Split payments in payment networks

Dmytro Piatkivskyi and Mariusz Nowostawski

NTNU, Norway
dmytro.piatkivskyi,mariusz.nowostawski@ntnu.no

1 Abstract

Traditional blockchain systems, such as Bitcoin, focus on transactions in which entire amount is transferred from one owner to the other, in a single, atomic operation. This model has been re-used in the context of payment networks such as Lightning network. In this work, we propose and investigate new payment model, called *split payments*, in which the total amount to be transferred is split into unit-amounts and is transferred independently through the same or different routes. By splitting the payments this way, we achieve an improved total liquidity of the payment network, simplify the route advertising, reduce the amount of funds needed to be locked in the channels, and improve the privacy properties of the payment network. This article provides details on the split payment method, motivation for the work, experimental setup as well as results obtained from simulating various topologies using atomic and split payment mechanisms.

2 Introduction

The scalability problem of Bitcoin has received considerable attention by the community. Various solutions have been proposed [1–3] and one of the most promising is the utilization of off-chain transactions, for example through the Lightning network [4].

In the battle with the blockchain's main problem, scalability, the idea of off-chain payment networks has evolved. Off-chain payment network is based on the concept of state channels that can operate offline, consulting the blockchain only when opening or closing a channel. The state channels form a payment network which allows for peer-to-peer instantaneous transactions.

The front-runner of the payment networks in the area of cryptocurrencies is the Lightning network [4]. Even though our work focuses specifically on the Lightning network, it applies to any payment network that follows the idea of state channels.

Off-chain payments are a novel concept that has received relatively little academic attention. Nevertheless, the first publications have demonstrated that the off-chain payment network cannot satisfy all the desirable properties and a number of trade offs will have to be made to balance security, privacy, throughput and liquidity of the network. In addition, the self-emerging network topology and its usage is highly uncertain, as it depends on market forces, public uptake and

financial feasibility of operating the intermediary nodes. Yet, it is the topology that will play one of the fundamental roles in the performance and characteristics of the network properties. For that reason, we have studied the network properties and worked on an operational mode that improves various properties of the payment network. Our suggestion is to split payments and spread them among a number of unit payments. We experimentally demonstrate, that such a method not only solves some of the principle issues within the network, but also increases the liquidity, while reducing the need of investment from intermediate nodes.

The concept of shifting from atomic payments to money flows has been discussed before [5, 6, 4]. It changes and redefines the concept of money. In addition to store of value and medium of exchange, it becomes apparent that aspects such as fairness and easiness of transacting one's funds are fundamentally important. The efficient money flow networks have not yet been technically designed, nor adequately studied. The cryptocurrency domain offers numerous advancements and insights that can be investigated in the context of money flows. One of them, deployed and used on top of Bitcoin network, is the Lightning network, that provide relative simplicity when operating money flows. It makes the network appropriate for practical use and, possibly, wider market adoption.

The payment networks ultimately solve the inherent blockchain scalability limitation, however, the payment networks themselves are limited in many ways. They require careful consideration and appropriate balancing of multiple, often competing, trade offs. Many properties of the network will depend on the way the network organizes itself. Implementation choices will make a great difference.

How to choose the appropriate protocol? What topology would facilitate better network properties? How can the network increase its liquidity while preserving other properties at the same level? What parameters affect the network liquidity – the number of channels in the network, their capacities, the path selection algorithm? The answer to these questions requires thorough research on a number of individual research questions. In this research we are specifically focusing on the liquidity of the network, simplicity of the route advertisements, and funds being locked in the channels.

The main constraint of the payment-channel networks, such as the Lightning network, is that funds have to be invested in channels beforehand, and, if used inefficiently, can be considered locked in the corresponding channels for an extended periods of time. Therefore, this suggests an obvious incentive to invest in the network as little as possible, to achieve little funds lock-in. However, the network has to provide enough liquidity to route all the payments, therefore it is natural that larger investments (larger amounts being locked in the channels) will offer improved transaction liquidity. Beside locking money in channels, these networks pose a number of other trade offs – privacy, security, concurrency, efficiency, complexity of route advertising, and others.

In this paper we demonstrate how to better organize the network. In particular, we suggest to abandon the idea of single atomic payments and to embrace the concept of money flows, and the use of split payments. We show that split-

ting payments into a number of unit payments improves a number of important properties, such as liquidity, funds lock-in, and privacy.

3 Past work

Payment networks are not a new concept. They are studied under different variations of the notion – trust networks [5–7], credit networks [8], path-based transaction (PBT) networks [9] and Payment-Channel Networks (PCN) [10]. These concepts differ in various aspects, and at the same time, they have many things in common. They all rely on the concept of a network (directed graph) and interactions between participating nodes. An important aspect, in the light of this paper, is the theoretical foundation laid by Dandekar et al. [8]. They dived into the problem of network liquidity, defined the prominent metrics that affect it and demonstrated credit networks' effectiveness with the growing number of connections.

There were efforts undertaken to improve the Lightning network. There were efforts undertaken to extend specifically the Lightning protocol, or define the parts that have not been specified in the original paper. Some of them target payment routing, i.e. a way to find a route between nodes. The first proposal, Flare [11], suggests maintaining routing tables to be able to discover paths in the network. Roos et al. [9] proposed an alternative routing scheme that is privacy preserving. Grunspan and Pérez-Marco [12] put forward an idea of ant routing where path lookup requests are passed from node to node in the network. Another group of papers suggested changes to the initial protocol proposal. Decker et al. [13] suggested an improvement over the Lightning transaction update mechanism. Malavolta at al. [14] studied the mechanism which binds transactions together, so they can be routed. They formally defined multi-hop locks and suggested an improvement over the technique. Another paper from this research group [10] demonstrated a rather surprising trade off between privacy and concurrency in PCNs, and impossibility to achieve both simultaneously. In the later parts of this article, we will show that our proposed mechanism addresses and mitigates the problem. Piatkivskyi et al. [15] brought attention to the problem of colluding nodes and discussed how it influences forensics of the Lightning network. Herrera-Joancomarti et al. [16] gave an overview on the state of the art in privacy issues of payment networks.

