Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Hubs, Rebalancing and Service Providers in the Lightning Network

MARCO CONOSCENTI[1], ANTONIO VETRÒ[1], AND JUAN CARLOS DE MARTIN.[1]
[1]Nexa Center for Internet & Society, Politecnico di Torino (DAUIN), Italy (e-mail: name.surname@polito.it)
Corresponding author: Marco Conoscenti (e-mail: marco.conoscenti@polito.it).

ABSTRACT Payment channel networks are the most developed proposal to address the well-known issue of blockchain scalability. Currently, the Lightning Network (LN) is the mainstream and most used payment channel network. In a previous work we introduced CLoTH, a payment channel network simulator we developed to analyze capabilities and limitations of such networks. In this work, using CLoTH, we present results of three groups of simulations on a recent snapshot of the LN, aimed to shed a light on the following aspects. Firstly, we investigated how hubs influence the LN performance. Then, we analyzed the effectiveness of two different channel rebalancing approaches, an active and a passive one. Eventually, we studied performance of the LN when a few service-providers nodes receive payments from the other network nodes, which is a typical use case of payment channel networks. We found that the LN is resilient to the removal of hubs, that our passive rebalancing approach reduces of about one fourth the payments failures due to channel unbalancing, and that in the service-providers scenario a consistent part of payments fails because channels directing to the service providers become unbalanced. Our work contributes to prove the strengthen of the Lightning Network when removing hubs and its weakness in the service-provider scenario. In addition, the passive rebalancing approach proposed in this work is a good candidate for the implementation in the Lightning Network protocol to mitigate channel unbalancing.

INDEX TERMS bitcoin; blockchain; blockchain scalability; lightning network; payment channel; payment channel network; simulator.

I. INTRODUCTION

Bitcoin is a cryptocurrency that allows parties to transact without any trusted central entity [1]. It relies on the blockchain, a distributed public ledger which registers all Bitcoin economic transactions. Bitcoin and other blockchain-based cryptocurrencies do not scale. The replication of the ledger and the synchronization mechanism of the replicas, in fact, entail a capped transaction throughput [2].

Payment channels are the most explored technical proposals that address the blockchain scalability issue [3]–[9]. They enable off-chain payments, i.e., payments that do not need to be registered on the blockchain and thus are not subject to the blockchain throughput limit. A payment channel is a two-party ledger which is updated off-chain: the two involved parties can bidirectionally exchange an unbounded number of off-chain payments through the channel. The blockchain is only used to open/close channels and does not register the payments that take place in the channels during their lifetime.

Two-party payment channels can be linked together to build a payment channel network (PCN). This allows parties not directly connected by a payment channel to send/receive off-chain payments which are routed across linked payment channels.

The Lightning Network (LN) [3], namely, the payment channel network built for Bitcoin, is the most developed and used PCN. In fact, at the time of writing, the LN is constituted by more than 9 thousands nodes and more than 30 thousands payment channels, with around 900 bitcoins in the network (being worth more than 9 millions of dollars). The Lightning Network is an HTLC-based PCN. The HTLC (which stands for Hashed Timelock Contract) is a contract that allows the transfer of off-chain payments across multiple payment channels in a trustless way.

At its current state of development, the Lightning Network presents critical features that might undermine its correct functioning and therefore need to be investigated and improved. Some examples of these critical features are channel economic capacity, which limits payment amounts, channel unbalancing, which makes payment channels unusable in one direction, and uncooperative behavior of nodes, which causes

increased payment time and failures.

In a previous work [10], we introduced CLoTH, an original simulator for HTLC payment channel networks [10]. CLoTH takes parameters defining a payment network (e.g., number of channels per node, average channel capacity) and parameters defining payments (e.g., payment amounts and payment rate) as input. On the base of these input parameters, it simulates the payments on the HTLC payment network. It produces performance measures in terms of payment-related statistics, such as payment failure probability and time to complete payments.

In this paper we conducted simulations on the Lightning Network using CLoTH. The specific objective of the study is to analyze the effect of hubs, rebalancing approaches and service providers on the Lightning Network performance. Therefore, we performed the following three separate groups of simulations providing a recent snapshot of the LN as input to CLoTH.

1) Simulations on hubs. In the Lightning Network there are hubs, i.e., nodes which have an high number of open payment channels. We removed the most connected hubs from the LN to understand how hubs influence the LN performance
2) Simulations on rebalancing. We implemented rebalancing approaches, i.e., approaches that aim to mitigate channel unbalancing. We ran simulations in order to analyze the effectiveness of these approaches.
3) Simulations on service providers. The service-providers scenario is a typical case of use of the Lightning Network, in which a few nodes in the network (service providers) are payees only, as they are paid for their services. The majority of network nodes, instead, are payers only, as they send payments to the service providers. The goal of these simulations is to understand whether the LN can support this typical use case in which most of the payments are directed to only a few service-providers nodes.

The main findings resulting from the above-described simulations are:

- The Lightning Network is resilient to the removal of the most connected hubs, as the probability of payment success do not significantly decrease when the hubs are removed.
- A rebalancing approach proposed in this work reduces by one fourth the probability of payment failures for unbalancing.
- In the service-providers scenario, channels directing to the service providers become unbalanced, thus producing a relevant number of payment failures.

The remainder of the paper is organized as follows. In Section II we provide the necessary background on Bitcoin and the Lightning Network. In Section III we briefly present the main features of the CLoTH simulator. In Section IV, we illustrate the study design, describing some exploratory simulations. In Sections V, VI and VII, we describe the simulations on hubs, rebalancing and service providers respectively, including the simulation results. Finally, in Section VIII, we provide conclusions and delineate future work.

II. BACKGROUND

In this Section we provide the background on Bitcoin, the blockchain (and its scalability issue) and the payment channel networks.

A. BITCOIN AND THE BLOCKCHAIN

Bitcoin is a digital payment system running on a peer-to-peer network with no central authority. A Bitcoin transaction allows parties to exchange Bitcoin currency (whose unity is known as bitcoin, symbol BTC). A transaction is made of inputs and outputs. An output contains the bitcoins transferred and a locking script, which codes the conditions necessary to spend the bitcoins contained in the output. An input is a reference to a previous output and an unlocking script that satisfies the spending conditions and spends the bitcoins contained in the referenced output.

To prevent double-spending, all Bitcoin transactions are registered in the blockchain, a distributed public ledger. It is called "blockchain" because its structure is a chain of blocks. A block is a collection of transactions. Each block has a reference to the previous block, thus creating a chain.

Each Bitcoin peer (that runs a full node) stores a copy of the entire blockchain. A distributed consensus protocol is used to keep consistent all the blockchain replicas. The Bitcoin distributed consensus protocol is based on a computational intensive crypto-puzzle and on economic incentives. On average every ten minutes, a Bitcoin peer (called miner) finds a solution to the crypto-puzzle, known as Proof of Work, which grants him the right to add a new block to the blockchain. If the block is valid and contains valid transactions, the miner receives as a reward freshly minted bitcoins; otherwise, the block is rejected by the Bitcoin network. Therefore, economic incentives incentivize miners to produce valid blocks. Such distributed consensus protocol constitutes the key novelty of Bitcoin, as it works in a decentralized environment: each Bitcoin peer, without being trusted and identified, manages the blockchain and validates new transactions.

B. THE ISSUE OF SCALABILITY

In Bitcoin the transaction throughput is capped to a maximum of around seven transactions per seconds [11]. Such transaction throughput is far lower than the thousands of transactions per second which world-wide payment systems are supposed to support [12]. This prevents Bitcoin from scaling.

