Scalable Decentralized Routing for Blockchain Payment Networks

Xiaoxue Zhang, Shouqian Shi and Chen Qian
University of California, Santa Cruz
Email: {xzhan330, sshi27, cqian12}@ucsc.edu

ABSTRACT

Payment channel networks have been designed and utilized to resolve the throughput limitation of blockchains. The ultimate goal of payment channel networks is that everyone can conduct payment transactions with any other person in the world using a social trust network without the intervention of centralized organizations such as banks and governments. However routing is a critical problem in payment channel networks and existing routing solutions for overlay network do not satisfy the scalability and decentralized requirements.

In this paper, we propose WebFlow, a set of greedy routing protocols for payment channel networks, which only requires each user node to maintain localized information and can be used for massive-scale networks. We utilize the extended routing algorithms based on the multi-hop Delaunay triangulation (MDT). Our simulation studies indicate that WebFlow can achieve low delay and small probing overhead with guaranteed success of finding paths for all transactions.

1. INTRODUCTION

Blockchain is a promising solution for decentralized digital ledgers. Since it was invented as a public transaction ledger of the cryptocurrency Bitcoin in 2008 [14], there have been many other payment systems emerging based on blockchain, such as Ripple [3], Stellar [1], and Ethereum [8]. While blockchain has achieved a wide success in many payment systems, scalability remains a huge problem with growing number of users and transactions [6, 17]. For instance, bitcoin could only support fewer than 25 transactions per second at peak in 2019 [2]. In contrast, some widely used centralized payment hubs such as visa and MasterCard can process more than 47,000 peak transactions per second during 2013 holidays [17]. The reason of this low throughput of blockchain is that every node processes all transactions and the consensus is achieved by proof-of-work, a time- and resource-consuming process. Every time when new block arrives, all the nodes in the network have to process it and update the state of the blockchain. Using blockchain as a global transaction system for massive users is impractical at this moment. There have been some improvements of basic blockchain such as Bitcoin-NG [7], Monoxide [21] and so on. However their performance is still limited by the processing capacity of the nodes and network bandwidth and cannot be used as a global transaction system.

The recently proposed concept of payment channel networks (or offchain networks) [10,17] provide a high-throughput solution for blockchain based payment systems. In a payment channel network, each user can conduct transactions with another users through a bi-directional channel. Instead of broadcasting all transactions to every node in the network, only two transactions related to this channel need to be recorded on the blockchain: opening and shutting down the channel. Each user commits certain fund in the opening of this channel. Then the two users can make any number of transactions that update the tentative distribution of the channel's funds without broadcasting them to the blockchain. Two users sharing a channel reflects some level of trust. In addition, a user can make a payment to (or receive fund from) another untrusted user via a multi-hop path, where any two consecutive users on the path share a channel. The ultimate vision is that everyone can conduct payment transactions with any other person in the world using a social trust network [9] without the intervention of centralized organizations such as banks and governments.

Routing, i.e., finding a path between two arbitrary users, is a critical problem in payment channel networks. Current routing solutions of overlay networks (including payment channel networks) can be classified in four types. 1) Centralized routing that assumes every node knows the entire network topology based on which the routing paths are determined [17, 22]. This approach has several major problems to be used for large-scale payment channel networks, including massive control message broadcast, memory and computation scalability on each node, and privacy leakage of the global topology. 2) Landmark routing, where selected nodes (called routers) store routes for the rest of the network, and other nodes only need route transactions to a landmark [13, 20]. However, the landmarks may be congested by massive traffic (transaction) and become system bottlenecks. The landmarks again becomes a certain level of centralized nodes such as banks, which violate the decentralized nature of payment channel networks. 3) Structured overlay networks such as distributed hash tables (DHTs), which force each node to maintain links to designated nodes and use the links for routing. DHTs cannot be used for pay-

Third International Symposium on Foundations and Applications of Blockchain (FAB '20) May 1, 2020, Santa Cruz, California, USA.

ment channel networks because one cannot be forced to build channels with untrusted users. 4) Gossip routing, whose major weakness is no guaranteed success of finding a path.

To our knowledge, this work is the first attempt to design a scalable and fully decentralized routing solution for payment channel networks, where no central authority or landmarks are needed. We investigate the possibility of using greedy routing that has been used for wireless networks [5, 11, 12, 19], data center networks [18, 23], and memory interconnections [16]. We propose a routing protocol called WebFlow. WebFlow allows every node (user) to calculate a set of Euclidean coordinates and uses the coordinates to perform greedy routing [12]. To guarantee the success of path finding, nodes can maintain a multi-hop Delaunay triangulation (MDT) based on only the channels with trusted users. It has been proved that on an MDT, greedy routing always succeeds to find the destination.

