A Survey into the Bitcoin Scaling Dilemma

Raja Atluri
University of Pennsylvania, School of Engineering and Applied Science
EAS499: Senior Capstone Thesis
Advisor: Brett Hemenway
April 25, 2018

Introduction: The Great Bitcoin Scaling Debate

Bitcoin was conceived with the vision of one day usurping major credit card networks such as Visa in capacity and scalability. Pseudonymous creator of Bitcoin, Satoshi Nakamoto, himself explained in April of 2009, "the existing Visa credit card network processes about 15 million Internet purchases per day worldwide. Bitcoin can already scale much larger than that with existing hardware for a fraction of the cost" (BitcoinTalk.com). Still, an even nine years later, bitcoin transaction levels remain orders of magnitudes short of that target, processing on average just over 200,000 transactions per 24 hour period year to date as of 2018, or ~140 transactions per second (Blockchain.info). In this paper, the underlying causes behind scalability bottlenecks are introduced and a variety of compelling solutions are discussed and assessed.

Origins of Bitcoin's Scaling Dilemma

Abridged Bitcoin Context

Bitcoin's blockchain is simply a decentralized append-only ledger in which 1MB "blocks" of transactions are added every 10 minutes on average (Nakamoto 2009). The blocks are "chained" in the sense that each block has within its metadata a hash of the previous block. The ecosystem of participants is tricameral: users, miners, and nodes. Nodes maintains their own version of the blockchain, working to constantly update their version of transaction history as new blocks are propagated through the network. Users are those who broadcast transactions to the ledger to transfer value in the form of bitcoins (BTC). Any given transaction receives BTC by accessing unspent transactions that still have "unclaimed" BTC as inputs and then sending that BTC to new unspent transactions as outputs. To access these unspent transactions, a user "proves" ownership of the BTC in the transaction by digitally signing the transaction to prove that he or she is the intended recipient of the transaction. Miners take on the role of validating blocks of transactions via the mechanism described below.

Proof-of-Work Consensus Mechanism

A driving innovation of Nakamoto's Bitcoin protocol is its proof-of-work (PoW) consensus mechanism. Nakamoto did not invent the notion of PoW; however, its existence in academic literature is documented at least twice before the publishing of Nakamoto's whitepaper in 2009. First origins of PoW date to 1992, during which Dwork and Naor proposed the mechanism as a means of mitigating spam (Dwork & Naor 1992). In particular, forcing email senders to compute a brute-force solution to a computationally intensive "puzzle" limits the extent of emails a theoretical spammer could send. The second academic mention of PoW relates to Adam Back's Hashcash algorithm in 1997, which, incidentally, was similarly used to combat

Denial-of-Service (DoS) attacks also with a focus on spam prevention (Back 1997). In this system, the particular computational exercise is to find a SHA-1 hash of a header that includes the proposed email recipient's email address as well as the current date (Back 1997). The exercise is completed by appending random nonces in the hopes of finding the pre-image that corresponds to the following constraint: having at least 20 leading zeros in its 160 bit output.

Bitcoin's PoW takes Back's Hashcash a step further in two key ways. First, Bitcoin uses the SHA-256 algorithm via use of two successive SHA-2 hashes. Second, Bitcoin's PoW enables the difficulty of its hashing puzzle to be varied via different choice of constraints. In particular, valid hashes are mandated to have a value below an arbitrary but particular integer *t* (Bano 2017). Therefore, the difficulty is inversely proportional to *t*. That is, the lower value is chosen for *t*, the more challenging it is to produce a valid hash. This *t* value is adjusted every 2016 blocks to account for variance in the net hash power working to append a valid block so that a block is appended approximately every 10 minutes (Nakamoto 2009). For instance, in periods when hash power increases, *t* is lowered to increase mining difficulty and vice-versa. Empirically, *t* has, with a few exceptions, been mostly monotonically increasing given the near exponential growth in hash power from 2016 to present (Blockchain.info). In particular, aggregate Bitcoin hash power as of April 2018 hovers just below 30 million TH/s (Blockchain.info).

Nodes willing to generate hashes are the so called *miners* in the system. Said miners calculate potential hashes of proposed blocks to be added and should they find a valid hash, are rewarded under the following scheme:

> The first miner is to receive 50 bitcoins for appending a block (and its associated transactions) to the chain. Every 210,000 blocks, or approximately ~4 years taking into account that the aforementioned *t* is varied such that a block is produced every 10 minutes, the "block reward" is halved (Bedford Taylor & Taylor, 2013).

The first halving occurred on September 28, 2012 (Donnelly 2016). Following was the second on July 9, 2016 (Donnelly 2016). As of April 2018, the current block reward is 12.5 bitcoin. The next halving is projected to occur on May 31, 2020 (Halvening). The net effect of the block reward halving is that the number of bitcoin in circulation asymptotically converge to just under 21 million coins in circulation. Bitcoin is then provably scarce as the block reward will asymptotically approach zero, meaning there will exist no further way to increase the number of bitcoin in circulation. Further, said result implies that miners will at some point need to be incentivized by transaction fee once the block reward approaches 0.

In the case that there exist multiple blocks referencing the same parent block, typically referred to as a *fork*, the consensus mechanism dictates that the "longest chain" is selected because this chain has provably executed the most computational work (Nakamoto 2009).

Implications of Proof of Work

As is, one can see that the Bitcoin protocol has set a rigid parameter for block frequency and block size. Namely, block frequency is every 10 minutes and block size is capped at 1MB. This parameter sets a tight constraint on the number of transactions that can be processed per second. Because a block is on average confirmed only every ten minutes, the protocol is constrained to handling in expectation a small fraction (1/36,000) of a block per second. As a result, the question of scalability is in many ways compressed into some variation of how to fit as many transactions in a block as possible. Different flavors of solutions have emerged to answer this question, as described later, from increasing the block size of a transaction to lowering the average size of a transaction in bytes in a variety of ways.Other solutions are second-layer, building on top of the existing protocol. Still others are more deeply architectural, re-establishing mechanisms and parameters with scalability specifically in mind.

Classes of Bitcoin Scalability Solutions

For years, Bitcoin developers have brought forth a number of solutions to make Bitcoin's blockchain scale. Most have been described in a July 2017 update to the Bitcoin "Scaling Roadmap" that focuses on improving bitcoin along two key dimensions: capacity, as measured by theoretical transactions per second (tps), and scalability, the ease with which capacity tps can be reached (Sztorc 2017**)**. Beyond Bitcoin developers, various entrepreneurs have developed alternative protocols that emphasize scalability from the "ground-up." Academics, too, have put forth a variety of compelling protocols that in different ways minimize the number of transactions that happen "on-chain." Still, the Lightning Network is the solution that stands out, having the estimated potential to increase scalability by orders of magnitude more than any other solution while being fully compatible with Bitcoin's protocol after a series of enabling improvements were made to the protocol.

Solution Class #1: Compressing the Transaction

Bitcoin developers have focused their attention on a number of ways to compress the metadata in a given transaction. Below are three leading ways that developers have managed to shrink the bytes required for a transaction.