The transaction throughput is capped by the maximum block size allowed in Bitcoin. The reason of imposing a block size limit is to contain the bandwidth, computational and storage cost of storing the blockchain, validating and transmitting transactions. In this way, the decentralization degree is kept high, as an high number of peers is able to store, validate and transmit blocks and transactions [13].

Different solutions have been proposed to address the scalability issue:

- Re-parametrization of the block parameters, namely, reduction of the inter-block latency and/or increase of the maximum block size. It has been proved [13], [14] that, if an high decentralization degree of the system is an essential requirement, re-parameterization does not significantly increase transaction throughput.
- Alternative consensus protocol (such as Proof of Stake [15], [16]) which aim at replacing the Proof of Work and at increasing the rate at which new blocks of transactions are produced. The security properties of alternative consensus protocol constitute an open research question.
- Sharding [17], [18], which consist in distributing portions of the blockchain to multiple sets of nodes, so that not every node has to store and manage the entire blockchain. Whether sharding approaches preserve blockchain decentralization is an open research question.
- Payment channel networks (PCNs), which solve the scalability issue by moving off-blockchain as many payments as possible. In fact, throughput of off-chain payments is not capped by the maximum block size, as those payments do not need to be stored on the blockchain.

PCNs seem to constitute the most promising solution to the scalability issue, as they could not significantly compromise blockchain security and decentralization. They could enable payments that, with respect to transactions registered on the blockchain, are: cheaper, as lower fees are required than the fees for on-blockchain transactions; faster, as they do not need to be registered on the blockchain; and more privacy preserving, as they are not visible in the public blockchain.

C. PAYMENT CHANNEL NETWORK

A payment channel is a two-party channel in which the parties exchange off-chain payments. A payment channel network is a network of payment channels where off-chain payments are routed. A PCN constitutes a second layer, on top of the blockchain: the latter is only used to open and close payment channels.

The Lightning Network [3] is the mainstream PCN, built on top of the Bitcoin blockchain. The LN protocol specifies how to open payment channels and to route off-chain payments across a network of payment channels.

Assume a payment channel between Alice and Berto. Assume that Alice allocates 0.5 BTC in the channel and Berto allocates 0.5 BTC in the channel: the initial balance of Alice in the channel is therefore 0.5 BTC, and the initial balance of Berto is 0.5 BTC. The total capacity of the channel is the sum of the two balances, i.e., 1 BTC. The operational cycle of the payment channel in the Lightning Network is described in the following.

a: Channel funding

Alice and Berto create a funding transaction, the transaction required to fund the channel. The funding transaction is broadcast to the blockchain and, when confirmed, the channel is considered open. Alice and Berto create also a commitment transaction, which represents their respective current balances in the channel: such transaction spends the bitcoins in the funding transaction and transfers 0.5 BTC to Alice and 0.5 BTC to Berto. The commitment transaction is not broadcast to the blockchain.

b: Payment execution

When Alice wants to pay 0.1 BTC to Berto, the two parties create a new commitment transaction which reflects the new balances: 0.4 BTC of Alice and 0.6 BTC of Berto. No transaction is sent to the blockchain: the operation is done off-chain.

c: Channel closing

When the two parties want to close the channel, they send the last updated commitment transaction to the blockchain.

d: Punishment

Berto can punish Alice if Alice misbehaves (or vice versa), thanks to the Revocable Sequence Maturity Contract (RSMC), a contract implemented by a script in the spending condition of the commitment transaction. Alice could be tempted to broadcast an old commitment to the blockchain, for instance, the commitment preceding her payment to Berto, when she owned more funds (0.5 against 0.4 BTC). If Alice tries to do so and Berto notices the old commitment transaction in the blockchain, Berto can take all the funds of the channel (even the 0.4 BTC of Alice's balance), by broadcasting the last commitment transaction with the RSMC to the blockchain. Therefore, since Alice may lose all her funds, she is disincentivised to misbehave.

e: Unbalancing

A channel is said unbalanced when one balance is much higher than the other. If payments always flow from Alice to Berto and no payments flow from Berto to Alice, Alice's balance may go to zero. Channel unbalancing constitutes a problem, as a payment cannot traverse a channel in a given direction if the balance in that direction is lower than the payment amount.

A network of payment channels allows off-chain payments even to parties not directly connected by a payment channel. The Lightning Network is an HTLC-based PCN, as routing of a payment across multiple channels is done via a specific contract called Hashed Timelock Contract (HTLC). The HTLC ensures trustlessness: a party involved in a payment route is guaranteed it does not lose money, even in case the other parties in the route are not trustworthy and misbehave.

The HTLC implements off-chain conditional payments in a payment channel. It is coded as a spending condition in the

commitment transaction of a payment channel. When Alice establishes an HTLC with Berto of value 0.1 BTC, it means that Alice will pay Berto 0.1 BTC if Berto shows a certain value R (called preimage); otherwise, if Berto does not show R within a certain timeout, the payment does not take place.

In the following we explain the phases of a multihop payment using HTLCs. For instance, assume that Alice sends 0.1 BTC to Davide traversing the following payment channels: the channel between Alice and Berto, the channel between Berto and Carola, the channel between Carola and Davide.

f: HTLC establishment

For the multi-hop payment originated by Alice and trying to reach Davide, the following HTLCs are established: (1) the HTLC between Alice and Berto, where Alice pays 0.1 BTC to Berto if Berto demonstrates to know R within a certain timelock, e.g., three days; (2) the HTLC between Berto and Carola, where Berto pays 0.1 BTC to Carola if Carola demonstrates to know R within two days; and (3) the HTLC between Carola and Davide, where Carola pays 0.1 BTC to Davide if Davide demonstrates to know R within one day.

g: HTLC fulfillment

Alice sends R to Davide, and HTLCs from Davide to Alice are fulfilled: Davide shows R to Carola within one day, and Carola pays him 0.1 BTC; Carola shows R to Berto within two days, and Berto pays her 0.1 BTC; Berto shows R to Alice within three days, and Alice pays him 0.1 BTC. At the end, 0.1 BTC has been transferred from Alice to Davide. For their work of forwarding the payment, Berto and Carola can withhold a small fee when transferring the payment (which, for simplicity, we omitted in this example). It is important to highlight that all the described operations are performed off-chain, without the need to interact with the blockchain. Finally, it is worth noticing that, from Davide to Alice, HTLCs have increasing timelocks. This ensures that each party has enough time to know R and claim her funds: Davide shows R to Carola in one day, Carola pays 0.1 BTC to Davide; after that, Carola still has one day to claim 0.1 BTC from Berto, as the timelock with Berto is set to two days.

h: HTLC failure

In the case R is never revealed, HTLCs must be failed and, after the timelock expiration, funds of the HTLC return to the payer in each channel. If Davide does not reveal R, after one day, when the timelock expires, Carola gets back the funds of the HTLC. This can be done cooperatively with Davide, by creating a new commitment transaction off-chain; instead, if Davide is not cooperative, Carola can send the most recent commitment to the blockchain and close the channel. Then Carola propagates back the failure, so that the HTLCs in all the other involved channels are failed. It is important to notice that, if R is not revealed, funds in the HTLC stay locked up to the timelock expiration.

III. THE CLoTH SIMULATOR

In [10] we introduced CLoTH, the simulator for HTLC payment channel networks we developed[1]. In this Section we recap the main features of the simulator, necessary to understand the remainder of the paper (for the complete details, we refer to our previous paper). Moreover, we list the updates we recently made to the simulator and we illustrate the related work.

CLoTH is a discrete-event simulator. As input, it takes a definition of an HTLC network and a payment script to be played during simulation. It simulates payments in the HTLC network by locally running a discrete-event mapping of `lnd` – one of the implementation of the Lightning Network protocol. It produces performance measures in the form of payment-related statistics (e.g., the probability of payment failures and the mean payment complete time).