2. WEBFLOW OVERVIEW

WebFlow is a distributed routing system for payment channel networks. In payment channel networks, each user is called a *node*. We call two nodes are *physical neighbors* and they share a *physical link*, if a bi-directional channel exists between these two users to allow them to make several transactions without broadcasting them to the blockchain.

Problem definition. The routing problem of WebFlow is described as follows. Consider a transaction t initiated by node s (called the *source*) that should be received by d (called the *destination*). WebFlow needs to find a path of links from s to d, where two consecutive nodes on the path should share a physical link (payment channel). The success of routing implies that s can make a transaction with d by a sequence of transactions involving other nodes, even if s and d have no trusted channel.

Every node locally maintains the links and capacity to its own neighbors. As the topology of the network is locally unknown, it is hard to figure out the hopcount between two arbitrary nodes. So we introduce the coordinates here to estimate the hopcount (distance) between two nodes. Each node computes a set of coordinates for itself by the *user positioning* algorithm. When two nodes are neighbors, they know the coordinates of each other. Every node needs to store the coordinates of itself and its neighbors. When two nodes need to make transactions, they exchange their coordinates first. Greedy routing based on MDT can then be performed among all nodes that have computed the coordinates.

2.1 USER POSITIONING

For fast convergence and scalability, the proposed user positioning algorithm is in two steps. First a small group of nodes are selected as trackers based on certain criteria or simply random nodes. These trackers first compute their coordinates, and then in the second part, any ordinary node can compute its own coordinates according to the messages sent from these trackers. **Note the trackers in WebFlow do not participate routing and only support coordinates computation. Hence it avoids the landmark limitations in landmark-based routing [13, 20].**

2.1.1 Tracker operations

We model a payment channel network as a graph $G = (V, E)$ in a Euclidean space S, where V is the set of nodes and E is the set of links. Each node N will compute a set of coordinates c_N^S in S. The goal is to let nodes maintain coordinates that characterize their locations in the network such that hopcount can be predicted by evaluating a distance function over their coordinates. We denote the actual hopcount between nodes N_1 and N_2 as $h_{N_1N_2}$. The Euclidean distance between the coordinates of N_1 and N_2 is denoted as $d^S_{N_1N_2}$.

A set of trackers, $T = t_1, ..., t_k$ are randomly selected (for example, based on the user IDs). $|T| = k$. Note that for a D-dimensional Euclidean space, k needs to be at least $D + 1$ [15].Here for simplicity, we assume WebFlow uses a 3-dimensional Euclidean space and $k \geq 4$.

Once trackers are chosen, each tracker broadcasts the network, resulting in a spanning tree. Then all nodes including other trackers will know their hopcounts to this tracker. Hence all trackers know a $k \times k$ hopcount matrix characterizing the distances between every pair of the trackers. We denote the actual hops between t_i and t_j as $h^S_{t_it_j}$, and the computed distance between these two trackers as $d^S_{t_it_j}$. In order to predict hopcount between nodes accurately, we need to find a set of coordinates $c^S_{t_1}$,...,$c^S_{t_k}$, to make the overall error between actual hops and distances minimized, which can be computed by existing algorithms of multi-dimensional scaling [4].

2.1.2 Node Operations

Once the coordinates of trackers, $c^S_{t_1}$,...,$c^S_{t_k}$, are determined, they are broadcasting to all other nodes. Then every node can compute their own coordinates in a distributed way. When constructing the spanning trees for trackers, ordinary nodes have already known the hops to each tracker and locally recorded these values. We still firstly assign a random coordinate c_N for the node N same as Tracker operations. Using the k measured nodes-to-tracker hopcounts, h_{Hl_i}, node N can compute its own coordinates c_N by minimizing the overall error between actual hops and computed distances.

As payment channel network is very dynamic and nodes may join and leave the network. We have to consider these two situations. When a new node wants to join the network, it just asks all its physical neighbors for their hopcounts to the trackers, and directly get its own hopcounts to the trackers. As the node positioning algorithm is distributed, this new node could easily compute its coordinate at any time without synchronization with others. After a time period, if a lot of nodes go offline and many new nodes join in, the whole network topology will change a lot, and the original coordinates are not accurate to reflect the routing distance. Here, the above two steps can be periodically re-executed to get updated coordinates.