Serialization

First, notable bitcoin developer Gregory Maxwell made the observation that the serialization of Bitcoin transactions is an area of redundancy. While the unit impact of shrinking the byte stream representation of a transaction by a few bytes seems small, given the number of transactions that have the potential to take place per day, even this small change can drive scalability increases of between 20 and 30%. One example of a serialization improvement is in handling of the default value of "Use (N-th) prior value" of a number of fields (Maxwell 2016). In particular, one of the metadata fields in a given transaction allows users to save a few bytes by toggling that a particular field, say the value of the transaction, can simply be set as the immediately prior value. Maxwell proposes that this state persists on a global level, beyond just the transaction. In this way, one does not need to continually request each incremental transaction to use the prior value. Instead, once this flag is set, that state will persist until another transaction specifically designates a value. Therefore, bytes for using the N-th prior value don't need to be allocated on a per-transaction basis and can be used less frequently, saving space in th memory.

Schnorr Signature Aggregation

A second key scaling innovation is the notion of Schnorr Signature Aggregation. Schnorr signatures leverage the idea that many transactions can ultimately share a signature, leading to compression in the space required for a broad range of transactions (bitcoincore.org). This improvement is particularly beneficial because signatures comprise a non-trivial number of bytes of size to a bitcoin transaction. Consider that the current maximum size of a block in the Bitcoin blockchain is 1 MB. Charts from Blockchain.com indicate that the average number of transactions per block range generally range from 1500 to 2500 transactions per block in the past year (Blockchain.info). Therefore, the average transaction requires roughly 400 to 700 bytes per transaction. Further, Bitcoin's digital signature algorithm makes it such that transaction signature space requirements are lower bounded at 64 bytes per input to the transaction, not inclusive of several header bytes (BitcoinWiki). Therefore, digital signatures can comprise a large proportion of transaction size, especially in the case of multi-signature transactions or those with many inputs.

Take the example of multi-signature transactions. By supporting multi-signature transactions natively, Schnorr signatures can seamlessly aggregate all the requisite signatures in a multi-sig transactions into a single valid one combining the balances of all the inputs and consuming the amount of space as just one signature (bitcoincore.org). As a result, an arbitrarily large n-of-n multi-signature transaction can be signed by a signature that takes up the same space as a transaction with a single signer.

Segregated Witness

Segregated Witness is the third key transaction compression improvement. SegWit's key idea is to reorganize the way metadata is stored in transactions by separating the signature from key transaction characteristics such as the inputs, outputs, and value of the balance to sent (bitcoincore.org). Consequently, a given transaction ID is set in advance of it being signed, making the ID itself immutable in a way that it previously was not. By doing so, SegWit made a previously concerning class of attacks infeasible. These attacks focused on taking advantage of the malleability of Bitcoin transactions. Because prior to SegWit signature information was embedded in the transaction ID, bad agents could have chosen to broadcast a transaction identical to a legitimate one in all ways but a nominal change in signature and hope that their transaction was the one that broadcasted first and then take advantage of the resulting confusion in a variety of ways. This vulnerability will be discussed at length later in the paper.

SegWit is projected to improve capacity by up to 2.2 times pre-SegWit levels (Sztorc 2017**)**. From a scalability perspective, SegWit most crucially makes signature operations move from quadratically increasing with respect to the number of parties involved in a transaction to linearly (Sztorc 2017). A case in which this innovation is particularly relevant occurs when a Bitcoin transaction collects inputs from many or sends an input to many outputs. A real-world example might be a transaction that crowdfunds from twenty different individuals to collectively fund one project.

Solution Class #2: Splitting the Transaction Load with Separate Chains

Drivechains

Drivechains operate on the premise that Bitcoin's current blockchain should not be the only "chain" that can leverage the maximal 21,000,000 BTC that will be in circulation (Drivechain.info). Instead of a mono-chain approach, advocates of this approach focus on the potential for alternative "sidechains" to offload certain transaction activity off of the main Bitcoin blockchain such that capacity increases without any underlying changes to Bitcoin's protocol. Such an approach allows for new protocols that support a broader range of transaction types to emerge or prioritize differently on trade-offs between say auditability and security while still being compatible with Bitcoin's blockchain.

Solution Class #3: Reparametrization of Protocol Parameters

When Satoshi Nakamoto conceived Bitcoin in his seminal 2009 paper, he did so with several default values for key Bitcoin parameters in mind. Examples of such parameters include block size and block frequency. Nakamato set the former to 1 MB and the latter to 10 minutes for reasons left unstated. An alternative class of solutions focuses on optimization of these and other parameters to increase maximal number of transactions per second while keeping transaction latency in check. Indeed, even with efficient reparametrization of the block size and block frequency parameters, there is less than an order of magnitude improvement in transaction throughput. For instance, recent academic literature shows that changing the block size to an optimal 4MB, and the block frequency to no lower than 12 seconds results in throughput of 27 transactions per second (Croman et al., 2016).

Aside - Bitcoin Cash

Interestingly, on mainly the basis of this reparametrization observation, Bitcoin Cash "forked" from Bitcoin on August 1, 2017 (Wikipedia). After a block #478558, Bitcoin Cash began to maintain its own decentralized ledger, entirely separate from that of Bitcoin's (Wikipedia). The principal reason for the fork revolves around increasing Bitcoin's block size to 8MB such that transaction throughput could increase and consequently transaction fees for confirming a block could begin to decrease as it became easier to confirm a unit transaction given more transactions could now fit in a block. But while varying the parameters in Bitcoin's protocol is a source of improvement, reparametrization does not drive scalability enough to support global wide-spread use of Bitcoin.

Solution Class #4: Alternative Protocols Prioritizing Scalability

Another class of solutions has emerged in the response to scaling decentralized blockchains. These solutions focus on the development of new protocols with consensus mechanisms that, in the trade-off between scalability and security, prioritize scalability more than bitcoin does. In general, the below approaches have in common a lack of a computationally intensive mining process like the one that Bitcoin employs. Instead, these consensus mechanisms emphasize securing their underlying ledgers via primarily economic incentives by working to make the cost of deviating from expected behavior exceed its benefit. Three mechanisms will be summarized below: Proof-of-Stake (PoS), Delegated Proof-of-Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT).

Proof-of-Stake (PoS) Protocols

Instead of hash rate determining the likelihood of a particular miner finding a valid hash and correspondingly collecting the "block reward," PoS's analogous metric is the percentage of coins in circulation owned. Therefore, stakeholders with significant "stake" in the network via a significant ownership in tokens outstanding are more likely to earn the right to add the next block and in turn receive the associated reward. To be clear, simply owning coins does not give a "stakeholder" the chance to create the next block. The stakeholder must typically set these coins aside for staking to be held for that purpose for at least the short to medium term. In other words, one cannot trade coins that are being used to stake.

Proponents of PoS consensus mechanisms note that any stakeholder owning a non-trivial quantity of coins in circulation is never incentivized to act outside of the interest in the network. The only case in which such bad action might be incentivized is when the benefit to the individual exceeds the overall cost to the network. Second, the cost of attack increases with time if the supposed crypto-asset is appreciating in price as the cost of securing the majority of coins grows linearly with the price.

Delegated Proof of Stake (DPoS) Protocols

DPoS shares underlying principles with PoS but differs in a few key ways. Small stakeholders in the network have little incentive to stake their assets. Due to the probabilistic election of winning stakeholder in proportion to tokens owned, most small stakeholders will undergo many periods without earning a transaction fee but still have to face storage and bandwidth costs associated with staking their coins. In response to this lack of incentive, DPoS proposes that stakeholders use their holdings to proportionally vote for a witness, the party elected to validate a block. In DPoS, The first witness to receive a 50% vote of tokens outstanding verifies the proposed transaction and all are free to vote for more than one witness (Bano 2017).