The computation flow of the simulator is constituted by three phases, herein briefly described: pre-processing phase, simulation phase, post-processing phase.

A. PRE-PROCESSING PHASE

The pre-processing phase serves for generating the HTLC PCN and the payments which are executed in the network during the simulation phase.

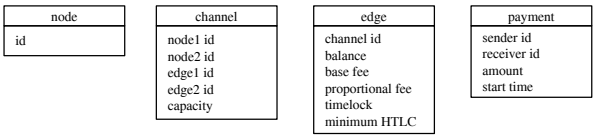

FIGURE 1: CLoTH simulator data structures.

The payment channel network and the payments are represented in the simulator with the data structures in Figure 1. A `channel` connects two `node` (each one represented by an ID) and has a certain capacity. In addition, since a channel is bidirectional, namely, payments can traverse it both from `node1` to `node2` and from `node2` to `node1`, it contains two edges, each one representing a direction of the channel. An `edge` contains: the ID of the `channel` the edge belongs to; the available balance in the direction represented by the edge; base and proportional fee, which constitute the fee required for forwarding a payment in the direction of the edge (proportional fee is the part of the fee which depends on the payment amount, while base fee is the constant fee applied regardless of the payment amount); the timelock set in the HTLCs established in the direction of the edge; and the minimum value allowed for payments forwarded in the direction of the edge. A `payment` is described by a sender, a receiver, the payment amount and the payment start time, that is the instant in which the payment occurs in the simulator.

There are two possible input modes for populating the simulator data structures:

[1]The code of the simulator in pre-alpha is available online at https://researchdata.nexacenter.org/payment-network-simulator/cloth-0.0.2.zip

1) by directly providing a complete specification of each attribute of the data structures;
2) by providing a few input parameters that statistically define an HTLC network and a script of payments to be simulated. In this case, the network and payment generator engines of the simulator generate through random variables an instance of HTLC network and an instance of payment script that match the input description.

TABLE 1: CLoTH simulator input parameters.

Type	Symbol	Definition
Network	N_n	Number of nodes
	N_{ch}	Average number of channels per node
	σ_t	Tuner of network topology
	$P_{\bar{c}_b}$	Uncooperative nodes probability before HTLC establishment
	$P_{\bar{c}_a}$	Uncooperative nodes probability after HTLC establishment
	C_{ch}	Average channel capacity
	G	Gini index of channel capacity
Payments	r_π	Payments per second (off-chain)
	N_π	Number of payments
	σ_a	Tuner of payment amounts
	F_{sr}	Fraction of same-recipient payments

Table 1 shows input parameters of the simulator with their symbols. Here follows an explanation of the parameters used for simulations in this paper:

- r_π is the average number of payments per second. In particular, the payment inter-arrival time is modeled as a negative exponential random distribution.
- σ_a tunes payment amounts. It is the width of the Gaussian distribution whose tail is used to choose the orders of magnitude of payment amounts. The greater the width is, the higher the payment amounts.

B. SIMULATION PHASE

During the simulation phase, the simulator simulates the execution of the input-defined payments in the input-defined HTLC network; in particular, the simulator runs a discrete-event simulation.

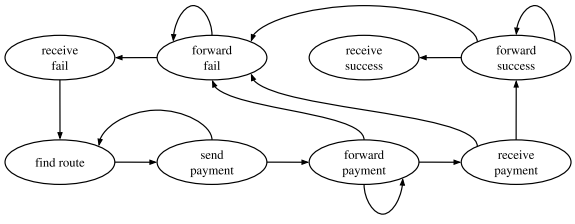

FIGURE 2: CLoTH simulator events state diagram.

In the CLoTH simulator, an event represents a state of a payment. An event is generated each time a payment changes its state according to the state diagram in Figure 2, and it is processed by a function of the same name of the event. For example, the event "find route" represents a payment for which a route has to be found. When such event is extracted from the queue, a function of the same name is called, which searches for a payment route. If the route is found, a "send payment" event is generated for that payment, which represents the state of a payment that has to be sent by the payment sender.

Payments are routed in the network using HTLCs, as specified by the Lightning Network protocol. The simulator functions are a map of the functions of `lnd`, the implementation of the Lightning Network taken as reference for implementing CLoTH: the completeness of the mapping ensures the validity of simulation results.

C. POST-PROCESSING PHASE

At the end of the simulation phase, the simulator produces per-payment data (e.g., the result -failure or success- of a payment, the payment end time). The post-processing phase of the simulator transforms those data into statistically meaningful performance measures, by leveraging the batch means method [19].

Table 2 shows the performance measures produced by the simulator with their symbols. Here follow some clarifications:

- P_{f_r} is the probability that a payment fails due to the absence of a route connecting sender and receiver.
- P_{f_b} is the probability that a payment fails because a channel in the route was unbalanced (namely, its balance was lower than the payment amount) and an alternative route without the unbalanced channel is not found.
- $P_{f_{\bar{c}}}$ is the probability that a payment fails because a node in the route was uncooperative and an alternative route is not found.
- $P_{\bar{t}}$ is the probability that a payment fails because it took more than a timeout of 60 seconds to complete.
- $P_{\bar{k}}$ represents the remaining fraction of payments for which we do not know whether they failed or succeeded, as they ended after the time validity window of our simulation. For example, this category can encompass payments delayed after a long timelock, as a node was uncooperative after establishing the HTLC for the payment.
- T is the mean time for a successful payment to complete.
- N_{att} is the mean number of times a successful payment is re-attempted.
- L_r is the mean route length traversed by a successful payment.

D. CLoTH UPDATED VERSION

The simulations in this paper are run using an updated version of CLoTH with respect to the one used in our previous paper [10]. The updated version reflect a newer version of `lnd`, namely, `lnd-v0.5-beta`. The main changes with respect to the previous version are:

TABLE 2: CLoTH simulator performance measures.

Symbol	Definition
P_s	Probability of payment success
P_{f_r}	Probability of payment failure for no route
P_{f_b}	Probability of payment failure for unbalancing
$P_{f_{\bar{c}}}$	Probability of payment failure for uncooperative nodes
$P_{\bar{t}}$	Probability of payment failure for timeout expiration
$P_{\bar{k}}$	Probability of unknown payments
T	Payment complete time
N_{att}	Number of payment attempts
L_r	Payment route length

- A payment fails if it does not complete within a timeout of 60 seconds.
- Nodes and edges that are blacklisted during a payment attempt (e.g., because an edge has not sufficient balance), are removed from the blacklist after five minutes and five seconds respectively.
- The minimum HTLC channel policy is introduced.
- Dijkstra's algorithm considers also channel fees as distance metric (while in the previous version of `lnd` considered only timelocks).

E. RELATED WORK

Several works in the literature analyzed payment channel networks through simulations.

In [20], the author proposes and evaluates through simulations a routing protocol for payment channel networks. The authors of the Flare routing protocol [9] run simulations with 100,000 nodes to study the performance of the protocol. In [21] Piatkivskyi et al. developed a Lightning Network simulator called Blyskavka to evaluate their approach of splitting payments when routing them in LN. It is a multi-agent discrete-event based simulator built for general purpose payment network simulations; it also simulates HTLCs. In [22], Ruozhou Yu et al. implemented a simulator to evaluate CoinExpress, their proposed payment routing mechanisms for payment channel networks. The simulator is a payment channel network simulator based on network simulator `ns-3`. Stasi et al. [23] developed a simulator to evaluate their proposed improvements to the LN protocol: a novel fee policy that aims at reducing channel unbalancing; and a multipath routing payment scheme. Finally, Reynolds [24] developed ocalm code for basic simulations on LN.

