# LRFP : Extending local routing protocols in layer 2 networks with a secure fee model

**Oliver Neut, Stefanie Roos, Oghuzan Ersoy**

Delft University of Technology

**Abstract**

Blockchains like Bitcoin are known to be victim of scalability issues. The lack in high throughput and low latency form a great bottleneck to its network. A promising solution are layer 2 protocols, more precisely payment channel networks (PCN). Payment success rates are a common metric in these networks. These rates can be increased by tweaking the routing of payments in the network. Local routing is a form of routing that allows payments in such networks to be split over multiple paths to reach its receiver. This significantly increases the rate of payment successes, however there is no trivial way to integrate fees in such protocol. This paper focuses on the integration of fees in local routing protocols by proposing a viable solution. Local Routing Fee Protocol (LRFP) is a protocol designed to extend an existing local routing protocol and is proven to be secure. It is a light addition but works as intended. Proofs on security guarantees and a formal description on the protocol form the main contribution of this paper.

## 1 Introduction

Blockchain is a novel technology that has gained increased popularity in the last decade. It's most known for the ability to enable trusted collaboration between untrusted parties [5]. The most popular example of a technology that uses blockchain is Bitcoin [6]. A digital currency with no central entity controlling the distribution of money or censoring transactions [5]. Blockchain can be used in various financial services such as online payments and digital assets, and even non-financial services e.g. smart contracts [7]. Although the technology has great potential for the construction of future Internet systems, it lacks in scalability. Bitcoin's network takes 10 minutes or longer to confirm transactions and its throughput achieves a maximum of 7 transactions per second [8] [2]. Ethereum, which is the second largest cryptocurrency in the world in terms of market capitalization, achieves roughly double that amount of transactions per second. In comparison, global payment networks such as Visa or other centralized payment service providers confirm a transaction within seconds and process thousands of transactions per second [5]. In order for Bitcoin to succeed in its vision as a global payment system, it needs to improve on both latency and throughput. It should also be noted that an average transaction fee in Bitcoin's network is usually over 1 USD [1].

A proposed solution to the bottleneck formed by the blockchain mechanism are layer-two protocols. This layer exists on top of layer one, the blockchain [5]. One of which

are off-chain payment channels and they form a great solution to the problems described above. By constructing payment channels between nodes and locking collateral in an opening transaction (which is broadcasted to the underlying blockchain), parties can send funds to each other. A payment channel consists of 2 parties, each party has some coins available for sending and receiving coins. Informally, when Alice pays Bob 2 coins, the balances of the channel that Alice and Bob share get updated. Transactions are reflected in the balance of the corresponding channel instead of the blockchain. The only operations that involve blockchain are the opening and closing of a channel. This takes a significant load of the blockchain since transactions can be performed off-chain [1].

When multiple nodes participate in payment channels, they form a payment channel network (PCN). A popular example of a PCN is Lightning, the layer 2 protocol accompanying Bitcoin. It consists of more than 21 thousand nodes and more than 50 thousand channels at the time of writing [9], [3]. Such a network allows nodes to send a payment to a receiver over the network by paying fees to incentivize intermediaries. This protocol (Lightning) uses source routing to route its payments. The sender determines a path based on the topology of the network and on the initial capacities of the channels. This is a key reason for high failure rates in the Lightning network, according to Roos et al [4]. To combat this, protocols have been developed that use local routing. In this type of routing, the sender only decides on the first hop of the payment and each subsequent node decides on where the payment goes next [4]. Additionally splitting payments across multiple nodes is possible. It allows an intermediary to forward a payment even if he shares no channel with sufficient capacity for the entire payment value. Subsequently, gaining flexibility in the network and increasing the probability of payment success [4].

While a PCN that uses local routing has higher success rate, it does introduce new problems into the protocol such as : how can fees be integrated while maintaining the corresponding security properties? The integration of fees isn't a trivial problem to solve. The Lightning method of integrating fees isn't compatible with local routing, this will be explained further on in the paper. The solution for this problem should ensure that the security properties are satisfied. Because the network is designed to perform monetary transactions, the security is of great importance. With the help of cryptographic proofs, these security guarantees can be proven to be correct. The main contribution of this work is a design on how fees can be integrated in a payment splitting protocol followed by proofs on the security guarantees.

Section 2 outlines the background extensively, which is needed for understanding the protocol. The solution of the fee model protocol is described in section 3, after which the proofs of the protocols are presented in section 4. Section 5 contains the responsible research. In section 6 the results are summarized. Lastly, section 7 concludes the research question and provides a small discussion on future work.

# 2 Background

## 2.1 Payment channel networks

Payment channels provide a way for 2 parties in a network to exchange funds with each other. A channel is established using a MULTISIG Bitcoin transaction, that requires signatures from multiple parties in order to be spent [5]. Two nodes start by signing a MULTISIG transaction and deposit funds. This transaction needs to be confirmed by the blockchain first, after this they can start sending and receiving coins. To keep track of the balances

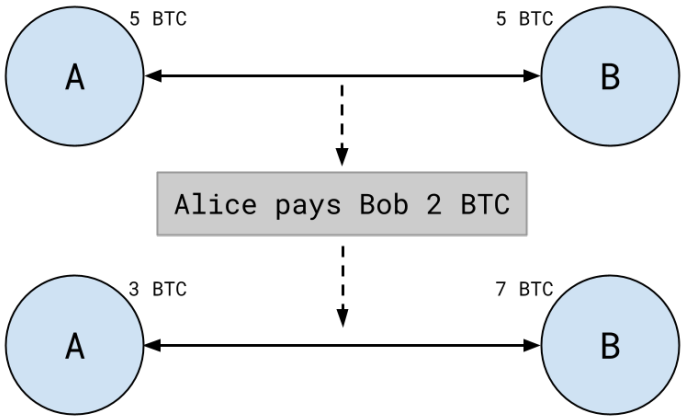


Figure 1: Payment channel transaction between Alice & Bob

in the channel, payments are done with "commitment transactions" [12]. These types of transactions do not require interaction with the blockchain. For a commitment transaction to take place, both parties need to sign the transaction and a timelock is added to make sure that the transaction cannot be redeemed before the timelock expires [5]. Later transactions in a channel have shorter timelocks to ensure that only the latest commitment transaction can be sumbitted on the blockchain. When 2 parties decide on closing the channel, they broadcast an on-chain transaction where they collect the final balances of the channel [5].

