Pervasive and Mobile Computing 83 (2022) 101584

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Empirical evaluation of nodes and channels of the lightning network

Philipp Zabka [a,*], Klaus-T. Foerster [c], Stefan Schmid [a,d], Christian Decker [b]

[a] Faculty of Computer Science, University of Vienna, Austria
[b] Blockstream, Zurich, Switzerland
[c] TU Dortmund, Dortmund, Germany
[d] TU Berlin, Berlin, Germany

ARTICLE INFO

Article history:
Received 5 April 2021
Received in revised form 17 October 2021
Accepted 15 March 2022
Available online 23 March 2022

Keywords:
Lightning network
Classification
Geographical analysis

ABSTRACT

Off-chain networks provide an attractive solution to the scalability challenges faced by cryptocurrencies such as Bitcoin. While first interesting networks are emerging, we currently have relatively limited insights into the structure and distribution of these networks. Such knowledge, however, is useful, when reasoning about possible performance improvements or the security of the network. For example, information about the different node types and implementations in the network can help when planning the distribution of critical software updates.

This paper reports on a large measurement study of Lightning, a leading off-chain network, considering recorded network messages over a period of more than two years. In particular, we present an approach to classify the node types (LND, C-Lightning and Eclair) in the network, and find that we can determine the implementation of 99.9% of nodes correctly in our data set. We then report on geographical aspects of the Lightning Network, showing that proximity is less relevant, and that the Lightning Network is particularly predominant in metropolitan areas. Furthermore, we address various aspects of channels in the Lightning Network combined with the data we classified. We also demonstrate that channel endpoints behave very fairly and rarely cheat, that the same channel endpoints tend not to reconnect after the channel connection has closed and that there are more inactive than active channels in the Lightning Network.

As a contribution to the research community, we will release our experimental data together with this paper.

1. Introduction

Blockchain technology enables mistrusting entities to cooperate in the absence of a trusted third party. The technology also forms the basis of cryptocurrencies such as Bitcoin or Ethereum. A main challenge faced by blockchains however regards their scalability: the usual literature example is that while custodian payment systems easily support thousands of transactions per second, blockchains currently merely support tens of transactions per second [1].

By allowing users to make payments directly, without global consensus protocols and without having to commit transactions on the blockchain, emerging off-chain networks (also known as payment channel networks or second-layer blockchain networks) [2] can greatly improve the scalability of cryptocurrency payment systems. Indeed, over the last

* Correspondence to: Waehringer Str. 29, Office 5.46, A-1090 Vienna, Austria.
E-mail address: philipp.zabka@univie.ac.at (P. Zabka).

https://doi.org/10.1016/j.pmcj.2022.101584

years, off-chain networks such as Bitcoin Lightning [3], Ethereum Raiden [4], and XRP Ripple [5], to just name a few, have received great interest.

As off-chain networks become more popular, the requirements on their performance and dependability increase as well. However, how to efficiently meet these requirements is still subject to ongoing research, and more critically, researchers often lack empirical insights into the currently deployed networks: the publicly available data on these networks is severely limited.

This paper reports on a major measurement study of Lightning, a most popular cryptocurrency network today. In a nutshell, in Lightning, nodes typically represent users (running different Lightning clients, e.g., LND, C-Lightning or Eclair) and edges represent funds that can be transacted between the endpoints of the edge. In order to improve scalability, Lightning supports multi-hop routing of transactions, and in incentivizes the intermediaries to contribute to the transaction routing through fee-based mechanism. To this end, Lightning relies on source routing and in order to support nodes in finding "cheap" routes, i.e., routes with minimal fees, Lightning provides route discovery and gossiping mechanisms.

We recorded network messages (e.g., generated by the gossiping mechanism) in Lightning over a period of almost two years. Based on this data, we contribute insights to three main areas, one related to the security of these networks, one related to the performance and one related to the performance of channels in particular:

1. *Node classification:* It can be very useful to know the frequency and distribution of the different clients in an off-chain network. Such knowledge can also be relevant for security considerations, e.g., when planning the deployment of security patches.
2. *Geographic distribution:* It is generally interesting to know the topological structure of geographically distributed off-chain networks. In addition to general considerations (e.g., related to economic or sociological aspects), the geographic distribution may also be relevant for the performance and dependability of these networks: network topologies with local biases may improve performance, but may be less robust.
3. *Channel analysis:* Channels are one of the key features of the Lightning Network, as they allow almost instant transactions. Performing an empirical analysis of them can lead to helpful and deeper insights in matters such as channel distribution, channel balances, channel activity etc.

1.1. Our contributions

This paper presents an empirical evaluation of a large data set that we collected about the communication (and gossiping) occurring in Lightning. In particular, we present an approach and classification of the node types in the Lightning Network and also provide empirical insights into the geographical distribution of the nodes. We find that many users stick with the default settings of nodes and channels of the respective implementation. Our method allows to infer the implementation of 99.9% of nodes in our data set we can hence observe that one implementation is particularly predominant in the network. Furthermore we try to elaborate on reasons why this is the case. We also find that payment channels only come with moderate geographic bias and that the Lightning Network is particularly predominant in metropolitan areas. We can also see that the implementations are similarly distributed across most countries. Moreover we can observe that there might be a correlation between channel connections and countries which share a common trait. We detect that LND–LND is the most prominent endpoint pair, examine the lifetime and balance distribution of channels and prove that these two aspects do not stand in dependence. Furthermore, we observe that endpoints seldom cheat and that reconnections are also rare in their occurrences. Finally, we also look into the distribution of active and inactive channels (channels that have been created, but were never used).

Lastly, as a contribution to the community and in order to facilitate reproducibility and future research in the area, we will make our data set public with the published version of this article.

1.2. Organization

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries and Section 3 describes the node classification, followed by the geographical analysis in Section 4. We then report on our empirical study of channel properties in Section 5. Lastly, We review related work in Section 6 and conclude in Section 7.

2. Preliminaries

We now introduce some of the basics of the Lightning Network and specific preliminaries for the remainder of this paper.

Clients. The Lightning Network can be accessed via three main implementations or clients: C-Lightning [6], written in C++, LND [7], written in Go, and Eclair [8], written in Scala. These clients have various features, but their fundamental purpose is to create nodes and channels with other participants and act as a ledger.

The Lightning Network. The Lightning Network consists of a collection of nodes and channels. Nodes can create bidirectional connections, called channels, with other nodes which can be then used to send payments almost instantly

back and forth between the two participants. The network operates on the blockchain, but unlike Bitcoin, not each payment has to be published onto the blockchain itself, but only the first transaction, known as funding transaction, to fund a channel, and the last transaction, known as closing transaction, to close a channel and end the connection. Between these two transactions users can send an unlimited amount of transactions to each other, as long as they have enough liquidity. Although only a pair of nodes can create a channel, payments can be routed via multiple hops through the network to a receiver node, which is not necessarily directly connected with the sending node. Nodes helping in forwarding payments through their channels will usually collect a small fee for this service.

Gossip Messages. To utilize a path of more than one channel as payment route, nodes have to be aware of the network topology, in order to know which channels can be used to route the payment to the final receiver. For this purpose gossip messages are propagated from one node to another, to either announce a newly created node, channel or an update of both. In the following we introduce the three most important types of gossip messages for our work and some of their contained information, which are specified in the Basics of the Lightning Technology (BOLT) [3]:

- `node_announcement message`: This message is propagated either when a node has been created and is now ready or it updates its information. The message contains important parameters which enable other participants in the network to start channels with the specific node. For example, when a node wants to connect to another node, the `node_id` is needed for identification. Other for us relevant parameters contained in this message are `alias`, a nickname for a node encoded in UTF-8, `color` encoded in hexadecimal and the `addresses` parameter, which can contain IPv4 or IPv6 addresses as well as Onion v2 or Onion v3 service addresses. The `alias` and `color` parameters have no importance in terms of the operation to the node and their purpose is mainly to differentiate one node from another e.g. in visualizations.
- `channel_announcement message`: Always when a new channel between two nodes is created a `channel_announcement message` is sent. This gossip message contains information regarding the newly created channel and is propagated exactly once in the network. Similar to the `node_id` each channel has an unique `short_channel_id` for identification. Furthermore, the message contains amongst other parameters the `node_id` of the two nodes connected by the channel.
- `channel_update message`: A channel is not practically usable until at least one side has announced its fees and expiry for the Hashed Time Locked Contract (HTLC) of the payment. A HTLC is a security measure to ensure that nodes along the routed path do not steal the payment. This gossip message is propagated at least once from each of the participating nodes, since the initial routing fee may differ depending on direction the payment is coming from, i.e. from node A to node B or from B to A. Also every time a side decides to change its channel parameters a `channel_update` message needs to be propagated again through the network. Further relevant parameters are `short_channel_id`, the `channel_flag` indicating the direction the channel update is coming from and then four parameters describing important channel settings, namely `cltv_expiry_delta, htlc_minimum_msat, fee_base` and `fee_proportional_millionths`.

2.1. Data set

We collected our data set with the help of C-Lightning nodes deployed in the Lightning Network. These nodes synchronize their view of the network topology by exchanging gossip messages. More in detail, the nodes will deduplicate messages, discard outdated `node_announcement messages` and `channel_update messages` and then apply them to their internal view. In the case of node failure or restart, nodes write the raw messages to a file called the `gossip_store`. In order to restrain the file size, the nodes will once in a while rewrite the file, skipping messages that have been superceded in the meantime. Our unique data set is comprised of the three types of gossip messages introduced in the previous section, which were propagated through the network from March 2018 to January 2020. In this time span, we recorded more than 400 000 `node_– announcement messages`, more than 1 000 000 `channel_announcement messages`, and over 6 400 000 `channel_update messages`. A first analysis shows that the real growth of the Lightning Network started in 2018, which is also the year where LND and C-Lightning released their first major update for their clients.

