On the Inefficacy of a Distributed Lightning Network

Daniel Adkins

May 2018, Block 522171

"I'm sure that in 20 years there will either be very large transaction volume or no volume."
- Satoshi Nakamoto

1 Background

NOTE: *On August 1, 2017, Bitcoin forked into two divergent protocols and resultant chains. The fork resulted from dispute over implementation of Segregated Witness. This paper focuses predominantly on the chain with Segregated Witness, which I will refer to as "Bitcoin Core" or simply "Bitcoin". All historical assumptions and statistics are derived from the state of the network at the time of final editing, at which point block height is 522171.*

In the summer of 2017, Bitcoin's scaling issue came to the forefront of the public and media hivemind, with miner fees reaching $10+ just to transact on the blockchain. Proponents of Bitcoin as a currency, instead of just as a store of value, identify the issue as detrimental to the network's scalability and adoptability. This scaling issue centers around the 1MB limit of data that can be included in each block along with the inherent scarcity of blocks (which can only occur approximately once every 10 minutes due to mining difficulty). Block congestion and transaction fees are the effect, requiring users to pay miners in order get their transactions included in the full or near-full blocks.

With the addition of Segregated Witness (SegWit) to Bitcoin Core came the long-awaited transaction malleability fix, allowing signatures to be separated from calculation of the $TxId$ value in a transaction. Previous to this development, there existed a limitation in Bitcoin in that parties were able to change their signature and thus change the $TxId$ before broadcasting a transaction to the network, so creating transactions that depended on another's $TxId$ was vulnerable to attacks depending on the context. However, with this transaction malleability fix, transactions now can have a set $TxId$ without risk of $TxId$ being manipulated and changed by an attacker editing their signature before broadcasting to the blockchain.

This key development in fixing transaction malleability allowed for the "Lightning Network" to come to fruition. Described by Poon & Dryja in the LN whitepaper [5], the Lightning Network aims to solve the issue of Bitcoin scaling, while offering a method for feeless, instant micropayments. Lightning Network focuses on moving a significant number of transactions to occur off-chain, where only essential information is recorded permanently on the blockchain. Removing intermediary transactions between two users and only recording the final balance on the blockchain results in increased privacy and potentially substantial decreases in data appended on the blockchain, depending on how often two users may transact with one another.

This paper contributes an explanation and analysis of the bidirectional payment channels (Section 2), a novel analysis of Lightning Network routing and random graph construction (Section 3), and an evaluation of the current development of the network (Section 4).

2 Bidirectional Payment Channels

The term "Lightning Network" (LN) describes the set of bidirectional payment channels that process payments off-chain through Lightning protocol. Lightning Network channels consist of four parts: a funding transaction, commitment transactions, Revocable Sequence Maturity Contracts, and Breach Remedy Transactions.

Funding Transaction

The funding transaction of a Lightning channel is the initial commitment of funds to be utilized in payments between two peers. The funding transaction is a 2-of-2 multisignature construction with outputs that map to each of the participants' initial inputs.

For example, consider Alice and Bob, who wish to create a Lightning channel with one another in order to enable consistent micropayments while avoiding fees. Alice and Bob will each initially commit some funding balances, B_{Alice} & B_{Bob}. The funding transaction will then be a 2-of-2 multisignature output that sends B_{Alice} to Alice and B_{Bob} to Bob upon closure of the channel through a commitment transaction. Once the funding transaction is created, its $TxId$ value can be calculated (without risk of malleability, thanks to SegWit) and used to construct the commitment transactions. Consider this funding transaction the "opening" of a Lightning channel, with the commitment transactions being the subsequent exchanges of value within the channel.

Commitment Transactions

The funding transaction represents a static balance sheet between two users. Commitment transactions are needed to be able to update this balance (the outputs of the funding transaction), in order to allow for the flow of money between the two peers. Once the funding transaction is created and broadcast to the blockchain, subsequent commitment transactions can be created and signed by both users. These transactions take as input the outputs of the funding transaction, and re-map the outputs to the two participants of the payment channel with the updated balances.