Take Alice (A) and Bob (B) as an example of 2 parties that decide on opening a channel, they both deposit some BTC. The initial balance is denoted by the capacity function $\mathcal{C}(A, B) = 5$, it represents the amount of BTC (Bitcoin) A has in the channel with B and $\mathcal{C}(B, A) = 5$ represents the amount of BTC B has in the channel with A. When Alice wants to pay Bob 2 BTC, the corresponding balances get updated: Alice's balance decreases by 2 BTC while Bob's balance increases by 2 BTC. Figure 1 describes this transaction. In the end Alice is owed 3 BTC and Bob is owed 7 BTC.

A network of these channels allows parties to conduct payments across multiple intermediaries to a receiver of choice. The routing of a payment can be done in multiple ways. In the Lightning network [3], this task is done by the sender in a pre-routing step. The sender knows the topology of the network and the total capacities of the channels. Because only the total amount of balances in a channel is known, there is no way to know for sure that a channel is able to forward a payment. For example if Alice has a balance of 1 BTC and Bob has 4 BTC, then Alice cannot forward a payment of more than 1 BTC through that channel. The routing algorithms are therefore optimized to compute a path to the receiver that is short and likely to succeed. This approach of sending payments has a relatively high failure rate.

To incentivize intermediaries to forward payments, an additional fee is added by the sender to tip the intermediaries. To compute the fee, Lightning let's each node in the network decide on their personal base fee and a proportional fee rate [13] [10]. The base fee is a flat fee that is always charged, regardless of the value the node is forwarding. While the fee rate is a fee charged for every satoshi (100 millionth of a Bitcoin) send through the

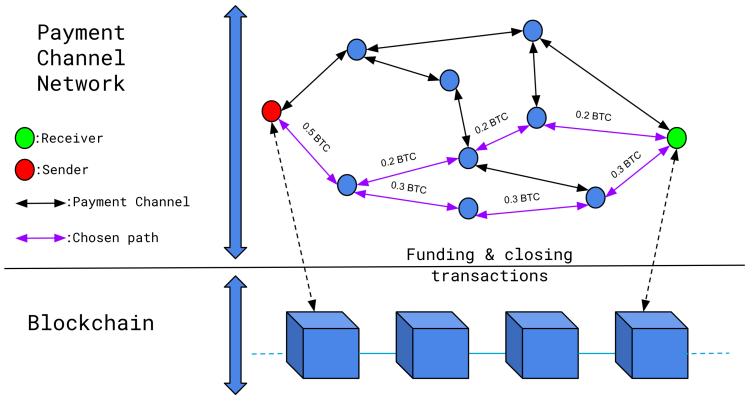


Figure 2: Payment channel network model

channel[9]. So based on the path, the source determines the size of the fee and adds this to the payment value.

A different method of routing payments is local routing, it allows each node on the path to freely choose the next hop based on the their local view on channel capacities. An additional feature of this kind of routing is that nodes are able to split a payment over multiple channels. Figure 2 depicts a PCN that uses payment splitting. This protocol is described in the work of Roos et al. [4]. Splitting payments has a positive effect on the probability of a payment succeeding, because the values of payments are smaller, the probability of a channel having insufficient funds to forward a payment is lower. However, it is this kind of routing where fees are still an undiscovered subject. This is simply because the sender has no clue on how large the fee should be, since there is no pre-routing step involved in local routing.

## 2.2 Interdimensional SpeedyMurmurs

In order to design a fee model, a good understanding of the payment splitting protocol is needed. The following subsections describe the local routing protocol designed by Roos et al[4].

### 2.2.1 Preliminaries

In this section, the basic notation and cryptographic primitives are explained. This will help with better understanding the protocol and its security properties.

*Basic notation* : the sets $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ denote the natural, integers and real numbers respectively. We denote the uniform sampling of a variable $x$ from a set $\mathcal{X}$ as $x \leftarrow_{\$} \mathcal{X}$. The security parameter is $n$ and all algorithms described in this paper run in polynomial time in $n$. A *probabilistic polynomial time* algorithm A (ppt) on input y, outputs x is denoted by $x \leftarrow A(y)$. $x := A(y)$ means that A is a deterministic. We denote an adversary with $\mathcal{A}$ and in this protocol he is able to corrupt a set of parties in the network. To express that a message m is send to a party P we denote : $m \hookrightarrow P$. Lastly, to express that a message m is received from party P we denote : $m \hookleftarrow P$.

*Cryptographic primitives* : A public key encryption scheme $\Psi$ is a scheme that has 3 ppt algorithms : generate key pairs ($Gen$), encrypt messages using a public key ($Enc_{pk}$), decrypt

messages using a secret key ($Enc_{sk}$). It also has 2 spaces, a message space $\mathbb{M}$, a cipher text space $\mathbb{C}$. The encryption schemes used in this protocol are defined to be indistinguisable under chosen plaintext attack (*IND-CPA*). This means that an adversary cannot distinguish the encryption of 2 messages of his choice [11]. A digital signature scheme $\Sigma$ is a scheme that has 3 ppt algorithms : generate key pairs (Gen), verify a signed message using a public key ($Vrfy_{pk}$), sign a message using a secret key ($Sign_{sk}$). The signature schemes in this protocol are defined to be existentially unforgeable under chosen message attack (*EUF-CMA*). This ensures that an adversary, while he can learn polynomially many signatures of messages, he cannot produce a signature to a message of his choice [11]. A hash algorithm $\mathcal{H} : \mathbb{P} \to \mathbb{H}$ is an algorithm that takes an arbitrary length bit strings as input and produces a fixed-length bit string as output : the hash value. This hash algorithm is preimage-resistant if it is polynomial-time computable and for every ppt adversary $\mathcal{A}$, given $y = \mathcal{H}(x)$, for a randomly sampled $x \in \mathbb{P}$, the probability that $\mathcal{A}$ outputs $x' \in \mathbb{P}$ s.t. $y = \mathcal{H}(x')$ is negligible [11].