3. Analysis of nodes in the lightning network

This section reports our main results from the node classification. The Lightning Network is currently comprised of three implementations: LND, C-Lightning and Eclair, each written in a different programming language and each using slightly different values for its parameters. Some of these values are public and can be obtained through message inspection of the gossip messages, others in turn are kept private for network security reasons.

One of these private parameters is `max_concurrent_htlcs`, which denotes the maximum capacity of HTLCS for a channel and which can play an important role in attacks on the networks topology: an attacker may want to determine how many HTLCS will be necessary to overload a channel, and hence make the channel unusable until the HTLCS resolve. These attacks can be targeted on the whole network or just affect a single node. For this reason precisely inferring a node's implementation to deduce the values for these parameters is key. Furthermore we want to analyze how the implementations are distributed in the network.

Table 1
Default parameters table.

Default Parameters			
Parameter name	LND	C-Lightning	Eclair
alias	–	Predefined	–
color	#3399ff	Derived from node_id	#49daaa
cltv_expiry_delta	40 (144)	14	144
htlc_minimum_msat	1000	1000	1
fee_proportional- _millionths	1	10	100
fee_base_msat	1000	1000	1000

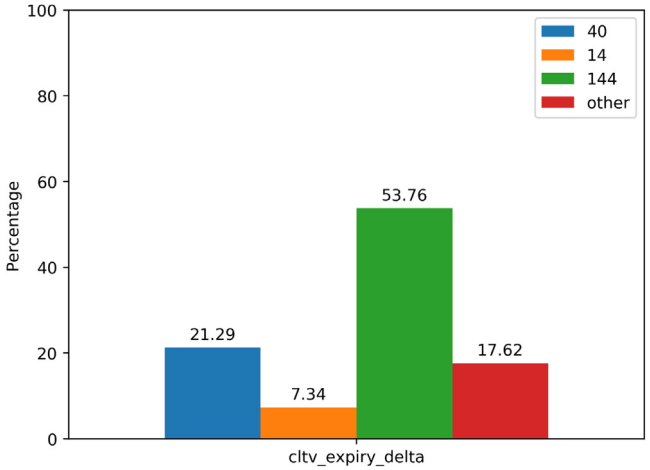

Fig. 1. cltv_expiry_delta distribution.

3.1. Analyzing default parameters

We now take a closer look at the parameters of interest.

Table 1 shows the default values for the three available implementations of the Lightning Network:

alias. The `alias` parameter for LND and Eclair does not have a default value and has to be manually configured when a new node is created.

C-Lightning uses NSA-Style names, which are created from an adjective and a noun, both originating from predefined lists in the C-Lightning source code [9].

The BOLT [10] documentation gives us two example names: 'IRATEMONK' and 'WISTFULTOLL'.

color. Both LND and Eclair have a predefined `color` parameter, which is automatically set up with each node creation.

For C-Lightning the node's `color` is automatically derived from the three first bytes of its `node_id`.

Next, we analyze the distribution of the introduced parameters in the network.

Other works such as [11–14] have already performed some analysis via snapshots taken with the help of LND's `describe graph` command.

However, as our data set covers gossip messages of almost two years, going vastly beyond single snapshots, we are able to obtain fairly precise insights on the parameter distributions.

ctlv_expiry_delta. We now examine the parameters in the `channel_update message` and start with the examination of the `ctlv_expiry_delta` parameter, which denotes the minimum difference in HTLC timeouts a node that is forwarding a payment will accept.

Fig. 1 shows us that 144, the old LND and current Eclair value, is represented in the data by 53.8%.

The new value for LND, 40, is represented by a share of 21.3%.

The value of 14 corresponding to the C-Lightning implementation takes a share of 7.3% in the overall data.

Overall 82.4% of the `cltv_expiry_delta` parameters use a default value.

We have also taken a closer look on how the distribution of 144 and 40 has developed over time, as LND changed its `cltv_expiry_delta` parameter from 144 to 40 in March 2019 [15], to get an first impression of how long it roughly takes for an update to be adapted by the majority of the nodes. Fig. 2 shows us that roughly two months after the change only 0.44% of the `cltv_expiry_delta` parameters in the network used the value of 40 until May 2019. However we observe that after the initial stagnation, usage of this value started to continuously increase, whereas the value of 144 started to continuously decrease. The new value of 40 surpassed the old value of 144 approximately mid December, from which we deduce that it took around 9 months for the update to reach the majority of LND nodes in the network.

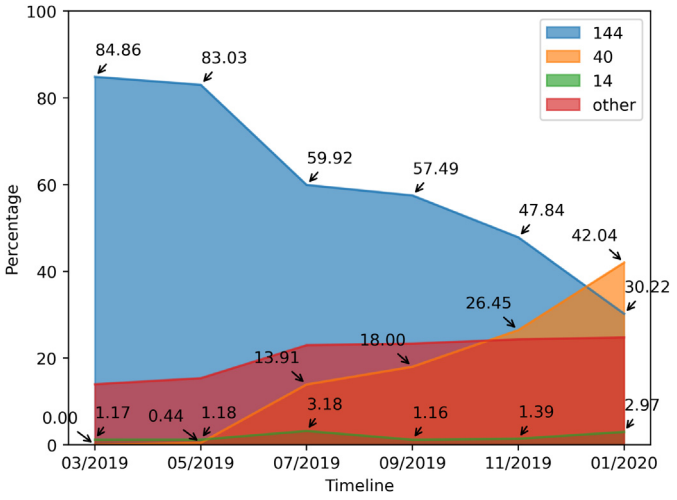

Fig. 2. Parameter distribution timeline.

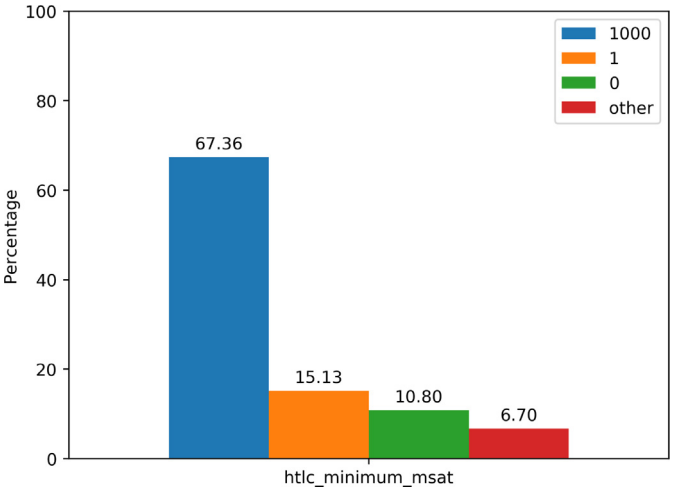

Fig. 3. htlc_minimum_msat distribution.

htlc_minimum_msat. Fig. 3 depicts the value distribution for the `htlc_minimum_msat` parameter, which denotes the minimum value in millisatoshi for a payment to be transferred of the channel.

LND's and C-Lightning's default value of 1000msat has a share of 67.4% and Eclair's value 1 has a share 15.1% in the data set.

Interestingly, in 10.8% of the updates the value 0 has been used, indicating no minimum value for payment transportation.

fee_proportional_millionths. Next, we studied the distribution of the `fee_proportional_millionths` parameter, which is the amount nodes will charge for each transferred satoshi over their channel.

The value 1 for LND can be found in 63.7% of the data, the value 10 used by C-Lightning is represented in 6.9% of the updates.

Lastly, Eclair's value of 100 can be found in 2.4% of the data.

Since this is the first parameter, which is different in all of the three implementations, Fig. 4 gives us a first insight in a possible distribution of the implementations in the network.

fee_base_msat. The `fee_base_msat` parameter depicted in Fig. 5, denotes the constant fee a node will charge for a transfer.

As the parameter is the same in all three implementations, it does not provide much insight in the distribution of the individual implementations, but interestingly it has the smallest overall share in the network, when summing up all the default values for the other parameters.

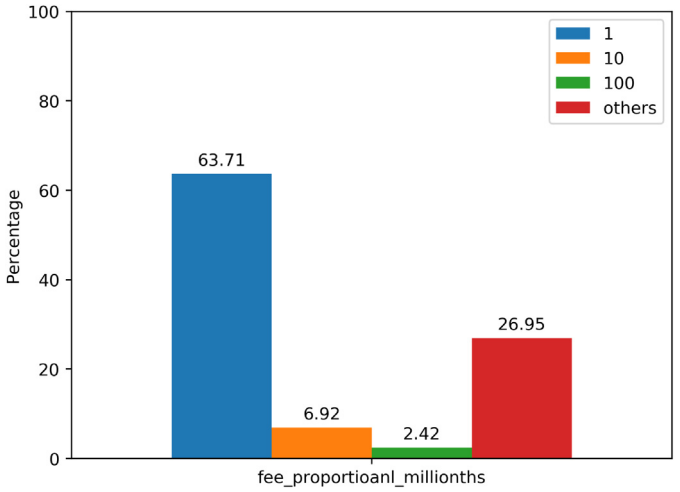

Fig. 4. fee_proportional_millionths distribution.