2.2 ROUTING ALGORITHM

It is well-known that simple greedy routing does not always succeed, because it may be stuck at a local minimum node, i.e., the node cannot find any neighbor that has closer distance to the destination [12]. Multi-hop Delaunay triangulation (MDT) [12] is a geographic routing protocol providing guaranteed delivery with low routing stretch.

For a given set of nodes S in a 2D space, a DT of S, $DT(S)$, is defined as a triangulation such that no forth node in S is inside the circumcircle of any triangle. Two nodes u and v are called *DT neighbors* if they share an edge in

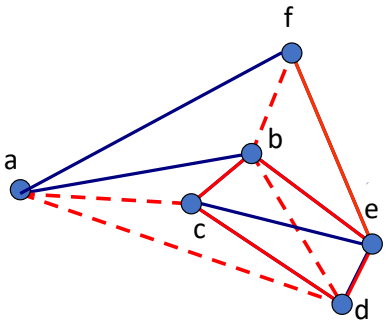


Figure 1: MDT graph of 6 nodes.

$DT(S)$. We denote $DT(S)$ as a tuple $\{< u, N_u > | u \in S\}$, where N_u is the set of u's DT neighbors, which is locally maintained by node u. It has been proved that greedy routing on the DT edges (i.e., assuming all DT neighbors are connected) always succeeds to find the destination [12]. This result can be extended to d-dimension for $d \geq 2$.

The **major challenging** of applying DT to a real network is that two DT neighbor may not be directly connected in the network topology. In the case of this work, two nodes that are DT neighbors may not have a trusted channel or physical link. To address this problem, the multi-hop DT is specified as a 3-tuple $\{< u, N_u, F_u > | u \in S\}$, where F_u is a soft-state forwarding table, and N_u is the set of u's DT neighbors. The extension of MDT from DT is that a node in N_u is not necessarily directly connected to node u by physical links. Apart from directly connected to u, it could also be multiple hops apart. In this case, we call those nodes who are u's DT neighbors but not physical neighbors communicate via a *virtual link* with u. Even if two DT neighbors do not have a trusted channel, they can be connected by virtual link because there is an underlying multi-hop path of physical links that connects one DT neighbor to the other. For example, nodes a and d in Figure 1 have no trusted channel, but a multi-hop path with physical link $a - b - c - d$ supports the virtual link ad. Note that in WebFlow no node should maintain a global view of the MDT such as the graph shown in Figure 1. Each node only maintains local information $< u, N_u, F_u >$ that is independent of the network size. The routing decisions are made locally. In addition there is no super nodes that handle most transactions such as the landmarks in landmark routing. Hence WebFlow is highly scalable and decentralized. **Given the destination coordinates, WebFlow routing using the MDT graph always finds a path to the destination based on local decisions of the nodes on the path.**

The routing algorithms of WebFlow contains several MDT protocols including the forwarding protocol, join protocol, maintenance protocol, leave protocol, failure protocol and initialization protocol. The forwarding protocol determines how a node should locally decide the next-hop node when routing to a destination, when a correct MDT is maintained. The other protocols are used to maintain a correct MDT graph. They are all decentralized algorithms.

2.2.1 *Forwarding Protocol*

Consider a transaction t initiated by node s that should be received by d. For each node u that needs to route this transaction t, if u can find a physical neighbor v such that the Euclidean distance $D(v, d) < D(u, d)$, u sends the transaction to v. Otherwise, u finds the DT neighbor v' that is closest to d among all u's DT neighbors. u then sends the transaction to v' using the virtual link. A correct MDT guarantees that this forwarding protocol will find the destination d in a finite number of hops.

2.2.2 *Join Protocol*

When a new node w boots up and wants to join the network, it first needs to know its physical neighbors and assigns its own coordinate. Then it will send join request to its neighbors trying to find all of its DT neighbors. To begin its search, it must find at least one neighbor in the new DT. By a property of DT, the closest node to w in the Euclidean space must be a DT neighbor.

2.2.3 *Maintenance Protocol*

Payment channel networks are very dynamic and there are always nodes and channels setting up and offline. Hence we need a maintenance protocol to fix the structure of MDT. The MDT graph is correct only if every node knows all of its DT neighbors. So in WebFlow, each node u queries some of its neighbors to see if they know mutual neighbors that node u does not know, and then sends neighbor-set requests to them. If node u discovers a new neighbor from neighbor-set replies, u will send a neighbor-set request to this new neighbor if they are vertexes in a same simplex in $DT(C_u)$. Every node runs this maintenance protocol locally, and every time when a node finishes running it, it will wait for a time period T_m until running it again.