Said witnesses are unlikely to deviate from expected behavior because the first instance of bad behavior will lead to a loss of votes beyond the 50% threshold and therefore a loss of value associated with future transaction fees. Second, the difficulty associated with becoming a witness increases in lockstep with the value of the network. If one assumes that a specific crypto-asset's value is proportional to the network value, then the more value the network accrues, the more profitable it becomes to become a witness as the "rewards" become valuable if if the reward is simply more of the token. Therefore, more and more tokenholders will accumulate tokens in order to attempt become a potential witness going forward to earn a share of these rewards.

Practical Byzantine Fault Tolerance (PBFT) Protocols

The PBFT consensus algorithm delegates a fraction of available nodes as "generals" to manage the consensus process. A state machine replication technique, the PBFT typically algorithm adds two additional constraints: that all generals' operations are deterministic and that all generals begin in the same state (Castro & Liskov, n.d.). Now, consider a hypothetical message (corresponding to a block) to be added to the ledger. Said message is broadcasted to all generals. All generals then run a computation using the inputs of the message as well as their current state. As soon as the strict majority of generals agree on the result of the computation, consensus on the validity of the message is reached.

PBFT centralizes consensus more than aforementioned alternatives as the sole determinants of consensus are the generals. Since election of generals is often carried out by some central agent, said agent, often the core of developers maintaining the protocol, has a higher influence on the state of the network than in alternative mechanisms (Castro & Liskov, n.d.). Therefore, relative to the other methods, PBFT is also less fault-tolerant as an attack vector targeting a general would pose a large risk to network reliability and availability. As a result, networks designed with BFT consensus algorithms are particularly prone to Sybil attacks as a result of the value associated with becoming a general -- typically, generals, are the nodes that accrue transaction fees and other block rewards if relevant.

Solution Class #5: The Killer Solution - Lightning Network

While compelling, the aforementioned solutions even if all successfully implemented in aggregate leave Bitcoin magnitudes behind the throughput that Satoshi once dreamed it could support. In the race to make Bitcoin scalable enough for wide-spread global adoption, one solution has very much differentiated itself, the Lightning Network. On the metrics of capacity and scalability, the Lightning Network has the potential to improve "both capacity and scalability [of Bitcoin] by a factor of ~1000" (Sztorc 2017). That is, taking 2018 year-to-date transaction levels per day of roughly 200,000 to 300,000, the Lightning Network could enable 200 to 300 million transactions a day, competing very favorably with any payment giant world-wide (Blockchain.info).

Overview

Recognizing that broadcasting all transactions to all nodes places an undue number of transactions on the ledger, Poon and Dryja propose a new solution: the Lightning Network (LN) (Poon and Dryja 2016). The LN leverages payment channels and time-locks to move most

transactions off-chain, appending to the ledger only net settlement or conflict transactions where parties disagree on the current state. By result, only in adversarial conditions does throughput drop to on-chain levels. The significance of the LN is in its ability to theoretically scale Bitcoin transaction throughput to any level with a latency of near instant payments while also maintaining the provable security that on-chain transactions can provide. Using two-way payment channels secured by time-locked contracts, the LN enables instant-time peer-to-peer payments, even when two users lack a direct channel between them. Further, the LN does so without a need for a trusted intermediary. Interestingly, the LN also enables cross-chain payments, so that any given payment could be routed through chains beyond just bitcoin's, assuming these other chains also use bitcoin's SHA-256 hashing algorithm and offer the functionality to lock payments for a period of time (Poon and Dryja 2016). The reason the other chains would need to use the SHA-256 hashing algorithm is because the pre-image and hash used in generating hash-time locked contracts need to be computed in an identical manner to validate that the HTLC is being executed according to protocol.

Transaction Kinds

In the LN, there are four different kinds of transactions: Funding, Commitment, Revocable Delivery, and Breach Remedy Transactions. Funding transactions are those that "fund" the balance of the channel between two parties. Commitment transactions capture a snapshot of the state of the channel and correspondingly, each party's balance. Revocable delivery transactions are the mechanism through which channels can be maintained trustlessly. These transactions offer each party security and return of the funds he or she committed to the channel in the case of the counterparty deviating from protocol. Finally, breach remedy transactions are the means through which old commitment transactions are invalidated so that channel state can continue to be updated without either party needing to worry about an "old state" of balances being broadcasted. Each will be discussed in more detail in the section to follow via a working example.

Introduction of Working Example: Direct Channel Payments

Funding Transaction:

Consider a hypothetical Alice and Bob who would like to route payments directly between each other -- that is, without using a series of intermediary people through which to route the payment on the way to its end destination. Funding transactions open the channel: Alice and Bob create a 2-of-2 multi-signature transaction that both parties sign, committing some quantity of bitcoins to be exchanged. This funding transaction is then posted to the ledger and begins the formation of a "channel" between Alice and Bob. These channels are designed to be

opened indefinitely, with two key exceptions. First is in the case that there is collective agreement from each party in the channel to close the transaction. Second is in the instance that parties disagree on the current state of balances within the channel. Both scenarios will be discussed. For reference, what differentiates a multi-signature transaction from a standard transaction on the Bitcoin network is the need for multiple parties to sign the proposed transaction.

Commitment Transaction:

Next, a commitment transaction is created, but not immediately broadcasted, in tandem with a revocable delivery transaction. Pairing the two enables the LN to automatically enforce trustless penalties associated with deviating from expected behavior (Poon and Dryja 2016). Stated in other words, the commitment transaction serves the purpose of ascribing blame. The commitment transaction can do so because for any proposed state update, two parallel half-signed commitment transactions are created. For instance, in the example discussed earlier, Alice would have her version of a commitment transaction, say CT_{Alice} that is signed by Bob and only she can broadcast. Bob too would have an analogous commitment transaction, CT_{Bob}. Of course, only one of these two transactions can be broadcasted, since they both spend from the same output. Therefore, on the basis of the party that broadcasted an incorrect commitment transaction, blame can be saliently ascribed. Now that the party to be blamed has been identified, the revocable delivery transaction automatically punishes the errant party, as will be shown below.

Revocable Delivery Transaction:

The revocable delivery transaction is introduced first as one of the outputs in the commitment transaction, the other being the counterparty in the channel. In specific, in the commitment transaction, the balance of the funding transaction is the input that is sent to two outputs: a revocable delivery transaction address and the intended recipient of the payment. For instance, say Alice and Bob began with 5 BTC each that was then sent to to the funding transaction address. Assume Alice and Bob desire the current state to reflect Alice having a balance of 4 BTC and Bob the remaining 6 BTC. Then, the resultant commitment transaction would pledge 4 BTC to the revocable delivery address and 6 BTC to Bob. Similarly, a parallel transaction would be created for Bob that sends 4 BTC to Alice and 6 BTC to the revocable delivery address.

