Non-Intrusive and High-Efficient Balance Tomography in the Lightning Network

Yan Qiao
qiaoyan@hfut.edu.cn
School of Computer Science and Information Engineering, Hefei University of Technology
Hefei, Anhui, China
Department of Computer Science, University of Victoria
Victoria, B.C., Canada

Kui Wu
wkui@uvic.ca
Department of Computer Science, University of Victoria
Victoria, B.C., Canada

Majid Khabbazian
mkhabbazian@ualberta.ca
Department of Electrical and Computer Engineering, University of Alberta
Edmonton, A.B., Canada

ABSTRACT

The Lightning Network (LN) is a second layer technology for solving the scalability problem of blockchain-based cryptocurrencies such as Bitcoin. The LN nodes (i.e., LN users), linked by payment channels, can make payments to each other directly or through multiple hops of payment channels, subject to the available balances of the serving channels. In current LN implementation, the channel capacity (i.e., the sum of the bidirectional balances in the channel) is open to the public, but the bidirectional balances are kept secret for privacy concerns. Nevertheless, the balances can be directly measured by conducting multiple *fake* payments to probe the precise value of the balance. Such a method, while effective, creates many fake invoices and incurs high cost when used for discovering balances for multiple users.

We present a novel *non-intrusive balance tomography (NIBT)* method, which infers the channel balances by performing legal transactions between two pre-created LN nodes. NIBT iteratively reduces the balance ranges and uses an efficient balance inference algorithm to find the optimal payment in each iteration to cut off the maximum balance ranges. Experimental results show that NIBT can accurately infer about 92% of all covered balances with an extremely low cost.

CCS CONCEPTS

• **Security and privacy → Spoofing attacks**; *Pseudonymity, anonymity and untraceability*; • **Computing methodologies →** *Optimization algorithms.*

KEYWORDS

Cryptocurrency; Blockchain; Lightning network; Network tomography

1 INTRODUCTION

As blockchain-based cryptocurrencies, such as Bitcoin [17] and Ethereum [27], have been widely adopted today, the number of cryptocurrency-based transactions increases in a fast pace. Due to the inherent feature of decentralization, however, blockchain-based cryptocurrencies often rely on some global algorithms, such as Proof-of-Work consensus algorithm [8], to confirm each transaction. This greatly limits the transaction rate within tens of transactions per second whereas other traditional payment networks such as Visa can support peaks of up to $47,000$ transactions per second [24].

To address the scalability issue of blockchain, extensive work has been done in the past years in several orthogonal directions [13]. Among them, *payment channel network* (PCN) [19], which carries out transactions off-chain, is one of the most promising proposals. Users, who would like to make payments over a PCN, open payment channels to PCN and lock a certain mount of fund as a deposit secured by a smart contract. Then payments are made by re-adjusting the fund allocation on the channels. These channels form a network in which payments can be routed between any two users (under constraints which will be disclosed later). Most transactions on PCN can be made off-chain without involving the main blockchain except some special situations, e.g., i) channel establishment, ii) channel close-out, and iii) the rare events of non-cooperative behaviors (e.g., dispute). Thus, the payment overhead on the main blockchain can be drastically reduced.

Currently, several fully-fledged PCN systems have been designed [3, 4, 19]. Among them, the Lightning Network (LN) [19] is recognized as the most prominent PCN in the Bitcoin community. LN not only significantly improves the scalability of Bitcoin blockchain, but also enables users to perform payments privately with low or negligible fees. To make a tradeoff between the routing efficiency and the user privacy, LN on one hand publishes the channel capacity (i.e., the sum of the bidirectional balances in the channel) together with the IP address of each LN node, but on the other hand preserves the balance of each node on the channel and applies onion-routing protocol [5]. With onion-routing, users other than the payer and the payee do not know who pays to whom; they only know from whom the payment is received (i.e., the immediate upstream neighbor) and to whom the payment should be forwarded (i.e., the immediate downstream neighbor). An illustrative example of the LN is shown in Fig. 1.

The balances of the channels are critical information in LN. *On the positive side*, this information may help users quickly search for feasible payment paths [20, 21, 28]. In addition, it can facilitate the detection of unbalanced channels (i.e., a channel is said unbalanced when one balance is much higher than the other), which can severely hinder the liquidity of LN [6, 22]. *On the negative side*, this information discloses the privacy of users, and can be used by misbehaving users to lock down other users' balances to obtain a dominant position in LN [18]. Moreover, a recent study shows that it

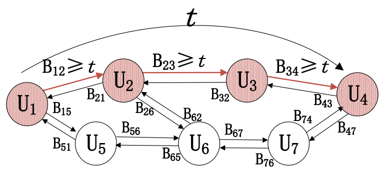


Figure 1: Example of the Lightning Network: the payment channel between user U_i and U_j includes the balance of U_i (denoted by B_{ij}) and the balance of U_j (denoted by B_{ji}). The channel capacity C_{ij} is defined as $C_{ij} = B_{ij} + B_{ji}$. C_{ij} is made open to the public but the balance of each user is private. The payer U_1 can send payment of amount t to the payee U_4 via the path $U_1 \to U_2 \to U_3 \to U_4$ only if B_{12}, B_{23}, B_{34} all are higher than t.

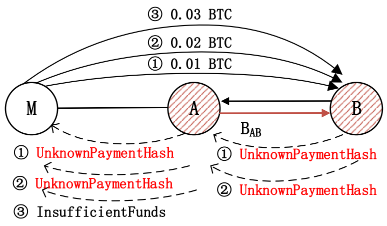


Figure 2: Direct measurement on channel balances [10].

would be possible to track payments from the sender to the receiver, thereby breaking relationship anonymity and value privacy, should the balance information be disclosed to the public [23]. Weighing the benefits and the damages, LN hides the balance information as its current practice.

Nevertheless, it is not difficult to disclose channels' balances. In this regard, the onion-routing policy of LN is actually a double-edged sword. It protects the privacy of the payer and payee, but in the meanwhile provides attackers with opportunities to discover the balances of channels [10]. As shown in [10], attackers can directly measure an unknown balance of a payment channel by executing multiple fake payments. In the example illustrated in Fig. 2, assume that the attacker M wants to discover the balance B_{AB}. It first opens a payment channel with A, and conducts three fake payments to B, which carried 0.01, 0.02, 0.03 BTC, respectively. The first two payments arrive at B, but B cannot redeem them due to the incorrect payment hash. Thus, B returns an error message to A which then forwards it to M. The third payment cannot go through A due to the insufficient balances of B_{AB} (i.e., $B_{AB} < 0.03$ BTC). In this case, A returns a message indicating "InsufficientFunds" to M. In this way, M can determine that the balance B_{AB} is between 0.02 BTC and 0.03 BTC. Such balance discovery attack is quite simple but effective since it is not easy to trace back to M due to the protection with onion-routing [10].

Note that "attackers" who would like to detect the balances of channels may not necessarily have malicious purpose. Instead, "attackers" may use the balance information for the social well-being, e.g., solving the unbalanced channel problem. However, either from a malicious purpose's view or from a well-intentioned purpose's view, measuring the channels directly by extensive fake payments

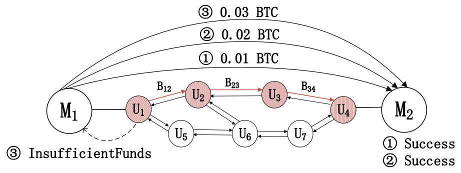


Figure 3: Infer channel balances with non-intrusive balance tomography.

may be problematic for the following reasons: (1) **Intrusive**, abundant fake payments may not only disturb the normal transactions of payees who lie behind the victim parties (i.e. user B in Fig. 2), but also cause dishonesty concerns; (2) **Inefficient**, to obtain one balance of a channel, "attackers" need to perform tens of iterations in average, consuming nearly one minute; (3) **Expensive**, to measure the balances on multiple nodes, "attackers" have to open multiple channels, each of which connects to one node, generally causing high fees in channel opening [1].

