T-DHT: Topology-Based Distributed Hash Tables

Olaf Landsiedel, Katharina Anna Lehmann, Klaus Wehrle
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen, Germany
{olaf.landsiedel, klaus.wehrle}@uni-tuebingen.de lehmannk@informatik.uni-tuebingen.de

Abstract

In this paper, we introduce topology-based distributed hash tables (T-DHT) as an infrastructure for data-centric storage, information processing, and routing in ad hoc and sensor networks. T-DHTs do not rely on location information and work even in the presence of voids in the network. Using a virtual coordinate system, we construct a distributed hash table which is strongly oriented to the underlying network topology. Thus, adjacent areas in the hash table commonly have a direct link in the network. Routing in the T-DHT guarantees reachability and introduces low hop-overhead compared with the shortest path.

1. Introduction

Research advances in low-power hardware, wireless communication technology, and highly embedded operating systems enable new types of ad-hoc networks, such as sensor networks. A sensor network may consist of several hundreds of nodes, each with very limited processing, storage, and communication capabilities. It provides large amounts of detailed measurements, which need to be aggregated, evaluated, and stored in the network.

In a sensor network, the derived data is commonly considered more important than the node that gathered or stores it. Thus, *data-centric* communication paradigms have been proposed. In such a scheme, data is named, e.g. “average temperature”, and communication refers to this name instead of a node’s network address.

The main contribution of this paper is a scalable and distributed algorithm for the construction of distributed hash tables which are strongly oriented to the underlying network topology. As a result, adjacent areas in the DHT commonly have a direct link in the network. Thus, routing in the hash table introduces low overhead compared with the shortest path – a crucial fact, as communication requires large amounts of energy. Furthermore, the proposed algorithm does not rely on position information. Building a T-DHT consists of two steps: (1) The construction of a virtual coordinate space which represents the network topology. (2) On top of these coordinates we build a two-dimensional hash table, where each node maintains the data close to its virtual position.

2. Lightweight Virtual Coordinate System

In this section we introduce our algorithm to construct the virtual coordinate system. For this, three (or more) reference nodes are randomly selected. Each node triangulates its virtual position from these reference nodes. Thus, the reference nodes flood the network with the distance in hops between each other. Using the hop distance to the reference nodes, each node can determine its position in the virtual coordinate space.

Note that this position only presents the node’s position in the network topology and not the physical position. However, unlike other approaches, we are not interested in the physical node positions. The consistent virtual coordinate system provided by the described algorithm is sufficient to construct a T-DHT.

3. Constructing Topology-Based Hash Tables

Inspired by the Content Addressable Network [1], we construct a two-dimensional DHT on top of the virtual coordinate system. The coordinate space is divided among the participating nodes. Each node maintains a rectangular area around its position in the virtual coordinate system.

In the hash table (key,value) pairs can be stored and retrieved deterministically. To store or retrieve data, a hash function maps the data to a position in the virtual coordinate space. Next, the data is routed through the hash table to the node maintaining the area in which the key is located.

Each node maintains a routing table containing the path to its neighbors in the coordinate system and the area each neighbor maintains. Routing in the two-dimensional hash space is intuitive. Using its routing table, a node forwards a message to its destination through simple greedy forwarding. Additionally, a node may have links to nodes which are not direct neighbors in the hash table. Thus, it can use these to “short-cut” the route.

Joining the T-DHT consists of three steps: (1) The node wishing to join must first find a node which is already in the T-DHT. (2) Via T-DHT routing, it finds the node maintaining the zone of its position in the virtual coordinate system. This zone is equally split between the two nodes. (3) The new member informs its neighbors about its presence.

To bootstrap, the first reference node maintains the whole DHT. When the virtual coordinates are assigned, it announces its T-DHT membership to its neighbors. Next, the neighbors join the distributed hash table and themselves announce their membership to their neighbors. Joining the T-DHT is distributed as a joining node only needs to contact the node maintaining the corresponding area. Thus, the algorithm does not place a high load on a single node or the routes to it.