Now, the revocable delivery transaction is to be created. One differentiating characteristic of the revocable delivery transaction is the incorporation of a lock-time, be it absolute or relative. That is, even with all requisite signatures, revocable delivery transactions must wait until some

designated number of blocks before being broadcast (relative) or until an absolute block height on the bitcoin ledger (absolute). Further, these transactions are also 2-of-2 multi-sig transactions. Continuing with the previous example, after the proposed commitment transactions have been created, Alice creates a revocable delivery transaction that returns its entire balance (4 BTC) to Alice after some time-lock expires. Bob does the same with the balance of BTC (6 BTC) earmarked for the revocable delivery transaction.

Scenario 1: Mutual Agreement to Close Channel

At this point, two different scenarios can occur. First is that Alice or Bob broadcasts the correct commitment transaction, correctly updating the state and consequently "closing" the channel as a result. One caveat worth mentioning is that the party that broadcasts the transactions suffers from a lock-up of his or her funds for the duration of the time-step. For instance, if Bob were to broadcast his version of the commitment transaction mentioned above, Alice would immediately receive 6 BTC. Bob, on the other hand, would have to wait for however many blocks the time lock mandates to broadcast the revocable delivery transaction and collect his 4 BTC. While not ideal, in some ways this lock-up is the price Bob pays for being able to protect himself from errant action by Alice without relying on a centralized intermediary.

Of course, the protection applies symmetrically to Alice. In this case, one can say the channel has been closed because the funding transaction, the basis of the channel, has been fully depleted of funds. That is, before Alice and Bob could create another pair of commitment transactions, they would first have to issue a new funding transaction, effectively "opening" a new channel. It is worth mentioning that one benefit of requiring a 2-of-2 multisignature commitment transaction is that Alice can only act errantly in one and only way: broadcasting an old commitment transaction. Alice cannot broadcast a transaction that is not parallel to Bob's (such as one where she receives more BTC), because Bob would refuse to sign said transaction.

Scenario 2: Channel Continuation

The second scenario is that Alice and Bob continue their channel, a scenario in which breach remedy transactions are introduced. There still exist two half-signed commitment transactions associated with the transaction mentioned prior, but neither of them are valid because they remain half-signed. In this scenario, Alice and Bob both may desire to update their balances, say to reflect both of them now having 5 BTC each. Here, Alice and Bob again issue a pair of half-signed commitment transactions as mentioned before, with one output being a revocable delivery transaction with a timelock and the other being the counterparty's address. In this example, we can see that in this new state update, Bob is strictly worse off as his balance

moves from 6 BTC to 5 BTC. As such, he is incentivized to broadcast the old commitment transaction that has already been half-signed.

Breach Remedy Transactions:

To deter this action, breach remedy transactions are introduced as a mechanism to punish the broadcasting of old commitment transactions. In particular, Bob and Alice both create a pair of 2-of-2 multisig half-signed breach remedy transactions in response to Alice and Bob creating a second set of commitment transactions. At a high level, this transaction will provide "insurance" against Bob and Alice trying to maliciously broadcast the old commitment transaction now that they would both like to reflect the new updated state implied by the second commitment transaction. The breach remedy transaction is a spend from the commitment transaction that supersedes the revocable delivery transaction (Poon and Dryja 2016). As such, Bob is maximally punished for broadcasting an old transaction since the entirety of funds in the channel are sent to Alice. Any BTC that would have been spent by a revocable delivery transaction is instead superseded by the breach remedy transaction that Alice can immediately spend from. Consequently, on the broadcast an old commitment transaction by Bob, Alice is immediately privy to the entire channel balance. However, if Alice does not spend from the breach remedy transaction within the locktime, then it is possible for Bob to broadcast his revocable delivery transaction and claim its balance (Poon and Dryja 2016). Therefore, Alice is mandated to act accordingly with the locktime and must be actively monitoring the channel to check that at any instant an old transaction has not been broadcast.

If both parties act in line with protocol, then the breach remedy transactions are never broadcasted. Then, the process can repeat itself with a second set of breach remedy transactions and a third set of commitment transactions if Alice and Bob would like to again update balances. And so the process can continue indefinitely, until Alice and Bob disagree, in which transactions move on-chain and the channel closes or until Alice and Bob agree to broadcast the current commitment transaction, settle the balance, and close out the channel.

<u>Non-Direct Payments: Multi-Hop Paths</u>

Thus far, we have only considered the scenario in which Alice and Bob have a direct channel. Alice, though, could very well desire to route a payment to someone whom she does not share a direct channel. Fortunately, the Lightning Network has the capability to handle transactions of this form as well.

Hashed Timelock Contracts (HTLC)

The LN is able to offer trustless multi-hop payments on the basis of using hashed timelock contracts (Poon and Dryja 2016). The core idea of the HTLC is to eliminate the need for custodial trust in multi-hop payments by allowing payments to cascade down a path only after the prior entity in the path has sent the payment to the following entity. That is, in a hypothetical 2-hop transaction from Alice to Bob to Charlie, Bob is not asked to send funds to Charlie until he has received the corresponding funds from Alice. Poon and Dryja describe a hypothetical HTLC between Alice and Bob below:

> 0. Alice chooses some arbitrary value (the pre-image) and hashes it, producing a hash value **H.**
> 1. If Bob can produce to Alice an unknown 20-byte random input data **R** (the pre-image) from hash **H**, within **D** days, then Alice will settle the contract by paying Bob **N** BTC.
> 2. If **D** days have elapsed, then the above clause is null and void and the clearing process is invalidated, both parties must not attempt to settle and claim payment after three days.
> 3. Either party may (and should) pay out according to the terms of this contract in any method of the participants choosing and close out this contract early so long as both participants in this contract agree.
> 4. Violation of the above terms will incur a maximum penalty of the funds locked up in this contract, to be paid to the non-violating counterparty as a fidelity bond.

Looking at the above definition, the validity of the contract is entirely predicated on knowing pre-image R given a hash H. It follows that Bob producing R is "proof" that payment has occurred. However, if Bob is not able to produce R within D days, there is effectively no proof of payment and Alice can retrieve her funds from the HTLC. Therefore, there are two execution paths for an HTLC: "delivery" or "timeout." The former scenario, providing R within D days, is an example of the delivery execution path and the latter of Alice retrieving funds is an example of timeout.

Implementation of the Lightning Network

Now that the core functionality has been described, discussion moves to implementation at scale. It is quickly apparent that establishing a channel between every possible pairing of potential nodes can be infeasible as the network scales. Such a naive solution would involve $O(n^2)$ channels for n network participants. Therefore, one key implementation challenge around the Lightning Network is the choice of routing or "path" algorithm from one arbitrary node to another.

Overview of Potential Routing Algorithms

A number of potential routing solutions have been proposed in response to the aforementioned scalability concerns around the Lightning Network. At a high level, two topologies have been proposed.

Hub-and-Spoke Topology

First, is a traditional hub-and-spoke model, analogous to the approach used by many leading airlines such as Delta or United Airlines [Towns]. This approach, as traditionally proposed, faces a number of challenges, From an incentive standpoint, creating dual classes of nodes, one for "hubs" and one for non-hub is particularly problematic. Non-hub nodes have little incentive to constantly run a node given that any transaction fees generated from a particular payment will accrue to the hub node. In particular, given a transaction originating from node A through hub H and terminating at node B, if node B's wallet were to be offline, the hub would be forced to hold the funds in custody until node B returned online, increasing time associated with the payment. Problematically, such an approach makes it implausible to upper bound any transaction as no guarantee can be made on how long a delinquent node such as node B may remain offline.

