An analysis of the decentralization and effectiveness of Lightning Network

L.W. Perik - l.w.perik@student.utwente.nl
University of Twente

***Abstract*—In 2021 alone, Bitcoin transactions added up to a total volume of $\$15.8 * 10^{12}$ USD, according to the Wall Street Journal. This was an increase of 567% with respect to 2020. However, smaller transactions can sometimes take several days before being confirmed and are subject to a significant fee. Bitcoin's standard implementation therefore leaves much room for improvement before it would be a viable option for micro-payments. In 2015, Lightning Network was introduced with the goal of solving Bitcoin's shortcomings. Enabling fast transactions and minimizing transaction fees, Lightning Network offers a peer-to-peer payment protocol layer on top of Bitcoin's blockchain. Using this implementation, peers can create transactions off the blockchain, resulting in a higher level of anonymity. Though various papers have been published suggesting the network is centralized, most research was conducted when the network was significantly smaller. In this paper, we answer the question of how centralized Lightning Network is in 2022. The contribution in this paper is unique in that nodes are indexed by size, and their properties analyzed to find specific trends. We identify how the usage of address protocols differs for nodes, including an analysis of the percentage of nodes that are cloud-hosted. Furthermore, we conduct a channel funding experiment to discover the usage patterns and characteristics, such as the minimum channel size, of nodes of different sizes. Our results suggest that Lightning Network is still fairly centralized, with large nodes having a substantial share in the connectivity and effectiveness of the network.**

Keywords: Blockchain, Bitcoin, Lightning Network

I. Introduction

To understand the usage and characteristics of Lightning Network, one must first understand why it was invented and what it's disadvantages are. To describe the trade-offs that are faced by developers incorporating blockchain technology, Vitalik Buterin, the co-founder of cryptocurrency Ethereum, coined the term *Blockchain Trilemma*. This concept states that a cryptocurrency must either make a compromise on decentralization, security or scalability of the blockchain. [1]

With Bitcoin being designed to be a decentralized and secure cryptocurrency due to the incorporation of Proof of Work, it only has a throughput of approximately 7 transactions per second, compromising the scalability aspect of the blockchain and hindering Bitcoin's adoption and possibility of becoming a global micro-payment provider. To solve this problem, Payment Channel Networks are often considered as a solution to Bitcoin's scalability issues.

Lightning Network is one such protocol in that it aims to solve scalability issues in Bitcoin's implementation by moving transactions off-chain. This means there is a reduction in the need for computation-heavy mining required for a large number of transactions, lowering the transaction fees and increasing throughput and transaction propagation.

With the white paper of Lightning Network introduced in 2015, the first Lightning transaction on top of the Bitcoin network was only made in 2019. Furthermore, most research on Lightning Network was done in 2019 and 2020, resulting in the need for new research into how the network has grown since then and what effect this had on the decentralization of the network.

In this paper, we will research the decentralization and effectiveness of Lightning Network as a payment-provider in 2022. This will be done by two experiments, one of which aims at finding out where nodes are located and how many of them are cloud-hosted. The second stage of this research will involve an experiment to determine node properties such as the minimum channel size, what client implementations nodes use and their responsiveness to connections.

II. Background

Bitcoin's blockchain is made up of blocks containing transactions. These blocks are included in the blockchain by solving a complex computational problem, which is called mining. As a result, the creation of blocks requires much energy and miners are compensated for each block they mine with a block reward. In addition, the miner gets a fee for every transaction it includes in the block it mines. This fee can be decided by the one publishing the transaction, and since many of the miners exist mainly with the goal of making profit, those miners will be more incentivized to include transactions with a higher fee in the block they are mining. A lower fee therefore results in the transaction on average taking longer to be included in the blockchain than transactions with a higher fee. [2]

Furthermore, each transaction has an input and an output. To create a valid transaction, one must unlock the *unspent transaction outputs* (*UTXO*'s) of another transaction by providing a correct digital signature in the input. After that, the funds in the unlocked outputs can be transferred again as part of the new transaction, which in it's turn must include in it's output a script that could be unlocked by another digital signature. This creates a chain of transactions, where every Bitcoin transaction, with exception of the *coinbase transaction* that includes the miner's block reward, should include a valid signature that unlocks Bitcoin's from another transaction. Besides having only the option to unlock funds using a signature, Bitcoin also allows for the creation of custom scripts that can specify certain conditions that need

to be met before the outputs of a transaction are allowed to be spent. It is the option of including such custom scripts that allow for the existence of layer 2 protocols that are built on top of Bitcoin's blockchain.

As mentioned previously, Lightning Network is one such layer 2 payment protocol with the aim of solving Bitcoin's throughput issue. With Lightning Network being a layer 2 protocol, this means it relies on Bitcoin's network for the opening and closing of channels. Most of Lightning Network's operation, however, happens outside of Bitcoin's network using a Lightning client that hooks into the Bitcoin Core client.

