Comparing Lightning Routing Protocols to Routing Protocols with Splitting

Dan Andreescu, Stefanie Roos, Oğuzhan Ersoy
d.andreescu@student.tudelft.nl, s.roos@tudelft.nl, o.ersoy@tudelft.nl

June 28, 2021

Abstract

The largest payment channel network, Bitcoin Lightning, shows a potential alternative to current financial systems, overcoming the scalability limitations of blockchain. Source onion routing is used to route payments, but novel routing protocols claim improved effectiveness by employing strategies not compatible with the current state of affairs. In this paper, we compare the current implementations deployed in the Lightning Network, with a novel routing algorithm with splitting and local routing. We find the latter option to significantly boost the success ratio of payments, while inducing some monetary overhead. Then, we propose future directions of research on Lightning protocols to boost their performance.

1 Introduction

Blockchains such as Bitcoin [1] are promising systems for storing incorruptible data (such as financial transactions), without the need for a central authority. To achieve consensus among all nodes in the network, in a fully distributed fashion, redundant computation and data multiplication are needed. That comes with scalability issues and even ethical considerations (such as pollution generated by this wasteful mechanism). Currently, it takes more than 5 minutes for a Bitcoin transaction to be validated, and less than 10 transactions per second can be processed in total (for comparison, VISA handles tens of thousands per second, and a transaction clears in seconds) [7]. However, the power consumption of the Bitcoin network is greater than that of The Netherlands [2].

Payment channel networks (PCNs) address these issues by proposing a new layer on top of blockchain, which allows nodes to transfer funds amongst each other much more efficiently, by not broadcasting every transaction on-chain. Concretely, two peers open a payment channel (PC) by locking some collateral in a multi-signature wallet. To send payments to each other, they have to agree on a redistribution of the locked collateral (provided there is enough available). They can also make conditional payments, i.e. a payment that is executed if the recipient reveals a certain secret, within a certain time limit (timelock delay). If the timelock expires, the payment fails. The payment value is locked until the payment either clears or fails. When the two peers want to withdraw the funds in the channel, they can agree on the most recent split and each takes their share. In case of a dispute, that is one on the parties wants to close the channel unilaterally, it makes a claim as to what is the most recently agreed split, and then the other party still has some time to prove the claim is invalid, in which case the latter takes all the funds if it does so in the allowed time frame. This mechanism is used to disincentivise unfair claims.

A network of such channels is called a PCN, and it allows for transactions between any two peers. In case they do not have a direct channel, intermediary nodes will route the payment for a fee, making use of the aforementioned conditional payments, i.e. each node on the path makes a conditional payment to the next, until the destination reveals a secret that settles all the intermediary transactions. Disputes

can always be settled on-chain, therefore the guarantee of no honest node losing funds, despite the trustless nature of the network [6].

The Lightning Network (LN) [3] is perhaps the most famous PCN. The Basis of Lightning Technology (BOLT) [4] specification describes various protocols needed for the nodes to interact. BOLT specifies an onion routing [5] protocol for routing payments, which limits the possible implementations to source routing protocols.

LND, Eclair, and C-lightning are the three main implementations of LN nodes. They all use source routing, preferring paths with low fees and short delays, while also including some randomness. Except for uncooperative nodes or failures/delays on other layers, the main reason for an attempted transaction to fail is a channel along its path with insufficient balance. Since nodes only know the balances of their own channels, this problem is difficult to mitigate when doing source routing. However, the aforementioned protocols attempt to do so by considering channel capacity, age, or last-recorded failures to estimate how likely a channel is to carry the payment successfully [8].

Novel routing protocols claim improved effectiveness over LND, Eclair, or C-lightning; many of them suggesting that splitting payments, or making local routing decisions are the driving factors behind these improvements [9, 10, 11, 12, 13].

This project aims to compare one of routing protocols that make use of splitting, namely Interdimensional SpeedyMurmurs (IDSM) to the ones already used in the LN (LND, Eclair, C-lightning).

The next section introduces the four routing protocols we shall compare. We then describe the evaluation setup and result, explaining why splitting payments, as implemented by IDSM, does improve the success ratio, while compromising on fees. Then, we propose future directions of research on how to improve the performance of routing protocols, and conclude our paper.