Hypothetically, one could modify the previous approach as follows: so long as a non-hub node is online and there is sufficient transaction fee volume to support it, each non-hub node could receive a pro-rata distribution of transaction fees every period (say each 24 hours) high enough to cover the costs of running said node as well as a few percentage point premium offered to incentivize the running of a node. Of course, this approach is not pareto-optimal -- leaving hub nodes strictly worse off. Also problematic: this approach is at odds with the ethos of decentralization. With only a few hubs, power in the network, as measured by volume of transactions processed at the node, is heavily concentrated. In a similar vein, the hub-and-spoke method has little fault-tolerance. Should one of the major hubs go offline for any reason, the network would be heavily compromised.

Organic Topologies

Global Beacons

Alternatively, a system of global beacons was proposed (Russell). Of the set of nodes, a pseudorandom subset are to be selected on a periodic basis. All other nodes attempt to find paths to these beacons and said paths are the ones used to route payments in the network. A salient concern with this approach is the risk of a bad actor engaging in a Sybil attack (Prihodko et al.

2016). Recognizing that global beacon nodes generate most of the transaction fees in a given period, a bad actor could subvert the network. By creating a number of fake nodes, these actors can increase their odds of becoming a global beacon node. The Lightning Network is particularly at risk due to the ease of generating new nodes. In fact, generating a new node simply requires syncing the latest Bitcoin blockchain and then spinning up a Lightning node. Additionally, computational requirements of serving as a global beacon could very well exceed the capacity of the typical modern computer, constraining the candidate pool for global beacons and inadvertently promoting centralization. Finally, the global beacon approach results in the creation of more channels than an unconstrained approach.

By forcing all transactions to route through the global beacon in each period, there become a number of unused channels. In particular, consider the following example. In period 0, a set S of nodes are selected as global beacons and are payments are routed through nodes in S. By period 1, a new set S' that shares no element with S, is selected as the class of global beacons. All channels from period 0 are now without use and must be closed. However, this approach broadcasts transactions to the ledger with a higher frequency than alternative approaches. Consider that each channel opening is marked by a funding transaction that *must* be published to the ledger. In the above example, all channels opened in period 0 but not used in period 1. In fact, given that the number of global beacon nodes will likely be an order of magnitude lower than the number of total nodes (to minimize repetition of global beacon nodes), the proposed scenario wherein no beacon nodes are shared in two successive periods is quite likely.

Local Beacons

Another proposed routing solution involves local beacons (Bairn). In this concept, each node has a list of beacons through which it must route all of its payments. Smoothing distribution of traffic across the network, this approach both gives every node a higher probability of being a beacon for another node and as a result provides potentially stronger incentive for a node to remain online in order to continue to reap transaction fees.

Other Routing Solutions

Inspiration for other solutions to routing transactions on the Lightning Network come from analogy (Prihodko et al. 2016). One analogy is to mobile ad-hoc networks (MANET). Indeed, the Lightning Network shares a number of commonalities with a MANET. As in a MANET, capacity in LN is varying: nodes new and old can appear and disappear on a whim since LN nodes have no bound on opening or closing a channel [Prihodko]. Two caveats are that

the incremental "hop" in a given transaction path increases the transaction fee paid by the sender and that transactions must originate from a specific source.

Proactive Protocols

A look at MANET algorithms reveals two broad classes of routing protocols: proactive (table-driven) and reactive (on-demand). Proactive protocols rely on routing tables. More specifically, these protocols distribute all knowledge necessary to build a routing table even before any traffic is transmitted. A representative proactive algorithm is the Optimized Link State Routing Protocol (OLSR) (Clausen, Jacquet, & Viennot, n.d.). Appendix A walks through the OLSR algorithm.

Reactive Protocols

On the other hand, reactive protocols find a path if and only if that path is requested. Two relevant examples of reactive routing algorithms are Ad hoc On Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) (Prihodko 2016).

Implications of MANET Routing Algorithms

There is an inherent trade-off at play in proactive and reactive protocols. Proactive protocols face significant overhead requirements due to the routing table needing to be stored in advance of any path finding but trade memory inefficiency for reliability in the form of low variance in time to find a path. On the other hand, reactive protocols trade reliability for overhead as these protocols defer all path-finding to the request of a specific path. Therefore reactive protocols appear to scale better (due to lower overhead requirements). In the context of the LN, however, reliability is a key attribute as many applications of LN revolve around payments and in the context of payments, reliability is a core attribute for both parties in a given transaction. Therefore, a hybrid routing solution is far more appropriate (Prihodko et al. 2016).

"Real World" Routing Approach

At least three organizations have worked on implementations of the Lightning Network: Blockstream (lightning-d), Lightning Labs (lnd - Lightning Network Daemon), and ACINQ (Eclair) (Wikipedia). Only the Lightning Labs implementation explains in its documentation its process for node and channel discovery. Below is an analysis of the Lightning Labs routing approach.

Lightning Labs - Lightning Network Daemon (lnd)

Lightning Labs specifies that its lnd closely follows a list of 11 "Basic of Lightning Technologies" (Github.com). BOLT 7 reveals the approach to routing and peer-to-peer channel discovery. To be clear, routing and peer-to-peer channel discovery is only necessary for multi-hop payments. If Alice and Bob have a direct channel, there is no need to evaluate possible paths. As alluded to above, the lnd implementation takes inspiration from both the proactive and reactive routing algorithms discussed above. For instance, the lnd routing algorithm is proactive in the sense that it maintains a local view of the channel topology in order to facilitate route finding, very similar to the routing tables used in proactive MANET routing algorithms. In other ways, the routing algorithm is reactive. For instance, given an availability of several paths from a hypothetical Alice to Bob, the lnd routing algorithm differentiates the paths on the basis of two key fields: *cltv_expiry_delta* and *fee_base_msat*. The former field refers to the worst-case bound on how long funds would be "locked up" in the case of a timeout HTLC execution. The latter field refers of course to the base fee rate per milli-satoshi ($1/10^{-11}$ units of BTC). Here, the routing algorithm chooses in real-time between several available paths attempting first to minimize the sum of *cltv_expiry_delta* values across the path and then the fees associated.

Lightning Network Enabling Technologies:

A series of technical improvements needed to be made to Bitcoin's protocol for the Lightning Network to be possible. Improvements to Bitcoin's protocol are made formally through Bitcoin Improvement Proposals (BIPs) and Github pull requests by Bitcoin core developers. Anyone can draft a BIP, after which it is iterated on and then made active and, if relevant, voted on. Below, the BIPs that solved the largest issue in implementing the Lightning Network will be discussed.

Key Issue: Transaction Malleability

Transaction malleability has been a critical flaw in Bitcoin's protocol that has taken at least 3 BIPs to resolve, as will be discussed. In short, transaction malleability attacks occur when a malicious agent broadcasts a transaction that is identical to a legitimate transaction in all ways but the transaction ID. The bad agent modifies the transaction ID in a very nominal way (usually through an equivalent change in his or her signature) in the hopes that his or her transaction is confirmed in advance of the legitimate transaction (Bitcoincore.org). If this is the case, the broadcaster of the legitimate transaction may find that his or her transaction never posted to the ledger because it was invalidated by the aforementioned transaction. As a result, he or she may

make the mistake of re-sending the transaction and as a result double-spend. Alternatively, the bad agent could continue to broadcast transactions using his or her transaction hash as the previous hash and invalidate transactions the other party has broadcasted. Below, two examples of malleability vulnerabilities and the corresponding BIPs developed as a result are discussed.

