. The objective of Bitcoin is to create a means of exchange, without dependence on a central authority, that could be transferred electronically in a secure, verifiable and immutable way. The most important attribute of Bitcoin is the decentralization nature of it - the lack of dependence on a central server or trusted parties. As mentioned in a forum post shortly after Bitcoin was launched, Satoshi wrote that "The root problem with conventional currency is all the trust that is required to make it work"[2].

In the following subsections, we will describe the main building blocks of the Bitcoin system.

### 2.1 Transactions

Transactions are the most important part of the Bitcoin system. Everything else in Bitcoin is designed so that transactions are able to be created, propagated, validated and added to the global distributed ledger used in Bitcoin (also known as the blockchain).

At an abstract level, a transaction essentially transfers bitcoins from one or more source accounts to one or more destination accounts. Each account is created from a public/private-key pair using public-key cryptography [3]. A *Bitcoin address* is derived from the public key of an account. The Bitcoin address is used to uniquely identify an account and is used as the destination account when receiving payment from other users. Ownership of the *private key* allows full control of the Bitcoin address associated with that private key. The private key can be used to move funds associated with the corresponding Bitcoin address by creating the digital signature that is required by transactions.

Transactions can be broken down into *transaction outputs*, *transaction inputs* and the *transaction ID*. The transaction inputs are the accounts of the payers and the transaction outputs is where the bitcoins are being sent to i.e. the payee's account. The transaction ID uniquely identifies each transaction [3]. Transaction outputs are fundamental in Bitcoin

[2] http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source.

transactions. Transaction outputs are indivisible chunks of Bitcoin currency that are recorded on the blockchain and are seen as valid and spendable by the Bitcoin network. Unspent Transaction Outputs (*UTXO)* are available and spendable transaction outputs. A user's Bitcoin "balance" is the sum of all the UTXO associated with the user's Bitcoin address.

Transactions consume UTXO which in turn creates new transaction outputs that can be spent by the payee. Every output of a transaction will also contain one or more inputs that indicates where the Bitcoin originated from before the transaction.

In order to transfer bitcoins to an account, the public key of the payee's account must be listed as the destination of the transaction. The payer must also sign the transaction. They do this by digitally signing a hash of the previous transaction and the public key of the next owner [1].

In order for a transaction to be valid, the following criteria must be fulfilled by the outputs claimed and created:

- An output may be claimed at most once.
- New outputs are created solely as a result of a transaction.
- The sum of the values of the inputs has to be greater than or equal to the sum of the values of the newly allocated outputs[3] [3].

## 2.2 Blocks, Mining and Proof-of-Work

A block is a data structure that is composed of a set of transactions and a block header. Blocks on the blockchain are identified via the block header hash. When a new transaction is propagated through the Bitcoin network, it is stored in each node's local mempool. The Bitcoin mempool is a pool of unconfirmed transactions in the Bitcoin network. Each node has their own mempool. The transactions in the mempool may be valid transactions but are not yet confirmed by the Bitcoin network. The transactions are not seen as confirmed transactions until they are included in a block that is on the blockchain. The process in which transactions are taken from the mempool and included in blocks is known as *mining*. As more miners join the network, the difficulty of the PoW gets harder and harder[4], in such a way the average time to mine a block is approximately every 10 min [4].

The process of mining a block is a computationally difficult process. The nodes which attempt to mine a block, known as miners, must find the solution to Bitcoin's PoW problem. The PoW problem consists of finding an integer value, known as a *nonce*, that when combined with the block header, will provide a hash with a given number of leading zeroes, known as the difficulty [3]. As cryptographic hashes are a one-way function [5], the only solution for miners to find the nonce that will satisfy the difficulty of the block is to use a brute-force approach, testing different values for the nonce until a suitable hash is found. The nonce which satisfies the difficulty check of

[3] If the inputs are greater than the required outputs, miners may collect the difference as a transaction fee or may be sent back to the payee's address as change.

[4] It may also decrease in difficulty, depending on the average block creation rate of the previous 2,016 blocks.

the block, known as the golden nonce, is therefore very difficult to find but once found, is straight-forward to verify it.