For example, if Alice wants to send 0.1 BTC to Bob and has this amount of balance in the payment channel, the associated commitment transaction would take the funding transaction's output and, with signatures from each user, send $B_{Alice} - 0.1$ BTC to Alice and $B_{Bob} + 0.1$ BTC to Bob. So, if the commitment transaction is broadcast, the channel will close with Bob having 0.1 BTC more than he started with. Once a commitment transaction is signed by both users, either party can broadcast it when they wish to close the channel and settle the funds, so a commitment transaction provides trustless assurance that the correctly balances of money are in fact available to each user if they were to publish it on the blockchain. However, if the commitment transaction is not published, Alice and Bob can create a new commitment transaction that represents a new balance of money between the two. From this, we can see how a multitude of feeless transactions can created without broadcasting any to the blockchain, by simply creating a new commitment transaction for every new transaction between Alice and Bob.

Since we do not need to publish intermediary balance updates (old commitment transactions) to the blockchain, we can thus neglect old commitment transactions and throw them away. However, here we run into a problem: what happens if there are multiple commitment transactions and one of the peers chooses broadcast an old commitment transaction? In this case, the malicious party has access to an old commitment with valid signatures, which still validly points to the open funding transaction (since no subsequent commitment transactions have been published to redeem the funds yet as the channel is still open). An uncooperative party will often be directly economically incentivized to publish old commitments. This would occur in any situation in which the uncooperative party pays funds to the victim party, since the old commitment would entitle the

uncooperative party to a higher balance than the newer commitments would upon publication to the blockchain.

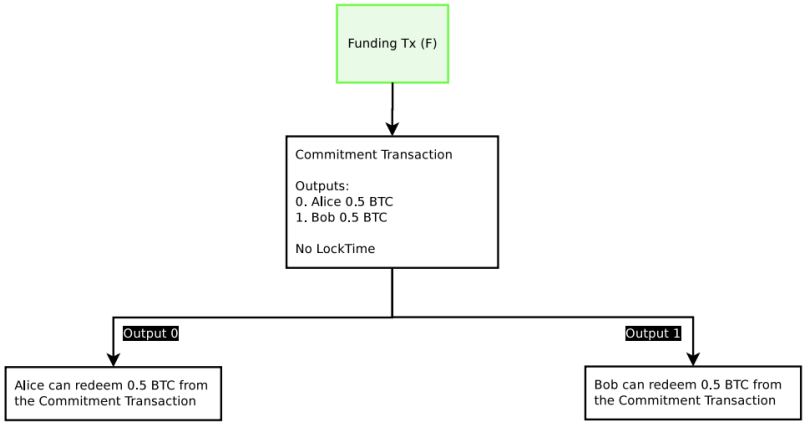

Figure 2a: A naive commitment transaction diagram (from LN whitepaper)

When we create a single commitment transaction that is signed by both parties and shared between both parties every commitment round, this makes it impossible to ascribe blame for a party publishing an old commitment transaction without the other party's consent. We can fix this problem by modifying the commitment construction to allow for proof of blame. In the modified construction described by the LN whitepaper, the participants will instead create two commitment transactions per commitment round. One commitment transaction, $C1a$, will be created and signed solely by Alice, which she will then send to Bob. Bob will do the same, creating and signing the commitment transaction $C1b$ for commitment round 1. As a result, each user has a half-signed commitment transaction containing the signature of the other party, so at any moment they can complete the transaction by signing with their own signature and broadcast to the blockchain to close the channel. In this case, there is now proof of which party broadcast the transaction to the blockchain (since $C1a \neq C1b$), which, using two types of scripts (RSMCs and BRs), can be used to enforce that neither party publishes old commitments.

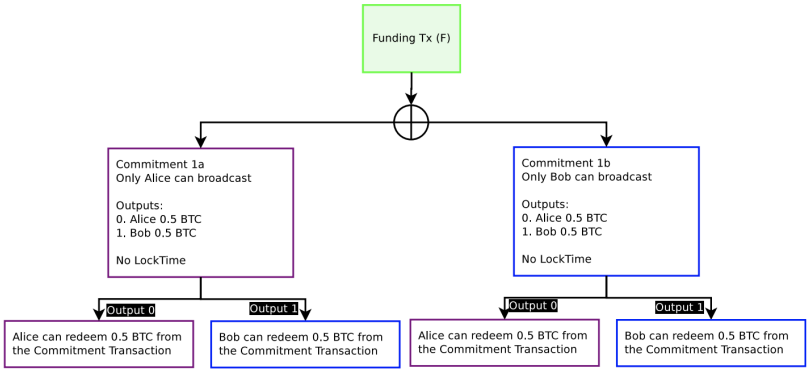

Figure 2b: A pair of commitment transactions that allow blame to be ascribed (from LN whitepaper)

Revocable Sequence Maturity Contracts

Since each commitment transaction now has an identity tied with it (where either Alice broadcasts $C1a$, or Bob broadcasts $C1b$), we can ascribe certain rules to the issuer of the commitment transaction to penalize misbehavior, i.e. publishing an old commitment transaction. First, the Lightning Network punishment system constructs Revocable Sequence Maturity Contracts (RSMCs) with every commitment transaction. These RSMCs are scripts that regulate the outputs of a commitment transaction when the commitment transaction is published.