BIP66

The first malleability vulnerability brings to light the Bitcoin protocol's willingness to consider padding characters as differentiating enough to merit a different transaction ID (Klitzke 2017). In particular, the metadata of a Bitcoin transaction are bundled into a Distinguished Encoding Rules (DER) Abstract Syntax Notion 1 (ASN.1). ASN.1 is a notation for describing abstract types and values and DER adds the constraint that each ASN.1 value has a unique encoding (Kaliski Jr. 1993). Further, historically Bitcoin used OpenSSL to validate the ASN.1 data of transactions proposed to be added to its ledger (Klitzke 2017). Problematic was OpenSSL's willingness to ignore padding type characters in the transaction hash. In other words, useless characters such as whitespace could be added to modify a transaction hash. Bad agents used this vulnerability to carry about an attack in the template described above. As a result, BIP66 was developed in order to specify more rigorously how ASN.1 data is to be encoded so that padded transactions, for example, would be swiftly rejected (Wuille 2015).

Github Pull Request #6769

The second vulnerability exposes the protocol's mishap of counting equivalent signatures as unique signatures (Klitzke 2017). More specifically, this flaw originates from the scheme Bitcoin uses to sign transactions: ECDSA. In particular, an ECDSA signature has two parts: *(r, s)*. Attackers found that if (r, s) was a valid signature, then so too was (r, $-s \bmod n$). As a result, these individuals could generate an equivalent signature knowing r and s without knowing the underlying private keys that went into the making of r and s. This vulnerability was promptly fixed by a Github pull request, specifically request #6769, in which a unique signature was forced by election of the lower value signature (Maxwell 2015).

BIP141

Finally, the BIP of much interest and controversy, BIP141: Segregated Witness, approaches transaction malleability from a different perspective. SegWit introduces the notion of separating core transaction data such as input, output, and value from signature metadata, a proposal that brings with it a number of benefits (bitcoincore.org). More granularly, SegWit introduces a new field into transaction metadata: wtxid, in which a transaction id is generated without using any ECDSA signature information (Lombrozo et al. 2015). Not only does the BIP

lower the number of bytes required per transaction (as ECDSA signatures occupy a significant share of bytes allocated for a transaction), but it also further prevents any transaction malleability attacks known to date. Given the wtxid is generated solely from a combination of inputs, outputs, and other immutable fields, wtxids cannot be retroactively modified (bitcoincore.org)

Now, with BIP141 (and related prior BIPs), the Lightning Network is able to be deployed because transactions meant to offer security such as the breach remedy transaction or the revocable delivery transaction can no longer be retroactively modified and so channel participants can rest assured that their provisioned funds are safe.

Challenges Associated with the Lightning Network

With the introduction of a second-layer protocol inevitably comes new risks and concerns. In particular, we will focus on risks of two key parties in the Lightning Network: miners and users. Most attacks on the Lightning Network arise as a result of collusion or Denial of Service (DoS) attacks.

Miner-induced risks

In the case that a channel counterparty broadcasts an old commitment transaction, the other party in the channel has the right to broadcast a breach remedy transaction that enables him or her to claim the full balance of funds in the channel. However, an adversarial miner might exclude these breach remedy transactions from their mempool of unconfirmed transactions. If the breach remedy transaction goes long enough without being broadcast, particularly beyond its locktime, then the malicious counterparty can successfully reap the benefits of broadcasting an old commitment transaction.

User-induced risks

<u>Forced Expiration Spam</u>

Poon and Dryja note the risk of forced expiration spam in their proposal. The attack is a Denial of Service (DoS) attack that seeks to overwhelm the blockchain with useless transactions so that valid transactions are delayed beyond their respective timelocks. As a result, malicious counterparties can broadcast old states and earn BTC to which they are not entitled. Such an attack might be carried out by an adversarial user creating many spam channels with trivial balances that all simultaneously expire, potentially overwhelming the capacity of the next few blocks.

The "Fraud Template"

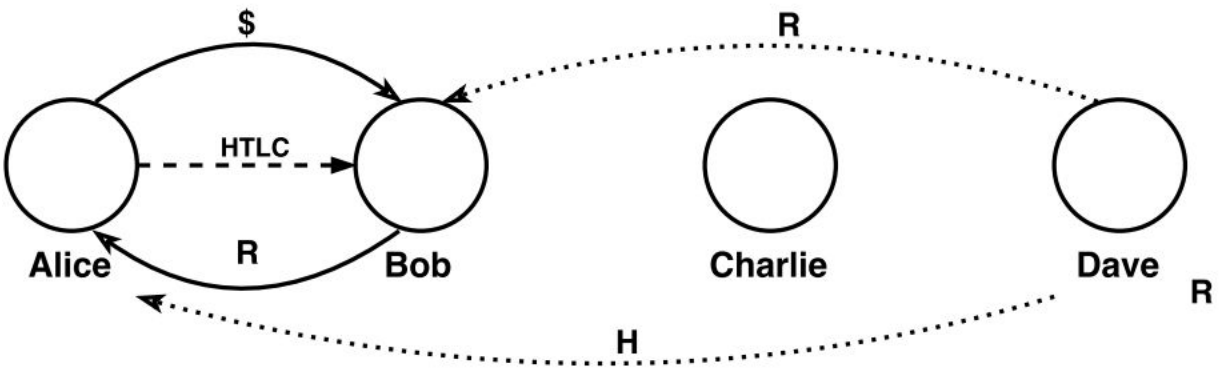

Figure 1: Prototypical Example of Fraud Scenario (Piatkivskyi, Axelsson, & Nowostawski, 2017)

One of the key vulnerabilities of the Lightning Network is that it equates knowledge of the pre-image R to proof of payment. As a result, the Lightning Network assumes it is never in a participant's interest to share R outside of when the protocol might dictate to do so. However, there do exist cases where this paradigm does not hold. Researchers demonstrate a vulnerability in the following example: Alice routes a payment to Dave through Bob and Charlie via cascading HTLCs introduced in the Lightning Network (Piatkivskyi, Axelsson, & Nowostawski, 2017). That is, Alice contracts with Bob, Bob contracts with Charlie, and Charlie contracts with Dave with HTLCs that have decreasing n-Locktimes.

As per the protocol, Dave generates an arbitrary pre-image R and finds its hash. He then distributes the hash to all other parties to use as the basis of the HTLC. Dave then deviates from the protocol by sharing R with Bob. Now, Alice has reason to believe that Bob has already paid Charlie. Because the protocol dictates that Bob could have only received the pre-image from Charlie if he had paid Charlie, Alice has no reason to suspect foul play. As such, she executes her HTLC and sends him the corresponding payment. Now, Dave claims rightfully that he never received payment. Further, Alice has no way of knowing ex-post that Bob and Dave had colluded. Bob then disappears with the funds, never in fact sending them to Charlie.