### 2.2.2 Formal description : Interdimensional SpeedyMurmurs

The protocol consists of a network of nodes, we denote this formally by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. With $\mathcal{V}$ representing the set of parties in the network and $\mathcal{E}$ representing the payment channels in the network between the parties. Note that this graph is connected and directed. Along with $\mathcal{G}$ comes a capacity function $\mathcal{C} : \mathcal{E} \to \mathbb{R}^+$. A payment channel between 2 parties P and Q is denoted by 2 directed edges such that : $(P, Q) \in \mathcal{E} \Leftrightarrow (Q, P) \in \mathcal{E}$. This way the capacity function is able to distinguish the balance of each party in the channel. $\mathcal{C}(P, Q)$ is the amount of coins P has in the channel with Q and vice versa.

To simplify the mechanics of the payment channels, an ideal functionality is used that acts as a black box where the parties of the network can interact with. The functionality of payment channels in this protocol is denoted using $\mathcal{F}(\mathcal{G}, \mathcal{C}_0, \Delta)$. Where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the connected directed graph of the network, $\mathcal{C}_0$ is the initial capacity function and $\Delta$ is the upper bound on the blockchain delay. Each party $P \in \mathcal{V}$ can instruct the functionality with a message "pay" to send $v$ coins from P to Q. The message "cPay" is used for sending conditional payments. A conditional payment is used in multi-hop payments, where intermediaries need to forward a transaction. Such payments require more parameters such as the condition $\varphi : \{0,1\}^* \to \{0,1\}$ on which the payment occurs, the timelock $T \in \mathbb{N}$ and auxiliary information in the form $info \in \{0,1\}^*$. If $\mathcal{C}(P, Q) \geq v$ then the functionality deducts $v$ coins from channel $(P, Q)$ and informs Q about the conditional payment. If Q wants to unlock a conditional payment, he needs to send "cPay-unlock" to $\mathcal{F}$ and provide the witness $w$ such that the condition holds, $\varphi(w) = 1$. This witness is a preimage $x_r$ of a hash value $h_r$ such that the condition $\mathcal{H}(x_r) = h_r$ holds. The reason for using this, will be explained in section 2.2.3. After $T$ rounds, the sender P of the conditional payment can send "cPay-refund" and receive the $v$ coins back.

$\Theta : \{0,1\}^* \to \{0,1\}^*$ is a function that keeps track of all conditional payments currently being executed. On input of a payment identifier $pid \in \{0,1\}^*$, the function $\Theta$ either returns $\perp$, meaning that there currently is no payment with that identifier, or $(e, v, \varphi, T)$. The parameters stand for respectively : $e \in \mathcal{E}$ the channel on which the payment takes place, $v$ the amount of coins being transferred, $\varphi : \{0,1\}^* \to \{0,1\}$ the condition of the payment and $T$ the timelock. In this protocol we need to assume that a ppt adversary $\mathcal{A}$ exists. This adversary can take control over every node $P \in \mathcal{V}$. We define the set of parties that are not corrupted by $\mathcal{A}$ to be $Honest \subseteq \mathcal{V}$. In order for the sender to prove

that he has paid an amount to the sender, the sender provides a receipt. This receipt can be defined valid with the function Validate: $\mathcal{V} \times \mathcal{V} \times \mathbb{R}^+ \times \{0,1\}^* \rightarrow \{0,1\}$, that takes as input a sender $S$, a receiver $R$, an amount of coins $v$ and a receipt rec $\in \{0,1\}^*$ and outputs a 0 or 1 depending on the validity. Lastly we define $\mathcal{C}$ and $\mathcal{C}'$ to be the capacity function before and after a payment in the network. The difference of the sum of all capacities a party $P \in \mathcal{V}$ has in all its channels after and before a transaction is defined as $Net_{C,C'}(P) := \sum_{W \in \mathcal{V}:(P,W)\in\mathcal{E}} \mathcal{C}'(P,W) - \mathcal{C}(P,W)$.

### 2.2.3 Payment process

Once a sender and a receiver agree on sending a payment of value $v$, the receiver starts by sampling a preimage $x_r$ and sending its hash value $h_r = \mathcal{H}(x_r)$ to the sender. The sender calls the function $\text{Route}_{\mathcal{G}}$, that decides on how to route the first hop of the payment. Either it splits the amount $v$ or it doesn't. $\text{Route}_{\mathcal{G}}(v, P, R, excl, \mathcal{C}_P)$ takes as input : the amount of coins $v$ to be routed, the identifier of the part $P$ performing the routing, the identifier of the receiver $R$, the set $excl$ containing the nodes that have already been visited on the path between the sender $S$ and party $P$ and the capacity function of $P$, $\mathcal{C}_P$. The function then returns either $\perp$ signaling that the routing failed or $k$ edge/value pairs $\{(e_j, v_j)\}_{j\in[k]} \subseteq (\mathcal{E} \times \mathbb{R}^+)^k$ such that (i) $\mathcal{C}_P(e_j) \geq v_j$ for every $j \in [k]$ (ii) $e_j = (P, Q_j)$ such that $Q_j \notin excl$ and (iii) $\sum_{i\in[k]} v_j = v$.

From this point on, the sender can use $h_r$ to either send 1 conditional payment to the first hop or multiple conditional payments that add up to $v$ coins to multiple hops. A conditional payment can only be unlocked if an $x_r$ is provided by the receiver of that payment such that $\mathcal{H}(x_r) = h_r$ holds. The intermediaries use the same $h_r$ of the conditional payment to send one to the next hop, this process is repeated until enough conditional payments arrive at the receiver with a value of at least $v$ coins. Then the receiver uses the preimage $x_r$ as a witness to unlock the incoming conditional payment(s). The intermediaries one hop before the receiver receive the preimage $x_r$ and can use it to unlock their incoming conditional payment. This process is repeated until all conditional payments are unlocked. The reason for using a preimage-resistant hash function is to make sure that only the receiver (who sampled the preimage) can start the unlocking of conditional payments. He will only do this once he received $x$ amount of conditional payments that add up to at least $v$ coins.

