# Comparative Analysis of Lightning's Routing Clients

Satwik Prabhu Kumble
TU Delft
s.prabhukumble@tudelft.nl

Stefanie Roos
TU Delft
s.roos@tudelft.nl

2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) | 978-1-6654-3485-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAPPS52256.2021.00014

***Abstract*—Lightning Network, the off-chain solution to Bitcoin, uses a combination of source routing, layered encryption and conditional commitments to perform anonymous and atomic multi-hop off-chain payments. The three predominant routing clients used in Lightning — LND, c-Lightning and Eclair — differ in their cost functions for adding channels to a payment path and hence result in differ paths. At the moment, there is no detailed study on how the different path selections affect the quality of the payment in terms of effectiveness and efficiency. Such an evaluation is needed to i) allow users to make informed choices of which client to use, and ii) enable developers of new routing algorithms to take lessons learned from the current designs into account. In this work, we analyze and compare the three predominant routing clients used by Lightning in terms of the path length, maximal delays a payment might experience, fee borne by the sender and success rate of multi-hop transactions. Based on our experiments and results, we recommend LND for users desiring payments with short paths and high success rates, c-Lightning for users desiring paths with low maximal latencies, and Eclair for those looking for cheap paths in terms of fees paid to the nodes that route payments as intermediaries. We further give guidelines for the design of future routing algorithms based on our insights gained from Lightning's current routing.**

***Index Terms*—Payment Channel Networks, Lightning Network, Routing**

## I. Background and Introduction

Payment channel networks like Lightning [14] constitute one of the most promising approaches to overcome the scalability limitations of Proof-of-Work blockchains such as Bitcoin [10] and Ethereum [21] by offloading transactions to an off-chain layer. Additionally, they also promise improved latencies and lower fees in comparison to on-chain transactions.

Two nodes open a payment channel by depositing collateral on the blockchain. After opening the channel, they can locally transact with each other as often as they desire provided that the transaction values are bounded by their locked collateral. Each such transaction changes the distribution of the collateral between the two nodes, i.e., the balance of each node. A channel can be closed unilaterally by any of the two nodes operating the channel, so enabling both the nodes to collect the funds they are owed based on the channel balance distribution. Thus, direct interaction with the blockchain is only necessary for opening and closing channels as well as for resolving disputes raised regarding the channel balances [4].

A payment channel network (PCN) like Lightning is the network of all such payment channels that enables multi-hop payments. Whenever the sender and the recipient do not share a channel and do not wish to deposit collateral for opening a new channel, they can route the payments over multiple existing channels in the network offering a small fee to intermediaries to participate in such a payment [1]. The sender determines the payment path to the recipient and the payment is executed by performing the same payment on each channel in the path atomically. Lightning uses Hashed Time-Locked Contracts(HTLCs) [20] to enforce each intermediary's commitment to make the payment. Each node on the path other than the sender will have to lock the payment amount on its outgoing channel in the payment for a maximal duration of *locktime*. Nodes can decide the locktime of their channel. Lower locktimes indicate a higher risk for the node to not be able to relay information within a time interval. Higher locktimes indicate that the maximal latency is higher. So, senders prefer lower locktimes while intermediaries might prefer the increased security of longer locktimes.

The three predominant protocols that Lightning uses for routing multi-hop payments – **LND** [2], **c-Lightning** [3] and **Eclair** [4] – help senders determine the payment paths that should have low locktimes, low fees and high probabilities for successful payments. The path length and locktime determine the total locked collateral in the channels involving the payment and the maximum time the collateral can remain locked. Channel fees determines the total monetary cost borne by the sender and success rate indicates whether the protocol can be relied on to make successful payments. The cost function of each of the three protocols uses the publicly available channel parameters to assign a cost to adding a channel to the payment path during its construction. The sender then selects the (one of the) paths for making the payment. For instance, the cost functions of **c-Lightning** and **LND** gives more significance to lower channel delay, while that of **Eclair** gives significance to the channel capacity, age and fee. There is however no current evaluation on the extent to which Lightning's routing protocols are effective in finding short paths with low locktimes and fees without compromising on the success of payments.