2 Routing Algorithms

LND, Eclair, C-lightning use source routing. That means the path a transaction shall take is determined by the sender. The main algorithm used is Dijkstra's shortest path. The three protocols differ by their definition of distance, or cost function. Such cost functions generally include monetary cost (i.e. the amount of fees to be paid to intermediaries) and time lock delay (i.e. how long could collateral be stuck in case the payment fails). Since the channel balances are not public, cost functions also try to approximate the comparative likelihood of a channel to be able to carry the payment. Such heuristics include the channel age, time since last failure, and total capacity. To protect against attacks, and to allow for sensible payment retries, randomness is often included in the routing algorithms.

IDSM uses a different approach. This novel routing protocol leverages local information about channel balances to make routing and splitting decisions. It uses a number of spanning trees to determine which hops are more suitable to forward a payment. It does not take fees into account, but all the aforementioned parameters considered by the Lightning protocols are no longer needed, as the local choice of hopping and splitting can avoid an immediate failure, if possible.

Let us first introduce the notation used to describe the three LN protocols.

Let $p_0, p_1, \ldots, p_{n-1}$ be a path of length n, from source to destination, where each p_i represents a node on this path.

Let $amt[i]$ be the amount node $p[i]$ should receive. Note that $amt[n-1]$ is the payment value excluding fees, and the total amount spent by the source is $amt[0]$.

To simplify notation, we introduce $fee[i]$, the fee that intermediary node p_i charges. According to the BOLT, regardless of the routing protocol, we have:

$$amt[i] - amt[i+1] = fee[i] = base[p_i, p_{i+1}] + rate[p_i, p_{i+1}] * amt[i]$$

Where $base[p_i, p_{i+1}]$ is the constant base fee of the channel from p_i to p_{i+1}, and $rate[p_i, p_{i+1}]$ is the rate of the fee that is proportional to the amount that is forwarded.

We further introduce $a[i]$, for each attribute a of the channel between p_i and p_{i+1}. Attributes are properties of channels either publicly available in the LN, or scheme-specific variables computed by a node from past experience.

Let $cost[i]$ be the (partial) cost of the path, from p_i to p_{n-1}, as computed by either of the protocols, where $cost[n-1] = 0$. Therefore, the total cost of the path is $cost[0]$.

2.1 LND

The LND cost function optimizes for low delays and fees, while trying to avoid channels likely to fail. LND uses Dijkstra to find the best path according to the cost function below, and if that fails, repeats the process until no more paths exist. When payment attempt fails, the channel that could not carry it is practically ignored in the retry attempt, as a penalty is added to its cost, which is infinite in the first hour, and then decays exponentially.

$$cost[i] = cost[i+1] + amt[i] * delay[i] * risk + fee[i] + \frac{penalty}{prob[i]}$$

Where the *delay* of a channel is the maximum amount of time collateral can be locked in that channel, in case of a failed transaction. The *risk* is a tweakable parameter, for which the default value is $1.5 * 10^{-8}$. The last term is used to avoid channels that have failed recently by estimating the success probability, and applying a penalty inversely proportional to that. The default *penalty* for a channel estimated to succeed for sure is 100. *prob* is computed as follows:

$$prob[i] = \begin{cases} 0 & \Delta t < 1 \text{ hour} \\ apriori * (1 - \frac{1}{2^{\Delta t}}) & \Delta t \geq 1 \text{ hour} \\ apriori & \text{no previous knowledge} \end{cases}$$

Where Δt is the time since the last failure, and *apriori* is the estimated a priori probability of success, for which the default value is 0.6.

2.2 Eclair

Eclair optimizes for a mix of low fees, delays and channel ages, and high capacities. It uses Yen's algorithm [16] to find the k (default $k = 3$) shortest paths according to the following cost function, then picks one at random, unless there is a direct route, which is preferred in that case. If a payment fails, Eclair just retries a few times (default 5), which can be effective due to the randomness involved.

$$cost[i] = cost[i+1] + fee[i] * (n_{delay}[i] * w_{delay} + (1 - n_{cap}[i]) * w_{cap} + n_{age}[i] * w_{age})$$

where n_{delay}, n_{cap}, n_{age} are the normalized channel delay, capacity, and age. Normalization is a linear mapping from the range of the variable to $[0, 1]$. w_{delay}, w_{cap}, w_{age} are the weights of these parameters, used to compute their weighted arithmetic mean. Their default values are 0.15, 0.5, and 0.35. The age and capacity of a channel are reasonable indicators of the balance. That is quite

intuitive for the capacity, which is the sum of the balances, but the age component accounts for the fact that channels tend to get depleted over time, i.e. all funds lay on one side. That is due to the fact that sender-receiver pairs are not uniformly distributed in practice, as some nodes have an intrinsic inclination towards cash inflow or outflow, e.g. retailers vs customers. Therefore, channels tend to be used asymmetrically.