This regulation is done by placing a maturity requirement on the publisher's output of the commitment. This maturity requirement would be placed on the outputs of $C1a$ that go to Alice if $C1a$ were published, or on the outputs of $C1b$ that go to Bob if $C1b$ were published. The maturity requirements describe a certain "gestation" period, in which the outputs of the commitment are not spendable until some block height away from the initial commitment transaction is reached. For example, if the maturity period for a commitment was set as 144 blocks (approximately 24 hours) and that commitment was published by one of the participants to close the channel, the participant that closed the channel would not be able to spend their funds (output of the commitment transaction) for the first 144 blocks after the transaction were included in the blockchain. In current Lightning protocol, the party who did not publish the commitment transaction will be able to spend their money immediately.

The number of blocks that the publisher's money is locked up for, *nSequence*, is agreed upon at the discretion of the two channel participants when the commitment transactions are signed and sent to one another. Once the commitment transaction is published on the blockchain, the parties can agree to nullify *nSequence* once it is agreed that fraud was not committed. However, if one party detects fraud from an old commitment transaction being published, this party can then publish a breach remedy transaction.

Breach Remedy Transactions

A Breach Remedy Transaction (BR) is the mechanism by which revocation of funds occurs for malicious parties in the event that one party publishes an old commitment transaction. As described before, when money moves in the Lightning channel and new balances are agreed upon, a new commitment is agreed upon and old commitments must be nullified. Since we have ascribed blame and identity to commitment transactions by splitting them up into party-dependent transactions, and have put a timelock on redeeming funds with the RSMC, we can now use a BR transaction to punish a participant who publishes an old commitment transaction.

The process for creating BR transactions occurs every time an old commitment should be nullified, which is when new commitments are agreed upon. The commitment construction is modified to include an associated transaction that each party receives, the BR. The BR entitles the party that publishes it to 100% of the output funds of the associated commitment transaction.

If Alice and Bob agree on a new balance, they create two new commitment transactions $C2a$ and $C2b$. Alice signs and broadcasts $C2b$ and the associated $BR1b$ to Bob, and Bob signs and broadcasts $C2a$ and $BR1a$ to Alice. As a result, if Bob publishes $C1b$ to the blockchain which contains the non-updated balances, Alice can publish $BR1b$ to Bob within the maturation period of *nSequence* blocks, which sends the outputs of the commitment transaction entirely to Alice. Therefore, since at every new commitment round both parties construct breach remedies for old commitments and give the BR to the counterparty, if any party were to publish an old commitment transaction, the cooperative party could publish the breach remedy and receive the entirety of the funds in the channel.

Implications

This construction of bidirectional, off-chain payment channels opens up a number of valuable use cases. Streaming payments (for example, paying for electricity as you use it, or paying for music streaming services as you listen), microtransactions, increased privacy and significantly less data on the blockchain all arise as possibilities from Lightning channels. The use cases are endless and the ability for instant and feeless transactions are central to the electronic economy of tomorrow.

Criticisms

The payment channel construction described above, with RSMC and BR scripts, requires active monitoring for fraud detection. That is, users have to constantly be watching the blockchain for old commitment transactions being published by the counterparty. In the event of one peer going offline or failing to monitor, a malicious party can easily publish an old commitment transaction and potentially receive their funds after maturation if the victim remains offline/inactive long enough. This failure to publish the breach remedy could come as a result of rational ignorance by the victim, since malicious parties are likely to be more invested in committing fraud than cooperative parties will be in preventing it. On the other hand, the victim party could be targeted through a DDoS or similar network isolation attack, in which the malicious party actively keeps the victim off the network for a period of time as the commitment transaction output matures.

With these vectors for attack, users will be incentivized to agree on *nSequence* values that are long in order to allow for a greater buffer in the event that one party is fraudulent. However, these *nSequence* values cannot be impractically long because of the time-value of money; there is a cost that comes with agreeing to longer maturation periods, since money will not be as readily spendable when the channel is closed.

