Coordinated Landmark-based Routing for Blockchain Offline Channels

Subhasis Thakur
National University of Ireland, Galway
Galway, Ireland
subhasis.thakur@nuigalway.ie

John G. Breslin
National University of Ireland, Galway
Galway, Ireland
john.breslin@nuigalway.ie

***Abstract*—Blockchain offline channels such as the Bitcoin Lightning Network will improve the scalability of blockchains by reducing the number of transactions needed to be recorded in the blockchain. Uncoordinated payment transfer in channel networks including path-based fund transfer may overuse few channels. This will lead to imbalanced channel networks where the channel balance of few channels is too low to remain operational. In this paper, we propose a coordination method for a landmark-based routing algorithm for fund transfer in offline channels. Our procedure allows the landmarks to route funds in complementary and non-overlapping paths which balances channel values in a bi-directional channel network. Using experimental evaluation with Bitcoin Lightning network data we prove that the proposed coordinated landmark-based routing algorithm keeps a better balance of the channels and significantly improves the success rate of fund transfer compared with existing landmark-based routing algorithms.**

***Index Terms*—Blockchains, Offline channels, Routing**

I. Introduction

Offline channels may improve the scalability of public blockchains by reducing the number of transactions needed to be recorded in the blockchain. Paths in the offline channel network can be used for fund transfer among peers without mutual channels. State of art routing algorithms for fund transfer in offline channels use the landmark-based [1], [2] protocols. In a bi-directional channel network such as Bitcoin Lightning Network, there are two channels between any pair of peers. Each channel's value indicates the amount of funds one peer can transfer to the other peer. For every transfer in an offline channel, the decrement in the value of one channel is equal to the increment of the value of the opposite channel. For example, in a channel (with an initial balance of 10 tokens where A can transfer 5 tokens to B and B can transfer 5 tokens to A) between A and B a sequence of 5 transfers from A to B (where the amount of tokens in each transfer be 1 token) will lead to the value of the channel $A \to B$ becoming 0 and the same for the channel $B \to A$ will become 10. Hence A can not further use this channel as it does not have any fund to transfer to B. Thus this channel will be closed as a new transaction will be created in the blockchain.

This project is funded by the Department of Business, Enterprise and Innovation, under the Government of Ireland's Project 2040 Plan and Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund and Science Foundation Ireland (SFI). This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and the Department of Agriculture, Food and Marine on behalf of the Government of Ireland under Grant Number [16/RC/3835].

If channels get imbalanced [3] quickly then it becomes non-operational and new transactions will be frequently created in the blockchain to close such channels. This will decrease the scalability of blockchains. Further, paths may be used in Path-based Fund Transfer (PBT) in the offline channel network which uses paths between two peers for fund transfer. Similar to individual channels, over-usage of the same path will lead make the channel in that path non-operational. Although such a landmark-based routing method can be efficient in terms of the path length of paths used in PBT execution, it does not solve the balancing problem of channels. The state of art routing algorithms for PBT uses landmark-based routing protocols [1], [2]. In this paper, we extend such landmark-based algorithms for balancing channel network.

We developed a method of coordination which the landmarks can use to form their respective rooted trees. The coordination procedure can be executed without any message exchange among the landmarks. This eliminates the problems with asynchronous communications among the landmarks. We evaluated the proposed coordinated landmark-based routing protocol using PBT execution simulations on Bitcoin Lightning network data. We found that the proposed routing algorithm keeps the channel network balanced by preventing over-usage of channels and the success rate of PBT executions is improved significantly. The paper is organized as follows: in section 2 we discuss related literature, in section 3 we explain blockchain offline channels and landmark-based routing procedure, in section 4 we present the coordinated landmark-based routing protocol, in section 5 we evaluate the proposed routing protocol, and we conclude the paper in section 6.

II. Related Literature

Proof of work-based blockchains was proposed in [4]. There are several variations of blockchains in terms of consensus protocols. Applications of these various types of blockchains are in various application areas such as energy trade [5], IoT service composition [6], etc. Bitcoin lightning network was proposed in [7] which allows peers to create and transfer funds among them without frequently updating the blockchain. Similar networks were proposed for Ethereum [8] and credit networks [2]. A privacy-preserving payment method in the credit network was proposed in [9]. A routing algorithm for the Bitcoin lightning network was proposed in [10]. A method for anonymous payment to improve privacy in PBT was developed in [11]. Authors in [12] developed a tree-based routing algorithm for credit networks. [13] proposed an offline payment network using trusted execution environments. [2] proposed a landmark-based routing protocol for fund transfer in a credit network. [1] enhanced the landmark-based routing algorithm developed in [2] by reducing the path length for PBT execution. We advance the state of the art by developing a coordination method to be used in landmark-based fund transfers to keep the channel network balanced.