2.3 C-lightning

C-lightning optimizes for low fees and delays. It uses Dijkstra to compute the shortest path according to the following cost function. If the path exceeds a certain maximum number of hops (default 20), it retries with different *risk* and *bias*. The former becomes close to 0, and the latter becomes the variable for a binary search. This has the effect of quickly converging to the best route according to the original cost function, while also keeping the hopcount below a certain maximum, as the greater the *bias* is compared to the *risk*, the more similar the cost function gets to a (scaled) hopcount.

$$cost[i] = cost[i+1] + (amt[i] + scale * fee[i]) * delay[i] * risk + bias$$

where *scale* is uniformly sampled form $[1 - fuzz, 1 + fuzz]$, with a default $fuzz$ of 0.05. This induces some randomness that aids in making the paths less predictable (thus less prone to attacks), while allowing the protocol to retry a payment with an non-zero chance of failing again. *risk* and *bias* are tweakable parameters for which the default values are 10 and 1.

2.4 Interdimensional Speedy Murmurs

Since local routing is used in IDSM, as opposed to the LN protocols, we shall formulate the problem differently. Instead of asking what the best path is (according to some cost), we shall ask how will an intermediary node forward a payment. Several options are presented for IDSM, and the problem is split into two: finding candidate next hops that are closer to the receiver, and splitting the payment value across them. Let us first introduced the notation used to describe these algorithms.

Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be a strongly connected graph, that represents the Payment Channel Network, with $\mathbb{V}$ a set of nodes, and $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ a set of edges.

Let $paths \colon \mathbb{V} \times \mathbb{V} \times \mathcal{P}(\mathbb{E}) \to \bigcup_{n \in \mathbb{N}} \mathbb{V}^n$, where $\mathcal{P}(\mathbb{E})$ is the power set of $\mathbb{E}$, be the set of all paths from a source to a destination, considering only a subset of $\mathbb{G}$'s edges.

$$paths(u, v, E) = \{\mathbf{p} \in \mathbb{V}^n \mid n \in \mathbb{N} \wedge p_0 = u \wedge p_{n-1} = v \wedge (p_i, p_{i+1}) \in E, \forall i \in [n-2]\}$$

Let $bfs \colon \mathbb{V} \to \mathcal{P}(\mathbb{E})$ be function that maps a node of $\mathbb{G}$ to a subset of $\mathbb{G}$'s edges, as selected by a breadth-first search algorithm (BFS) from that node to all the other nodes. We denote the dimensionality of $\mathbf{p}$ as $dim(\mathbf{p}) = n$, s.t. $\mathbf{p} \in \mathbb{V}^n$.

$$bfs(u) = \{(p_i, p_{i+1}) \mid i \in [dim(\mathbf{p}) - 2] \wedge \mathbf{p} = \underset{paths(u,v,\mathbb{E})}{\arg\min} \; dim \wedge v \in \mathbb{V}\}$$

Therefore, $Trees = \{bfs(v) \mid v \in \mathbb{V}\}$ is the set of all BFS spanning trees of $\mathbb{G}$, represented by the subset of $\mathbb{G}$'s edges that are part of the spanning tree. A spanning tree has the property that there is only one path between any two nodes. A BFS spanning tree also ensures that the path from any node to the root is minimal.

We denote the set of all subsets of size k of $Trees$ as $Trees^{[k]} = \{T \subseteq Trees \mid |T| = k\}$.

Let $hop\colon \mathbb{V} \times \mathbb{V} \times \mathcal{P}(\mathbb{E}) \to \mathbb{N}$, be the hop distance function between any two nodes in $\mathbb{G}$, considering only a subset of $\mathbb{G}$'s edges. That is the minimum number of edges over all possible paths between the source and destination, where a path is a sequence of nodes, such that there is an edge between every node and the next one in the sequence.

$$hop(u, v, E) = \min_{\mathbf{p} \in paths(u,v,E)} dim(\mathbf{p}) - 1$$

Let u be a node that has to send or forward a payment of size amt to node d, the destination of the payment. Let $Excl$ be the set of nodes that have seen this payment before. Note that $Excl = \emptyset$ if u is the sender of the payment. If u is an intermediary node, $Excl$ contains all nodes on the path the payment took to reach u, including the sender, but excluding u.

