Lot49

A lightweight protocol to incentivize mobile peer-to-peer communication

Richard Myers, Global Mesh Labs LLC (subsidiary of goTenna Inc.)
rich@goTenna.com

June 11, 2019
DRAFT version 0.8.5

Abstract: Mesh networks offer a decentralized alternative to centralized carriers and ISPs for mobile communication. To optimize successful message delivery, and improve network coverage and reliability, mesh nodes need a system to incentivize the relay behavior of peer network nodes. We propose a trust-minimized protocol for message senders to exchange incentive value with mesh nodes that relay their messages. Our approach is to create a payment channel network (PCN) based on the Bitcoin Lightning Network [12] but adapted for very low-bandwidth ad hoc mobile networks and therefore also applicable to less-constrained mesh topologies. To reduce incentive protocol overhead we propose using signature aggregation, simplex payment channel updates and payment channels formed between mesh nodes within direct communication range. Nodes primarily exchange payments to incentivize the delivery of their data without internet connectivity. Only when nodes less frequently establish, checkpoint or close payment channels must they relay payment data to a mesh connected internet gateway. This proposal requires that both the Schnorr signature [17] and SIGHASH_NOINPUT signature hash flag [43] protocol updates have been adopted by the Bitcoin community.

1. Introduction

It is estimated that people globally send over 80 billion mobile messages a day [1]. Mobile messaging has for many years been among the most popular uses of mobile phones worldwide [2]. However, consumers must rely on a few large corporations to provide mobile communication services. Many centralized mobile carriers and Internet service providers (ISPs) are also regional or national monopolies. Because of this centralization these services can be censored and surveilled, especially across political boundaries. Physically centralized infrastructure is also prone to catastrophic failure during natural or manmade disasters and often fails at serving last-mile communities. These problems exist despite the fact that most people possess mobile phones that can communicate directly with each other and bypass centralized infrastructure.

Mobile ad-hoc networks (MANETs) offer a promising solution to the problems of centralized carriers and ISPs. With the advent of smartphones, people now possess the technology required to power mobile mesh networks and communicate with each other in an entirely peer-to-peer manner. In a mesh network, data is transmitted directly to the devices of nearby people who relay and route the data until it reaches its destination. But to have the same utility as centralized networks, a mesh network must provide similar coverage and reliability.

Mobile mesh networks need a decentralized way to ensure relay and gateway nodes operate when and where they are needed to optimize coverage and reliability. We propose a protocol that allows data senders to pay relay nodes involved in data delivery with exchangeable, monetizable tokens. Nodes that receive tokens can use them to incentivize delivery of their own data or sell excess tokens to others. This allows people to earn value when they add value to the network.

1.1 Related Work

Early efforts to organize mobile mesh nodes in a decentralized way focused on embedding secure hardware modules [3] in devices and game theoretic solutions [4] to encourage nodes to cooperate. These approaches were primarily focused on preventing

routing misbehavior [5] and not on creating incentives to increase the coverage and reliability of the mesh network.

Some later proposals [6] introduced payments to incentivize relay behavior but required a trusted third party (TTP) to keep a ledger of payment balances. The advent of Bitcoin [7], a permissionless decentralized ledger system, made it possible to eliminate the TTP [8], but not to handle high transaction volumes on-chain due to scaling issues [9]. The invention of payment channels [10][11] and recent payment-channel networks (PCN) like the Lightning Network [12] now allows for efficient settlement of high volume micro-payments without a TTP and using untrusted relay nodes. Recent incentivized mesh projects [13][14] use payment channels to pay nearby nodes for providing internet access and bandwidth. These projects incentivize a certain quality of service (QoS) to transfer a quantity of data rather than charging for each data packet delivered.

For example, Althea Mesh[13] assumes fixed directional WiFi links between nodes. High-bandwidth links mean protocol overhead is negligible and nodes are assumed to have indirect access to the internet through gateways to set up and settle relatively long lived payment channels. These conditions are appropriate for fixed wireless community networks (WCN)[15] designed to replace wired internet connections, but do not apply for MANETs.

Mobile mesh devices rely on battery power and omni-directional transmissions that trade-off greater range for lower bandwidth. Network topologies in an ad hoc network will not be long lived because nodes move. Mesh subnets can become isolated from internet gateways so internet connectivity may be intermittent. These constraints require overhead be ruthlessly minimized because every transmission requires power and uses radio spectrum shared with nearby nodes. Payment channels must also be usable even when internet access is not immediately available for set up and settlement.

	WCN	MANET
Internet	Connected	Intermittent
Bandwidth	High	Low to High
Topology	Fixed	Dynamic
Power	Wall	Battery
Transmission	Directed	Omni
Metered	Amount of data at QoS	Per-packet delivered
Replaces	Wired Internet/ISP	Mobile Carrier/ISP

Table 1: Summary of differences between Wireless Community Networks (WCN) and Mobile Ad hoc Networks (MANET).

2. Overview

2.1 Goals

We propose a new payment channel communication protocol designed to incentivize the delivery of messages over even low bandwidth MANETs. We accomplish this by reducing the transmission overhead needed to make payments between nodes. Our protocol also minimizes the trust required between mesh nodes when not connected to the internet.

2.2 Approach

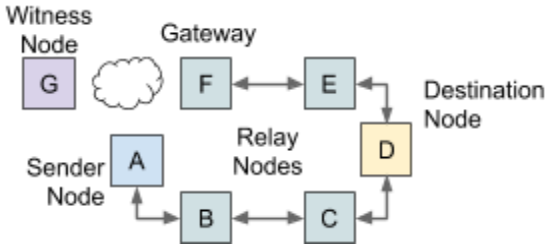

Figure 1: Mesh nodes are peers that take on different roles. In this example Node A is sending data (ie. SMS) to destination Node D and Nodes B and C are relaying the data. Once Node D confirms receipt of the message, anyone can send the signed incentive payment transactions made to the relays to a witness Node G via relay Node E and relay/gateway Node F connected to the internet. Node G settles these transactions on a global distributed ledger.

To reduce payment overhead that must be relayed to the internet via gateways, we use an *aggregate signature* scheme that maintains cryptographic security but only requires half the signature data. Signature data is the largest part of a transaction that

can not be inferred or compressed. Signature data is reduced by signers locally negotiating one aggregated signature instead of sending two signatures, one per signer, to confirm a transaction. This also reduces on-chain transaction fees.

After a payment channel has been established between nodes they only need to locally communicate with each other to update the state of their channel. We use an efficient commitment scheme to commit the payer to a new channel state without requiring the payment receiver to respond until they have proof that a message is delivered. This reduces total transmission overhead substantially.

We transmit to the internet only the minimum information necessary to reconstruct each *implied transaction* signed with a single aggregate signature. A small set of predefined transaction types also reduces the information that must be transmitted to reconstruct each implied transaction.

Payments to mesh relays and internet gateways are prepaid by the data sender and are only valid if the destination node returns a secret value created by the sender. This ensures that relay nodes are only paid for successfully delivered data. Destination nodes also earn value when they confirm they received data to ensure they cooperate in settling transactions.

Our system proposes the use of internet connected *witness nodes* to efficiently verify payment channel setup and settlement transactions for mesh nodes that are not directly connected to the internet. Nodes without internet connections communicate via gateway nodes that are both connected to the mesh and to the internet. All other transactions are negotiated locally between mesh nodes within direct communication range as payment channel updates. The latest channel update is cached for later online settlement if the channel is not cooperatively closed.

2.3 Protocol

In any mesh network, nodes transmit messages to nearby nodes that includes both a header and data payload. Payload data could be anything from an SMS message to arbitrary internet protocol packets. The header contains information on how to route the data payload to a destination node. Nearby relay nodes read and modify header information as they route the message to its destination.

Our incentive protocol builds on this existing behavior and adds additional header information to create and update payment channels between nodes. We also add an encrypted secret to the data payload that can only be revealed by the destination node; similar to the spontaneous payments proposed [45] for the Lightning Network. If a node has set up a payment channel with another node, they can authorize a payment to that node in exchange for it relaying some data to a destination node. Each relay node commits to pay the next node along the route and uses the same delivery condition. Each node proposes to pay the next node less than they received to transmit the data; the difference represents the value they earn for relaying the data. Each payment update must satisfy certain conditions to be valid. The primary payment condition is that the final destination node confirms receipt of the data by revealing a secret obtained from the data.

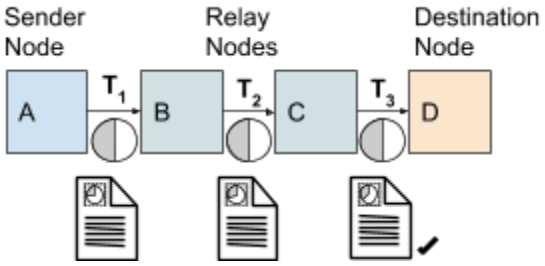

Figure 2: In this example, node A is sending data to node D via nodes B and C at time T_1. Nodes A&B, B&C and C&D have payment channels set up with each other and initially equal balances in the channels. Node A proposes to increase node B's balance if node D receives the data. Nodes B and C make similar proposals to their next hop.

Once the data has been received, the destination node transmits back a secret value to the data sender that satisfies the primary payment condition. This payment receipt is used by the nodes that relayed the data to update the state of their payment channels with each other.

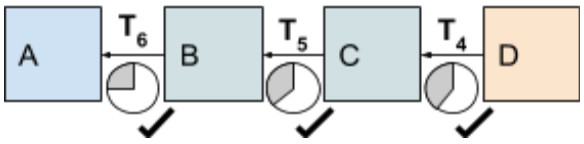

Figure 3: Node D has confirmed receipt of the data and finalizes at time T_4 the previously proposed balance transfer made by node C. Nodes C and B do the same with their respective upstream nodes B and A.

Once the destination node confirms receipt of the data by revealing a secret, any node that receives the

secret can settle their update transaction on the distributed ledger even if their channel partner disappears or becomes uncooperative. Nodes can observed transactions settled by other relay nodes involved with the same message delivery to learn the secret they need to settle their own channel updates.

When a payment channel does not already exist between two nodes it must be set up and funded. A transaction that funds a new channel can not be confirmed locally between mesh nodes because it involves a payment that could have been committed to fund a different channel. These transactions must be confirmed directly by the Bitcoin network before they are relied on.

A mesh connected node must use an internet gateway to access the Bitcoin network, but staying in sync with the state of the blockchain is impractical over a low bandwidth network. Instead, mesh nodes query a witness node they trust. Witness nodes are persistently connected to the internet so they can monitor and report in a low bandwidth way the current state of transactions of interest. Witness nodes will earn payments if they report truthfully about the state of the blockchain. They can be held accountable by checking the underlying blockchain data when a node has online access or by using multiple independent Witness nodes.

---

3. Concepts

Below we describe the foundational concepts used by our protocol. Those already familiar with the current Bitcoin Lightning Network can also skip to Appendix A for a technical overview of the changes we propose to create the Lot49 network.

3.1 Aggregate Signatures

The Bitcoin network uses the Elliptic Curve Digital Signature Algorithm (ECDSA) to sign transactions that transfer bitcoin from one address to another. Transactions are verified by checking associated ECDSA signatures. Even if other details of a transaction can be inferred or reconstructed, signatures must be supplied by the entity that authorizes a transaction. Because we want to optimize our protocol for low-bandwidth communication channels, we focus on how to reduce the size of the signatures that authorize transactions.

In particular, we would like a way to reduce the size of signatures for transactions with multiple signers. Multisignatures [18] offers a solution for combining the signatures from two or more signers on a single message into a single signature. This technique can dramatically reduces the amount of data needed to communicate a multi-signature transaction.

Unfortunately, no efficient scheme exists to aggregate ECDSA signatures so proposals for signature aggregation of Bitcoin transactions require a new signature algorithm. Schnorr signatures support has been proposed for Bitcoin [17] and would, among other things, provide support for the MuSig [18] multisignature schemes. One problem with using MuSig to reduce communication overhead is that it requires multiple rounds of interaction between signers. A suggestion for non-interactive half-aggregation [19] of Schnorr signatures has also been made, but would create signatures that grow in size with each signature added.

The BLS [20][21] signature scheme has also been suggested as a solution for creating multi-signatures for Bitcoin transactions [22]. BLS signatures can also be used for general signature aggregation [23]. Unfortunately, Bitcoin is unlikely to adopt BLS signatures due to their more complicated security assumptions and higher computational requirements for verification. Although BLS signatures require more computation to verify signatures, computation is relatively less power- and time-consuming than transmitting data for mobile devices [24]. For low-bandwidth mobile mesh networks the increased verification time of pairing based signature schemes like BLS is justified by the reduction in data transmission achieved using signature aggregation.

We discuss the trade-offs of using Schnorr multisignatures and BLS signature aggregation in Appendix D. Our conclusion is that Schnorr multisignatures have the best technical tradeoffs for this protocol.