[1] https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md
[2] https://github.com/lightningnetwork/lnd
[3] https://github.com/ElementsProject/lightning
[4] https://github.com/ACINQ/eclair

978-1-6654-3485-0/21/$31.00 ©2021 IEEE
DOI 10.1109/DAPPS52256.2021.00014

*Our Contributions*

In this work, we study the effectiveness of the above three clients in terms of length of payment path, total locktime, fees borne by the sender and success rate of payments. To this end, we simulate the routing protocols of each client with balance redistribution for successful payments on a real world snapshot of Lighting with nearly 5000 nodes.

We found that **LND** is the best of the three in choosing short paths with path lengths typically under four hops and **Eclair** is the worst with path lengths exceeding six hops nearly 40% of the time. **c-Lightning** was found to be effective in choosing paths with low total locktimes, typically under 300 blocks. **Eclair** performed worst in terms of total locktimes as well by exceeding 750 blocks nearly 20% of the time. In contrast, **Eclair** was highly effective in finding paths with low fees in comparison to the the other two, while **c-Lightning** was the worst in this aspect. With regard to the success rate of payments, **LND** was the best with 99% successful transactions an **Eclair** performed the worst with only 66% of transactions typically succeeding.

## II. Multi-hop routing in Lightning Network

We first present the multi-hop payment mechanism in Lightning followed by the route selection algorithms used.

### A. Hashed Time-Locked Contracts and Payment Execution

Multi-hop payments in Lightning use source routing with the sender choosing a path to the recipient based on its local knowledge of the network topology using one of the three path-finding protocols **LND**, **c-Lightning**, or **Eclair** which we will discuss in detail in Section II-B. The commitments of each node on the path to make a payment to its successor are enforced through the protocol *hash time-locked contract (HTLC)* [20].

In a HTLC for a single-hop payment, the recipient first chooses a random number and computes its hash value. It then sends the hash value to the sender. The two parties then form a contract that if the sender receives the pre-image of the hash within time *locktime*, the sender will pay the amount that is due to the recipient. On receiving the hash value, the sender locks the amount, i.e., it does not use it for any other payments. If it does not receive the pre-image within locktime, the sending party gets the locked amount back. Disagreements on whether a payment should be made lead to disputes on the blockchain. Each channel sets its locktime during its funding transaction and the locktimes are publicly known.

To execute a multi-hop transaction, HTLCs are set up separately on each channel of the path using the same hash value. The sender first computes a path to the recipient based on its routing client. The recipient then chooses the random value and sends its hash to the sender. The hash is then communicated to all nodes on the path, enabling them to set up the HTLCs. Note that knowing the hash is sufficient to set up a HTLC. Once all the contracts are set up, the receiver sends the pre-image to its predecessor on the path who then forwards it along the path, eventually reaching the sender to resolve the HTLCs. The choice of locktimes for the contracts is critical as each party needs sufficient time to receive the pre-image from their successor and forward it to their predecessor.

Concretely, the sender is the first to set up a HTLC with its successor on the path choosing the locktime to be the sum of the publicly known locktimes of all the channels on the path. We call this value the *total locktime* of the channel between the sender and its successor. Each channel along the path computes its *total locktime* by subtracting the locktime of its preceding channel from the total locktime of the preceding channel for the payment.

Now, a node can only claim the funds from its predecessor if it has received the pre-image from its successor and hence paid said successor. After receiving the pre-image, the time to forward the pre-image is exactly the locktime value of preceding channel, giving the node sufficient time to claim its own funds. Thus, Lightning ensures that no honest party loses funds. A payment fails if even one party in the payment path has insufficient funds or if any party fails to send the pre-image on time. This mechanism has been shown to be susceptible to grieving and denial-of-service attacks due to funds being potentially locked in each channel of the path for a duration of the locktime of that channel [15] [5] [19] [6].

### B. Route Selection

Before discussing the route selection algorithms in detail, we specify the information the sender has available during route selection. All channels are broadcast upon construction, hence the sender has a topology snapshot including the initial balances of each channel. This broadcast includes the fee and locktime parameters of the channels as well. As the balances change over time, the sender only knows the total to and fro capacity of the channel. In addition, nodes' public keys are distributed via gossip to all other nodes.