III. Fund transfer in Blockchain Offline Channels

A. *Offline channels*

A protocol for using an offline channel (for Bitcoin Lightning network [7]) is as follows:

1) Offline channels use Hashed Time Locked Contracts[1] to create and update channels.
2) Say Alice and Bob want to create a channel between them with balances 10 tokens (each contributes 5 tokens).
3) Alice and Bob create two pairs of a lock (hash) and key (random string). They exchange the locks.
4) Bob creates a 'confirmation transaction' as follows:
    a) There is a multi-signature address between them which requires a signature from both to transfer fund from it. We will call this address M_1.
    b) Bob creates transactions from M_1 which states that Bob will get 5 tokens and the remaining 5 tokens will go to another multisignature address between them. We will call this address M_2.
    c) The 5 tokens in M_2 will be given to Alice after 10 days or Bob can claim it if it can produce the key to the lock of Alice.
5) Bob signs this transaction and sends it to Alice who can use it to get tokens from the channel by signing it and publishing it to the blockchain network.
6) Alice produces mirrored confirmation transaction and sends it to Bob. The confirmation transactions ensure that both parties can recover from if they fund the channel between them.
7) Now Alice and Bob transfers funds in the multi-signature address by creating transactions in the blockchain and hence the channel becomes operational.
8) Both parties should exchange keys and create new confirmation transaction to update the channel. They do not need to update the blockchain as the update confirmation transactions.
9) If any party announces a confirmation transaction then the channel closes.

Further, the offline channel network supports Path-Based Fund Transfer (PBT). A PBT uses a path between two parties in a channel network for fund transfer between them. A PBT protocol [7] for this offline channel network is as follows:

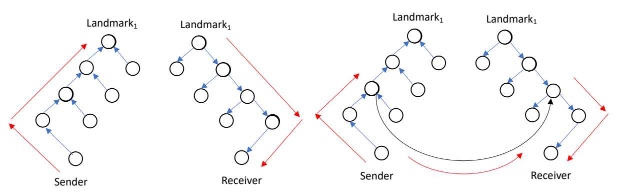

Fig. 1. A landmark-based routing protocol uses rooted trees to and from landmarks.The landmark-based routing protocol [1] (right figure) improves the algorithm [2] (left figure) by finding shortcuts in the path built through the trees rooted at the landmarks.

1) Say Alice wants to send funds to Carol via Bob.
2) Carol will create a lock and a key.
3) In the multi-signature address between Carol and Bob, a contract will be created as follows:
    a) Bob will send 5 tokens to this address.
    b) Bob will get these tokens back after 9 days if Carol does not claim it.
    c) Carol can claim it anytime if it can produce the key to the lock.
4) Similarly, another contract will be created between Alice and Bob as follows:
    a) Alice will send 5 tokens to this address.
    b) Alice will get these tokens back after 10 days if Bob does not claim it.
    c) Bob can claim it anytime if it can produce the key to the lock.
5) Thus Carol reveals the key to Bob as it collects the fund, which Bob uses to get refunded from Alice.

B. *Routing protocols for PBT execution*

There are two prominent landmark-based routing algorithms [1], [2]. Landmarks are set of high degree nodes and other non-landmark nodes depend on these landmark nodes to find paths. Each landmark creates tow rooted trees (with itself as the root). One tree for outgoing edges and one edge for incoming edges. First, the sender and the receiver agree on using a particular landmark. The sender finds a path to the landmark by using the rooted tree built with incoming edges. The receiver finds a path from the landmark to itself using the rooted tree built with out-going edges. The landmarks regularly probe the offline channel network and update its trees. The routing algorithm [1] improves the routing algorithm [2] by finding shorter paths built using rooted trees of landmarks.