Let us now dive into how u chooses a subset of its neighbours suitable to carry the payment, and how it splits the payment amt, according to the IDSM protocol.

Remember that every node has a global view of $\mathbb{G}$. In the initial phase of the protocol, or whenever the topology of $\mathbb{G}$ changes, u computes a set of k BFS spanning trees with random roots. We call that $KST \in Trees^{[k]}$.

Let $sm_k\colon \mathbb{V} \times \mathbb{V} \to \mathbb{N}$ be the Interdimensional SpeedyMurmurs distance function. It is the minimum hop distance across the k precomputed spanning trees. We shall only use this distance function in our simulations, as it is the one that performs the best in terms of success ratio [12].

$$sm_k(u, v) = \min_{ST \in KST} hop(u, v, ST)$$

Node u has to compute P, a subset of its neighbours which are potential candidates to forward a payment. A neighbour is suitable to forward if it is closer to the destination, and has not been visited before (to avoid loops).

$$P = \{v \mid (u, v) \in \mathbb{E} \wedge v \notin Excl \wedge sm_k(v, d) < sm_k(u, d)\}$$

Let us consider $p_0, p_1, \ldots, p_{n-1}$, an ordered sequence of the elements in P. The node u will split the amount it needs to forward amt across its n candidate next hops by sending the partial amount $split[i]$ to each p_i. Let $cap[i]$ be the unidirectional capacity (or balance) of the channel between u and p_i. The following two conditions need to hold.

$$split[i] \leq cap[i] \text{ for } i \in [n-1]$$

$$\sum_{i=0}^{n-1} split[i] = amt$$

The first condition ensures no payment attempt guaranteed to fail due to insufficient balance is made. The second condition ensures that cash inflow (or sent amount, if u is the sender) equals the cash outflow.

The splitting algorithm, for a certain order of the potential next hops, works as follows. It greedily assigns as much of the amt as possible to the first candidate, and then proceeds to the next ones in

the same fashion, until all the *amt* is assigned. If the last condition is not met, the payment fails, due to insufficient balance.

$$split[i] = min(cap[i], amt - \sum_{0 \leq k < i} split[k])$$

If we sort the candidate next hops before splitting by their (IDSM) distance to the receiver, that is $sm_k(p_i, d) \leq sm_k(p_{i+1}, d)$ for $i \in [0, n-2]$, we have the 'Split-by-Distance' protocol. If we sort by decreasing capacity, that is $cap[i] \geq cap[i+1]$ for $i \in [0, n-2]$, we have the 'Split-if-Necessary' protocol. We shall only evaluate the first protocol in our simulations, as it is the one that performs the best in terms of success ratio [12].

3 Evaluation

To evaluate the protocols, we extend a network simulator, generate various scenarios, and simulate them to produce metrics of interest. In our implementation, the parameters for the LN protocols were the default ones. For IDSM, we chose the number of spanning trees $k = 5$, as the improvement for higher values is not that significant. The splitting method we used was 'Split-closest', as that has been reported to be the most effective in terms of success ration. Comparisons of the SM variations can be found here [12]. It is out of the scope of our paper to investigate the effects of these variations. Our code can be found at https://github.com/dandreescu/PaymentRouting. We observe our result are in line with related work [8]. Although we could only find one paper comparing the LN protocols, and the experiments are different, the correspondence in results gives us confidence that our implementation of the protocols is accurate.

3.1 Simulator

GTNA [14] is a Java framework for graph-theoretical network analysis. It is used in the evaluation of IDSM [12]. We extend the IDSM simulator to support the three LN routing protocols. A PCN is represented as a graph, with edges carrying various parameters. Nodes do not send messages nor have any other functionality but to merely represent the network topology compactly. This simplification allows for reasonably large networks (comparable to the real LN), and workloads up to millions of transactions, to be simulated on a laptop, in a matter of hours. This does not compromise on the fidelity of the evaluations, as the routing protocols are not affected by other aspects, such as communication protocols, up to the computational latency that we consider negligible compared to the network delay.