Note, that there is no prior work specifically focusing on the payment flows or payment splitting that investigates the properties of the payment network based on split payment mechanism. Atomic Multi-Path Payments (AMP) [17] have been proposed to split payments across multiple paths to increase the payment flow possible between two nodes. This method can be seen as an evolution between single atomic payments and our proposal. The difference is discussed in Section 5.2. In fact, in our experiments we compare split payments to AMP.

4 Background

4.1 The Lightning network

The Lightning network is a payment protocol built on top of the Bitcoin protocol. It allows for transaction throughput scaling by keeping and updating Bitcoin transactions off-chain. A Lightning network transaction is processed within Lightning channels which are actually Bitcoin transactions. The idea is that two users mutually fund a Bitcoin transaction, the *Funding 1* transaction, in Figure 1, and spend it returning the invested funds with the *Commitment 1* transaction. They both sign the commitment transactions, but only publish the funding transactions on the blockchain.

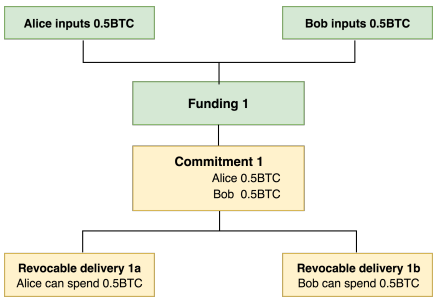

Fig. 1. A funding transaction

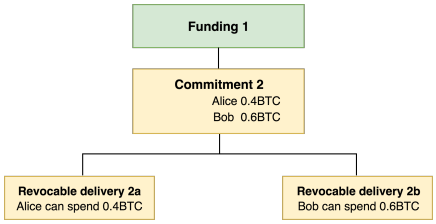

Fig. 2. Transacting 0.1BTC

Once a channel is established, i.e. the funding transaction reaches the blockchain, funds can be moved within the channel (up to the channel capacity) by simply updating the commitment transaction. When any participant wants to spend funds outside the payment network, the channel is closed by publishing the current state of the commitment transaction to the blockchain.

The described payment channels allow a nearly unlimited number of transactions within a channel. Note however, that this simple scheme requires two parties to have a pre-existing channel open. One would need to open a channel with everyone one ever interacts with. This is cumbersome, as well as costly in terms of initial channel setup and funding transaction costs. A solution to this problem is to route payments through existing channels. The next subsection describes this payment routing in detail.

Payment Routing The Lightning network protocol facilitates the ability for making a payment to a node, without having a direct channel with it. To achieve that, One can route a payment through intermediate nodes, each of which has a channel opened with one another. For example, if there is a channel between Alice and Bob and a channel between Bob and Charlie, Alice can send funds to Charlie through Bob. The technique that makes payment routing possible is

called *Hashed Timelock Contract* (HTLC). By utilizing *Hashed Timelock Contracts* (HTLCs), the two transfers, from Alice to Bob and from Bob to Charlie, are atomic, that is either both of them are executed, or none are. HTLCs allow Bob to receive money from Alice after Charlie has received money from Bob.

HTLC is an agreement of payment upon revealing a secret, pre-image R of a certain hash value H. The scheme starts with the recipient, Charlie, generating and sharing H with the sender, Alice. Based on the hash value H Alice creates an HTLC contract with Bob, which says that Alice will pay Bob when Bob knows R. Bob then creates an HTLC contract with Charlie (see Figure 3).

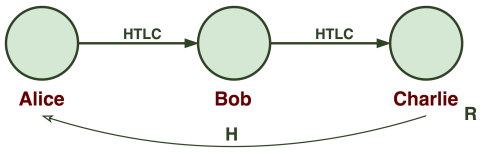

Fig. 3. A chain of HTLCs in a channel

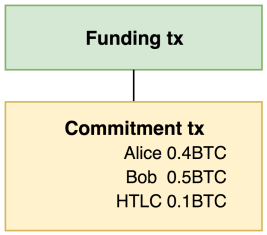

Fig. 4. Commitment transaction with an HTLC

Having received an HTLC from Bob, Charlie reveals R and gets his money from Bob even before Alice sends funds to him. Bob is safe, because he has his money promised by Alice. The promise is enforced on the blockchain if Alice does not comply. When Bob and Charlie execute their HTLC, Bob gets to know the R value. Knowing the R value Bob can pull the money from Alice by executing their HTLC. The scheme works the same way with more than three parties on the path.

An HTLC is realized as one additional output in the commitment transaction – the last output in the Figure 4. For more details on its implementation the reader should refer the original paper [4].

The Lightning network topology The Lightning network has been deployed on both, the Bitcoin testnet as well as the mainnet. However, it is still in its early stage of deployment and testing. Many properties of the system will depend on its wider user and service provider adaption. The main important uncertainty arises around the Lightning network topology [18]. The main possible emerging topologies are *hub-and-spoke* and *organic*. The hub-and-spoke topology implies big payment processing hubs which would route the payments. Such a topology comes with drawbacks such as high centralisation, threat to anonymity, and money locking. The alternative organic topology does not suffer from these problems to the same extent. However, taking into account the potentially low channel capacities, organic routing might be highly inefficient. Another downside for the organic topology is that the number of on-chain transactions is expected

to be much higher than with the hub-and-spoke topology, hence less users can be served having the same block size limit. The comparison of these topologies poses an interesting research question that has not yet been studied.

One aspect of the emerging topology can be safely assumed — there will be nodes in the network more connected than others (*hub*) and nodes that use the network to transfer funds (*end users*). The *hub* nodes will be routing payments between *end nodes*, or *end users*. We will be using the terminology borrowed from Prihodko et al. [11] and call the former *routing nodes* and the latter *wallet nodes*. Wallet nodes are end users that wish to send and receive payments. They are not expected to route payments, although can in principle do so. Routing nodes, on the other hand, are nodes that offer their participation in the infrastructure as a service, route payments and charge a fee for that. The whole responsibility of providing the necessary and efficient infrastructure lies on the routing nodes.

