Lightning Network: Pathfinding Algorithms

1 Core-Lightning

- Standard implementation in C
- Blocks per year = 52596 = $\frac{365.25*24*60}{10}$ ($\approx$ 6 blocks per hour)
- Price in risk as risk factor percent per year

1.1 Order of functions invoked in pathfinding algorithm:

- amount_msat_add_fee
- risk_price
  - amount_msat_scale
- path_score

Based on the above function calls, let's derive the cost function for CLN. amount_msat_add_fee function adds fee to the amount (to be sent from current node to destination).

$$totalamount = amount + fee \tag{1}$$

$$fee = fee\_base + amount * fee\_proportional \tag{2}$$

Therefore,

$$totalamount = amount + fee\_base + amount * fee\_proportional \tag{3}$$

From risk_price function:

$$risk = \frac{total\ amount * riskfactor * timelock}{Blocks\ per\ year * 100} \tag{4}$$

Therefore,

$$risk = \frac{(amount + fee) * riskfactor * timelock}{Blocks\ per\ year * 100} \tag{5}$$

We get Cost from path_score function as below:

$$Cost = distancebias + risk + totalamount \tag{6}$$

From equations (3) and (5) in (6), we get,

$$Cost = (amount + fee) * (1 + \frac{riskfactor * timelock}{Blocks\ per\ year * 100}) + distancebias \tag{7}$$

where, bias is the path length.

2 Rust Lightning (LDK)

- Pathfinding is not a faithful Dijkstra's implementation because values impacting earlier nodes can be changed while processing later nodes.
- The algorithm ignores htlc_minimum_msat limit path at first. But, it will try it later we wish to pay htlc_minimum with extra fees.
- Finds path for exact amount at first. If path is found, the algorithm immediately return the path. Otherwise, searches for paths until it has 3x the amount in total across the path then select the best subset at the end.

The algorithm checks the below parameters so that they are under certain maximum limit.

- path length
- Timelock (CLTV delta limit)
  - Timelock = CLTV * Fee + Amount
- Value Contribution : The maximum value a yet-to-be-constructed payment path might flow through the node.i.e how much is needed for a path being constructed, how much value can channels following the node (up to the destination) can contribute considering their capacity and fees.

Cost function is as below:

$$Cost = max(fee,\ path\ htlc\ minimum) + Penalty \tag{8}$$

$$path\ htlc\ minimum = max(Next\ hop\ htlc\ minimum + fee,\ htlc\ minimum) \tag{9}$$

Penalty is the path penalty, this is fixed penalty added to channel.

$$Penalty = Base\ Penalty + \frac{Multiplier * Amount}{2^{30}} \tag{10}$$

where, Base Penalty is a constant with defualt value 500 msat.
Multiplier is a constant with default value 8,192 msat.

3 LND

- LND considers the history of payment attempts made on the route or channel.
- Uses Dijkstra's algorithm for path finding.
- As per LND documentation the cost is given as below:

$$Cost = base\_fee + fee\_rate * amt + (attemptcost + attemptcostppm * amt)/probability \tag{11}$$

- Let us derive this equation from the implemented algorithm:
- Probability based cost is given as follows:

$$Cost = weight + \frac{penalty}{probability} \tag{12}$$

- It is enough to just compare $F + \frac{c}{P}$ between two routes.

  Consider an example with two routes A&B with fees F_a and F_b and success probabilities P_a and P_b.

  Cost of trying A first and then $B = P_a * F_a + (1 - P_a) * P_b * (c + F_b)$
  Cost of trying B first and then $A = P_b * F_b + (1 - P_b) * P_a * (c + F_a)$

  On comparing, we get the term that differentiates as $F + \frac{c}{P}$, where c is the virtual cost of failed payment attempt. It is the penalty or absolute attemptcost given as below:

$$Absoluteattemptcost = Defaultattemptcost * \frac{1}{0.5 - TimePref/2} - 1 \tag{13}$$

  - attemptcost = 100 // configured
  - attemptcostppm = 1000 // configured
  - $Defaultattemptcost = attemptcost + \frac{amt*attemptcostppm}{1000000}$
  - Timepref *=0.9
- Weight of an edge:

$$weight = fee + timelock\_penalty \tag{14}$$

$$timelock\_penalty = amount\_to\_recieve * timelockdelta * risk\_factor \tag{15}$$

Therefore,

$$weight = fee + amount\_to\_recieve * timelockdelta * risk\_factor \tag{16}$$

Using (16) in (12),

$$Cost = fee + amount\_to\_recieve * timelockdelta * risk\_factor + \frac{penalty}{probability} \tag{17}$$