Three types of scenarios are available in the simulator: Static, Dynamic, and Concurrent. The static one routes transactions one by one, resetting the channel balances after each transaction. This is not realistic for measuring the long term success ratio, but useful for estimating the success probability of a transaction. The dynamic scenario is a good choice for analysing the network over time, but still not realistic for heavy traffic workloads. The concurrent scenario allows for transactions to be routed at the same time, which is useful for studying the impact of congestion in the network. For our simulation, we chose the static scenario, as we want to highlight the differences in success ratio and monetary cost, but not necessarily how the protocols affect the network over time, or how well they deal with high concurrency.

3.2 Topologies

Topologies for network simulations can be either synthetic, or come from real world data. The LN topology is public, and most features are known (except the channel balances, as they are kept secret for privacy reasons), so that makes it a great candidate for our simulation. Synthetic topologies can

be created to resemble naturally occurring network topologies, and offer more versatility in studying how routing protocols behave in various scenarios.

As the three LN protocols were designed specifically for this network, we chose to run our experiments on real world data. We first took a snapshot of the LN topology. We then filtered out nodes and edges for which the necessary parameters were not available. We did that because we have found no suitable workaround to missing data. We further selected only the largest strongly connected component, as the other components were made up of only pairs of nodes, disconnected from the rest of the networks, so it made no sense to consider them for routing purposes.

3.3 Workloads

A workload is the time series of transactions, along with their value, sender, and receiver. Similarly to topologies, workloads can also be synthetic or come from real world data. Some real world data is available for the transaction amount distribution, such as the Ripple dataset [15], but not for the LN. Frequently used models for synthetic data include exponential distributions of the payment amount, as these represent many payments of small value, and few payments of large value, which is intuitively in line with expected customer behaviour. Choosing the sender-receiver pairs can be done by uniform sampling, but the argument can be made that it is not realistic, as some users have an inherent preference for either positive or negative demand (i.e. businesses and consumers). The correlations between demand, outgoing channel capacities, transaction value, and rate of payment attempts, are also not trivial, but out of the scope of this paper.

As we could not find suitable datasets for LN transactions, we synthesized 100000 transactions as follows. For the sender and receiver, we uniformly sampled the set of nodes in the graph. The payment value was exponentially distributed with mean 100000 satoshis, but upper bounded by the maximum flow from the sender to the receiver. The upper bound ensures that we do not attempt to route payments that are impossible to route even by a theoretically optimal protocol.

3.4 Metrics

Many relevant metrics are used to quantify the performance of routing algorithms. These include success ratio, time to completion, throughput, path length and so on. They could all provide some meaningful insights, but we shall focus only on the success ratio and monetary cost, as the acceptance of novel financial systems clearly depends on their reliability and their cost, while the other metrics are of less importance, or already considered good enough (e.g. time to completion is already orders of magnitude better than for the underlying blockchain).

We measured the success ratio, that is the number of successful payments, divided by the number of attempted payments. We further measured the average monetary cost over all payments, that is the average sum of fees that intermediary nodes would charge per payment. However, the second metric has not been measured accurately. There is no option to include fees in the SM protocols for now, although there is ongoing research into that. Therefore, we assume intermediaries charge no fees when routing a payment. Afterwards, we compute the fees that would be paid to every intermediary for the amount that was routed. This is merely an estimate of the actual fees that would be paid, as the sender would have to account for fees when sending the payment. Because of that, the payment value would be larger to include the fees, and so would the proportional fees charged by intermediaries. If the payment value includes fees, some channels may no longer be able to carry it, so the protocol might pick different paths, and thus pay different fees. For the sake of consistency, we used the same way of estimating fees for the three LN protocols, although we could compute them accurately. We argue this estimate for the monetary cost is realistic enough for comparative purposes.