Nodes within the network may sometimes have an inconsistent view of the blockchain due to the decentralization nature of the Bitcoin network. This inconsistency may occur when two nodes in the network discover and propagate different blocks at approximately the same time. The two different blocks will propagate through the Bitcoin network, arriving at nodes at different times. The nodes will accept the first block that they received and reject but save the other block when they eventually receive it [1]. Nodes in the network will now have a temporary inconsistent view of the blockchain, as there are now two blocks claiming to be the blockchain head. In order to resolve this inconsistency, nodes can work on either branch of the fork but are likely to work on the block that they received first [7]. The fork would likely be decided when the next block is mined and one branch becomes longer than the other. The longer branch will become the legitimate one and nodes working on the other branch will then switch to the longer one. This occurs as nodes always consider the longest chain to be the legitimate one and will keep working on extending it [1].

### 2.3 The Blockchain

Thus far, when blocks are mined and transactions are placed in the blocks, the blocks do not offer any synchronization or chronological ordering of the transactions. However, this changes when blocks are linked together sequentially, creating a chronological ordering over the blocks and therefore the transactions in the blocks [3]. This sequential formation of blocks is known as the blockchain [8]. When a block is created and propagated through the network, it is added to the blockchain by creating a reference to the latest block (the previous block) on the blockchain. The chaining of each block to the previous block is what creates a chronological ordering of transactions in the network. The referenced previous block is known as the *parent* block. Blocks may only have one parent block but can temporarily have multiple children during a blockchain fork. As every block references the previous block, the blockchain is made up of a single sequence of blocks from the first block, or the genesis block, to the latest generated block [8]. The distance between a block and the genesis block is referred to as its *block height*, and the block that is furthest away from the genesis block is known as the *blockchain head* [3] (Fig. 1).

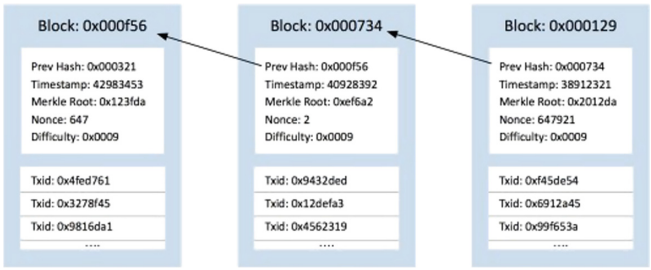


**Fig. 1.** Blockchain representation example [6]

## 3 The Bitcoin Network and Information Propagation

The Bitcoin network is structured as a decentralized P2P network. In a P2P network, nodes participating in the network are seen as peers. Peers are all treated as equal, with shared responsibility in providing network services. The Bitcoin network consists of over 10,000 nodes [9]. Each node in the network implements a version of the Bitcoin protocol through the use of a Bitcoin client. Although there are several Bitcoin clients available to use, the Bitcoin client used by the majority of the nodes in the network is Bitcoin Core, also known as the reference client or the Satoshi client.

Although all the peers in the network are equal, they may have different roles based on the different functions they support. For example, simplified payment verification (SPV) nodes do not keep a copy of the full blockchain and do not participate in mining. They are lightweight nodes that integrates the wallet and routing function, designed for peers with limited resources. In contrast, full nodes may be either a full blockchain node when it includes routing and full blockchain functions, or a solo miner when it includes routing, the full blockchain and mining functions [9]. However, in order to participate in the Bitcoin network, nodes must implement a routing function[5]. The routing function includes network discovery of new peers, establishing inbound and outbound connections, validating transactions and blocks and propagating information through the network [9].

When new transactions or blocks are created in the Bitcoin network, they must be broadcasted to the entire network to inform the peers in the network of the new transactions/block. As the Bitcoin network is a decentralized P2P network, there is no central authority to distribute the transactions/blocks to every peer in the network. As nodes in the network are only aware of their directly connected neighbours, Bitcoin implements a gossip-based flooding protocol to propagate transactions and blocks across the network.