Where, risk_factor = $15 * 10^{-9}$

- Weight of each channel is composed of 3 parts: fee, CLTV_Delta, and probability.
- Probability that this channel can afford the payments from a local perspective.
- If the route fails, then the node with insufficient capacity will actively include this information in the return message which can only be unlocked and seen by the payer. Then for example, the payer will reduce the probability of this channel in the local view and find the route again. (Treat it as setting the weight of this edge to very large)
- Retry mechanism: LND's mission control subsystem makes payment attempts and then tries to attribute failure to one or both nodes along the route, depending on the error we get.
- In case of insufficient balance, that means a node 'N' doesn't have outgoing capacity. So, attribute failure to N alone.
- Success probability is introduced in addition to cost to have reliability.
- Success probability P(ABC) = P(AB)*P(BC)
- Explore vs exploit: choose a route that we know will likely be successful (but might be costly) or try a new lower cost route that we are not sure about. Historical payment results:
- Just look at the last payment result for a channel. For failures-> change probability and consider the amount.
- We can slowly forget the failure. We can configure the forget rate through penaltyhalflife (Ex: 1hr).
- Apriori probability:
- What is the success probability of a route that we know nothing about?
- This is configurable based on confidence. Constant value per channel is assigned reflecting the current state of the network.
- Calculate success probability based on the weight of the last failure.
- When the failure is fresh, its weight is 1 and return probability 0. Over time the probability recovers to the node probability. It would be as if this channel was never tried before.
- $probability = nodeprobability * (1 - weight)$
- $exp = \frac{-age}{PenaltyHalfLife}$
- $weight = 2^{exp} = 2^{\frac{-age}{PenaltyHalfLife}}$
- Therefore, $Probability = nodeprobability * (1 - \frac{1}{2^{-exp}})$
- Node probability is derived as below:
- Apriori hop probability has the default value 0.6.
- Apriori hop probability is reduced if the amount comes closer to the channel capacity.

- $Apriori = Apriori\ hop\ probability * Capacity\ factor$
- Capacity factor is computed using the amount and capacity of the channel.
- $aprioriFactor = \frac{1}{(1-AprioriWeight)-1}$
- AprioriWeight has the default value of 0.5.
- $probabilitiesTotal = apriori * aprioriFactor$
- $totalweight = aprioriFactor$
- $Nodeprobability = \frac{probabilitiesTotal}{aprioriFactor}$

4 Eclair

- Uses Yen's algorithm to find k-shortest paths.
- Randomly selects one path out of k-shortest paths.
- Uses heuristics to calculate the weight of an edge based on channel age, cltv delta, capacity, and virtual hop cost.
- This algorithm favors older channels with bigger capacity and small cltv delta.
- Two things to note:
  - Weight Ratios: This comprises of the following:
    - Base factor
    - Age factor
    - Capacity factor
    - Hop cost
    - Cltv delta factor
  - Heuristics constants: This comprises of the following:
    - Locked fund risk (cost of having funds locked in htlc)
    - Failure cost (fee for a failed attempt)
    - Hop cost (virtual fee per hop)
- Weight/ cost calculations:
- $Fee = Total\ amount - Previous\ amount$
- $Total\ fees = Previous\ fee + Fee$
- $Total\ cltv = Previous\ cltv + cltv$
- Weights calculation is divided into two cases:
- **Case 1**: When weight ratios (will refer as WR) is passed/available to get edge weight.
- Every edge is weighted by channel capacity, block height, cltv delta.
- Less weight older block or large channel
- $Normalized\ Capacity\ factor = 1 - normalized(maximum\ edge\ capacity)$
- $Normalized\ age\ factor = normalized(block\ height)$
- $Normalized\ cltv\ factor = normalized(edge\ cltv\ expiry\ delta)$
- $Factor = WR\_basefactor + (Normalized\ cltv\ factor * WR\_cltv\ delta\ factor) + (Normalized\ age\ factor * WR\_age\ factor) + (Normalized\ Capacity\ factor * WR\_capacity\ factor)$
- $Total\ Weight = Previous\ weight + (fee + hop\ cost) * Factor$ (Note: Current edge weight = (fee + hop cost) * Factor)
- Note that weight ratios: cltv delta factor + capacity factor + age factor + base factor = 1
- **Case 2**: When Heuristics constants (will refer as HC) are passed/available to get edge weight.
- $Total\ hop\ cost = Previous\ virtual\ fees + hop\ cost$
- $Success\ probability = 1 - (\frac{previous\ amount}{capacity})$

- $Total\ success\ probability = Previous\ success\ probability * success\ probability$
- If log probability is to be used:
- $Risk\ cost = Total\ amount * cltv * HC\_locked\ fund\ risk$
- $Total\ Weight = Previous\ weight + fee + hop\ cost + Risk\ cost - HC\_failure\ cost * log(success\ probability)$
- Otherwise:
- $Total\ risk\ cost = Total\ amount * Total\ cltv * HC\_locked\ fund\ risk$
- $Total\ \ Weight\ \ =\ \ Total\ \ fees\ +\ Total\ \ hop\ \ cost\ +\ Total\ \ risk\ \ cost\ +\ (\frac{HC\_failure\ cost}{Total\ success\ probability})$

Note: "Previous" in previous weight is the weight of rest of the path so far. In the code, Previous is a class (RichWeight) consisting of objects - amount, length, cltv, success probability, fees, virtual fees, weight.

