BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2024
http://excel.fit.vutbr.cz

Discrete modeling of transaction propagation in Bitcoin

Tomáš Marek*

Abstract
The Bitcoin network is one of the largest cryptocurrency networks in the world. The number of reachable nodes is currently greater than 18,500 nodes [1]. Transactions sent over the network are propagated by a specific algorithm based on an exponential distribution function. The aim is to simulate the Bitcoin network and identify any significant nodes and possible source nodes of the transaction. The Bitcoin network simulation model with simplified Bitcoin client behavior was implemented in the OMNeT++ simulator. To keep track of the transaction information, a monitoring node was added to the Bitcoin network. This monitoring node collects the transaction information and exports a CSV file, which is analyzed. The analysis script was able to identify the significant nodes and the possible source node of a transaction.

*xmarek80@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The motivation is to explore the Bitcoin network by simulating it with a highly simplified Bitcoin client model that can create, receive, and send messages representing Bitcoin transactions and operate according to the algorithm from the Bitcoin Core Project implementation [2]. The main objective is to identify the significant nodes and possible source nodes of the transaction. Monitoring is ensured by adding a monitoring node to the Bitcoin network. This monitoring node is connected through outbound connections to all reachable nodes on the Bitcoin network. When the monitoring node receives a transaction, it writes the information about it to the CSV file, which is later analyzed. Since the monitoring node is connected to all reachable nodes, it was possible to identify the significant nodes and the possible source node of the transaction, or atleast the node that is close to it, in case it is behind NAT.

2. Implementation

The simulation was performed in the OMNeT++ simulator. The simulation model contains an optional number of Bitcoin reachable nodes, unreachable nodes, and monitoring nodes. A user can specify the hop distance and also whether or not the monitoring node should propagate transactions just like a Bitcoin client. The hop distance in this case indicates the number of nodes that do not accept inbound connections, which means that they are unreachable for other nodes, including the monitoring nodes. The hop distance also means the number of nodes through which the transaction must pass from the source to reach its destination. In this scenario, the source is bitcoinNode0 and the destination is the monitoring node. An example of a simple **topology** generated with 5 Bitcoin nodes, 1 monitoring node, and a hop distance of 1, is shown in Figure 1 on the poster. The topology is generated automatically by a script with parameters given by a user. The bitcoinNode0 is the source node that generates the transaction. The transactions in the Bitcoin network are transmitted to their peers by a three-step process shown in the sequence diagram in the bottom right corner in Section **Tx propagation process**.

This simulation represents only the process of sending inventory (INV) messages, which contains a Transaction ID (TXID). Inventory messages are broadcast to all peers when the TXID is not already known by the current node (meaning that the current node received this TXID for the first time, so it broadcast it to other peers). Each node has its own space for saving the received transactions to remember them, and its called a mempool. The transactions are propagated to their peers based on the type of connection which can be inbound or outbound. An inbound connection is initiated by a remote peer who wants to connect to a node. In contrast, a node initiates an outbound

connection to connect to a remote peer. Both types of connection use an exponential distribution when it comes to transmitting the transactions. The exact distribution function `GetExponentialRand()` from the Bitcoin Core Repository [2] was used. The difference in connections is in the average relay interval. Outbound connections use an interval of 2 seconds for broadcasting the transactions, and inbound connections use 5 seconds. The algorithm implemented to receive and transmit the transaction is shown in Section **Pseudocode of the Bitcoin client propagation algorithm**. Another difference between inbound and outbound connections is that for the inbound, the exponential distribution is called only once meanwhile for the outbound it is calculated separately for each node.

The monitoring node function in the topology is to simply collect information about the propagated transactions. This behavior is ensured by connecting the monitoring node to all reachable Bitcoin nodes through outboound connections. Additionally, the monitoring node can also propagate messages as a Bitcoin client with the same propagation algorithm. The simulation ends when the monitoring node receives the transaction from all connections. The simulation output is a **CSV file** that contains 4 types of records described below:

- run - number of the simulation run. The simulation is repeated several times to obtain meaningful results. In that case, the run number needs to be tracked to differ the collected data.
- TXID - Transaction ID representing unique identifier of the transaction in the Bitcoin network.
- peer - name of the neighbor who sent the transaction to the monitoring node
- timestamp - time in seconds at which the transaction was received by the monitoring node

3. Conclusion

The main objective of the simulation was to try to identify statistically significant nodes in terms of transaction propagation. A statistically significant node is considered to be a node that, in most cases, propagates a transaction in a faster time than others and thus is at the top positions of the CSV file in most simulation runs. Two types of metric were used for this analysis, and it was the position and timestamp at which the node broadcast the transaction. The simulation was run several times with different numbers of Bitcoin nodes (1000, 2000, 5000, 10000, 15000) and other parameters in various combinations. The differences between the results for simulations with a large number of nodes and partial number of nodes were minimal and thus had no effect on the overall result. The analysis script was able to identify significant nodes and determine the possible source node of the transaction, or atleast the node that was close to it, in case the source node is behind NAT.

Another result obtained is related to the adjustable parameter if the monitoring node should propagate messages. When the monitoring nodes are set to propagate messages, the messages spread throughout the network faster, and the entire simulation is completed in a faster time. This means that monitoring nodes can enable faster propagation of transactions and thus can affect the network in some way. From further running analyzes, it was evaluated that the more monitoring nodes in the topology, the faster the transaction propagated throughout the network. The output graph in Figure 1 shows the time for each simulation run in which the transaction was received by the monitoring node from all nodes on the Bitcoin network. The graph also shows the average transaction propagation time for both files, which is calculated from all runs.

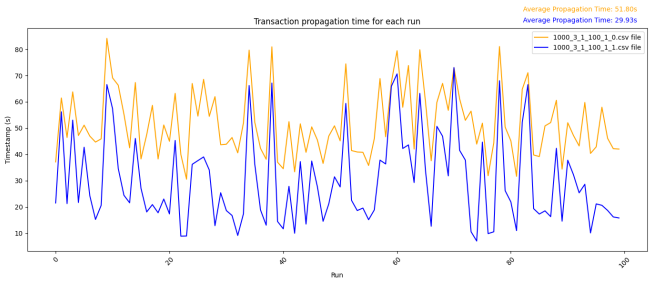

Figure 1. Comparison between the simulation with the monitoring nodes that propagate transactions (blue record) and the simulation where the monitoring nodes do not propagate (orange record) them. The comparison shows a significant speedup in transaction propagation when the monitoring nodes propagate them.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. Jan Zavřel, for his guidance, support, valuable feedback, and expertise throughout my research process.

References

[1] BITNODES. Bitcoin network 1 year chart. online, 2024.

[2] Bitcoin Core Repository. Github. online, 2009, 2024.