A. Bi-direction payment channels

Within Lightning Network, all funds are moved through payment channels. These bi-directional payment channels are created between two peers by either or both of them moving funds into a 2-of-2 multi-signature address on Bitcoin's blockchain, meaning a signature of both peers is required in order to unlock the funds in the channel [3]. Any further payments happen off-chain, where the channel serves as a kind of ledger or abacus that keeps track of how the funds are distributed between the two nodes, as shown in Figure 1.

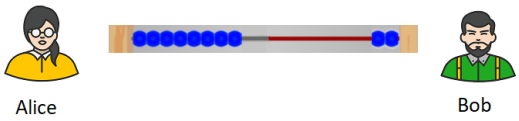

Fig. 1: Abacus analogy for moving funds through a Lightning Network payment channel [4]

For each off-chain transactions, the two nodes share a *commitment transaction* combined with a revocation key that can invalidate the previous commitment transaction if published. In case the closure of a channel is uncooperative, meaning a commitment transaction is published to the blockchain instead of a closing transaction containing the signature of both peers, there is a time delay for the peer publishing the commitment transaction before the outputs can be spent again. This time delay allows the other unresponsive peer to have some time to verify the transaction and make sure the published commitment transaction is the latest one. In case it is not the latest one and a peer tries to cheat by publishing an older commitment transaction, the other peer can use the revocation key to claim all funds in the channel. [3]

In practice, the majority of channel closures are undisputed with both parties signing and agreeing to the closing transaction. [5] This means the funds in the Bitcoin address can be spent again on the blockchain without any delay penalty for either party.

B. Payment routing and fees

Since opening a channel with another peer requires a fund, it is very costly to open a channel with every node one wants to transact with. Therefore, Lightning Network also offers the option to route payments through other nodes. This is done using *Hashed TimeLocked Contracts* (HTLC's). HTLC's use random numbers and their hash to ensure intermediate nodes follow through with properly routing the payment.

Figure 2 shows how Alice would be able to transfer 1 Bitcoin to Bob through an intermediate node using HTLC. First, Bob generates a random number and sends it's hashed version to Alice. After that, Alice sends a pending payment that is 'locked' to it's neighbouring intermediate node, in our case Carol. In the case of multiple intermediate nodes, these would do the exact same, creating a list of pending payments from one node to the other. Once the pending payment has arrived to Bob, he unlocks it using the random number he generated. Once he has done that, he hands over the key to his neighbour. This causes a kind of chain reaction, where each node in the chain can unlock the payment that was still locked, ultimately resulting in 1 Bitcoin being transferred from Alice to Bob. [3]

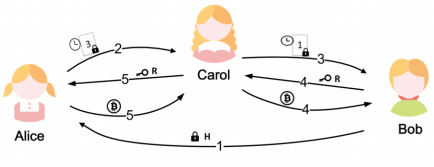

Fig. 2: Payment routing using HTLC [6]

Moreover, every node in the network may specify a fee that it collects when it routes a payment. When someone wants to send a payment to a specific node, it is the Lightning client's job to find an optimal path from source to destination with a minimal fee. As the routing of transactions using Lightning Network does not require any significant computational effort, the routing fees are on average very small as compared to Bitcoin's fees. The main expense for nodes that route Lightning payments, on the other hand, is the opening and closing of channels as these are included in the blockchain, therefore needing a relatively large fee to incentivise miners to include the transaction in a block. Even though the fees that a node can collect from routing payments are small, they can therefore still make money by gaining a favourable position in the network, where there is an equal amount of transactions in both directions. This reduces the risk of all funds being located at one side of the channel, resulting in the need for costly channel rebalancings that involve the opening and closing of new channels. However, this competition over gaining a large share in the network does come at the cost of increased centralization.

C. Protocol implementations

The Lightning Network protocol is described in a set of RFC documents called the Basics of Lightning Technology (BOLTs) [7]. Currently, there exist 3 major implementations of Lightning Network clients: Lightning Network Daemon (LND) [8], Core Lightning [9] and Eclair [10]. Furthermore, besides using standard IP, Lightning Network has the option to route packets using Tor. By using Tor, nodes prevent their IP address and thus their location from becoming public

knowledge. Consequently, the nodes that are solely using Tor require a proxy to connect to them, lowering their availability for nodes that don't use Tor.

III. Research objectives

By moving the transactions off-chain, Lightning Network compromises the decentralization of the network while increasing the scalability by offering a higher transaction propagation speed and allowing for more transactions per second. This would make Lightning Network ideal for future micropayments solutions, though further research in needed into how much the decentralization and effectiveness aspect are affected as a result of a better scalability. One of the disadvantages of Lightning Network that may make people decide to not use it is the initial barrier of entry to the network. In order to properly contribute to the network, one must not only operate a running Bitcoin instance (which at the time of writing takes up approximately 420GB of storage), but also have a Lightning client up and running 24/7 to prevent other nodes from abusing security features. Furthermore, since the opening and closing of a channel is costly, many nodes want to prevent this by requiring an initial deposit that is quite substantial, decreasing the need for channel closures and new openings.