To avoid above issues, we exploit an indirect way to *infer* channel balances based on legal payments, which are routed through the victim channels. More specifically, we use a pair of accounts, connecting to LN with two payment channels, and perform transactions between our two accounts to infer all the intermediate balances of channels that covered by the transactions. The two accounts could be created by the same attacker or by two collusive attackers. The payment is made from one attack account to the other attack account just like transferring money from the left hand to the right hand of the same person. By doing this, balances information can be obtained silently - without any abnormal events, efficiently - all covered balances are inferred simultaneously, and economically - only need to open two channels. We name this method *non-intrusive balance tomography (NIBT)*, and the framework is shown in Fig. 3.

In summary, this paper makes the following contributions:

- We pioneer the use of network tomography [26] for inferring the channel balances in LN. Through conducting *legitimate* payments between our two accounts, we infer the balances using the results of these payments: if a payment is successfully fulfilled, all balances along the payment paths are larger than the payment amount; Otherwise, the balance of the channel that failed the payment is smaller than the payment amount. Thus, balance inference incurs min-type operations and is more challenging than existing network tomography research that studied additive metrics, such as delay and traffic rate [15, 26], where the value of a path is the total value of all the links on the path.
- We design an adaptive payment strategy to infer all covered balances accurately and efficiently. For a given path set between the two accounts, we compute an optimal payment on each of the path that can mostly cut off the current possible ranges of balances, and conduct the payment sequentially on the optimal path to shrink the ranges of balances to their minimum. To accelerate the algorithm, we prove that the gain function is a piece-wise concave function, based on which binary search can be applied to search for the global optimal solution.

- We evaluate NIBT with simulation over real Lightning Network topology. Experimental results show that NIBT can accurately infer about 92% of all covered balances simultaneously with the cost less than 4 USD in total. Besides, it can also discover 94% and 98% of the slightly and seriously unbalanced channels, respectively.

2 BACKGROUND AND RELATED WORK

2.1 The Lightning Network (LN)

LN is built on top of the bitcoin blockchain. It is a peer-to-peer network, where nodes can make transactions with each other without involving the main blockchain. Each node in LN is linked by *payment channels*. To initiate a payment to a remote node, the payer has to i) discover a *feasible route* to the recipient, ii) generate a *security key* to protect the funds from nodes along the path, iii) transfer the funds to its neighbouring node by adjusting the two *balances* in their payment channel.

2.1.1 Payment Channel. A payment channel between two bitcoin parties can be established by both parties depositing a certain amount of funds into a joint account with 2-by-2 multi-signature address. The total deposited funds is called the *channel capacity*. Once a payment channel is established, both parities, who share the payment channel, register an actual Bitcoin transaction on the Bitcoin blockchain and then announce it to the rest of LN.

2.1.2 Balances in Payment Channels. The actual transferring ability of a payment channel is determined by how the funds are distributed between the two parties. The division of the funds between party A and party B is called the balance of A and the balance of B, respectively. Assume A's balance is B_a. Then A can make a payment to B with an amount P_a as long as $P_a \leq B_a$. After the transaction, the balance of A will be decreased by P_a while the balance of B will be increased by the same amount. The total capacity of the payment channel remains unchanged and is known in the whole LN. Nevertheless, the balances of both parties are secretly preserved for the sake of privacy.

2.1.3 Routing in LN. LN adopts a source routing policy. Any user, who would like to make a payment to another user other than its direct neighbor, should find a feasible routing path, and pay a small fee to the intermediate nodes on the path. Payments on the paths are conducted with the onion-routing protocol [9]. In addition, LN includes mechanisms to protect the security and ensure the atomicity of transactions [19].

2.2 Criticality of Channel Balances

In present LN specifications [19], the balances on both directions of a channel are considered private information as disclosing this information to public may allow a malicious user to launch various attacks.

- **Lockdown attack:** The goal of the lockdown attack is to block one or more Lightning nodes in multipath payments, in order to win a dominant position in LN [18]. With the exact balance information, misbehaving nodes can make several payments that go through the victim nodes to use up all balances of these nodes, hence limiting the capability of these nodes in relaying payments.
- **Payment retrospect:** LN performs onion routing for each payment to protect the privacy of payers and payees. By monitoring the variations of the real-time balances, it is not difficult to reconstruct the whole path of each payment [11].

On the other hand, the absence of knowledge of channel balances may, to a certain degree, degrade the performance of certain functionalities of LN.

- **Inefficient routing:** To make a payment to a recipient, the payer should find a feasible path with sufficient funds route the payment. Without the knowledge of balances, the payer may select an infeasible path (i.e., a path on which there is a channel with insufficient balance). As a result, the payer may have to try several paths before he finds a feasible one.
- **Skewed channels:** A skewed channel means one balance in the channel is much higher than the other. The skewed channel is generally formed when the payment flows from one direction are more than the other direction. If such a situation lasts for a long time, one balance in the channel will become zero, which means no payment flow can go through from this direction. The skewed channels seriously hinder the liquidity of LN, and cause other concerns as stated in [22][6]. The absence of balance information makes skewed channels invisible, and impede routing strategies from fairly allocating payment channels.

2.3 Challenges in Inferring Balances

A channel's balance can be inferred by repeatedly probing the channel with multiple payments of different amounts [10]. At a first glance, probing balances of a channel may seem like bandwidth inference in traditional computer networks. Unfortunately, the balance inference problem in LN is more challenging than the bandwidth inference problem as explained below.

2.3.1 Cost. In the balance inference problem one needs to open at least one payment channel in LN, deposit enough funds in the channel and make multiple payments. These operations impose three kinds of cost [12]: (1) the transaction fee of opening and closing channels, (2) the routing fee collected by intermediate LN nodes once the payments have been fulfilled, and (3) the *potential* loss for locking funds in the channel. An efficient balance inference method should minimize the overall cost. Among the above three, the first cost (about 1.53 USD[1] per channel) dominates the second cost (about 1.09×10^{-6} USD per-hop per-USD transfer). The third cost is difficult to evaluate because it totally depends on how you invest the locked funds in other places. Hence, we only consider the first two costs in the paper.

2.3.2 Constraint on the Maximum Payment. LN limits the maximum amount one can transfer in a single payment to about 0.043 BTC, and the maximum amount one can put in a channel (i.e. the

[1]In LN, the cost is calculated in the unit of satoshis. In the paper, we use the market price as of May 25, 2020 for cost estimation, which is 1 satoshis = 8.8982×10^{-5} USD. The first cost is estimated according to the averaged transaction fee from Feb 25, 2020 to May 25, 2020 [1].

maximum channel capacity) to about 0.167 BTC [19]. These constraints make it difficult to infer the exact balances of a channel with large capacity.

2.3.3 Balance Dynamics. Once a payment is successfully delivered, all the balances of the channels on the payment path will decreases accordingly by the payment amount. Therefore, unrestricted probing payments may lead to extensive skewed channels, which may block the normal transactions as well as the subsequent probings. In addition, the inferred results may become meaningless if the inference process alters the values of balances.

2.3.4 Dishonesty Concerns. Recent work propose several balance disclosure solutions [10, 23, 25]. All these solutions conduct many fake payments in order to reduce the cost and keep balances unchanged. Although, at the present, LN specification does not specify any punishment for making such fake payments, there is no reason to believe that the LN community is not concerned when facing spikes of fake payments.

We aim to tackle all the above challenges, and design an efficient balance inference approach, which has low cost, can infer large balances, does not change current balances and, above all, generates no fake payments (i.e., non-intrusive).