For the source to determine the path, it furthermore requires knowledge of the values of fee and locktime $lt[u, v]$ parameters for each channel. In Lightning, an intermediary charges two types of fees, a constant base fee $bf[u, v]$ for using the channel and a proportional fee, that is the product of the fee rate $fr[u, v]$ parameter and the amount $amt[u, v]$ to be transferred, where $u$ and $v$ are the channel end-points. The fee charged by an intermediary $u$ is

$$fee[u, v] = bf[u, v] + fr[u, v] \cdot amt[u, v]. \quad (1)$$

The locktime and fee parameters can be different for each direction of the channel and any changes to these values need to be broadcast via gossip across the network. Additionally, the source also knows the total channel capacity $cap[u, v]$ and the channel age $age[u, v]$.

Each client designs their cost function in an attempt to find short paths with low locktimes and fees. Shorter paths imply that there are lesser points of failure, lower locktimes imply that the collateral is locked for smaller durations and lower fees imply lower costs for the sender to send the payment.

*Cost functions:* Each routing client used by Lightning determines a payment path from a sender to a recipient using Dijkstra's shortest path algorithm or Yen's algorithm [22] with a different cost function for including a channel in the path. The payment path is constructed starting from the recipient backwards to the source, since the proportional fee charged by an intermediary depends on the amount forwarded by it. Let us now discuss the cost functions for LND, c-Lightning and Eclair.

1) **LND**: The key idea of the cost function of **LND** is to weigh between paths of low timeouts and low monetary costs. In addition, there is a bias against channels that have entailed failures in attempts to conduct a payment in the recent past. The exact cost function is

$$cost[u,v] = amt[u,v] * lt[u,v] * rf + fee[u,v] + bias[u,v] \quad (2)$$

where $rf$ is a risk factor set to $15 \cdot 10^{-9}$ by default and $bias[u,v]$ accounts for previous payment failures caused by the channel $[u,v]$.
Each channel entails a success probability based on its past failures:

$$Pr[u,v] = \begin{cases} Pr_A \cdot (1 - \frac{1}{2^t}) & t >= \text{1hour} \\ 0 & t < \text{1hour} \\ Pr_A & t \text{ unknown} \end{cases} \quad (3)$$

where $Pr_A$ is the apriori probability set to 0.6 by default. Finally $bias[u,v]$ is calculated.

$$bias[u,v] = penalty/prob[u] \quad (4)$$

where penalty is set to 100 by default and $prob[u]$ is the product of success probabilities $Pr$ of each channel that have already been added to the path and $Pr[u,v]$. Additionally, **LND** also has a lower limit on $prob[u]$ for each $u$, failing which the node $u$ is not considered to be added to the path.
If the payment fails, the sender is able to track at which channel the failure occurred and marks the time of failure for future reference and calculation of $bias$. A new path is tried until all paths between the sender and recipient have been attempted.
It is clear from Equation 2 that **LND** gives higher importance to paths having lower fees than paths having lower locktimes as the locktime parameter is heavily scaled down by the parameter $rf$.

2) **c-Lightning**: Similarly, **c-Lightning** also uses Dijkstra's shortest path algorithm with the cost function again depending on the fee and the delay but it multiplies the two factors rather than adds them. Furthermore, c-lightning introduces some randomness in the path selection, which decreases both predictability and the probability of selecting a route that leads to failure repeatedly. The randomness is introduced via a parameter $fuzz$, set to 0.05 by default, and then computes a scaling factor, $scale = 1 + random(-fuzz, fuzz)$.
Randomization is based on the computation of a one-way 64-bit siphash. In addition, there is a bias against long paths. Then, the cost function is

$$cost[u,v] = (amt[u,v] + scale * fee[u,v]) * lt[u,v] * rf + bias \quad (5)$$

where $rf$ and $bias$ are set to 10 and 1 initially.
If the path resulting from the first search exceeds the maximal permissible path length of 20, then $rf$ is set to a value close to 0, which essentially turns the search into a shortest path algorithm. A binary search with parameter $bias$ is executed that aims to find a $bias$ that entails a route of permissible length. As the diameter of Lightning is much lower than the maximal permissible length, it is highly unlikely that the adjustments of $riskFactor$ and $bias$ will ever be executed in practice. Failed attempts do not entail automatic retries.
From Equation 5, we can see that **c-Lightning** gives high preference to paths with low locktimes as compared to low fees.

