Topology Routability in Payment Channel Networks

Ben Weintraub
Northeastern University

Matthew Jagielski
Northeastern University

November 5, 2019

Abstract

Payment Channel Networks (PCNs) are a promising method for improving cryptocurrency throughput and privacy. However, their longterm stability is not well understood. Transactions in a PCN are routed along paths that have links with sufficient credit, but unlike other transactional networks these links decrease in value with every transaction that is sent across. This depletion in credit eventually makes the links unusable for future transactions. In this paper we propose a metric for *routability*, a measure of a topology's fitness for having a known set of transactions routed through it. We then build on that metric by proposing an algorithm for predicting routability without simulating a set of transactions. Using these properties, we show that PCNs with the desired routability and transaction success rate can be generated, and we provide algorithms for doing so.

1 Modeling Data Sets

In creating a dataset, there are a few concepts that we want to be able to model. The first is the topology. According to [1], the Lightning Network can be characterized as both scale-free and small-world. Both of these network types have various parameters that can be adjusted, and experiments will need to be done to figure out both what is optimal and what is realistic–though these may not be the same. In fact, it's not clear that these are even the two best topologies for routing in a PCN. This is an interesting line for future research.

The second concept to model is the distribution of transactions. In a PCN there are likely to be a wide range of transaction values, but what the distribution of the transaction is can only be speculated. Intuitively, it would seem that there would be many small to medium sized payments and fewer very large payments. An exponential or powerlaw distribution might model this well. This could be an interesting area of measurement in the Lightning Network or even non PCNs, like Bitcoin.

Another aspect that should be modeled is the distribution of transacting pairs. It seems most likely that a larger proportion of transactions would occur between nodes that are close rather than far (in terms of the number of hops). A payment that has to travel accross many links will incur great transactions fees. This may be acceptable as one-time costs, but eventually the transaction fees will outweigh the cost of setting up a direct channel. This suggests that frequent transactions will be closer. Intuitively, we will assume that the probability of a transaction is inversely proportional to its length, formally stated later in Assumption 1.

The last concept we wish to model is for the routing algorithm used to bootstrap linkweights. In order to get the success ratio of 1 as predicted by theorem 1, the same deterministic routing algorithm must be used. Any variation from that will most likely cause at least some of the transactions to fail. Algorithm 1 operates on an unweighted graph, so the natural choice for a routing algorithm here would be a shortest path algorithm.

2 Graph Generation

Here, we consider the payment channel weight assignment problem. In this problem, there is a payment network, which is some weighted directed graph $G_0 = (V = \{u\}, E = \{(u, v, w)\})$. In a payment network, a transaction can occur between a source u, a destination v, and with a weight w—the full transaction t is written as (u, v, w). A routing algorithm A, which can return multipaths and may be randomized, takes as input the graph G and a transaction t, and returns $R, G' = A(t, G)$, where the route R is the set of all links which have been modified, and G' is the graph after all links are modified. If no route is possible (not enough weight is on links connecting u to v), A returns $\perp, G$, as G is not modified. Forming G' is equivalent to, for each hop $(a, b, m) \in R$, either m weight is decreased from the link (a, b) in G or m weight is added to the link (b, a).

The basic form of the routing problem we consider takes a list of transactions $\mathcal{T} = [t_i = (u_i, v_i, w_i)]_{i=1}^{n}$, a routing algorithm A, and G_0 and asks for a weight assignment for G_0 such that all of $\mathcal{T}$ is routable. That is, we want to set G_0's weights to never have A return $\perp$—many settings of G's weights w are not amenable to carrying all of the transactions in $\mathcal{T}$. We begin by considering algorithms for assigning these weights to ensure high *transaction completion rate*, defined below to formalize the intuitive desire to complete as many transactions as possible.

Definition 2.1 (Transaction Completion Rate)**.** For a graph G, routing algorithm A, and transaction set $\mathcal{T}$ consisting of n transactions, the *transaction completion rate* of A for $\mathcal{T}$ on G_0 is

$$TCR(G_0, A, \mathcal{T}) = \frac{1}{n}|\{R_i \neq \perp | t_i \in \mathcal{T}, (R_i, G_i) = A(t_i, G_{i-1})\}|.$$

2.1 Full Knowledge

In the full knowledge setting, we assume perfect knowledge of G_0, A, and $\mathcal{T}$. We show in Algorithm 1 how to assign edge weights for this setting.

2.1.1 Algorithm 1

Track maximum edge weight in each direction for each link, that's that!

Algorithm 1 Full knowledge algorithm for determining sufficient edge weights for routability of 1.