Differently from all the above-mentioned simulators, CLoTH is a complete and precise mapping of the Lightning Network code. The functions of `lnd` implementing the modified version of Dijkstra's algorithm and the HTLC routing are exactly implemented also in CLoTH. This is the key feature of CLoTH which make it different from the other simulators and this ensures the validity of the results produced by the simulator.

IV. STUDY DESIGN

In all the simulations discussed in this paper, a recent snapshot[2] of the Lightning Network is taken into consideration. In this Section we present: the main features of the Lightning Network at that snapshot (Section IV-A); exploratory simulations conducted on that snapshot to establish the interval of the independent variables of the simulations (Section IV-B); the design of the simulations presented in this work (Section IV-C).

In this and in the remaining Sections, only the most relevant simulation results are showed. The complete results are available in Appendix A.

A. FEATURES OF THE LIGHTNING NETWORK

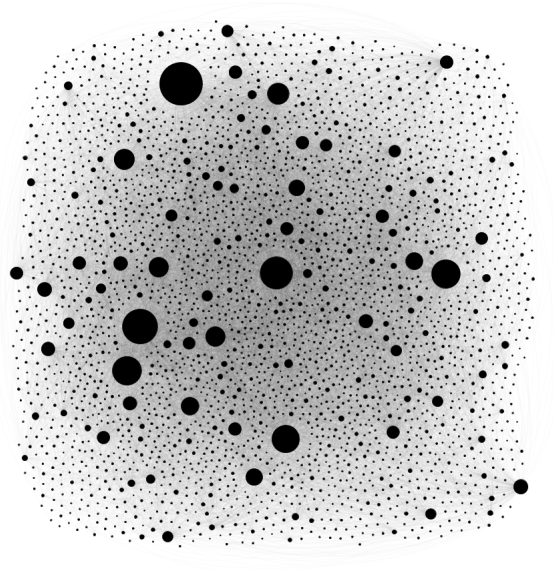

FIGURE 3: A graphical view of the Lightning Network on February 12nd, 2019.

Figure 3 is a graphical view of the Lightning Network on February 12nd, 2019. Size of nodes is directly proportional to their number of open channels. The network presents the following characteristics:

- Number of nodes with at least one open channel: 3148;
- Number of channels: 24683;
- Mean and standard deviation of channel capacity: 0.0267 BTC (mean), 0.04361 BTC (standard deviation).

Table 3 shows the values of the channel policies in the LN. For each policy, the table shows also the default value set when opening a channel with the `lnd` client. Monetary policies are expressed in millisatoshis[3].

[2]Downloaded on February 12nd 2019 at 16:36 CET from https://rompert.com/recksplorer/

[3]1 satoshi corresponds to 10^{-8} bitcoin.

TABLE 3: Channel policies in the Lightning Network.

Policy	Mean	STD	Default value
Minimum HTLC (msat)	1034.79	40300.17	1000
Base fee (msat)	959.58	2504.26	1000
Proportional fee (msat)	671.66	22173.42	1
Timelock (blocks)	134.75	68.29	144

B. SIMULATION INDEPENDENT VARIABLES

This Section presents exploratory simulations conducted on the LN with the goal of defining the variation intervals of the independent variables.

The possible independent variables considered for the simulations of this paper are: payment amounts tuner σ_a; and payment rate r_π. The other simulator input parameters are not considered as independent variables for the following reasons:

- The network parameters are the ones of the snapshot of the Lightning Network.
- The probabilities of uncooperative nodes are fixed to zero, since the uncooperative behavior of nodes is not under observation in the simulations of this work.
- The percentage of same-recipient payment is fixed to zero, since the network behavior in the scenario of payments directed to only a few nodes is studied by simulations in Section VII.

In the following we show the exploratory simulations through which we define the variation intervals of r_π and σ_a.

1) Payment Rate Variation Interval

We conduct exploratory simulations on the Lightning Network varying payment rate from 10 payments per second to 100 payments per second. The payment amounts tuner σ_a is fixed and payment amounts are uniformly distributed between 1 and 10^5 satoshis: the lower limit is the minimum allowed HTLC amount set by default in channel policy, the higher limit is chosen to avoid payments higher than the average channel capacity of the LN.

As results in Figure 4 show, payment success probability is around 88% for each payment rate. As Table 6 in Appendix A shows, the most significant reasons of failures are: mainly insufficient channel capacities for routing payments of the highest amounts; unbalancing of channels. However, it can be noticed that payment rate does not significantly impact performance: payment success probability remain almost the same for each payment rate. Therefore, for the simulations in this work, we use a fixed payment rate of 100 payments per second. We choose this value because it constitutes the middle order of magnitude between 10 payments per second - the Bitcoin transaction rate which the Lightning Network is supposed to overcome - and 1000 payments per second - the payment rate of well-established payment systems that the Lightning Network aspires to reach in the future [12].

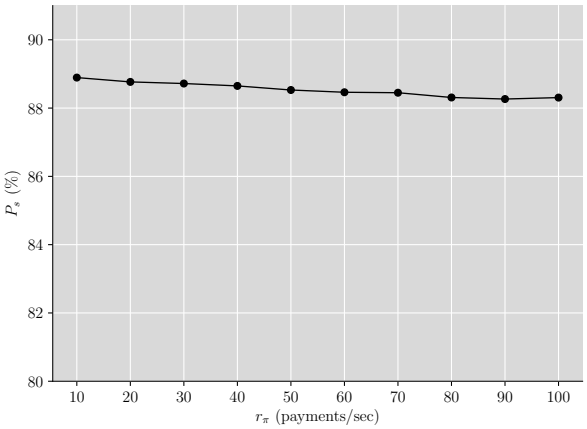

FIGURE 4: Payment success probability varying payment rate r_π.

2) Payment Amounts Variation Interval

The orders of magnitude of payment amounts are chosen between 1 and 10^5 satoshis: the minimum is chosen because the default minimum HTLC policy is 1 satoshi, therefore payments lower than 1 satoshi would almost certainly fail; the maximum is chosen to avoid to produce payments higher than the average channel capacity of the LN.

The amount tuner σ_a sets the distribution of payment amounts in the above-defined orders of magnitude. To define the limits of the variation interval of σ_a, we proceeded as follows: the lower limit of σ_a is set to 1, because this value produces almost all payments of the lowest amount possible (1 satoshi); the upper limit is chosen via simulations on the LN . In these simulations, σ_a is increased until the probability of payment success goes below 90%. This is to avoid to use values of σ_a for which it is already known that they cause low performance. Table 4 shows the distribution of payment amounts for each value of the variation interval of σ_a defined as above.

TABLE 4: Ratio of payments with a certain order of magnitude for each value of σ_a.