IV. Fund transfer with coordination among landmarks

We developed a method that allows the landmarks to coordinate while they construct their respective rooted trees. Briefly, it is as follows:

[1]https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

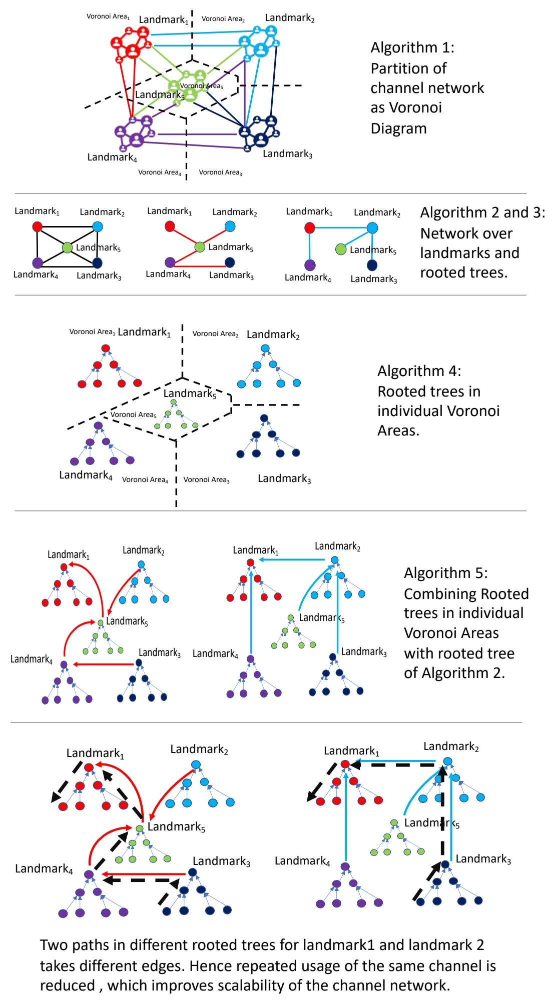
Voronoi Area1
Landmark1
Voronoi Area2
Landmark2
Voronoi Area5
Landmark5
Landmark4
Landmark3
Voronoi Area4
Voronoi Area3
Algorithm 1: Partition of channel network as Voronoi Diagram
Algorithm 2 and 3: Network over landmarks and rooted trees.
Algorithm 4: Rooted trees in individual Voronoi Areas.
Algorithm 5: Combining Rooted trees in individual Voronoi Areas with rooted tree of Algorithm 2.
Two paths in different rooted trees for landmark1 and landmark 2 takes different edges. Hence repeated usage of the same channel is reduced , which improves scalability of the channel network.

Fig. 2. Overview of solution: Algorithm 1 produces a Voronoi diagram over the channel with landmarks as the seeds. Algorithm 2 produces the network over the landmarks by creating edges between landmarks whose Voronoi areas share a boundary. Algorithm 3 generates rooted trees for each landmark in the network over the landmarks. Algorithm 4 generates rooted trees for each Voronoi area with the seed landmark as the root. Algorithm 5 connects these rooted trees according to the set of edges of the root trees among the landmarks produced by algorithm 3.

1) As shown in figure 2, the coordination method uses two types of rooted trees, one among the landmarks and another for each landmark over a partition of the channel network.
2) First we partition the channel network as a Voronoi diagram where one Voronoi area is centred around one landmark. Next, we form a network over the landmarks where two landmarks are neighbours if their corresponding Voronoi areas are adjacent.
3) Next, for each landmark we find its rooted tree in the network over the landmark.
4) Next, for each landmark we find its rooted tree for the peers in its Voronoi area. Now we combine these rooted trees according to the rooted tree over the landmarks.
5) If there is an edge from $Landmark_2$ to $Landmark_1$ in the rooted tree for $Landmark_1$ in the network over the landmarks, there we add edges from $Voronoi\_Area\_2$ (with $Landmark_2$ as the seed) to $Voronoi\_Area\_1$ (with $Landmark_1$ as the seed). Note that, in the rooted tree for $Landmark_2$ there will be an edge from $Landmark_1$ to $Landmark_2$. Hence in the rooted tree for $Landmark_2$ we will use edges from $Voronoi\_Area\_2$ to $Voronoi\_Area\_1$. Thus the rooted trees will have complementary edges. Hence paths proposed by the landmarks will not use the same channels and it will improve the balance of the channels.

A. Partitioning Channel Network as Voronoi Diagram

We will use Algorithm 1 to partition the channel network as a Voronoi diagram where each Voronoi area will be centred around one landmark.