Participants may delegate their breach remedy responsibility to one or multiple third parties, who are trusted to remain online reliably and continue monitoring the blockchain. When fraud is detected, these parties can publish the user's breach remedy transaction, which will revoke the funds and still be sent to the user. However, this responsibility clearly cannot come altruistically, and will require payment of some level. There will likely be a form of bounty system that develops, where trusted third parties monitor for fraud and publish breach remedies in exchange for commission on the saved money or for a flat-rate fee. Both situations put users with lower-staked Lightning channels more at risk.

Routing

Lightning Network channels can also have associated hashed timelock contracts (HTLCs). HTLCs allow users to redeem their money from another user if they can prove they have received some hash preimage within some set timespan (that is, they can generate some random R image from input hash H).

This allows for the possibility of multi-hop routing between connected channels. A series of HTLCs can be constructed in a route of channels such that each timelock is decremented from the previous channel's timelock, beginning at the initial sender and going to the recipient of the cross-network transaction. The construction of these decrementing timelocks and HTLCs is expanded upon in more detail in Herlihy's formalized evaluation of Atomic Cross-Chain Swaps [4].

Implications

Consider the situation in which Alice would like to pay Carol, but Alice does not have a Lightning channel open with Carol. However, each of Alice and Carol have a channel open with Bob, who, through HTLC construction, can act as a trustless intermediary for the payment between Alice and Carol. Upon HTLCs being constructed and verified by each user automatically, the hash secret

is revealed by the recipient party, allowing Carol to redeem money from Bob, and Bob to redeem money from Alice. As a result, the flow of value effectively traveled from Alice to Carol with Bob acting as a router.

The term "Lightning Network" describes a connected graph in which any user can route a payment to any other user in the network while only maintaining a few open channels. The advantage to this is the avoidance of creating funding transactions, which require an on-chain transaction and thus associated overhead (confirmation time, potential fees).

Criticisms

This example requires Bob to have the funds required to transfer Alice's value to Carol, which will not always be the case in an economic system with disparate channel stakes, making long-distance hops more difficult. Furthermore, routing nodes also need to stay online for the duration of contracts whenever needed for routing.

Similar to before, the locking of funds with HTLCs results in a time-value cost, since routing a peer's funds requires some of the router's funds to be locked up for a short period of time as the HTLCs execute. The implications of this are that there will be incentive to charge fees for routing and agreeing for money to be locked up for a period of time.

Furthermore, the Lightning Network paper makes a dangerous assumption when they state that "with a sufficiently robust and interconnected network, the fees should asymptotically approach negligibility for many types of transactions". As I will argue in the next section, such a "sufficiently interconnected network" may prove difficult or impossible to grow.

3 Topological Analyses

"...with a sufficiently robust and interconnected network, the fees should asymptotically approach negligibility for many types of transactions." [5]

The above quotation is, in my eyes, the most difficult and consequential assumption made in the Lightning Network whitepaper. The Lightning Network, with its current protocol and routing requirements, cannot maintain a distributed, highly connected network in which nodes will be able to avoid fees and trust by creating multiple long-distance, multi-hop routes. Instead, it is more likely that the network will end up with centralized, trusted, and most importantly, fee-requiring hubs that facilitate routing.

As a complex graph projected to contain up to millions of nodes in its adopted stage, it is difficult to accurately and fairly generalize the topology of the Lightning Network. However, in this section, I will visit the extreme cases and argue why it is more likely that Lightning Network topology will trend towards centralization.

There are two sides of the spectrum of topology that the Lightning Network may taken on: distributed and centralized. However, first, I will show that a near-complete graph in which every node maintains a high number of channels open is not possible to create.

Completeness

From the Lightning Network Github documentation (`https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md#appendix-a-expected-weights`), the expected weight of a Lightning Network funding transaction, which opens a channel between two peers, is expected to be at least 800 bytes. Under SegWit implementation, blocks can contain a total block weight of 4MB. As a result, a block can at most contain 5000 funding transactions, generously assuming that the block is full when published, there are no other transactions included in the block, and the funding transactions are all on the low end of expected weight.