4. Evaluation

In this section we evaluate the performance and scalability of the presented scheme. Figure 1 shows the cumulative distribution function for the number of extra hops T-DHT routing takes over shortest path routing. The simulated network contains 500 nodes distributed over a 200m x 200m area, each with a communication range of 20m (averaging 15.5 edges per node). Using three reference nodes, the T-DHT is able to route nearly 70% of the packets with two or less extra hops through the network.

To join, a node has to send a message through the T-DHT to the node maintaining the area it wants to split. Based on the T-DHT scheme, adjacent nodes commonly share adjacent areas in the T-DHT. Our simulation results show that over 91% of the joining nodes split an area with an adjacent node. The average number of hops to the node maintaining the area to be split is 1.4. After the split, a node needs to inform its neighbors about its presence. For the 500 node simulation, each node has an average of 4.6 neighbors in the T-DHT, with more than 81% one-hop distance. On average, it takes 1.7 hops to reach a neighbor. Thus, to join and inform the neighbors, a node has to send an average of 5.6 messages, resulting in a total of 9.2 hops. Furthermore, our simulation results show that the number of neighbors is independent of the number of nodes in the system. Thus, the number of messages sent and the corresponding hops do not depend on the number of nodes in the system, resulting in high scalability of the T-DHT.

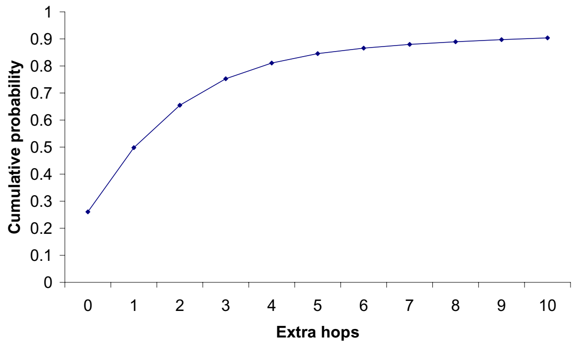

Figure 1. CDF of extra hops over shortest path routing, averaged over 10 simulation runs

5. Related Work

Recent research provides many distributed hash table systems for the Internet, such as CAN [1], Chord [2], and Pastry [3]. However, they do not take the underlying network topology into account. New approaches focus on building topology-aware distributed hash tables for the Internet [4, 5]. Nonetheless in the Internet, nodes are connected on a logical level and not on a physical level as in ad hoc and sensor networks. Thus, the proposed algorithms do not apply to such a network. Data-centric storage in sensor networks was first introduced by [6]. However, the proposed technique relies on the fact that each node knows its physical position. Like our approach, GEM [7] proposes an algorithm for data-centric storage and routing. However, their algorithm is not decentralized.

6. Conclusion and Future Work

In this paper we propose a topology-based distributed hash table (T-DHT) for data-centric routing and storage in ad hoc and sensor networks. Using a simple virtual coordinate system, we construct a distributed hash table which is based on the underlying network topology. Our evaluation shows that the proposed scheme results in low overhead compared with shortest path routing and is highly scalable. The T-DHT guarantees reachability although each node only maintains a small routing table, which contains its neighbors in the T-DHT and the nodes to which it has a direct link to.

The presented scheme is ongoing work, and many interesting questions still need to be addressed. Currently, we are evaluating the impact of the positions of the reference nodes as well as the number of reference nodes. Furthermore, we will evaluate the improvement in routing performance arising from maintaining a two-hop neighborhood.

References

[1] S. Ratnasamy et al., "A scalable content-addressable network," in *Proc. of SIGCOMM*, September 2001.

[2] I. Stoica and R. Rinaldi, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of SIGCOMM*, Aug. 2001.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM Int. Conf. on Distributed Systems Platforms*, Nov. 2001.

[4] M. Waldvogel and R. Rinaldi, "Efficient Topology-Aware Overlay Network," in *Proc. of HotNets*, Oct. 2002.

[5] Z. Xu, C. Tang, and Z. Zhang, "Building Topology-Aware Overlays using Global Soft-State," in *Proc. of ICDSC*, May 2003.

[6] S. Shenker et al., "Data-Centric Storage in Sensornets," in *Proc. HotNets*, Oct. 2002.

[7] J. Newsome and D. Song, "GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information," in *Proc. of SenSys*, Nov. 2003.