Every time when we compute total weight after adding an edge to the path, we update the class RichWeight (Previous).

Previous amount is the total amount received by the current edge/node. Payment amount is fixed however, Eclair adds edge fees to the payment amount every time when a new edge is added. Ex: For 1st node, Total amount = Payment amount.
Previous amount = Total amount
2nd node, Total amount = Previous amount + 1st edge fee
Previous amount = Total amount
3rd node, Total amount = Previous amount + 2nd edge fee
And so on

Design Elements

LND

- Risk Factor:

  In LND, the parameter 'Risk Factor' is used to control the influence of time lock delta of a channel on route selection. Essentially, there may be several channels between the sender and the recipient when a payment is sent over the Lightning Network. The routing algorithm assesses various paths and chooses the one with the shortest predicted time or that is most cost-effective.

  'Risk Factor' parameter plays an important role in this decision-making process. Risk Factor is chosen based on the formula: 1 + timelock + fee * fee. This formula includes the fee penalty and time lock penalty. The main purpose of 'Risk Factor' is to balance these penalties. Fee penalty reduces the time lock penalty for larger amounts and increases it for smaller amounts. This is done to encourage the selection of paths with smaller time lock values while still considering fee. The default value of Risk Factor is 15 * 10^{-9}, which is relatively small to avoid drastic changes in path finding behavior but still significant enough to influence routing decision as it is big enough to give some effect with smaller time lock values.

- Attempt cost:

  In LND, the pathfinding algorithm aims to find a balance between the likelihood that the route succeeds, and the cost of sending through that route. Some routes might have a higher likelihood of success but also come with higher fees, while others might be cheaper but have a higher likelihood of failure. The trade-off can be tuned by the attempt cost proportionality factors and is expressed in terms of fees.

  Attempt Cost is a fixed value representing the cost of a failed payment attempt, regardless of the actual amount sent. This parameter allows the Path finding algorithm to trade off potentially better routes against their success probability which means it introduces a penalty for routes that have high probability of failure.

  During the evaluation of multiple paths determined by the routing algorithm, the algorithm considers both the total cost of sending through that route and the Attempt Cost associated

with each path. It can be inferred from this that LND's pathfinding algorithm does not blindly select the cheapest route but also the likelihood of failure associated with that route is considered.

When the Attempt Cost factors are set to higher values, it means that the routing algorithm prioritizes routes that have a lower virtual cost, i.e., the routes with higher likelihood of success. However, these routes may come with higher fees. Conversely, when the Attempt Costs are set to zero, the algorithm will optimize for cheapest fees without considering likelihood of success i.e., the algorithm may select routes with lower fees even if they have a higher risk of failure.

The default value of Attempt Cost is 100 and Attempt Cost PPM is 1000. (Attempt Cost PPM is the proportional virtual cost in path finding of failed payment attempt)

- Apriori Estimator:

  Apriori, meaning "from the previous," is the original metric used to determine the probability of a successful payment through a given route. Apriori probability calculations in the LND are based on assumptions and historical information about routing nodes and channels. These calculations guide payment routing decisions by estimating the likelihood of success for different payment routes.

  Apriori probability calculation starts with the assumption that each hop in a payment route has a base success likelihood of 60%. This value is considered the default and can be adjusted. If a previous payment through a hop was successful, any new and smaller payment through that same hop is expected to succeed with a higher probability of 95%. This assumption is based on the idea that if a hop has successfully forwarded a payment before, it's likely to do so again for smaller payments. Conversely, if a previous payment through a hop was unsuccessful, any payment larger than the previous one is considered impossible to route through that hop. This assumption reflects the notion that if a hop has failed to forward a payment once, it may not have enough capacity or liquidity to handle larger payments. As the payment size approaches the capacity of a channel, the probability of it succeeding diminishes. The idea here is attempting to send a payment close to or exceeding the channel's capacity is more likely to result in failure.

  Over time, the failure probabilities will converge back to 60% as the balance is expected to change over time. Channels are assumed to keep a high success probability until proven otherwise.

  Previous payment attempts for a node are used to calculate a reputational score. This score can extend the a priori probability and be used to focus on channels associated with well-managed routing nodes. The assumption is that nodes with a good track record of successfully forwarding payments are more likely to have liquid and reliable channels.

Eclair

-