Furthermore, while it is extremely difficult to assess the number of total Bitcoin users that exist, we know that there are 28.5 million wallets with at least 0.001 BTC in them, as well as 13 million Coinbase users at the time of writing. As a result, it is fair to assume with some confidence that there are at least 10 million Bitcoin users at the moment. Since 5000 funding transactions can be created per block, it would thus take 2000 blocks, 20000 minutes, or 13.888 days to open a Lightning channel for every Bitcoin user with another unique user (resulting in 2 channels per user), so it would take 6.9444 days for one half of the network to each open one bidirectional channel with the other half of the network. In the scheme of adoption, this is a reasonable amount of time to take to open up one channel per user. However, as the network grows to be "highly connected" or near complete with the number of channels per user becoming magnitudes higher, it becomes much slower (on the order of years for every user to have 100 open channels) to create these funding transactions. It is clear that we must optimize for connection with fewer channels per node.

Distributed Topology

Distributed topology describes a model where users are able to reach almost all other users through multi-hop routing across a series of nodes while avoiding high-degree nodes, which would be centralized routing hubs that dominate a large number of channels. This section contributes the use of the Erdős–Rényi model to analysis of Lightning Network topology, argues that it is in fact possible to create a randomly connected Lightning Network with the minimum number of channels in an efficient number of blocks, but that it is not plausible considering a number of restrictions to our LN graph construction. Furthermore, this section will argue that the same restrictions make the multi-hop routing across the network infeasible in a decentralized manner, even if we can create a connected graph in the ideal random setup.

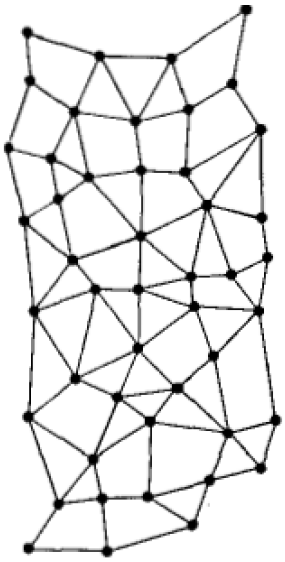

Figure 1: A uniform distributed graph topology (note that this is not necessarily what our random graph will look like)

Let us construct the random topology according to the *Erdős–Rényi* model for random graphs, that is, we create a $G(n, p)$ which is a graph with n nodes and a probability p of an edge between two nodes occurring in the graph.

The Erdős–Rényi equations [3] show that the graph is very likely to be disconnected if $p < \frac{ln(n)}{n}$, and the graph is very likely to be connected if $p > \frac{ln(n)}{n}$. As a result, for the graph to be connected, we have a minimum p value (edge probability) of $p_{min} = \frac{ln(n)}{n}$.

Since p represents the probability of an edge occurring between two nodes in the graph, the number of edges for each node $c = \frac{np}{2}$, and the total number of edges in the graph is thus $E = c * n = \frac{n^2 p}{2}$, and $p = \frac{2c}{n}$.

So, our minimum number of channels per node for a connected topology with our random distribution of edges c_{min} will be:
$\frac{2c_{min}}{n} = \frac{ln(n)}{n}$
$c_{min} = \frac{ln(n)}{2}$

The minimum total number of channels required to be open for a connected G is equal to the average minimium number of channels c_{min} times n, so $c_{total} = \frac{ln(n)*n}{2}$.

The low-end estimate for the number of bytes of data d required to create these funding transactions $d_{total} = c_{total} * 800 = 400nln(n)$. With a block weight of 4000000 bytes per block, constructing a connected topology will take at least $b(n) = \frac{400n*ln(n)}{4000000} = \frac{nln(n)}{10000}$ blocks.

For almost all reasonable values of n, $b(n)$ is a reconcilable number of blocks to create funding transactions in that it would not take unreasonably long to construct the minimum number of channels.

Since we have shown how to construct a connected graph, we are now able to create paths from any node to any other node. For any given node in the network, there are $n-1$ other nodes that the node should be able to route to (for large values of n, we can simplify this to n instead of $n-1$). Furthermore, since every node has c random edges to other nodes, there will be some value h representing the number of hops needed to be made that reaches all n nodes in the graph. If we are constructing a route from one node to all others, we should visit first the c immediate neighbors of the starting node, then the c immediate neighbors of those nodes, until we have completed h hops so that $c^h = n$, meaning we can visit every node in the network on average with h hops. Thus, $n = c^h$, and $h = \frac{ln(n)}{ln(c)}$.
Since $c_{min} = \frac{ln(n)}{2}$, $h = \frac{ln(n)}{ln(\frac{ln(n)}{2})}$.