3.2 Payment Channels

Payment channels [10][11] are used to make local bidirectional payments between nodes. They reduce the transactions that must be settled on the distributed ledger to only setup and settlement transactions. Update transactions exchanged directly between nodes to modify the current split of value in a shared

multisignature address do not need to be immediately communicated to an internet connected Bitcoin node.

Validating many small transfers between nodes using the Bitcoin blockchain directly would require internet access for each payment. With payment channels, value is locked in a shared transaction output which both nodes must authorize any spending from. Nodes directly negotiate how tokens stored in the shared transaction's outputs should be divided. Nodes only need to communicate the final balance split with a final settlement transaction sent to the blockchain once they are done exchanging payments.

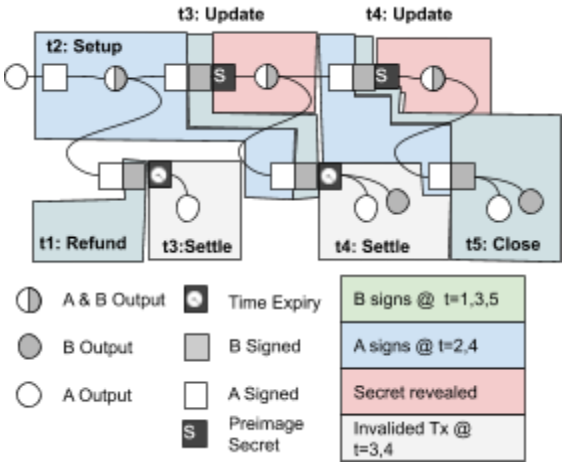

Figure 4: At time t1 Node B proposes a payment channel by signing a *Refund* transaction back to Node A. At time t2 Node A opens the channel by signing a *Setup* transaction to fund the channel and both paths of an *UpdateAndSettle* transaction to offer a new state update to Node B. To be valid the transaction requires Node B's signature and a secret that proves message *m* was delivered. Node B can update the transaction state at time t3 by adding their own signature and the secret for message *m* that proves it was delivered. At time t4 Node A can sign a final **Close** transaction to make the transfer to Node B final without a delay if Node B presents the secret that proves delivery for a new message *m*.

Payment channel transactions use scripts that encode the rules for how to update and settle the value split between nodes. These rules include a timeout so that if one node stops responding, the other can finalize the most recently settled split. This timeout is tuned to prevent a malicious node from trying to finalize an out of date balance update. In our usage, settlement transactions also require proof that a message has been delivered. This makes channel updates valid only when an associated message delivery is successful.

Spend from $Tx_{0,AB}$:
IF
  path 1 (settle)
  Delay >1 week
  $PkA_{S,0}$ & $PkB_{S,0}$ signs Tx_1
ELSE
  path 0 (update)
  State > 0
  PkA_U & PkB_U signs Tx_1
  preimage from m_1
ENDIF

Figure 5: Pseudo code for *UpdateAndSettle* output script that must be satisfied to spend from a payment channel transaction (eg. $Tx_{0,AB}$). The first path transfers tokens from the jointly controlled transaction 1 week after the last channel update was committed. The second path immediately transfers all of the value to a new $Tx_{1,AB}$ controlled by both Nodes A and B if the new transaction state is greater than the old state and if Node D has revealed the preimage associated with message m_1. If the update transaction is signed and settled before the 1 week delay, then the update to the next state of the channel (with a different value split) makes the old settlement transaction invalid.

Our payment channels are based on Decker-Russell-Osuntokun eltoo channels [25] and assume a conditional input script that either cooperatively transfers the tokens to an address with a newer state or settles the channel unilaterally after some delay (Figure 5).

For example, Node A commits to pay Node B to relay data m_1 to message destination node D by signing a settlement transaction that unlocks the balance controlled by $Tx_{0,AB}$ with output script *UpdateAndSettle* (figure 6). If Node B also signs the script and reveals the secret from Node D for the data m_1 then Node B can settle the payment channel one week after the transaction that funded $Tx_{0,AB}$ is settled. Tokens controlled by the transaction $Tx_{0,AB}$ will be distributed based on the allocation (eg. 7 to Node A and 13 to Node B) Node A committed to pay to Node B if data m_1 was delivered.

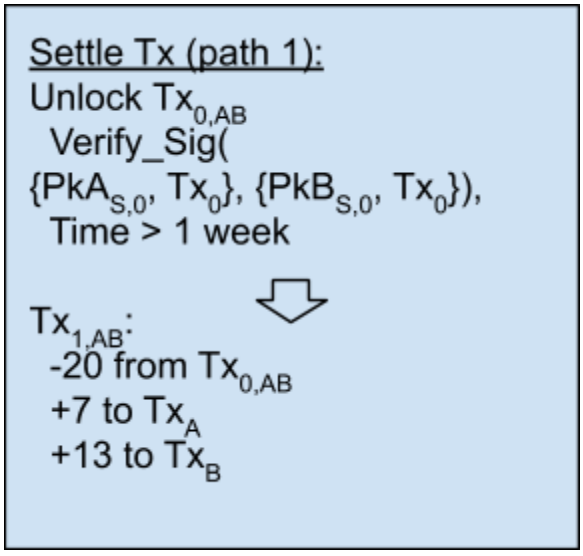

Figure 6: The pseudo code *Settle* input script is only valid if it includes signatures from the settlement public keys used for state 0 by Nodes A and B and 1 week has passed since the last update transaction.

Even if Node A becomes unresponsive, as long as Node B has proof that Node A also signed the transaction and Node D's signature for m_1 proving it was delivered, then Node B can settle the payment channel after one week. However, if Node B wishes to commit to a new split of tokens with Node A, Node B can simply prove to Node A they have Node D's signature for m_1. Both nodes can then safely negotiate a new split that assumes the previous agreed settlement split was confirmed. Node A can now sign a new transaction with the same *UpdateAndSettle* output script but with a newer state. Node A would sign a commit to the new split that pays Node B more tokens that the previous split conditioned on proof that some new data was delivered to a destination node.

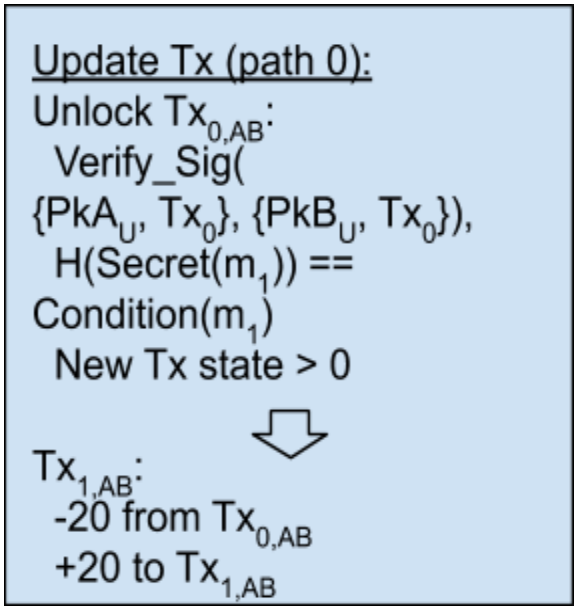

Figure 7: The pseudo code *Update* input script immediately transfers value from the previous state to a new one controlled by both Nodes A and B if it includes signatures from Nodes A and B, Node D's secret for message m_1 and if the state of the new transaction is greater then the current one.

In this way, nodes can make arbitrarily many bi-directional transfers between themselves without needing internet access to connect to the distributed ledger. If either node becomes unresponsive, then both will have enough information to settle the last payment they received after some delay. If at any point both nodes wish to settle the current state of their balances, they can cooperatively sign a closing transaction that does not have a 1 week delay encumbrance. The closing transaction can also be negotiated with an aggregate signature to reduce the amount of data that needs to be settled online.

3.3 Payment Channel Network

Our design builds off the concept of a payment channel network (PCN) pioneered by the Bitcoin Lightning Network [12]. In this type of network, pairs of nodes connected by payment channels are organized into a connected graph that can safely route payments between nodes that do not have a direct payment channel established.

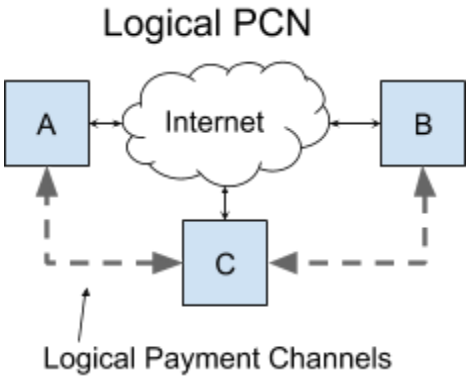

Figure 8: Existing PCNs communicate via the internet and route payments via a logical network of payment channels.

We propose a system that routes both data and payments over the same point-to-point communication channels. Current PCNs form an overlay network that routes payments over logical connections between nodes, but communicate using a different set of internet connected data routers.

Our physical PCN creates payment channels directly between nodes within radio range based on their physical location. Both data and payments are routed over the same network topology.

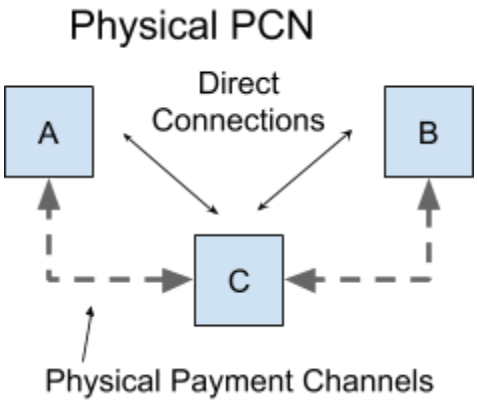

Figure 9: In a mesh based PCN nodes route payments and data together over the same physical links.

Existing online PCNs assume internet connectivity and direct access to the distributed ledger to confirm channel funding transactions when a new channel is set up.

In our physical PCN, transactions to set up and settle payment channels on the distributed ledger must be relayed through the mesh to an internet gateway.

3.4 Implied Transaction

Nodes transmit incentive headers to commit to payment channel transactions that transfer value from one node to another. The commitment to a transaction can be verified by checking the signatures created for these transactions. These transactions are not explicitly included in each incentive header. Instead nodes only transmit the information for each transaction that cannot be otherwise inferred from other information in the transmission.

For example, an incentive header might include a payment channel Update Tx transaction sent from Node A to Node B but does not need to include the payment addresses for Nodes A and B. These addresses can be deterministically computed by looking up a cached extended public key (xpub) [26][27] for the nodes involved. For example, addresses can be deterministically generated by multiplying cached xpubs for Nodes A and B by a known nonce, such as the hash of the message. xpubs can be exchanged and cached when new node IDs are first encountered or synced when a node is online.

Nodes must also aggregate their signatures for an implied transaction to be valid. Any node that reconstructs an implied transaction can use the aggregate signature to confirm the correct implied payment addresses were included in the implied transaction.

An important implied value in our protocol is a secret value computed from the payload data. This value is committed to by each message sender along with their payment. The message sender generates a secret nonce for each message and encrypts it with the message payload. Only the destination node can decrypt the payload and recover the secret nonce.

Payment channel update transactions are not valid unless the data being sent is received by the destination node; this condition can only be satisfied after the destination node decrypts the message and returns the secret nonce.

Transactions are also predefined to eliminate values that can be standardized. For example, we can create a standard relay fee where each node pays one less token than it received. This information need not be transmitted to reconstruct the implied transactions.

Knowing the starting relay payment amount and how many hops have occurred is enough to reconstruct the payment amounts of every implied transaction. We can also allow nodes to set different values for the payment they will take. For example, a node that bridges two large subnets could charge more than a node within a large well connected network. Variable relay fees can be implemented by adding a few more bits of information to the hint for each hop.

Large values like extended public keys can be looked up in locally-stored caches from shorter node IDs that map to the larger public key value. If a particular value is not already locally cached, it can be transmitted first in long-form and from then on only the shorter hash ID need be sent.

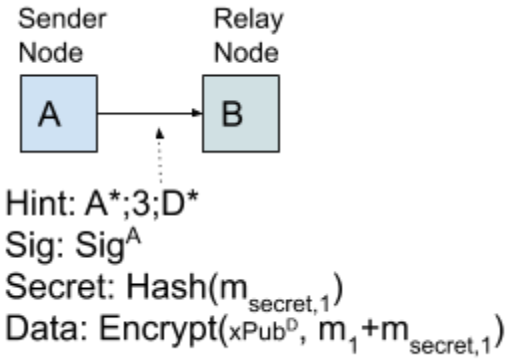

Figure 10: An implied transaction transmitted from Node A to Node B includes hints that imply a transaction where Nodes A will pay 3 tokens to Node B if Node D signs for receipt of data m_1 by returning a value that matches the preimage condition $Hash(m_{secret,1})$. The normal routing header should already include the IDs for Nodes A and D, so the only additional information that must be transmitted is the number of tokens Node A spends (eg. 3) and the signature signed by Node A (eg. Sig^A).