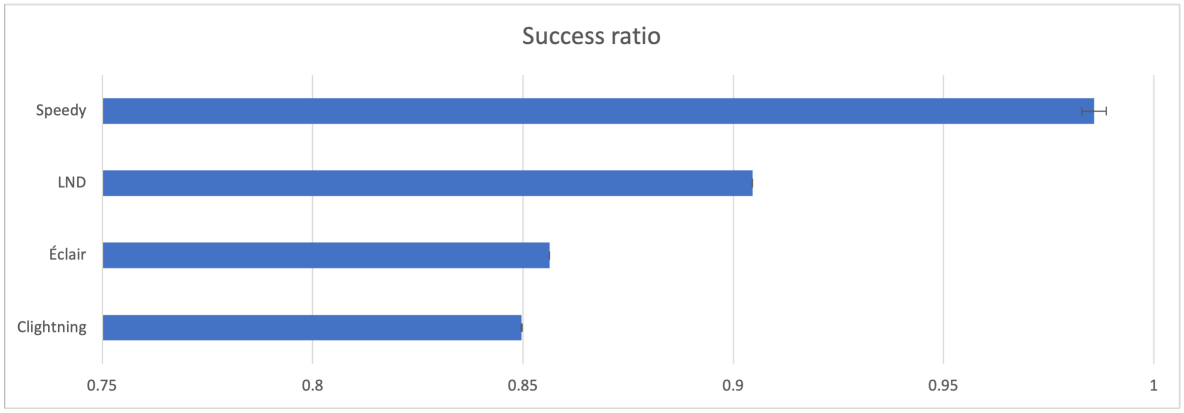

Figure 1: Success ratio

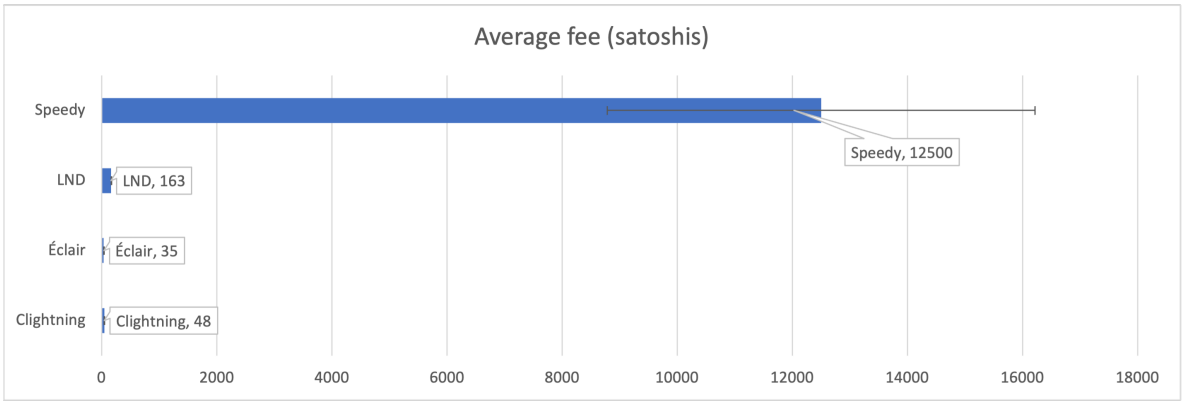

Figure 2: Average fee

3.5 Results

Looking at the three LN protocols, we notice LND outperforms the other two in terms of success ratio, as seen in fig. 1. We believe that is due to its retry mechanism, together with its penalty for channels that failed recently. This allows LND to learn about the channel balances implicitly. Concretely, if a payment fails due to insufficient channel balance, LND will retry the path computation ignoring that channel, as opposed to Eclair and CLightning, for which the retry mechanisms are only based on randomness. Eclair still slightly outperforms CLightning in terms of success ratio, and we believe that is due to the latter one not using the channel capacities in its cost function, as in our static simulation scenario, the capacity is a great indication of the balance. In the real network, we would expect channels to get depleted over time, and thus their capacity to not be such a good indication of the balance, whereas their age could actually make a difference, as opposed to our setup, where the age is not correlated with the balance. When it comes to fees, Eclair is the clear winner, as seen in fig. 2. We attribute that to its cost function giving more weight to fees than the others.

We notice that SM significantly outperforms the three Lightning protocols in terms of success ratio. We argue that is due to splitting and local routing, as it allows for *difficult* payments to be routed more effectively. A *difficult* payment is a payment that is too large for most possible paths to handle. In practice, IDSM deals with such payments by doing a lot of splitting and choosing out of the few (potentially not so straightforward) paths with sufficient balance. However, in terms of monetary cost, it is the other way around. To understand this phenomenon, we repeated the experiment with a