Therefore, for a randomly connected network of 1 million users, $c_{min} = 6.9$ and $h = 7.14$. This is where we face heavy drawbacks and limitations to multi-hop routing. While it is likely that we would be able to route with 2 hops in a more densely connected graph, increasing number of hops beyond 3 or 4 likely will result in significant problems for our routing system.

The first limitation that we will face in routing on the scale of $7-8$ hops is fund availability along the route. For Alice to route to Carol in the previous example, they required an intermediary Bob who held the amount of Bitcoin in his channel with Carol equal to the amount that Alice wanted to send to Carol. When we attempt multi-hop routing, we require a series of users that each have enough stake available to route. In this construction, we have shown that we require at least $c_{min} = \frac{ln(n)}{2}$ channels for routing, and in an active network like the LN aims to be, these routes will be constantly used. Thus, a node's stake will be locked up by constantly formed HTLCs for routing

between channels. If every node has some average balance B_{avg} and are routing for some significant proportion of their c_{min} channels at any time, when a node is being routed through by all of its neighbors using HTLCs, the most that it can route for any neighbor is $\frac{B_{avg}}{c_{min}}$.

Furthermore, the variance between channels used in routing greatly inhibits the ability to route in many hops. Stake in each channel is required to be available to route across the network, so to move large amounts of money requires equally large staked channels to move through in order to transfer value across the network. As it is likely that nodes are not able to designate all of their available funds at any time, moving large amounts of money requires split routes among many different paths. With many different paths handling money with different HTLCs, HTLC failure can result in lossy delivery (such as in Internet routing). Furthermore, larger movements of money place significant strain on routing in a distributed network since each node already only has a small fraction of their funds available for routing at any given time. In an economic system where users are likely to have very disparate levels of currency available, routing large amounts of money becomes a complex issue that needs to be split up among many nodes and rely on a significantly higher number of HTLCs and paths to deliver money.

Additionally, if we choose to optimize for a low number of channels open (in order to create a highly connected network in the most efficient block weight possible), we suffer the tradeoff of lack of routing strain as described before which may result in a fee network to route. With a multi-hop system on the scale of 7 or more hops across a network, significant fees could be incurred as many intermediary nodes delegate their funds to route for a cross-network transaction, hindering the use case of cross-network microtransactions.

On top of these issues that may become evident with a multi-hop routing system, this section allowed a significant number of generous assumptions for the possibility of creating routes. First, as pointed out already, channels will have disparate amounts of stake, which greatly limits the ability to route across long distances. Additionally, in the natural development of payment channels, the creation of the graph edges will not be random but instead trend toward more channels opened with early users (see "Section 4: Current State of the Lightning Network"), so peripheral nodes will be much more difficult to connect in the graph. Users also will not remain online to route other users' transactions, limiting the number of paths available greatly at any given time.

In summary, it is possible in theory to create a connected, distributed topology based off of random edge generation if the entire network were to coordinate to construct this graph in the most efficient manner possible. In this scenario, each node would generate $c_{min} = \frac{ln(n)}{2}$ channels with randomly chosen peers. New participants should join the network by choosing c_{min} peers to initialize edges with. This would provide privacy benefits in that transaction data would be distributed and transferred long distance, making bad actors unable to discern significant information from routing requests. However, there are a number of drawbacks that make this difficult, such as economic disparity, unavailability of funds depending on network activity which could result in a fee network, and difficulty of coordination. There is a trade-off evident between number of channels opened per node, which strains the blockchain and divides nodes' funds even further, and ability to construct longer distance routes. These longer routes would have even lower amounts of BTC transferable, since nodes have their funds divided among more channels.

If this network topology were developed, there are a number of proposed methods for creating routes that would attempt to satisfy the requirements of routing across the network with confidence based off of an analysis of available funds and channel availability, such as SpeedyMurmurs [1]. These path-based tranaction protocols are out of the scope of this paper.

Centralized Topology

More likely is that the Lightning Network will evolve into a "centralized" topology as it scales further due to the challenges of creating a distributed network.