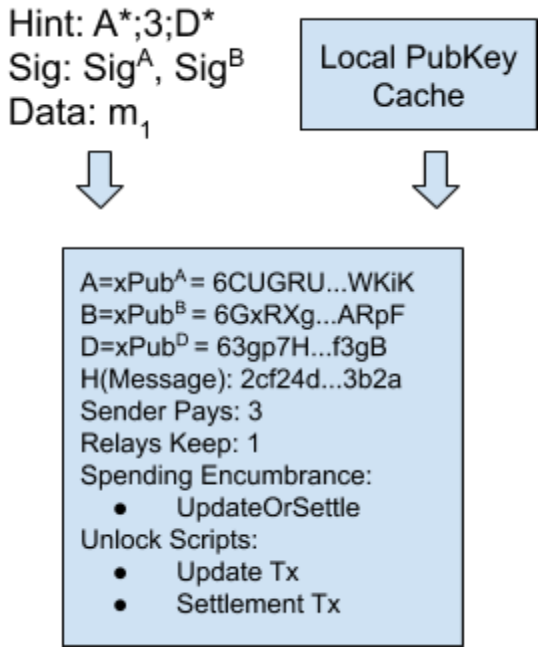

Figure 11: Node B can reconstitute the full signed transaction from the hints and routing header sent by Node A and from their public key cache. The implied transaction updates the payment channel between Nodes A and B to a new state if Node B has the preimage encrypted for Node D with message m_1.

3.6 Witness

When a new payment channel is set up, a funding transaction commits tokens to the channel. Until the funding transaction is validated and settled by the consensus mechanism of the blockchain, there is a risk the funding payment can be double-spent.

Nodes could keep an up-to-date view of settled transactions by operating as a full node that witness all new ledger updates as they are produced. However, over a low bandwidth mobile mesh network the overhead of broadcasting this information to all nodes is impractical. Instead, offline mesh nodes will delegate this task to online witness nodes they operate themselves or trust.

Online witness nodes should always return the secret value included with the transaction that they receive from offline nodes to indicate they received the request. Witness nodes should also return a message to the requesting offline node that commits to whether or not the transaction is valid on the Bitcoin blockchain.

Interactions between offline mesh nodes and internet connected witness nodes are handled like a normal communication between nodes, but must include an internet gateway node. Validating a transaction requires making a payment to the relay nodes and gateway node that relays the information across the mesh network to the internet. Like other destination nodes, witness nodes will receive payment for confirming receipt of a valid payload, in this case the payload is a transaction that must also be confirmed by the Bitcoin network. A payment sent to a witness node could also require one of two valid preimages. This would enable the witness node to confirm the message was received and signal to the sender that the transaction was successfully confirmed or not depending on which preimage they return.

3.7 Settlement

We propose using the Bitcoin blockchain as a permissionless distributed ledger to maintain an authoritative global record of incentive payment transactions. A distributed ledger ensures there is no central point of failure that could compromise the operation of the system. The alternative to this approach would be a centralized ledger run by a single or federated group of operators. A centralized ledger would be technically easier to scale by leveraging well-established database technology, but would not be as socially scalable [28] or as resilient to downtime. Trusted third parties are single points of failure that introduce the same censorship and rent-seeking opportunities available to centralized ISPs and mobile carriers.

Bitcoin was the first successful permissionless decentralized ledger network and remains the most reliable example. It has proven its resilience with over 99.99% uptime since 2009 [29]. It's ability to resist attacks in a highly adversarial environment has established the Bitcoin network as the world's preeminent decentralized financial ledger.

We also plan to use Bitcoin in order to take advantage of the many scaling and security features that have been tested and refined for the Bitcoin protocol. However, to allow us to verify transactions over bandwidth constrained mesh networks composed of mostly offline nodes we need to a support for a signature aggregation scheme like MuSig. For this reason we do not propose to initially settle transactions on the existing Bitcoin ledger, but will instead use a new Bitcoin testnet that includes the modifications we need to test the Lot49 protocol. In particular, Schnorr signatures [17] and the SIGHASH_NOINPUT signature hash flag [43].

During the initial testing phase of the network we anticipate running a single centralized instance of the ledger. During this phase token issuance and redemption will be pegged to some nominal unit of value. This approach removes speculative price volatility and ensures tokens will only be acquired to be used for data delivery. Tokens will be redeemed by users that have earned an excess by their efforts relaying data for others.

Once the necessary protocol changes have been adopted by the Bitcoin community on the main Bitcon network, we will transition to using it for settlement. We could also potentially use a federated sidechain [38] to create a network with value pegged to the main Bitcoin blockchain if the protocol changes we expect to use are not adopted by the Bitcoin community.

---

4. Protocol Details

Below we describe in more detail the components and mechanism of our proposed incentive protocol.

4.1 Nodes

Mesh nodes generally use the same hardware and software, but serve different roles in the process of transferring data through the mesh network. Depending on the situation, they can be the sender, relayer or destination for data. Relay nodes can also act as gateways for sending data from a local mesh network to a regional or global network like the internet.

Nodes connected to the internet may also act as a provider of blockchain transaction services. For bandwidth constrained networks we expect blockchain access will involve private interactions with an online trusted witness node.

Sender Node

A node that wishes to send data commits to spend a specific number of incentive tokens conditioned on confirmation the data is delivered. A sender commits to spend more tokens to improve the likelihood of a

successful delivery, increase the distance data will be relayed and/or incentivize a gateway to relay the data. The mesh routing protocol will generate an estimated delivery cost as part of its routing metrics and can update its estimates based on the delivery success of prior messages.

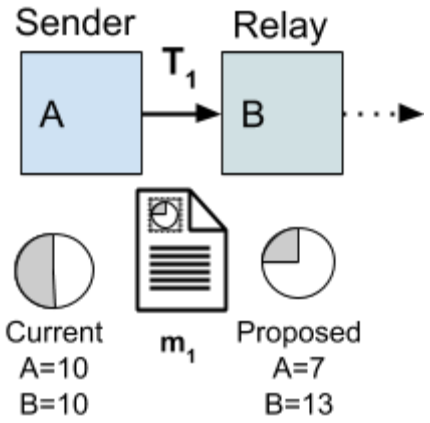

Figure 12: Sender Node A sends payload data m_1 to relay Node B and proposes to increase Node B's share of their joint account by 3 tokens in exchange for proof that destination Node D received m_1.

Relay / Gateway Nodes

A node that receives a message from a sender or other relay node will cache and potentially retransmit the message. A relay may use heuristics based on both topological and incentive costs to determine the likelihood a message will ultimately be delivered and use this information to decide whether or not to relay it. A node's current battery charge and its other cached messages are also considered. A relay may also prioritize messages from senders with a confirmed commitment of value to the payment channel or if it helps to rebalance a channel. A relay node that also has internet access can earn value as a gateway by advertising this via the routing protocol. Nodes either charge a standard price to relay, or advertise their fee in advance as part of their routing metrics.

Destination Node

A node that receives a message for which it is the destination acts much like a relay node, but they can immediately confirm the payment channel update that increases their channel balance. When they reveal the encrypted secret that came with the message they received, any node involved in the delivery of the message can also update or settle their associated transaction.

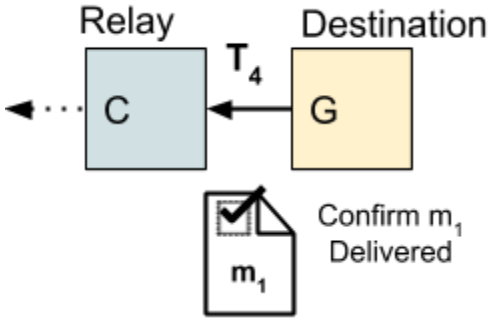

Figure 13: Node G produces the secret that proves delivery of payload data m_1. Once their delivery receipt is received their last corresponding update and settle transactions becomes valid and can be recorded on the Bitcoin blockchain.

All relay nodes involved with delivering the message can use the final revealed secret to confirm their own payment channel updates. Once a node has a delivery receipt, they can perform a unilateral close of the channel if a node they relayed for becomes unresponsive. After a node receives proof that a message they relayed is received, they can negotiate with their upstream relay node to update or cooperatively close their respective payment channel balances instead of immediately committing the confirmed update to the blockchain.

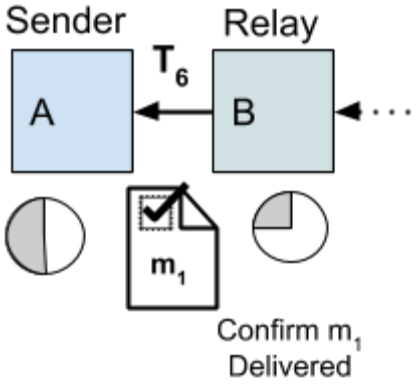

Figure 14: Node B negotiates with upstream Node A by sending the delivery receipt secret produced when destination Node D received m_1. If Node A can not be reached, Node B can unilaterally use this information to settle the update on the blockchain. However, instead of settling, Nodes A and B can keep the channel open and create their next update with the balance agreed to by the last valid update.

Nodes that re-transmit message delivery receipts do not need to wait for an on-chain confirmation of an incentive payment. Also, as they re-transmit the payment receipt back to the message sender, they encourage more messages to be sent along their

route. Proof of delivery implies future transmissions along the same route will also result in payment. Extra incentive payments can also prepay a message receiver to reply. Any reply could also include the previous delivery receipt secret to increase efficiency.

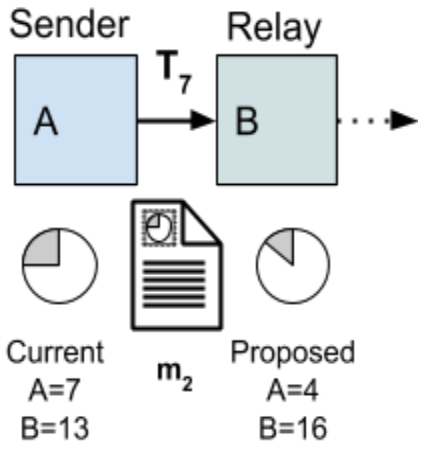

Figure 15: Node A proposes to send a new message m_2 and update their balance with Node B again. Node B has already proven that Node D received m_1, so Node A knows Node B can settle the last update directly on the blockchain without Node A. When Node A proposes a new update, they must treat the previous update as the current state otherwise Node B will not accept a new update from them.

Witness Node

Witness nodes are expected to be persistently connected to the internet and operated by mesh network users to facilitate more efficient confirmation of settled transactions. A witness node receives transactions as a data payload from mesh nodes that wish to have payment channel set up and settlement transactions confirmed by the Bitcoin blockchain. If a transaction is valid, but not yet confirmed on the ledger, the witness node will submit the transaction to the blockchain and wait for it to be confirmed. They then send the secret included with the transaction payload back to the original node that requested it be validated.

Instead of running their own online witness node, off-grid mesh nodes can also use nodes run by others they trust and can incentivize their function of validating transactions by including payments that the witness node can only collect if the transaction is confirmed on the blockchain. It also may be possible to redeem prepaid Lightning invoices with online nodes to trustlessly fund new channels. [44]

4.2 Data Relay

Here we describe in detail the steps required for nodes to send and receive token payments in exchange for delivering data on the mesh network.

Sync

To optimize for low bandwidth communication between mesh nodes, information necessary for incentive payments should be cached and inferred instead of transmitted whenever possible.

We assume a sync phase where nodes periodically connect to the internet directly and cache information useful for recreating implied transactions. Nodes cache this information and can also broadcast it over the mesh network to nodes that do not have synchronized caches. Nodes can also publish their own public keys and cache metadata useful to optimize message delivery such as their geolocation and the witness node(s) they trust to verify transactions. The information for each node is associated with a unique short node ID used to look up each node's extended public key. The node ID is also used by the routing protocol to identify nodes.

Node information can be published to witness nodes who gossip it to other nodes and archive it for offline nodes. Cached data is signed by the public keys they are associated with and can have an expiration date to protect against syncing wrong or out-of-date data.

After the initial sync phase, offline nodes will have a cached version of the extended public key, incentive balances and other hints of all other nodes. To reduce the size of this information nodes can choose to sync only information from nodes that register as being active in a subset of geographic locations.

Delivery

A node normally sends data to another node using the standard MANET routing protocol supported by the device. Our optional incentive protocol appends additional header information to the standard network level headers and uses augmented routing heuristics. Nodes will prioritize the relay of messages that include valid incentive headers in order to earn incentives from successfully delivered data.

Before incentivized message delivery proceeds, mesh nodes must build their routing tables—starting with learning about nearby nodes. Once a node identifies nearby nodes from their broadcasts it can propose to create payment channels with them. A node must have a payment channel with any node identified as the next hop to deliver a message to. A node sending data negotiates channel updates conditioned on their data being received. Each relay node follows the same procedure as the sending node, but negotiates to pay the next node less than they receive.