3 NON-INTRUSIVE BALANCE TOMOGRAPHY

3.1 Overview

We design a novel balance inference approach, NIBT, where we generate two LN nodes (e.g., M_1 and M_2 as shown in Fig. 3) and conduct end-to-end payments between the two nodes. This is the first time that the concept of network tomography [26] is introduced to this area and is modified to well address the above challenges for balance inference in LN. In our new approach, we first open two channels to connect the two nodes to LN. Next we search a group of paths (refer to Section 4.1 for details of path construction) between the two nodes to form a candidate path set. For each path, we calculate an optimal payment amount that can obtain the most balance information on that path. Among all paths, we select the path that can gain the most balance information and conduct payment with the optimal amount on this path. Finally, we update the balance distributions according to the result of the payment. The whole process repeats until the number of payments exceeds a given budget or we cannot obtain any more balance information.

Remark 1. *It is worthy noting a subtle detail in the initial stage when the two monitor nodes establish channels in LN. Their connected nodes may not deposit any funds in the channels! For example, in Fig. 3, the balance from U_1 to M_1 and the balance from U_4 to M_2 may both be zero in the beginning, and in this case a payment from M_1 to M_2 would not be possible. This is the so-called inbound capacity problem [14], which may exist for any newly built channel. The problem can be easily solved with various methods, e.g., our two accounts can increase the inbound capacity by spending in LN [14].*

Now we explain why our approach can tackle the challenges of balance inference in LN.

- **Low cost:** As mentioned earlier, among the three kinds of cost for performing balance inference, the fees for opening and closing channels dominate the others. Compared to the direct probing method (shown in Fig. 2) that attacks one node with one channel [10], we just need to open two channels in total to infer all balances covered by our payment paths.
- **Non-interference in current balances:** A successful payment changes the balances of the channels on the payment path. To restore the balances, we require the recipient quickly refund the sender using the reverse path.
- **Non-intrusive for other LN nodes:** All payments that we use for balance inference are legal payments without any misbehavior. In addition, since our method does not interfere the current balances, it causes no disturbance on LN users' transactions.
- **Applicable for large channels:** All existing approaches for balance disclosure conduct *fake* payments to probe the channels. It is impossible for these methods to probe the channels whose balances exceed the maximum amount allowed in one payment (0.043 BTC). This is because they do not create any successful payment and thus cannot accumulate multiple payment amounts to probe a large balance. With our approach, we can let the recipient node hold multiple payments simultaneously and then pay back the payments to the sender node after the information of the large balance has been probed.

3.2 Analysis and Problem Formulation

We model LN with a directed graph $G(N, E)$, where N is the set of LN users and E is the set of channels between the users. For a channel between user a and user b, b_{ab} represents the balance from the user a to user b. $c_{ab} = b_{ab} + b_{ba}$ represents the capacity of the channel. u_{ab} and l_{ab} denote the upper and the lower bounds of balance b_{ab}, respectively.

As shown in Fig. 3, M_1 and M_2 are our two monitoring nodes, which connect to LN and make payments to each other to discover channel balances. A path $p_i = \{e^i_1, \cdots, e^i_{n_i}\}$ includes a set of channels that carry a payment from one monitor to the other monitor. We use b^i_k and c^i_k to denote the *corresponding* balance (i.e., the balance along the path direction from the sender to the recipient) and the capacity of channel e^i_k, respectively. The upper bound and the lower bound of b^i_k is represented by u^i_k and l^i_k, respectively.

Remark 2. *We do not need to infer the balances on the two channels directly connected to the two monitors, e.g., in Fig. 3 the balance from M_1 to node U_1 and the balance from U_4 to M_2, as these balances are known to the monitors. Hence, $p_i = \{e^i_1, \cdots, e^i_{n_i}\}$ does not include the first and the last channels.*

To infer balances, the range of balances of the channels on path p_i is shrunken by conducting payments on the path. We use an indicate variable $p_i(m)$ to denote the result of a payment on p_i with an amount m, where $p_i(m) = 0$ means the payment has been successfully fulfilled, and $p_i(m) = k (0 < k \leq n_i)$ means the payment failed at the k-th channel. Note that LN offers the above information to the sender (i.e., a payment is either successful or failed at an intermediate channel).

Clearly, there is a correlation between the ranges of balances on the path and the payment result $p_i(m)$:

(1) $p_i(m) = 0$ if and only if all balances on the path are no smaller than m;
(2) $p_i(m) = k (0 < k \leq n_i)$ if and only if all balances before the k-th channel are no smaller than m and the balance of the k-th channel is smaller than m.

Therefore, we can deduce the upper bounds and lower bounds of balances on p_i based on the payment result on the path. Given a series of payment results $\mathbf{P}_i(\cdot) = \{p_i(m_1), p_i(m_2), \cdots, p_i(m_{t_i})\}$, the lower bound and the upper bound of balance $b_k^i (1 \leq k \leq n_i)$ can be updated, respectively, by

$$l_k^i = \max_{p_i(m_j)=0 || p_i(m_j)>k, 1 \leq j \leq t_i} m_j, \tag{1}$$

$$u_k^i = \min_{p(m_j)=k, 1 \leq j \leq t_i} m_j. \tag{2}$$

Since the range of a balance may be reduced multiple times during the inference process, based on the payment results of all paths that traverse balance b_{ab}, the lower bound and upper bound of balance b_{ab} can be updated, respectively, by

$$l_{ab} = \max_{b_k^i = b_{ab}, 1 \leq i \leq n} l_k^i \tag{3}$$

$$u_{ab} = \min_{b_k^i = b_{ab}, 1 \leq i \leq n} u_k^i, \tag{4}$$

where n is the total number of candidate paths between the two monitors.

To infer all balances accurately, we can conduct multiple payments with different payment amounts on all candidate paths and shrink the distribution ranges of balances based on the payment results. However, conducting too many payments not only takes time, but also increases the routing fees. Therefore, given a candidate set of end-to-end paths $\mathbf{P} = \{p_1, \ldots, p_n\}$ and a total budget η, we aim to shrink all balance ranges as much as possible. The optimization problem is formulated as follows:

$$Minimize \quad \sum_{b_{ab} \in \mathbf{e}} (u_{ab} - l_{ab}) \tag{5}$$

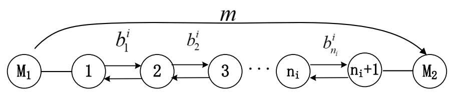
$$s.t. \quad \sum_{1 \leq i \leq n} |\mathbf{P}_i(\cdot)| \leq \eta \tag{6}$$

$$u_{ab} = \min_{b_k^i = b_{ab}, 1 \leq i \leq n} u_k^i \tag{7}$$

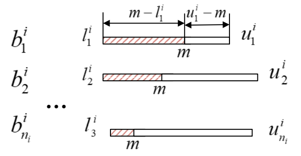
$$l_{ab} = \max_{b_k^i = b_{ab}, 1 \leq i \leq n} l_k^i \tag{8}$$

REMARK 3. ***(The budget** η**)** As discussed in Section 2.3, the cost for opening/closing channels in our method is fixed (i.e., fees for opening/closing two channels). Hence, we only need to control the routing fee. When setting a suitable budget η, we should also consider another important factor, the time used for balance inference, which is directly linked to the total number of payments (i.e., probes). For this reason, we use the total number of (end-to-end) payments to control the "budget" η, i.e., inequality (6). Since the per-hop transfer fee is extremely low, we can treat the routing fee for each end-to-end payment roughly the same, and as such the η value also well reflects the actual monetary cost.*

Due to the iterative bound update process, the optimization problem (5) does not render an analytical solution. To tackle this, we first calculate the optimal payment amount on *each path* that



(a) Payment amount m on path p_i.



(b) Balance ranges on p_i.

Figure 4: An example of path p_i carrying payment amount m.

can cut off the most balance ranges (Section 3.3). Then we conduct the payment and update the balance ranges based on the payment result. The above steps are repeated until the number of payments exceeds the budget η or no balance ranges can be cut off further (Section 3.4).

