[[1]](#footnote-1)

Analysis of the Lighting Network(2021)

Joseph Lazarus, Puri Rudick, and Rick Fontenot

*Abstract*— **Bitcoin is the most established cryptocurrency in the market but is still rapidly developing in structure. The original decentralized transaction system now has added second level protocols to facilitate larger volumes of transactions with faster clearing times and lower fees. In this emerging second level transaction marketplace, there are low barriers to entry and opportunities to optimize structure and revenue as a transaction clearing house (node). Our goal is to analyze publicly available data on the market, set up our own node to process transactions, and study methods of optimizing our node structure and fee rates. Beyond the timeline of this initial project and as we collect private data on our node transactions, we aim to build models predicting total revenue as a response to our fee rates and liquidity. The models optimum setup would then be tested on our own node and iteratively updated with new data. Phase 2 of this project could potentially be evaluated as a capstone project.**

*Index Terms*— Bitcoin, Blockchain, Cryptography, Lightning Network, Network Analysis, Optimization, Payments,

# Introduction

B

itcoin is a digital cryptocurrency created in 2009. Unlike fiat currency, bitcoin is created, distributed, traded, and stored with the use of a decentralized ledger system, known as a blockchain. It has lower transaction fees than traditional online payments using centralized banks and government issued currencies.[1]

# On Chain Vs Off Chain Transactions

Bitcoin uses a blockchain as a ledger to record all transactions. All bitcoin in existence can be found somewhere on the blockchain. All Bitcoin transactions recorded on the blockchain are considered **on-chain transactions** and the ledger is available to the public. Bitcoin is locked in an address, and, using the private key corresponding to that address, you can sign a transaction sending this bitcoin to a new address. All on-chain transactions must pay a transaction fee in order to be included in a block. The higher the fee, the faster the transaction will be confirmed. On-chain transactions are highly secure and reliable; however, transaction fees are rising based on demand and the time to confirm and clear a transaction can take 10 minutes to multiple hours. Bitcoin blocks can only hold a certain number of transactions so this method of transaction lacks scalability.[2]

## Off Chain transactions

Off-chain transactions occur through a private peer-to-peer network where a channel is set up to make many transactions between merchant and/or individual pairs. When the parties have concluded a group of transactions, similar to a tab, the net transfer amount is recorded as a single on-chain transaction and becomes part of the public blockchain ledger. The benefits are lower transaction fees, faster (near instant) confirmation and clearing of transactions and improved scalability for large numbers of transactions. The tradeoffs are higher risk private transactions, capital to meet liquidity requirements, and capacity limits on channels which limit large transactions.[2]

## Blocksize Dilemma

The reader at this point may wonder why not increase the blocksize. Doing so would increase the throughput of the system and eliminate the need for a second layer payment network. The answer is complicated. Bitcoin protocol is not controlled by anyone person or body. The Bitcoin miners are one body within the network that assert influence. The miners solve a hash problem to assemble a new block. This earns them six Bitcoins per block and processes transactions with the remaining space on the block. Each block has a theoretical maximum size of 4 megabytes but a more realistic maximum size of 2 megabytes.[9] User who run an active node assert another vector of influence. Nodes validate transactions and blocks and accepts transactions from other nodes to support the network. Nodes check each transaction by verifying the rules of the Bitcoin Protocol are met. If a transaction breaks the rules the nodes wont to accept it as a valid transaction. This prevents miner’s from creating a Bitcoins out of thin air. Or unilaterally changing the Blocksize limit. Bitcoin Core developers are the third body within the network that exert influence. They test the protocol, debug and support miners and nodes.

Thus, changing the protocol requires a large consensus of compute power on Bitcoin’s network from miner’s and nodes, that are supported by developers. Such a change was proposed in 2017. “Big Blockers” arguing in favor of increasing the artificial block limit. They argued doing so would benefit the network by increasing throughput. “Small Blockers” resisted the change. Their argument was that large blocksizes would inhibit ordinary individuals from running a node. This they argued would centralize the network to a few nodes that could handle such dramatic increase in compute and bandwidth requirements.

August of 2017 a contingent of big blockers hard-forked the protocol and started Bitcoin Cash. This created a non-backwards compatible implementation protocol that would enable bitcoin’s use as a medium for daily exchange. Less than a year later Bitcoin Cash underwent a hard fork that saw the creation of Bitcoin SV. Both of these new protocols lacked the general support from developers, miners and nodes.