In practice, this means one would usually need to dedicate a significant amount of money to get started contributing to the network. Furthermore, the existence of big hubs may result in a network that is heavily centralized. In this paper, we will answer the following questions:

1) How centralized is Lightning Network?: Lightning Network was made with the intention of having a network that allows for fast transactions, while compromising on the decentralization of the network. In this section we aim to find how centralized the network is and in what ways said centralization can be observed. We will show some common characteristics of the bigger nodes in the network, and estimate what percentage of nodes can only be reached through the largest nodes in the network. Furthermore, we will have a look at the reliance of nodes on specific cloud-providers.

2) How effective is Lightning Network as a payment processor?: As previously mentioned, the barrier to entry of the Lightning Network is quite substantial, with new users having to dedicate a significant amount of money to get started with participating in the network. As a part of this question we aim to see how favorable the network is to new nodes joining the network. More specifically, we will find the minimum channel size that nodes require and see how this differs for big nodes as compared to smaller ones. Moreover, we will identify the address protocols that are announced by nodes. This indicates, for instance, what part of the network can be reached without a proxy by nodes that have not setup Tor.

IV. Related work

In 2019, D. Šatcs showed in their paper [11] that the majority of the nodes in the network supported an IP protocol, making them publicly available. They also determined the address protocol usage of nodes in three categories: nodes with 1 channel, nodes with 5-10 channels and nodes with the highest number of channels. What they did not do, however, is to find how exactly the usage of address protocols differed for nodes of arbitrary sizes.

Among other things, F. Waugh and R. Holz [12] also researched the usage of address protocols in Lightning Network. Their contribution, however, only includes an analysis of the number of addresses that they could find for each protocol, including how these numbers changed over time.

Instead of grouping nodes into categories, we will sort nodes by their size and identify how the presence of different address protocols changes depending on node size. Furthermore, we will show that there are clear usage patterns for nodes of different sizes.

In paper [5], P. Zabka *et al.* classified the client implementation that nodes used based on a few properties they announce. However, for their analysis they use parameters that can be manually configured. In our contribution we identify the implementation that nodes use based on the error message that they return, something that cannot readily be altered. Moreover, in their paper they researched the geographical location of nodes by continent, plotting scattered data to identify the presence of Lightning nodes in metropolitan areas. They did not, however, identify whether the high densities could be the result of nodes that are cloud-hosted. In our contribution we feature a more thorough analysis of what countries nodes are located in, including what countries have a high number of cloud-hosted nodes.

Furthermore, as far as we are aware no research has been done to identify the minimum channel sizes that nodes of different sizes require. Our contribution is also unique in that we identify particular usage patterns for nodes of different sizes. Lastly, most of the above research was conducted around late 2019 or 2020. With Lightning Network having grown from approximately 2000 to more than 17.000 nodes since 2019 [13], among other things due to the growth and interest in Bitcoin in recent years, these results are likely to be out of date.

V. Methodology

For the conducted experiments, the client Core Lightning was used. This client includes a Python library, allowing access to Remote Procedure Calls (RPC's) and creation of plugins that alter it's capabilities. To grow the list of discovered nodes and increase the list of channels, a script was used that connected to the largest nodes in the network.

A. Address protocols

The goal of the first experiment was to determine what address protocols nodes in the network advertise and to compare this to previous research. Furthermore, we will identify the geographical locations of nodes and the number of nodes that are cloud-hosted.

All node information was found using the *listnodes* RPC call. To obtain the geographical location and usage types of nodes having an IP address, we used a database provided by

ip2location.com. Since the database came in the form of a *CSV* file, all data was first imported to a *SQLite* database that could be queried with an IP address.

B. Channel funding experiment

In order to route payments to other peers, one must first open a channel with another node. Since opening a channel requires a fund, the minimum channel size that nodes require has a big influence on the number of nodes that can be reached with a limited investment. To determine the minimum fund that nodes require, an experiment was done where we connected to nodes and then tried to fund a channel using a fund that was too small. As a result, either a connection error message was returned or the node sent back an error announcing it's minimum channel size.

The experiment was performed on a Virtual Machine running Ubuntu. Furthermore, we captured a snapshot of the network by saving the channel data of all known channels in a *JSON* file using the *listchannels* command. The script was left running until all nodes that were found in the list of channels had been sent a channel funding message.

Lastly, the amount that was used for the experiment was approximately 2000 satoshis. This is well under the default minimum channel size of the two most popular Lightning clients LND and Core Lightning, respectively 20.000 and 10.000 satoshis.

C. Visualisation

All graphs from the experiments were made using Python. The data that was collected was plotted by node index, based on the size of the node in the network.

To find a correlation between the size of nodes and the parameter being measured, graphs were plotted using a convolution with a uniform kernel. As a result, the convolution yields the walking average of the data, which makes it easier to interpret and spot trends for nodes of different sizes. In practice, however, there can exist a difference between the number of nodes that can be found using the *listchannels* command (used for indexing the nodes according to size), and nodes found using the *listnodes* call (usually containing the parameter that is being measured). This leaves gaps of unknown values in the data that need to be accounted for. Therefore, we used a *convolution* function that was able to handle unknown values by interpolating the values that are around it [14]. Hence, all data could be reconstructed even in the presence of unknown values.