3.3 Optimal Payment on a Single Path

Before performing any measurement, we do not have any information regarding the balances. As such, the best one can do is to follow the "principle of insufficient reason" [7], i.e., "assigning uniform prior distributions to unknown parameters". Using this principle, we assume initially that the balance b_{ab} follows the uniform distribution on the range $[l_{ab}, u_{ab}]$. The probability density function of the balance b_{ab} is thus

$$f(x) = \begin{cases} \frac{1}{u_{ab} - l_{ab}} & l_{ab} \leq x \leq u_{ab} \\ 0 & otherwise \end{cases} \tag{9}$$

When we start to conduct payments the probability density function will be updated to a conditional probability function when the payment results influence the distribution of the balance. Given a payment amount m on path $p_i = \{e_1^i, \cdots, e_{n_i}^i\}$, the payment result will influence the distribution of b_k^i if and only if both of the following two conditions are satisfied:

(1) $l_k^i < m < u_k^i$;
(2) $p_i(m) = 0$ or $p_i(m) \geq k$;

In this case, the probability density function of b_k^i will be updated by

$$f(x | p_i(m) = 0 \; or \; p_i(m) > k) = \begin{cases} \frac{1}{u_k^i - m} & m \leq x \leq u_k^i \\ 0 & otherwise \end{cases} \tag{10}$$

or

$$f(x | p_i(m) = k) = \begin{cases} \frac{1}{m - l_k^i} & l_k^i \leq x \leq m \\ 0 & otherwise \end{cases} \tag{11}$$

Once any distribution of a balance on path p_i is updated by a payment result, we say that this payment on path p_i can *obtain a gain.* We use $G^i(m)$ to denote the expected gain of the payment, which is calculated by the expected length it can reduce from the balance ranges on p_i.

According to Remark 2, we do not need to consider the channels directly connected to the monitors. Hence, the expected gain $G^i(m)$ can be calculated by

$$\begin{aligned} G^i(m) =& G_1^i(m) + prob(b_1^i \geq m)G_2^i \\ &+ \cdots \\ &+ prob(b_1^i \geq m) \cdots prob(b_{n_i-1}^i \geq m)G_{n_i}^i \end{aligned} \tag{12}$$

where $0 \leq prob(b_k^i \geq m) \leq 1$, $1 \leq k \leq n_i$ is the probability that payment amount m can go through b_k^i and G_k^i is the expected gain on b_k^i. Suppose that the payment amount m splits the range of balance b_k^i on p_i as shown in Fig. 4. The probability of $prob(b_k^i \geq m)$ can be calculated by

$$prob(b_k^i \geq m) = \frac{u_k^i - m}{u_k^i - l_k^i} \tag{13}$$

The expected gain that the payment amount m can obtain from b_k^i is

$$\begin{aligned} G_k^i(m) =& \frac{u_k^i - m}{u_k^i - l_k^i}(m - l_k^i) + \frac{m - l_k^i}{u_k^i - l_k^i}(u_k^i - m) \\ =& \frac{u_k^i - m}{u_k^i - l_k^i} \cdot 2(m - l_k^i) \end{aligned} \tag{14}$$

Then Eqn. (12) can be rewritten as

$$G^i(m) = \sum_{k=1}^{n_i} 2(m - l_k^i) \cdot \prod_{j=1}^{k} \frac{u_j^i - m}{u_j^i - l_j^i} \tag{15}$$

Denote each item in Eqn. (15) as:

$$H_k^i(m) = 2(m - l_k^i) \cdot \prod_{j=1}^{k} \frac{u_j^i - m}{u_j^i - l_j^i} \tag{16}$$

In Eqn. (13)~(16), $\forall k : 1 \leq k \leq n_i$, if $m < l_k^i$, $m - l_k^i = 0$ and $\frac{u_k^i - m}{u_k^i - l_k^i} = 1$. If $m > u_k^i$, $m - l_k^i = u_k^i - l_k^i$ and $\frac{u_k^i - m}{u_k^i - l_k^i} = 0$.

Remark 4. *When m exceeds the maximum payment amount allowed in one payment (denoted by z), m will be split into multiple sub-payments: $z, z, \ldots, m - \lfloor \frac{m}{z} \rfloor \cdot z$. Then $G^i(m)$ will be calculated by summing up the gain of each sub-payments. To avoid redundancy, we only present the partial algorithm and algorithm analysis based on Eqn. (12) in this section, while the evaluations in Sec. 4 are carried based on the full version of our algorithm.*

Denoting $\hat{u}_k^i = \min\{u_j^i\}$, $1 \leq j \leq k$, we have $\forall k : 1 \leq k \leq n_i$,

$$H_k^i(m) \begin{cases} > 0 & l_k^i < m < \hat{u}_k^i \\ = 0 & \text{Otherwise} \end{cases} \tag{17}$$

Due to Eqn. (17), we define the interval $(l_k^i, \hat{u}_k^i)$ as the *effective interval*, denoted by EI_k, of the function $H_k^i(m)$.

Let $l^i = \min_{k=1}^{n_i} l_k^i$ and $u^i = \max_{k=1}^{n_i} \hat{u}_k^i$. Clearly the interval (l^i, u^i) is the widest range where $G^i(m)$ may be positive. Sort the values $\{l_1^i, \hat{u}_1^i, l_2^i, \hat{u}_2^i, \cdots, l_{n_i}^i, \hat{u}_{n_i}^i\}$ in the ascending order. Then these ordered values can divide (l^i, u^i) into *at most* $2 \times n_i - 1$ sub-intervals, which are denoted by $\lambda^i = \{\lambda_1^i, \lambda_2^i, \cdots, \lambda_{2\times n_i - 1}^i\}$. Each sub-interval λ_j^i falls in one of the following three (exclusive) cases:

(1) $|\lambda_j^i| = 0$, i.e., λ_j^i is a point.
(2) $|\lambda_j^i| > 0$ and $\forall k, m : 1 \leq k \leq n_i, m \in \lambda_j^i, H_k^i(m) = 0$.
(3) $|\lambda_j^i| > 0$, there exist some non-empty set $A_j^i \subseteq \{1, 2, \cdots, n_i\}$, such that $\forall k, m : k \in A_j^i, m \in \lambda_j^i, H_k^i(m) > 0$. In other words, some $H_k^i(m)$ are positive in the sub-interval λ_j^i.

Clearly, for the first and the second cases, $G^i(m) = \sum_{k=1}^{n_i} H_k^i(m) = 0$. For the third case, $\forall m : m \in \lambda_j^i, G^i(m) = \sum_{k=1}^{n_i} H_k^i(m) = \sum_{k \in A_j^i} H_k^i(m) > 0$. Hence, in the third case, we call λ_j^i as a *positive interval* of $G^i(m)$.

Proposition 1. *$G^i(m)$ is a strictly concave function with only one maxima on its positive intervals.*

The proof of Proposition 1 can be found in Appendix.

By Proposition 1, for each path, we can find the optimal payment amount m^* by searching the maximas on each *positive intervals* and selecting the maximum value among all these maximas. On each *positive interval* λ_j^i, we use binary search to find the maxima. This requires $O(\log |\lambda_j^i|)$ iterations. We name this process *searchOptAmount* and omit its pseudo-code to save space.

3.4 Iterative Payment and System Update

3.4.1 *Main Idea.*

After obtaining the optimal payment amounts on each path, we can form the optimal gain set $\mathbf{G}^* = \{G^1(m^{*1}), G^2(m^{*2}), \cdots, G^n(m^{*n})\}$. Then we conduct the payment with amount m^{*i} on path p_{i*}, where $i* = \arg\max_{i=1}^{n} G^i(m^{*i})$. This process is fullfiled with two functions *optPayment* and *conductPayment*. After that, we wait for the payment result.

Once a payment result $p_i(m)$ has been observed, we update the ranges of balances related to the payment result as follows:

(1) If $p_i(m) = 0$,

$$l_k^i = \max\{m, l_k^{\prime i}\}, \quad 1 \leq k \leq n_i \tag{18}$$

where $l_k^{\prime i}$ is the original lower bound of b_k^i.