When propagating information across the Bitcoin network, a node maintains a message queue for all of their connected neighbours. This message queue may contain different types of messages that a node may want to send to their neighbours, such as transaction hashes or block hashes etc. Along with the message queue, there is a timer associated with each neighbour. All the messages within the message queue will be sent to the associated neighbour when the timer elapses. The time-out is calculated using a Poisson distribution [10].

In order for a node not to send the same transactions or blocks that their neighbouring peers may already have, transactions and blocks are not forwarded directly to their neighbours. Instead, an **INVentory** message or **INV** message is sent to their neighbours. The INV message transmits one or more inventories of objects known to the transmitting peer and are now available to be requested from the transmitting peer if the receiving node is missing one or more of the inventories of objects in the INV message [3]. If the receiving node requires any of the transactions or blocks within the INV message, they will respond to the sender node with a **GETDATA** message, which contains the hashes of the information the node requires. Once the GETDATA message

[5] Users may turn off the routing function in Bitcoin Core if desired.

is received, the sender node will send the requested block or transaction via individual *block* or *tx* messages [3]. However, if a node receives an INV message that contains transactions and blocks that the node already possesses, the node will simply ignore the INV message, and not respond with a message to the sender node.

Although sending INV messages to neighbouring peers will prevent the peers from receiving duplicate transactions, peers may still receive duplicate INV messages for the same transaction. This occurs as a node's neighbours does not know which transactions the node currently has or is missing. Therefore, if a node's neighbour recently received new transactions, they will add it to the INV message that will be sent to the node as they assume the node might not have the transactions they just received. As every node's neighbours may think the same, a node may receive an INV message for the same transaction from all of their connected neighbours (125 worst case) whereas 1 INV message would have sufficed to send the transaction to the node (Fig. 2).

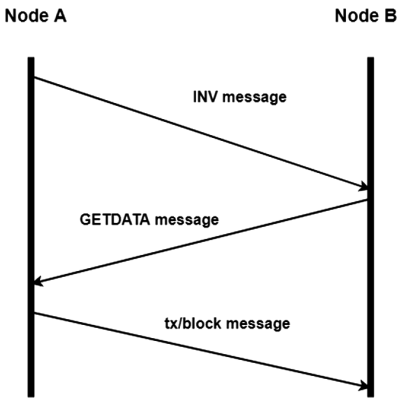


**Fig. 2.** Messages exchanged between two nodes for information propagation

## 4 Related Work

Fadhil et al. present a new protocol, Bitcoin Clustering Based Super Node (BCBSN) as a mechanism to speed up information propagation in the Bitcoin network [13]. In this protocol, the Bitcoin network is divided into geographically diverse clusters. Within each cluster, there is a cluster head or super node responsible for maintaining the cluster. Each peer is connected to a cluster head, and each cluster head is connected to other cluster heads. The claim is that this would reduce the propagation delay as it reduces the number of non-compulsory hops that blocks, or transactions require to reach all the peers in the network. Nodes at each cluster are geographically localized, with the hope of reduction in the link latencies between nodes at each cluster. The BCBSN protocol resulted in a reduction of the transaction propagation time variances, compared to that of the normal Bitcoin network. Possible limitations of the BCBSN protocol may include a successful attack on a cluster head. By successfully attacking a cluster head, the nodes in the associated cluster are unable to connect to

connect to the rest of the network as the cluster head was their means of contact to the rest of the network. If the cluster head was infiltrated by a malicious node, they have essentially partitioned the nodes within the cluster from the rest of the network and may carry out an eclipse attack [15]. As nodes in the clusters are geographically localized, this may make the network highly prone to partitioning.

Following on from BCBSN, Fadhil et al. proposed a proximity-aware extension to the current bitcoin protocol, named Bitcoin Clustering Based Ping Time Protocol (BCBPT) [14]. Based on their previous work BCBSN, which placed nodes in clusters based on their geographic location, BCBPT will place nodes in clusters based on their ping latency. Nodes that are geographically close could be quite far away from each other on the physical internet [14]. The results of BCBPT show that the protocol maintains an improvement in variances of delay over their previous work, BCBSN. This may be due to the fact that in BCBSN, clusters are based on geographic location, meaning they could be close geographically but far away on the physical internet. By creating clusters based on ping latencies, Fadhil et al. concluded that proximity awareness in the physical internet improves delivery latency with a higher probability than clusters based on geographic locations. The protocol is split into two phases:

1. Distance Calculation
2. Cluster Creation and Maintenance

### 4.1 Distance Calculation

In the distance calculation phase, each node is responsible for gathering proximity knowledge regarding discovered nodes. This is done by calculating the distance in the physical internet between the node and the discovered nodes. Proximity is defined as how far a node is from another node in the physical internet.