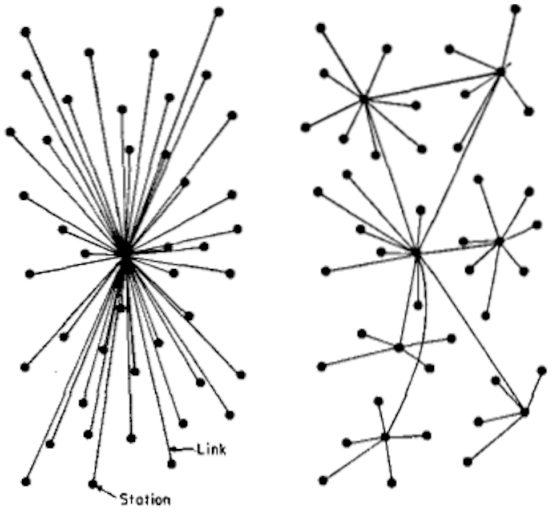

Figure 2: Examples of two logically centralized networks that rely on dominant, central routing hubs)

With multi-hop routing, intermediary nodes have a number of responsibilities to facilitate their part of routing a transaction. First, intermediary nodes have to construct and stay online for the duration of HTLCs, so that they can transfer the secret across the network and assure that the HTLCs do not default and cause the route to fail. Second, intermediary nodes have to be able to find a route from sender to recipient. Third, intermediary nodes need to have the amount of stake needed to facilitate the transaction available. Finally, intermediary nodes need to commit their funds to a timelock for the duration of the HTLCs, rendering their funds temporarily unavailable.

Staying online for the duration of HTLCs presents a significant difficulty in creating a distributed network. It is clear that every node will not stay online to route other transactions altruistically, and it is much more likely that, as the LN whitepaper concedes, specialized hub nodes will stay online at all times and peripheral nodes will be able to go offline and use the larger hub nodes for routing.

Finding a route from recipient to sender also presents the same difficulties as described before. Due to the responsibilities and limitations to acting as an intermediary routing node, a number of nodes will be unavailable as intermediary nodes. Since these requirements can be viewed somewhat as a binary cutoff (Can the node stay online? Does the node have enough stake to handle network activity?), there will be a small set of nodes that are able to act and scale as intermediary nodes, where the majority of nodes will not. As a result, there will only be a few nodes involved in actual routing, each of which have a high number of channels open, so the number of hops required to get from peripheral node to peripheral node is greatly decreased.

Having the amount of stake needed and being able to commit that stake to a timelock for routing is also a significant economic requirement. Intermediary nodes cannot route more BTC than they have available, so the most efficient routing nodes will be nodes that can easily maintain high levels of available BTC and delegate their available funds for the purpose of routing. These types of nodes will come from highly staked actors, of which there are few. As a result, it will be most efficient to route through wealthy, centralized actors for which the time-value of money is less.

While periods of high network activity will likely incur higher fees for routing through these highly staked nodes, the fewer hops required to send money between the peripheries of the network, in addition to the general higher availability of money and lower opportunity cost of routing any given small transaction provided by centralized nodes, will result in lower fees compared to a distributed system in situations where the Lightning Network is being heavily strained by routing.

The key drawback here is privacy. While routing can remain trustless through HTLC

construction, so central nodes don't necessarily control your money, significantly more information can be learned about a transaction depending on proximity of sender and recipient to the routing nodes. In the most extreme cases, Alice will route her money through central hub Bob to Carol, and since Bob is an immediate neighbor with both the sender and recipient and the route remains the singular movement of money, he can discern and record information about the transaction. In a centralized network, this could come from regulatory agency pressure, as well as incentive for hubs to learn about transactional patterns. There are developments that can be made to enhance privacy in a routing system that are out of the scope of this paper, but can be studied through similar routing networks such as Interledger Protocol [6].

4 Current State of the Lightning Network

Statistics from the current topology of the Lightning Network support many of the previous arguments and findings. The following statistics were published on in April of 2018 by Medium user CryptoMedication [2]. They represent an accurate snapshot of the Lightning Network at the beginning of April, and the topology of the network can be explored further at `https://explorer.acinq.co/`.

There are 7.2 channels per node open in the Lightning Network. As shown previously, channel counts this low are feasible to maintain as the number of nodes in the network increases. However, the 90th percentile for channel counts is 15 open channels, whereas the 50th percentile for channel counts is 2 channels. This is a heavy indicator that routing through the Lightning Network would currently be heavily centralized, as argued previously.

The average stake of Bitcoin per channel is 263051 satoshi, or 0.00263051 BTC. However, the 50th percentile of channel stake is only $\frac{1}{5}$th of this average stake, residing at 50000 satoshi. From this we can conclude that long-distance routing that involves intermediately staked channels would have to be split up and rely on a number of different paths to deliver the full value of a transaction, whereas it would be more efficient to route through the more wealthy channels in fewer hops and paths.