## 2.3 Security

An important part of this protocol is that transactions can occur in a secure manner. A few security properties are designed to assure the security of such networks. ***Termination*** assures that all honest parties produce an output in finitely many rounds. All nodes except the sender produce an output of the form $\top$ to signal the end of their cooperation. The sender outputs the receipt if the payment succeeded or $\top$ if a non valid signature was provided. The next property, ***bounded loss for the sender***, ensures that the sender never loses more coins than the value of the payment. ***Balance neutrality*** means that no intermediary or receiver ever loses coins in a transaction. An important property is ***atomicity***: it states that either a transaction occurs or it didn't happen, the sender can prove this by showing a receipt. If the sender is able to provide a valid receipt, it means that the receiver has at least $v$ coins (the payment amount) added to his balance. If a sender lost $v$ coins, it means that he has a valid receipt signed by the receiver that he paid $v$ coins. Lastly ***correctness*** ensures that if the capacity of all channels is at least v and all parties are honest, then the payment completes successfully. The integration of fees in the protocol

change the security properties, therefore the proofs need to be modified accordingly. The original security properties can be found in the work of Roos et al [4].

### 2.4 Fee motivation

Fees in payment channel networks are typically significantly lower than in blockchain transactions. However they play an important role in the network, reasons for this are the following. Fees act as a incentivization mechanism in payment channel networks. When there isn't a benefit to routing a payment to the next hop, why would a node participate in this procedure? Fees are the solution for the incentivization of intermediaries.

When an intermediary gets a conditional payment that should be forwarded, its funds are locked for a certain time window in the channel it shares with the next hop. During this, the channel can be seen as illiquid. Therefore it makes sense to provide value to such intermediaries when they help conduct a payment. As a result, the total balance of a node in all its channels will increase.

### 2.5 Problem description

The problem which is ought the be tackled in this paper revolves around the fees in a PCN. More precisely a PCN that uses local routing and payment splitting. A formal description of a fee integration in such networks doesn't exist. This is not a trivial task, because the fee model that Lightning uses does not work in a protocol that uses local routing. The lack of a pre-routing step in Interdimensional SpeedyMurmurs makes it difficult for the sender to decide on a reasonable fee. Since the sender does not know in advance how many nodes are forwarding parts of the payment, there is no way to guess the combined value of all the base fees and fee rates. A formal fee model that extends the Interdimensional SpeedyMurmurs protocol is needed that solves this problem. Part of creating a fee extension to the payment splitting protocol is proving that the protocol is secure. The security properties will need small alterations to make them applicable to the new protocol. Subsequently, the proofs of these properties will need a revision as well.

## 3 Local Routing Fee Protocol

Now that the basics of payment channel networks are clear, we can move on to the description of the developed fee model. The solution forms an extension to the payment splitting protocol provided by Roos et al [4]. A fitting name for this fee protocol was chosen to be : LRFP, an acronym for **local routing fee protocol**.

### 3.1 Protocol design

The protocol is designed to be used in a payment channel network that uses local routing and payment splitting. We let the intermediaries in the PCN decide on their own base fee, denoted by $r_I$. Simply because, if we require everyone to charge the same base fee, there is no easy way to know if someone isn't cheating and charging more. The idea is that the sender can determine the fee as the maximum value he's willing to pay for a transaction. We denote this value by $fee_{max}$. This decision of the $fee_{max}$ has 2 outcomes: (i) either the value of the fee attached is too low, or (ii) it is (just) enough. In the first case (i), intermediaries

subtract their base fee $r_I$ from the $fee_{max}$ until a next hop (we call him Q), realizes there is no fee left to profit from, because $fee_{atQ} < r_Q$ ($fee_{atQ}$ is the fee remaining once it arrives at Q). Thus Q aborts the payment and all parties before him are refunded. The sender is able to retry with a slightly higher fee. In the latter case (ii), the fee provided by the sender is enough, the payment succeeds and if the last intermediary before the receiver has some leftover fee, he can choose to keep it or send it over to the receiver. In order for the sender to determine a reasonable fee, network statistics of fees and corresponding payment values can be used as guidance.

## 3.2 Generic description of LRFP

By extending the protocol with a fee model, the inner workings of the protocol don't change significantly[1]. A few modifications are made in the Route$_{\mathcal{G}}$ algorithm : it takes an extra fee value $f$ parameter and outputs k tuples of the form $\{(e_j, v_j, f_j)\}_{j\in[k]} \subseteq (\mathcal{E} \times \mathbb{R}^+ \times \mathbb{R}^+)^k$. For which the following conditions hold : (i) $\mathcal{C}_P(e_j) \geq v_j + f_j$ for every $j \in [k]$ (ii) $e_j = (P, Q_j)$ such that $Q_j \notin excl$ and (iii) $\sum_{i\in[k]} v_j = v$ , $\sum_{i\in[k]} f_j = f$. When this function is called, it decides on whether to split the payment across multiple nodes or 1 node. If multiple nodes are chosen, the fee is split proportionally over the different nodes. [2] If 1 node suffices in capacity to transfer the coins and fee, the output fee remains the same. The altered Route$_{\mathcal{G}}$ algorithm is depicted in figure 3. The next modification was made in messages to the payment channel functionality $\mathcal{F}$. Specifically the 'cPay' message, in which a party P (sender or intermediary) sends a conditional payment to a next hop Q. An extra parameter is added, the value of the fee $f$. $\mathcal{F}$ takes this value and subtracts $(v + f)$ from P's channel. Once this message is unlocked with 'cPay-unlock' by Q, the same value $(v + f)$ is added to Q's channel. This functionality is depicted in figure 5. Lastly, an intermediary $I$ chooses his own base fee $r_I$ and has to check a few conditions when forwarding a payment. The base fee $r_I \in \mathbb{R}^+$ is the fee value that $I$ charges when asked to forward a payment. He subtracts the incoming fee $f$ with his base fee $r_I$ and sends the resulting value $f' = f - r_I$ to the next hop(s). Note that if $f' < 0$ or if he received a 'cPaid' message with $f = 0$, $I$ aborts [3]. The pseudocode of the algorithm can be found in figure 4. The highlighted parts in figure 3, 4 and 5 resemble the parts that are altered for the fee model integration.

Route$_{\mathcal{G}}(v, P, R, excl, \mathcal{C}_P, f)$

// Get candidate set

$\{((P, U_j), c_j, d_j)\}_{j\in[k']} \leftarrow$Closer$(P, R, \mathcal{C}_P)$

// Remove visited nodes

$\mathbf{M} \leftarrow \emptyset$

**for** $j \in [k']$ **do**

  **if** $(U_j \notin excl)$ **then**

    $\mathbf{M} := \mathbf{M} \cup \{((P, U_j), c_j, d_j)\}_{j\in[k']}$

// Split

$\{(\tilde{e}_j, \tilde{v}_j)\}_{j\in[k']} \leftarrow$Split$(\mathbf{M}, v)$

**return** $\{(\tilde{e}_j, \tilde{v}_j, \frac{\tilde{v}_j}{v} \cdot f)\}_{j\in[k]}$

Figure 3: Altered Route$_{\mathcal{G}}$ algorithm of LRFP

[1]For a complete generic description of the IntSM protocol, consult the paper by Roos et al[4].