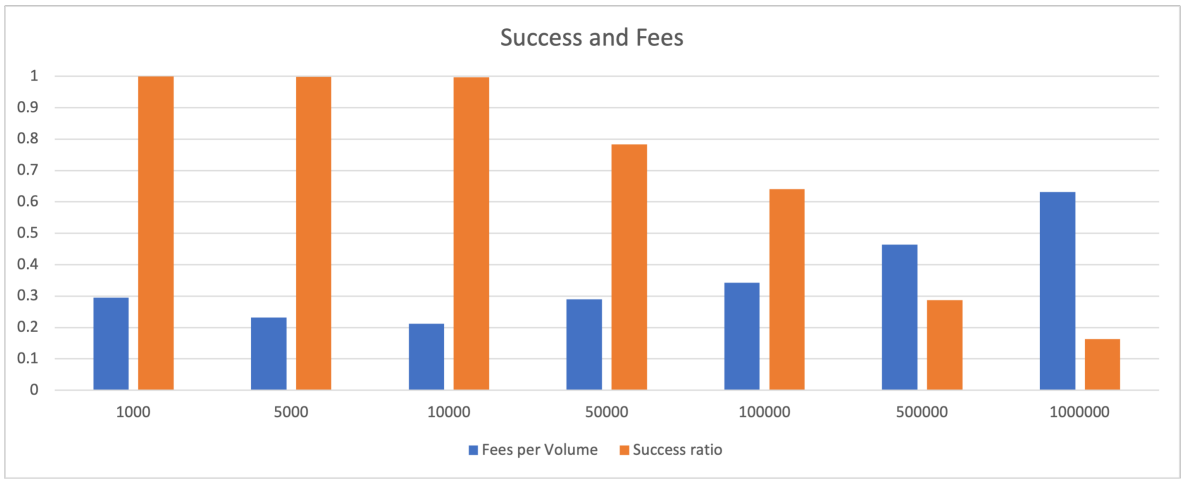

Figure 3: Fees per volume for IDSM

constant distribution of payment values, and varied this parameter. We then plotted the success ratio of SM, as seen in fig. 3, along with the ratio of fees over the success volume. That is the total amount of fees, divided by the total value of successful payments.

As expected, the success ratio is close to 1 for smaller values and then starts decreasing. What is interesting is that the fee rate, although small in the beginning, seems to reach a minimum around the point where the success ratio starts decreasing, and then increases up to over 0.6 for the largest payment value we used. We believe increase in fee rate this also due to the *difficult* payments, as they tend to take much longer paths, which translates to more fees. They also require more splitting, and, as the LN fee protocol also includes a constant base fee for using a channel, along the proportional fee, splitting induces some monetary overhead.

4 Responsible Research

It is of utmost importance to thoroughly address the ethical implications of research, as well as the reproducibility of results, no matter the topic. We shall first touch upon the ethical, societal, environmental and financial considerations of cryptocurrencies, then describe the steps we have taken to ensure the accuracy and reproducibility of our results.

Cryptocurrencies such as Bitcoin could revolutionize the current financial system by providing a fully distributed and highly secure manner to handle monetary transactions. This would eliminate the need for centralized banking systems, and with that, numerous issues such as costly international transfer and retailer fees, or even malevolent actions by central authorities. However, they also pose some concerns, due to the increased anonymity. As opposed to traditional bank accounts, no identification is required to create a Bitcoin wallet and use it to make payments. Although privacy is usually a desirable system feature, after a certain extent, this feature allows bad people to do bad things with money, with little fear of law enforcement. Another potential issue with Bitcoin, and blockchain in general, is the amount of redundant power consumption. The Bitcoin consensus mechanism, needed for all parties to agree on transactions in a trustless fashion, is based on a 'proof-of-work' protocol. That means a lot of 'miner' nodes perform redundant computation to validate transactions. It is estimated that this redundant computation accounts to an energy consumption greater to that on the Netherlands. This economically and environmentally problematic issue can be mitigated by the use of Payment Channel Networks, as they drastically reduce the amount of transactions that need to be validated on the blockchain. Thus, we notice both advantages and drawbacks to the potential mainstream usage of Payment Channel Networks. It is, however, beyond our qualifications to discuss

the extent of regulations that are needed, in order to allow technical progress to improve our lives, while mitigating the risks associated with that, so we leave that to the competent authorities.