1) It will be assumed that all landmarks know the offline channel network and identity (public key) of all landmarks.
2) Each landmark will execute this algorithm by itself and if their knowledge about the channel network is the same then they will produce the same Voronoi diagram as a partition of the channel network. Although our solution does not need the same partition of the channel network, it is needed that the set of landmarks who want to coordinate fund routing among themselves know exactly about their identity so that the correct number of partitions can be generated.
3) As shown in Algorithm 1, each landmark is a seed of the Voronoi diagram and a peer (who is not a landmark) is added to the Voronoi area of a landmark if its distance (number of edges) to the landmark is the shortest (if there are more than one such landmarks, the peer is added to the Voronoi area of any such landmark chosen uniformly at random).

B. Rooted Trees over Hubs

Next, each landmark will construct a graph over the landmarks using Algorithm 2.

1) Two landmarks are neighbours in this network if there is at least one edge in the channel network which begins from the Voronoi area of one landmark and ends on the Voronoi area of another landmark. Algorithm 2 describes this algorithm.

Now, each landmark will find a rooted tree in the network over landmarks as follows (using Algorithm 3):

1) In this algorithm produces the rooted tree for each landmark from the network $g1$ over the landmarks formed with Algorithm 2.
2) First, (line 3) an empty graph $g2$ is formed.
3) Next, neighbours of the landmark k (who is executing this algorithm) is recognised as the parent node.
4) All neighbours of each parent node are found and they were added to $g2$ if they have not been added before.

5) Next, newly added child nodes become the parent node for the next iteration of the algorithm. This process continues until all nodes of $g1$ is added to $g2$.

C. Rooted Trees for Each Voronoi Area

Next, a landmark finds rooted trees for each subgraph identified as one Voronoi area (shown in Algorithm 4).

1) In each Voronoi area the landmark acting as the seed of that Voronoi area is found and all other peers in the same Voronoi area are identified.
2) Next, two rooted trees are formed in the subgraph of the channel network only containing the peers in this Voronoi landmark area.
3) The landmark acting as the seed of this Voronoi area acts as the root of both trees. One such rooted tree t_{out} only contains outgoing edges from the root and another rooted tree t_{in} which only contains incoming edges of the root. The rooted tree formation process is similar to Algorithm 3.

D. Integrating the rooted Trees for Each Voronoi Area

Finally, a landmark combines all rooted trees found in Algorithm 4 to create a rooted tree over the entire channel network whee this landmark is the root. It is as follows (shown Algorithm 5):

1) A landmark will complete its rooted tree in the channel network by augmenting the partially formed rooted tree in the previous step with additional edges from its rooted tree in $g1$. In this procedure, edges among landmarks are found according to each edge over the rooted tree in $g1$. Note that we only show the outgoing rooted tree formation process. A similar procedure can be used for incoming rooted tree formation.

E. Execution of the coordinated routing protocol

The coordinated routing protocol execution is as follows:

- All landmarks who want to coordinate their routing processes form the group of landmarks and a member of this group knows all other landmarks.
- Each landmark executes Algorithm 1, 2, 3, 4 sequentially and independently to form its rooted tree over the entire channel networks.
- Next, a landmark waits for a peer's query to find a path to another peer. It uses its rooted trees to find such paths and informs the peer. PBT executed if corresponding channels agree to execute PBT and have sufficient balance to execute the PBT.
- Thus the only coordination action required for the landmarks is to form the group of landmarks.

V. Evaluation

We use Bitcoin Lightning network data from [14]. It used an API to access the Lightning network data. The downloaded data is in JSON format and the RJSONIO package was used to process the data. The data contains (a) information about each node, i.e., the public key, and (b) network structure as

Algorithm 1: Voronoi Diagram of the Channel network.

```
Data: $g = (V, E)$ as the channel network,
      $landmarks \subset V$ be the set of landmarks
Result: Voronoi Diagram of $g$.
begin
    $voronoi \leftarrow |landmarks| \times |V|$
    $voronoi[, 1] \leftarrow landmarks$
    $added \leftarrow landmarks$
    for $d_1 \in [1 : Diameter(g)]$ do
        $n_1 \leftarrow \mathcal{N}(g, d1, landmarks, \text{“}out\text{”}) - added$
        if $|n_1| > 0$ then
            $d_x \leftarrow |n_1| \times |landmarks|$ (initially 10000)
            for $j \in [1 : |n_1|]$ do
                $p_1 \leftarrow \mathcal{P}(g, n_1[j], landmarks, \text{“}out\text{”})$
                for $k \in [1 : |landmarks|]$ do
                    $p_2 \leftarrow p_1[k]$
                    if $p_2 > 0$ then
                        $p_3 \leftarrow p_2[2 : |p_2|]$
                        if $|p_3| == |voronoi[k, ] \cap p_3|$ then
                            $d_x[j, k] \leftarrow |p_3|$
                if $min(d_x[j, ]) < 10000$ then
                    $id_1$ index of minimum element of $dx[j, ]$
                    $id_2$ index of $voronoi[id_1, ]$ which is 0
                    $voronoi[id_1, id_2] \leftarrow n_1[j]$
                    $added \leftarrow c(added, n_1[j])$
```