[2]E.g. take $f = 1$ and $v = 5$ and $v$ coins are split over 2 nodes A and B, where A receives $v_A = 1$ coin and B $v_A = 4$ coins, then A receives $f_A = 0.2$ fee and B $f_B = 0.8$ fee accordingly.

[3]In both cases the fee provided by the sender S is too small

Sender $S(\mathcal{G}, \mathcal{C}_S, S, R, v, f_{max})$

$out := \emptyset$, rec:= $\top$

In round $t_0 + 1$

**if** $(init, h_R, \sigma) \hookleftarrow R \wedge Vrfy_{pk_R}((S, R, v, h_R), \sigma)$ **then**
  // Split and send payments
  $T := t_0 + 1 + |\mathcal{V}| \cdot (1 + 2 \cdot (\Delta + 1))$
  $excl := S$
  $(e_j, v_j, f_j)_{j \in [k]} \leftarrow Route_{\mathcal{G}}(v, S, R, excl, \mathcal{C}_S, f_{max})$
  $(h_j, c_j, x_j)_{j \in [k]} \leftarrow HLocks(h_R, Enc_{pk_R}(0), k, pk_R)$
  **for** $j \in [k]$ **do**
    $pid_j \leftarrow_\$ \{0,1\}^*$
    $(cPay, pid_j, e_j, v_j, f_j, h_j, T, (c_j, R, excl)) \hookrightarrow \mathcal{F}$
    $out := out \cup (pid_j, x_j)$
**else** TerminateS()

Intermediary $I(\mathcal{G}, \mathcal{C}_I, r_I)$

$fw := \emptyset$

(cPaid, $pid, e, v', f, h, T, (c, R, excl)) \hookleftarrow \mathcal{F}$

**if** now $> t_0 + |\mathcal{V}| \vee f = 0$ **then**
  **abort** // too late to route
**else** // Split and forward payment
  $T' := T - 2(\Delta + 1)$
  $excl := excl \cup I$
  $f' := f - r_I$ // decrease $f$ with $r_I$
  **if** $f' < 0$ **then abort** // fee is too small
  $(e_j, v_j, f_j)_{j \in [k]} \leftarrow Route_{\mathcal{G}}(v', I, R, excl, \mathcal{C}_I, f')$
  $(h_j, c_j, x_j)_{j \in [k]} \leftarrow HLocks(h, c, k, pk_R)$
  **for** $j \in [k]$ **do**
    $pid_j \leftarrow_\$ \{0,1\}^*$
    $(cPay, pid_j, e_j, v_j, f_j, h_j, T', (c_j, R, excl)) \hookrightarrow \mathcal{F}$
    $fw[T'] := fw[T'] \cup (pid_j, pid, x_j)$

Figure 4: Generic description of LFRP

**Payment channel functionality $\mathcal{F}(\mathcal{G}, \mathcal{C}_0, \Delta)$**

The initial state is set to $\mathcal{C} := \mathcal{C}_0$ and $\Theta(pid) := \perp$ for all $pid \in \{0,1\}^*$.

- Upon receiving (cPay, $pid, e, v, f, \varphi, T, (c, R, excl)) \hookleftarrow P$, where $e = (P, Q) \in \mathcal{E}$, $\mathcal{C}(e) \geq (v + f)$ and $\Theta(pid) = \perp$, wait for one round to set $\mathcal{C}(e) := \mathcal{C}(e) - (v + f)$, store $\Theta(pid) := (e, v, f, \varphi, T)$ and send (cPaid, $pid, e, v, f, \varphi, T, (c, R, excl)) \hookrightarrow Q$.
- Upon receiving (cPay-unlock, $pid, w) \hookleftarrow Q$, wait for at most $\Delta + 1$ rounds. If $((P, Q), v, f, \varphi, T) := \Theta(pid) \neq \perp$ and $\varphi(w) = 1$, then set $\mathcal{C}(e') := \mathcal{C}(e') + (v + f)$, for $e' = (Q, P)$, set $\Theta(pid) = \perp$ and send (cPay-unlocked, $pid, w) \hookrightarrow P$.
- Upon receiving (cPay-refund, $pid) \hookleftarrow P$, wait for at most $\Delta + 1$ rounds. If $(e, v, f, \varphi, T) := \Theta(pid) \neq \perp$ and the current round number is larger than $T$, then set $\mathcal{C}(e) := \mathcal{C}(e) + (v + f)$, $\Theta(pid) = \perp$ and send (cPay-refunded, $pid) \hookrightarrow P$.

Figure 5: Payment channel functionality $\mathcal{F}$ of LRFP

# 4 Security proofs

The following section contains the altered security properties and its proofs. The changed parts are highlighted in the properties below. The symbol Π denotes **LRFP** (Local Routing Fee Protocol).

**The adapted security properties can be defined formally :**

- **Balance neutrality** :
  - $R \in Honest \Rightarrow net_{C,C'}(R) \geq 0$ (for the receiver)
  - The payment succeeds $\wedge$ $P \in Honest \backslash \{S, R\} \Rightarrow net_{C,C'}(P) \geq r_P$
- **Bounded loss for sender** : $S \in Honest \Rightarrow net_{C,C'}(S) \geq -(v + f_{max})$
- **Atomicity** :
  - $S \in Honest \wedge net_{C,C'}(S) < 0 \Rightarrow Validate(S, R, v, rec) = 1$
  - $R \in Honest \wedge Validate(S, R, v, rec) = 1 \Rightarrow net_{C,C'}(R) \geq v$
- **Correctness** : *If* $\mathcal{V}$ = Honest *and for every* $e \in \mathcal{E}$ *it holds that* $\mathcal{C}(e) \geq v + f$ *and* $\sum_{I \in \mathcal{V} \backslash \{S,R\}} r_I \leq f_{max}$*, then it holds that* $net_{C,C'}(S) = -(v + f_{max})$*,* $net_{C,C'}(R) \geq v$ *and* $net_{C,C'}(I) \geq r_I$ *for all* $I \in \mathcal{V} \backslash \{S, R\}$.