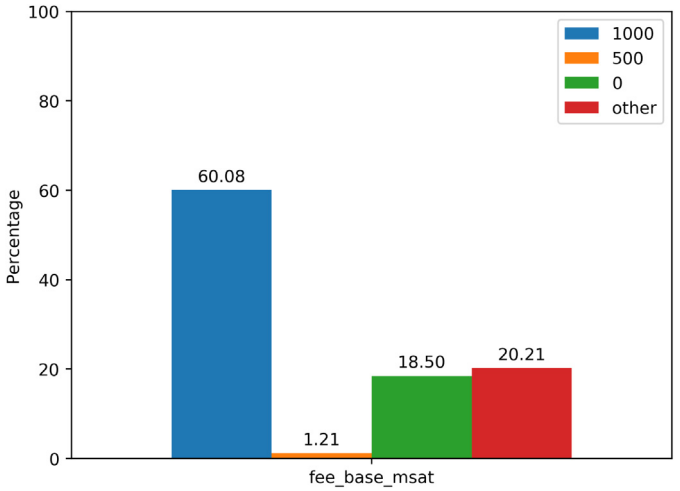

Fig. 5. fee_base_msat distribution.

`cltv_expiry_delta` and `htlc_minimum_msat` have a share of over 82% (82.4% and 82.5%) and `fee_proportional_millionths` has a share of 73%.

Further we can observe that there is a discrepancy between the individual parameters of some of the implementations.

For example, for Eclair's default parameter: 15.1% of the `channel_update messages` use 1 as an value for `htlc_minimum_msat`, but only 2.4% of the nodes use 100 the default parameter for `fee_proportional_millionths`.

C-Lightning on the other hand shows more consistency, `cltv_expiry_delta` and `fee_proportional_millionths` are with 7.3% respectively with 6.9% more similarly distributed.

3.2. Data preparation

In a time span of over two years a node can update itself and the values of its channels multiple times.

Some of them even flood the network with over 1000 `channel_update messages` a day.

Our data set is comprised of over 6.4 million of these `channel_update messages` and over 40000 `node_announcement messages`.

In order to obtain the most accurate results from the classification of the nodes we have to process our data accordingly.

For this purpose we select two methods to process our data, a pre-processing and a post-processing one.

We next introduce both methods:

- **Most Frequent Values (Pre-processing)**: From all the values a node has used in the recorded time, we analyzed which values were used the most in each message to obtain the most representative data for our classification.

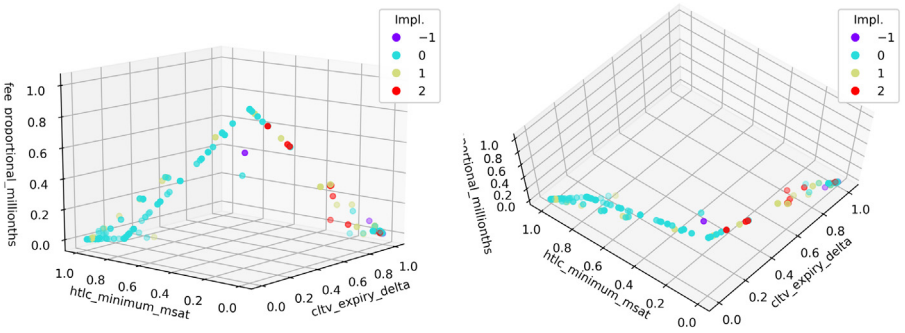

Fig. 6. Labeled nodes after pre-processing: -1: unclassified, 0: LND, 1: C-Lightning, 2: Eclair. Darker points indicate a higher node density.

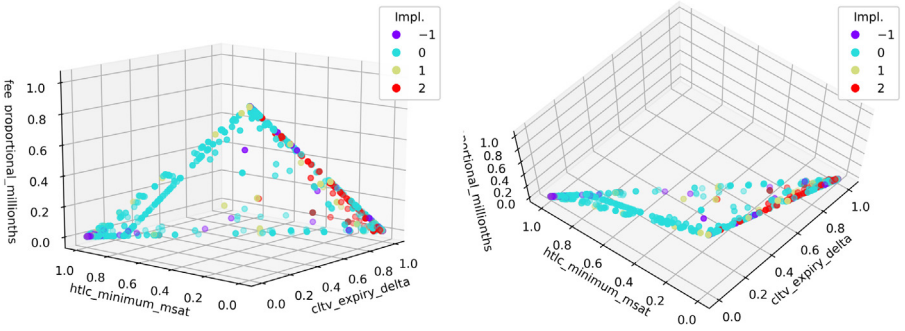

Fig. 7. Labeled virtual nodes: -1: unclassified, 0: LND, 1: C-Lightning, 2: Eclair. Darker points indicate a higher node density.

Fig. 6 shows the classified nodes after using the pre-processing method. The parameters were normalized for plotting purposes only.

- **Most Frequent Classification (Post-processing)**: Other than in the previous method, where we first sum up all messages from a single node and then choose the most frequently used values, here we treat every message as a virtual node and classify every single message on its own. After all virtual nodes have been classified, we choose the classification with the highest confidence for a node. Fig. 7 depicts the classification results of every virtual node after the post-processing then determines the implementation with the highest confidence for each real node.

Both methods exclude each other, which means that only one can be applied at a time. The first method produces one single input and consequently one single labeled output for a node, whereas the in second method produces a list of inputs and the classification outputs the same amount of labeled data, from which the classification with the highest probability is taken as the final result.

3.3. Implementation

Our analysis shows that because of how the data is structured, it is of little avail to classify the nodes with either a machine learning algorithm or a clustering approach.

Our classification algorithm can be seen as an extension from [14]. The algorithm takes six parameters as input, namely the `node_id`, `alias`, `color`, `cltv_expiry_delta`, `htlc_minimum_msat` and lastly `fee_proportional- _millionths`.

We chose not to include the `fee_base_msat` parameter, as it is the same for all three implementations; however one could use it as an additional check, as it was done in [13]. The `node_id` is, as mentioned earlier, needed for the derivation of the C-Lightning `color` parameter. Each parameter is inspected individually to infer its implementation based on the default values. The result of each classification operation of each parameter is a vector with a "1" entry at a certain index. However, if all entries are "0" no implementation has been found. A vector consists of three indices, each index representing a certain implementation. The result of this process is a 5×3 matrix, where each result vector represents a matrix row. We then sum up each row creating a 1×3 matrix, where again each index represents a implementation. Afterwards, the matrix is normalized to get the confidence of each implementation and finally determine the final implementation.

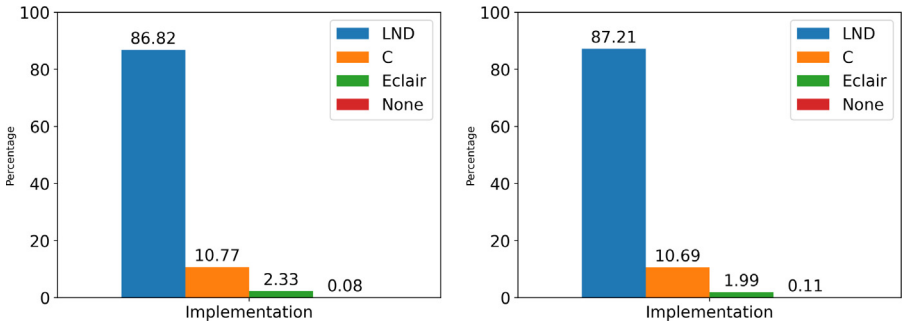

Fig. 8. Labeling results: Pre-processing and Post-processing.

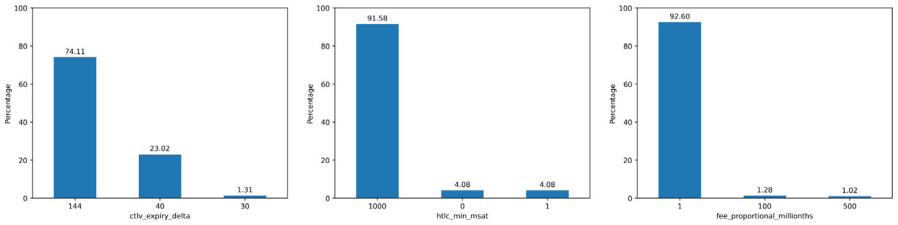

Fig. 9. LND: cltv_expiry_delta (left), htlc_minimum_msat (middle) and fee_proportional_millionths (right).

3.4. Results and evaluation

Fig. 8 shows that we have been able to classify almost all nodes in our data set, as 99.9% of the nodes have been classified successfully. Both data processing methods have classified the nodes very similarly. We can see that our analysis resulted in labeling approximately about 87% of the nodes as LND nodes, about 11% of the nodes as C-Lightning and about 2% of the remaining nodes as Eclair, making it the least used implementation in the network.

As far as the accuracy of the classification is concerned, the next graphs give us more insights. For all nodes labeled as a certain implementation, we considered the parameters based on which the classification algorithm made its decision. Fig. 9 shows the parameter distribution for `cltv_expiry_delta`, `htlc_min_msat` and `fee_proportional_millionths` for all nodes inferred as LND. The left plot from Fig. 9 shows us that in fact the two most used values for the parameter `cltv_expiry_delta` 144 and 40, yielding 97.13% are default values for LND. Interestingly, 1.31% of the nodes labeled as LND use 30 as a parameter, this value is often used by a certain node provider known as "LNBIG.com". For the next parameter `htlc_minimum_msat`, 91.58% of the nodes use the value 1000, also a default value. Some individual nodes also use a value of 0 or 1. For the last parameter `fee_proportional_millionths`, also very few nodes use a value other than the default value of 1. 500 is used by "LNBIG.com" and 100 is again used by some individual nodes. We have evaluated labeling precision of C-Lightning and Eclair as well, showing a similarly high distribution of the default values. We can deduce from these results that the nodes have been classified precisely.

3.5. Discussion

We have demonstrated, that with enough data, it is possible to precisely infer the implementations of almost all nodes. The problem with node classification in the Lightning Network is that a user in fact can easily change these parameters we based our classification upon. Once all of these parameters have other values than the default ones, a classification with these parameters is no longer possible and new methods have to be explored. However, in Section 3.1 we have shown that the usage of default values for parameters is very common, from which we can deduce that most users stick with the default settings. Also, our data set covers messages of almost two years. In this time span we have gathered numerous data points for the individual nodes in the network, which increases the chance to discover default values for nodes, even if their users changed these values at some point in the past.