3. PERFORMANCE EVALUATION

In this section, we conduct some initial experimental evaluation of the WebFlow routing algorithms based on simulations. The evaluation aims to answer the following questions:

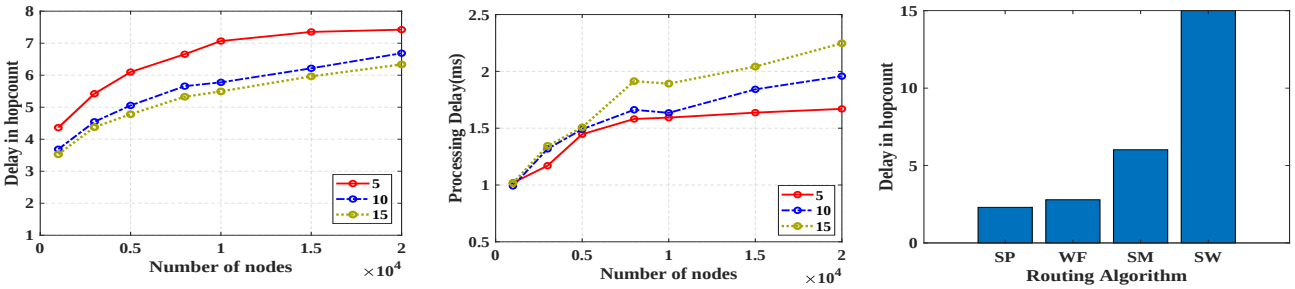
- How does the WebFlow routing perform under simulated payment channel network topologies?
- How effective is the routing algorithm under different numbers of nodes?
- How does the number of channels per node (average degree) affect the routing performance?

3.1 Methodology

Setup. We implement payment channel network topologies and routing schemes using the networkx package in Python in the simulations. We study several real-world offchain networks: Ripple and Lightning. Ripple includes 1,870 nodes and 17,416 edges from January 2013 to November 2016, and the Lightning network has 2,511 nodes and 36,016 edges on one day of December 2018 [22]. Based on these observations, in our simulation, we consider the number of nodes varying from 1,000 to 20,000. And we randomly set up some channels between those nodes by varying the average degree with 5, 10, and 15. Finally, we generate payments between arbitrary nodes to test WebFlow.

3.2 Overall Performance and Overhead

Delay (in terms of the number of routing hops). We first evaluate the delay in the average number of routing hops, which shows the efficiency of WebFlow. In Figure 2(a),



(a) Delay vs N. (b) Processing delay vs N. (c) Performance of different routing algorithm

Figure 2: Performance results by varying the number of nodes and average degree, and comparison of different algorithm: shortest-path, MDT, WebFlow, SpeedyMurmurs and SilentWhispers.

the result show the delay in the average number of routing hops, by varying the number of nodes from 1,000 to 20,000. The three curves in the figure are different in the average number of physical neighbors (average degree) per nodes, which reflects if a network is densely connected. The result shows that even if the number of nodes reaches 20,000, and the average degree is 5, WebFlow can finish an arbitrary transaction in less than 8 hops.

Processing time. We then evaluate the processing time of WebFlow, which is the probing delay to find a transaction path. Figure 2(b) shows the results. As the number of nodes increases, it is obvious that the processing time will also increase. Even when the number of nodes is 20,000, the probing time is still less than 2.5 ms for each node. The greedy routing in WebFlow minimizes unnecessary paths as much as possible. Hence the computation overhead is low due to less unnecessarily path computation.

Comparison to other schemes. We compare WebFlow with shortest-path, SpeedyMurmurs [20] and SilentWhispers [13] in Figure 2(c) based on Ripple topology. The result shows that delay hopcount of WebFlow is lower than that of SpeedyMurmurs and SilentWhispers. It is close to the delay hopcount achieved by shortest-path routing algorithm.

In summary, our initial experiments show that WebFlow is highly scalable with good performance.

4. FUTURE WORK

WebFlow achieves efficient and effective routing in payment channel networks. This paper only presents the initial work. There are several challenges that require further studies. We will mainly focus on the following three aspects in our future work.

Capacity. Our existing design has not considered the capacity of each channel. This is an extremely important problem in payment channel network, because each channel on a path must has higher capacity than the transaction amount. We will work on an extended MDT routing protocol with channel capacity considerations. When one path cannot support a transaction, multi-path routing needs to be executed.