### 4.2 Cluster Creation and Maintenance

When joining the network for the first time, a node *N* will learn about other available Bitcoin nodes in the network from a list of DNS services. The node *N* will calculate the proximity distance to each of the discovered nodes. The node *N* will then send a JOIN request to the closest node *K* of the discovered nodes. Once node N establishes a connection with node *K*, it will receive a list of IPs of nodes that is in the same cluster as node *K*. Node *N* will then connect to all the nodes in the cluster. If node *N* discovers a node that is physically closer than the current cluster, node *N* will leave to join the nearer cluster.

Although the transaction propagation time and variances are lowered in the proposed protocol, the same issues from BCBSN can be applied to the proposed protocol. As mentioned by Fadhil et al., they identify that eclipse and network partition attacks have great potential due to the clustering based on countries. An attacker might concentrate a number of bad peers within a cluster in order to create a malicious cluster on the network [14].

Marcal [10] proposes a new protocol for the dissemination of transactions in the Bitcoin network. The protocol proposes a bias to disseminate transactions to neighbours that are more likely to reach miners quickly, as miners are the nodes that need knowledge

of the transactions in the network as they are responsible for placing the transactions in blocks, and subsequently placing the block on the blockchain.

The protocol encompasses three changes to the Bitcoin dissemination protocol:

1. Nodes maintain for each of their neighbours, a list of transactions sent by their neighbour and how long it took for these transactions to be included in a block.
2. Nodes maintain for each of their neighbours, the time it took to disseminate a new block to the node.
3. Use the metrics collected above to rank their neighbours and prioritise the dissemination of transactions based on the rankings.

The proposed protocol was able to reduce the bandwidth usage by 10.2% and reduce the number of messages exchange in the network by 41.5%. Some issues with the aforementioned protocol is that the commit time of transactions may increase as transactions are reaching miners, but may not necessarily reach the miner who is going to mine the next block [10].

Other related works focus on exploiting the current dissemination protocol in order to gain an advantage for the attacker or put the victim at a disadvantage. For example, Courtois and Bahack [11] indicate miners could have a specific mining strategy known as *selfish mining*. In selfish mining, nodes purposely withhold mined blocks from the network, only revealing the mined block(s) in a selective way which benefits the selfish miners. Eyal and Sirer [12] show that through the use of selfish mining, the selfish pool's reward exceeds its share of the networks computational power.

## 5 Probabilistic Flooding

As described in Sect. 3, when propagating a transaction through the network, an INV message will be sent to the node's neighbours 100% of the time. This flooding mechanism implemented by Bitcoin produces many duplicated INV messages being received by nodes in the network.

The solution and protocol change that we propose changes the current flooding mechanism approach that was described above to a probabilistic flooding approach. Our approach aims to maintain a probability for each of the node's neighbours. This probability is the probability that a node will send an INV message to the associated neighbour. The probability is calculated based on the number of INV messages sent to the neighbour and the number of GETDATA messages received in return from the neighbour.

**Formula for Calculating a Neighbours Probability**

$$neighbour\ Probility = \frac{total\ ttet\ Data\ From\ Neighbour}{total\ Inv\ sen\ to\ Neighbour}$$

The idea of sending INV messages based on probability is centered around the fact that nodes in the Bitcoin network have a large variance in the number of connected neighbours. A node may be well-connected, and in the best case, have 125 neighbours whereas another node may have as low as 8 neighbours. The node with 125 neighbours is more likely to have already received the transactions contained in the INV message that it

received and therefore will not reply to the INV message with a GETDATA message. The idea of the protocol change to a probabilistic flooding approach is based on the criteria that well-connected nodes will already have the transactions contained in an INV message and will not need to receive an INV message 100% of the time, whereas a node that is less connected may need to receive an INV message the majority of the time.