## 4.1 Termination

Because the changes made in the protocol pseudocode do not modify any timelock or affect the behavior of the timelocks, the termination property of the extended protocol doesn't change. Therefore its proof remains unaltered. The proof of Termination can be read in [4].

**Claim 1** *The protocol* Π *terminates in finitely many rounds.*

## 4.2 Balance neutrality

Balance neutrality claims that the monetary loss of an honest sender is bounded by the amount of coins that he wants to send, and none of the other honest parties can ever lose coins. To this end, we make the following simple but important observation about the payment channel ideal functionality $\mathcal{F} := \mathcal{F}(\mathcal{G}, \mathcal{C}, \Delta)$ that parties call in the protocol execution.

Observation 1. *The ideal functionality* $\mathcal{F}$ *never reduces coins of any party unless it receives the instruction 'cPay' or 'pay' from this party.*

**Claim 2** (Bounded loss for the sender). *It holds that* :

$$S \in Honest \Rightarrow net_{C,C'}(S) \geq -(v + f_{max})$$

*Proof.* The proof states that an honest sender S never sends a 'pay' message. Only if the signature from the receiver is valid, S sends $k$ cPay-messages of values $(v_1, ..., v_k)$ and corresponding fees $(f_1, ..., f_k)$. Since the values $(v_1, ..., v_k)$ and $(f_1, ..., f_k)$ are returned by the routing algorithm $\text{Route}_{\mathcal{G}}$, we know that $\sum_i v_i = v$ and $\sum_i f_i = f_{max}$. Hence, the total value of all the cPay-messages sent by $S$ to $\mathcal{F}$ is at most $v + f_{max}$ which by Observation 1 implies that $net_{\mathcal{C},\mathcal{C}'}(S) \geq -(v + f_{max})$. □

The following proves the balance neutrality for the intermediaries and the receiver. It is trivial for the receiver $R$, since $R$ never sends a 'pay' or 'cPay' message. For an honest intermediary $I$, it needs to be shown that $I$ never sends more than he conditionally receives and that $I$ has the ability to unlock an incoming conditional payment. Which requires $I$ : (i) to have the revealed witness $x_r$ from an outgoing payment and (ii) enough time to unlock the incoming payment. In the basic protocol, proving (i) is trivial and in the extended protocol it follows from the homomorphic property of $\mathcal{H}$. The following states an auxiliary lemma about additively homomorphic functions.

Lemma I.1. *Let $\mathcal{H} : \mathbb{P} \to \mathbb{H}$ be an additively homomorphic function. Let $h \in \mathbb{H}$, $x_i \in \mathbb{P}$ and let us define $h_i := h + \mathcal{H}(x_i)$. Then for every $x' \in \mathbb{P}$ s.t. $\mathcal{H}(x') = h_i$, it holds that $\mathcal{H}(x' - x_i) = h$.*

*Proof.* By definition of $h_i$, we know that $h = h_i - \mathcal{H}(x_i)$, hence

$$h = h_i - \mathcal{H}(x_i) = \mathcal{H}(x') - \mathcal{H}(x_i) = \mathcal{H}(x' - x_i)$$

□

**Claim 3** (Balance neutrality for the receiver) . *It holds that* :

$$R \in Honest \Rightarrow net_{C,C'}(R) \geq 0$$

*Proof.* Since an honest receiver R never sends any pay-message or cPay-message to $\mathcal{F}$, by Observation 1, $net_{\mathcal{C},\mathcal{C}'}(R) \geq 0$. □

**Claim 4** (Balance neutrality for the intermediaries) . *It holds that* :

The payment succeeds $\wedge$ $P \in Honest \backslash \{S, R\} \Rightarrow net_{C,C'}(P) \geq r_P$

*Proof.* An honest intermediary $I$ never sends 'pay' and never sends 'cPay' without receiving a 'cPay' message. In other words, every outgoing conditional payment is triggered by an incoming conditional payment.

Assume now that $I$ receives a conditional payment of value $v$, condition $h$, fee $f$ and time-lock $T$. The fee $f$ is subtracted with the base fee $r_I$ of I, $f' = f - r_I$. Then $I$ executes the routing algorithm $\text{Route}_{\mathcal{G}}$ on input value $v$ and $f'$ and obtains $k$ values $(v_1, ..., v_k)$ and $(f_1, ..., f_k)$ such that $\sum_i v_i = v$ and $\sum_i f_i = f'$. Thereafter, the intermediary executes the algorithm HLocks on input $h$ and $k$ and obtains $k$ hash values $(h_1, ..., h_k)$. For every $i \in [k]$, the intermediary sends a cPay-message of value $v_i$, fee $f_i$, condition $h_i$ and time-lock $T' := T - 2 * (\Delta + 1)$. This implies that an honest $I$ never conditionally pays more coins than what he can conditionally receive minus his base fee $r_I$. As a next step, we prove that if one of the outgoing payments is completed, i.e., $I$ pays ($v_i$ coins, then $I$ has the guarantee of unlocking the corresponding incoming payments, i.e. receive $v$ coins.

First we show that if $I$ learns a preimage of at least one of the hash values $(h_1, ..., h_k)$, then he can compute a preimage for $h$.
$\underline{\Pi = \Pi_b}$: Since for every $i \in [k]$ we have $h_i = h$, the statement trivially holds.
$\underline{\Pi = \Pi_{ext}}$: For every $i \in [k]$, the value $h_i$ is computed as $h + \mathcal{H}(x_i)$. Upon learning $x'$, such that $h_i = \mathcal{H}(x')$, $I$ computes $x' - x_i$ which by Lemma I.1 is a preimage of $h$.
The proof further shows that the intermediary $I$ has enough time to learn the preimage and submit it to unlock the incoming payment. This part isn't affected by the fee model. □

### 4.3 Atomicity

The atomicity property isn't affected by the integration of fees. This is because it simply states that a sender S has a valid receipt if he lost coins and that a receiver has at least $v$ coins if the sender has a valid receipt of sending $v$ coins to R. The integration of a fee model doesn't change the procedure of providing receipts, therefore its proof remains the same. The proof of Atomicity can be read in [4].