Input: Directed graph $G = (V = \{u\}, E = \{(u, v)\})$, Transaction list $\mathcal{T} = [(u_i, v_i, w_i)]_{i=1}^{n}$, Routing Algorithm A
$\textsc{MaxWeight} = \{\}$
$\textsc{CurWeight} = \{\}$
for $u, v \in E$ **do**
    $\textsc{MaxWeight} = \{(u, v) : 0\}$
    $\textsc{CurWeight} = \{(u, v) : 0\}$
for $t_i = (u_i, v_i, w_i) \in \mathcal{T}$ **do**
    $R_i, G_i = A(t_i, G_{i-1})$
    for $(a, b, m) \in R_i$ **do**
        $\textsc{CurWeight}[a, b] += m$ ▷ Apply transaction
        if $(b, a) \in \textsc{CurWeight}$ **then** ▷ If this canceled weight out
            $\textsc{MinWt} = \min(\textsc{CurWeight}[b, a], \textsc{CurWeight}[a, b])$
            $\textsc{CurWeight}[b, a] -= \textsc{MinWt}$
            $\textsc{CurWeight}[a, b] -= \textsc{MinWt}$
        $\textsc{MaxWeight}[a, b] = \max(\textsc{MaxWeight}[a, b], \textsc{CurWeight}[a, b])$
return MaxWeight

2.2 Analysis

Theorem 1. *Algorithm 1 is guaranteed to produce an assignment of weights allowing for a transaction completion rate of 1.*

3 Transaction Distribution

Having full knowledge of the transaction sequence is an unrealistic assumption. A more natural assumption would be that transactions are sampled from some distribution. A distribution over transactions allows us to analyze average-case behavior of the payment network; this requires some amendments to our definitions: $\mathcal{T}_\mathcal{D}$ is now transaction distribution over $V \times V \times \mathbb{Z}$. Recall the informal claim made in section 1 stating that transactions are more likely to occur between nodes which are close to each other.

Assumption 1. $\forall u, v_1, v_2 \in V, d(u, v_1) < d(u, v_2) \iff P_{\mathcal{T}_\mathcal{D}}[(u, v_1, \cdot)] > P_{\mathcal{T}_\mathcal{D}}[(u, v_2, \cdot)]$

While Assumption 1 only makes a statement about the ordering of probabilities, we will need to specify concrete probabilities. Some natural distributions to use here specify an inverse polynomial (e.g. $\propto 1/d(u, v)^c$) or inverse exponential probability (e.g. $\propto \exp(-cd(u, v))$).

Definition 3.1 (Distributional Transaction Completion Rate)**.** For a graph G, routing algorithm A, and transaction distribution $\mathcal{T}_\mathcal{D}$, from which n transactions are drawn, the *transaction completion rate* of A for $\mathcal{T}$ on G_0 is

$$TCR(G_0, A, \mathcal{T}_\mathcal{D}) = \mathbb{E}\left[\frac{1}{n}|\{R_i \neq \perp | t_i \in \mathcal{T}, (R_i, G_i) = A(t_i, G_{i-1})\}|\right].$$

That is, we sample n sequential transactions and observe the fraction which complete.

4 Expected Routability

Transaction completion rate is a useful metric, but it requires fully simulating a set of transactions. This may be a slow process depending on the routing algorithm and the size of the network. To account for this, we provide an alternative metric, *expected routability* which we measure on a directed graph $G_0 = (V = \{u\}, E = \{(u, v, w)\})$. The intuition behind this metric is that transactions which have their source and sink within a single connected component will have a much higher likelihood of completion than a source and sink that are unconnected.

Algorithm 2 Naive algorithm for determining a routability metric where all possible pairs of sources and sinks are enumerated. The routing algorithm A used should be one that finds paths in a directed, unweighted graph.

Input: Directed graph $G = (V = \{u\}, E = \{(u, v)\})$, Routing Algorithm A, Transaction distribution $\mathcal{T}_\mathcal{D} = [(u_i, v_i)]_{i=1}^n$
$CC = \textsc{CalculateConnectedComps}(G)$
$P = 0$
for $(u_i, v_i), p_i \in \mathcal{T}_\mathcal{D}$ **do** ▷ p_i is the probability of the pair being selected
  if $\textsc{GetConnectedComp}(u_i, CC) = \textsc{GetConnectedComp}(v_i, CC)$ **then**
    $P += p_i$
return P

This is an exhaustive algorithm and may not be suitible for large networks, but there are several ways it can be sped up. Instead of iterating through every pair in the graph, it may be sufficient to select n pairs from $\mathcal{T}_\mathcal{D}$. There also may be some optimizations related to calculating the connected components in

repeated runs of this algorithm; in this case, it may be possible it track the evolution of the connected components on the fly instead of recalculating them everytime.

It would be interesting to measure *expected routability* at regular timesteps during a full simulation run to see how it changes over time.

References

[1] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. *arXiv preprint arXiv:1904.10253*, 2019.