(2) If $p_i(m) = h$,

$$l_k^i = \max\{m, l_k^{\prime i}\}, \quad 1 \leq k < h \tag{19}$$

and

$$u_h^i = \min\{m, u_h^{\prime i}\} \tag{20}$$

where $u_h^{\prime i}$ is the original upper bound of b_h^i.

If some balance ranges have been reassigned, we recompute the optimal payment amount of the paths which cover the updated balances.

Alg. 1 shows the pseudo-code of our non-intrusive balance tomography algorithm. The algorithm receives the candidate paths set $\mathbf{P}$ and capacities of covered channels $\mathbf{C}$ as inputs, and outputs the intervals of all balances $\mathbf{I}$. Alg. 1 first initializes the intervals of balances with their capacities $\mathbf{C}$ (line 1), i.e., the lower bound as 0 and the upper bound as the channel capacity. This initialization also indicates that the intervals have been updated. The number of payments is initially set to 0 (line 2). Then the algorithm starts to find out the optimal payment and updates the intervals by the payment result round by round until there is no payment that can obtain any gain or the number of payments n_p exceeds its threshold (line 3~14).

Algorithm 1: Non-Intrusive Balance Tomography (NIBT)

```
Input: P, C
Output: I
1  $\mathbf{I} \leftarrow init(P, C)$
2  $n_p \leftarrow 0$
3  while $max(G(m^*)) > 0 \& n_p \leq \eta$ do
4      $\mathbf{P}_{update} = findUpdate(\mathbf{I})$
5      for $p_i \in \mathbf{P}_{update}$ do
6          $\mathbf{EI}^i \leftarrow effectiveIntvl(\mathbf{I}, p_i)$
7          $\lambda^i \leftarrow positiveIntvls(\mathbf{EI}^i)$
8          for $\lambda^i_j \in \lambda^i$ do
9              $[Gain^{*i}_j, m^{*i}_j] \leftarrow searchOptAmount(\lambda^i_j, \mathbf{I}, p_i)$
10         $[Gain^{*i}, m^{*i}] \leftarrow \max_{\lambda^i_j}(Gain^{*i}_j, m^{*i}_j)$
11     $[p_{i*}, m^{*i}] \leftarrow optPayment(Gain^{*1}, \cdots, Gain^{*n})$
12     $Result = conductPayment(p_{i*}, m^{*i})$
13     $\mathbf{I} \leftarrow updateRange(p_{i*}, m^{*i}, Result)$
14     $n_p{+}{+}$
15 return I
```

In the *while* loop, it first detects the changes in **I** and determines the paths that cover these changes (line 4). Then for each path in $\mathbf{P}_{update}$, it computes the *EI*s for each balances on the path (line 6), and determines the positive interval set λ^i (line 7). On each interval in λ^i, it searches for the maximum point of the gain function and the corresponding payment amount, then assigns the maximum gain with its payment amount to the the optimal payment on the path (line 10). After updating the optimal gains of paths in $\mathbf{P}_{update}$, it finds out the path with the maximum gain over other paths and conducts the optimal payment amount on the path (line 11). Finally, the algorithm obtains the result of payment (line 12) and updates the intervals of balances **I** with the result (line 13).

3.5 Analysis of Algorithm

3.5.1 Time Complexity. The most time-consuming step in Alg. 1 is searching the optimal payment amount in each positive interval (line 9). The time complexity for calculating the gain of payment along path p_i is $O(|p_i|)$, where $|p_i|$ is the number of balances on path p_i. By Proposition 1, the gain function is concave on each positive interval, and we can use binary search to locate the optimal payment amount. Therefore, the maximum number of iterations on interval λ^i_j is $\log|\lambda^i_j|$. Therefore, the maximum complexity for searching an optimal payment amount on λ^i_j is $O(|p_i| \cdot \log|\lambda^i_j|)$. The time complexity for searching the optimal payment amount among all positive intervals is $O(|p_i| \cdot |\lambda^i| \cdot \log|\lambda^i_m|)$, where $|\lambda^i_m|$ is the average length of positive intervals in λ^i. Suppose the maximum number of payments is η and the averaged number of paths that need to be updated in each round is $\bar{n}$. The overall time complexity of Alg. 1 is $O(\eta \cdot \bar{n} \cdot |p_m| \cdot |\lambda^i| \cdot \log|\lambda^i_m|)$, where $|p_m|$ is the averaged length of the paths in **P**.

As the LN limits the maximum funds in each channel to 1.67×10^7 satoshis and the maximum path length to 20 hops, we have $\forall i$, $|p_i| \leq 20$ and $|\lambda^i| \leq 39$. In the worst case, $\forall i, j$, $|\lambda^i_j| = \frac{1.67\times10^7}{39} = 4.28\times10^5$. Therefore, the worst time complexity of Alg. 1 is $O(\eta \cdot |\mathbf{P}| \cdot 20 \cdot 39 \cdot \log(4.28\times10^5))$, i.e. $O(1.5\times10^4 \cdot \eta \cdot |\mathbf{P}|)$.

Nevertheless, the time complexity of Alg. 1 is extremely low in practice for the following reasons. First, according to Sec. 3.5.2, long paths are unlikely to be selected as candidate paths; Secondly, the majority of the channels have small capacities [2]; Finally, our experimental results demonstrate that the number of paths that need to be updated in each round only takes up less than 10% of all candidate paths in **P**. Experimental results in Sec. 4.2 show that it only requires 5 seconds to search 5300 optimal payments over 200 candidate paths.

Furthermore, numerical results show that the probability that the optimal payment amount falls in intervals other than the interval with the maximum midpoint gain is very small (less than 5%). Hence, the time complexity of Alg. 1 can be further reduced if we do not always require the optimal payment amount.

Remark 5. *When η value is set very big, Alg. 1 may not perform η payments and pre-terminates when no payment can create any gain.*

3.5.2 Accuracy. For a given path p_i, which covers balances $\{b^i_1, b^i_2, \cdots, b^i_{n_i}\}$, we use a set of n_i intervals I^i_k, $1 \leq k \leq n_i$ to denote the ranges of the balances (after our inference) as follows: $\forall k : 1 \leq k \leq n_i, b^i_k \in I^i_k$. Let y^i_k denote the minimum balance of the first k balances covered by path p_i, i.e., $y^i_k = \min\{b^i_1, b^i_2, \cdots, b^i_k\}$.

Proposition 2. *Suppose the original range of b^i_k is $[l^i_k, u^i_k]$ before inference. Then, the expected length of the range after inference $|I^i_k|$ is inversely proportional to y^i_{k-1} and proportional to k.*

The proof of the above proposition can be found in the Appendix.

By the above proposition, Alg. 1 can obtain a higher accuracy if the balances on most paths are in a descending order. In addition, shorter paths are more likely to improve the accuracy. These insights are used as heuristics in constructing candidate paths in the next section.

Remark 6. *Since we have two LN accounts, payments can always be conducted in two directions. Therefore, even in the worst case that all balances are in monotonically increasing order on one path, we can easily construct the candidate path in its opposite direction.*

4 PERFORMANCE EVALUATION

4.1 Experimental Settings

4.1.1 Lightning Topology. We generate the LN topology based on the snapshot of LN taken on July 05, 2020. The whole network includes 6072 nodes and 30026 payment channels.

4.1.2 Capacities and Balances. We generate the capacities of channels based on the public LN statistics [2], where the upper bound of the capacity is 0.167 BTC. Once the capacity c_{ij} for channel e_{ij} is generated, the balance of one side b_{ij} is then drawn from $[0, c_{ij}]$ following uniform distribution. The balance on the other side is then set to $b_{ji} = c_{ij} - b_{ij}$.

4.1.3 Candidate Paths. We create two nodes as our two LN accounts, and connect them to LN by two payment channels. In order to verify performance of NIBT under different candidate path sets, we use two methods to connect our accounts to the LN nodes (1)

Connecting method #1 (CM#1): connect our two accounts to the top two nodes that have the highest node degree; (2) Connecting method #2 (CM#2): connect our two accounts to the top two nodes that have the largest total channel capacity. We use three methods to construct the paths between our two accounts: (1) Path constructing method #1 (PCM#1): construct the top n shortest paths; (2) Path constructing method #2 (PCM#2): construct n paths whose capacities are (roughly) in descending order and their hops are as few as possible; (3) Path constructing method #3 (PCM#3): randomly construct n paths. Hence, we form 6 candidate path sets as shown in Tab 1.