During the same period work on the Lightning Network continued. Second layer payment network gained wider adoption. The market for Bitcoin has increased and Bitcoin hard forks and trading at less than 1% of Bitcoin’s value. Seemingly the small blockers won that skirmish. Consensus for achieving scalability through a second layer payment network has won out, for now.[10]

# What is the Lightning Network?

Bitcoin Lightning Network is a second-layer protocol designed to enable off-chain Bitcoin transactions. It aims at breaking the trade-off between block size and centralization by processing most of the transactions off-chain: it is a 'layer 2' protocol that can operate on top of Blockchain-based cryptocurrencies such as Bitcoin.  Because they are not recorded on the blockchain, and thus require no mining, Lightning payments are extremely fast and cheap. Unlike the Bitcoin network, Lightning transactions are not publicly broadcast and instead, individual Lightning nodes transact with one another privately.

For example, a transaction from party A to party D is not direct and channel is opened Party A 🡪 Node B 🡪 Node C 🡪 Party D. Each node comprising the channel earn a fee for each transaction run through the channel in each direction. When all transactions are complete a single net on-chain transaction is made to the public ledger. The combination of small node transaction fees and the final larger on-chain fee result in a lower fee per transaction and transactions clear much faster. [2]

I In a payment channel, cryptographic protections are used to ensure that channel updates in both directions are executed atomically, i.e., either both or neither of them are performed. In addition, incentive-based protections are also implemented to prevent users from stealing funds in a channel, e.g., by committing a revoked state. Similar techniques allow payment routing for longer paths. [3]

Furthermore, payment router intermediaries are financially motivated to relay payments as they are entitled to claim transaction fees after each successfully routed payment. Lightning Network (LN) as a Payment Channel Network (PCN) consists of nodes representing users and undirected, weighted edges representing payment channels. Users can open and close bidirectional payment channels between each other and route payments through these connections.

# The Goal

Network participation is incentivized through the reward of routing payments. Payment routing is a passive means of earning Bitcoin. We sought to analyze the Lightning Network to come up with a strategy for maximizing profits for our Bitcoin node.

# Collecting and Analyzing Data

Throughout our work we tried several methods of collecting Data. BTCPayServer is an open-source project that allows online vendors to easily accept and generate invoices. MyNode is a private company that offers licensed software compatible with market available hardware to run a Bitcoin Node. Simulated data from Ferenc Beres et al. Lightning Network Simulation [3]. Once collected the collection plan called for storing the data in MongoDB.

## Issues with Collecting Data from Our Node

BTCPay Server an open-source stack that allows users to run a Bitcoin Node using virtual private clouds (VPCs). Their solution provides an attractive advantage of not having to maintain any hardware. The software stack is easy to run and operate a node, but our team found it difficult to off load this data. Advantages of BTCPay are that users are able to run the open-source software stack on a Virtualized Machine. We built two BTCPay Server nodes. One implementing a solution on AWS EC2 instance. Memory one of our Docker Images filled to its max capacity during the synchronization of the state of the Bitcoin’s Blockchain.

Synchronization is a major step in any implementation of a Bitcoin Node. For node to participate in the network it must first download a copy of the entire Blockchain. Referring to the discussion on the blocksize dilemma the Lightning Network implementation is seeking to solve an issue for the small block size. Even with the small blocksizes the synchronization is no small step. With larger blocksizes this project may not have been attainable for the resources available to our shoestring budget.

With the EC2 instance it was on day 3 of the synch when the docker image filled up crashing our node. Unfortunately, once a docker image is filled to maximum compacity there is nothing a user can do.

We started the process again. This time using LunaNode. BTCPay server and LunaNode collaborated to create a seamless integration allowing users to leverage advantages built in. For example, LunaNode provides a feature for users to temporarily increase compute capacity just for the synchronization process of the node. Useful to decrease the length of time needed for a node to synch with the state of the current blockchain. Dynamic memory capacity solved the docker image problem faced in the EC2 instance.

Once up and running we found that the implementation of BTCPay server a useful for anyone to easily create a Bitcoin node. Yet we found the open-source software lacking in a few key areas of interest. It seems the BTCPay server is highly optimized for the use of online vendors. The software stack can automatically generate invoices. Easily add the ability to conduct transactions on the Onion Network. Even the possibility for users to accept alternative cryptocurrencies. Yet, we were unable to gather data about the state of the Lightning Network.

MyNode is a promising application that allowed the team to take physical control of the node and data storage. This implementation of a Bitcoin node is an on-premises solution. Raspberry Pi 4 provided the compute and requires a 1TB 2.5’’ SSD drive for memory. This implementation is limited to the compute power of the Raspberry Pi and its write to memory speed. Already a slower process, the Blockchain synch with MyNode was slowed further by hardware issues. Raspiberry Pi 3.2 USB connection proved fickle. It had trouble writing to the first SATA connection 3.2 connection. MyNode Troubleshooting steps provided Raspberry Pi preferred 3.2 USB SATA connection. Even with the preferred equipment used the synchronization took more 3 weeks to complete. The delay in gathering data through this method meant moving on to analysis phase without our own nodes data.