The sender encrypts a secret with the message they send to the destination node. Any node involved in relaying the message commits to a payment to the next relay hop if the decrypted secret is presented. Transactions that create or settle a payment channel are sent to an internet connected witness node. When nodes renegotiate a local payment channel update they invalidate the previous channel update. A new channel update must be published within a typically long time window (eg. one week) after the last update to prevent an outdated update from becoming final.

Six message delivery protocol steps are described below:

Step 1: Discover Topology

Our incentive protocol assumes an "on-demand" style routing protocol. Before routing tables have been populated, messages are sent using flooding. Flooded messages do not specify the ID of the node that should be the next hop for delivery. During this phase, the incentive protocol is not active. As messages are successfully delivered, routing information propagates between nodes to populate their routing tables. Nodes also advertise the cost for different routes so this information can be considered when routes are selected.

Step 2: Propose Channels

To create a payment channel between nodes A and B requires both nodes sign a series of transactions spending from a 2-of-2 multi-signature address. Suppose Node B will receive payments from Node A first. Then Node B proposes the channel to adjacent Node A who will fund it. Because the transactions are partially signed by Node B before Node A funds the channel, Node A can be confident that a refund transaction from the multi-signature address will be valid if Node B becomes unresponsive.

Node B can propose opening a channel any time after receiving a route discovery flood message that includes Node A's ID. Node B includes in their transmission an incentive header (Setup_1) that partially signs a set of transactions necessary to open the payment channel funded by Node A.

Step 3: Send Message

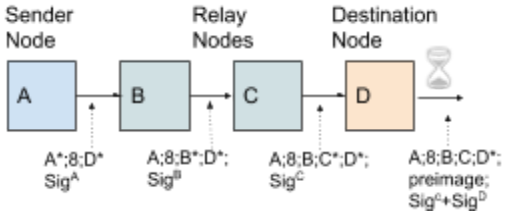

Figure 16: Node A sends a message to Node D. This example shows the additional information (eg. "A*;8;D*, Sig^A") that must be transmitted between nodes to enable the reconstruction of the implied transactions. The Node IDs with asterisks do not need to be transmitted because the information already exists in the network headers required to deliver the message. The initial amount of tokens spent by Node A to send the message is 8 and Sig^A is Node A's signature for the first set of implied transactions. The final transaction sent by Node D has a complete set of signatures and the preimage so it can be settled on the blockchain.

After receiving a payment channel proposal from Node B, Node A has what they need to send a message to some other node in the network via Node B. For example, when Node A sends a message to Node D along a route that includes Node B as the first hop. Node A includes in their transmission an incentive header (Setup_2) with a signature to add value to the payment channel with node B and an incentive header (Negotiate_1) or if the channel has already been funded adds header (Negotiate_2). These headers update the channel to transfer some amount of incentive value to Node B. Node B will only be able to settle the payment channel update if Node D reveals the preimage secret to confirm they received the message from node A. Each transaction in the chain of relay nodes contains the same condition: to reveal a preimage included in the message that Node A sent to Node D.

If Node C is the next hop that Node B must use to relay the message to Node D, Node B must follow a similar procedure as Node A to create and/or update a payment channel with node C. Likewise, if node C is adjacent to Node D, then they create and/or update a channel paying any remaining tokens to Node D.

Step 4: Message Received

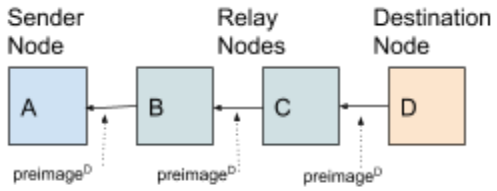

Figure 17: Node D sends a return receipt message containing the preimage to Node A. This example shows that Node B must inform Node A that Node C relayed the message to Node D; the other nodes can infer this when validating preimage is correct. The final signed transaction and preimage can be used by each node to negotiate a channel update with its upstream partner or to immediately close the payment channel.

When the message arrives at its destination—for example, Node D has received the message from Node A—then Node D updates the state of their payment channel with Node C by transmitting an incentive header (Receipt_1) that includes their preimage confirming receipt of the message. Now the transaction is complete and can be presented to a witness node to be published on the global blockchain. After Node C receives the message receipt confirmation they can include an incentive header (Receipt_2) to update the state of their payment channel with Node B. Node B follows the same procedure to update the state of their payment channel with Node A.

Once a node has updated the state of their payment channel they can send additional messages that update the channel state or cooperatively close the channel.

Note that a node that receives incentive payments from a newly created channel can use that value in the reverse direction without themselves committing value. In this situation they are refunding credit they received from an upstream node. It is only important to receive confirmation from an online witness node of the on-chain settlement of a channel in situations where a node does not reciprocally relay messages for another node. The reverse acknowledgement shown in Figure 16 is used to updated channel states and so relaying it does not require additional incentive. The data receipt acknowledgement can also be combined with new data flowing in the reverse direction for efficiency.

Witness

The message delivery steps described above commit to incentive payment transactions that must be settled on the distributed ledger to be finalized. Until the transaction that funds a payment channel has been settled, any transaction chain that includes a payment from that new channel could become invalid. Likewise, a final channel update transaction must be settled within a specific time after the payment channel is closed.

Nodes can continue to use a channel with an unverified funding transaction, but if the funding transaction ultimately does not verify, then any payments they made to other nodes for delivery can still be collected by those originating nodes.

When a payment channel between two nodes is closed, either cooperatively or unilaterally, the last update should also be recorded or verified within a standard timeout window to prevent an old update from being settled.

To verify and settle transactions we add steps 5 and 6 to the base message delivery protocol:

Step 5: Record Transactions

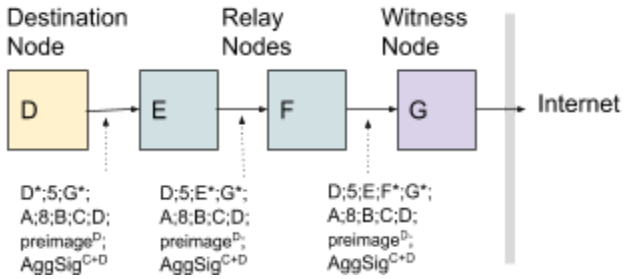

Figure 18: Node D sends a message to witness Node G containing the hints for the transaction chain from Figure 16. Node G will verify the closing transaction with the multisignature from Nodes C and D, and if valid, publish it to settle the transaction.

Any node that has an incentive transaction can settle it on the blockchain if it includes the message receipt preimage from the destination node required to make the transaction valid. To do this, a node can repeat step 3 with a message composed of the "hint" information needed by a witness node to reconstruct the transaction to be verified or recorded. The destination node for this message is a witness node trusted or operated by the sender. The "hint" information and incentive header must pass through

an internet gateway if the sending node is not connected to the internet.

The node that sends a message containing a transaction to be verified or recorded must include an incentive token payment that will be collected by the relay nodes and the witness node that verifies or records it. So settling a transaction requires creating a second transaction with relay incentives from the sending node to the witness node. Using an aggregate signature created by both channel parties reduces the amount of data that must be verified to confirm a transaction.

Step 6: Verify Transactions

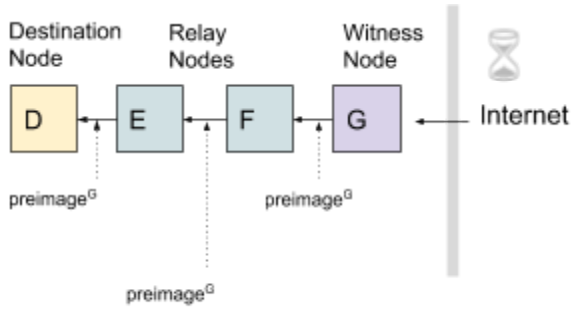

Figure 19: Witness Node G returns their preimage as proof of receipt of the transaction sent by Node D and returns the preimage to confirm that the transaction has been received. Witness node G must send a separate message to D to indicate if the transaction is verified and settled on the blockchain.

A witness node receives, reconstructs, validates and records incentive transaction chains and receives an incentive payment in the same way as other relays or message receivers. Once the publishing transaction is sufficiently confirmed in the distributed ledger, the witness node can also provide a message that proves the transactions is settled, in addition to providing their preimage. The witness node sends an incentive header (Receipt_1) back through the mesh network to the node that requested the transactions to be recorded in the same way as step 4 above. Any node that trusts the witness node can now depend on transactions that were included in or depend on the now confirmed transaction chain.

5. Safety

Mesh nodes by default relay data without payment. Our incentive protocol augments the default relay heuristics to encourage better delivery for senders that spend value to incentivize relays. Misbehaving relay nodes should not be able to gain unearned value, but we can not always prevent limited free data delivery for data senders. We would like the network to err on the side of data being delivered.

With this framework in mind, we examine below some problems and general protections against degenerate behavior by nodes.

5.1 Offline Nodes

The normal security assumption for PCNs includes the possibility that nodes could be offline and not monitoring the blockchain. During this offline period, a malicious counter party could try to settle an out-of-date payment channel transaction. The offline node must react to this attempt and publish a more current state to prevent the malicious counterparty from rolling back payments.

Because mesh nodes will be offline most of the time we must consider this situation carefully. Our protocol must adapt the techniques used by other PCN protocols to the constraints of a mesh network.

Long Timeouts

The primary security for offline nodes comes from long delays for transactions that unilaterally close a channel. This gives the other party time to notice and submit a more current update. Our system will have to use timeouts adapted to the characteristics of the off-grid community. If users typically have internet access at home or via gateways once a day, then timeouts can be set shorter than if access is once a week. One consequence of this is that the less frequently nodes have internet access, the more incentive they have to cooperatively close channels to prevent locking up their liquidity.

Watchtowers

The Lightning Network white paper contemplates a private and trust minimized third party, also called a “watchtower”[12][30], who could be incentivized to

always be online to watch for channels settled with out of date update transactions. It is more difficult to outsource channel monitoring for an offline mesh network node because it requires sending encrypted transactions to the watchtower. This may not be possible in a bandwidth efficient way over a mesh network. Instead it is more likely that witness nodes could be incentivized to monitor and notify offline nodes when channels are settled as an additional service to offline nodes. Watchtowers could also be opportunistically sent the latest fully signed update transaction held by a node. These checkpoints can be sent when near a gateway and could only be used by the watchtower when an invalid update appears on the blockchain.

5.2 Other Risks

Invalid Transactions

Channel updates from newly created payment channels should be considered potentially invalid until confirmed on the distributed ledger. If a channel funding transaction is included, but not ultimately confirmed, the transaction will become invalid. This means a relay node could owe their downstream node value when the channel funding transaction from their upstream node fails to validate.

For nodes that reciprocally relay messages for each other in roughly equal amounts, it is less critical how long it takes to confirm channel funding transactions on-chain because their channel balances will mostly net out close to zero.

Each node along a route can heuristically decide whether or not to relay a message when the transaction chain includes an unconfirmed transactions that funds a new channel. Priority would be given to messages that are not at risk from an unconfirmed channel funding transaction.

Unsigned for Delivery

The data receiver must reveal a preimage derived from the data for the incentive payment associated with that data to be valid. There is no way for the last relay node to know if their next transmission, which includes the data payload, will be overheard by the message receiver. A malicious message receiver could receive a message and choose not to acknowledge receipt.

Message receivers also receive incentive payments when they sign to prove they received the message. This is the primary way we encourage message receivers to follow the incentive protocol. However, when a message sender and receiver are colluding, or controlled by the same entity, this payment may not be relevant. For example, if the sender and receiver are the same entity then the sender can avoid the cost to send a message if the receiver always fails to sign and prove they received it. This situation can only be solved by nodes noticing the pattern and blacklisting responsible nodes.

Sybil Relays

Message destination nodes should receive whatever value was committed by the sender, but not used by relay nodes. This discourages relays from claiming more fees than they advertised. A message destination node that does not collect any value when they receive a message may not reveal the preimage that proves they received it. Also, the more value received by the destination node the more likely they are to forward the transaction to an internet gateway to be confirmed on the blockchain which reveals the preimage to other relays nodes that depend on that preimage. If a relay wishes to earn more value they can publish a higher relay cost so nodes can incorporate this information in their routing metrics.

False Witness Nodes

A witness node can falsely confirm for an offline node that a transaction is valid. This could result in a relay node accepting payment from a new payment channel that in fact was not funded. This can be discouraged by using multiple witness nodes and by rechecking transactions when a node has direct internet connectivity. Witness nodes will earn tokens for confirming transactions and so have an incentive to perform their function honestly. Because anyone can run a witness node, those with the best uptime and longest track record should be favored.

Transaction Fees

The fees to settle transactions on the blockchain may exceed the value transferred as a micropayment to deliver a message. This could lead to payment transactions that are uneconomical to spend in the case of an uncooperative payment channel close.