For example in Fig. 3, node *A* will send an INV message to node *B* with a higher probability than sending an INV message to node *C*. This is due to the fact that node *B* has a total of three neighbours and is less connected than node *C*, who has a total of five neighbours. As node *C* is more well-connected, it is more likely that node *C* may already have the transactions contained in the INV messages, whereas node *B* is less likely to have the transactions as it has two less neighbours than node *C*. The probability is based on previous message exchanges between the nodes. Node A may have previously sent 54 INV messages to node *C* and may have only received 34 GETDATA messages in return. In this case, the probability that node A will send an INV message to node *C*, based on the formula mentioned above, will be 63% (34/54). The probability of sending an INV message from node *A* to node *B* is also based on the exchange of previous messages between the two nodes. In this case, node *A* sent 77 INV messages to node *B*, whilst receiving 64 GETDATA messages in reply. Based on the formula of calculating neighbour probability, the probability node *A* will send an INV message to node B will be 83% (64/77).

From the example, node *B* replies to INV messages more times than node *C*, and therefore will have a higher probability of receiving an INV message in the future from node *A*. The higher probability can be attributed to the fact that node *B* only has three neighbours and is not as well-connected as node *C*, who has five neighbours. As node *B* has fewer neighbours, this leads to fewer options for which it may receive an INV message for certain transactions in the network, leading to a higher GETDATA response rate when it receives an INV message. Conversely, node *C* is better connected than node *B*, having five neighbours. This leads to more avenues for which node *C* may receive INV messages, therefore leading to a lower response rate to INV messages. As node *C* has more neighbours, this leads to a higher probability that they have already received the transactions contained in the INV message.

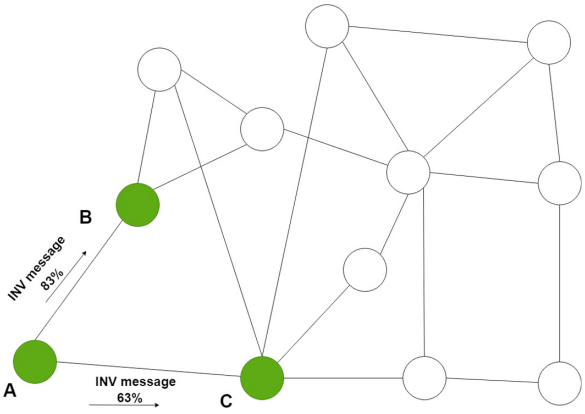


**Fig. 3.** Probabilistic flooding example

## 6 Evaluation

In order to test our protocol changes we ran a number of simulations. *Joao Marcal's bitcoin-simulator*[6] [10] supports the newest versions of Bitcoin and recorded a number of important metrics that would be vital to compare and contrast the current Bitcoin protocol, and the proposed probabilistic flooding approach. The metrics recorded by the Bitcoin simulator were as follows:

- Average number of INV messages sent per node
- Average total number of sent messages per node
- Percentage of duplicated messages received per node
- Total transactions created
- Percentage of transactions created and committed
- Total number of forks created

The simulator is an event-driven simulator, where the behaviour of each node in the network is defined by a deterministic state machine, that consumes events and produces events. Each cycle in the simulation represents a second in real-time. The default settings and the settings for which the results are formed are based on the following configurations:

- Number of nodes in the network - 625
- Number of miners - 5
- Minimum neighbourhood size of each node - 8
- Number of cycles - 208800

Algorithm 1 represents how the probability of sending an INV message to a specific neighbouring node is calculated.