Order of Magnitude (Satoshi)	σ_a							
	1	2	3	4	5	6	7	8
10^0	97.44%	65.58%	42.93%	31.42%	26.64%	22.74%	21.12%	20.13%
10^1	2.55%	25.41%	26.97%	24.92%	21.86%	21.16%	19.98%	19.61%
10^2	0.01%	7.05%	16.15%	18.32%	18.50%	17.81%	17.58%	17.61%
10^3	0.0%	1.63%	8.61%	12.29%	15.19%	15.58%	16.08%	15.54%
10^4	0.0%	0.28%	3.82%	8.01%	10.16%	12.90%	13.50%	14.21%
10^5	0.0%	0.05%	1.52%	5.04%	7.65%	9.81%	11.74%	12.90%

Figure 5 shows the results of the above-discussed simulations on the LN varying σ_a. The probability of payment success goes below 90% for $\sigma_a =$ 8.0, so this value is chosen as upper limit of the variation interval. When $\sigma_a = 1$, the probability of payment success is 99.81% and the only reason of payment failures is for absence of route. While increasing σ_a, the failures for no route increase and some

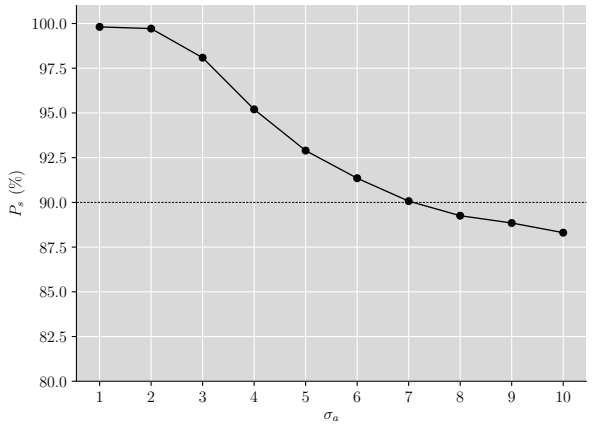

FIGURE 5: Payment success probability varying payment amounts σ_a.

failures are also due to unbalancing and to payment timeout expiration. In general, as Table 7 in Appendix A shows the most occurring reason for failures is the absence of route (around 8%); channel unbalancing, instead, is the cause of around 2% for the highest values of σ_a analyzed. The reason of this behavior is that payments with increasing amounts do not find a route with sufficient channel capacities, and also cause channel unbalancing.

C. SIMULATIONS DESIGN

In this paper we present three groups of simulations conducted on the snapshot of the Lightning Network described in Section IV-A.

1) Simulations on hubs. We remove from the Lightning Network the hubs, i.e., the nodes with the highest number of channels. We run simulations giving the networks without those nodes as input to CLoTH. The goal is to understand whether the Lightning Network is resilient to the removal of the hubs.
2) Simulations on rebalancing. We implement in the simulator two separate approaches aiming at rebalancing channels. We run simulations to study the effectiveness of the approaches at reducing payment failures for unbalanced channels.
3) Simulations on service providers. We simulate a typical use case of the Lightning Network, in which a few service providers receive payments from the remaining network nodes. Therefore in the Lightning Network we configure three classes of nodes: payees only (the service providers), payers only and hybrid payees/payers. We run simulations giving in input to CLoTH the Lightning Network with the classes of nodes.

V. HUBS

In this Section we discuss a set of simulations which analyze the Lightning Network performance when hubs of the network are removed. The goal is to understand how hubs influence the LN.

a: Identification of hubs

In network science, a hub is a node characterized by an high degree [25], i.e., an high number of connections. Specifically, in this work we identify as hubs the first six nodes of the LN ordered by number of channels. Figure 6 is the bar plot of the number of channels per node in decreasing order for the first 50 most connected nodes. We identify the first 6 nodes as hubs: this is justified by the fact that there is the highest step between the sixth and the seventh node (the seventh node has 159 less channels than the sixth). The chosen hubs have in total 4695 channels, that constitute around 20% of the total number of channels in the LN. Those nodes became hubs for different reasons. For instance, some of them incentivized nodes to open a channel with them by promising low-fee channels, or by simplifying the channel opening procedure.

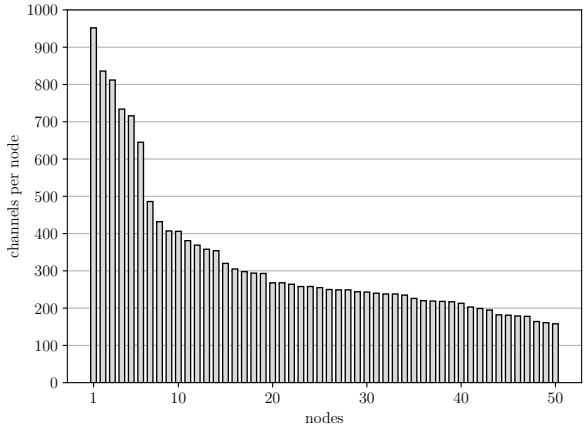

FIGURE 6: Number of channels per node of the first 50 most connected LN nodes.

b: Simulation strategy

The simulation strategy adopted for studying the influence of hubs in the network consists in disconnecting one-by-one the hubs and to run a simulation on the resulting network without those hubs. From the considered snapshot of the LN, the first hub (the one having the highest number of channels) is removed with all its channels and a simulation is run on the network without that hub. Then, from the network without the first hub, the second hub is removed and a simulation is run on the resulting network (namely, the LN lacking of the first and second hub). The same process is repeated until all six hubs are removed.

The only independent variable of these simulations is the number of disconnected hubs, as the objective of the simulations is to analyze their impact on performance. The payment amount parameter σ_a is fixed to 4, that represents the middle of its variation interval defined in Section IV-B.

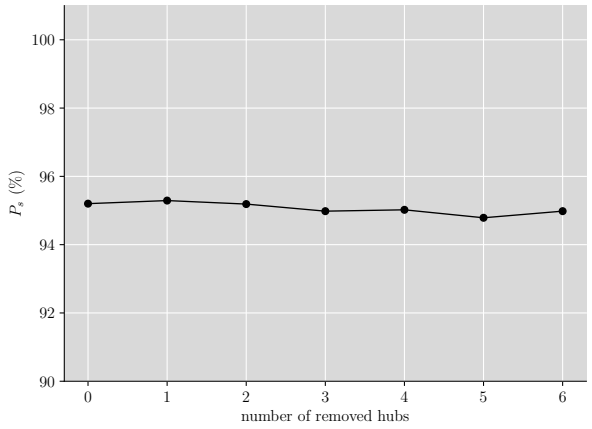

FIGURE 7: Payment success probability when removing hubs in the Lightning Network.

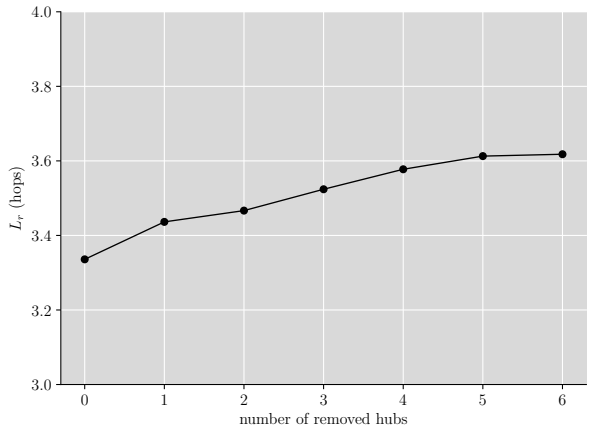

FIGURE 8: Average payment route length when removing hubs in the Lightning Network.

A. SIMULATION RESULTS

Figure 7 shows the probability of payment success when varying the number of hubs removed from the LN. As already mentioned, in each simulation σ_a is fixed to 4 (see Table 4 for the corresponding distribution of payment amounts).

The payment success rate is not significantly affected by the presence of hubs: when all the hubs and their channels are connected, such rate is 95.2%, while when all hubs are disconnected, it is 94.98%, that means a decrease of around 0.2%. Such decrease is entirely due to payments not finding a route.

The most visible -and expected- effect resulting from hubs removal is a slight increase of the average route length, as Figure 8 shows: from 3.34 to 3.62 number of hops.