Another possibility of performing a classification could be by incorporating some measurements concerning node performance in the network, since all implementations use a different programming language which could affect its efficiency. This method can be especially interesting when the implementation of a certain node is of interest and previous channel or node setting data is not available or the user has changed these parameters. However, further testing, data gathering and research will be necessary.

Lastly we want to address the implementation distribution. From our results and also the results of [14] we can observe that LND is by far the most popular client for the Lightning Network. Though we cannot precisely substantiate why this

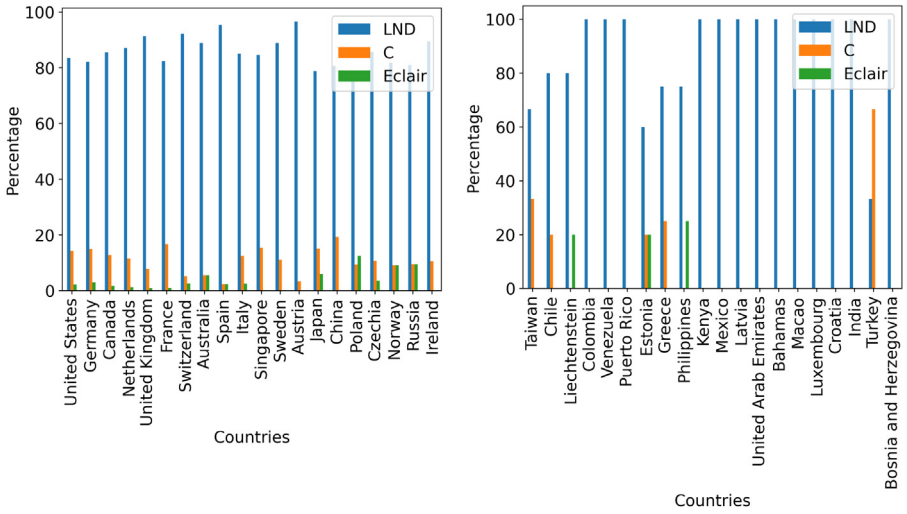

Fig. 10. Client distribution in individual countries.

is the case, we can make some assumptions based on facts. Firstly, C-Lightning has already released first versions of its clients by 2016. These releases where only provided as source code and had to be compiled manually by the user using Linux or possibly other UNIX based operating systems. LND released its first major version in 2017. However, in contrast to C-Lightning or Eclair, LND offered already precompiled versions of the client for the various CPU architectures and operating systems, including Windows. Especially, making the client available for Windows could have made a huge impact on LND's popularity.

4. Analysis of the geographical distribution

In this section we present our findings with regard to the geographical conditions of the Lightning Network. Similar studies have been carried out in the past, mainly concerning cryptocurrency networks for example Bitcoin. In this paper, we go beyond previous work as we perform this kind of analysis on a novel payment channel network, in particular on the Lightning Network. Our analysis includes 81 countries, in which the Lightning Network is present. Our focus lies especially on North America, Europe and Asia, since these are the continents with 94% of the overall node population.

4.1. Implementation distribution in countries

After obtaining the labels for each node, we now look at the implementation distribution in the individual countries. In the previous section, we have seen how the labels are distributed in general. We observed that LND is with about 87% by far the most frequently used implementation, followed by C-Lightning and Eclair, with 11%, respectively 2%.

By considering the IP-address for each classified node, if available, we can approximately determine its location, using an API [16]. We analyze the implementation distribution in each country, to see if the results on a country level mirror our previous results.

Figs. 10 shows the distribution in 40 individual countries. The results show that the implementations within a country are similarly distributed as in our general labeling results. For all the countries, which we could determine through a node's IP-address, we observed that LND is predominant in 78 out of all 81 countries. Only Turkey, Iran, and the Isle of Man shows a higher C-Lightning usage. With respect to Eclair, there is no country where it has a higher share than LND. However, in seven countries C-Lightning has an equal distribution, and in Poland, Belgium, Lichtenstein and the Philippines, Eclair holds a higher distribution. Further analysis (Fig. 10) shows that countries, where a single implementation is represented by 100% of the nodes or where one of the other two implementations holds a higher share than LND, only have a few or just one single node. For example, in India there is a total of four nodes, from which all use the LND implementation.

4.2. Channel connection behavior analysis

We next analyze if there is a geographical preference in the connection behavior of nodes. For example, do nodes connect to more geographically closer nodes for network latency benefits, or does the location not matter at all and are there other reasons for establishing a connection. Seres et al. [17] state that nodes, because of the way they are implemented, tend to create channels with already large hubs rather than smaller nodes; this makes the network prone to topological attacks, since the removal of only one hub can already heavily impact the network's connectivity.

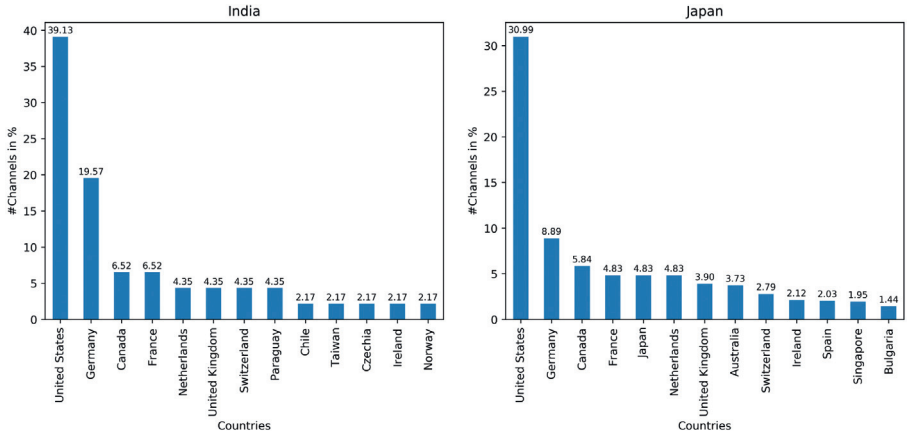

Fig. 11. Channel Connections: India (left), Japan (right).

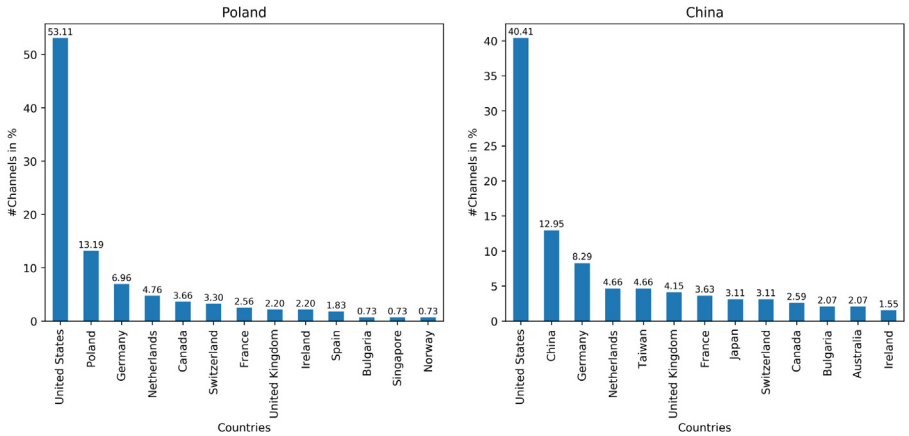

Fig. 12. Channel Conn.: Poland (left), China (right).

For each country, we examined the top countries the nodes in that specific country connect to and analyzed their endpoints. Our empirical analysis shows that nodes indeed tend to connect to large hubs, even if there is a large distance between them. Almost every node from all 81 considered countries connects to the same six countries: the United States, Germany, Canada, the Netherlands, United Kingdom, France, and maybe interestingly, Switzerland. In almost all cases, the USA is the leading country to connect to, followed by Germany. A significant portion of the nodes in the Lightning Network are located in these countries, and also the most connected nodes, with the United States having more than 280 000 channels, Germany having more than 3000, and Canada and the Netherlands having more than 3000 and 2000 channels. In Fig. 11, depicting the channel connections in India and Japan, we can observe this pattern as well. Interestingly, a lot of countries, i.e., Japan, Poland, or China, also show a high node connectivity within the country, depicted in Figs. 11 and 12. This pattern could be due to country specific node providers, whose nodes are interconnected within a country.

4.3. Analyzing node location

A large share of the nodes in the Lightning Network are located in North America with 44.8% and Europe with 43.1%. The remaining nodes are located in Asia with 6.2%, Oceania with 2.2% and lastly South America and Africa with each having 0.8% and 0.6% of the nodes in the Lightning Network. For 2.3% of the nodes we could not determine a location because of missing IP-addresses. In Fig. 13 we can observe the node location distribution on three continents: Europe, North America and Asia. By plotting the node latitude and longitude coordinates, we can clearly identify Europe and North America, due to the high node density in these continents. Looking at Fig. 14 (left) we can see that most of the nodes are located in Central Europe. Fig. 14 (middle) shows a very high node distribution on both the West Coast and the East Coast, as well as occasionally inside the country. In Asia, depicted in 14 (right), most of the nodes are located on the coasts of South Korea, China and Japan.

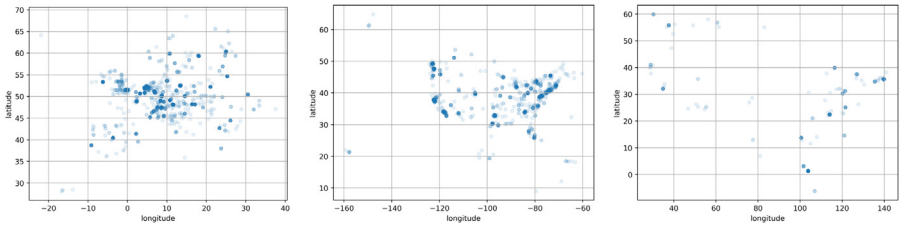

Fig. 13. Nodes in Europe (left), North America (middle) and Asia (right).