We do not consider more intelligent path construction in this paper. Nevertheless, even with candidate paths built with simple heuristics (such as *Candidate-1* and *Candidate-2*), we can obtain good inference results. More intelligent path construction methods are left as our future work.

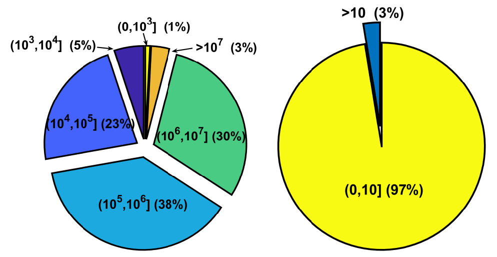
4.1.4 Payments. Payments are conducted sequentially after the result of the last payment is obtained. According to LN specification, the maximum amount in one transaction is 0.043 BTC. Therefore, when the conducted payment amount exceeds this value, we need to divide the payment into several sub-payments each of which carries an amount within 0.043 BTC.

4.1.5 Estimation of Cost. We estimate the cost for opening and closing one channel as 1.72×10^4 satoshis ($\approx$ 1.53 USD) according to the averaged transaction fee from Feb 25, 2020 to May 25, 2020 [1]. The routing fee charged by intermediate nodes includes two parts: a constant base fee that nodes charge per transfer and a flexible additional fee that is proportional to the transferred amount. For LN nodes, the default values for base fee and additional fee are 1000 msat (10^{-3} satoshi) and 1 msat (10^{-6} satoshi) per transferred satoshi. According to the statistics in [16], 98% and 65% of LN nodes define their base and additional charges, respectively, equal to or smaller than the default values. Hence, without a risk of underestimating the cost, we assume that all channels use the default values as their base and additional charges for transfers.

4.2 Results

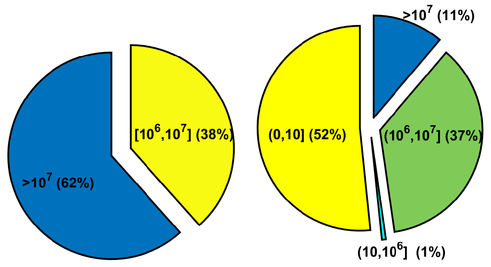
Tab. 1 compares the performance of our method under the six candidate path sets. We did not limit the budget in Tab. 1. Instead, we only stop the inference when there is no payment can cut more balance ranges. We separate the balances into two groups: (1) small balances that are smaller than 0.043×10^8 ($\approx$ 382.6 USD), taking up about 89.8% of all balances and (2) large balances that are larger than 0.043×10^8, taking up about 10.2% of all balances. We can see from the table that as the number of paths in the candidate set increases, the number of covered balances, the number of required payments, and the routing fees all grow. Nevertheless, the percentage of balances that can be accurately inferred is relatively stable.

Comparing the first three candidate sets, *Candidate-1* and *Candidate-2* are much more accurate than *Candidate-3* as they have much shorter paths. This is consistent with Proposition 2. However, they also require the higher number of payments and the higher routing fees to reach their highest accuracies. *Candidate-1* and *Candidate-2* have similar performances. That also indicates that when the length of a candidate path is quit short (such as 2 hops), considering the orders of capacities seems not have much superiority.



(a) Ranges of balances before inference. (b) Ranges of balances after inference.

Figure 5: Inferring the ranges of small balances: each slice denote the proportion of the marked ranges. Small balances mean the balances smaller than 0.043×10^8 satoshis ($\approx$ 382.6 USD). They take up about 89.8% of all balances.



(a) Ranges of balances before inference. (b) Ranges of balances after inference.

Figure 6: Inferring the ranges of large balances: each slice denote the proportion of the marked ranges. Large balances mean the balances larger than 0.043×10^8 satoshis ($\approx$ 382.6 USD). They take up about 10.2% of all balances.

Comparing the last three candidate sets, *Candidate-5* is more accurate than the other two candidates, which indicates when the path length is greater than 2, the order of capacities begins to play a role. The loss of accuracy of *Candidate-6* is also due to the extra length of the paths.

Comparing the first three candidates with the last three candidates, *Candidate-1*, *Candidate-2* and *Candidate-3* generally performs better than *Candidate-4*, *Candidate-5* and *Candidate-6*, which indicates that connecting with the nodes with higher degrees can make a better inference.

In the following, we evaluate NIBT from different aspects under *candidate-1*. As different number of paths in the candidate set did not present much difference on accuracy, we only report the results of *candidate-1* with 200 paths in the set.

4.2.1 Ability to Cut off Balance Ranges. Fig. 5 and Fig. 6 plot the distribution of the ranges of balances in the small and large groups, respectively. Note that the range of a balance equals the upper bound minus the lower bound of the balance. So the smaller the

Table 1: Performance of NIBT under Different Candidate Path Sets

Candidate Path Sets		Number of Covered Balances	Averaged Path Length	Fraction of Determined Balances			Number of Required Payments	Routing Fees (USD)
				Total	$\leq 4.3 \times 10^6$	$> 4.3 \times 10^6$		
Candidate-1 CM#1,PCM#1	100	398.0	1.99	91.51%	96.91%	46.35%	2843.8	0.2359
	200	738.0	2.14	92.47%	97.34%	51.64%	5322.9	0.4760
	300	942.0	2.43	93.52%	97.87%	57.70%	6866.1	0.6349
Candidate-2 CM#1,PCM#2	100	398.0	1.99	91.36%	97.02%	47.01%	2851.3	0.2679
	200	688.2	2.27	91.78%	97.30%	54.08%	5062.5	0.5561
	300	904.0	2.51	92.08%	97.21%	61.87%	6804.4	0.8857
Candidate-3 CM#1,PCM#3	100	408.4	9.09	62.40%	67.97%	16.70%	2114.2	0.0216
	200	755.3	8.98	62.68%	68.05%	17.97%	3952.4	0.0535
	300	1085.6	9.05	62.06%	67.11%	18.87%	5623.5	0.0743
Candidate-4 CM#2,PCM#1	100	388.7	2.85	82.44%	88.46%	36.87%	2573.7	0.1938
	200	734.7	3.18	76.05%	82.00%	28.89%	4562.4	0.2645
	300	1021.3	3.42	73.48%	79.33%	26.74%	6123.6	0.3442
Candidate-5 CM#2,PCM#2	100	332.9	2.98	91.16%	92.95%	59.09%	2314.2	0.1490
	200	616.9	3.18	89.36%	91.01%	58.06%	4350.2	0.2427
	300	863.1	3.37	88.79%	90.48%	57.05%	6101.5	0.2859
Candidate-6 CM#2,PCM#3	100	418.4	10.00	52.68%	56.99%	11.97%	1888.1	0.0236
	200	760.7	9.93	51.46%	55.97%	11.01%	3407.2	0.0376
	300	1083.5	9.94	50.31%	54.93%	9.90%	4892.4	0.0580

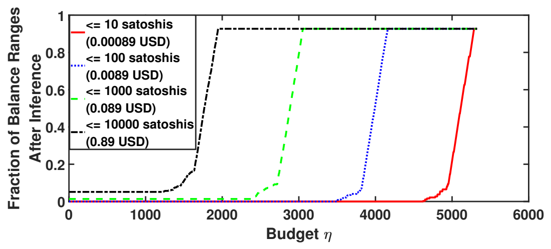


Figure 7: Accuracy of NIBT under different budgets η.