VI. REBALANCING

In this Section, we describe two different rebalancing approaches, whose objective is to rebalance channels, thus avoiding that payments fail because of unbalanced channels. After implementing the rebalancing approaches in the simulator, we run simulations to understand whether the approaches are effective, namely, whether they succeed in reducing the probability of failures for unbalancing. We design two different rebalancing approaches: active and passive rebalancing.

a: Active rebalancing

The active rebalancing consists in executing a payment with the purpose of rebalancing channels. If a node has a channel with low balance and also another channel with high balance, it makes a rebalancing payment, which transfers funds from its high-balance channel to its low-balance channel.

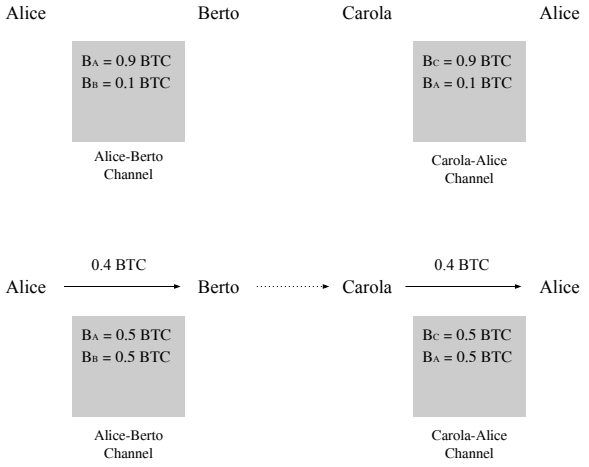

FIGURE 9: Rebalancing payment in the active rebalancing approach.

Consider the situation in Figure 9. In the channel with Carola, Alice has low balance (10% of the total channel capacity), while in the channel with Berto she has high balance (90% of the total channel capacity). Therefore, she executes a rebalancing payment that moves 0.4 BTC from the channel with Berto to the channel with Carola. In this way, the channel with Carola will result balanced, as she and Carola own both 50% of the total channel capacity.

The active rebalancing implemented in the simulator is triggered when a channel balance goes below 20% of the total channel capacity. The amount transferred with the rebalancing payment is the amount necessary to increase the channel balance up to the 50% of the channel capacity.

The active rebalancing approach fails in the following cases:

- When the node with a low-balance channel does not have any candidate channel for rebalancing. The candidate channel must satisfy two conditions: (i) the node's balance in that channel must be greater than half of the total channel capacity; (ii) the node's balance in that

channel must be sufficient to transfer the rebalancing payment amount.
- When the rebalancing payment fails for no route (in the example of Figure 9, if there is no route between Berto and Carola).
- When the rebalancing payment fails because there is not enough balance in the channels of the found route (in the example of Figure 9, when one or more channels in the route between Berto and Carola have balance lower than 0.4 BTC).

For the sake of simplicity, we assume that rebalancing payments are executed instantly and no fees are charged to them.

b: Passive rebalancing

The passive rebalancing consists in adjusting the fee policy of a channel according to the channel balance, in a way that fee amount is inversely proportional to channel balance. Therefore, this approach encourages payments to traverse channels with high balance. In fact, the Dijkstra's algorithm used for finding a route for a payment tends to prefer routes with lower fees: therefore, if the fee of a channel is kept inversely proportional to the channel balance, payments should tend to traverse channels with high balance and to avoid channels with low balance. A linear function is used to map channel balance to channel fee policy.

c: Simulation strategy

To study the effectiveness of the rebalancing approaches discussed, we run simulations using a version of the simulator that implements the approach under observation. For each rebalancing approach, multiple simulations are conducted varying σ_a. The above-discussed snapshot of the LN is given in input to the simulator in these simulations.

A. SIMULATION RESULTS

a: Active rebalancing

Figure 10 shows the probability of payment failures for unbalancing when active rebalancing is implemented, compared with the case in which no rebalancing approach is implemented.

As it can be noticed, the difference of the two cases is slight: the probability of payment failures are almost the same in both cases.

The low effectiveness of the active rebalancing is due to the fact that most of the attempts of rebalancing fail. Table 5 shows, for each value of the payment amount tuner σ_a, the total number of active rebalancing attempts, the percentage of succeeded and the percentages of failed for each of the three causes explained above (absence of a candidate channel for unbalancing, failure of the rebalancing payment for no route, failure of the rebalancing payment for no balance). The Table shows that for any σ_a, just few rebalancing attempts succeed: at maximum around 16%. The majority of attempts fail because there is not enough balance to transfer the rebalancing payments; a substantial part also fails because

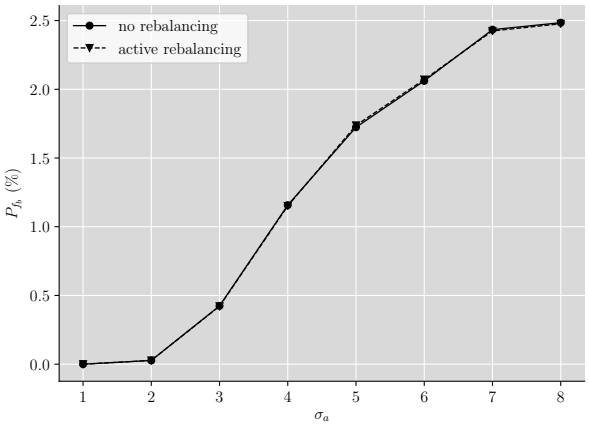

FIGURE 10: Probability of payment failures for unbalancing with/without active rebalancing varying payment amounts σ_a.

a candidate channel from which executing the rebalancing payment is not found.

TABLE 5: Rebalancing attempts.

σ_a	Attempts	Succeeded (%)	No channel (%)	No route (%)	No balance (%)
1	1726	4.06	20.74	2.84	72.36
2	3644	9.03	27.72	2.00	61.25
3	22067	11.48	21.52	2.56	64.44
4	36032	12.89	25.85	2.81	58.44
5	44819	16.32	23.11	2.08	58.49
6	52079	16.05	24.71	2.16	57.07
7	52557	15.52	27.16	2.47	54.85
8	57013	15.87	27.51	2.64	53.99

From that follows that the active rebalancing approach implemented is not effective for two reasons: mainly because channels with not sufficient balance in the network cause failures of the rebalancing payments; secondarily, because a candidate channel for executing the rebalancing payment is not found.

b: Passive rebalancing

Figure 11 shows the probabilities of failures for unbalancing with and without the passive rebalancing approach. Passive rebalancing is more effective than active rebalancing: the probability of failures for unbalancing is lower than the case without rebalancing for any value of σ_a.In particular, failures for unbalancing decrease of about one fourth in each case when passive rebalancing is implemented.

Passive rebalancing causes also a slight increase of failures for no route, as results in Appendix A show (Table 10). The reason for such increase is that certain payments must pay higher fees than in the case without passive rebalancing: these payments fail when there are not sufficient capacities in the channels for transferring the payment amounts plus the increased fees. However, the decrease of failures for no

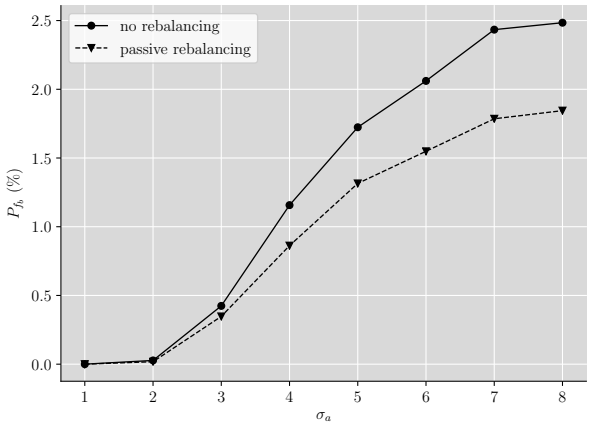

FIGURE 11: Probability of payment failures for unbalancing with/without passive rebalancing varying payment amounts σ_a.

balance is greater with respect to the increase of failures for no route, hence passive rebalancing is still effective.