Key Example: Money Laundering

Leveraging the above vulnerability, a group of bad agents can launder money with plausible deniability (Piatkivskyi, Axelsson, & Nowostawski, 2017). Continuing with the above example, assume that Alice, Bob and Dave are now colluding. In particular, the intent is to launder money to Bob in an untraceable manner. Here, Alice claims that her funds were never sent to the intended recipient and so requests her funds to be returned. Dave, being complicit in the laundering, returns the funds to Alice. Bob then disappears with the funds. Now, Dave was able to "pay" Bob with plausible deniability since he "graciously" returned funds that were sent but never received by him.

Deanonymization of Intermediate Node's Position in a Routing Path

Given that a hypothetical Alice would like to route a payment to Bob through a set of intermediaries, it can become possible for the intermediary to identify its relative position along the path to final payment, at least in the way that the Lightning Network is implemented in lnd by Lightning Labs as a result of two factors (Github.com). First is the availability of the *cltv_expiry_delta* field for the HTLC; second is the public availability of the topology of the network. Given the two, an intermediate node can approximate how far along the path toward the final recipient that he or she is situated by comparing the difference between the current block height and expiry block height of the HTLC relative to his or her *cltv_expiry_delta* value. One solution to this issue is simply to add a "shadow route extension" or simply an arbitrary additional quantity of blocks before the HTLC expires so that "guessing" one's position along a payment route becomes more difficult.

Solution Class #6: Serious Bitcoin Protocol Modification

Teechan

Observing that the Lightning Network would require several material changes to the underlying Bitcoin protocol, in 2016, researchers put forth a proposal called Teechan involving the use of payment channels in trusted execution environments (TEEs) (Lind et al. 2016). Notably, Teechan was found to achieve "2,480 transactions per second" with "sub-millisecond latencies" on the Bitcoin test network (Lind et al. 2016). Teechan achieves these throughput levels by moving most transactions off-chain using two-way payment channels where transfers of value are orchestrated by TEEs on behalf of channel participants.

Three stages of the protocol exist: channel establishment, channel operation, and channel settlement (Lind et al. 2016). In the channel establishment stage, a set-up and refund transaction are established, much like the funding and revocable delivery transactions in the Lightning Network. Unlike the Lightning Network, Teechan's protocol asks that channel parties share their

private keys with their respective TEE's in this step, making the solution not truly "trustless." After a secure communication channel is established between the TEEs, one party, without loss of generality, broadcasts the setup transaction and so begins the channel. Both parties have the ability to broadcast the refund transaction and have their original balances refunded to them at any point after the some locktime-designated number of blocks have passed.

Next, in the channel operation stage, either party can simply request that the TEE send some quantity of BTC. After the TEE verifies the balance, it updates its internal balance state and relays the size of the payment to the other party's TEE so that it too updates its state. Finally, channel settlement can occur in two ways. First, if either party sends a "terminate request" to its TEE then it sends a settlement transaction reflecting the net balances in the channel to the party, destroys all its memory, and halts (Lind et al. 2016). Alternatively, once the locktime has passed, channel participants can choose to broadcast the refund transaction or use the aforementioned settlement transaction. In either case, the channel has been closed.

One appealing aspect of Teechan's protocol is that the entire process of transacting with a counterparty via a channel requires only two transactions: one to establish the channel and one to terminate the channel and offer settlement of the balance. Also, while the Lightning Network requires broadcast of four transactions in case of disagreement, Teechan only broadcasts two (Lind et al. 2016).

Moreover, during the lifetime of a channel, it can be used an infinite number of times, just as the Lightning Network can. Further, payments are instant-time (sub-millisecond) once a channel is opened. Additionally, there is no risk of the channel counterparty misappropriating funds that do not belong to him or her; a payment can only be claimed only if a counterparty issues one. Further, Teechan offers "peace of mind" in the sense that channel nodes have no need to monitor the blockchain for the entire duration of the channel for malicious action, as a channel participant would have to in the Lightning Network if a counterparty broadcasted an old commitment transaction. This guarantee is provided by the TEE, as secure enclave environments that TEEs operate can detect roll-backs or the equivalent of broadcasting old state information (Lind et al. 2016).

Analysis of Teechan

Of course, the largest risk factor around Teechan revolves around the TEE. Any party wishing to open a channel is obligated to share his or her private key with the TEE. Still, by using Intel SGX as the TEE of choice in their implementation, the authors are able to offer a secure enclave wherein attackers even with physical access to the machine including memory access are still prevented from accessing the enclave and so the private keys are fairly protected

(Lind et al. 2016). Another risk factor is around the TEE failing. If this is the case, then a viable solution is simply to persist the state of one TEE onto secondary storage and let the party that faced the failed TEE continue state updates from this secondary storage source (Lind et al. 2016). This solution is feasible because the states of any pair of TEEs will be consistent with one another. Because payments can only be claimed and not requested, if one TEE is inconsistent with another TEE, it can only be because it failed to claim a payment, leaving it worse off. Therefore, TEEs will always be in the same state. A final challenge around Teechan is its inability to offer multi-hop payments. Note that Teechan can only offer payments between parties in a channel, whereas the Lightning Network can route payments between any two parties in which a path can be drawn using existing channels, if sufficient channel balances exist along the path.

Overall, Teechan is a promising solution to the problem of scalability in Bitcoin. On the other hand, it performs worse on the metrics of throughput than the Lightning Network. Fortunately, the Lightning Network is compatible with Teechan (Lind et al. 2016). No proposals have been put forth to date integrating the two, but, taken together, the two solutions could offer a very compelling solution to blockchain scalability.

Thunderella

Another proposal, Thunderella, emphasizes a different off-chain approach to scalability. The Thunderella paradigm aims to have instant transaction confirmation time in the majority of transactions, but via elected committees as opposed to via payment channels. In particular, a designated leader or "accelerator" serializes ands batches transactions and sends them to a committee (Pass & Shi 2017). Of course, selection of the committee is a key step in the process as these committee members in aggregate have the ability to instantly confirm any transaction. The most promising solution to committee selection is to use the miners of the most recent blocks (Pass & Shi 2017). Then, the committee members approve or deny the validity of a given transaction, signing the transaction to indicate approval. If the majority of the members agree on a transaction, it is instantly confirmed, subject to any delays in propagating the transaction information to the network of nodes. On the other hand, Thunderella proposes moving on-chain for transactions where the majority fails to "notarize" a transaction (Pass & Shi 2017). In parallel, the committee waits until the problematic transactions have been broadcasted to continue notarizing future transactions so that the ordering of transactions is still in tact.

Analysis of Thunderella

The principal challenge around the Thunderella approach is more economic than theoretical. In particular, introducing a "leader' and a "committee" into the protocol system

magnifies governance and incentive-design challenges by a non-trivial extent. Bitcoin's protocol is as much a cryptographic breakthrough as it is is in market design, governance and economic incentives. Turning a previously tricameral system of miners, users, and nodes into a five-pronged system is likely to introduce a whole host of unforeseen second-order interaction effects. Moreover, uncertainty abounds over how the size the rewards associated with being a leader or a committee member, especially as the roles relate to vulnerability to a potential Sybil attack. Further work also has to be done to determine the length that a particular entity serves in these roles. Therefore, while Thunderella introduces a solution that is provably scalable, it comes with a host of incentive and ecosystem challenges. The elegance of the Lightning Network is in its ability to "plug-in" to the existing Bitcoin ecosystem nearly trustlessly, therefore mitigating (though not eliminating) the need to consider incentives in off-chain payments.