3) **Eclair**: In addition to the fee, **Eclair** considers the locktime, capacity and the age $age(u,v)$ of a channel (in blocks) as parameters that impact its cost. The algorithm first normalizes the locktime, capacity, and the age by expressing them in relation to their minimal and maximal values. Concretely, the algorithm computes $n_{lt}, n_{cap}, n_{age}$ as the corresponding normalized values[5] between the maximum and minimum admissible values for $lt$ ,$cap$ and $age$, respectively. Eclair further utilize default weights $lt_{ratio} = 0.15$, $cap_{ratio} = 0.5$, and $age_{ratio} = 0.35$ for the impact of locktime, capacity, and age, respectively. The complete cost function of eclair then is

$$cost[u,v] = fee[u,v] \cdot (n_{lt}[u,v] \cdot lt_{ratio} + (1 - n_{cap}[u,v]) \cdot cap_{ratio} + n_{age}[u,v] \cdot age_{ratio}). \quad (6)$$

Randomness is introduced by not necessarily choosing the path of the lowest cost. Rather, it chooses among the $k$-cheapest paths based on Yen's $k$ shortest paths algorithm. The default value of $k$ is set to be 3. Additionally, every payment is retried 5 times by default until it succeeds.
In a traditional Yen's algorithm, the $i^{th}$ best path is used to compute the $i + 1^{th}$ best path. This is done by setting each node starting from the source to the node before the target as the *spur-node* and finding alternate paths from the spur-node to the target. In Eclair, the spur-node was originally chosen starting from the

[5]The normalized value $n_D(v)$ of a real number $v$ within range D is computed as $(v - \min D)/(\max D - \min D)$

sender ending with the node next to the recipient in the previous best path found similar to the traditional algorithm. Recent code changes [6] modified this approach to change the order of choosing the spur-node by starting from the recipient and ending with the node next to the sender. This is an important modification since the direction followed in the Yen's algorithm should match the direction followed in the Dijkstra's algorithm for better tracking of already explored spur paths which improves performance (especially tail latency).

From Equation 6, we can see that the cost function of **Eclair** unlike **LND** and **c-Lightning** considers capacity and age of a channel in addition to the fee and delay. Moreover, the default values of $lt_{ratio}$, $cap_{ratio}$ and $age_{ratio}$ indicate that more importance is given to the capacity and age parameters in comparison to the locktime parameter with the fee parameter given the most importance.

While we can intuitively speculate that **c-Lightning** would perform best in terms of locktimes and **Eclair** would be effective in terms of low fees, no such speculations can be made on the clients that achieve lower path lengths or the clients having higher success rate by just looking at the cost functions. We evaluate the effectiveness of each client in terms of locktime, fees, path lengths and success rates in more detail in Section III.

## III. Evaluation

In this section, we first describe the simulation model and metrics in Section III-A followed by the datasets and parameters used in Section III-B and finally the results in Section III-C

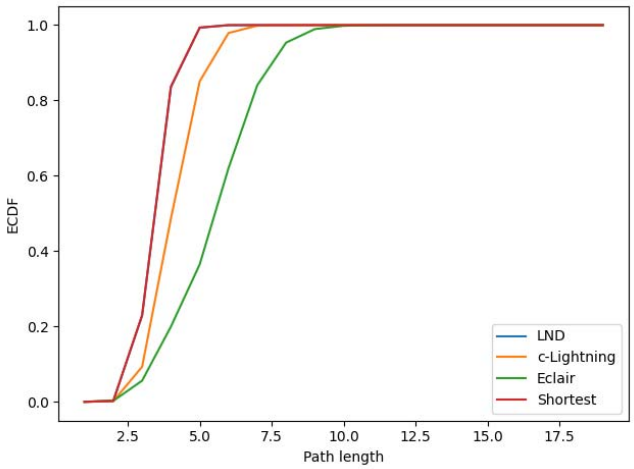
### A. Simulation Model and Metrics

For simplicity, we evaluate each routing client separately on the same set of transactions and the same initial network configuration with the assumption that each node in the network uses the same routing client. A transaction is simulated by first finding a suitable path depending on the routing client used. Once a path is found, the channel balances of all parties involved in the payment are changed accordingly along with paying intermediaries routing fees. A payment is considered a failure if no suitable path is found or any of the intermediaries have insufficient balances.