VII. SERVICE PROVIDERS

In this Section the LN is studied in the service-providers scenario. The service-providers scenario is a typical case of use of the Lightning Network, in which a few nodes in the network (service providers) are payees only, as they are paid for their services. The majority of network nodes, instead, are payers only, as they send payments to the service providers.

The goal of the simulations in the service-providers scenario is to understand whether the LN can support this typical case of use in which most of the payments are directed to only few service-providers nodes.

a: Classes of nodes

To configure the scenario, three classes of nodes are set in the LN:

- Payees: they represent the service providers, namely, nodes that only receive payments. They are nodes directly connected to one of the six hubs of the LN (according to the identification of hub made in Section V). The reason of this choice is that it is convenient for a service provider to be connected to a hub, as also many other nodes are connected to the hub and therefore their payments can easily reach the provider. Service providers are expected to be a minority, so payees are 10% of the nodes directly connected to a hub. In addition, each payee is chosen in a way that, in the channel with the hub, the hub's balance is higher than the payee's balance, to avoid that payments directed to the payee through the hub fail for not sufficient balance. Payees are in total 188 nodes.
- Payers: they are the nodes than only send payments. For the same reason of having an efficient connection to the service providers, also payers are directly connected to a hub. Therefore, they constitute the remaining nodes directly connected to a hub which are not payers: in total, they are 1781.
- Hybrid nodes: they are nodes that both send and receive payments. they are the remaining nodes of the LN which do not belong to the categories of payees and payers: they are in total 1179 nodes.

In the simulations on the LN where the above-defined classes of nodes are set, 80% of the simulated payments flow from payers to payees, while the rest is exchanged among hybrid nodes, in order to preserve a certain level of traffic in the network not directed to the service providers.

b: Simulation strategy

To study the performance of the LN in the service-providers scenario, multiple simulations are run varying σ_a in the defined interval. The snapshot of the LN with the above-defined classes of node is given in input to the simulator in each simulation.

A. SIMULATION RESULTS

Figure 12 shows the probabilities of payment success of the simulations in the service-providers scenario, comparing it to the normal scenario (where no classes of nodes are set in the LN).

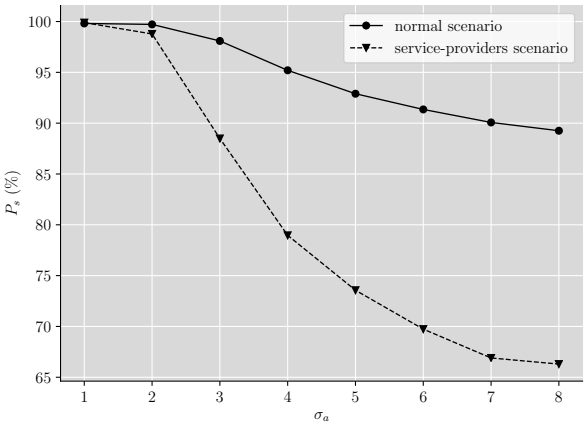

FIGURE 12: Payment success probability in the service-providers scenario varying payment amounts σ_a.

It can be noticed that, as payment amounts increase, the success rate in the service-provider scenario decreases, up to around 66%. The main reason of payment failures is channel unbalancing, which in the worst case (in correspondence to the highest σ_a) reaches a probability of around 26% (see results in Appendix A, Table 11). Such high probability of failures is due to the fact that, since payments are mostly directed to a few nodes, channels connecting to the nodes become unbalanced.

For this same reason, it can be noticed an increase of average payment attempts necessary before a payment succeeds, as Figure 13 shows. In fact, since payments bump into

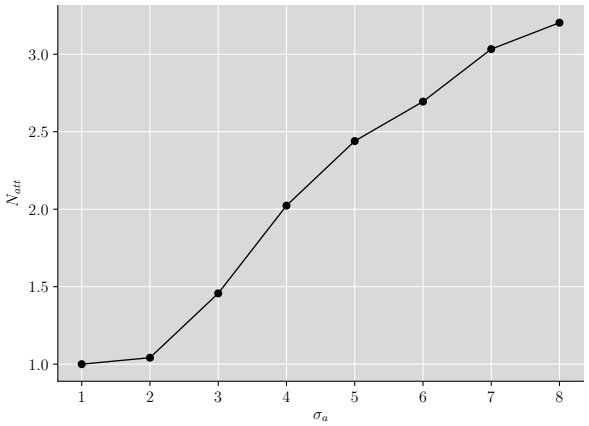

FIGURE 13: Average payment attempts in the service-providers scenario varying payment amounts σ_a.

unbalanced channels, they must be re-attempted more and more times.

VIII. CONCLUSIONS AND FUTURE WORK

The Lightning Network is the most promising solution to the well-known blockchain scalability issue. In this work we conducted simulations on the LN using CLoTH, an HTLC payment channel network simulator that we developed. The purpose was to investigate the effects of hubs, rebalancing approaches and service providers in the Lightninig Network.

Simulations on rebalancing have found that our passive rebalancing approach based on fee adjustment reduces by one fourth the probability of failures for unbalancing. Simulations on hubs revealed that the LN is resilient to the removal of six hubs, as the probability of payment success decreases only by 0.2% when all the hubs are removed. Finally, simulations on the service-provider scenario have shown that up to 26% of payments fail because channel directing to the service providers become unbalanced.

Our work shed new light on some capabilities and limitations of the Lightning Network: its resilience to the removal of hubs and a scarce performance in the service-providers scenario. In addition, we proposed an effective rebalancing approach which can be implemented in the LN protocol to address channel unbalancing.

Still, our work presents some limitations that will be addressed by future work. Both passive and active channel rebalancing approaches will be further investigated: to improve passive rebalancing, different non-linear fee-balance mapping functions will be tested, while to make active rebalancing effective, the algorithm will re-attempt failed rebalancing payments, possibly until they succeed. Further simulations will be conducted to understand whether the proposed rebalancing approaches can address channel unbalancing also in the service-provider scenario.

In addition, we will analyze the Lightning Network from the perspective of graph theory. For instance, instead of using the simulator itself for establishing the variation intervals of independent variables, measures like maximum flow will be used to define the limits of simulated payment amounts. Moreover, eigenvector centrality and other centrality measures will be used to identify the most central network nodes: similarly to the simulations on hubs, simulations will be run on the Lightning Network without the identified central nodes.

Finally, we plan to introduce the blockchain in CLoTH in order to study the interaction between the payment channel network and the blockchain (specifically, the opening and closure of payment channels), which currently is not captured by the simulator.

ACKNOWLEDGMENT

We wish to thank Mr. Giuseppe Futia and Prof. Guido Boella for their precise and precious revisions.

REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
[2] Y. Sompolinsky and A. Zohar, “Accelerating bitcoinâĂŹs transaction processing,” 2013.
[3] J. Poon and T. Dryja, “The bitcoin lightning network: Scalable off-chain instant payments,” 2016.
[4] C. Decker and R. Wattenhofer, “A fast and scalable payment network with bitcoin duplex micropayment channels,” in Symposium on Self-Stabilizing Systems. Springer, 2015, pp. 3–18.
[5] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, “Sprites: Payment channels that go faster than lightning.” CoRR, vol. abs/1702.05812, 2017.
[6] “Raiden network.” [Online]. Available: https://raiden.network/
[7] C. Burchert, C. Decker, and R. Wattenhofer, “Scalable funding of bitcoin micropayment channel networks,” Royal Society open science, vol. 5, no. 8, p. 180089, 2018.
[8] R. Khalil and A. Gervais, “Revive: Rebalancing off-blockchain payment networks,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 439–453.
[9] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, “Flare: An approach to routing in lightning network,” 2016.
[10] M. Conoscenti, A. Vetrò, J. De Martin, and F. Spini, “The cloth simulator for htlc payment networks with introductory lightning network performance results,” Information, vol. 9, no. 9, p. 223, 2018.
[11] “Maximum transaction rate.” [Online]. Available: https://en.bitcoin.it/wiki/Maximum_transaction_rate
[12] M. Trillo, “Stress test prepares visanet for the most wonderful time of the year.” [Online]. Available: https://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html
[13] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer et al., “On scaling decentralized blockchains,” in International Conference on Financial Cryptography and Data Security. Springer, 2016, pp. 106–125.
[14] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 3–16.
[15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in Annual International Cryptology Conference. Springer, 2017, pp. 357–388.
[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017, pp. 51–68.
[17] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A secure sharding protocol for open blockchains,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 17–30.
[18] A. E. Gencer, R. van Renesse, and E. G. Sirer, “Service-oriented sharding with aspen,” arXiv preprint arXiv:1611.06816, 2016.

[19] R. Jain, The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. John Wiley & Sons, 1990.
[20] A. van Schie, "Routing scalable bitcoin payments," 2015.
[21] D. Piatkivskyi and M. Nowostawski, "Split payments in payment networks," in Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, 2018, pp. 67–75.
[22] R. Yu, G. Xue, V. T. Kilari, D. Yang, and J. Tang, "Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks," in 2018 27th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2018, pp. 1–9.
[23] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing payments on the lightning network," in Proceedings of the 2018 International Conference on Blockchain, 2018.
[24] D. Reynolds, "Simulating a decentralized lightning network with 10 million users." [Online]. Available: https://hackernoon.com/simulating-a-decentralized-lightning-network-with-10-million-users-9a8b5930fa7a
[25] A.-L. Barabási et al., Network science. Cambridge university press, 2016.

.

APPENDIX A COMPLETE SIMULATION RESULTS

Tables 6 to 11 show the complete results of the simulations presented in this paper. Each Table shows the independent variable and the output performance measures ($P_{\bar{k}}$ and $P_{f_{\bar{c}}}$ are omitted since they are always zero). Only the mean values of the performance measures are presented; the complete results containing also variances and confidence intervals are available online[4].

TABLE 6: Complete simulation results varying payment rate.

r_π (pay/s)	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
10	88.89%	9.07%	1.97%	0.07%	403.23	3.33	1.17
20	88.77%	9.09%	2.07%	0.07%	407.25	3.33	1.20
30	88.72%	8.98%	2.24%	0.06%	408.09	3.33	1.21
40	88.65%	8.98%	2.33%	0.05%	411.61	3.33	1.23
50	88.53%	9.00%	2.43%	0.04%	412.96	3.33	1.24
60	88.46%	8.98%	2.52%	0.04%	417.99	3.33	1.26
70	88.45%	8.94%	2.58%	0.04%	418.73	3.33	1.27
80	88.31%	8.99%	2.66%	0.04%	420.71	3.33	1.28
90	88.26%	9.02%	2.69%	0.03%	422.83	3.33	1.29
100	88.31%	8.94%	2.73%	0.03%	425.98	3.33	1.30

TABLE 7: Complete simulation results varying payment amounts.

σ_a	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
1	99.81%	0.19%	0.00%	0.00%	365.35	3.36	1.00
2	99.72%	0.26%	0.03%	0.00%	368.21	3.34	1.01
3	98.09%	1.48%	0.42%	0.01%	377.79	3.35	1.05
4	95.20%	3.63%	1.16%	0.01%	389.82	3.34	1.12
5	92.90%	5.36%	1.72%	0.03%	399.67	3.33	1.18
6	91.35%	6.57%	2.06%	0.02%	407.18	3.34	1.21
7	90.07%	7.46%	2.43%	0.03%	415.68	3.34	1.25
8	89.26%	8.23%	2.48%	0.03%	418.40	3.34	1.27
9	88.85%	8.45%	2.67%	0.03%	422.05	3.34	1.28
10	88.31%	8.94%	2.73%	0.03%	425.98	3.33	1.30

TABLE 8: Complete simulation results varying number of hubs removed $N_{\bar{n}}$.

$N_{\bar{n}}$	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
0	95.20%	3.63%	1.16%	0.01%	389.82	3.34	1.12
1	95.29%	3.53%	1.17%	0.01%	399.07	3.44	1.10
2	95.19%	3.64%	1.16%	0.01%	405.77	3.47	1.11
3	94.98%	3.82%	1.18%	0.02%	403.79	3.52	1.11
4	95.02%	3.79%	1.17%	0.02%	422.68	3.58	1.15
5	94.79%	3.93%	1.26%	0.03%	431.99	3.61	1.15
6	94.98%	3.83%	1.17%	0.02%	431.26	3.62	1.15

TABLE 9: Complete simulation results with active rebalancing.

σ_a	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
1	99.81%	0.19%	0.00%	0.00%	365.36	3.36	1.00
2	99.72%	0.26%	0.03%	0.00%	366.84	3.35	1.00
3	98.09%	1.48%	0.42%	0.01%	374.89	3.35	1.04
4	95.21%	3.63%	1.15%	0.01%	384.31	3.34	1.09
5	92.88%	5.36%	1.74%	0.02%	394.81	3.34	1.14
6	91.34%	6.57%	2.07%	0.03%	402.97	3.35	1.18
7	90.08%	7.46%	2.42%	0.03%	408.09	3.35	1.20
8	89.26%	8.23%	2.48%	0.03%	412.69	3.35	1.22

TABLE 10: Complete simulation results with passive rebalancing.

σ_a	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
1	99.81%	0.19%	0.00%	0.00%	311.83	2.85	1.00
2	99.72%	0.26%	0.02%	0.00%	313.15	2.86	1.00
3	98.13%	1.52%	0.35%	0.00%	320.75	2.92	1.01
4	95.31%	3.83%	0.86%	0.00%	330.26	2.98	1.02
5	93.04%	5.64%	1.32%	0.00%	330.81	3.00	1.02
6	91.65%	6.80%	1.55%	0.00%	336.62	3.02	1.03
7	90.40%	7.81%	1.79%	0.00%	337.94	3.03	1.03
8	89.63%	8.52%	1.84%	0.00%	338.94	3.03	1.04

TABLE 11: Complete simulation results in the service-providers scenario.

σ_a	P_s	P_{f_r}	P_{f_b}	$P_{\bar{t}}$	T (ms)	L_r (hops)	N_{att}
1	99.89%	0.11%	0.00%	0.00%	340.62	3.13	1.00
2	98.78%	0.14%	1.09%	0.00%	350.62	3.13	1.04
3	88.49%	1.13%	10.36%	0.01%	455.52	3.22	1.46
4	78.96%	2.87%	18.07%	0.10%	593.20	3.24	2.02
5	73.56%	4.43%	21.71%	0.30%	687.33	3.26	2.44
6	69.74%	5.60%	24.12%	0.55%	746.51	3.28	2.69
7	66.90%	6.23%	26.17%	0.70%	829.97	3.30	3.03
8	66.30%	6.86%	26.10%	0.74%	867.99	3.30	3.20

[4]https://researchdata.nexacenter.org/payment-network-simulator/hubs-rebalance-sp-results.zip