5.3 Solutions

The risks described above would all require nodes to run a malicious version of the node client software. It is important that these attacks can at best result in free message delivery, but not result in unearned fungible value. This means only nodes using the system for message delivery have a reason to attack it. Solutions that isolate and reduce delivery success for malicious nodes should be sufficient to prevent attacks. Only an attack that could not be detected would be worth implementing.

Local Blacklists

If a node does not properly follow the incentive protocol, then any nearby nodes will overhear it at the RF radio layer. Locally misbehaving nodes can be safely ignored. For example, a destination node that does not sign to prove they received a message is either offline, out of range or misbehaving. To rule out channel errors, nodes should not be ignored until enough evidence of misbehavior has been observed. Whatever the reason, local nodes can safely stop routing messages to them. The incentive protocol extends trust to nodes beyond the range a message sender can directly monitor by allowing nodes to monitor transactions and identify routes more or less likely to result in confirmed message delivery.

Message receivers could still collude with senders to periodically change their identity, move and/or wait for new peers that have not witnessed their misbehavior when sending or receiving messages. However this would require establishing new payment channels and propagating new routing information to relays each time a node changes its identity.

Relay Heuristics

Ultimately, what is most important for a relay node that earns incentives for message delivery is how likely it is that a given message they relay will result in a successful and acknowledged message delivery. Delivery failure can occur for many reasons. The destination node could be offline, isolated or malicious. Downstream relays may likewise have communication failures that result in a message not being delivered. The only reliable data a node has is their own gathered statistics about success and failure for the messages they have sent or relayed. Heuristics based on this information are the best way for nodes to prioritize which messages they will relay.

A local heuristic that optimizes successful message delivery is also good for the network at-large. Flooding the network with messages that are unlikely to be received decreases the capacity of the network for everyone. Each node must balance their personal cost to transmit messages against the likelihood of successful delivery. A node must also participate in the overall relay protocol to be discoverable and learn new routes.

A node unconstrained by power or opportunities to transmit can always relay every message it receives. A power-constrained node or one located in a critical location may be unable to relay every message it receives. This should lead successful nodes to adopt a strategy that prioritizes relaying incentivized messages that are most likely to be delivered.

Rational Micropayments

To solve the problem of settling micropayments that are not economically viable due to transaction fees various solutions have been proposed [41]. For our system we believe probabilistic micropayments would be a viable option if they are supported by the Bitcoin script language.

It is also possible to amortize payments over multiple messages if the payment for a single message is not economical. In this case relay nodes take a larger risk that they will not receive payment until multiple messages have been delivered. However, this would be no worse than accepting micropayments that they are not able to spend.

Another approach currently implemented in the Lightning Network is to ‘trim outputs’ [42] that are below a predefined value. The value from trimmed outputs are committed to the transaction fee instead of creating an unspendable output. Trimmed outputs can later be credited to economically rational outputs when a payment channel is settled.

---

6. Incentive Economy

Any node can earn a reward for relaying data for others and by being at the right place at the right

time. The position of a mesh node relative to other mesh nodes determines whether their channel balance is likely to increase, decrease or stay even. Being an internet gateway, or near one, is also likely to increase the value earned due to increased traffic and the ability to confirm newly created payment channels.

6.1 Relay Pricing

A node that originates a message must determine how much to prepay the relay nodes that deliver their data. Each relay node can set the price they expect in order to retransmit a packet of data. Gateway and witness nodes may also set prices for their services.

Relays will charge a *price* to relay that adequately compensate them for their *cost* to transmit a message plus some fee. For mobile nodes their cost derives from the opportunity cost of using battery power to transmit for other nodes instead of reserving it for their own use. Gateway and witness nodes can include a larger fee based on the demand for their services. Relay nodes that bridge subnets could also add a higher fee than those where alternative routes exist.

Nodes should also adjust their relay price to account for the PDR (packet delivery ratio) of the network. The lower the PDR the more packets a relay will have to transmit before one is delivered. Relays are not compensated for data that is not ultimately delivered. The price a relay node *i* should charge for a single relay transmission is then:

$$price_{relay_i} = (cost_{relay} + fee) \times \frac{1}{PDR}$$

Each relay node advertises their transmission price plus the minimum price advertised by all nodes *n* in the set of neighbor nodes *N* that advertise a route to the destination node.

$$price_{relay_i \rightarrow dest} = price_{relay_i} + \forall n \in N,\ min(price_{relay_n \rightarrow dest})$$

Neighboring nodes that include too high a fee or require more intermediate nodes to reach a destination will advertise a higher relay price and not be selected as the next hop to deliver a packet.

6.2 Issuance

Nodes must obtain bitcoin that can be spent to incentivize other nodes to transmit their data. Bitcoin can then be redeemed by users that provide relay services to the network in excess of their use of the network.

For this analysis we will consider the primary use to be sending short messages (eg. SMS, Whatsapp, etc). More data intensive applications over higher bandwidth links would require more initial value to be stored per device.

We can assume each device using the protocol sends 50 messages per day, each message requires no more than 10 hops to be delivered and relay nodes expect to earn one satoshi per hop. Using these assumptions we can predict a device uses roughly 500 satoshi per day.

We need to also factor in that there is a delay between when value is earned and when transactions are settled and the bitcoin becomes re-spendable. This delay depends on how often offline nodes are expected to access the internet. If we assume users have internet connectivity at least every seven days, then each device needs a minimum of 3500 satoshi to incentivize every message they send.

A device that only spends tokens, but never relays for others, will run out of bitcoin after about a week. However, if a node relays more messages then it sends, it will end the week with a net gain in bitcoin. Whether a node gains or loses value depends on the behavior of the user (eg. a node must be kept powered on to relay) but also on the location of the device and the topology of the network.

6.3 Central Nodes

Nodes that are situated in a central location within a well connected mesh network will easily earn value for relaying but tend to not receive more messages to relay then they are capable of relaying. Central nodes are unlikely to have a backlog of messages because they must share the local radio channel with other nearby nodes. They will earn incentives steadily but also spend them regularly in equal proportion for a net even balance.

6.4 Bridge Nodes

Nodes located in a sparsely meshed position that bridges two more connected meshed areas are more likely to receive a backlog of messages to relay, and will relay more for others than for themselves. These nodes will be able to prioritize messages with incentive value attached and are more likely to be net earners of incentive value.

6.5 Gateway Nodes

Nodes that provide a gateway to the global internet are also likely to be net earners of incentive value. Nodes that often send messages via gateway nodes will likely have their incentive balances decrease over time. These nodes will likely be a bottleneck and prioritize messages with incentive tokens.

6.6 Edge Nodes

Nodes that are not a bridge between subnets, but are located at the edges of a mesh network will have fewer opportunities to relay for other nodes. These nodes will likely have their incentive balances decrease over time unless they become a gateway or bridge node. These nodes represent "last-mile" nodes and any relay nodes that serve them will be net earners of tokens.

6.7 Confirmed Channels

A relay node should expect to receive more messages and earn more value from a route that includes only confirmed payment channels. This gives nodes some incentive to proactively confirm transaction chains that include funding transactions for new payment channels. All nodes, including the message receiver, are more likely to receive incentive value from channels with confirmed funding transactions.

6.8 Free Relay

Nodes need not always spend or require incentive to relay data. A node may send a message without adding an incentive header and only route through nearby nodes that likewise do not advertise a relay fee. These messages will be delivered as long as a path of free relay nodes is found. Sending to a node within direct radio range also does not require incentives.

For example, when nodes only send short distances and have many nearby nodes available to relay. In these cases the incentive protocol adds overhead but may not improve message delivery. Relaying for free can also help nodes build their routing tables and become discoverable by other nodes in the network.

For incentive payments to become useful, nodes using the incentive protocol must create a network with better message delivery than would emerge with only free relay behavior. This is more likely when sending data across many hops and when bridging isolated networks. Incentives also improve local delivery for networks that suffer from the "free-rider" problem where many nodes only send data but do not relay it.

6.9 Resale

A robust market already exists for converting bitcoin to local fiat currency. The price nodes charge to relay data will ultimately float based on the exchange between bitcoin and fiat currencies.

6.10 Coverage

It is not necessary to separately incentivize nodes to provide coverage in a particular area. Incentives for message delivery will encourage nodes to operate where there is demand. It is also easy to spoof the proofs of location that any proof of coverage system would rely on.

---

7. Analysis

In our simulations, we analyze the effect of adding our incentive protocol to the IETF standard on-demand routing protocol called "Ad hoc On-Demand Distance Vector" routing (AODV) [33].

7.1 Delivery Ratio

We model the effect of increased packet sizes due to additional incentive protocol overhead. We compare

delivery rates with and without the addition of a nominal 50 byte incentive header.

Incentive headers are appended to the standard routing headers that accompany payload data. The largest part of an incentive header is a 64 byte signature. We assume additional hint information of approximately 6 bytes plus 4 bytes per hop with an average of 3 hops for an estimated total of 100 bytes per incentive header.

We use an ns-3 [34] simulation of a mesh network composed of 30 nodes running the AODV routing protocol at different node densities. Nodes randomly move and run an application that transmits a 50 byte payload to another random node selected from a fixed subset of originating nodes. Nodes send data every 30 seconds for a total of 20 minutes with some initial random start time and transmit at a data rate of 100 Kbps.

Our simulations focus on packet delivery ratio (PDR) which measures the ratio of packets sent versus the number delivered.

We look at different ratios of nodes originating data versus those that only relay data for others. Non-originating nodes are meant to model incentivized nodes that are available to relay data even when they are not sending or expecting to receive data themselves.

Our simulations looked at the following questions:

1. The effect on PDR of adding an additional 50 byte incentive header to each 50 byte payload.
2. The effect of increasing the proportion of non-originating nodes due to incentivization.

Figures 19 shows that adding an additional 50 bytes of incentive header does not significantly reduce the PDR in our simulations except modestly for low network densities.

Figure 20 shows the effect of increasing the proportion of non-originating (relay only) nodes in our simulations. Relay only nodes represent the effect of incentives to motivate people to leave their nodes on when not actively using them.

Overall these results show that the decrease in PDR caused by incentive header overhead is modest and only at low node densities. Incentives that increase the proportion of non-originating nodes increase PDR, especially at low densities.

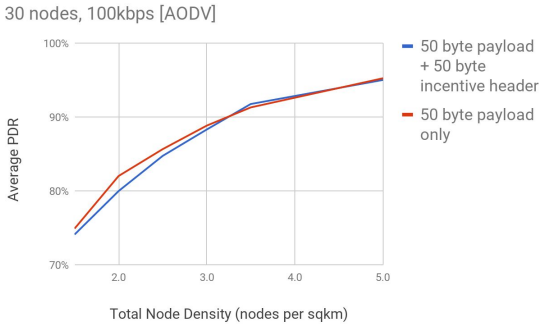

Figure 20: This graph shows the average PDR using the AODV protocol on a network with 30 nodes at different densities.

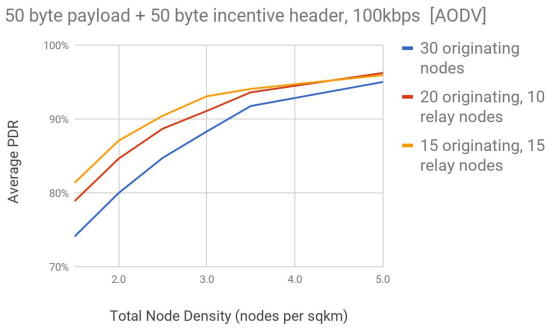

Figure 21: This graph shows the average PDR using the AODV protocol across a range of node densities. We compare a network of 30 originating nodes to one with a third and a half non-originating relay only nodes.

---

8. Future Work

Our proposal describes a design sufficient to incentivize nodes in a mobile mesh network, but there are other areas worth exploring over time that would increase the reliability and speed of the network, as well as minimize the need for internet connectivity.

- **Incentivized Flood.** Including incentive payments during "Flood" mode message relay could extend the protocol to encourage route discovery. In this mode nodes could receive the same messages from different routes and would relay the one that arrived along the shortest path (or that pays the most tokens).

- **Channel Factories.** Groups of nodes that are well connected on a mesh subnet, but are not well connected to the internet, could fund channel factories [35] to facilitate dynamically creating pairwise payment channels with each other while offgrid.
- **Local Rebalance.** A protocol to detect and rebalance payment channels among nearby nodes. This could also be done with a routing protocol metric that tries to maintain balanced payment channels with nearby nodes.
- **Block digests.** Broadcast compact block filter digests using the scheme of BIP-158 [37] in order to keep nodes on a mesh subnet in sync with the global blockchain with minimal access to the distributed ledger. This method would reduce the need for trusted witness nodes.
- **Geographic Sharding.** Allow mesh nodes to lock payments to specific geographic areas. This would allow for constructing and broadcasting smaller block filter digests specific for different geographic regions. Because nodes know a priori their location, they can reject funding transactions that are not locked to their current geographic location.
- **BLS Signatures.** Instead of using Schnorr based multisignatures, BLS-based signature aggregation could potentially be used to settle multiple transactions with a single short signature.
- **Offline Channel Funding.** Fund new offline channels using Lightning payments from an online Lightning node using previously shared invoices with long timeouts and known preimages. [44]
- **Proofmarshal.** Witness nodes could return a compact commitment to a state of the blockchain to confirm the existence of a transaction in the UTXO set at a given time. Similar to [47].