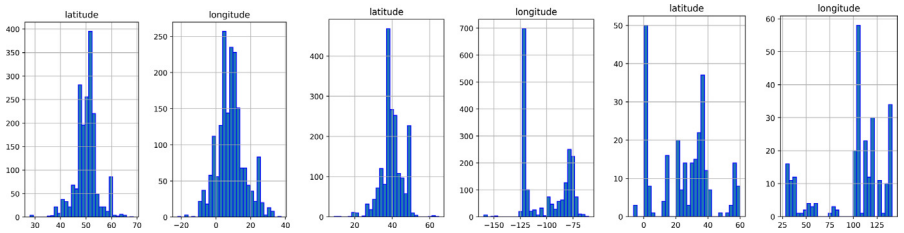

Fig. 14. Node latitude and longitude in Europe (left), North America (middle) and Asia (right).

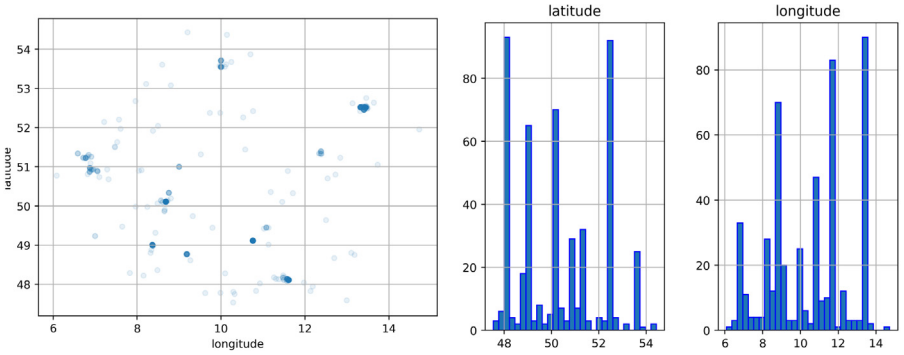

Fig. 15. Nodes and latitude and longitude in Germany.

From Fig. 13 we can also observe that locations, where the node density is higher, tend to have a better infrastructure, e.g., Central Europe–Eastern Europe, North America West/East Coast–North America Inland.

To further evaluate on this aspect, we studied the node location distribution inside of a country, e.g., in Germany and Japan. In Fig. 15 we can see multiple node clusters in Germany, each centered around one of Germany's larger cities, with the largest being in the metropolitan area of Berlin (52.52, 13.40) and second and third largest around Munich (48.13, 11.57) and Frankfurt (50.11, 8.68). For Japan depicted in Fig. 16, the largest node hub is located in the metropolitan area of Tokyo (35.65, 139.74), followed by the metropolitan areas of Osaka (34.66, 135.49), and Kobe (34.68, 135.19).

4.4. Discussion

Our findings exhibit how the Lightning Network and its implementations are distributed in the world. We observe that LND is popular in almost all countries and also showed that within a country nodes form clusters around cities and expand into their metropolitan areas. Also, infrastructure plays a significant role in the distribution of nodes within a continent or country.

Our analysis of channel connections between countries, shows a pattern of nodes always connecting to the same countries, and we have found other possibly interesting patterns as well. By normalizing the number of channels each country shares with other countries by the number of nodes in these countries, we observe that nodes in some countries, which are in relatively close proximity, tend to establish channels as well. Our first analysis has shown that Argentina shares 80% of the channels with Paraguay, then Peru with 10% of the channels and lastly with Chile and Venezuela with around 2%–4%. Kenya shares more than 70% of the channels with South Africa, China shares most channels with Taiwan and also several with Japan and Hong Kong, Slovenia shares channels with Croatia, Czechia and Bulgaria and Mexico with Colombia, Chile, Puerto Rico and Argentina. However, more studies have to be carried out in order to evaluate this behavior.

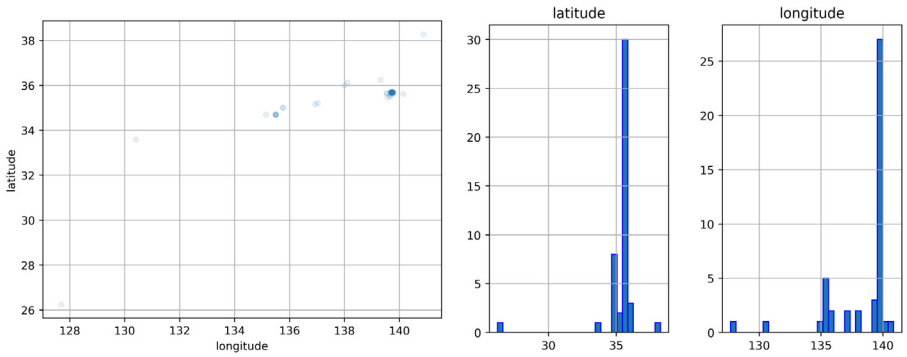

Fig. 16. Nodes and latitude and longitude in Japan.

In the next section, we will explore channels in Lightning further, investigating their relevant properties and combining our insights with our results from Section 3.

5. Analysis of channels and their properties

In this section we take a closer look at some interesting aspects of the channels in the Lightning Network such as balance distribution, lifetime or channel collaboration. Furthermore, we label each channel endpoint with its respective implementation, which not only enables us to get new insights concerning the connection behavior of the Lightning Network, but also allows us to explore new facets of already studied aspects in this field.

5.1. Data set

Our new data set consists of approximately 100 000 unique channel entries. To get the desired channel data we queried the Blockstream API [18] with the `sc_ids` as input, obtained from the `channel_announcement messages` in our previous data set. As a reminder, every time a new channel is created a `channel_announcement message` is broadcasted, which contains the unique channel ID, as well as the IDs of both channel endpoints. After we queried all `channel_announcement messages` in our previous data set we then extracted all the necessary information and created a completely new data set. We then proceeded to use the labeled `node-_ids` from Section 3, to assign each channel endpoint a corresponding label if a matching `node_id` was found. This process resulted in approximately 60 000 unique channels entries together with their labeled endpoints.

5.2. Channel analysis parameters

In this subsection we introduce important parameters, which were used in our channel analysis. Some of them are directly presented in the response JSON of the Blockstream API, others need further inquiry to be deduced.

sc_id. The `short_channel_id` is a unique description of the funding transaction to create a channel and therefore uniquely identifies a channel in the Lightning Network. The `sc_id` consists of the block height, transaction index and output index. The encoded block height is also the creation date of the channel.

node_id. Similarly to the `short_channel_id` of a channel the `node_id` is used to uniquely identify a channel endpoint (a node). But contrary to the `sc_id`, the `node_id` does not include more information.

status. This parameter tells us if the channel is still open or has already been closed. We check if the funding output of the funding transaction is still unspent in which case the channel is still open. If this is no longer the case, the channel is already closed and we then proceed to check the closing transaction.

lifetime. The lifetime of a channel is measured in blocks. Only channels that have been closed have a lifetime. However, it is possible to get the current lifespan of an open channel by subtracting the funding transaction block height with the current block height of the blockchain.

close_type. We deduced a channel `close_type` from its output types. The two endpoints closed a channel in a `collaborative` way (both endpoints wanted to end the connection) if and only if the output types are `p2wpkh`. In this case the `close_tx` has only one or two outputs, depending if the balance was only on one side of the channel. However, if one of the output types is `p2wsh` the channel has closed in an `unilateral` way (only one endpoint wanted to end the connection).

penalty. We also take a look at the `to_local` output to determine if it was swept (correct close) or if it was penalized (the closer attempted to cheat). In our data set this parameter is either `True` if both sides worked correctly, or `False` if one of them tried to cheat.

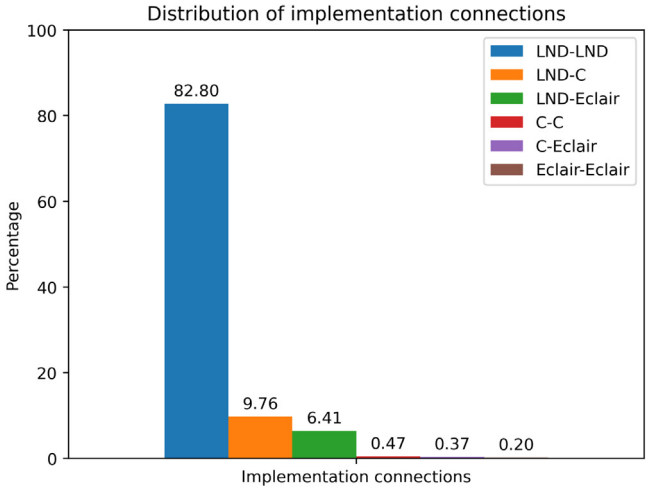

Fig. 17. Implementation connections.

5.3. Distribution of implementation connections

Since we have added labels to the channel endpoints we can now explore their connection distribution in the network. Fig. 17 depicts all possible pairs and their share in percent. Note that Lightning Network channels are bidirectional which means that LND-C is the same as C-LND. We can observe that channels with both endpoints being LND make up by far the largest share with 82.80%. The next largest share, but significantly lower, is made up of LND-C endpoints, with 9.76%, followed by LND–Eclair having a share of 6.41%. At the end of the spectrum we can see that C–C makes up a share of 0.47%, C-Eclair makes up 0.37% and lastly Eclair–Eclair makes up only 0.20%. In our previous sections we have already observed that LND is, by a large margin, the currently most popular implementation and in Section 3.5 we elaborated why this might be the case. Also LND is used by numerous LN node providers, whose nodes tend to be highly interconnected.