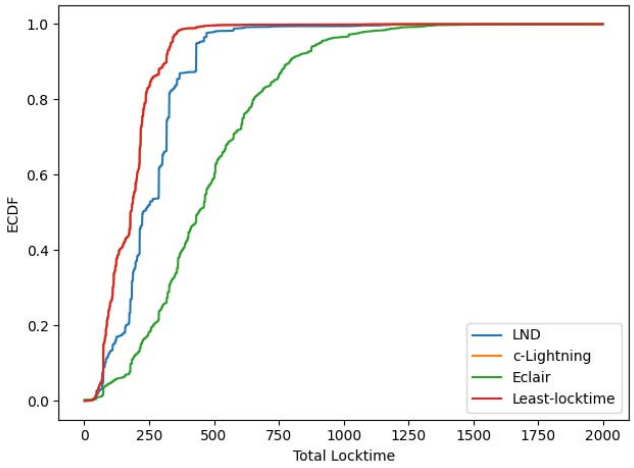
The performance of the different Lightning clients is evaluated based on the following metrics :

1) Path length $L$ and its deviation from the shortest possible path $\delta L$.
2) Total Locktime $D$ for the sender in blocks and its deviation from the path with the least locktime $\delta D$.
3) Fee ratio $F$, that is the ratio of the amount paid including fees to the actual transaction amount, and its deviation from the cheapest path $\delta F$.

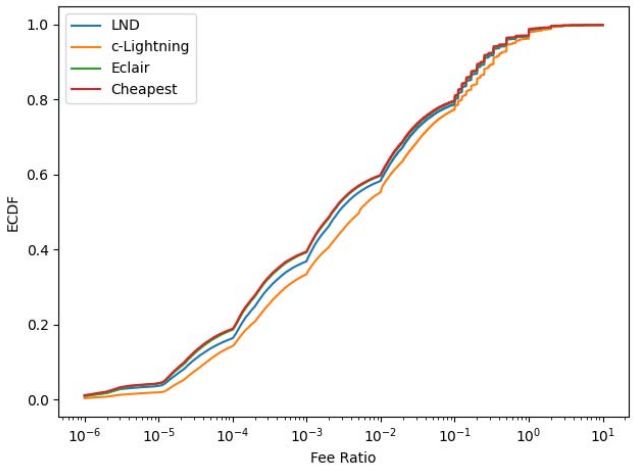
[6]https://github.com/ACINQ/eclair/commit/75cb777c61e49afcc9351feeb4114268da9792bf#diff-bf37f521a1042929f59bba1b27dea85a2f39fb495316342e19df076a3f197e90




(a) Path Length




(b) Total locktime




(c) Fee Ratio

Fig. 1: Cumulative distribution of path length, locktime and fee ratio for transactions on the snapshot

| *Client* | **LND** | **c-Lightning** | **Eclair** |
|---|---|---|---|
| $\delta L$ | **0.0025**±0.0011 | 0.6531±0.0043 | 2.038±0.0166 |
| $\delta D$ | 77.29±1.244 | **0.2778**±0.0344 | 293.7±1.963 |
| $\delta F$ | 0.0123±0.0013 | 0.0328±0.0014 | **0.0045**±0.0006 |
| $SR$ | **0.9998**±0.0001 | 0.9002±0.0079 | 0.6628±0.0332 |

TABLE I: Deviation of path length, total locktime and fee ratio as well as the success rate of each client.

| *Client* | **shortest path** | **least locktime path** | **cheapest path** |
|---|---|---|---|
| $\delta L$ | ± | 0.6549±0.0054 | 0.4088±0.0071 |
| $\delta D$ | 110.8±1.490 | ± | 146.3±1.992 |
| $\delta F$ | 0.1375±0.0625 | 0.3226±0.2437 | ± |
| $SR$ | 0.9435±0.0023 | 0.9053±0.0083 | 0.9055±0.0063 |

TABLE II: Deviation of path length, total locktime and fee ratio as well as the success rate of the protocols optimizing each of the above metrics.

4) Success Ratio $SR$, that is the proportion of transactions that were successful.

All simulations were executed using *Python 3*. We used the networkx module to handle graph related operations. The source code can be found on GitHub [7]

### B. Datasets and parameters

We obtained a snapshot of Lightning[8] containing 4791 nodes and 27994 channels and used it as the PCN graph for our simulations. The parameters $lt$, $cap$, $bf$, $fr$ and $age$ for each channel were obtained from the snapshot. We used the total capacities obtained from the snapshot for each channel. The initial balance distribution of a channel was set by randomly dividing the total capacity between the two nodes of the channel.