We call an investment in the network all the funds that are locked in channels with the purpose of facilitating payments. Generally, these are the funds invested by the routing nodes. We expect routing nodes not to invest in channels with wallet nodes, or invest short-term, which we do not count as a network investment.

4.2 Model

A payment network is modelled as a directed graph $G = (V, E)$, where V is a set of payment network nodes (e.g. Lightning network nodes) and E is a set of channels between them. A node is an abstract notion, that represents an entity holding control of a number of Bitcoin addresses that has locked funds in Lightning funding transactions. An edge $e = (v_1, v_2)$ in the graph G corresponds to a Lightning channel (or possibly multiple channels) between two nodes v_1 and v_2. For convenience, we consider a bidirectional Lightning channel as two directed edges (v_1, v_2) and (v_2, v_1). Hence, it holds that if $(v_1, v_2) \in E$, then $(v_2, v_1) \in E$. The two edges are updated together, if both parties agree. We further use the terms `edge` and `channel` interchangeably.

A channel has a number of properties associated with it. $cap(v_1, v_2)$ is channel capacity, i.e. the amount of bitcoins node v_1 can send to node v_2. Sum of opposite edges $c(v_1, v_2) + c(v_2, v_1)$ is a constant, equalling to the total amount of funds locked in the channel (in the funding transaction). It is possible that two nodes create multiple channels between themselves. If so, we consider capacity of such edge as a sum of channel capacities.

A transaction in the network is defined by a tuple (s, t, v), where s is the source, t is the destination and v is the value of the transaction. Each transaction needs to find a route to be routed by. It is posed as maximum flow problem. Please note that we already assume that a payment can be split as the maximum flow consists of possibly multiple flows. However, such a splitting is not timely spread. A transaction upon its successful execution changes the distribution of balances of all the channels on the route.

5 Split payments

5.1 Payment splitting proposal

The core idea of our proposal is to split payments into a number of smaller sub-payments of equal amounts, i.e. a number of payments of unit amounts, and send them independently, not preserving the atomicity property. There are various ways to split payments up. One way of doing so is amounts of the orders of 10. If a user wants to pay 23k satoshi, she splits the payment into 2 sub-payments of 10k satoshi and 3 sub-payments of 1k satoshi.

Split payments are to be sent independently. At the moment of payment initiation, the sender calculates the cheapest path and begins establishing HTLCs by that path starting with the larger sub-payments. If at any point of time an HTLC establishment fails, or the sender receives a fee update, she suspends the sub-payments for which HTLCs have not yet been established and re-calculates the cheapest path again. Then the suspended sub-payments are resumed to be sent by the new cheapest path. It may happen that for some larger sub-payment there is no path of needed capacity. In such a case the sub-payment has to be further split. If there is no capacity to route any payments in the needed direction, the whole sub-payment queue is suspended for a timeout. After the timeout is elapsed, an attempt to send the sub-payment is repeated. This process could continue indefinitely until the payment is complete. The user sets a time frame within which the payment is expected to execute. We call such parameter `time to live` (TTL). Obviously, some payments have to be carried out instantly, while other can wait. It will make a trade off between the time it takes to complete a payment and the fee paid for that payment. A payment is considered successful if all sub-payments are successfully delivered within the set TTL. If a payment has not been delivered within the set TTL, it is marked as failed, even though it could have been partially sent. Such payments are not sent back, consequentially failed payments change the balances of the nodes en route. Partial transitions are further discussed in the following sections.

5.2 Atomic multi-path payments

Atomic multi-path payments (AMP) [17] are the implementation of the concept of flows in a flow network. There are number of principal differences that sets AMP and split payments apart. The main difference is that the whole AMP flow executes atomically, that is either all of the sub-payments are sent at once or none. For that, each extended sub-payment remains pending until all sub-payment flows are extended. That increases the duration of funds being locked.

The superiority of split payments success rate comes from the fact that an AMP fails if maximum flow between two nodes is less than the payment amount. Split payments still attempt to execute. As sub-payments are timely spread, there is a chance that within the execution window some payments will pass in the opposite direction increasing the maximum amount that can be sent. A substantial disadvantage is that a payment can be only partially executed.

Notwithstanding the transaction acknowledgement complications, we deem partial payments harmless as the rest of the payment can be sent on-chain using splicing [19], if to be sent from a Lightning channel. We stress that partially executed payments do not introduce additional inconvenience. If a payment cannot be executed, it has to be sent in any other way anyway. The non-executed part of a payment can be sent the very same way the whole failed payment would have to be sent.

5.3 Network analysis

Privacy The privacy benefits of the proposed strategy are many. First of all, there is no need for routing nodes to disclose current channel capacities, which are highly volatile, and typically have to be constantly requested, which otherwise may yield high error rates. Instead, the paying node, knowing the static topology, can simply request unit payments. The probability that a particular path is able to route a payment is relatively high, given the small unit payment amount.

The compromise here is that unlike in [10], where channel participants are kept secret when routing and only disclosed on the chain when closing the channel, participants have to be known when advertising channels. The major gain comparing to their scheme is route calculation efficiency. The two schemes are not in competition with each other and can be combined and work together. Besides, the fact that all the payments could only be of a certain unit amount makes the payment correlation analysis difficult. Timing analysis will be significantly complicated as there expected to be a large number of such unit payments in the network.

Even if a sub-payment is related along the route it is just one out of many sub-payments. The routing nodes will not know the actual amount being sent as the task of de-multiplexing a payment, that is to reconstruct a single payment from a number of sub-payments, poses a big challenge even if all of sub-payments were routed through the same router. Some sub-payments may originate from the sender, or from nodes before the sender, and be destined further than the receiver. Thus, even payment correlation by the single value of HTLC riddle becomes a minor threat, if considered to be a threat at all, as each sub-payment is based on a different riddle. Finally, be the sub-payments not all sent using the same route, an attacker would have to control all the nodes used to be able to reconstruct the payment.

Even though the question of privacy constitutes a research endeavour on its own and goes beyond the scope of this article, we are confident to claim that split-payment network will improve privacy and it would make the transaction tracking analysis harder.