**Claim 5** (Atomicity for the sender) :
$S \in Honest \wedge net_{C,C'}(S) < 0 \Rightarrow Validate_{\Sigma,\mathcal{H}}(S, R, v, rec) = 1$

**Claim 6** (Atomicity for the receiver) :
$R \in Honest \wedge Validate_{\Sigma,\mathcal{H}}(S, R, v, rec) = 1 \Rightarrow net_{C,C'}(R) \geq v$

### 4.4 Correctness

We need to prove that our protocol satisfies correctness meaning that if all parties are honest and all channels in the network have enough coins and all intermediaries combined charge no more fees than $f_{max}$, then the payment succeeds. The main steps of our proof are the following. Since both sender and receiver are honest, the sender initiates conditional payments of total value $v + f_{max}$. Then we show that the sender provides a sufficient time-lock for the partial payments to arrive at the receiver. We now state and proof the correctness of our protocol formally.

**Claim 7** (Correctness) :
*If* $\mathcal{V}$ = Honest *and for every* $e \in \mathcal{E}$ *it holds that* $\mathcal{C}(e) \geq v + f$ *and* $\sum_{I \in \mathcal{V} \setminus \{S,R\}} r_I \leq f_{max}$, *then it holds that* $net_{C,C'}(S) = -(v + f_{max})$, $net_{C,C'}(R) \geq v$ *and* $net_{C,C'}(I) \geq r_I$ *for all* $I \in \mathcal{V} \setminus \{S, R\}$.

*Proof.* Since the receiver $R$ is honest, the sender $S$ receives a valid signature $\sigma$ on the statement $(S, R, v, h_R)$ from the receiver $R$ in the round $t_0 + 1$. This means that the sender executes the algorithm $\text{Route}_{\mathcal{G}}$ on input $excl = \emptyset$. We know that the routing algorithm returns $\{(e_i, v_i, f_i)\}_{i \in [k]}$ and hence the sender initiates $k$ conditional payments - each of them with set $excl := \{S\}$. Assume for now that at least one of the conditional payments is completed and hence $net_{C,C'}(S) < 0$. By claim 5, this implies that the sender outputs a valid receipt. Since the receiver is honest and the sender outputs a valid receipt, by claim 6 we know that $net_{C,C'}(R) \geq v$. Moreover, by claim 4 we know that the intermediaries cannot lose coins and earn at least $r_I$ (given that the payment succeeds), i.e., $net_{C,C'}(I) \geq r_I$ for every $I \in \mathcal{V} \setminus \{S, R\}$. Hence we have

$$net_{C,C'}(S) \leq -\left( net_{C,C'}(R) + \sum_{I \in \mathcal{V} \setminus \{S,R\}} net_{C,C'}(I) \right) \leq -(v + f_{max})$$

The bounded loss for the sender, claim 2, guarantees that $net_{C,C'}(S) \geq -(v + f_{max})$, hence it must holds that $net_{C,C'}(S) = -(v + f_{max})$. This in turn implies that $net_{C,C'}(R) \geq v$ and it holds that $\sum_{I \in \mathcal{V} \setminus \{S,R\}} r_I \leq \sum_{I \in \mathcal{V} \setminus \{S,R\}} net_{C,C'}(I) \leq f_{max}$ and thus $net_{C,C'}(I) \geq r_I$.

The remaining part of the proof shows that at least one of the conditional payments made by the sender is unlocked before the protocol terminates. This part is not affected by the fee model. □

### 4.5 Unlinkability

On a high level, the unlinkability of the protocol ensures that an honest party (sender or intermediary) that sends a payment of $v$ coins into $k$ partial payments $(v_1, ..., v_k)$, which he routes over $k$ next hops $P_1, ...P_k$. Even if all of these neighbours collude, they cannot decide whether these conditional payments are from the same payment or if these originate from $k$ different payments [4]. This functionality is facilitated with the use of homomorphic addition of hashes, more precisely the hashes used for sending conditional payments. No changes are made that affect the functionality of unlinkability, so this property is unaltered in the protocol. Therefore the proof needs no changes and can be consulted in the paper of Roos et al [4].

## 5 Responsible Research

This section reflects on the ethical aspects and the reproducibility of the research done in this paper. Since the adoption of cryptocurrencies is gaining traction, the research in this field becomes of greater importance. It is therefore crucial to consider the consequences and implications of the enhancement of these technologies.

### 5.1 Reproducibility

In this paper a solution to the problem : "how can fees be integrated into a PCN that uses local routing such that the protocol still achieves corresponding security guarantees?" is proposed. The proposed protocol (LRFP) only serves as a proposed design and it is proven to be sound, there are no quantitative results derived from this model. Since this is largely a descriptive research paper, it can be defined as reproducible. Everyone is perfectly able to reproduce the solution that is found to the research question.

### 5.2 Ethical aspects

The pseudocode of the solution is in the paper, this can be used to implement the system in an existing protocol. This integration could essentially be used in a real working system with humans exchanging real funds e.g. Bitcoin. However, there are some possible consequences for the future of this research. If this proposed fee model contains mistakes in the pseudocode or in the proofs of the security guarantees. Someone that implements this protocol in an existing system might break the security or basic functionality of his protocol. This could lead to undesirable behaviour in the protocol and in the worst case, honest parties in such a network could lose funds.

Another consequence concerns the mass adoption of payment channel networks in the cryptocurrency space. Because layer 2 protocols are one of the most promising realisations to tackle the scalability problems of blockchains, the adoption of it has great potential. Contributing to layer 2 protocols consequently has real implications. Payment channel networks

drastically increase the amount of transactions while maintaining security aspects. Hence it opens up a lot of possibilities for blockchains in industries of scale, which will drive the adoption even more. A significant advantage of the use of layer 2, is the diminished interaction with the blockchain. Bitcoin tends to use enormous amounts of energy [14], even surpassing some countries in energy consumption. Reducing the load of the blockchain, by performing more transactions off-chain would have a positive effect on the energy consumption. Thus layer 2 solutions provide a way to maintain security, a higher throughput and would reduce the energy consumption of e.g. Bitcoin's blockchain [5] [14].