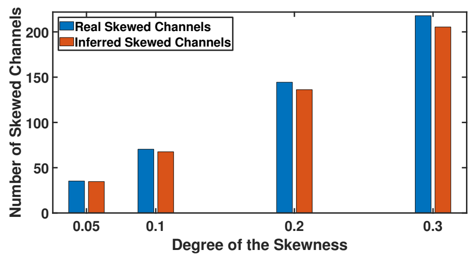


Figure 8: The number of skewed channels that were detected.

range value, the more accurate the balance estimation. The number $(0, 10^3](1\%)$ in Fig. 5(a) means that 1% of balances range are in the $(0, 10^3]$. From Fig. 5(a), 94% of the original balance ranges are larger than 10^4 satoshis ($\approx$ 0.8898 USD), and 72% are larger than 10^5 satoshis ($\approx$ 8.898 USD) before we perform balance inference. After inference, 97% of balances have the ranges smaller than 10 satoshis ($\approx$ 0.00089 USD). From Fig. 6(a), all balance ranges are larger than $4.3 * 10^6$ ($\approx$ 382.6 USD), 62% of them are larger than 10^7 ($\approx$ 889.82 USD). After inference, 52% of them have the ranges smaller than 10 satoshis ($\approx$ 0.00089 USD) and only 11% of them are lager than 10^7 satoshis ($\approx$ 889.82 USD).

Fig. 7 plots the performance of our method under different payment budgets. From the figure, our method requires less than 2000 payments to cut off the ranges of more than 92% balances to 0.89 USD, and requires about 5300 payments to cut off these ranges to 0.00089 USD. Searching for the optimal payments can be performed extremely fast that the 5300 payments only require about 5 seconds in total.

4.2.2 Ability to Detect Skewed Channels. Fig. 8 presents the number of skewed channels covered by the end-to-end paths and the number of skewed channels that can be detected during the inference. The fraction of the abscissa means the *degree of the skewness*, defined as the ratio between the smaller balance over the capacity of the channel. For example, 0.1 means a balance on one side of this channel only accounts for 10% of the channel capacity, while the balance on the other side accounts for 90%. From the figure, the majority of skewed channels can be detected during the balance inference, especially for the seriously skewed channels (e.g., when the degree of skewness is 0.05).

4.2.3 *Summary.*

NIBT has both the ability of inferring balance - accurately inferring more than 97% of small balances (smaller than the maximum payment amount in one transaction) and nearly 52% of large balances (larger than the maximum payment amount in one transaction) and the ability of detecting skewed balances - detecting 98% seriously skewed balances and 94% slightly skewed balances.

4.3 Comparison with Direct Measurement

NIBT is *in principle* different from the method using direct measurement (DM) [10]. Nevertheless, we can still compare these two methods from different angles, from which readers can have a better view on the cons/pros of both methods. We averaged the results and compared the two methods in Tab. 2.

DM in average needs to conduct 14 fake payments to obtain an accurate balance. Among them, about half of these payments cause error messages "unknown payment hash". The other half of the payments fail due to the insufficient funds of the balance. NIBT creates no error payments. NIBT in average requires about 9 payments to infer each balance, about 5 of them failed due to "insufficient funds" and about 4 of them are successfully fulfilled[2]. For the balances that are smaller than the maximum payment amount allowed in one payment, DM can accurately find out all of their balances through multiple iterations. NIBT can accurately infer 97% of them because the information of about 3% of the balances is lost due to the smaller balances on the same path. For large balances that are larger than the maximum balances allowed in each payment, DM is not able to obtain their balances because the recipients will not hold multiple fake payments. NIBT can infer about 52% of these balances because the recipients can hold multiple successful payments and perform inference all together. 48% of the balances cannot be inferred because the smaller balances on the same paths filter out the information of large balances. As all payments in DM are fake payments, its cost only contains the fees for opening/closing channels. However, as DM can only measure the balances of node that is directly connected with the "attacker" node, it needs to open many channels for network-wide balance inference. Since the average transaction fee for channel opening/closing is about 1.53 USD per channel, the total cost will be considerably higher when the number of involved nodes is big. For NIBT, the total fees for opening/closing channels is only 3.06 USD as we only need to open two channels in total. The average routing cost for successful payments is 0.00064 USD per balance. Even if we infer all balances in the sampled LN (30026 channels), the total routing cost is less than 20 USD.

5 CONCLUSION

We pioneered *balance tomography* for accurately inferring balances of LN without any interference or fake payments. Our method first opens two accounts each opening a channel in LN, and then conducts multiple legal payments sequentially between the two accounts. To optimize each payment, we calculate an optimal payment amount on each path based on the current balance distribution, and then update the distribution after we conduct an optimal payment and observe the payment result. We evaluate our method using an snapshot of the LN Network. Experimental results show that about 92% of all covered balances can be accurately inferred with extremely low cost.

In this paper, we used heuristics for building the candidate paths between the two accounts. As our future work, we plan to use reinforcement learning to find the best places for creating the channels and intelligently constructing the candidate paths.

[2]The number of successful payments includes both the payments we use to infer the balance and the payments we use to compensate the balances.

6 ACKNOWLEDGEMENTS

This research was supported by Anhui Provincial Natural Science Foundation (No. 2008085MF203), the Open Foundation of State Key Laboratory of Networking and Switching Technology (No. SKLNST-2018-1-10), and the Natural Sciences and Engineering Research Council of Canada (NSERC) (No. RGPIN-2018-03896 and No. RGPIN-2016-05660).

REFERENCES

[1] [n.d.]. Bitcoin Transaction Fee historical chart. https://bitinfocharts.com/en/comparison/bitcoin-transactionfees.html. Project's website.

[2] [n.d.]. Bitcoin Visuals. https://bitcoinvisuals.com/lightning. Project's website.

[3] [n.d.]. Raiden network. http://raiden.network/. Project's website.

[4] [n.d.]. Thunder network. https://github.com/blockchain/thunder. Project's website.

[5] Sergio Castillo-Pérez and Joaquin Garcia-Alfaro. 2013. Onion routing circuit construction via latency graphs. *Computers & security* 37 (2013), 197–214.

[6] Marco Conoscenti, Antonio Vetrò, Juan Carlos De Martin, and Federico Spini. 2018. The cloth simulator for htlc payment networks with introductory lightning network performance results. *Information* 9, 9 (2018), 223.

[7] Bradley Efron and Trevor Hastie. 2016. *Computer age statistical inference*. Vol. 5. Cambridge University Press, Shaftesbury Road, Cambridge, CB28BS, United Kingdom.

[8] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. Association for Computing Machinery, Vienna, Austria, 3–16.

[9] David Goldschlag, Michael Reed, and Paul Syverson. 1999. Onion routing. *Commun. ACM* 42, 2 (1999), 39–41.

[10] Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. 2019. On the difficulty of hiding the balance of lightning network channels. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, Auckland, New Zealand, 602–612.

[11] George Kappos, Haaroon Yousaf, Ania Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. 2020. An Empirical Analysis of Privacy in the Lightning Network. *arXiv preprint arXiv:2003.12470* (2020).

[12] Nida Khan et al. 2019. Lightning network: A comparative review of transaction fees and data analysis. In *International Congress on Blockchain and Applications*. Springer, Ávila, Spain, 11–18.

[13] Soohyeong Kim, Yongseok Kwon, and Sunghyun Cho. 2018. A survey of scalability solutions on blockchain. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Jeju, South Korea, 1204–1207.

[14] Dmitry Laptev. 2019. Solutions to inbound capacity problem in Lightning Network. *URL: medium.com/lightningto-me/practical-solutions-to-inbound-capacity-problem-in-lightning-network-60224aa13393* (2019).

[15] Liang Ma, Ting He, Kin K Leung, Don Towsley, and Ananthram Swami. 2013. Efficient identification of additive link metrics via network tomography. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, Philadelphia, PA, USA, 581–590.

[16] Ayelet Mizrahi and Aviv Zohar. 2020. Congestion attacks in payment channel networks. *arXiv preprint arXiv:2002.06564* (2020).