Reproducibility of experimental results is crucial to a healthy academia and society in general. People tend to blindly believe numbers and graphs that look plausible. That is a tremendous problem, especially in our times, when information travels so fast and very few people take the time to double-check. Therefore, we take great precautions to facilitate the reproducibility of our result. We do so by documenting our work in extensive detail, with clear, non-ambiguous descriptions of the experimental setup. We further make our code and data public, while making sure to abide by standard coding practices, that would make it understandable and verifiable by an overwhelming majority of computer science bachelor students. We use no other dependencies but Java, which is one of the most widely used programming languages, especially due to its portability. This ensures that the code can be run on any system supporting Java and produce similar results. Since randomness is involved in our experiments, we use a large number of transactions (100000), average the results across 5 runs, and report the 95% confidence interval.

5 Future Recommendations

Since we notice this tremendous tradeoff in terms of fees and success ratio, we suggest further research on this topic. A natural first question regards the distribution of fees for IDSM. Possible experiments include measuring the IDSM fees only for the payments that the LN protocols routed successfully. If our hypothesis about the *difficult* payments is correct, we should notice a significant difference. Another suggestion is including the possibility to pay fees in the IDSM protocol, as it is non-trivial to compute the fee to be added to a payment, if the path is not known. Following that, some fee estimate can be added to the distance function of IDSM, to promote cheaper paths. The other way around in also an interesting research direction: integrating local decisions and splitting into a new LN protocol. That is also non-trivial, due to the source onion routing constraint. One option would be to include up to k possible paths, and allow an intermediary to chose one (or multiple, if splitting). The k paths could be chosen as computed by Eclair, using Yen's shortest paths algorithm. This option would not compromise on fees too much, as Eclair was already willing to choose either one of those. A more aggressive option would be to start with one path, as computed by either of the protocols, and exclude the edge with the lowest capacity before computing the next path, and repeat the process. Instead of the edge with the lowest capacity, other metrics such as LND's last recorded failure could be used to exclude some edges. Hybrid protocols such as these could also dynamically tradeoff between expected success ration and expected monetary overhead in their retry attempts, e.g. first try to optimize fees, while that fails, retry with relaxed fee constraints.

6 Conclusion

In this paper, we presented experimental result that highlights the tradeoff between success ratio and monetary cost for payments routed through a Payment Channel Network. Due to local routing and splitting payments, IDSM significantly outperforms the currently deployed implementations in the Lightning Network in terms of success ratio, and underperforms in terms of monetary cost. We argue the importance of *difficult* payments for both effects: some payments will be just to large to route monetarily efficient through a certain network topology. We also suggest further research directions such as hybrid protocols that could potentially allow a voluntary tradeoffs between expected success ratio and expected monetary cost.

References

[1] https://bitcoin.org/en/

[2] https://cbeci.org/cbeci/comparisons

[3] https://lightning.network

[4] https://github.com/lightningnetwork/lightning-rfc

[5] "Hiding Routing Information," Information Hiding, R. Anderson (editor), Springer-Verlag LLNCS 1174, 1996, pp. 137-150

[6] L Gudgeon, P Moreno-Sanchez, S Roos, P McCorry, A Gervais: SoK: Layer-Two Blockchain Protocols, Financial Cryptography and Data Security 2020

[7] K Croman, C Decker, I Eyal, A Gencer, A Juels, A Kosba, A Miller, P Saxena, E Shi, E Sirer, D Song, R Wattenhofer: On Scaling Decentralized Blockchains (A Position Paper)

[8] S Tochner, A Zohar, A Schmid: Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff

[9] S Dziembowski, P Kedzior: Non-Atomic Payment Splitting in Channel Networks

[10] V Bagaria, J Neu, D Tse: Boomerang: redundancy improves latency and throughput in payment-channel networks, Financial Cryptography and Data Security 2020

[11] M Dong, Q Liang, X Li, J Liu: Celer Network: Bring Internet Scale to Every Blockchain

[12] L Eckey, S Faust, K Hostakova, S Roos: Splitting Payments Locally While Routing Interdimensionally

[13] V Sivaraman, S Venkatakrishnan, K Ruan, P Negi, L Yang, R Mittal, G Fanti, M Alizadeh: High Throughput Cryptocurrency Routing in Payment Channel Networks, 17th USENIX Symposium on Networked Systems Design and Implementation 2020

[14] B Schiller, D Bradler, I Schweizer, M Muhlhauser, T Strufe: GTNA - A Framework for the Graph-Theoretic Network Analysis

[15] S Roos, P Moreno-Sanchez, A Kate, I Goldberg: Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions

[16] J Yen: An algorithm for finding shortest routes from all source nodes to a given destination in general networks, Quarterly of Applied Mathematics, 1970