**Algorithm 1.** Function to calculate the probability of sending INV message to each neighbouring node

```
1: function get probability(myself, neighbouring_node)
2:     total_inv_sent ← get_total_inv_sent(myself, neighbouring_node)
3:     total_getdata_received ← get_total_getdata_received(myself, neighbouring_node)
4:     probability_to_send ← total_getdata_received/total_inv_sent
5:     return probability_to_send
```

Algorithm 2 is the function that will determine whether or not a node will send an INV message to its neighbouring node. Algorithm 2 is called every cycle for every node, as long as the adjusted probabilistic flooding mechanism is enabled in the simulation. Algorithm 2 will firstly get the current time of the simulation. For each of the node's neighbours, the algorithm will receive the calculated probability of sending an INV message to that specific neighbour based on Algorithm 1. Associated with each neighbouring node is a timer which is calculated using a Poisson distribution [10]. The node will receive the timer for the neighbouring node and will determine whether or not the

[6] https://github.com/JoaoBraveCoding/bitcoin-simulator.

timer elapsed for sending a message to the neighbouring node, based on the current time received at the start of the algorithm. If the timer elapses and the probability of sending an INV message is satisfied, the node will send the INV message to the neighbouring node and increment the INV messages sent to that neighbour counter. This is to ensure that the data used in Algorithm 1 to calculate the probability of sending an INV message to neighbouring nodes is up to date. However, if the timer elapsed but the probability of sending is not satisfied, the INV message scheduled to be sent to the node is ignored.

As each cycle represents a second in real-time, the experiments were run for a simulation time of 58 h. The first five hours and the last five hours of the simulation were discarded in order to study the system in a stable state. The simulator is tuned to generate blocks at the Bitcoin desired rate of 1 block per 10 min, as well as creating 2 transactions per second.

**Algorithm 2.** Broadcast Inventory Messages

```
1: function broadcast_invs_prob_flooding(myself)
2:     now ← get_current_time()
3:     for neighbour in neighbourhood do
4:         probability_to_send ← get_probability(myself, neighbour)
5:         time_to_send ← get_time_to_send(neighbour)
6:         timeout ← now > time_to_send
7:         send_inv_based_on_prob ← random.random() < probability_to_send
8:         if timeout and send_inv_based_on_prob then
9:             sim.send(myself, neighbour, INV message)
10:            myself.increaseInvSentToNeighbour(neighbour)
11:        if timeout and not send_inv_based_on_prob then
12:            myself.increaseIgnoredMessagesCount()
```

The most relevant and important metrics when comparing the two protocols are:

- **Percentage of Committed Transactions** is the most vital metric when comparing the two protocol changes. The percentage of committed transactions indicates whether or not every transaction that was created during the simulation period was eventually committed into a block. As Bitcoin is the most popular cryptocurrency and has a market cap of approximately 72$ billion, it is essential that every transaction that is created is eventually committed in a block to maintain the reliability of the system. The main objective of the protocol change is to reduce the number of redundant messages being exchanged on the network. However, if the protocol change negatively impacts the percentage of committed transactions, reducing the 100% commitment rate of transactions then regardless of the potential reduction of redundant messages, a less then 100% committed transactions rate would be detrimental to the system and unacceptable.
- **Total Number of Messages Sent Per Node** is an important metric when comparing the two protocols. As the main objective of the probabilistic flooding approach is to reduce the number of redundant messages exchanged on the network, comparing the total number of messages sent per node between the two protocols would indicate exactly how many messages were saved as a result from the protocol.

– **The Commit Time of Transactions** is also an important metric to consider when comparing the two protocols. The commit time represents the time between when a transaction was created to when it was placed in a block. As commit times of transactions is an extremely important aspect in cryptocurrencies, having an increased commit time when implementing the proposed probabilistic flooding approach may not be worth the trade off in potential messages saved within the network.

### 6.1 Percentage of Committed Transactions

As mentioned previously, the most important metric when comparing the proposed probabilistic flooding protocol to the current Bitcoin flooding protocol is the percentage of committed transactions. Both protocols produced a 100% transaction commitment rate, committing every transaction to a block during the simulation period. We can conclude from these results that the adjustment of the flooding protocol to a probabilistic flooding approach did not have an adverse effect on the number of committed transactions during the simulation. A 100% transaction commitment rate ensures that the system remains reliable when the probabilistic flooding approach is implemented.