We simulate 10 sets of transactions, each set containing 10000 transactions. The transactions are simulated sequentially and the transaction amounts are exponentially distributed between 1 and 100000 satoshis. The sender and recipient for each transaction are chosen randomly using a uniform distribution for all nodes in the network.

### C. Results

We first show the emperical cumulative distribution function of the metrics *path length*, *total locktime* and *fee ratio* after averaging the probabilities for each value over the 10 sets of transactions.

From Figure 1a, we can see that **LND** performs the best in terms of path length and **Eclair** performs the worst. From Figure 1b, it is evident that c-Lightning is the most effective in choosing paths with low locktimes while again **Eclair** performs the worst of the three. We can attribute this trend to the higher preference given by **c-Lightning's** cost function in Equation 5 as compared to **Eclair's** cost function in Equation 6. **Eclair** does give maximum preference to the channel fees and this is reflected in Figure 1c, with **Eclair** performing the best in terms of the fee ratio and **c-Lightning** performing the worst.

We also summarize the values of deviations of the above metrics as compared to the best possible paths with respect to each metric. Each entry in Table I represents the mean and standard deviation over the 10 sets of transactions highlighting the best performing client for each metric. Table I reinforces our observations from Figure 1. In addition, we also observe that **LND** achieves the highest success rate while **Eclair** achieves the lowest success rate.

Looking at the above results, one might consider using a simple protocol that optimizes only one metric. It is then necessary to evaluate the said simple protocol in terms of the other metrics as well. To that end, we measure the effectiveness of the protocols optimizing each metric **shortest path**, **least locktime path** and **cheapest path** with respect to the metrics that they do not optimize. From Table II, we can observe that the path length is not greatly increased by the protocols optimizing the other metrics. However, this is not the case for the total locktime and fee ratio metrics. The success ratio of the three protocols are similar with the **shortest path** protocol slightly outperforming the remaining two.

We can now compare Lightning's routing clients with the simpler protocols optimizing each metric. While **LND** slightly increases the path length in comparison to **shortest path**, the former is clearly better in terms of total locktime, fee ratio and the *success rate*. A similar comparison can be made between **c-Lightning** and **least locktime path**. While the former slightly worse in terms of locktime and both are similar in terms of path length and success ratio, it is much better than the latter in terms of fees. **Eclair** however is worse than **cheapest path** in all aspects. Thus, we discourage the use of **Eclair** for users who desire a protocol that performs well in terms of the metrics discussed here. It might be the case that **Eclair** has better privacy, which however needs a seperate evaluation. We recommend the usage of **LND** for short paths and **c-Lightning** for low total locktimes instead of using **shortest path** and **least locktime path** respectively. For users desiring a protocol that promises a high success rate, we recommend **LND**.

## IV. Related work

Off-chain networks have been studied with great interest recently. The survey by Gudgeon et al. [4] provides an understanding of off-chain networks including payment channel networks and highlights avenues for future work. With regard to Lightning, there have been several recent studies that address its topological properties [8], [15], [16], [19], its susceptibility to channel exhaustion and denial-of-service

[7]https://github.com/SatwikPrabhu/Comparison-of-Lightning-s-clients
[8]https://ln.bigsun.xyz

attacks [9], [15], [19], the confidentiality of channel balances and privacy [7], [11], [18], the anonymity of transaction endpoints [7], [11] and the profitability and economy of fees in Lightning [2], [3]. Zabka et al. [23] provide an approach to identify the routing client used by a majority of the nodes in Lightning. Tochner et al. [19] evaluate the three Lightning clients based on transaction fees and path length. However, this study is only limited for transactions of a fixed size (1000 satoshis).

Several recent works [12] [17] [13] [1] propose a switch from single-path to multi-path payments and suggest different protocols of doing the same. Bagaria et al. [1] propose *Boomerang*, a novel way of using redundant payments on top of existing multi-path routing schemes to reduce latency free of counterparty risk. They use shortest paths for constructing the routes. Sivaraman et al. [17] propose *Spider* a more dynamic approach to route selection with the channel cost changing over time based on the imbalance of the channel balance and the congestion on the channel. The rate of sending payments over a particular path will then depend on the cost of all the channels on the path.

## V. Conclusion

We compared the three routing clients **LND**, **c-Lightning** and **Eclair** in terms of their effectiveness with respect to the path length, total locktime, fee ratio and success rate. We found that each client heavily compromises on at least one of the four metrics. While **LND** compromises in terms of locktime and fee ratio, it is near optimal in path length and success rate. **c-Lightning** and **Eclair** are near optimal in total locktime and fee ratio respectively while compromising on the rest of the metrics.