---

9. Conclusion

We have designed a system that can securely and efficiently incentivize data delivery over low bandwidth MANETs. Existing payment channel protocols do not focus on efficiently routing data and payments at the same time over multiple low bandwidth hops. We solve this problem by using a more data efficient channel update scheme and signature aggregation to settle transactions. We believe our system opens up a flexible design space for optimizing the coverage and reliability of mobile mesh networks in a way that preserves their decentralization.

---

10. Acknowledgments

Our work was inspired and is made possible by the development of the Bitcoin Lightning Network by Poon and Dryja, the eltoo update scheme proposed by Decker, Russell, and Osuntokun and the MuSig signature aggregation scheme of Maxwell, Poelstra, Seurin and Wuille.

We would like to thank Daniela Perdomo and the goTenna team for their generous support of this project. Thanks to Albert Wenger and Nick Gross from USV for their encouragement, review and suggestions. Thanks to Jay Graber for early discussions that led to this proposal.

We are also grateful to the readers of early drafts of this proposal who generously took time to give us useful suggestions, constructive criticisms and thoughtful feedback: Alex Akselrod, Bryan Bishop, Nic Carter, Robin Chan, Arthur Demeester, Adam Gibson, Fabian Jahr, Bob McElrath, Steve Myers, Elaine Ou, Mats-Erik Pistol, Teemu Päivinen, Ram Ramanathan, Dan Robinson, Lucas Ryan and Peter Todd.

References

[1] Goode, L. Messenger and WhatsApp process 60 billion messages a day, three times more than SMS. The Verge. URL: https://www.theverge.com/2016/4/12/11415198/facebook-messenger-whatsapp-number-messages-vs-sms-f8-2016. Accessed, 2018-09-29.

[2] Follow the leaders: highest ranking Apps in Google Play Store, United States. SimilarWeb. URL: https://www.similarweb.com/apps/top/google/app-index/us/all/top-free. Accessed, 29-Sep-2018.

[3] Buttyán L, Hubaux JP. Mobile Networks and Applications (2003) 8: 579.

[4] Wu F, Chen T, Zhong S, Li L, Yang R. (2008). Incentive-compatible opportunistic routing for

wireless networks. In Proceedings of the 14th ACM international conference on Mobile computing and networking (MobiCom '08). ACM, New York, NY, USA, 303-314.

[5] Marti S, Giuli TJ, Lai K., Baker M (2000). Mitigating routing misbehavior in mobile ad hoc networks. In Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom '00). ACM, New York, NY, USA, 255-265.

[6] uncorrelated, "A very rough draft of a new Hocnet Whitepaper • r/hocnet," reddit. URL: https://old.reddit.com/r/hocnet/comments/204kma/a_very_rough_draft_of_a_new_hocnet_whitepaper/. Accessed, 29-Sep-2018.

[7] Nakamoto, S (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing list at https://metzdowd.com. URL: https://bitcoin.org/bitcoin.pdf Accessed, 29-Sep-2018.

[8] Open Libernet [DRAFT] (2015). URL: http://www.openlibernet.org/paper/open-libernet.pdf. Accessed, 15-Aug-2018.

[9] Croman K, Decker C, Eyal I, Gencer A, Juels A, Kosba A, Miller A, Saxena P, Shi E, Gün Sirer E, Song D, Wattenhofer R (2016). On Scaling Decentralized Blockchains. In: Clark J., Meiklejohn S., Ryan P., Wallach D., Brenner M., Rohloff K. (eds) Financial Cryptography and Data Security. FC 2016. Lecture Notes in Computer Science, vol 9604. Springer, Berlin, Heidelberg.

[10] Bitcoin Wiki. Contracts: Example 7: Rapidly-adjusted (micro)payments to a pre-determined party. URL: https://en.bitcoin.it/wiki/Contracts#Example_7:_Rapidly-adjusted_.28micro.29payments_to_a_pre-determined_party. Accessed, 29-Sep-2018.

[11] bitcoinj. Working with micropayment channels. URL: https://bitcoinj.github.io/working-with-micropayments. Accessed, 29-Sep-2018.

[12] Poon J, Dryja T (2015): The bitcoin lightning network. URL: https://lightning.network/lightning-network-paper.pdf. Accessed, 29-Sep-2018.

[13] Tremback J, Kilpatrick J (2017). Althea: An incentivized mesh network protocol. URL: https://altheamesh.com/documents/whitepaper.pdf. Accessed, 29-Sep-2018.

[14] Ernst J, Wang Z, Abraham S, Lyotier J, Jensen C, Quinn M, D'Souza A (2017). Decentralized Mobile Mesh Networking Platform Using Blockchain Technology and Tokenization. URL: https://www.rightmesh.io/docs/RightMeshTechnicalWhitepaper3.1.pdf. Accessed, 29-Sep-2018.

[15] Frangoudis P, Polyzos G (2011). Wireless Community Networks: An Alternative Approach for Nomadic Broadband Network Access. Communications Magazine, IEEE. 49. 206 - 213.

[16] Boneh D, Gentry C, Lynn B, Shacham H (2003). A Survey of Two Signature Aggregation Techniques. CryptoBytes. 6.

[17] Wuille P. bip-schnorr.mediawiki. URL: https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki. Accessed, 29-Sep-2018.

[18] Maxwell G, Poelstra A, Seurin Y,Wuille P (2018). Simple Schnorr Multi-Signatures with Applications to Bitcoin. IACR Cryptology ePrint Archive 2018: 68 (2018).

[19] Dryja T. [bitcoin-dev] Per-block non-interactive Schnorr signature aggregation. URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-May/014272.html. Accessed, 29-Sep-2018.

[20] Boneh, D, Lynn B, Shacham H (2001). Short Signatures from the Weil Pairing. In: Boyd C. (eds) Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science, vol 2248. Springer, Berlin, Heidelberg.

[21] Chia BLS Signatures implementation. URL: https://github.com/Chia-Network/bls-signatures. Accessed, 29-Sep-2018.

[22] Boneh D, Drijvers M, Neven G (2018). Compact Multi-Signatures for Smaller Blockchains. IACR Cryptology ePrint Archive 2018: 483 (2018).

[23] Boneh D, Gentry C, Lynn B, Shacham H (2003). Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: Biham E. (eds) Advances in Cryptology — EUROCRYPT 2003. EUROCRYPT 2003. Lecture Notes in Computer Science, vol 2656. Springer, Berlin, Heidelberg.

[24] Szczechowiak P, Oliveira L, Scott M, Collier M, Dahab R (2008). NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In: Verdone R. (eds) Wireless Sensor Networks. EWSN 2008. Lecture Notes in Computer Science, vol 4913. Springer, Berlin, Heidelberg.

[25] Decker C, Russell R, Osuntokun O (2018). eltoo: A Simple Layer2 Protocol for Bitcoin. https://blockstream.com/eltoo.pdf. Accessed, 29-Sep-2018.

[26] Maxwell G (2011). Deterministic wallets. URL: https://bitcointalk.org/index.php?topic=19137.0. Accessed, 29-Sep-2018.

[27] Wuille P. BIP-32, Hierarchical Deterministic Wallets. URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki. Accessed, 29-Sep-2018.

[28] Szabo N (2017). Money, blockchains, and social scalability. URL: http://unenumerated.blogspot.com/2017/02/money-blockchains-and-social-scalability.html. Accessed, 29-Sep-2018.

[29] findbl0k. Bitcoin Uptime. URL: http://bitcoinuptime.com Accessed, 29-Sep-2018.

[30] The MIT Digital Currency Initiative. watchtower - watch channels for fraudulent transactions. URL: https://github.com/mit-dci/lit/tree/master/watchtower. Accessed, 29-Sep-2018.

[31] Dryja T (2017). Unlinkable outsourced channel monitoring. Scaling Bitcoin 2017. Transcribed by Bryan Bishop (kanzure). URL: https://diyhpl.us/wiki/transcripts/scalingbitcoin/milan/unlinkable-outsourced-channel-monitoring

[32] Herlihy M. (2018). Atomic Cross-Chain Swaps. URL: http://arxiv. org/abs/1801.09515. White Paper. Accessed, 2018-08-29.

[33] Perkins C, Royer E (2000). The ad hoc on-demand distance vector protocol. In: Ad hoc Networking, Addison-Wesley, pp. 173–219, 2000.

[34] NS-3. URL: http://www.nsnam.orgl. Accessed, 2018-08-01.

[35] Burchert C, Decker C, Wattenhofer R (2017). Scalable Funding of Bitcoin Micropayment Channel Networks. In: Spirakis P., Tsigas P. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2017. Lecture Notes in Computer Science, vol 10616. Springer, Cham.

[36] Gentry C, Ramzan Z (2006). Identity-Based Aggregate Signatures. In: Yung M., Dodis Y., Kiayias A., Malkin T. (eds) Public Key Cryptography - PKC 2006. PKC 2006. Lecture Notes in Computer Science, vol 3958. Springer, Berlin, Heidelberg.

[37] Osuntokun O, Akselrod A. BIP 158: Peer Services. URL: https://github.com/bitcoin/bips/blob/master/bip-0158.mediawiki. Accessed, 2018-04-13.

[38] Back A, Corallo M, Dashjr L, Friedenbach M, Maxwell G, Miller A, Poelstra A, Timón J, and Wuille P (2014). Enabling Blockchain Innovations with Pegged Sidechains. https://blockstream.com/sidechains.pdf. Accessed 12-Feb-2019.

[39] Wahby, R. and Boneh, D. (2019). Fast and simple constant-time hashing to the BLS12-381. elliptic curve. Cryptology ePrint Archive, Report 2019/403. ia.cr/2019/403. Accessed 23-Apr-2019.

[40] Snigirev, S. (2018). BLS signatures: better than Schnorr. Medium.com post. cryptoadvance/bls-signatures-better-than-schnorr-5a7fe30ea716. Access 23-Apr-2019.

[41] Discussion topic “HTLCs don't work for micropayments”? on StackOverflow.com . https://bitcoin.stackexchange.com/questions/85650/htlcs-dont-work-for-micropayments. Accessed 16-May-2019.

[42] Basis of Lightning Technology (BOLT) specification. Bolt #3.

https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md#trimmed-outputs. Accessed 2019-May-16.

[43] Decker C. bip-0118.mediawiki. URL: https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki. Accessed, 16-May-2019.

[44] Comte V. Insight #3: Running a Lightning enabled Point of Sale offline. URL: https://www.puzzle.ch/de/blog/articles/2018/11/21/takeaways-from-the-lightning-hackday-in-nyc. Accessed, 17-May-2019.

[45] Roasbeef. [WIP] multi: add new draft key send mode for spontaneous payments. URL: https://github.com/lightningnetwork/lnd/pull/2455. Access, 29-May-2019.

[46] Dryja, T. (2019).Utreexo: A dynamic hash-based accumulator optimized for the Bitcoin UTXO set. Cryptology ePrint Archive, Report 2019/611. ia.cr/2019/611. Accessed 4-June-2019.

[47] Todd, P. (2017). Scalable Semi-Trustless Asset Transfer via Single-Use-Seals and Proof-of-Publication. URL: https://petertodd.org/2017/scalable-single-use-seal-asset-transfer#proof-of-publication-ledger. Accessed 4-June-2019.

Appendix A. Technical Overview

Lot49 vs. Lightning Network

To reduce the amount of data exchanged between nodes for each payment we use a mechanism that requires fewer rounds of communication. To reduce the amount of data that must be settled on the blockchain through internet gateways we use a signature aggregation scheme. Specifically, this project proposes using the eltoo payment channel update mechanism and signature aggregation using the MuSig scheme as described below. Using these techniques can reduce the amount of signature data exchanged between nodes by approximately 50%. With these changes we believe incentive overhead can be reduced to near 100 bytes per transmission on average.

Other ways Lot49 deviates from the existing Bitcoin Lightning Network are described below:

Data Delivery

The Lot49 protocol is primarily designed to use micropayments to incentivize successful delivery of data in a peer-to-peer communication network. Where possible we minimize the incentive protocol overhead required for micropayments to increase the bandwidth available for data delivery. Payments to relay nodes are small and fixed, or are based on the amount of data transferred, not the amount of value transferred. This proposal should be compatible with the Sphinx keysend system if that proposal is extended to support sending arbitrary data over the Lightning Network.