Security Generally, we believe that with the proposed strategy a payment network becomes much more robust to instabilities and dishonest nodes. Money locking is an inherent problem where a dishonest node stops responding with the purpose of increasing the time for which money are locked in passing HTLCs. The

larger the payment being sent, the greater the associated risk. Consequentially, in our proposal the collateral risk is relatively low with split payments because all the actual payments are of unit amount only. If a sub-payment gets stuck, the sender stops using the routing node that failed. Therefore, the collateral risk is notably reduced, and contained only to the unit amount. Moreover, as there exists a threat of losing money to colluded receiver and a node en route [15], the maximum loss is limited by the amount of the largest sub-payment. If the sender loses money to the colluding nodes, it can simply stop casting the flow.

Concurrency Split payments transform the problem of possible occurrence of a deadlock into a performance bottleneck. While deadlocks are still theoretically possible, for it to happen a number of sub-payments totaling to the channel capacity have to conglomerate simultaneously at two nodes. We consider the chance of such a deadlock negligible, resolving the trade-off between privacy and concurrency in payment networks described in [10]. Besides, a naive approach to cancel stuck payments would easily resolve a deadlock. We can simply assume that the split payment mechanism resolves the trade-off between privacy and concurrency in payment networks described in [10], and it becomes a non-issue. After all, the whole cryptography is based on making the chance of failure incredibly small. Here, we tolerate negligibly small chance of payments being stuck for a while and having to be resent.

Lower fees The described use of network will presumably yield lower fees for transactions. First of all, it may happen that the cheapest path is not able to process the whole payment due to capacity limitation. If not splitting the payment, the sender would have to use another, a more expensive, channel. Secondly, since payments are sent sequentially, it may happen that a cheaper path will appear some time after the payment has been issued. Most importantly, the more efficient network will naturally drive the transaction cost down. This will both, improve the liquidity as well as keep the dynamic fee pricing efficient for the network.

6 Simulation

The choice of methods in our research has been dictated by the complexity of the theoretical model which is limited in what answers it can provide. Using a simulation for such an immense problem space is both laborious and error prone. Even so, it can provide insightful answers and help to shape the research landscape.

A massive effort have been invested into developing a Lightning network simulator called Blyskavka (a Ukrainian word for Lightning). The simulator is meant to be open source, yet making it public requires certain preparations which hinders the release. Blyskavka is a multi-agent simulator that was built for general purpose payment network simulations. It is written in java and uses

MASON [20] as a simulation engine. The choice of the simulation engine comes at a high cost of poor scalability. We were able to run networks up to 50000 nodes, which, however, was too slow to run extensive simulations, therefore the largest network we run in our experiments is of 10000 nodes. Blyskavka simulates the Lightning network operation rather than the Lightning network itself, meaning it does not simulate the actual transactions being signed and the blockchain communication. It does open and close channels that are modelled as graph edges. It also simulates HTLC's by blocking and then releasing amounts on the path.

The simulation is built in a modular way, so each module can be substituted with an alternative implementation in code, and be configured through configuration file. An example of configuration file is provided in Appendix A. Structural and behavioral parts of the network are separated as well. Network graph generation is decoupled from the simulation at all. The network generation module, cracow, outputs the generated graph into a file which the simulation then reads and translates into entities it operates with. At the moment, of writing, cracow module The simulation works with the *Newman-Watts–Strogatz* model of the small-world network family, the *uniform random* graph model and a custom model that we call *peripheral*. A respective model is specified in the configuration file with $ALGO$ parameter. Every model takes the $NETWORK\_SIZE$ and the number of channels K as parameters. In addition to the network graph, cracow generates, currently uniformly only, capacities and costs for each edge. The respective parameters are $MIN\_CAPACITY$, $MAX\_CAPACITY$, MIN_COST and MAX_COST. The *peripheral* graph model differentiates between the core network that consists of routing nodes and the wallet nodes that connect to the core network — the peripheral network. The core network is generated following any other model. If the parameter $ALGO$ is set to $PERIPHERAL$, the program expects a block of parameters with prefix $CORE\_$ parameterizing the core network model. Having generated the core network, the wallet nodes are added, choosing randomly K routing nodes to connect to.

The behavioral part of the simulator distinguishes node behavior and network behavior. The node behavior defines payment issuance and, in theory, channel management. The experiments this article describes treat the topology as static, consequently include no channel management. The network behavior defines the way paths are discovered and payments are sent.

The simulation is discrete event based. Each simulation step a node decides what to do — whether to send a payment or not. In all the experiments the payment frequency for each node is once every 25 simulation steps. If a node decides to transfer money, it randomly with uniform distribution selects the destination node and the payment amount within the set range — between 0.1 and 20. As a result, nodes create uniform and symmetric traffic in the network. The node behavior is defined within a single class. It can be simply substituted with any implementation of the node behavior interface to fit one's needs. Currently, all users are endowed the same behavior which could be simply changed as well. As payments can be delayed, they are scheduled as well. The network behavior

in the simulation is described by entity *Payment*. Each step a payment makes an attempt to execute itself – atomic payments at once, split payments a unit amount sub-payment at a time. If a payment cannot be executed, it is scheduled for the next step until it is out of time to live (TTL).

Paths are discovered in an extensive depth-first search, until one is found. Two strategies were examined – randomizing paths on each hop or choosing next hop in an ordered fashion. Both have proven to produce the same effect on the studied metrics. The explanation comes from the work of Dandekar et al. [8] where they proved that path choice in payment networks does not affect the liquidity of the network. Even though the underlying assumptions are different in the Lightning network, the theorem seems to stand. We are set to investigate in full the theorem's applicability to the Lightning network and split payment as well. It is to be described in yet another paper.

An important feature implemented in Blyskavka is the for-of-war like visibility, as Flare [11] suggests. Each node is only aware of channels within the reach of N hops. We used $N{=}3$ which, given the small size of the network, gave nearly full visibility of the network. Before searching for a path, two nodes, the sender and the receiver, join their views on the network, after which the path discovering algorithm is applied. The simulation is flexible on what metrics it can take measurements of. For the purpose of the described research, only success rate was of interest. An example of program output is given in Appendix B

Financial services are expected to meet the most strict requirements. Usability and availability are among them, although not currently met by Bitcoin. The foremost requirement for the payment network is to always be able to route a payment. Under this assumption, we set a goal of providing all time available service, having invested as less in the network as possible. We do not consider anomalously large payments. If an unusually large payment is issued, the network has to rebuild itself to be able to route it. The traffic we create in our simulation is rather uniform and constant. We consider it to be the peak traffic. Therefore, anything less of that can be easily processed by the network.