Lastly, increasing the window size of the convolution logically results in the spikes being smoothened. In this way, the window size acts as a low pass filter for the data. The window size that was used is given in the title of the figures.

VI. Results

After having run the script for a day, the number of nodes that could be found using the *listchannels* RPC call was 15616, having in total 75777 number of channels. Of these 75777 channels, 65052 were announced by both nodes while the remaining were announced by only one node. On the other hand, 15641 nodes were found using the *listnodes* command.

The number of nodes found using the list of nodes is therefore around 89% of the nodes as discovered by Lightning Network search engine 1ml.com [15] (17.668 at the time of writing), meaning the results are sufficiently representative of the entire public network. The following experiments were conducted on the 15th of June 2022, and the results are therefore representative of the network around that time.

Node indexing: To find a correlation between the data, nodes were either indexed by their total node capacity or number of channels, where the node with index 0 is the largest node and the subsequent nodes are increasingly smaller. Furthermore, the total node capacity is defined as the total capacity in all channels it has, whether that fund came from the node itself or from it's peer. If a window is specified, the y-value at any point is equal to the average of the next number of nodes as specified by the window size.

A. Channel analysis

To get an idea of how centralized the network is, this section will analyze results that were obtained using properties of the network derived from the the list of nodes and channels. Figure 3 shows the amount of nodes that can only be reached through the largest nodes in the network. As you can see, approximately 30.5% of the network can only be transacted to by routing through one of the top 250 nodes (by total node capacity). If we order the nodes by the number of channels that they have, this number is even higher at 40.5%. This means that those nodes do not have a channel to any node outside of the 250 largest ones. In practice, transactions to a substantial amount of nodes will therefore rely on the connection to one of the big hubs in the network.

Furthermore, Figure 4 shows that the largest 250 nodes in the network (by total node capacity) are together responsible for approximately 65.5% of the capacity of all channels in the network (independent of which of the two nodes funded the channel). When the nodes are order by the number of channels, this number is slightly lower at 60%. This, however, intuitively makes sense since one would expect the largest nodes ordered by capacity to collectively have a higher capacity than the same number of nodes order by the number of channels that they have. Furthermore, the above results suggest that the larger hubs in the network are of significant importance when it comes to routing transactions to a significant number of nodes in the network.

B. Address protocols

Of the 15641 nodes that were announced to our node, 3307 (21.1%) had an IPv4 address. The majority of the nodes turned out to solely support Torv3, approximately 11246 (71.9%). Furthermore, only 82 nodes were found using Torv2. This value is very low as is to be expected since the Tor project started the deprecation of version 2 in September of 2020, with support for that version having stopped July of 2021 [16]. Moreover, the usage of IPv6 also turned out to be relatively low, with only 121 nodes (0.8%) having an IPv6 address. This

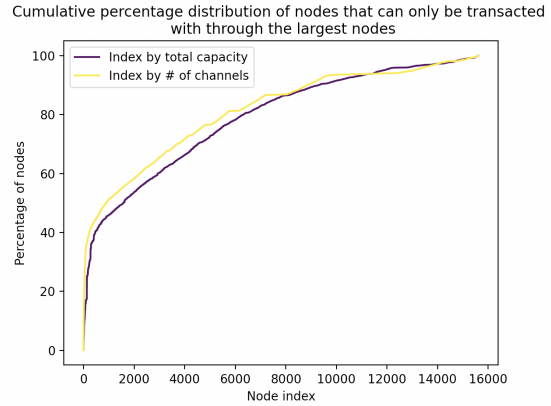

Fig. 3: A majority of nodes in the network can only be reached through one of the larger nodes

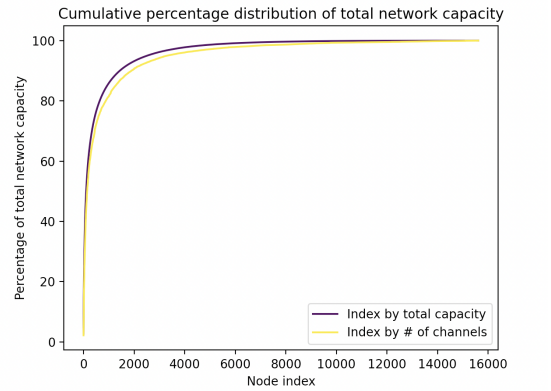

Fig. 4: Most of the capacity in the network belongs to channels with the largest nodes

is a substantial difference from the results in paper [11], where the author found that IP-usage was at worst 51% (for nodes with 5 to 10 channels). This suggests that a larger percentage of the network has started using only Tor, *e.g.* for the increase in anonymity.