# 6 Discussion

The aim of this paper was to provide a solution to the fee problem in local routing protocols and prove the solution provided is a sound one. To evaluate this contribution, the benefits and limitations of the Local Routing Fee Protocol are covered in this section. Hereafter, alternatives to LRFP are proposed and described. LRFP is a protocol that extends the Interdimensional SpeedyMurmurs protocol with a fee model [4]. Because the procedure of the fee model is very simple, the integration of it requires very little modification to the existing protocol. This also makes it easy to understand and to integrate in other protocols that make use of local routing.

A major limitation of the protocol is that the sender never receives any of his $f_{max}$ back. This is because none of the intermediaries or sender send the remaining fee of $f_{max}$ back to the sender. Hence this would be similar as sending a payment from the receiver to the sender and would in turn also need fees. Since the fee values in a PCN are significantly smaller than average Bitcoin fees [9], anyone is still better of by choosing a PCN to send his payment.

Compared to Lightning, this protocol only allows intermediaries to charge a base fee $r_I$ and no fee rate. This fee rate would allow an intermediary $I$ to charge a fee for every satoshi passing through their channel. The protocol currently doesn't include this. However, this could be a simple addition to LRFP.

Another limitation of LRFP is that a sender cannot calculate the exact fee needed. Because of the local routing, conditional payments are made while routing. There is no easy way to solve this issue. Therefore the sender should advise himself on statistics of fees in the network [9].

Currently, in the protocol, the payment value $v$ and fee $f$ are distinguishable as 2 parameters in the messages to $\mathcal{F}$. These values can also be merged into 1 parameter, this has different consequences. First, if the $f_{max}$ provided was too low, the payment would always fail at the sender since there would be no way for the intermediaries to know if they are charging fees from the payment value $v$. Second, the exact payment value $v$ is not known to the intermediaries which is beneficial for privacy. Third, since the last intermediary before the sender doesn't know what $v$ is, he will not be able to claim all the remaining fee and the sender will end up receiving the surplus. This would improve on privacy, but knowing that the sender won't receive any of his $f_{max}$ back, these other changes are of no importance to the sender.

Protocols like Boomerang [15] provide a way for the sender to send more coins than $v$ (the payment amount) over the network. Once the receiver, receives at least $v$ coins, he can send the redundant amount back to the sender in a secure manner. Combine this with fees in a local routing protocol and this solves a major issue of this protocol. This requires more research and proving the soundness of this is likely more complex.

The Merchant [16] is a protocol that integrates a fee model designed to increase the long-term health of the network. It counteracts the depletion of channels by incentivizing rebalancing channels with the help of fees. This is not a solution to solving the fee problem in local routing, but it does provide a way to incentivize rebalancing. This isn't trivial to integrate in the current protocol and requires more research.

# 7 Conclusions and Future Work

The content of this paper contains a solution to the problem : "how can fees be integrated into a PCN that uses local routing such that the protocol still achieves corresponding security guarantees?". It is not evaluated on how well it performs nor is it an analysis on what its advantages or disadvantages are. It is merely a proposed method to integrate fees in a local routing protocol. That is proven to be sound and completes the task it needs to fulfil, namely provide intermediaries in a PCN with fees. Note that there are different solutions to this problem, that can be more complex or simpler. The solution that this paper proposes is one design that is proven to be a viable one. This system however is easy to comprehend and to integrate in an existing local routing protocol.

This work is a basis for fee models in local routing and payment splitting payment channel networks. This field is wide open for improvement. Some future work can include the integration of The Merchant [16] or to combine the protocol with Boomerang [15]. A simple upgrade can be made with the $v + f$ parameter, subsequently boosting the privacy of payments. An evaluation of this protocol in an existing network based on its fees is recommended. This would be useful to compare the performance of it to other future fee models.

# References


[1] Dziembowski, S., Kedzior, P. (2020). Non atomic payment splitting in channel networks. `https://eprint.iacr.org/2020/166.pdf`

[2] Visa. (2018). Visa Fact Sheet. `https://usa.visa.com/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf`

[3] Poon, J., Dryja, T. (2016). The Bitcoin Lightning Network: scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`

[4] Eckey, L., Hostáková, K., Faust, S., Roos, S. (2020). Splitting Payments Locally While Routing Interdimensionally. `https://eprint.iacr.org/2020/555.pdf`

[5] N. Papadis L. Tassiulas, "Blockchain-Based Payment Channel Networks: Challenges and Recent Advances," in IEEE Access, vol. 8, pp. 227596-227609, 2020, doi: 10.1109/ACCESS.2020.3046020.

[6] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), available at: `https://bitcoin.org/bitcoin.pdf`

[7] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557-564, doi: 10.1109/BigDataCongress.2017.85.

[8] Croman K. et al. (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. https://doi.org/10.1007/978-3-662-53357-4_8

[9] "Real-time Lightning Network statistics." [Online]. Available: https://1ml.com/statistics

[10] G. Di Stasi, S. Avallone, R. Canonico and G. Ventre, "Routing Payments on the Lightning Network," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 1161-1170, doi: 10.1109/Cybermatics$_2$018.2018.00209.

[11] Smart, N. P. (2016). Cryptography Made Simple (1st ed.). Springer Publishing Company, Incorporated.

[12] Gudgeon L., Moreno-Sanchez P., Roos S., McCorry P., Gervais A. (2020) SoK: Layer-Two Blockchain Protocols. In: Bonneau J., Heninger N. (eds) Financial Cryptography and Data Security. FC 2020. Lecture Notes in Computer Science, vol 12059. Springer, Cham. https://doi.org/10.1007/978-3-030-51280-4_12

[13] lightningnetwork. (2019). Basis of Lightning Technology. GitHub. https://github.com/lightningnetwork/lightning-rfc/blob/master/00-introduction.md

[14] Huynh, A.N.Q., Duong, D., Burggraf, T. et al. Energy Consumption and Bitcoin Market. Asia-Pac Financ Markets (2021). https://doi.org/10.1007/s10690-021-09338-4

[15] Bagaria V., Neu J., Tse D. (2020) Boomerang: Redundancy Improves Latency and Throughput in Payment-Channel Networks. In: Bonneau J., Heninger N. (eds) Financial Cryptography and Data Security. FC 2020. Lecture Notes in Computer Science, vol 12059. Springer, Cham. https://doi.org/10.1007/978-3-030-51280-4_17

[16] Engelshoven, Y. V., Roos, S. (2020). The Merchant: Avoiding Payment Channel Depletion through Incentives. https://arxiv.org/pdf/2012.10280.pdf