Generated networks For this research we generate both, hub-and-spoke and organic topologies. Hub-and-spoke topology correspond to the *peripheral* graph model, organic topology is described by either the *Newman-Watts–Strogatz* model or by the *uniform random* graph model.

In our experiments we study networks of 1000 and 10000 nodes with different level of connectivity, defined by the parameter K. The small network size is dictated by the poor scalability of the simulator. Hub-and-spoke network has the core network consisting of 20 nodes for the network size of 1000 nodes and 50 nodes for the network size of 10000 nodes. The numbers are chosen deliberately so that the success rate starts low enough to show its increase with TTL value. Each node, regardless if it is a routing or wallet node, has K channels with initial capacity of 5 on each side. For this research channel cost is disregarded.

Intuitively, and then proven experimentally, the organic topology efficiency grows with the number of channels in the network. Organic topology with $K{=}2$

is very inefficient, hence we set *K=4*. To match the total funds invested into the hub-and-spoke network under scrutiny, the initial channel capacities should remain 5. We also generate higher connectivity networks with *K=4* for hub-and-spoke network and *K=8* for organic network.

Table 6 summarizes the generated networks.

Table 1. Generated networks

	Hub-and-spoke	Organic
1000 nodes, low connectivity	K=2	K=4
1000 nodes, high connectivity	K=4	K=8
10000 nodes	K=2	K=8

7 Experiments

It is hard to study the effects a change of a variable causes in the various network properties. For that, all variables that could also affect the success rate of the network have to be fixed, while still conserving the adequate liveliness and soundness of the network. To reduce the number of confounding variables, we randomly generate an instance of a particular network topology, with a given fixed properties. Any comparison is done then for exactly the same network configuration.

In the effort to define the network variables important to the research we also have to think about implicit correlations that may exist. To give an example, comparing split and non-split payments we have to make sure that the particular path selection strategy that we employ does not favor either of them (see Section 7.2).

In the experiments described in this article TTL is an independent variable, i.e. we study the dependency of success rate on the TTL value. For that, TTL is being varied from 50 simulation steps (where slight TTL increase brings a considerable difference in success rate) to 2000 simulation steps (where there is no longer any substantial increase in success rate with growing value of TTL). That demonstrates how the network improves with longer TTL.

The experiments were performed within a strictly defined framework. Each run lasts at least $N = 3 * TTL$ simulation steps and repeated multiple times to investigate the variance across runs. Furthermore, network topologies were generated randomly and for the same configuration there were generated multiple instances of the same topology, to compensate for a particular instance favouring one or the other model, just out of pure chance of the connectivity of a given instance.

The ultimate benchmark for the network we deem the success rate. To adequately measure it, we run the simulation for a number of steps N, but stop accounting transactions into statistics TTL steps before it finishes. This leaves

each transaction enough time to either complete or fail. The transactions in the network, however, are continously generated and they continue creating traffic.

7.1 Split payments vs. AMP

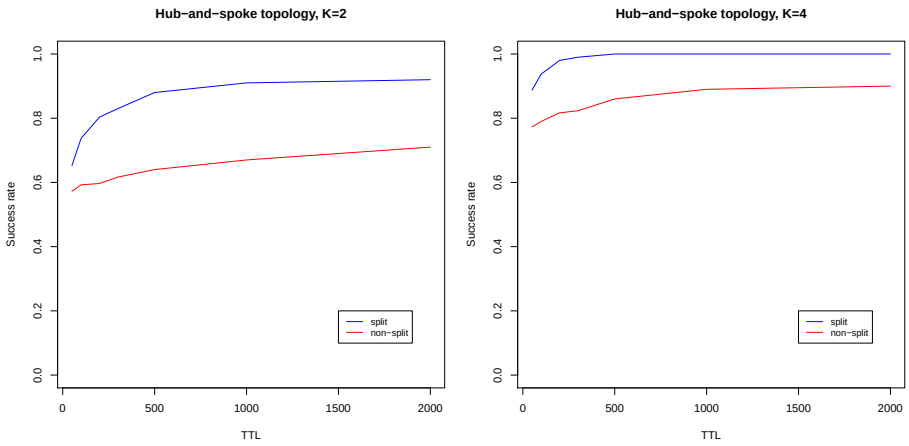

Fig. 5. A 20-1k hub-and-spoke network.

The core of the experiments was focused on the comparison of split payments and AMPs. Split payments were implemented in the simulation as described in Section 5.1. Below are shown two families of charts – for hub-and-spoke (Fig. 5) and organic (Fig. 6) topologies, having had invested the same amount in the network. Both topologies are generated for different sizes of the network – 1000 (Fig. 5 and 6) and 10000 nodes(Fig. 7).

The first thing to notice is the striking difference the splitting makes across all charts, particularly in case of the organic topology. With $TTL > 1000$, the difference is above 35%. The hub-and-spoke network also shows a significant improvement which is more constant and makes up over 10% improvement, reaching 20% difference for $TTL > 1000$. Those experimental results demonstrate the superiority of split payments when it comes to liquidity. Apart from proving the better performance of split payments these graphs provide hints about what network configurations are more efficient. Even though promising, those hints are not to be considered facts and need to be verified in a more rigorous manner.

Better connectivity networks In all of the experiments we keep the amount invested in the network at the same level to make the comparisons meaningful. Therefore, when increasing the number of channels K twofold, we divide the average channel capacity by 2. All the figures suggest that it is better to invest

less in a single channel and have more channels established. In other words, the more interconnected the network the greater its performance. The better connectivity comes at the cost of more on-chain transactions, the problem the payment networks are trying to solve. For that reason, the trade off will settle naturally depending on the block size and the number of users in the system. It would be interesting to give an estimate once fees for Bitcoin and the Lightning network settle down.