Figure 5 shows the combinations of address protocols used by the nodes found using the *listnodes* RPC command. As you can see, bigger nodes in the network had a higher tendency to support a wider range of address protocols. This is likely due to the fact these nodes are used as hubs that want to make money from fees by routing as many transactions as possible, meaning it is advantageous to have different ways for nodes to connect to it. Approximately 73.3% of the largest 250 nodes had at least an IPv4 or IPv6 address. Furthermore, the chance that no address could be found was found to increase for smaller nodes. A relatively low percentage of nodes had a miscellaneous set of address protocols, for instance two IPv4 addresses, as indicated by *Other* in Figure 5.

Of the 3307 nodes that had announced their IP address, 2071 (62.7%) belonged to a data-center or web-hosting service (DCH). Furthermore, 1111 (33.6%) were found to belong to a mobile/fixed line internet service provider (ISP). The remaining 3.7% was either hosted at a content delivery network, commercial or educational facility. Figure 6 shows that

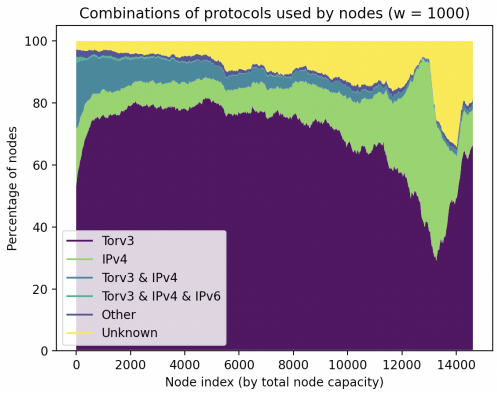

Fig. 5: Combinations of address protocols used by nodes in the network

the percentage of nodes that were cloud-hosted went from approximately 75% for the largest 1000 nodes to around 40% for smaller nodes. However, there is an increase of nodes that are cloud-hosted at around node index 12.000 to 14.000. Why the nodes in this region had a higher tendency to be cloud-hosted will be explained in the next section.

When it comes to what hosting providers are used for Lightning Network nodes, *Amazon* turned out to be particularly popular, with 627 nodes being hosted there. The second biggest cloud-provider was *LunaNode Hosting Inc.* with 244 nodes. LunaNode provides cloud-hosting for Lightning nodes [17], including the ability to host a BTCPay server [18], which is an open-source cryptocurrency payment processor. Most of the solutions that LunaNode offers require little to no setup. This could explain why LunaNode is particularly popular among small to medium nodes as shown in Figure 7, with usage slightly decreasing for larger nodes. Next on the list of used cloud-providers were SHRD Sarl (227), DigitalOcean (137), Hetzner (121), Three Fourteen Sasu (104), Google (73) and Contabo GmbH (64). Figure 8 shows that the majority of nodes that announced an IP address were located in the United States, Germany and Canada. Furthermore, we found that most nodes were located in North America (44.8%) and Europe (45.6%), with Asia (6.0%), South America (1.2%), Oceania (1.6%) and Africa (0.8%) only contributing little to the amount of nodes in the network. This data corresponds surprisingly well to the results found in a paper published in January of 2021 [5], with the percentage of nodes located in North America not having changed at all and that of Europe being 2.5% higher. Lastly, the percentage of nodes that are cloud-hosted being particularly high for Canada is caused by the fact that *LunaNode* is based there. [17]

C. Channel funding

In order to determine the minimum channel size of nodes, each node was sent a *fund channel* message. Of all 15616 nodes that we connected to, 6082 nodes returned an error. The chance that an error was returned after connecting to a node was therefore approximately 38.9%. Figure 9 shows if and what connection errors were returned upon funding of a