Algorithm 2: Algorithm to form a graph over the landmarks.

```
Data: $g = (V, E)$ as the channel network,
      $landmarks \subset V$ be the set of landmarks
Result: $g1$ as a graph over the landmarks
begin
    $g1 \leftarrow graph.empty(n = |landmarks|, directed)$
    for $i \in [1 : |landmarks|]$ do
        $x \leftarrow voronoi[i, which(voronoi[i, ] > 0)]$
        $n1 \leftarrow unlist(\mathcal{N}(g, 1, x, \text{“}out\text{”}))$
        for $j \in [1 : |landmarks|]$ do
            if $i \neq j$ then
                $y \leftarrow voronoi[j, which(voronoi[j, ] > 0)]$
                if $|n1 \cap y| > 0$ then
                    $g_1 \leftarrow add.edges(g1, c(i, j))$
                    $E(g1)[|E(g1)|]\$weight \leftarrow |n1 \cap y|$
```

Algorithm 3: Rooted trees for the landmarks over the network $g1$

```
Data: $g = (V, E)$ as the channel network,
      $landmarks \subset V$ be the set of landmarks, $g1$ be
      the network over the landmarks (algorithm 2).
Result: Rooted tree $g2$ for landmarks on $g1$
1  begin
2      $k \in landmarks$ be the landmark for which the
        rooted tree is being constructed
3      $g2 \leftarrow graph.empty(n = |V(g1)|, directed)$,
        $x \leftarrow k$, $added = x$, $flag1 = TRUE$
4      while $flag1 == TRUE$ do
5          $y \leftarrow \emptyset$
6          for $i \in [1 : |x|]$ do
7              $x1 \leftarrow x[i]$, $n1 \leftarrow \mathcal{N}(g1, x1, \text{“}out\text{”})$,
                $n2 \leftarrow n1 - added$
8              if $|n2| > 0$ then
9                  $dis$ is $|n2|$ length array initially 0
10                 for $j \in [1 : |n2|]$ do
11                     $dis[j]$ is weight of the edge
                        $(x[i], n2[j])$ in $g1$
12                 $or_1$ is decreasing order $dis$
13                 if $|n2| > 1$ then
14                     $add.edges(g2, c(x[i], n2[or_1[1]]))$,
15                     $add.edges(g2, c(x[i], n2[or_1[2]]))$
16                     $added =$
                        $added \cup (n2[or_1[1]], n2[or_1[2]])$,
17                     $y = y \cup (n2[or_1[1]], n2[or_1[2]])$
18                 else
19                     $add.edges(g2, c(x[i], n2[or_1[1]]))$
20                     $added \leftarrow added \cup n2[or_1[1]]$
21                     $y \leftarrow y \cup n2[or_1[1]]$
22         $x \leftarrow y$
23         if $|y| == 0$ then
24             $flag1 = FALSE$
25     Return(g2)
```

the edge list. The data was accessed on 1st March 2020. We use a connected subgraph of this network. The network is as follows:

# of Nodes	# of Edges	Mean Degree	Min Degree	Max Degree
3070	16268	2	138	10

We identify nodes with more than 50 as the set of landmarks. There are 105 landmarks. We generate 8 subgraphs of the Bitcoin Lightning network with the following characteristics: all nodes have a degree at most 50 and:

# Nodes	490	405	397	819	658	650	524	569
# Edges	1166	952	966	2066	1624	1626	1330	1408

Algorithm 4: First part of the rooted trees for the landmarks over g.

```
Data: $g = (V, E)$ as the channel network,
      $landmarks \subset V$ be the set of landmarks,
      $hub\_edges$ as the rooted trees over the
      landmarks in $g1$
Result: First part of the rooted trees for the landmarks
        over $g$
1  begin
2      $t_{out} = graph.empty(|V(g)|, directed)$
3      $t_{in} = graph.empty(|V(g)|, directed)$
4      for $i \in [1 : |landmarks|]$ do
5          $v1 \leftarrow i$,
6          $vx \leftarrow landmarks[v1]$
7          $v' \leftarrow voronoi[v1, which(voronoi[v1, ] > 0)]$
8          $g' \leftarrow induced.subgraph(g, v')$
9          $t'_{out} \leftarrow get\_tree\_out(g', which(v' == vx))$
10         $t'_{in} \leftarrow get\_tree\_in(g', which(v' == vx))$
11         $E_{out} \leftarrow get.edgelist(t'_{out})$
12         $E_{in} \leftarrow get.edgelist(t'_{in})$
13         for $j \in c(1 : length(E_{out}[, 1]))$ do
14             Add edge $(v'[E_{out}[j, 1]], v'[E_{out}[j, 2]])$ to
                $t_{out}$
15         for $j \in c(1 : length(E_{in}[, 1]))$ do
16             Add edge $(v'[E_{in}[j, 1]], v'[E_{in}[j, 2]])$ to $t_{in}$
```

We simulate PBT execution in each of these 8 networks. We assume that the value of the channels is a number between $[0, 5]$ chosen uniformly at random and each PBT execution transfers .1 tokens. In each simulation execution, each peer attempts to execute one PBT. We assume that there are 15 landmarks in each simulation. Landmarks are chosen as nodes with a degree of more than the average degree of each network. We use agent-based modelling to simulate a blockchain and PBT executions. The algorithm for the simulation is as follows: We execute 4 processes in each iterated execution of the simulation.

1) First, each peer finds another peer for a fund transfer destination. A peer randomly chooses a landmark. It finds a path to the landmark and another path from the landmark to the destination of the fund transfer.
2) Second, after finding a path to the fund transfer destination, each peer requests the owners of the channels in such a path for its usage in the fund transfer.
3) Third, each peer checks if it has received a channel usage request if there is sufficient balance in its channel a peer will allow usage of its channel for a fund transfer. The peer informs the requesters about its decision on channel usage.
4) Fourth, each peer checks the messages received from other peers regarding their decisions on using their respective channels. If a peer is allowed to use all channels in the path to the fund transfer destination then

Algorithm 5: Second part of the rooted trees for the landmarks over g

```
Data: $g = (V, E)$ as the channel network,
      $landmarks \subset V$ be the set of landmarks,
      $hub\_edges$ as the rooted trees over the
      landmarks in $g1$
Result: Second part of the rooted trees for the
        landmarks over $g$
1  begin
2      $e \leftarrow hub\_edges[, ((2(v1-1))+1) : ((2 \times (v1-1))+2)]$
3      $t'_{out} \leftarrow t_{out}$
4      for $i \in [1 : |e[, 1]|]$ do
5          if $e[i, 1] \neq 0$ then
6              $x \leftarrow e[i, 1], y \leftarrow e[i, 2]$,
7              $lm_1 \leftarrow landmarks[x]$,
8              $lm_2 \leftarrow landmarks[y]$,
9              $v'_x \leftarrow voronoi[x, which(voronoi[x, ] > 0)]$,
10             $v'_y \leftarrow voronoi[y, which(voronoi[y, ] > 0)]$,
11             $v$" is decreasing order of $v'_x \cup v'_y$,
12             $g' \leftarrow induced.subgraph(g, v")$
13             $paths \leftarrow \mathcal{P}(g', which(v" == lm_1), which(v" == lm_2), \text{"out"})$
14             $e' \leftarrow \emptyset$
15             for $j \in c(1 : length(v'_x))$ do
16                 $n1 \leftarrow \mathcal{N}(g, v'_x[j], \text{"out"}) \cap v'_y$
17                 if $length(n1) > 0$ then
18                     $p1 \leftarrow \mathcal{P}(g, lm_1, v'_x[j], \text{"out"})$
19                     for $j1 \in [1 : |p1|]$ do
20                         if $j1 \neq |p1|$ then
21                             $e' \leftarrow e' \cup Eid(g, (p1[j1], p1[j1+1]))$
22                     $p1 \leftarrow \mathcal{P}(g, n1[1], lm_2, \text{"out"})$
23                     for $j1 \in [1 : |p1|]$ do
24                         if $j1 \neq |p1|$ then
25                             $e' \leftarrow e' \cup Eid(g, (p1[j1], p1[j1+1]))$
26             if $|e'| > 0$ then
27                 $e1 \leftarrow get.edgelist(g)$
28                 for $k \in c(1 : length(e'))$ do
29                     $t'_{out} \leftarrow add.edges(t'_{out}, e1[e'[k], ])$
30     return($t'_{out}$)
```

the PBT is executed. If PBT is executed in a path then channel balances of channels in that path are decreased by .1 tokens and the balance of dual path is increased by .1.