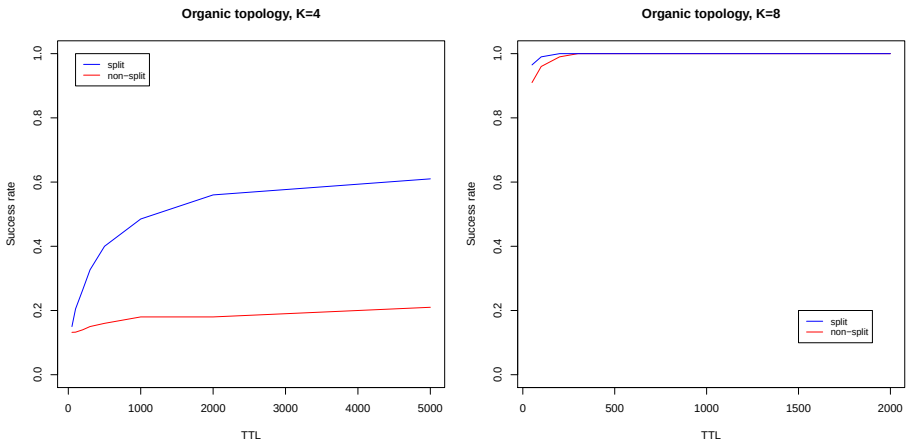

Fig. 6. An organic topology of 1000 nodes.

Hub-and-spoke topology efficiency Hub-and-spoke topology is by far outperforming the organic topology, even when the latter has twice as many channels (of half capacity, so the total investment in the network remains constant). Important to note that wallet nodes in the hub-and-spoke topology are not considered when routing. If they take part relaying payments, the efficiency, hence the liquidity, grows considerably. This suggests that in spite of all the shortcomings, some form of centralization will be present as it constitutes a major factor to the network efficiency.

High intensity traffic Having the same core network, but increased number of wallet nodes increases the success rate in the hub-and-spoke network. The organic network, on the other hand, suffers from the growing number of nodes. On the other hand, the splitting strategy shows the best efficiency for larger organic topologies making the difference of about up to 65%, taking up 34% success rate of atomic multi-path payments to 99% of split payments. This is a rather considerable improvement over the atomic baseline.

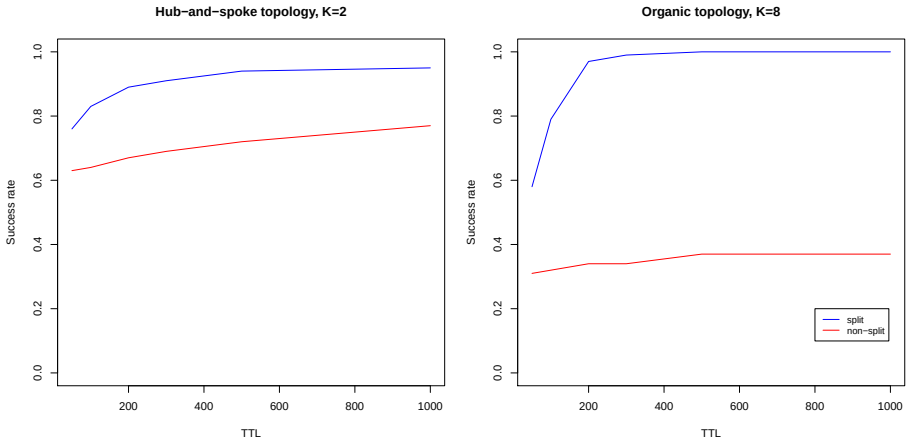

Fig. 7. Networks of 10000 nodes.

7.2 Split amounts

This section attempts to discover an optimal splitting strategy. Naturally, we need to answer what amounts should the payments be split in. The experiments described above used the strategy of breaking the payments into single units of value – 1s (unit payments), but the payment amounts in the simulation where generated with decimal precision. When the payment amount to be sent becomes less than 1, the rest is sent in 0.1s (one tenth of a unit payment).

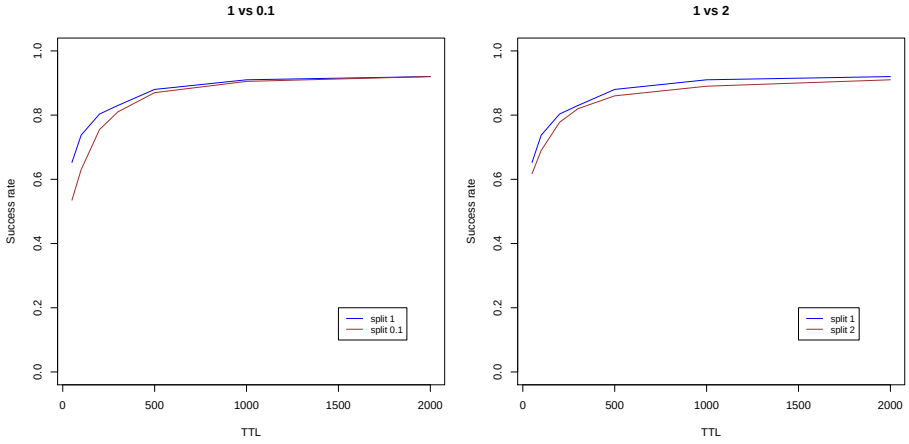

Fig. 8. Network performance depending on splitting amounts.

In this experiment we have introduced two alternative splitting strategies – sending bigger bulks of 2 units and smaller in 0.1s. The results are depicted in Fig. 8. Both strategies are of the same performance as the reference strategy. The minor deviations are explained by the not ideal precision of the simulation due to randomness introduced to preserve the network liveliness. The heavy tail of low valued TTLs for 0.1 strategy is explained by the fact that some high amount payments are split in the number of sub-payments higher than TTL, so they cannot be completed within such low TTL.

The actual amounts to be split in will be dictated by the network use. With the growth of the splitting unit amount the efficiency will start dropping. On the other hand, the sub-payments being too small bring operational overhead, therefore a balance between these two competing objectives needs to be established.