Concurrency. In our design, we separately probe the transaction operations. If several paths want to use the same channel during probing, two situations might happen. First, the two paths share this channel, but a solution needs to be found when the channel capacity is not sufficiently high. Second, the channel is reserved in a first-come first-serve manner. Then we need to provide a solution to find another path for the second transaction.

Privacy. There are three privacy properties for payment channel network: value privacy, sender privacy, and receiver privacy. We will consider the privacy of WebFlow under both attacker models: when the attackers is on the path and not on the path. We plan to apply an idea similar to the Voronoi routing [5] based on MDT. To hide the exact destination coordinates, we may use a ray to indicate the routing direction, and we are able to prove that Voronoi routing guarantees to find the actual destination if the ray intersects the Voronoi cell of the destination node.

5. CONCLUSION

In this work, we present the design of a scalable and decentralized routing solution called WebFlow for large and dynamic payment channel networks, which resolves the scalability problems of prior routing methods. The initial results demonstrate that WebFlow provides low delay, low probing overhead, and guaranteed success of finding paths, even for big networks. Our future work will be on the channel capacity constraints, concurrent routing control, and privacy protections.

6. ACKNOWLEDGMENT

The authors were supported by NSF Grants 1750704 and 1932447. We thank the comments from anonymous reviewers.

7. REFERENCES

[1] Stellar website. http://www.stellar.org/. Accessed Feb, 2020.

[2] Transaction rate of bitcoin. http://www.blockchain.com/en/charts/transactions-per-second. Accessed Feb, 2020.

[3] Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., and Zenner, E. Ripple: Overview and outlook. In *Proceedings of the 8th TRUST* (2015), Springer, pp. 163–180.

[4] Borg, I., and Groenen, P. J. *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media, 2005.

[5] Bose, P., and Morin, P. Online routing in triangulations. *SIAM journal on computing 33*, 4 (2004), 937–951.

[6] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P.,

Shi, E., Sirer, E. G., et al. On scaling decentralized blockchains. In *Proceedings of the 20th FC* (2016), Springer, pp. 106–125.

[7] Eyal, I., Gencer, A. E., Sirer, E. G., and Van Renesse, R. Bitcoin-ng: A scalable blockchain protocol. In *Proceedings of the 13th NSDI* (2016), pp. 45–59.

[8] Foundation, E. Ethereum project. http://www.ethereum.org/. Accessed Feb, 2020.

[9] Fugger, R. Money as ious in social trust networks and a proposal for a decentralized currency network protocol. Tech. rep., 2004.

[10] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th SOSP* (2017), pp. 51–68.

[11] Karp, B., and Kung, H. Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. of ACM Mobicom* (2000).

[12] Lam, S. S., and Qian, C. Geographic routing in d-dimensional spaces with guaranteed delivery and low stretch. *ACM SIGMETRICS Performance Evaluation Review 39*, 1 (2011), 217–228.

[13] Malavolta, G., Moreno-Sanchez, P., Kate, A., and Maffei, M. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *Proceedings of NDSS* (2017).

[14] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Manubot, 2019.

[15] Ng, T. E., and Zhang, H. Predicting internet network distance with coordinates-based approaches. In *Proceedings of the 21th INFOCOM* (2002), vol. 1, IEEE, pp. 170–179.

[16] Ogleari, M., Yu, Y., Qian, C., Miller, E., and Zhao, J. String Figure: A Scalable and Elastic Memory Network Architecture. In *Proc. of IEEE HPCA* (2019).

[17] Poon, J., and Dryja, T. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[18] Qian, C., and Lam, S. ROME: Routing On Metropolitan-scale Ethernet . In *Proc. of IEEE ICNP* (2012), pp. 1–10.

[19] Qian, C., and Lam, S. S. Greedy Distance Vector Routing. In *Proc. of IEEE ICDCS* (June 2011), pp. 857–868.

[20] Roos, S., Moreno-Sanchez, P., Kate, A., and Goldberg, I. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748* (2017).

[21] Wang, J., and Wang, H. Monoxide: Scale out blockchains with asynchronous consensus zones. In *Proceedings of the 16th NSDI* (2019), pp. 95–112.

[22] Wang, P., Xu, H., Jin, X., and Wang, T. Flash: efficient dynamic routing for offchain networks. In *Proceedings of the 15th CoNEXT* (2019), pp. 370–381.

[23] Yu, Y., and Qian, C. Space shuffle: A scalable, flexible, and high-performance data center network. In *Proc. of IEEE ICNP* (2014).