## Open-Source Data

GitHub user “FiatJaf” created an open-source PostgreSQL database of the Lightning Network.[3] The data is collected by a python script that uses the following c-lightning, a compliant lightning implementation in C. Bitcoin-Knots a desktop application to run a Bitcoin Wallet. Esplora, an open source Blockchain explorer. Fiatjaf’s script runs once a data and creates snapshots of the network.

Ferenc Beres et al. Lightning Node Simulator providing the team with a simulated Lightning Network traffic data.[2] It used snapshots of the network collected from a period of from 2018 through 2019. Using the Lightning Network simulator, we accepted a few strong assumptions. First, that data from that time period is still relevant today. Fixed payments amount is another strong assumption. However, various distributions such as Poisson on Pareto greatly increases the complexity of the experimentation. At the onset of each simulation randomized capacity between channel and endpoints was used.

Regardless of the source of the data, our node, FiatJaf, and Ferenc Beres et al.’s LNSimulator, we expected the size to be great. A method for storing the data in order to create a data science pipeline required database integration.

# Database Integration

## PostgreSQL Integration

Fiatjaf provides a downloadable PostgreSQL dump file (.dump) for Lightning Network data via his website.[5] This data dump file collects data from 2018 until current date with an update once per day.

The dump method is to generate a text file with SQL commands that, when feed back to the server, will recreate the database in the same state as it was at the time of the dump. pgAdmin provides the utility to dump and restore the PostgreSQL database.

The database from Fiatjaf’s website is massive. It contains over one million records. The team decided to create a database cluster and store the database within local machine.

## MongoDB Integration

Ferenc Beres et al.’s LNSimulator gave our team useful insights into the Lightning Network. The simulator allows users to tweak the internal state of the simulation prior to running. Output files from the simulation came in the form of four csv files and one json file. MongoDB provided a great resource for the team to store, label, and create a data science pipeline from.

The flexibility of json like documents made capturing the output from the csv files possible and a desirable feature. MongoDB collections made tracking the internal state of the simulation that produced a particular output easy as well. By giving a collection a snapshot ID, the team could track what simulation conditions produced a particular output.

Database integration also provided the team with an increase in availability. One person could run a simulation and through a predetermined naming convention could share the results. Allowing any of the team participants to access this data. The sheer size of the Lightning Network data necessitated using databases for analysis. Which could not have otherwise been performed on an ordinary computer importing a csv file.

# Results

## Analysis of Base Fee Rate on the Lightning Network

With the objective finding the optimal base fee rate for our node we analyzed the open-source data on the Lightning Network collected by FiatJaf. In the following diagram (Figure 1), you can see the distribution of base fees that are being used for channels on the Lightning Network. The base fees are announced in the gossip protocol. What we observed are the most used base fees are zero and one thousand Millisatoshis. The Y scale is logarithmic, which indicates that these most common base fee rates are extremely common.

Chart, histogram

Description automatically generated

Figure 1. Distribution Base Fees in Millisatoshis. Note that the Y scale is logarithmic. The most common base fees are 0 and 1,000 Millisatoshis.

Another way to absorb this data is to look at the culminative distribution function (Figure 2). The cumulative distribution function graph shows us that almost all nodes have a lower base fee rate of 1500 Millisatoshis. Is this a good base fee rate? Data will never include transaction amounts, sources, and targets in any form, and it is very unlikely that it will give information on the capacity distribution over the channels, since that would leak information on the actual transactions. Thus, in order to explore this idea, we turned to the LNSimulator.

Chart

Description automatically generatedFigure 2. Cumulative distribution function of Base Fee Rate. What this shows is for every base fee rate, what is the number of nodes that have a lower base fee.

## Analysis of Simulated Lightning Network Transactions

The network in the simulator has a fixed number of nodes *N* – corresponding to all Bitcoin users that may decide to switch to the LN – and is *sparse*, i.e., the number *M* of edges is *M* ≪ *N*2. If we consider node *i* and *j* having fitness *xi* and *xj* respectively, a LN channel, i.e., an edge between them, is added with probability [4]

Text

Description automatically generated Furthermore, payment router intermediaries are financially motivated to relay payments as they are entitled to claim transaction fees after each successfully routed payment. LN as a PCN consists of nodes representing users and undirected, weighted edges representing payment channels. Users can open and close bidirectional payment channels between each other and route payments through these connections. Therefore, LN can be modeled as an undirected, weighted multigraph since nodes can have multiple channels between each other. The weights on the edges correspond to the capacity of the payment channels.[3] By simulating transactions at different traffic volumes and transaction amounts, we shed light on the fee pricing policies.