7.3 Path choice

A question of great interest that relates to payment routing is whether the choice of path selected for a particular payment affects the overall efficiency of the network and its liquidity. There are many on-going projects, Khalil et al. [21] for example, that suggest using re-balancing techniques in the fashion of cycle transactions to increase network liquidity. The idea is to re-distribute the channels' balances in the network, so a transaction of interest becomes possible. All done not changing the total balance of each individual. This suggests that having the same balances, the distribution of funds in channels matters.

Dandekar et al. [8] provide a strong contradiction to such a claim. They prove the following theorem for trust networks.

Theorem 1. *Let $(s_1, t_1), (s_2, t_2), ..., (s_T, t_T)$ be the set of transactions of value $v_1, v_2, ..., v_T$ respectively that succeed when the payment from s_i to t_i is routed along a path P_i. Then the same set of transactions succeed when the payment from s_i to t_i is routed along any other feasible path P'_i.*

In the words of the authors this "obviates the need to route payments along the shortest path". Obviously, not taking the implementation details of a particular payment network. Moreover, in their proof they state the following lemma.

Lemma 1. *For any equivalence class C of a given network, if a transaction (s, t, v) is feasible in some state $S \in C$, it is feasible in all states $S' \in C$.*

An equivalence class is defined by a generalized score vector, i.e. nodes' balances. A state is a distribution of the the balances in the channels. Paraphrasing the lemma, total balances of nodes in the network define the feasibility of a particular transaction, not the distribution of balances.

Consequentially, the choice of path does not affect the liquidity of the network, when dealing with atomic payments that are strictly ordered. Proving this theorem for split payments poses a bigger challenge. We omit the proof for this paper and only verify it experimentally. For this experiment two path selection algorithms were implemented. One imposes an order on candidate channels

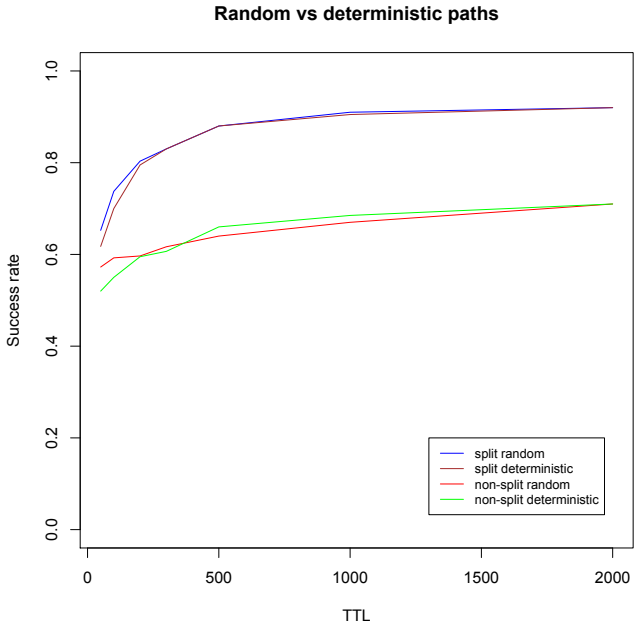

Fig. 9. Network performance with different path selection algorithms.

and always selects the first candidate. An individual node, by such strategy, always chooses the same channel until it depletes. The second algorithm randomly selects channels on each hop, keeping channels rather balanced out. Both algorithms ended up with exactly the same performance (see Fig. 9), supporting the theorem.

8 Discussion

This paper only presents the concept of split payments and demonstrates the performance gain it can yield. The technique needs an extensive discussion before it can take a certain implementation. One way the idea of split payments can be taken further is to abandon HTLCs and turn to script-less micro-payments. A script-less micro-payment is completely insecure and is basically a trustless and unenforceable request to forward a small amount payment. Any intermediate node is free to receive such a payment and refuse to forward it. If a node cheats and saves the payments for itself, the sending node punishes it by blacklisting it and never routing through it again. The premise here is that nodes' reputations worth more than the value of a micro-payments. Adaption of script-less micro-payments has many advantageous. Among them are the lower computational load on devices as no HTLCs have to be created, and removal of the limit

on the maximum number of payments in fly for one channels (the number of simultaneous HTLCS in one channel is currently restricted by transaction size limit [22]).

Since the form the split payments will take is not known, it is hard to estimate, at the moment, every aspect of the system that will be affected. For instance, splitting involves increased complexity which inevitably results in higher computational load on devices. On the other hand, many processes may be simplified with split payments. For example, abandoning scripting as described earlier in this section. Further in this section we consider split payments effect as if they were adopted in the Lightning network as it is now.

Naturally, the computational cost will increase many fold as there will be many more transactions. That is a severe restriction. None the less, this should increase the best effort time from initiating a payment to its confirmation only sublinearly, as sub-payment can be executed in parallel. Additionally, we foresee no communication overhead as all the hashes' pre-images can be generated and shared in one round.

Another possible complication is payment acknowledgement, which has been actively discussed by the community in the thread on the mailing list which presented AMP [17]. In the discussion there was found no general consensus on the matter. It is not known yet what level of inconvenience the splitting will bring to invoicing.

Important to note that in our study of split payments we disregard fees in the system. Whereas being a determinative part of the system, it does not affect the success rate of payments. It does, however, suffers consequences from adapting split payments. Currently, fees are $baserate + ratepertransactionsize$. Such approach lays a burden on splitting by paying multiple base rates. Should splitting be adopted, base rate component is likely to be abandoned.

Another problem with fees is that they have to be attached to payments, changing payment amounts. A node forwarding payments could then try to speculate its position on the route by analyzing payment amounts. A possible way around could be paying fees separately to each routing node prior to the actual money flow. Alternatively, the amounts to be split in could be not unit amounts sharp, but randomly chosen within a small window around unit amounts.

Finally, there is a failure mode when many simultaneous payments originate in one part of the network. When two payments, for instance, start being simultaneously cast through the same path in the same direction, it may happen that both of them will be partially executed while if being atomic payments one of them would get fully executed one would not initiate the execution at all. If that proves to be a real threat, the proposal has to be modified as follow: try sending a payment atomically first, if not possible split it up.