5.4. Channel lifetime

In this subsection we examine the lifespan of channels in the Lightning Network. Fig. 18 depicts the average channel lifetime of each implementation pair. Interestingly, we can observe that almost all channels with endpoints having the same implementation have a noticeably longer lifetime than nodes where the implementations differ. LND–LND and C–C channels have a relatively identical lifetime of 30,522 and respectively 29,395 blocks. As of performing this research, it takes approximately 10 min on the Bitcoin blockchain to discover a new block. This lifetime in blocks would therefore convert to nearly 212 days for LND–LND channels and 204 days for C–C channels. Next, LND-C channels have a lifetime of 26,460 blocks or 185 days and marginally lower Eclair–Eclair channels have a lifetime of 26,063 blocks or 181 days. Lastly, with the overall lowest lifetime, C-Eclair channels live for 23,660 blocks or 164 days on average and LND–Eclair channels live for only 21,402 blocks or 149 days.

Now let us take a more detailed look at the channel lifetime of each implementation pair. Fig. 19 depicts a box plot for each implementation pair with and without outliers. At the first glance we can see that all channels where LND is at least one endpoint, have a predominant amount of outliers. We can also observe that the minimum lifetime value for all pairs is near zero, indicating that some channels live only for a momentary period of time. Furthermore, the median values of all pairs are roughly located around the 20 000 block mark, except for the C–C pair where the median value and the average are extremely close by. However, this can be easily explained, since C–C has surprisingly no outliers. In all other cases, we can see that all outliers are located above the maximum value (since the minimum value is almost zero), which also explains why the average value is above the median by a larger margin. For LND–LND and C–C not only the average values are within a close range, but also the first quartile (Q1), the third quartile (Q3) and the maximum value. Now let us inspect some of the extreme outliers, beginning with LND–LND. Here the largest outlier lies in between the 150 000 to 155 000 block mark, which would be around 1042 to 1076 days or almost three years (2.9 years). For LND-C and LND–Eclair the largest outliers are located at the 140 000 and 130 000 block mark, converting to approximately 2.6 and 2.5 years. This tells us that some of the channels in the Lightning Network have been alive almost since its creation (specifically the creation of the respective implementations). Keep in mind that we have only looked at already closed channels, so there could potentially be still open channels which are older or/and could have an even longer lifetime.

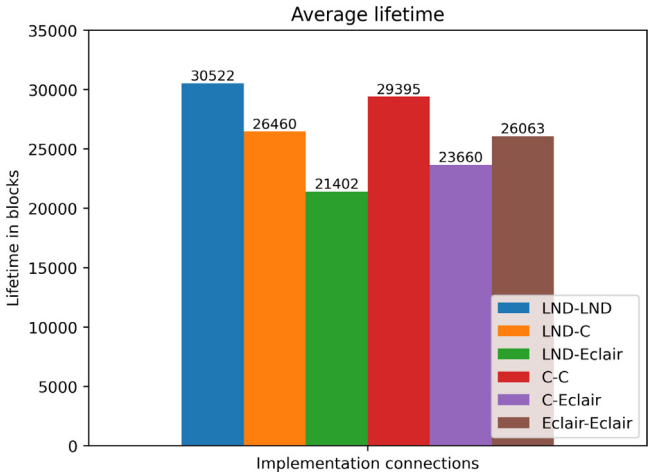

Fig. 18. Average lifetime of channels.

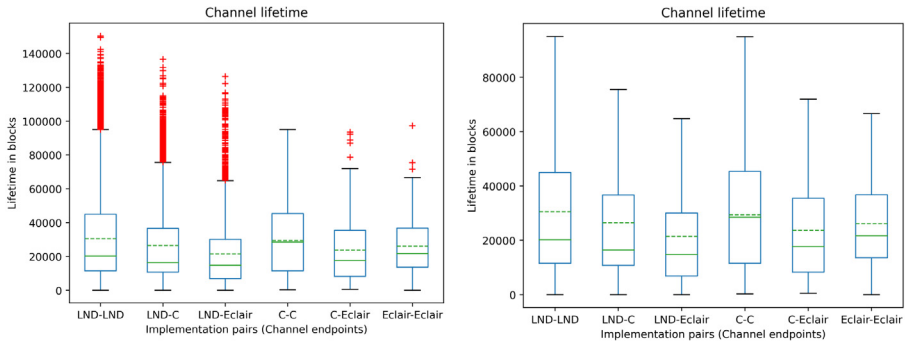

Fig. 19. Channel lifetime box plots with (left) and without outliers (right).

5.5. Channel balance

At the beginning, we can observe in Fig. 20 that LND–LND and LND-C channels continue to correspond to the pattern we have already seen in the previous subsections, namely that they share very similar values. The average initial balance of LND–LND channels is 3 113 993 satoshi and respectively 2 869 396 satoshi for LND-C. So a difference of only 244 597 satoshi, which converts to approximately 145 USD at the time of writing (April 2021). LND–Eclair channels seem to have a bit higher initial channel balance of 3 993 481 satoshi. However, by looking at the other pairs we can observe a drastic difference in balances. C–C channels seem to be the poorest of all with only 1 373 892 satoshi, but Eclair–Eclair channels on the other hand seem to be the richest of all pairs with having an average initial balance of 7 584 335 satoshi or an average of 4480 USD. Though, C-Eclair channels have a higher initial balance than C–C channels, none of them comes near to Eclair–Eclair. This fact raises the question, why this specific pair has such a high initial channel balance, while C–C and C-Eclair channels do not and which endpoints might be responsible for such high balances, C or Eclair endpoints? However, averages can be very biased, when exorbitantly large or low values appear in the data set.

For this reason we take a closer look at Fig. 21 (left), which gives us a more in-depth view. In fact, we can clearly see that the average values (dotted line) and median values (normal line) of all pairs differ heavily. In the case of LND–LND and LND-C both averages are close to the Q3 value. This is also the case for LND–Eclair and C–C, where the average value is extremely close to Q3. In the case of Eclair–Eclair we can now detect that the average value was biased by a large value, because the median is in the range of approximately 700 000 satoshi. Overall, we can observe that the median values of all implementation pairs are in the range of 700 000 to 1 000 000 satoshi.

Lastly, we want to examine the most extreme outliers, which are depicted in Fig. 21 (right). We see that the channel pairs not only have numerous outliers but are also distributed excessively, which results in squeezed and barely readable box plots. The largest outliers in LND–LND and LND–Eclair are located at about the same height, namely 260 000 000 satoshi. This amount converts around 153 600 USD. LND-C, C–C and C-Eclair have their maximum outliers also around the same height, namely 25 000 000 satoshi, which would be 14 800 USD. The outlier of Eclair–Eclair is located at 180 000 000 satoshi or 106 343 USD.

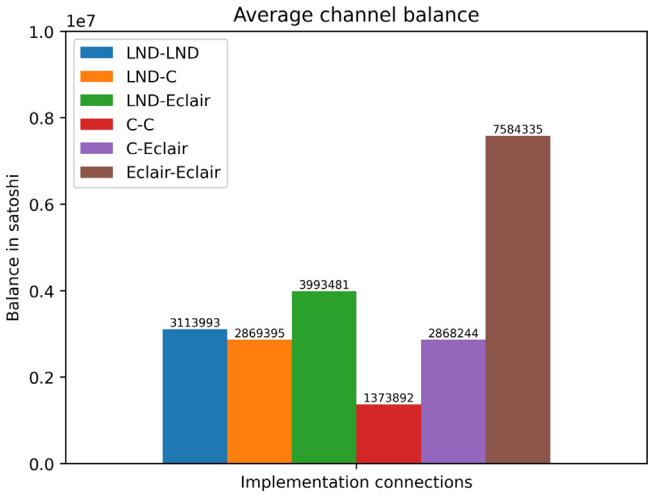

Fig. 20. Average initial channel balance (left) and final channel balance (right).

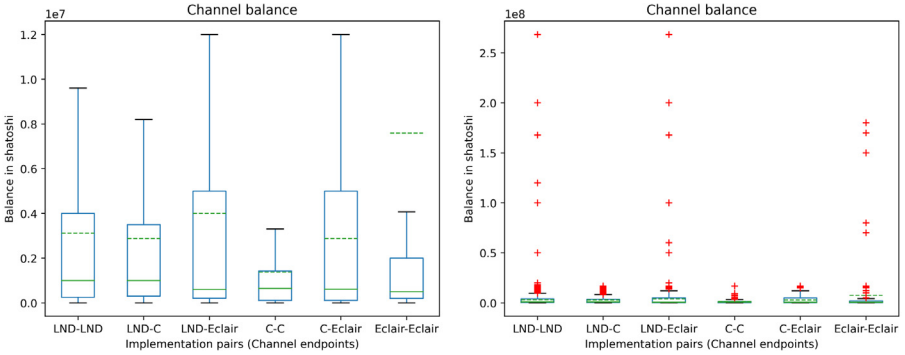

Fig. 21. Channel balance without outliers (left) and with outliers (right).

5.6. Lifetime-balance correlation

After discussing both channel lifetime and channel balance we want to examine if these two attributes correlate. Our initial assumption is that a higher initial channel balance might lead to a longer overall channel lifetime. However, after plotting both attributes we can see that this is not the case. In Fig. 22, 60 000 channels are depicted. The *x*-axis encodes the lifetime attribute and the *y*-axis encodes the channel balance of each channel. There is no correlation visible, the data points rather seem to have an even distribution. We can observe that the largest share of channels has a broad and even distribution throughout the *x*-axis from almost zero blocks to 140 000 blocks lifetime and that most of them are in a balance range of zero to almost 25 000 000 satoshi. Furthermore, there are multiple extreme cases, where the channel balance is extraordinarily high, but the channel lifetime is on the lower end and vice versa. To further prove our observations we calculated the Pearson correlation coefficients for every channel of an implementation pair and the results stand in agreement with our presumptions we gained from the plots. The coefficients for all pairs range from −0.09 to 0.07. Performing the same calculations with final balances and other correlation coefficients yield about the same results.