[17] Satoshi Nakamoto and A Bitcoin. 2008. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* (2008).

[18] Cristina Pérez-Sola, Alejandro Ranchal-Pedrosa, J Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquin Garcia-Alfaro. 2019. LockDown: Balance Availability Attack against Lightning Network Channels.

[19] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments.

[20] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. 2016. Flare: An approach to routing in lightning network. *White*

Table 2: Comparison with Directly Measurement (DM)

Methods	NIBT	DM
Error payments per balance (Due to fake invoices)	0	7.07
Failed payments per balance (Due to insufficient Funds)	5.13	7.02
Successful payments	4.15	0
Prob. to disclosure small balances	97%	100%
Prob. to disclosure large balances	52%	0
Cost on channel opening (USD)	3.06 in total	1.53 per node
Cost on routing (USD)	0.00064 per balance	0

Paper (2016).

[21] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2017. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748* (2017).

[22] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia Fanti, and Mohammad Alizadeh. 2020. High Throughput Cryptocurrency Routing in Payment Channel Networks. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. USENIX, SANTA CLARA, CA, USA, 777–796.

[23] Sergei Tikhomirov, Rene Pickhardt, Alex Biryukov, and Mariusz Nowostawski. 2020. Probing Channel Balances in the Lightning Network. *arXiv preprint arXiv:2004.00333* (2020).

[24] Manny Trillo. 2013. Stress test prepares VisaNet for the most wonderful time of the year. *URl: http://www. visa. com/blogarchives/us/2013/10/10/stress-testprepares-visanet-for-the-most-wonderful-time-of-the-year/index. html* (2013).

[25] Gijs van Dam, Rabiah Abdul Kadir, Puteri NE Nohuddin, and Halimah Badioze Zaman. 2019. Improvements of the Balance Discovery Attack on Lightning Network Payment Channels. *IACR Cryptol. ePrint Arch.* 2019 (2019), 1385.

[26] Yehuda Vardi. 1996. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American statistical association* 91, 433 (1996), 365–377.

[27] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[28] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, Dejun Yang, and Jian Tang. 2018. Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, Hangzhou, China, 1–9.

APPENDIX

Proof of Proposition 1

Proof: A detailed proof is lengthy and tedious, so we only provide the critical steps.

Suppose $\lambda_j^i = (\underline{\lambda_j^i}, \overline{\lambda_j^i})$ is a *positive interval* of $G^i(m)$. That is, there exist some non-empty set $A_j^i \subseteq \{1, 2, \cdots, n_i\}$, such that $\forall k, m : k \in A_j^i, m \in \lambda_j^i, H_k^i(m) > 0$. On λ_j^i, the first derivative of $G^i(m)$ with respect to m is

$$\frac{\mathrm{d}G^i(m)}{\mathrm{d}m} = \sum_{k \in A_j^i} a_k^i \cdot \left(\frac{1}{m - l_k^i} - \sum_{s=1}^{k} \frac{1}{u_s^i - m}\right) \cdot (m - l_k^i) \cdot \prod_{j=1}^{k} (u_j^i - m) \tag{21}$$

where $a_k^i = \frac{2}{\prod_{j=1}^k (u_j^i - l_j^i)}$. The second derivative of $G^i(m)$ with the respect to m is

$$\frac{\mathrm{d}^2G^i(m)}{\mathrm{d}m^2} = \sum_{k \in A_j^i} a_k^i \cdot \Big(\sum_{s=1}^{k} \frac{m - l_k^i}{u_s^i - m} \cdot \big(-\frac{2}{m - l_k^i} + \sum_{1 \le t \le k, t \ne s} \frac{1}{u_t^i - m}\big) \cdot \prod_{j=1}^{k} (u_j^i - m)\Big) \tag{22}$$

To prove that $G^i(m)$ is a strictly concave function, we need to show that the second derivative of $G^i(m)$ is negative. For this, we only need to consider the items $\sum_{1 \le t \le k, t \ne s} \frac{1}{u_t^i - m}$ and $\frac{2}{m - l_k^i}$ in Eqn. (22), because all other items in (22) are positive due to the fact that λ_j^i is a positive interval.

We expand λ_j^i to $(0, \overline{\lambda_j^i})$, then $\sum_{1 \le t \le k, t \ne s} \frac{1}{u_t^i - m}$ is an increasing function of m from $\sum_{1 \le t \le k, t \ne s} \frac{1}{u_t^i}$ to $\sum_{1 \le t \le k, t \ne s} \frac{1}{u_t^i - \overline{\lambda_j^i}}$, while $\frac{2}{m - l_k^i}$ is a decreasing function[3] of m from $+\infty$ to $\frac{2}{\overline{\lambda_j^i} - l_k^i}$, there must be a point m_c such that when $m < m_c$, $\frac{\mathrm{d}^2G^i(m)}{\mathrm{d}m^2} < 0$.

Now we prove that, $m_c \ge \overline{\lambda_j^i}$. To ease notation, we define a function $f(x)$ on domain $(b, \min_{i=1}^n a_i)$, where $b < \min_{i=1}^n a_i$, as

$$f(x) = -\frac{2}{x - b} + \sum_{i=1}^{n} \frac{1}{a_i - x} \tag{23}$$

It is easy to prove that there exits a point x_c that $f(x) < 0$ when $x < x_c$, and $f(x) > 0$ when $x > x_c$, and x_c will reach its minimum value when both $a_i (1 \le i \le n)$ and b achieve their minimum values.

Based on the above analysis, for (22), m_c will reach its minimum value when $\forall k : k \in A_j^i, u_k^i = \overline{\lambda_j^i}$ and $l_k^i = 0$. To compute the

[3]In Eqn. (22), $\forall k : k \in A_j^i$, if $m < l_k^i$, $m - l_k^i = 0$.

minimum value of m_c, we rewrite $G^i(m)$ by

$$\tilde{G}^i(m) = 2 \cdot (\overline{\lambda^i_j} - m) \cdot \left(1 - \frac{(\overline{\lambda^i_j} - m)^{|A^i_j|}}{(\overline{\lambda^i_j})^{|A^i_j|}}\right) \tag{24}$$

where $|A^i_j|$ is the size of A^i_j. Then the second derivative of $\tilde{G}^i(m)$ with respect to m is

$$\frac{\mathrm{d}^2\tilde{G}^i(m)}{\mathrm{d}m^2} = -2 \cdot |A^i_j| \cdot (|A^i_j| + 1) \cdot \frac{(\overline{\lambda^i_j} - m)^{|A^i_j|-1}}{(\overline{\lambda^i_j})^{|A^i_j|}} \tag{25}$$

According to (25), if $m < \overline{\lambda^i_j}$, $\frac{\mathrm{d}^2\tilde{G}^i(m)}{\mathrm{d}m^2} < 0$. Therefore, $m_c \geq \overline{\lambda^i_j}$. □

Proof of Proposition 2

Proof: Case 1: $b^i_k \leq y^i_{k-1}$. If the η value in Alg. 1 is sufficiently large, the algorithm will keep probing until a payment reaches b^i_k. In this case, the value of b^i_k can be inferred.

Case 2: $b^i_k > y^i_{k-1}$. In this case, $I^i_k = [y^i_{k-1}, u^i_k]$ and $|I^i_k| = u^i_k - y^i_{k-1}$. Therefore, the expected length of interval $|I^i_k|$ is

$$\begin{aligned} \mathbb{E}(|I^i_k|) =& Prob(b^i_k > y^i_{k-1}) \cdot (u^i_k - y^i_{k-1}) \\ =& \frac{(u^i_k - y^i_{k-1})^2}{u^i_k - l^i_k} \end{aligned} \tag{26}$$

According to Eqn. (26), $\mathbb{E}(|I^i_k|)$ is inversely proportional to y^i_{k-1}. As $y^i_1 \geq y^i_2 \geq \cdots \geq y^i_{n_i}$, we have that $\mathbb{E}(|I^i_k|)$ is proportional to k.□