9 Future work

The introduced direction of research in the money flow networks is new and promising. A large number of questions needs to be answered. We are currently

working on some that we list below. The most interesting and the most important issue is the one of the network formation. It is yet to be studied what topology of the payment network is the most feasible and which one is the most efficient. It will be interesting to see if the optimal topology depends on the amounts invested in the network.

Fees will be a powerful instrument to dictate a certain behavior in the network. We are set to determine if a particular fee strategy can potentially improve the network properties, such as channel balancing and liquidity. For example, smart fee strategy could keep the network constantly balanced, resulting in shorter paths. Finally, path selection algorithm may not change the liquidity of the network, but they may well change the path lengths, which in turn can have other implications. For example, longer paths typically mean greater collateral risk of locking funds.

Game theory of node behaviors is another question to look at. Routing nodes will try to make the most out of their investments. Such behaviors should be predicted and analyzed how they would affect the global network properties. Last, but not least, various implementations of the payment networks have to be studied to test if there any crucial difference that affect their routing performance.

10 Conclusions

We have introduced the area of research focusing on payment networks and money flows. We have investigated an improvement in the design of the payment network, based on the split payment model. The new strategy has been experimentally demonstrated to substantially increase the liquidity of payment network. We have tested the path indifference theorem and showed it to be true for split payments. We have investigated what payment network topological characteristics tend to yield better liquidity. Another important contribution is the Lightning network simulator, named Blyskavka, that has been designed to be general-purpose and we expect it to be used in the future research work on payment networks.

References

1. Andrew Poelstra. Mimblewimble. 2016-10-06.
2. Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.
3. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

4. Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. *Draft, version 0.5.9.2*, 2016-01-14.
5. Arpita Ghosh, Mohammad Mahdian, Daniel M. Reeves, David M. Pennock, and Ryan Fugger. Mechanism design on trust networks. In Xiaotie Deng and Fan Chung Graham, editors, *Internet and Network Economics*, pages 257–268, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
6. Dean Karlan, Markus Mobius, Tanya Rosenblat, and Adam Szeidl. Trust and social collateral*. *The Quarterly Journal of Economics*, 124(3):1307–1361, 2009.
7. Paul Resnick and Rahul Sami. Sybilproof transitive trust protocols. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 345–354, New York, NY, USA, 2009. ACM.
8. Pranav Dandekar, Ashish Goel, Ramesh Govindan, and Ian Post. Liquidity in credit networks: A little trust goes a long way. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, EC '11, pages 147–156, New York, NY, USA, 2011. ACM.
9. Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *CoRR*, abs/1709.05748, 2017.
10. Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 455–471, New York, NY, USA, 2017. ACM.
11. Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. Flare: An approach to routing in lightning network, 2016.
12. C. Grunspan and R. Pérez-Marco. Ant routing algorithm for the Lightning Network. *ArXiv e-prints*, June 2018.
13. Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A Simple Layer2 Protocol for Bitcoin. https://blockstream.com/eltoo.pdf, 2018.
14. Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Multi-hop locks for secure, privacy-preserving and interoperable payment-channel networks. *IACR Cryptology ePrint Archive*, 2018:472, 2018.
15. Dmytro Piatkivskyi, Stefan Axelsson, and Mariusz Nowostawski. *A Collusion Attack on the Lightning Network — Implications for Forensics.* 2017.
16. Jordi Herrera-Joancomartí and Cristina Pérez-Solà. *Privacy in Bitcoin Transactions: New Challenges from Blockchain Scalability Solutions*, pages 26–44. Springer International Publishing, Cham, 2016.
17. Olaoluwa Osuntokun. AMP: Atomic Multi-Path Payments over Lightning. 2018-02-06.
18. Chris Pacia. Lightning Network Skepticism. https://goo.gl/obxpQm, Dec 2015.
19. reducing the number of blockchain transactions used by the LN, and the fees paid to confirm them. 2017-12-21.
20. Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, July 2005.
21. Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 439–453, New York, NY, USA, 2017. ACM.
22. Payment channel congestion via spam-attack. 2017-03-24.

A An example of the configuration file

```
EXPERIMENT_NAME=TTL200
TTL=200
IS_SPLIT=FALSE
K=2
NETWORK_SIZE=1000
MIN_CAPACITY=1
MAX_CAPACITY=10
MIN_COST=1
MAX_COST=10
ALGO=PERIPHERAL
PAYMENT_FREQUENCY=25
DELIBERATE_AMOUNT_SPLITTING=TRUE

## SET VALUES FOR THE CORE NETWORK ##
CORE_NETWORK_SIZE=20
CORE_ALGO= UNIFORM
CORE_K=2
CORE_MAX_CAPACITY=5
CORE_MIN_CAPACITY=5
MIN_COST=1
MAX_COST=10

## MISCELLANEOUS
LOGGER_LEVEL=TRACE
GRAPH_FILE=graph1.ser
```

B An example of the simulation output

```
20180607EXPERIMENT-same-graph.jar -Xms4096m -for 300 -
    repeat 5 -filename TTL50.config
2018/06/08 13:58:00
1528455480728
{CORE_MIN_CAPACITY=5, PAYMENT_FREQUENCY=25, NETWORK_SIZE
    =1000, DELIBERATE_AMOUNT_SPLITTING=TRUE, IS_SPLIT=
    FALSE, GRAPH_FILE=graph1.ser, CORE_MAX_CAPACITY=5,
    ALGO=PERIPHERAL, LOGGER_LEVEL=DEBUG, MIN_CAPACITY=1,
    CORE_ALGO=UNIFORM, CORE_NETWORK_SIZE=20, TTL=50, K=2,
    MAX_CAPACITY=10, CORE_K=2, EXPERIMENT_NAME=TTL50}
InvestedInNetwork: 21766.0
InvestedInCoreNetwork: 210.0
TotalTx: 3356
SuccessfulTx: 1805
FailedTx: 1551
```

```
PendingTx: 0
SuccessRate: 0.54
FailureRate: 0.46
EstimatedSuccessRate: 0.54
EstimatedFailureRate: 0.46
```