Keeping in mind the assumptions of LNSimulator, fixed payment amounts and 2019 snapshots of the Lightning Network we found the following distribution of router fees (Figure 3), i.e., the fee charged by a node to route a payment. The histogram shows router fee in Millisatoshis. The most common router fee is between zero to one thousand Millisatoshis, which corresponds with Lightning Network routing base fee rate data from FaitJaf as mentioned in section VII-A.

Chart, histogram

Description automatically generated

Figure 3. Histogram of Router Fees in Millisatoshis. Note, a fee of 0 to 1,000 Millisatoshis is the most common fee. Also supported by our open-source Lightning Network data.

Figure 4 shows the total income distribution in Millisatoshis from simulation. This indicates that most nodes earn between zero to one thousand Millisatoshis, which could be the result from common router fees.

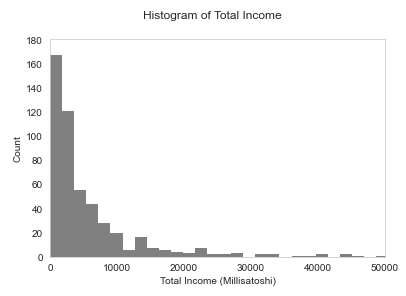


Figure 4. Income Distribution in Millisatoshis: Right Tailed distribution. Because most nodes have a fee of 0 to 1,000 Millisatoshis.

The LNSimulator provides optimal fee output which is itself a simulation run in parallel to the first simulation. The transactions take the map of the network based on path lengths and fees simulate where on the network the sender would be routed for lowest cost. The optimal fee then loops through the possible sources, increases their fees and determines if their income increases or not. If it produces an increase in income the simulation calculates a fee difference and outputs the optimal.

Chart

Description automatically generated with medium confidence

Figure 5. Optimal Fee Distribution. Note, another right tailed distribution. Most optimal increases are between 0 and 4000 MilliSatoshi's.

What does this mean for our optimizing our node? Leaving the routing fee at zero produces no income. Exactly what fee we should start will depend on several factors. Where our node fits into the Lightning Network plays a part in routing algorithm. Establishing channels with active nodes will help improve our nodes position to charge a higher fee. Maintaining an active node will help decrease failed transaction rate. Keeping liquidity in the node also improves volume performance. Nodes that are not adequately funded cannot forward payments behind their own channel liquidity. Unfortunately, the team ran out time.

The learning was steep and our missteps were taken. Bitcoin is only a 10 year old technology and the lightning network is less than half that age. Exploring different node packages took time. Synching the nodes to the blockchain was another large hurdle that ate away out our time. Finding the data sets was another brute force activity. Despite these set backs the team has laid out a foundation of research that can be built upon for further study. Should we or other researchers decide to explore this topic our work provides a methodology for continuing the research.

We release the source code of our research and final paper for further research at https://github.com/JosephLazarus/  
JosephLazarus-bitcoin-lightning-node-database

References

1. J. Frankenfield, “Bitcoin”, https://www.investopedia.com/terms/b/bitcoin

.asp, 2021

1. River Financial, “On-Chain vs. Off-Chain Bitcoin Transactions”, https://river.com/learn/on-chain-vs-off-chain-bitcoin-transactions/
2. F. Beres, I. A. Seres, and A. A. Benczur, “A Cryptoeconomic Traffic Analysis of Bitcoin's Lightning Network”, arXiv preprint, arXiv:1911.09432, 2019
3. F. Beres, I. A. Seres, and A. A. Benczur, “LNTrafficSimulator”, Retrieved from https://github.com/ferencberes/LNTrafficSimulator, 2019
4. FiatJaf, “Lightning Network Data”, Retrieved from https://ln.fiatjaf.com and https://github.com/fiatjaf/lnchannels, 2021
5. BitMEX. “The Lightning Network (Part 2) - Routing Fee Economics”, https://blog.bitmex.com/the-lightning-network-part-2-routing-fee-

economics/, March, 2019

1. D. Easley, M. O'Hara, and S. Basu, “From mining to markets: The evolution of bitcoin transaction fees”, Journal of Financial Economics, 2019
2. Y. Zhang, D. Yang, and G. Xue, “Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks”, In ICC 2019-2019 IEEE International Conference on Communications (ICC), pages 1-6. IEEE, 2019
3. “What is the Bitcoin Block Limit”, Bitcoin Magazine, August 17, 2020, ‘https://bitcoinmagazine.com/guides/what-is-the-bitcoin-block-size-limit’
4. Mannack, Ronald. “The Lightning Network: How to install and (hopefully) make money”, Jul 18, 2018, ‘https://medium.com/coinmonks/the-lightning-network-how-to-install-and-hopefully-make-money-6e3058e3fa7c’

1. 1 “Bitcoin summary is supported by the Investopedia website. Reference: https://www.investopedia.com/terms/b/bitcoin.asp [↑](#footnote-ref-1)