### 6.2 Transaction Commit Time

Another important metric that was previously discussed when comparing the two protocols is the time taken to commit a transaction. Figure 4 represents the average time taken for a transaction to be committed into a block for the two protocols. As you can see from Fig. 4, the time taken to for a transaction to be committed into a block for both protocols were very similar. The small difference between the two protocols is negligible. This is very important as the results indicate that changing the flooding protocol to a probabilistic flooding approach does not have a negative effect on the transaction commitment time. As transaction commit time is an extremely important aspect for cryptocurrencies, if there was a significant increase in transaction commit time when changing to the probabilistic flooding approach, the potential reduction in redundant messages may not be worth the trade-off in increased transaction commit time.

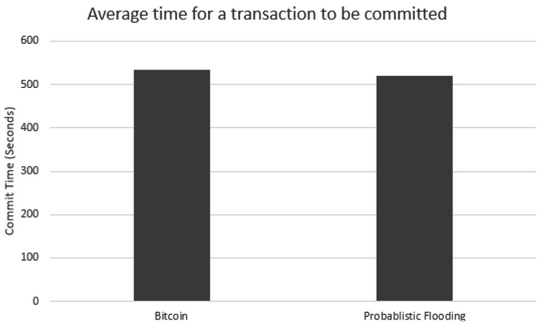


**Fig. 4.** Average time taken to commit a transaction.

### 6.3 Total Sent Messages

As mentioned in Sect. 1, the main objective of changing the current Bitcoin flooding protocol to the probabilistic flooding approach is to reduce the number of redundant and duplicated messages that are currently being generated in the Bitcoin network. Figure 5, represents the number of total sent messages gathered from our simulations for the two protocols.

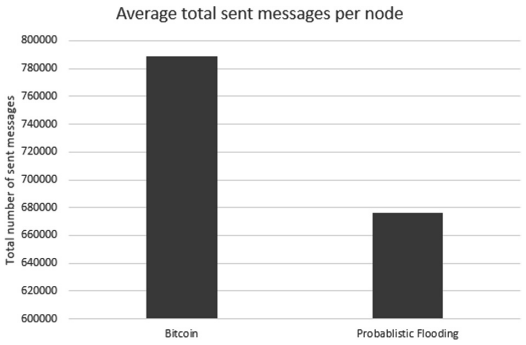


**Fig. 5.** Average total sent messages per node.

The results show that when running the probabilistic flooding approach, there was a significant decrease in the total number of messages sent per node during the simulation. When running the Bitcoin flooding protocol, the simulation showed that there was approximately 790,000 total messages sent per node, whereas when the simulation was run with the probabilistic flooding protocol implemented, there was approximately 675,000 total sent messages per node. This results in a **15%** reduction in the total number of messages sent per node during the simulation period. As the 115,000 reduction in messages mentioned are the amount of messages saved per node, the total number of messages saved throughout the entire network can be estimated at approximately **70** million messages as there are 625 nodes participating in the network during the simulation. Figure 6 represents the aforementioned statistics between both protocols.

The limitations of using simulations when testing the proposed protocol are as follows:

- The current simulator does not take into account other factors that may affect transactions being included into blocks such as incentives i.e. transaction fees.
- Each node in the simulation network has a wide variance of the number of neighbours they are connected to, with 8 neighbours being the minimum. However, an important characteristic of the Bitcoin P2P network is that nodes are able to join and leave the network as they desire. A node may receive a low probability of being sent an INV message as they are highly-connected in the network, however, many of their neighbours could potentially leave the network. This leads to the node still receiving a low probability of being sent an INV message, even though they may be less-connected than before. An area of future work could be to run further simulations.

| | AVG INV SENT PER NODE | % OF DUPLICATED INV | AVG TOTAL SENT MESSAGES PER NODE | % OF TRANSACTONS ADDED TO BLOCKS | Transaction Commit Time (Seconds) |
|---|---|---|---|---|---|
| Full Bitcoin Protocol | 270,633 | 8.33% | 789,220 | 100% | 534 |
| Probabilistic Flooding Approach | 191,335 | 7.45% | 677,222 | 100% | 519 |
| Difference | -79,278 | -0.88% | -111,988 | 0% | -15 |
| Total Network Savings | 49,548,750 | | 69,992,500 | | |

**Fig. 6.** Table representing the aforementioned statistics.

- where the probabilities of each neighbours are reset after a certain period of time. This allows for nodes who were initially well-connected but then as their neighbours leave the network, become less-connected, and vice versa, to receive updated probabilities from each of their peers.
- The simulator is tuned to create 2 transactions per second (TPS), leading to a daily average of 172,800 transactions, which were the number of daily transactions created when the simulator was created. However, Bitcoin can handle up to 7TPS [16] but currently has a daily trading volume of approximately 4TPS[7].