5.7. Close types

Channels can close collaboratively, unilaterally and even with penalties if one of the endpoints tried to cheat. We examined the closing behavior of the channels in our data set per pair presented in Fig. 23. Both collaborative and unilateral closures seem to be highly present, with unilateral closures being predominant in four of the six types of endpoint pairs. Interestingly, cheating is not very common in the Lightning Network with being under 1% in all cases. LND–LND is the pair with most endpoints closing the channels collaboratively yielding 60.43%. LND-C continues with this trend, however, in comparison to the first pair we note that unilateral closures gained approximately 5%, whereas collaborative closures lost this share. All pairs with at least one being Eclair exhibit more unilateral closures than collaborative. C–C

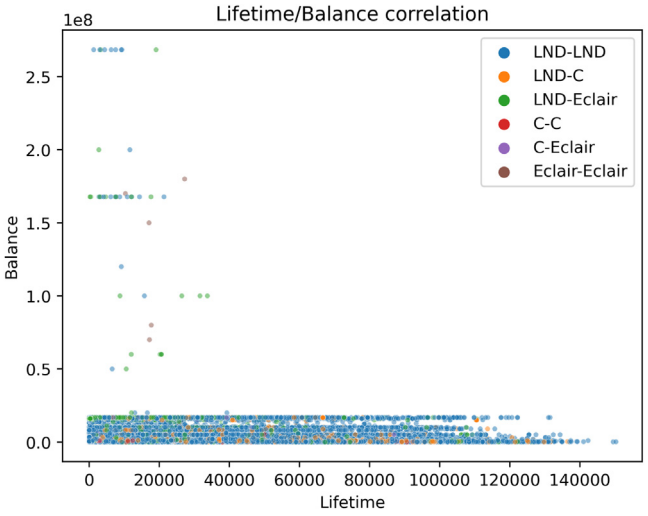

Fig. 22. Balance and lifetime correlation.

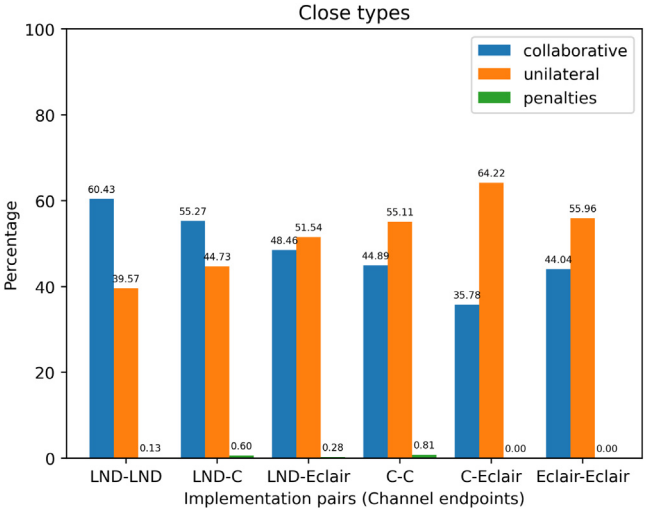

Fig. 23. Close types of channels.

follows this trend as well and additionally experiences the highest share of penalties with 0.81%. Although it seems that LND nodes tend to close channels collaboratively and Eclair nodes on the other hand unilaterally, we want to emphasize that because of the uneven distribution of the pairs in the data set the shown data might not be representative, since there are substantially fewer Eclair nodes than LND nodes in our data set[1] and overall in the Lightning Network.

5.8. Endpoint reconnection

In this section we examine if endpoints reconnect to the same endpoints after a channel has closed or if they connect to new ones. Fig. 24 (left) depicts the share for reconnection of all endpoints in our data set. We can observe that only 13.80% of all endpoints reconnect to a former partner with which they have had a channel in the past. The substantially larger share of 86.20% of the endpoints are connections to a new partner, with which they have never had a channel before. The average of reconnections is 1.20 and the maximum amount of reconnections is 147 times all done by only two endpoints. Now let us take a more detailed look at the individual pairs shown in Fig. 24 (right). In all of the pairs endpoints connect to a new endpoint significantly more often than they reconnect to an old one. LND–Eclair and Eclair–Eclair shows the highest share of reconnectivity of all pairs with 26.62% and 38.81%. This may be again due to the circumstance that there

[1] See Figs. 8 and 17.

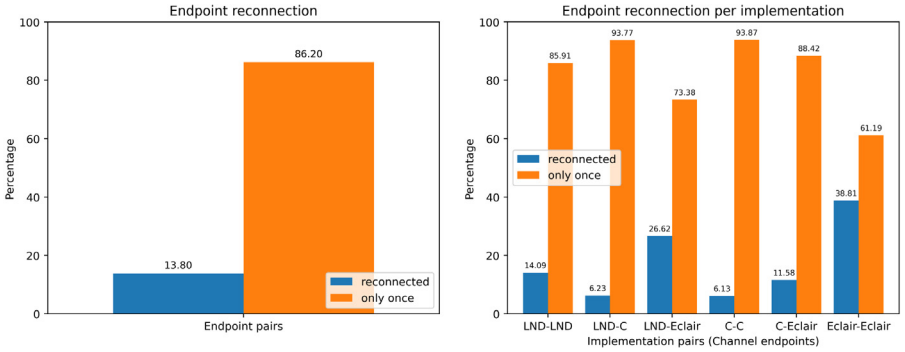

Fig. 24. Endpoint reconnections in total (left) and per implementation (right).

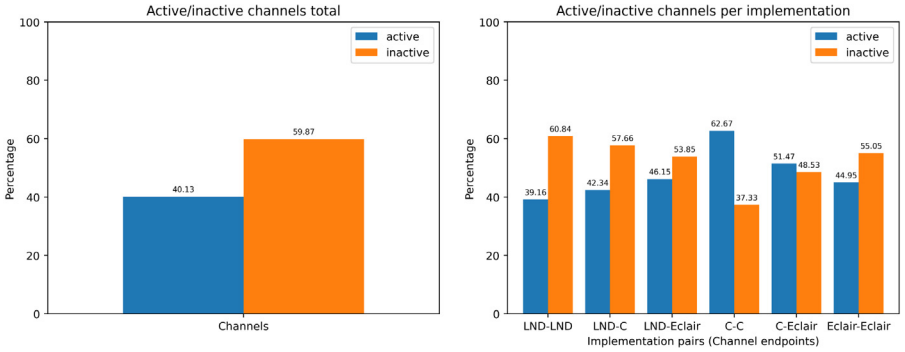

Fig. 25. Channel activity in total (left) and per implementation (right).

are fewer Eclair nodes in the Lightning Network and in consequence the node pool they can choose from is highly limited, if they want to stay connected with an endpoint of the same implementation. However, this is just an assumption and needs further research to be validated.

5.9. Channel activity

Finally, we examine the channel activity in the Lightning Network. In our definition a channel has been inactive (has never been used), when at the time of closing only one endpoint gets its credit back, ergo the `close_transaction` has only one output. Our interpretation is simple: If the channel liquidity is only on one side of the channel at the closure, this means that money has never flown through the channel, otherwise there would not be only one output. This includes both, direct transactions between the channel endpoints and also indirect cases where the channel has never been used for forwarding transactions between endpoints which are not directly connected. Our results, depicted in Fig. 25 (left), show that of all closed channels in our data set, 40.13% have been active at some point in their lifetime. The bigger share of channels, namely almost 60%, has never been used after the funding stage. In Fig. 25 (right) we can observe the channel activity broken down based on the endpoint pairs. LND–LND and LND-C again share similar results, as we have seen before. Only 39.16% and 42.34% of the channels have been used in the cases of LND–LND and LND-C. LND–Eclair and Eclair–Eclair endpoints also have a similar distribution of 46.15% active channels in the first case and 44.95% in the latter. Surprisingly, C–C endpoints seem to have the most active channels with a share of 62.67%. Also the C-Eclair endpoint pair shows a higher share of active channels with 52.67%.

5.10. Discussion

Our discoveries in this section reveal numerous new aspects of channels in the Lightning Network. We observe the channel distribution in terms of implementation endpoints and find that channels with both endpoints being LND are the most prominent in the network. Our channel lifetime analysis shows that channels have an average lifespan of 26250 blocks, with some LND–LND and LND-C channels having a maximum lifespan of more than 140 000 blocks and therefore being almost as old as the implementations themselves which launched in 2017 for LND and respectively 2016 for C-Lightning. The channel balance analysis gives us new insights into the balance distribution of channels. Lightning Network channels posses an average balance of 3 632 990 satoshi and a median balance of 700 000 to 1 000 000 satoshi.

Additionally, Fig. 21 demonstrates that channel balances vary largely and can take up values as high as 260 000 000 satoshi. Moreover, we have refuted our assumption of channel lifetime and balance being in any kind of dependency. Furthermore, we have examined in which manner channels tend to close their connections. Both, collaborative and unilateral closures are very common. Interestingly, penalties are extraordinarily rare, meaning that endpoints seem to behave quite fairly. Then, we observe that reconnections to the same endpoint are rather rare. Endpoints tend to reconnect only 1.2 times on average with 147 reconnections being the maximal number in our findings. Lastly, we have seen the distribution of active and inactive channels. Finding that overall there are less active channels than inactive ones and that C–C has a significantly higher amount of active channels than any other implementation pair.

