VHash: An Optimized Voronoi-Based Distributed Hash Table

Double Blind

***Abstract*—Distributed Hash Tables (DHT) provide a fast and robust decentralized means of key-value storage and retrieval and are typically used in Peer-to-Peer applications. DHTs assign nodes a single identifier derived from the hash of their IP address and port, which results in a random overlay network. A random overlay network is not explicitly optimized for certain metrics, such as latency, trust, energy, or hops on an overlay network. It is often desirable to generate an overlay network that is optimized for one or more specific metrics.**

This paper presents VHash, a DHT protocol to construct an overlay optimized for such metrics. VHash exploits a fast and efficient Delaunay Triangulation heuristic in a geometric space. We used VHash to generate an overlay with edges that minimize latency. While we focused on latency in this paper, VHash optimizes on any defined metrics. This overlay outperformed an overlay generated by the Chord DHT protocol in terms of lookup time. VHash provides a robust, scalable, and efficient distributed lookup service.

I. Introduction

A Distributed Hash Table is used provide an overlay network for many P2P applications. Each node in the overlay network maintains a routing table of a subset of other nodes in the overlay. The configuration and rules of the routing table vary from one protocol and another; the entries of the table might be separated by powers of 2 [1], be determined by shared prefixes [2], or be chosen according to a probabilistic distribution [3], but the goal is to minimize the number of overlay hops the distributed lookup needs to make.

A routing table created to minimize overlay hops does exactly that; it does not necessarily create routes with minimized latency. However, what if more information about the nodes could be encoded as part of the overlay, such as the node's latency or energy? We wanted to design an overlay where network metrics, such as inter-node latency, node energy levels, or non-euclidean metrics like trust could be embedded in the network. The most straightforward way is to assign each network metric to a node as a coordinate in a space with the hashed key treated as an additional coordinate. If each node is defined as object in this space, we can then use a distance function to choose the shortest path over these metrics.

This opens up two problems: how do we generate an multidimensional overlay that incorporates network metrics as part of routing and how do we generate the coordinates that correspond to the network metrics? We present VHash as our solution to these problems.

VHash is a DHT designed to take inter-node latency information into account when generating an overlay, although other network metrics could easily be applied. VHash creates an approximation of a Voronoi network and Delaunay Triangulation to define the routing tables and dictate where content is stored in the network. We accomplish this by assigning each node d coordinates, rather than than a single key. We use an basic spring model to embed inter-node latency graph in the overlay. A result of this is that nodes in VHash can *move* along an axis; their position is not nessarily fixed. Our paper presents the following:

- We describe the VHash protocol (Section II) and the underlying approximation algorithm of overlay's Delaunay Trinangulation and the corresponding Voronoi diagrams. Our approximation is distributed, greedy, efficient, and accurate in arbitrary number of dimensions and creates an overlay with edges that minimize distance over network metrics while maintaining robustness, scalablilty, and polylogarithmic lookup time in overlay hops.
- We use VHash to provide us with an overlay for embedding network metrics, our other innovation. We present our basic spring model for embedding nodes in the overlay with latency information and discuss other network metrics that can be used with VHash (Section III).
- We created simulations (Section IV) to prove that the overlays created by VHash are accurate enough for routing messages from arbitrary source nodes to random destination locations. We also show that by embedding the latency graph, our routing dramatically outperforms Chord (and by extension, other overlays with $\lg(n)$-sized routing tables) in terms of latency.
- We compare the VHash protocol to the other protocols that are based off of Voronoi region approximation. We also contrast the properties of VHash with well known extant protocols.
- We discuss future work that follows from our what plans we have for embedding different problems using VHash.

II. VHash

The process of generating the keys/coordinates is covered in the next section.

III. Network Metric Embedding

IV. Simulations

Scale free networks are internet shaped[**?**]

We implemented two experimental tests to prove that VHash acts as as a stable DHT and is able to minimize the latency on lookup time. While we have implemented functional software versions of the protocol, it was more economical to simulate VHash behavoir in a centralized fashion for these experiments. First, we simulate hit-rate over time as a VHash network is established from a random graph. Second we compare the distributions of latency in Vhash and Chord DHTs to examine

benefits from optimizing the network topology for minimum latency.

A. Convergence Simulation

As proof that the VHash protocol converges to a stable overlay from an inclement starting topology, we simulate the hit-rate of the network as the network as the protocol builds a topology from an initally random starting network. We compare these results to RayNet, which preformed similar experiements to demonstrate fast self organizing behavoir.

RayNet proposed that a random k-connected graph would be a good demonstration of convergance for a DHT. To generate this graph Raynet, for the first two generations, added 10 randomly selected nodes to the peerlist of each node. Then the network's hit rate is sampled and recorded. Hit rate of the network is calculated by taking taking 2000 samples as follows:

1) Choose a node N at random as a starting point.
2) Choose a destination point P at random from the VHash space.
3) Use VHash routing to recursively seek the responsbile node R for p starting from N
4) Calculate the closest node C to P.
5) If C == R then register a hit
6) Otherwise register a miss

1) Experimental Results: Results describe a curve of fast growth in hit-rate then convergance a a hitrate of 1.0. As the network size and number of dimensions increase, convergence slows.

B. Latency Distribution Test

In this experiment we are primarilly concerned with VHash's ability to actively lower latency. We considered it important to examine how latency generated by the network is distributed to examine where VHash is making improvments. We use Chord as a DHT representative of O(log(n)) lookup topology.

Rather than examine the number of hops on the overlay network as our primary metric like previous works, we are concerned with the latency taking the properties of the underlay network into account. Our unit of latencty is "hops on the underlay". We generate a 10,000 node Scale Free Network (approximate diameter of 3) of which a subset are selected to act as members of the DHT. We have no basis by which to assign latency values to each edge on the underlay network, so we measure latency in terms of hops on this underlay network. VHash generates an embeding of the latency graph utilizing the distributed spring model algorithim described above.

1) Experimental Results: VHash has significantly shorter lookup latency then the chord graph with smaller variation in latency. Both are generally worse than the real network diameter of 3. As the size of the network increases, the accuracy of the graph embedding falls (as scale free graphs difficult to embed.)

C. Conclusions

In general, in similar circumstance we converge faster than Ray-Net. Convergence rate varies with network size and number of dimensions of the VHash space. Utilizing the current latency-embedding technique VHash is usefull for building DHT networks of reasonable size.

V. Related Work

VI. Future Work

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM Computer Communication Review*, vol. 31, pp. 149–160, ACM, 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*, pp. 329–350, Springer, 2001.

[3] J. M. Kleinberg, "Navigation in a small world," *Nature*, vol. 406, no. 6798, pp. 845–845, 2000.