5) We assumed synchronous execution of the PBT execution simulation and we assume that all messages are received within a fixed finite time. Although asynchronous simulation may replicate actual PBT execution, the impact of delay in receiving messages will impact both PBT execution without coordination among landmarks and with coordination among landmarks.

We evaluate the performance of coordinated PBT execution in terms of the standard deviation of channel balances, success rate, and the failure rate of PBT executions. Channels are better balanced if the increase in the standard deviation of channel values remains low. This will mean the complimentary usage of channels or fewer overuse channels. Figure 3 shows the standard deviation of channel values with and without coordination among landmarks. It clearly shows that coordination among landmarks eventually leads to a lower standard deviation of channel values. Next, we analyse the failure and success rate of PBT execution with and without coordination among landmarks. As shown in figure 4 the failure rate of PBT execution with coordination among the landmarks is significantly lower than the same without coordination among the landmarks. Also, as shown in figure 5 the success rate of PBT execution with coordination among landmarks is significantly higher.

VI. Conclusion

In this paper, we proposed a coordinated landmark-based routing protocol for blockchain offline channels. Using experimental evaluation we proved that the proposed routing protocol keeps channels balanced by eliminating over-usage of channels and it significantly improves the success rate of fund transfer.

References

[1] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, “Settling payments fast and private: Efficient decentralized routing for path-based transactions,” *CoRR*, vol. abs/1709.05748, 2017. [Online]. Available: http://arxiv.org/abs/1709.05748

[2] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, “Silentwhispers: Enforcing security and privacy in decentralized credit networks,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 1054, 2016.

[3] S. Thakur and J. G. Breslin, “A balanced routing algorithm for blockchain offline channels using flocking,” in *Blockchain and Applications*, J. Prieto, A. K. Das, S. Ferretti, A. Pinto, and J. M. Corchado, Eds. Cham: Springer International Publishing, 2020, pp. 79–86.

[4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *www.bitcoin.org*, 2008.

[5] B. Hayes, S. Thakur, and J. Breslin, “Co-simulation of electricity distribution networks and peer to peer energy trading platforms,” *International Journal of Electrical Power & Energy Systems*, vol. 115, p. 105419, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061519302972

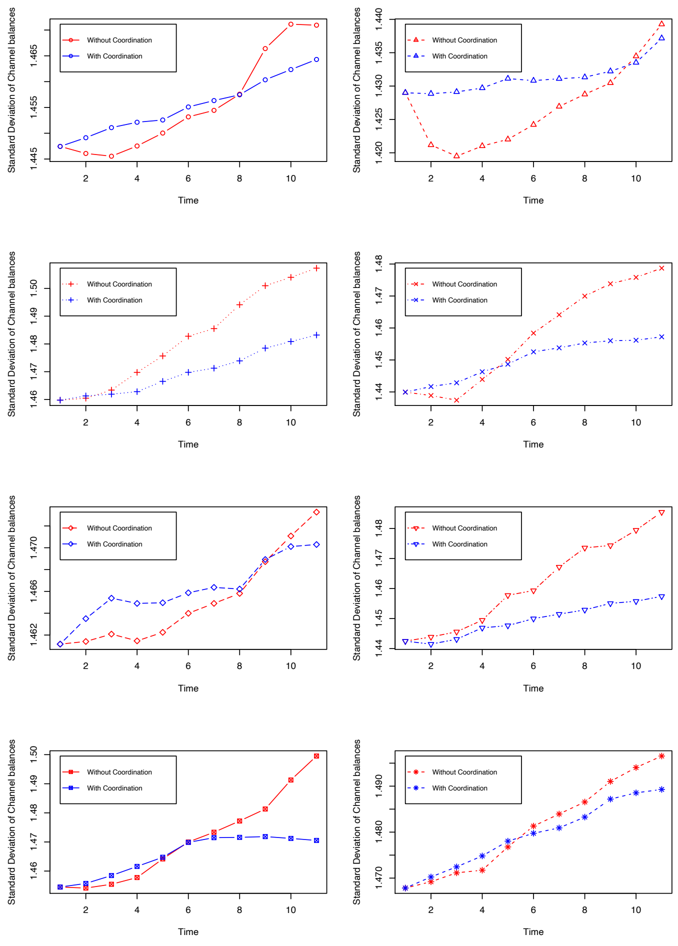

Fig. 3. Standard deviation of channel values for 8 networks. It clearly shows standard deviation of channel values remains for PBT execution with coordinated hubs remains lower than the same for PBT execution without coordination over the landmarks.