The transitivity of the network is a measure of the ratio of actual triangles present in the graph (three nodes all interconnected with one another) to the potential triangles present in the graph. A high number of triangles represents a amount of route variability, whereas low transitivity would signal central routing hub presence, as there are fewer triangles formed to route around central hubs. While determining what transitivity correlates to what level of practical centralization is difficult, it is worth monitoring the trend of transitivity as the network develops over time. The transitivity of the network at the time of article publication was 0.087, or 8.7%, which marked a decrease from the 11.1% transitivity value in February, when the Network was in its infancy.

Figure 3: ACINQ topological rendering of network, with 889 *nodes and* 2367 *channels*

The ACINQ graph backs up the fact that early edges in the graph will centralize around heavy hubs at the beginning instead of developing in a randomized pattern, with the node associated with `yalls.org` having 487 channels open out of a total 2367 channels on the network. Similarly, IslandDSP has 187 open channels.

5 Conclusion

The Lightning Network offers a trustless, extremely efficient construction for opening bidirectional payment channels. These payment channels allow for the majority of transactions between two users to be abstracted off of the blockchain, which should provide needed relief for the Bitcoin network. Furthermore, Lightning channels have privacy and speed benefits as a result of most transaction information only being shared between the two participants, instead of having to be gossiped and agreed upon across the network.

Through hashed timelock contracts (HTLCs), the Lightning Network is also able to route value between users who do not have a channel open, but have connecting routes of channels through shared intermediary nodes. While this may provide a level of efficiency in routing over a small number of intermediary nodes, long-distance network routing faces a number of limitations, such as graph setup, uneven stake in channels, and availability of users to be online and willing to delegate funds.

We can optimize graph setup by following the *Erdős–Rényi* model when constructing the network and joining as a new node. By this random graph setup in which nodes open channels with other network participants based off of a small random probability $p = \frac{ln(n)}{n}$ for each edge, we

can feasibly and confidently create a fully connected network with the fewest number of channels possible. The tradeoff between the disadvantages of increasing the number of channels opened versus the benefits of decreasing the number of hops across the network is an open question requiring closer evaluation.

However, we still face the issues of uneven staking and unavailability of users. As a result of these issues and the natural development of payment networks, the Lightning Network will likely trend towards increasingly centralized routing hubs for cross-network payments. These payments will occur when it is more economically viable to route across the network instead of opening a direct channel on the blockchain and paying miner fees, so this hub routing will likely manage microtransactions. We can expect a fee network to occur if routing is sufficiently in demand (perhaps because fees become significant on the blockchain) as a result of the time-value of money and responsibilities required of hubs.

The current state of the network reflects this prediction and is worth monitoring. While centralization allows for efficiency benefits that cannot be easily attained otherwise, there must be a focus on privacy on the Lightning Network so that transaction data is less revealing.

Lightning Network implementation represents an existential moment in the development of Bitcoin protocol and the history of money, providing, for the first time, a vector for millions of people to trustlessly purchase a cup of coffee, stream payments, and send money across the world in mere seconds.

References

[1] Stefanie Roos et al. "Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions". In: (2018). URL: https://www.ndss-symposium.org/wp-content/uploads/sites/25/2018/02/ndss2018_09-3_Roos_paper.pdf.

[2] CryptoMedication. *Lightning Network Status Report: MAJOR Flaws and Topology Concerns.* 2018. URL: https://medium.com/cryptomedication/lightning-network-status-report-major-flaws-and-topology-concerns-ae2c9f0d1be.

[3] Paul Erdős and Alfréd Rényi. "On Random Graphs I". In: (1959). URL: http://snap.stanford.edu/class/cs224w-readings/erdos59random.pdf.

[4] Maurice Herlihy. "Atomic Cross-Chain Swaps". In: *CoRR* abs/1801.09515 (2018). arXiv: 1801.09515. URL: http://arxiv.org/abs/1801.09515.

[5] Joseph Poon and Thaddeus Dryja. "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments". In: *Lightning Network Docs* (). URL: https://lightning.network/lightning-network-paper.pdf.

[6] Stefan Thomas and Evan Schwartz. "A Protocol for Interledger Payments". In: (2014). URL: https://interledger.org/interledger.pdf.