Network Topology

The Lot49 protocol assumes a mesh radio network where nodes can only communicate directly with other nodes within radio range. A micropayment is committed by a node to incentivize a nearby relay or gateway node to deliver their data. A node only directly incentivizes nodes they can directly communicate with. No such constraint exists in the Lightning Network because you can route a payment though any internet connected node.

Bitcoin Network Access

Nodes do not always have direct access to the internet to confirm bitcoin transactions on the Bitcoin network. Any communication with the Bitcoin network is via a gateway node with a direct connection to the internet. Sending transaction information to the Bitcoin network using relay and gateway nodes may require including incentive payments just like any other data originated from an off-grid node.

We assume the most bandwidth efficient method for off-grid nodes to confirm and commit transactions is to communicate with an internet connected node they control or trust. Other light client techniques are possible but not contemplated by this proposal.

Mesh Routing

Data is source-routed using a low-overhead technique specific to the mesh radio network being used. We assume nodes start with knowing a route, if available, to the node they wish to communicate with. The use of incentives to exchange accurate routing information or cryptography to validate route information is not contemplated in this proposal. We also do not assume using onion routing to keep route information private.

Inferred Transactions

To reduce the payment overhead communicated between nodes we assume only the minimal amount of data needed to update their payment state is ever communicated. This is similar to the way the Lightning Network peer-to-peer protocol currently works, as described in Bolt #2. Our proposal assumes even more standardization of default values to further reduce the amount of data transmitted between nodes. We also assume using hierarchical deterministic wallets, or similar techniques, to pre-share public key information between nodes. Because signature information represents the large set of data that can not be inferred and must be transmitted, we attempt to reduce the total amount of signature data that must be broadcast over the network.

Payment Network

Each message sent via a relay or gateway node includes a message header which commits to pay for proof the message was delivered. Enough value is committed to pay every relay node and the message receiver a previously agreed amount. Each message must also include the preimage that unlocks the HTLC incentive payments. The preimage should be encrypted so that only the message receiver can decrypt it, and only if they receive the full message.

eltoo

We propose using the eltoo channel state update mechanism because it allows simplex payments between nodes to update their payment state. After a channel has been set up, only the node that is forwarding a payment must transmit their signatures to the payment receiver. The payment sender does not need the payment receiver's signatures because they will always have their own signatures for the last payment they received, or for the opening refund transaction. This reduces the amount of data that must be communicated to update a channel by more than half. With the current Poon-Drjya update scheme both sides need to transmit two signatures and revocation information from the last state.

MuSig

When two nodes commit to an eltoo trigger, refund or closing transaction, they would normally need to communicate two 64 byte signatures to the Bitcoin blockchain to spend from the 2-of-2 multisig transaction. The MuSig signature aggregation scheme can reduce this to a single multisignature for both signers. In the general MuSig signing session case, each signer transmits 96 bytes of information in three communication rounds to create a single 64 byte multisignature. However, for the special case of a two-party session, only one node needs to send a nonce commitment before the other party sends their nonce, thus saving 32 bytes. This means that two nodes exchange only 32 bytes more data to create an aggregate signature than if they exchanged two full 64 byte signatures. Each relay hop that the transaction must be sent to reach the internet requires half the bandwidth when sending a single multisignature compared to sending two 64 byte signatures.

Open Questions

Public Keys

This public key information must be exchanged between payment channel partner that cosign transactions. The eltoo protocol also requires using new public keys for every new settlement transaction. The most space efficient way to exchange public keys is with some pre-shared table that maps the extended public key of a node to a shorter node ID. The nonce or index to specify the exact path of each public key would also need to be sent for each transaction. We have not specified how this public key information will be communicated and used to recreate complete transactions that can be settled on the Bitcoin network.

Bitcoin Script

We have not yet specified how the Bitcoin scripts that would be used differ from the ones described in the eltoo paper and lightning network BOLT specification. We also need to specify how to reduce the amount of data needed to reconstruct these scripts from the data communicated between nodes.

Peer-to-Peer Messages

We still need to specify the exact messages and message passing sequence used to set up, update, and close payment channels between nodes.

Off-chain Optimizations

Techniques that minimize the amount of data that must be communicated via internet gateways and settled on the Bitcoin network deserve further exploration. Ideally we could setup, close and monitor channels with minimal or no communication through internet gateways. It should also

be possible to amortise many payments and channel operations into a small number of infrequent on-chain transactions. Some potential ways to do this include:

Channel Factories

Instead of always anchoring payment channels with a new on-chain set up transaction, the eltoo update mechanism enables the use of channel factories to negotiate opening and closing channels with a federation of available off-grid nodes. Channel factories can also reduce the time required to create new channels between off-grid nodes.

Light Client Support

Broadcast block headers and compact block filter digests using the scheme of BIP-158 in order to keep nodes on a mesh subnet in sync with the global blockchain with minimal access to the distributed ledger. This method would reduce the need for trusted witness nodes.

Appendix B. Incentive Headers

Header Name	Description	Nodes	Signed Transaction(s)
Setup_1	Node B signs a transaction similar to the initial Settlement Transaction from eltoo [25]. Node B signs to commit to refund the initial channel funding from node A. This header from B proposes creating a channel so that A can pay B.	Sender: B Receiver: A	Settlement Tx Signed by Node B
Setup_2	Node A signs a transaction similar to the initial Funding Transaction from eltoo [25] to commit the initial channel funding to a multisig address controlled by both Nodes A and B. This transaction can be fully refunded because Node B should have already signed a settlement transaction that refunds the value back to Node A.	Sender: A Receiver: B	Funding Tx Signed by Node A
Negotiate_1	Node A proposes an update to the payment channel balance that increases Node B's balance if Node D returns the preimage from the attached message m.	Sender: A Receiver: B Destination: D	Update Tx & Settlement Tx Signed by Node A

Negotiate_2	Node C proposes an update to the payment channel balance that increases Node D's balance if Node D returns the preimage from the attached message m. This is similar to Negotiate 1 but Node C proposes to pay the value committed by Node A to send the message, less the amount taken by relay nodes.	Sender: C Receiver: D Destination: D	Update Tx & Settlement Tx Signed by Node C
Negotiate_3	Node D proposes an update to the payment channel balance that increases Node C's balance if Node A signs the preimage attached message m. This is similar to Negotiate 2 but Node D proposes pays along an existing path to reduce incentive hint overhead.	Sender: D Receiver: C Destination: A	Update Tx & Settlement Tx Signed by Node D
Receipt_1	Node D proves delivery of message m by returning the preimage contained in message m and hashed with the message.	Sender: D Receiver: C	Update Tx & Settlement Tx & H(m) Signed by Node D
Receipt_2	Node C proves delivery of message m to renegotiate the state of their channel with B.	Sender: C Receiver: B	Update Tx & Settlement Tx Signed by Node C
Close_1	Node B proposes to close the channel with node A at the last negotiated distribution.	Sender: B Receiver: A	Close Tx Signed by Node B
Close_2	Node A agrees to close the channel with node B at the last negotiated distribution.	Sender: A Receiver: B	Close Tx Signed by Node A

Appendix C. Implied Transactions / Input Scripts

Transaction Name	Description	Nodes	Condition(s)
Setup Tx	Spend from 'Funding' UTXO to 'Setup' UTXO when a new channel is funded. Node A will not sign this until they have a signed Refund Tx from Node B.	Sender: A Receiver: B	A & B sign

Refund Tx	Spend from ‘Setup’ UTXO to ‘Funding’ UTXOs after some delay. Specifically created when a channel is first created.	Sender: A Receiver: B	A & B sign; Valid after a delay relative to when the transaction it spends from is committed.
Update Tx	Spend from ‘Setup’ or older ‘Update’ UTXO to 'Update' UTXO with newer state. Occurs immediately when this transaction is committed.	Sender: A Receiver: B	A & B sign; Current channel state > channel state of transaction it spends from
Settlement Tx	Spend from ‘Setup’ or ‘Update’ UTXO to final single signature ‘Funding’ UTXOs after some delay. Encodes a specific balance division.	Sender: A Receiver: B Destination: D	A & B sign; Valid after a delay relative to when the transaction it spends from is committed; D signs hash of Message
Close Tx	Unconditionally move funds from the most recent ‘Update’ UTXO to a set of balances at final single key ‘Funding’ UTXOs.	Sender: A Receiver: B	A & B sign

Appendix D. BLS versus Schnorr

	BLS	Schnorr	ECDSA
Signature Size (EC curve)	48 bytes (BLS12-381)	64 bytes (secp256k1)	Up to 72 bytes (secp256k1)
Signature Aggregation	Non-interactive, general aggregation	3 rounds (MuSig), multi-signature aggregation	No
Cryptographic Assumptions	CDH in a gap Diffie–Hellman group [20]	Same as ECDSA, simpler to implement	Computational Diffie-Helman (CDH)

Bitcoin Support	Unlikely	Proposed Softfork	Standard

Signature Size

BLS signatures use pairing friendly elliptic curves and currently the most popular choice is the BLS12-381 curve [39] which requires 48 bytes to represent a signature. Schnorr signatures proposed for Bitcoin use the same secp256k1 curve as ECDSA signatures currently used in Bitcoin. Because Schnorr signatures are not malleable, they can be represented as 64 bytes, whereas ECDSA signatures are variable sized up to 72 bytes.

Signature Aggregation

BLS signatures can be used for general non-interactive signature aggregation. This means that signatures on different messages can be aggregated into a single signature without additional communication rounds. The MuSig multi-signature scheme proposed for Bitcoin uses Schnorr signatures to aggregate multiple signatures on the same message. MuSig also requires three rounds of communication between signers to prevent certain attacks.

A MuSig like scheme could be used to sign the trigger and closing transactions between pairs of nodes. This would require exchanging 96 bytes instead of 64 bytes per signature, but could be improved if the commitment round is eliminated using zero knowledge proofs [18]; see Appendix E for a discussion the transmission overhead using this approach. Pairs of nodes can also eliminate one exchange of commitment information.

Cryptographic Assumptions

BLS signatures require additional cryptographic assumptions compared to Schnorr signatures. Schnorr signature implementations are also generally simpler than for ECDSA signature implementations and have similar cryptographic assumptions [40].

Bitcoin Support

Because of the difference in cryptographic assumptions, it seems likely that Schnorr signatures and a multi-signature scheme like MuSig will be adopted by the Bitcoin community before BLS signatures and general message aggregation. From a purely technical perspective BLS may be superior for the Lot49 use case, but that is less relevant if BLS does not have the support of a large community of cryptocurrency users. Currently the oldest and largest cryptocurrency community uses the Bitcoin protocol so this is an important factor in our choice of signature scheme.

Appendix E. Transmission Overhead

In this appendix we analyze the transmission overhead of three smart contract systems for paying incentives to mobile mesh nodes for successful message delivery. This analysis focuses

on the communication overhead to update channel states and close channels. We assume the channel setup overhead is roughly similar for all three. We also assume the same simplifying optimizations have been made to all three systems to minimize communication overhead. For each system we analyze the total communication overhead needed to update the payment channels of an example three hop message delivery. We also examine the amount of data that must be communicated to the internet to close all three channels.

The three smart contract systems are based on the following protocols:

- Poon-Dryja state revocation (with ECDSA signatures)
- eltoo state revocation (with Schnorr MuSig multi-signatures)
- eltoo state revocation (with BLS non-interactive signature aggregation)

The Poon-Dryja state revocation protocol requires multiple communication phases for both the sender and receiver of a payment to exchange commitments to the new state and revoke the previous state. The eltoo state revocation protocol uses transaction replacement and only requires the payment sender to commit to a new channel state. In the eltoo/Schnorr variant all nodes involved in the message delivery only sign two transactions to update their channels along the route. In the eltoo/BLS variant each node signs a transaction to update their channel with the next node in the route.

Schnorr and MuSig Multisignatures

The MuSig[18] signing scheme requires three rounds of communication for a set of nodes to collaboratively generate a multi-signature for a transaction. For each multi-signature, a node must exchange with each other signing node a 32 byte nonce commitment, a 32 byte nonce and a 32 byte partial signature. A valid multi-signature is composed of a 32 byte combined nonce and 32 byte combined partial signature.

This means that two nodes transmit 96 bytes to each other, and then either node can transmit a single 64 bytes of signature information to the blockchain. In total they transmit 256 bytes for a single signing session.

Without multisignatures, each node needs to transmit their complete 64 byte signature to the other node, and two 64 byte signatures to the blockchain for a total of 256 bytes.

The total is the same, but the ratio of local transmission versus final signature size is different.

Example

Assume four nodes create setup transactions for three channels (A <> B <> C <> D) with their neighbors. Each channel setup transaction requires two signatures. Also assume transactions must be relayed over 3 hops to be settled via an internet gateway node.