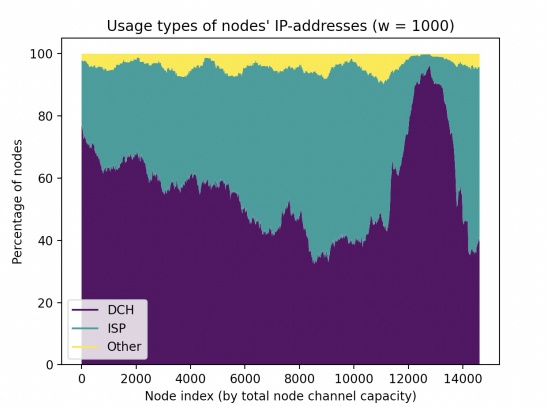

Fig. 6: Usage types of IP addresses of nodes using IPv4 or IPv6

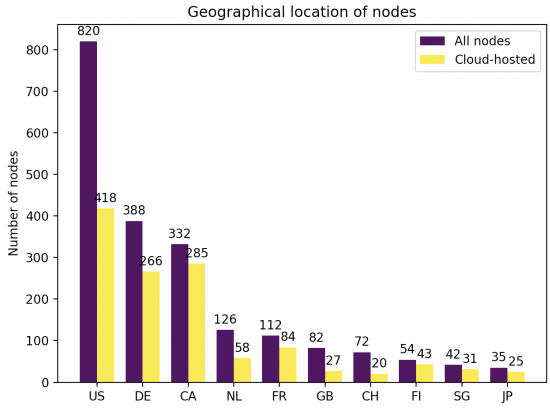

Fig. 8: Geographical location of nodes combined with number of cloud-hosted ones

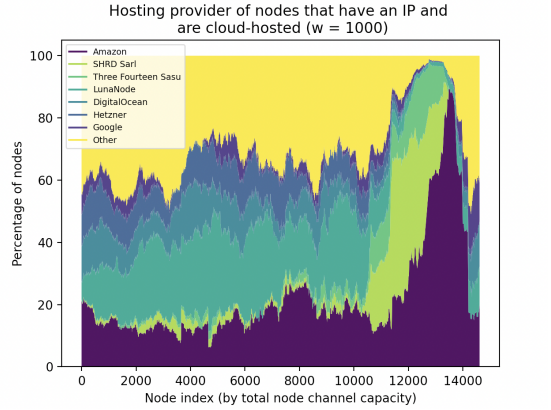

Fig. 7: Hosting providers used by Lightning nodes that announced an IP address and are cloud-hosted

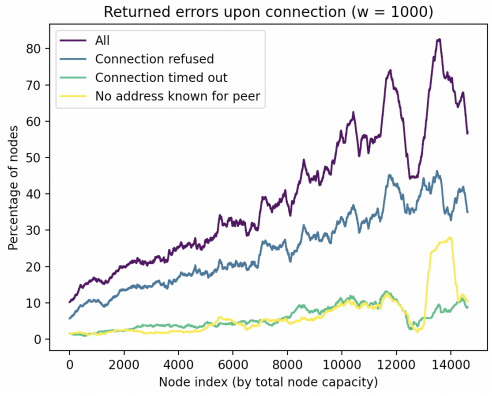

Fig. 9: Errors that were returned if the connection with a node failed

channel. As you can see, the chance that a node refused the connection went from around 5% for the largest 1000 nodes to approximately 40% for the smallest nodes in the network. Furthermore, smaller nodes had a higher chance of not having their address announced to our node, meaning no connection could be made to them. One might also observe that at around node index 14.000 there exist a large number of nodes of which the address was unknown, and that the *'No address known for peer'* line matches well with the peers that did not have any address as shown in Figure 5. Lastly, the results show that smaller nodes tend to be less responsive to a connection attempt, *i.e.* have a higher percentage of connection attempts that timed out.

Figure 10 shows the resulting minimum channel sizes that the nodes returned, indexed by both the capacity of the nodes and the total number of channels the nodes had (and thus approximately the number of nodes that it can route transactions to). Where the minimum channel size of small to medium nodes (index 4.000 to 14.000) was approximately 41.000 satoshis, the top 250 nodes required on average at least 0.018 BTC ($\$361,18$ USD as of June 2022), which is approximately 44 times as much. As you can see, our results suggest that bigger nodes tend to require on average a higher minimum channel size. Seeing as 35% of nodes only have a channel to one of the top 250 nodes as shown previously, nodes with a limited investment that only have a channel to one of the smaller nodes may therefore require more intermediate nodes to route a transaction to another node in the network. Consequently, the chance that no route can be found to those nodes, *e.g.* due to any channel along the route not being properly balanced, increases.

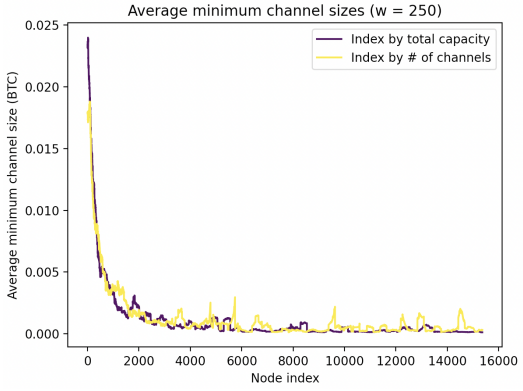

Fig. 10: Minimum channel sizes that nodes announced

Since the message that is returned when the channel fund

capacity is too low is specific to the used client, it can be used to identify the share of each client in the network. Of all 9534 nodes that had responded to our message with an error indicating the fund was too low, 95.2% used LND, while 4.6 used Core Lightning and 0.2% Eclair. The usage of these clients differed slightly for larger nodes compared to smaller nodes as can be seen in figure 11. This is likely due to the fact Core Lightning is very customizable and has better performance than the other implementations due to being implemented in the low-level programming language C. LND, on the other hand, is more widely supported and has a wider variety of user interfaces one can use with it, making it a better alternative for beginners [19]. Furthermore, the popularity of Core Lightning was higher in previous years (approximately 11% of clients, as shown in [5]) and has therefore declined, meaning newer nodes likely have a higher chance of using LND while older nodes that have dedicated more funds towards Lightning Network and have been in the network for longer have a relatively higher use of Core Lightning.