[6] I. A. Ridhawi, M. Aloqaily, A. Boukerche, and Y. Jaraweh, "A blockchain-based decentralized composition solution for iot services," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.
[7] J. Poon and T. Dryja, "The Bitcoin Lightning Network:Scalable Off-Chain Instant Payments." [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[8] "Raiden network," http://raiden.network/, accessed 2018.
[9] P. Moreno-Sanchez, A. Kate, M. Maffei, and K. Pecina, "Privacy preserving payments in credit networks: Enabling trust with privacy in online marketplaces," in *NDSS*, 2015.
[10] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare : An approach to routing in lightning network white paper," 2016.
[11] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 473–489. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134093
[12] B. Viswanath, M. Mondal, K. P. Gummadi, A. Mislove, and A. Post, "Canal: scaling social network-based sybil tolerance schemes," in *EuroSys*, 2012.
[13] J. Lind, I. Eyal, P. R. Pietzuch, and E. G. Sirer, "Teechan: Payment channels using trusted execution environments," *CoRR*, vol. abs/1612.07766, 2016. [Online]. Available: http://arxiv.org/abs/1612.07766
[14] S. Thakur and J. Breslin, "Collusion attack from hubs in the blockchain offline channel network," in *1st International Conference on Mathematical Research for Blockchain*, 2019.

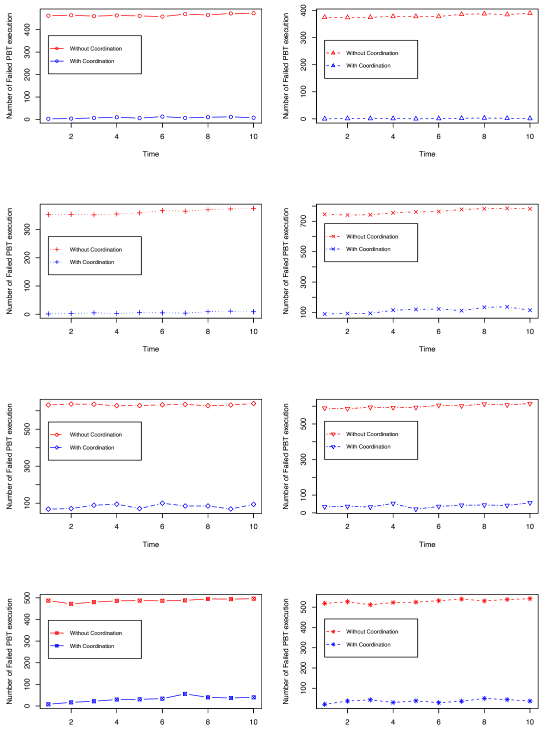

Fig. 4. It shows the number of failed PBT execution with and without coordination among the landmarks.

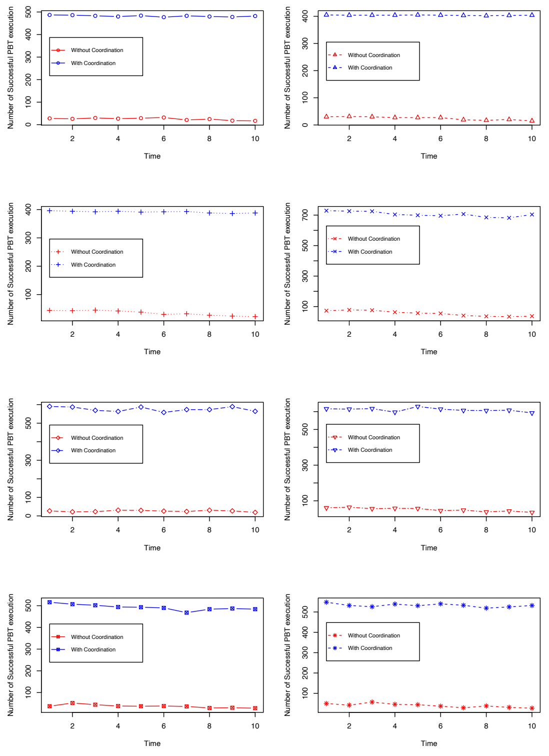

Fig. 5. It shows the number of successful PBT execution with and without coordination among the landmarks.