Concluding Thoughts

Bitcoin's future is unbundled with uncertainty. Whether the crypto-asset is ever able to find the demand to reach Satoshi's vision of Visa level transaction output will be the result of the confluence of a number of factors both technical and market-driven. Still, the lessons learned from solving Bitcoin scalability can be extended to the protocols that may end up setting the dominant design in the blockchain protocol space -- even if Bitcoin does not end up being the proverbial answer. The collective extent and rate of innovation in blockchain scalability is simply remarkable. From first-layer solutions ranging from Schnorr Signature Aggregation to SegWit to second-layer proposals such as the Lightning Network or Thunderella, the continued range and quality of proposals suggest that, it is not a matter of if, but when the right solution is found. Alternatively, perhaps the answer lies in a recombinant approach marrying currently compatible approaches such as the Lightning Network with Teechan.

Appendix A: **OLSR Algorithm (Maccari)**

Time to live (TTL): an 8-bit field that stipulates how long data is to last in a network, traditionally used to improve caching (Meteoswiss)
N: the number of nodes in the network
n_i: the set of neighbors of an arbitrary node i
n_i^2: the set of neighbors two edges away from node i
m_i: the set of multipoint relay nodes chosen by node i
m_i: the minimal subset of n_i wherein all nodes in n_i^2 share an edge with a node in m_i

First, each node sends out a "HELLO" message wherein TTL is set to value 1 regularly. Within this message, each node broadcasts each of its neighbors with two additional bits: one for noting if the neighbor is symmetric or not (as in a two way directed edge versus one way directed edge) and another for designating if the minimum cardinality MPR includes the particular neighbor. One can find the minimum MPR using a known minimum spanning tree algorithm.

Second, for each node i, its corresponding m_i broadcasts a Topology Control (TC) message with a longer TTL value to relay information about nodes and their corresponding edges. This TC message broadcasted by a node in m_i also will include the node i that selected it. Using the information from the TC messages, every node is able to either build an adjacency list or matrix to represent its edges and can therefore construct a routing table.

List of Works Cited

Back, A. (1997, March 28). A partial hash collision based postage scheme. Retrieved from http://www.hashcash.org/papers/announce.txt

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., & Danezis, G. (2017). SoK: Consensus in the Age of Blockchains. *The Alan Turing Institute*. Retrieved from https://arxiv.org/pdf/1711.03936.pdf.

Bairn, A. (2015, September 23). Ionization Protocol: Flood Routing. Retrieved from https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-September/000212.html

Bedford Taylor, M., & Taylor, M. B. (2013). Bitcoin and the age of Bespoke Silicon. In *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. https://doi.org/10.1109/cases.2013.6662520

Bitcoin Cash. (2018, April 24). Retrieved from https://en.wikipedia.org/wiki/Bitcoin_Cash

BITCOIN HALVING IN. (n.d.). Retrieved from http://www.thehalvening.com/

BOLT 7: P2P Node and Channel Discovery. (n.d.). Retrieved from https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md#recommendations-for-routing

Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. Retrieved from https://people.csail.mit.edu/alinush/6.824-spring-2015/papers/pbft.pdf.

Castro, M., & Liskov, B. (n.d.). Byzantine fault tolerance can be fast. In *Proceedings International Conference on Dependable Systems and Networks*.

https://doi.org/10.1109/dsn.2001.941437

Clausen, T. H., Jacquet, P., & Viennot, L. (n.d.). Investigating the impact of partial topology in proactive MANET routing protocols. In *The 5th International Symposium on Wireless Personal Multimedia Communications*. https://doi.org/10.1109/wpmc.2002.1088405

Confirmed Transactions Per Day. (n.d.). Retrieved from https://blockchain.info/charts/n-transactions

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., … Wattenhofer, R. (2016). On Scaling Decentralized Blockchains. In *Lecture Notes in Computer Science* (pp. 106–125).

Default TTL Values in TCP/IP. (n.d.). Retrieved from http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm

Donnelly, J. (2016, July 08). What is the 'Halving'? A Primer to Bitcoin's Big Mining Change. Retrieved from https://www.coindesk.com/making-sense-bitcoins-halving/

Drivechain. (n.d.). Retrieved from http://www.drivechain.info/

Dwork, C., & Naor, M. (n.d.). Pricing via Processing or Combatting Junk Mail. In *Lecture Notes in Computer Science* (pp. 139–147).

Hash Rate. (n.d.). Retrieved from https://blockchain.info/charts/hash-rate

Kaliski, B., Jr. (1993, November 1). A Layman's Guide to a Subset of ASN.1, BER, and DER. Retrieved from http://luca.ntop.org/Teaching/Appunti/asn1.html

Klitzke, E. (2017, July 20). Bitcoin Transaction Malleability. Retrieved from https://eklitzke.org/bitcoin-transaction-malleability

Lind, J., Eyal, I., Pietzuch, P., & Sirer, E. G. (2017). Teechan: Payment Channels Using Trusted

Execution Environments. Retrieved from https://arxiv.org/pdf/1612.07766.pdf.

Lombrozo, E., Lau, J., & Wuille, P. (n.d.). BIP141. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

Maccari, L. (n.d.). *OLSR: Optimized Link State Routing*. Lecture.

Maxwell, G. (n.d.). Compacted transaction. Retrieved from https://people.xiph.org/~greg/compacted_txn.txt

Maxwell, G. (n.d.). Test Lows in Standardness, Removes Nuisance Malleability Vector. by gmaxwell Pull Request #6769. Retrieved from https://github.com/bitcoin/bitcoin/pull/6769

Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf.

Pass, R., & Shi, E. (2018). Thunderella: Blockchains with Optimistic Instant Confirmation. *Advances in Cryptology – EUROCRYPT 2018 Lecture Notes in Computer Science,* 3-33. doi:10.1007/978-3-319-78375-8_1

Piatkivskyi, D., Axelsson, S., & Nowostawski, M. (2017). Digital Forensic Implications of Collusion Attacks on the Lightning Network. In *IFIP Advances in Information and Communication Technology* (pp. 133–147).

Prihodko, P., Zhigulin, S., Sahno, M., Ostrovskiy, A., & Osuntokun, O. (2016). Flare: An Approach to Routing in Lightning Network. Retrieved from http://bitfury.com/content/5-white-papers-research/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf

Russell, R. (2015, September 21). Ionization Protocol: Flood Routing. Retrieved from https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-September/000199.html

Segregated Witness Benefits was published on January 26, 2016 . (n.d.). Segregated Witness Benefits. Retrieved from https://bitcoincore.org/en/2016/01/26/segwit-benefits/

Sztorc, P. (2017, July 10). Updating the Scaling Roadmap. Retrieved from https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-July/014718.html

Technology roadmap - Schnorr signatures and signature aggregation was published on March 23, 2017 . (n.d.). Technology roadmap - Schnorr signatures and signature aggregation. Retrieved from https://bitcoincore.org/en/2017/03/23/schnorr-signature-aggregation/

Towns, A. (2015, September 15). Network Topology and Routing. Retrieved from https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-September/000188.html

Wuille, P. (n.d.). BIP66. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0066.mediawiki