6. Related work

For an overview of the blockchain and Bitcoin in general we refer the reader to the paper by Antonopoulos [1], for a survey on blockchain networking covering both onchain and offchain aspects to [19], and for an overview of off-chain networks specifically to [2,19]. There exist many clever route discovery algorithms in the literature, e.g., SpeedyMurmurs [20] and SilentWhispers [21], to improve the routing efficiency in off-chain networks. However, it has also been shown that the gossiping and probing mechanisms needed in off-chain networks to support efficient routing, may introduce security issues, e.g., harm privacy [22] and/or performance if nodes behave selfishly [12].

Some papers already have explored node classification in the Lightning Network, mostly as a preparation measure for an attack. Mizrahi et al. [14] performs a classification for a congestion attack on the Lightning Network and also suggests mitigation techniques. The data for this work was gathered with the `describegraph` command of LND, which returns a JSON describing the networks topology to a given timestamp. Our data was gathered by a node logging the messages it received from March 2018 until January 2020. We also consider more parameters to ameliorate the classification results and perform some further analysis with the results. Pérez-Solà et al. [13] proposes an attack to discover channel balances in the network and also infers implementations for deriving a private parameter's value.

Not only off-chain networks but also on-chain networks have been studied intensively in the literature already. BTCdoNET [23] is a measurement tool for Bitcoin's P2P network which relies on crawling Bitcoin peers' list of known other peers. For example, the authors find that the nodes placed in United States and China amounted to 37% of the discovered nodes. The degree of centralization was measured by Gencer et al. [24], comparing Bitcoin and Ethereum (the latter has also been studied in [25]), and using IP addresses as hints of geo-location. In [26], Miller et al. determine the Bitcoin topology by utilizing the update method of the timestamp field. Another aspect that has been studied empirically in the literature are blockchain forks in Bitcoin [27].

More generally, measurement studies, e.g., also considering geographic aspects, have been performed on many other peer-to-peer and social networks, also before cryptocurrencies. For example, Schiöberg et al. [28] conducted an analysis of the social network Google+, which also includes an examination of user locations. Scellato et al. [29] study how geographic distance affects social ties in a social network and Mislove et al. [30] geographical, gender and racial aspects of Twitter users to the U.S. population. Measurements and implications on attacks in the peer-to-peer network Kad have been discussed by Locher et al. [31]. Dotan et al. [32] recently presented an overview of cryptocurrency networks, which also includes a survey of empirical studies on geological characteristics of the Bitcoin and Ethereum networks. However, we are not aware of works investigating similar aspects on off-chain payment channel networks so far. The Lightning Network's topology has been analyzed by Seres et al. [33]. The work studies the robustness of the network against random failures of nodes and targeted attacks. The authors also propose some countermeasures to make the network more resilient. A similar, but more in depth work has been carried out by Rohrer et al. [34]. As far as privacy is concerned, Kappos et al. [35] performed an empirical analysis of the Lightning Network based on three attacks. The analysis also included measurements concerning network, node, and channel parameters.

7. Conclusion

Analyzing a big data set collected on the communication in the Lightning networks, we have shown that it is possible to accurately classify the node types in Lightning Network with high probability, potentially providing important security insights. We understand our work as a first step and we plan to continue collecting data for more extensive studies, also of the network evolution over time. We also believe that our work opens several interesting questions for future research. For example, it will be interesting to explore alternative classification algorithms, improving the accuracy further and to investigate the applicability of our methods on alternative off-chain networks.

In order to support such future research, our data is available at https://github.com/lnresearch/topology [36].

Declaration of competing interest

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

Acknowledgment

This research is supported by the Vienna Science and Technology Fund (WWTF), Austria ICT19-045 project.

References

[1] A.M. Antonopoulos, Mastering bitcoin: Unlocking digital crypto-currencies, 2nd, O'Reilly Media, Inc., 2017.
[2] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, SoK: Off the chain transactions, IACR Cryptology ePrint Archive (2019).
[3] Lightning Network, Lightning network specifications, 2020, Online https://github.com/lightningnetwork/lightning-rfc/. (Accessed 01 July 2020).
[4] Raiden Network, Raiden network, 2020, Online https://raiden.network/. (Accessed 02 January 2020).
[5] R. Fugger, Money as IOUs in social trust networks & a proposal for a decentralized currency network protocol, 106, 2004, Hypertext Document. Available Electronically at http://ripple.sourceforge.net.
[6] C-lightning GitHub repository, 2020, Online https://github.com/ElementsProject/lightning. (Accessed 15 July 2020).
[7] LND GitHub repository, 2020, Online https://github.com/lightningnetwork/lnd. (Accessed 15 July 2020).
[8] Eclair GitHub repository, 2020, Online https://github.com/ACINQ/eclair. (Accessed 15 July 2020).
[9] C-lightning alias, 2020, Online https://github.com/ElementsProject/lightning/blob/v0.6rc2/lightningd/options.c. (Accessed 16 July 2020).
[10] Lightning Network, BOLT 7: P2P node and channel discovery, 2019, Online https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md. (Accessed 4 December 2019).
[11] S. Tochner, S. Schmid, A. Zohar, Hijacking routes in payment channel networks: A predictability tradeoff, 2019, CoRR abs/1909.06890 arXiv:1909.06890.
[12] S. Tochner, S. Schmid, On search friction of route discovery in offchain networks, 2020, CoRR abs/2005.14676 arXiv:2005.14676.
[13] C. Pérez-Solà, A.R. Pedrosa, J. Herrera-Joancomartí, G. Navarro-Arribas, J. García-Alfaro, LockDown: Balance availability attack against lightning network channels, IACR Cryptol. EPrint Arch. 2019 (2019) 1149.
[14] A. Mizrahi, A. Zohar, Congestion attacks in payment channel networks, 2020, CoRR abs/2002.06564 arXiv:2002.06564.
[15] LND change 144 to 40, 2020, Online https://github.com/lightningnetwork/lnd/commit/c302f1ea3a91ccfa382d56851d23f4c73656208c#diff-356ddb2e7efca712327c3b2d94d3afd3. (Accessed 16 July 2020).
[16] Ipinfo, 2020, Online https://ipinfo.io. (Accessed 15 July 2020).
[17] I.A. Seres, L. Gulyás, D.A. Nagy, P. Burcsi, Topological analysis of bitcoin's lightning network, 2019, CoRR abs/1901.04972 arXiv:1901.04972.
[18] Blockstream, Blockstream API, 2021, Online https://blockstream.info/. (Accessed 10 March 2021).
[19] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, A. Zohar, Survey on blockchain networking: Context, state-of-the-art, challenges, in: Proc. ACM Computing Surveys, CSUR, 2021.
[20] S. Roos, P. Moreno-Sanchez, A. Kate, I. Goldberg, Settling payments fast and private: Efficient decentralized routing for path-based transactions, 1709.05748, 2017, CoRR.
[21] Malavolta, et al., SilentWhispers: Enforcing security and privacy in decentralized credit networks, in: NDSS, 2017.
[22] U. Nisslmueller, K. Foerster, S. Schmid, C. Decker, Toward active and passive confidentiality attacks on cryptocurrency off-chain networks, in: ICISSP, SCITEPRESS, 2020, pp. 7–14.
[23] J.A.D. Donet, C. Pérez-Sola, J. Herrera-Joancomartí, The bitcoin P2P network, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 87–102.
[24] A.E. Gencer, S. Basu, I. Eyal, R. Van Renesse, E.G. Sirer, Decentralization in bitcoin and ethereum networks, in: International Conference on Financial Cryptography and Data Security, Springer, 2018, pp. 439–457.
[25] S.K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, M. Bailey, Measuring ethereum network peers, in: Proceedings of the Internet Measurement Conference 2018, 2018, pp. 91–104.
[26] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, B. Bhattacharjee, Discovering bitcoin's public topology and influential nodes, 2015, et al.
[27] T. Neudecker, H. Hartenstein, Short paper: An empirical analysis of blockchain forks in bitcoin, in: International Conference on Financial Cryptography and Data Security, Springer, 2019, pp. 84–92.
[28] D. Schiöberg, S. Schmid, F. Schneider, S. Uhlig, H. Schiöberg, A. Feldmann, Tracing the birth of an OSN: social graph and profile analysis in google+, in: WebSci, ACM, 2012, pp. 265–274.
[29] S. Scellato, C. Mascolo, M. Musolesi, V. Latora, Distance matters: Geo-social metrics for online social networks, in: WOSN, USENIX Association, 2010.
[30] A. Mislove, S. Lehmann, Y. Ahn, J. Onnela, J.N. Rosenquist, Understanding the demographics of Twitter users, in: ICWSM, The AAAI Press, 2011.
[31] T. Locher, S. Schmid, R. Wattenhofer, Edonkey & eMule's kad: Measurements & attacks, Fund. Inform. 109 (4) (2011) 383–403.
[32] M. Dotan, Y.-A. Pignolet, S. Tochner, S. Schmid, A. Zohar, Cryptocurrency networking: Context, state-of-the-art, challenges, in: Proc. 15th International Conference on Availability, Reliability and Security, 2020.
[33] I.A. Seres, L. Gulyás, D.A. Nagy, P. Burcsi, Topological analysis of bitcoin's lightning network, in: Mathematical Research for Blockchain Economy, Springer, 2020, pp. 1–12.
[34] E. Rohrer, J. Malliaris, F. Tschorsch, Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks, in: EuroS&P Workshops, IEEE, 2019, pp. 347–356.
[35] G. Kappos, H. Yousaf, A.M. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, S. Meiklejohn, An empirical analysis of privacy in the lightning network, 2020, CoRR abs/2003.12470 arXiv:2003.12470.
[36] C. Decker, Lightning network research; topology datasets, 2020, https://github.com/lnresearch/topology. (Accessed 01 October 2020).