One Multisignature per channel

If we create one multi-signature with two signers per channel, then we must transmit 2 * 96 bytes (nonce commitment, nonce, partial signature) per channel to create one 64 byte multi-signature. The total transmitted to sign setup transactions for each of the three channels is

3 * 2 * 96 = 576 bytes. To transmit the multi-signature for a single channel three hops to a gateway requires transmitting 64 * 3 hops = 192 bytes, or an additional 576 bytes for three channels. The total transmitted is thus 576 + 576 = 1152 bytes.

No Multisignature

Without multi-signatures, each pair of nodes would exchange a 64 byte signature with their channel partner, or 64 * 2 * 3 channels = 384 bytes of information. Each of the three channels will also need to send 64 * 2 = 128 bytes of signature information to the blockchain over three hops for a total of 128 * 3 * 3 hops = 1152 bytes of transmission. The total data transmitted would be 384 + 1152 = 1536 bytes.

BLS Signatures

Each BLS signature is 48 bytes and can be non-interactively aggregated with other {message, signature} pairs. Signing three channels would require transmitting 48 * 2 * 3 = 288 bytes of signature data to create an aggregate signature of 48 bytes for each channel. The total data transmitted would be 288 + 48 * 3 * 3 hops = 720 bytes.

There are also additional optimizations that can be made that would reduce this number further such as only transmitting a single signature to sign all three setup transactions.

Conclusion

This analysis only focuses on signature data, which is the largest but not the only information that must be transmitted between nodes and between nodes and the internet via gateways.

While BLS signatures are by far the most space saving option, we can achieve a substantial improvement over current ECDSA multisig signing using MuSig and Schnorr signatures. Not only would this technique reduce overall signature transmission overhead by 50%, but it would reduce the amount of signature data that must be transmitted via relays to a gateway by 50%.

Importantly, Schnorr based MuSig multi-signatures are likely to be supported by Bitcoin in the foreseeable future, whereas BLS signatures are not currently supported by any widely used distributed ledgers.

Simplifying Assumptions:

We assume all three systems use the following simplifications and optimizations to reduce the bandwidth required to update their channel state.

- Each node has a short 2-byte node ID.
- The extended public key for each node ID is pre-shared.
- New public keys are deterministically derived from the extended public key for each transaction.
- A route of relays nodes from the sender to a destination node are computed by the sender and communicated in clear text (no onion routing).
- Transactions, including script details and defaults, can be inferred from a 1 byte type ID.

- The destination node signs a hash of the message to confirm receipt and unlock channel updates between nodes.

For this analysis, the following data structures are used to compute the overhead of simplified versions of the protocol messages described in "Bolt #2 : Peer Protocol For Channel Management." using the current Lightning Network's Poon-Dryja update scheme, or a proposed Lot49 scheme based on the eltoo update scheme.

LN Update Channel

The channel update phase (described in the Normal Operation section of Bolt #2) requires the update proposer to send the update_add_htlc message, and then both the sender and receiver to commit to a new channel state and revoke the previous channel state.

update_add_htlc

	Original Size (bytes)	Simplified Size (bytes)	Note
type	1	1	128 (update_add_htlc)
channel_id	32	-	inferred from destination node
id	8	2	ID to refer to this new HTLC
amount_msat	8	2	smaller max payment range
payment_hash	32	-	inferred from message data
cltv_expiry	4	-	use standard value
onion_routing_packet	1366	12	up to 6 clear text hops, one 2 byte node ID per hop
Total		17	

commitment_signed

	LN Size (bytes)	Simplified Size (bytes)	Note

type	1	1	132 (commitment_signed)
channel_id	32	-	inferred from destination node
signature	64	64	sign remote commitment
num_htlcs	2	-	assume single htlc signature
htlc_signature	num_htlcs* 64	64	commit to previous HTLC payments
Total		129	

revoke_and_ack

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	132 (commitment_signed)
channel_id	32	-	inferred from destination node
per_commitment_secret	32	32	revoke last commitment
next_per_commitment_point	33	33	new commitment point
Total		66	

Example

To update three channels requires transmitting the update_add_htlc message once per channel and the commit_signed and revoke_and_ack messages six times.

Unfortunately a Schnorr multi-signatures can not be used to reduce the overall amount of data transmitted between nodes to update a channel. Signatures are only included in the commitment_signed messages and nodes only update their own signature; the signatures from both sender and receiver are not transmitted together during a channel update. Both parties signatures are only combined when a channel is closed cooperatively by signing a partially signed closing transaction or uncooperatively by signing an unrevoked HTLC transaction

Message Type	Simplified	Total data transmitted by both sender and receiver (bytes)

	Size (bytes)	Update three channels ECDSA or Schnorr
update_add_htlc*	17	51
commitment_signed	129	774
revoke_and_ack	66	396
Total:		1221

* transmitted three times by payment sender only

LN Channel Setup

The setup phase for a new payment channel (described in the Channel Establishment section of Bolt #2) requires sending the following messages: the funding node sends 'open_channel', the receiving node sends back 'accept_channel', the funding node sends 'funding_created', the receiving node sends back 'funding_signed', the funding node sends 'funding_locked' and finally the receiving node replies with its own 'funding_locked' message.

For the simplified versions of these messages it is assumed that we can use defaults for many of the parameters exchanged to open a channel, and that information about the funding pubkey is transmitted, but the 'basepoint' information can be inferred. For example, using BIP-32 extended public keys, and agree (either in the protocol spec or in the protocol messages) on which BIP-32 child key derivation paths (child paths) to use.

open_channel

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	32 (open_channel)
chain_hash	32		Use defaults
temporary_channel_id	32	2	inferred from destination node
funding_satoshis	8	-	use defaults
push_msat	8	-	use defaults
dust_limit_satoshis	8	-	use defaults
max_htlc_value_in_flight_msat	8	-	use defaults
channel_reserve_satoshis	8	-	use defaults

htlc_minimum_msat	8	-	use defaults
feerate_per_kw	4	-	use defaults
to_self_delay	2	-	use defaults
max_accepted_htlcs	2	-	use defaults
funding_pubkey	33	33	DER Encoded secp256k1, half of 2:2 multisig
revocation_basepoint	33	-	inferred ?
payment_basepoint	33	-	inferred ?
delayed_payment_basepoint	33	-	inferred ?
htlc_basepoint	33	-	inferred ?
first_per_commitment_point	33	-	inferred ?
channel_flags	1	-	not used
shutdown_len	2	-	not used
shutdown_scriptpubkey	shutdown_len	-	not used
Total		36	

accept_channel

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	33 (accept_channel)
temporary_channel_id	32	2	inferred from destination node
dust_limit_satoshis	8	-	use defaults
max_htlc_value_in_flight_msat	8	-	use defaults
channel_reserve_satoshis	8	-	use defaults
htlc_minimum_msat	8	-	use defaults

minimum_depth	4	-	use defaults
to_self_delay	2	-	use defaults
max_accepted_htlcs	2	-	use defaults
funding_pubkey	33	33	DER Encoded secp256k1, half of 2:2 multisig
revocation_basepoint	33		inferred ?
payment_basepoint	33		inferred ?
delayed_payment_basepoint	33		inferred ?
htlc_basepoint	33		inferred ?
first_per_commitment_point	33		inferred ?
shutdown_len	2	-	not used
shutdown_scriptpubkey	shutdown_len	-	not used
Total		36	

funding_created

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	34 (funding_created)
temporary_channel_id	32	2	inferred from destination node
funding_txid	32	-	inferred
funding_output_index	2	-	inferred
signature	64	64	96 for MuSig interactive signing
Total		67	

funding_signed

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	35 (funding_signed)
channel_id	32	2	inferred from destination node
signature	64	64	96 for MuSig interactive signing
Total		67	

funding_locked

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	36 (funding_locked)
channel_id	32	2	inferred from destination node
next_per_commitment_point	33	-	inferred ?
Total		3	

Summary for Setup

Example

To setup three channels requires transmitting the above messages once per channel and the ‘funding_locked’ message twice per channel. Using Schnorr multi-signatures requires each signer send an addition 32 bytes as part of the interactive signing protocol.

Message Type	Simplified Size (bytes)	Total data transmitted by both sender and receiver (bytes)	
		Setup three channels ECDSA or Schnorr	Setup three channels with Schnorr Multi-Signature
open_channel	36	108	108

accept_channel	36	108	108
funding_created	67	201	297
funding_signed	67	201	297
funding_locked*2	6	18	18
Total:		636	828

LN Channel Close

closing_signed

	LN Size (bytes)	Simplified Size (bytes)	Note
type	1	1	39 (closing_signed)
channel_id	32	-	inferred from destination node
fee_satoshis	8	2	smaller max amount
signature	64	64	96 for MuSig interactive signing
Total		67	

Summary for Close

Both parties exchange close messages to settle immediately with the current channel balances. Using Schnorr multi-signatures requires each signer send an addition 32 bytes as part of the interactive signing protocol.

Message Type	Size Schnorr (bytes)	Total data transmitted by both sender and receiver (bytes)	
		Close three channels ECDSA or Schnorr	Close three channels with Schnorr Multi-Signature
closing_signed	67	402	594
Total:		402	594

Lot49 Network

The Lot49 proposal uses the eltoo scheme to update channel states. There is no revocation phase in this update protocol which reduces the amount of data transmitted. It also means that nodes that are part of the same *transaction chain* can cooperatively sign a single combined transaction to update their respective channel states. In our example the *transaction chain* includes the transactions that update the three channel states between the message sender and destination node. If an out-of-date update is committed by one of the nodes, it will not result in a penalty and can be replaced by any of the nodes by committing a more recent update within the timelock period.

update (update_add_htlc & commitment_signed)

	Size Schnorr (bytes)	Size BLS (bytes)	Note
type	1	1	2 (Negotiate_1)
channel_id	-	-	inferred from destination node
id	-	-	ID of the sender is included in the normal mesh routing header
prepaid_tokens (amount_msat)	1	1	sat amount committed by message sender
payment_hash	-	-	inferred from message data
cltv_expiry	-	-	use standard value
relay_path (onion_routing_packet)	12	12	up to 6 clear text hops, one 2 byte node ID per hop; also used to reconstruct the complete transaction chain
agg_signature	64	48	commit to new transaction chain
Total	78	62	

All of the nodes along the relay path collaboratively sign a single transaction (with Schnorr) or set of transactions (BLS). This involves each node modifying the *update* message to aggregate their signature with the agg_signature field data they received before retransmitting it to the next relay node. To update all three channels requires transmitting the *update* message three times.

A Schnorr multisignature or BLS aggregate signature is used to reduce the total amount of data transmitted between nodes. If using Schnorr signatures, each node aggregates their signature

to commit to spend from a single *setup* transaction that must be signed by all of the involved relay nodes. If using BLS signatures, each node aggregates their signature to commit to spend from a *setup* transaction established with the next relay node along the route.

For each update, nodes do not need to revoke the previous channel state. This reduces the amount of signature information that must be transmitted. If at any point a relay becomes unresponsive, a single transaction signed with the last complete multisignature (Schnorr) or aggregate signature (BLS) can be committed to settle all of the channels involved.

Message Type	Size Schnorr (bytes)	Size BLS (bytes)	Total data transmitted by both sender and receiver (bytes)	
			Update three channels	
			Schnorr	BLS
update	78	62	234	189
Total:			234	189

Lot49 Channel Setup / Close

The setup phase for a new payment channel starts with a nearby node proposing a transaction to refund the setup transaction. The receiving node that is funding the channel can hold that transaction until they need it because they have the 2nd signature.

Lot49 Channel Setup / Close

Setup / close (closing_signed)

	Size Schnorr (bytes)	Size BLS (bytes)	Note
type	1	1	1 (Setup_2) or 7 (Close_1)
channel_id	-	-	inferred from destination node
id	-	-	ID of the sender is included in the normal mesh routing header
prepaid_tokens (amount_msat)	1	1	msat committed by message sender

payment_hash	-	-	inferred from message data
cltv_expiry	-	-	use standard value
relay_path (onion_routing_packet)	12	12	up to 6 clear text hops, one 2 byte node ID per hop; also used to reconstruct the complete transaction chain
agg_signature	64	48	commit to new transaction chain; 96 bytes for Schnorr if MuSig signing used
Total	79	62	

Summary for Lot49 Setup and Close

Message Type	Size Schnorr (bytes)	Size BLS (bytes)	Total data transmitted by both sender and receiver (bytes)			
			Setup / Close three channels		Data transmitted to a gateway	
			Schnorr	BLS	Schnorr	BLS
Setup / Close	79	62	474	372	234	62
Total:			474	372	234	62

Conclusion

For an example three hop message delivery, the eltoo revocation system would require transmitting 1/5 as much data to update channels (234 vs 1221 bytes) compared to the current update mechanism used by the Lightning Network. BLS signature aggregation could reduce the amount of data that must be transmitted to an internet gateway to setup and close channels by 2/3 of that required when each channel is independently setup or closed (62 vs 234 bytes). The reduction in transmission overhead from using eltoo and signature aggregation increases linearly for longer relay paths.