All the above results can be attributed to how the cost functions of each client are constructed, thus raising the question on whether better cost functions can be constructed. Recent works on designing routing protocols [17] [1] give importance to reducing latency and avoiding congestion on channels. For instance, Boomerang [1] uses shortest paths instead of a more sophisticated path finding protocol. Our results indicates that properties of **LND**'s cost function such as taking recent failures into consideration had a positive impact on success ratio in contrast to others such as shortest paths. We hence suggest to incorporate these lessons learned into the design of multi-path routing as well, which is currently not done.

## Ackowledgement

The work was partially supported by Ripple's University Blockchain Research Initiative and the Distributed ASCI Supercomputer (DAS) was used for experiments.

## References

[1] Vivek Bagaria, Joachim Neu, and David Tse. Boomerang: Redundancy improves latency and throughput in payment-channel networks. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 304–324, Cham, 2020. Springer International Publishing.

[2] Ferenc Béres, István A. Seres, and András A. Benczúr. (v1) a cryptoeconomic traffic analysis of bitcoin's lightning network. *Cryptoeconomic Systems*, 6 2020.

[3] Oguzhan Ersoy, Stefanie Roos, and Zekeriya Erkin. How to profit from payment channels. In *Financial Cryptography and Data Security*, 2020.

[4] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Off the chain transactions. In *Financial Cryptography and Data Security (FC)*, 2020.

[5] Jona Harris and Aviv Zohar. Flood & loot: A systemic attack on the lightning network. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, page 202–213. Association for Computing Machinery, 2020.

[6] Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. On the difficulty of hiding the balance of lightning network channels. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 602–612, New York, NY, USA, 2019. Association for Computing Machinery.

[7] George Kappos, Haaroon Yousaf, Ania Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the lightning network. *arXiv preprint arXiv:2003.12470*, 2020.

[8] Jian-Hong Lin, Kevin Primicerio, Tiziano Squartini, Christian Decker, and Claudio J Tessone. Lightning network: a second path towards centralisation of the bitcoin economy. *New Journal of Physics*, 22(8):083022, Aug 2020.

[9] Ayelet Mizrahi and A. Zohar. Congestion attacks in payment channel networks. *ArXiv*, abs/2002.06564, 2020.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at: https://bitcoin.org/bitcoin.pdf.

[11] Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. Toward active and passive confidentiality attacks on cryptocurrency off-chain networks. *arXiv preprint arXiv:2003.00003*, 2020.

[12] Olaoluwa Osuntokun. Amp: Atomic multi-path payments over lightning. 2018. Available at: https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html.

[13] Dmytro Piatkivskyi and Mariusz Nowostawski. Split payments in payment networks. In Joaquin Garcia-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 67–75, Cham, 2018. Springer International Publishing.

[14] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: scalable off-chain instant payments, 2016. Available at: https://lightning.network/lightning-network-paper.pdf.

[15] E. Rohrer, J. Malliaris, and F. Tschorsch. Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2019.

[16] István András Seres, László Gulyás, Dániel A. Nagy, and Péter Burcsi. Topological analysis of bitcoin's lightning network. In *Mathematical Research for Blockchain Economy*, 2020.

[17] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia Fanti, and Mohammad Alizadeh. High throughput cryptocurrency routing in payment channel networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 777–796, Santa Clara, CA, February 2020. USENIX Association.

[18] Sergei Tikhomirov, Rene Pickhardt, Alex Biryukov, and Mariusz Nowostawski. Probing channel balances in the lightning network. *arXiv preprint arXiv:2004.00333*, 2020.

[19] Saar Tochner, Stefan Schmid, and Aviv Zohar. Hijacking routes in payment channel networks: A predictability tradeoff. *arXiv preprint arXiv:1909.06890*, 2019.

[20] Bitcoin Wiki. Hashed timelock contracts, 2019. Available at: https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts.

[21] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[22] Jin Y Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 1970.

[23] Philipp Zabka, Klaus-Tycho Förster, Stefan Schmid, and Christian Decker. Node classification and geographical analysisof the lightning cryptocurrency network. In *International Conference on Distributed Computing and Networking*, 2021.