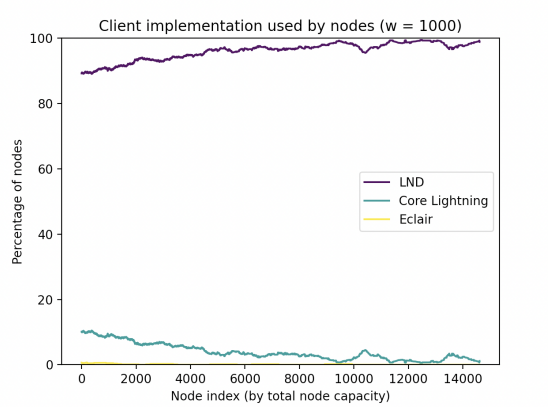

Fig. 11: Used clients for all nodes indexed by total node capacity

D. Layer 3 protocols

One feature that is persistent among many the above figures and has not yet been accounted for is the anomaly at around node index 12.000 to 14.000. The nodes in this region were found to be characterized by on average a lower response time, a minimum channel capacity of approximately 100.000 satoshis and high usage of IPv4, almost all of which were cloud-hosted as shown in Figure 6. After further investigation these nodes seemed to be hosted primarily at three different hosting-providers: SHRD Sarl, Three Fourteen Sasu and Amazon. Furthermore, we found that most of the nodes that were hosted at SHRD Sarl and Three Fourteen Sasu had an alias name with 'nodl' in it. *Nodl* is a company that provides cloud-hosting solutions for Lightning nodes [20], and after having contacted the owner of these nodes, they all seemed to belong to a layer 3 protocol called *Sphinx Chat* that uses Lightning Network for sending encrypted messages. [21]

The presence of the nodes that belong to *Sphinx Chat* can explain a few of our observations. Firstly, most *nodl* nodes turned out to only have one channel with 100.000 satoshis in it, while for the ones hosted at Amazon this amount was anywhere between 70.000 and 100.000, hence the Amazon peak occurring at a slightly lower node index in Figure 7. Moreover, when looking at the error percentages for nodes in Figure 9, we can see a dip at node index 12.000 followed by a peak at approximately 14.000. This feature can be attributed to the fact that the *nodl Sphinx Chat* nodes all accepted our connection, while the ones hosted at Amazon all refused any connection attempt. Another artifact of these *Sphinx Chat* nodes can be seen in Figure 11, again at around node index 12.000. Almost all nodes in this region were found to use LND, which can be explained by the fact that LND is the default client implementation used by *nodl* [20].

Lastly, using an analysis of the increase in IP-usage and cloud-hosting, we managed to identify at least 739 nodes that belong to *Sphinx Chat*'s infrastructure.

VII. Discussion

In our results, we found that Lightning Network is very prone to centralization, with 35% of nodes in the network only having a channel to the largest 250 nodes. This increases the reliance of the network on those nodes for routing transactions, decreasing the decentralization. We have also shown that the largest nodes in the network shared many common characteristics. Not only were the largest nodes found to have a higher chance of supporting multiple address protocols, they also had an increased chance of being cloud-hosted. Moreover, our results suggest that larger nodes are more responsive to connections and that they require a minimum channel size that is on average significantly higher as compared to smaller nodes.

Consequently, new nodes joining the network are met with the decision whether or not the aforementioned advantages are worth the increase in initial investment required to open a channel to the larger nodes in the network. Also taking into account the little maintenance that having only one channel has, we consider these large hubs to be a compelling option for nodes that want to start using the network. However, in the case of many nodes opting to go this route, this does come at the price of even more reliance on the larger nodes, further increasing centralization within the network.

For our experiment, we showed that we were able to find 15641 nodes using the RPC call *listnodes*, which was 89% of the nodes as reported by Lightning search engine *1ml* [15]. We therefore consider the results to be sufficiently representative of the entire public network. Furthermore, two figures were shown that indicate a significant amount of nodes in the network are cloud-hosted. Important to note, however, is that this data is only representative of a minority (21.1%) of the network that announced an IP address. This means that in some cases only 15 to 20% of nodes could be used to determine the percentage of nodes that was cloud-hosted in a specific range. To invalidate our claim of the network being very reliant on hosting provided by commercial companies, one could therefore argue that people using Tor may have a higher chance of having hosted their nodes at home, since doing so would prevent their IP, and thus geographical location,

from becoming public knowledge. However, given the high IP-usage of the largest nodes in the network (around 73.3% for the largest 250 nodes) and their share in the effectiveness of the network, we do consider our results to be accurate enough to support the claim that the network is very dependent on services provided by commercial companies. Consequently, this negatively impacts the decentralization of the network.

Moreover, we managed to identify nodes belonging to a layer 3 application built on top of Lightning Network called *Sphinx Chat*, using some common characteristics that the nodes shared. The presence of an application like *Sphinx Chat* raises the question what other layer 3 implementations might have been built on top of Lightning Network and what part of the network is used with a purpose other than transacting with peers. Additionally, with one of Lightning Network's aims being to increase scalability, it is likely also prone to companies that offer non-custodial wallet services. How many of these non-custodial wallets are already present in Lightning Network and what effect this has on the centralization in the network is something that we suggest be further researched.