## 7 Conclusion

In this paper, we proposed a novel protocol that aims to reduce the number of redundant messages being generated by the current flooding protocol. The proposed protocol changes the current flooding protocol implemented by Bit-coin to a probabilistic flooding approach. The proposed probabilistic flooding approach presented in this paper is based on the idea that well-connected neighbours will more likely not respond to an INV message compared to a node that is less-connected, therefore the probability of sending an INV message to a less-connected node is higher than that of a well-connected node.

As we have shown in Sect. 6, the proposed protocol is able to significantly reduce the total number of messages being exchanged on the network, whilst maintaining the reliability of the system. The number of INV messages sent per node and the total number of messages sent per node decreased by **29%** and **14%** respectively when running the probabilistic flooding protocol. During the 58 h simulation period, the total number of messages saved when running the probabilistic flooding approach when compared to the current flooding protocol was approximately 70 million messages.

[7] https://www.blockchain.com/charts/n-transactions.

The proposed protocol met the objectives that it aimed to achieve - reducing the number of redundant messages on the network whilst maintaining the reliability and resilience of the system. We have shown that the current flooding protocol implemented by Bitcoin for the dissemination of information across the network is inefficient and wasteful, and have proved that the mechanism can be improved upon whilst maintaining the reliability and integrity of the system. This leads to many possible, alternate flooding solutions to the current flooding protocol for future work.

## References

1. Nakamoto, S., et al.: Bitcoin: a peer-to-peer electronic cash system (2008)
2. Essaid, M., Kim, H.W., Park, W.G., Lee, K.Y., Park, S.J., Ju, H.T.: Network usage of bitcoin full node. In: International Conference on Information and Communication Technology Convergence (ICTC), pp. 1286–1291. IEEE (2018)
3. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P 2013 Proceedings, pp. 1–10. IEEE (2013)
4. Ghimire, S., Selvaraj, H.: A survey on bitcoin cryptocurrency and its mining. In: 26th International Conference on Systems Engineering (ICSEng), pp. 1–6. IEEE, December 2018
5. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL — a one-way hashing algorithm with variable length of output (extended abstract). In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 81–104. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_54
6. Sharkey, S., Tewari, H.: Alt-PoW: an alternative proof-of-work mechanism. In: IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON), San Francisco, California, USA, 5–8 April 2019
7. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: research perspectives and challenges for bitcoin and cryptocurrencies. In: IEEE Symposium on Security and Privacy, pp. 104–121. IEEE (2015)
8. Liu, Y., Chen, X., Zhang, L., Tang, C., Kang, H.: An intelligent strategy to gain profit for bitcoin mining pools. In: 10th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, pp. 427–430. IEEE (2017)
9. Deshpande, V., Badis, H., George, L.: BTCmap: mapping bitcoin peer-to-peer network topology. In: IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), pp. 1–6. IEEE (2018)
10. Maral, J.E.: Adaptive information dissemination in the bitcoin network (2019)
11. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718 (2014)
12. Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018)
13. Fadhil, M., Owenson, G., Adda, M.: A bitcoin model for evaluation of clustering to improve propagation delay in bitcoin network. In: IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) and 15th International Symposium on Distributed Computing and Applications for Business Engineering (DCABES), pp. 468–475. IEEE (2016)
14. Owenson, G., Adda, M., et al.: Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network. In: IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 2411–2416. IEEE (2017)

15. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium (USENIX Security 2015), pp. 129–144 (2015)
16. Miraz, M., Donald, D.C.: Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities. In: Annals of Emerging Technologies in Computing (AETiC), vol. 3 (2019)