Furthermore, most research into Lightning Network was done around 2019. In the years since it's inception, the network has grown substantially. This calls for new research into how, for instance, the maximum payment amount that can be routed through Lightning Network has changed over time.

Lastly, we want to mention that the results obtained in this paper do not necessarily indicate Lightning Network is failing it's goal of becoming a suitable option as a large-scale micro-payment processor [3]. On the contrary, the opportunity that Lightning Network presents does not only come from the fact that it offers fast propagation of transactions, but also from the freedom that users can choose themselves whether to compromise on decentralization or on scalability. In this way, Lightning Network does not decrease the need for Bitcoin transactions, but instead complements Bitcoin's main network in becoming a network suitable for micro-payments, where users can decide on what to use depending on their needs. The results are, however, indicative of a network that is prone to centralization with a significant barrier of entry for new nodes joining the network, though further research should point out what implications centralization has on the security of the network.

VIII. Conclusion

In conclusion, the results in this paper suggest that Lightning Network is fairly centralized with a large part of the network's total capacity existing in channels that are controlled by the biggest nodes in the network. More specifically, approximately 35% of the network was found to only have a channel to one of the largest 250 nodes. Additionally, the minimum channel size that the largest 250 nodes required was on average 44 times higher compared to smaller nodes in the network. This means that new nodes joining the network often times need a significant fund in order to reach a large part of the network. As for the address protocols that nodes use, the majority of nodes were found to only allow connections using Tor. This is a substantial difference from previous research conducted in 2019 that pointed out that a significant portion of nodes were available with an IP address [11] [12]. Of the nodes that announced an IP address, we found that approximately 62.7% was hosted at a data-center or web-hosting provider. The majority of these nodes were hosted at Amazon (30.2%) and LunaNode (11.8%). This shows that a large part of the network is reliant on large companies that offer cloud-hosting solutions, lowering the decentralization of the network. Furthermore, we showed that there exist a big difference in responsiveness to connections between nodes of different sizes. Compared to the largest nodes in the network, the smallest nodes were approximately 8 times more likely to refuse the connection. Moreover, using an analysis of nodes' minimum channel capacities, IP-usage and channel information, we managed to identify an implementation built on top of Lightning Network called Sphinx chat.

The results we obtained in this work are therefore indicative of a network that is prone to centralization, with a large part of the network relying on a relatively small percentage of nodes. Furthermore, we found that there exist a significant barrier of entry for new nodes joining the network, lowering the effectiveness of Lightning Network as a large-scale micro-payment solution. Lastly, further research should point out what exact implications centralization has on the security of the network.

References

[1] M. S. Abdelatif Hafid, Abdelhakim Senhaji Hafid, "Scaling blockchains: A comprehensive survey." [Online]. Available: https://www.researchgate.net/publication/342639281_Scaling_Blockchains_A_Comprehensive_Survey

[2] J. K. Shoji Kasahara, "Effect of bitcoin fee on transaction-confirmation process." [Online]. Available: https://arxiv.org/abs/1604.00103

[3] The bitcoin lightning network: Scalable off-chain instant payments. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[4] R. Sheinfeld, "Understanding lightning network using an abacus." [Online]. Available: https://medium.com/breez-technology/understanding-lightning-network-using-an-abacus-daad8dc4cf4b

[5] S. S. C. Philipp Zabka, Klaus-T. Foerster, "Empirical evaluation of nodes and channels of the lightning network." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119222000323

[6] Y. Guo, "A measurement study of bitcoin lightning network." [Online]. Available: https://ieeexplore.ieee.org/document/8946216

[7] Lightning network bolts. [Online]. Available: https://github.com/lightning/bolts

[8] Lightning network daemon (lnd). [Online]. Available: https://github.com/lightningnetwork/lnd

[9] Core lightning (previously c-lightning). [Online]. Available: https://github.com/ElementsProject/lightning

[10] Eclair. [Online]. Available: https://github.com/ACINQ/eclair

[11] Understanding the lightning network capability to route payments. [Online]. Available: https://essay.utwente.nl/82015/1/Satcs_BA_EEMCS.pdf

[12] R. H. Finnegan Waugh, "An empirical study of availability and reliability properties of the bitcoin lightning network." [Online]. Available: https://arxiv.org/abs/2006.14358

[13] Bitcoin visuals. [Online]. Available: https://bitcoinvisuals.com/lightning

[14] Astropy convolution. [Online]. Available: https://docs.astropy.org/en/stable/convolution/index.html

[15] 1ml. [Online]. Available: https://1ml.com

[16] V2 onion services deprecation. [Online]. Available: https://support.torproject.org/onionservices/v2-deprecation/

[17] Lunanode. [Online]. Available: https://www.lunanode.com

[18] Btcpayserver. [Online]. Available: https://btcpayserver.org

[19] Namcios. What implementation of bitcoin's lightning network should you pick? [Online]. Available: https://bitcoinmagazine.com/technical/tradeoffs-of-bitcoin-lightning-implementations

[20] Nodl. [Online]. Available: https://www.nodl.it
[21] Sphinx chat. [Online]. Available: https://sphinx.chat