How to Implement DHTs in Mobile Ad Hoc Networks?

Himabindu Pucha, Saumitra M. Das, and Y. Charlie Hu
Purdue University
West Lafayette, IN 47907
Email: {hpucha, smdas, ychu}@purdue.edu

1 Motivation

Recently, Distributed Hash Tables (DHTs) such as CAN, Chord, Pastry and Tapestry have been proposed as a novel platform for building a variety of scalable and robust distributed applications for the Internet, such as distributed storage systems, application level multicast, and content-based full-text search. A DHT substrate shields many difficult issues including fault-tolerance, locating objects, scalability, availability, load balancing, and incremental deployment from the distributed application designers.

We argue that the DHT abstraction, if deployed in MANETs, could similarly provide an efficient way of constructing distributed applications and services. For example, applications such as file sharing and resource discovery can benefit from the distributed insert/lookup convergence provided by DHTs. However, DHTs have been designed for the Internet, and bandwidth limitations, node mobility, and multi access interference pose unique challenges to deploying DHTs in MANETs. In this poster, we study how to efficiently support a DHT abstraction in mobile ad hoc networks.

2 Design Options

We explore two opposite options in the design space of implementing a DHT in MANETs using a proximity-aware DHT Pastry [3] and an on-demand MANET routing protocol DSR [1] as concrete examples.

Layered Approach In the layered approach, a proximity-aware DHT Pastry is directly layered on top of MANETs in the same way it is layered on top of the Internet. Pastry maintains its leaf set and routing table entries without source routes and DSR maintains source routes passively as per the demand of Pastry routing state.

However, a straightforward layering is not pragmatic, and three modifications are made to accommodate the shared medium access nature of MANETs: (1) Pastry's node joining process is modified to use expanding ring search for locating a bootstrap node to join the network; (2) The original Pastry uses an expensive "ping" mechanism with a delay metric to measure and maintain the proximity of nodes in its routing tables. Since delay is affected by many factors and has a high variability in MANETs, we modified Pastry to use a hop count metric for proximity; (3) To reduce the cost of this proximity probing, we modified DSR to export an API that allows Pastry to inquire about the proximity values for nodes it is interested in. DSR can then use its cache to reply to "pings" from Pastry if there is a cached path to the node being pinged. In the absence of such a cached path, a ROUTE REQUEST is initiated by DSR.

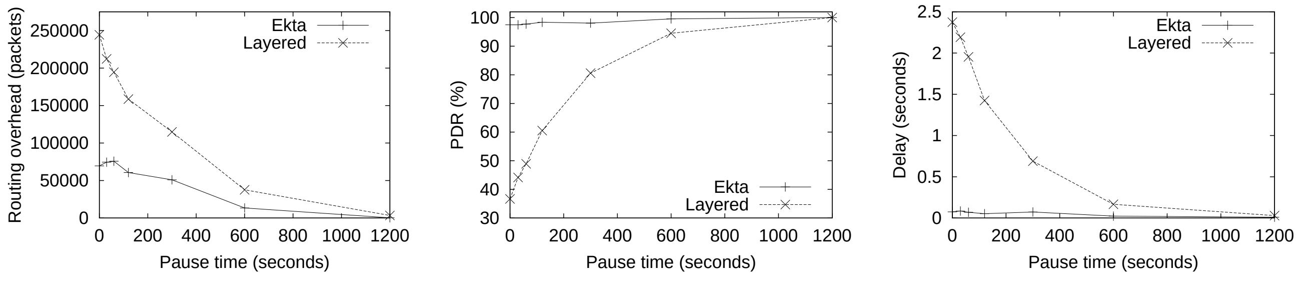
In summary, the layered design is similar to implementing a DHT in the Internet; it leverages the existing routing infrastructure for MANETs to the full extent. This approach, while consistent with the layered principle of the ISO model of networking, makes it difficult to exploit many optimization opportunities from the interactions between the DHT protocol and the underlying multi-hop routing protocol. For example, it is difficult for the routing structures of the DHT and the route cache of DSR to coordinate with each other to optimally discover and maintain source routes.

Integrated Approach In the integrated design, called Ekta, the DHT abstraction is supported by integrating Pastry and DSR at the network layer of MANETs via a one-to-one mapping between the IP addresses of the mobile nodes and their nodeIds in the namespace. With this integration, the routing structures of a DHT and of a multi-hop routing protocol, e.g., the route cache of DSR, are integrated into one structure which can maximally exploit the interactions between the two protocols to optimize the routing performance.

As in Pastry, Ekta assigns unique 128-bit nodeIds to nodes in a MANET by hashing the IP addresses of the nodes using a collision-resistant hashing function such as SHA-1.

The structures of the routing table and the leaf set stored in each Ekta node are similar to those in Pastry. The difference lies in the content of each leaf set and routing table entry: Each entry in Ekta's leaf set and routing table stores a source route to reach the designated nodeId. As in Pastry, any routing table entry is chosen such that it is physically closer than the other choices for that routing table entry. This proximity awareness is continuously maintained by making use of the vast amount of indirectly received routes (from overhear-

Student poster, *the 10th ACM International Conference on Mobile Computing and Network (MobiCom)*, Philadelphia, PA, September 26-October 1, 2004.



(a) Routing overhead (b) PDR (c) Delay

Figure 1: Routing overhead, PDR, and delay with varying mobility.

ing or forwarded messages) or by using prefix-based route discovery as described next in routing.

In Ekta, a message with a 128-bit key is routed using Pastry's prefix-based routing procedure and delivered to the destination node whose nodeId is numerically closest to the message key. When a route lookup for the next logical hop returns a next-hop node from the leaf set for which a source route does not exist, Ekta initiates a route discovery to discover a new source route. On the other hand, if the node selected as the next hop is from the routing table and does not have a route, a *prefix-based route discovery* is performed to discover routes to any nodes whose nodeIds match the prefix for that routing table entry.

Ekta uses a modified form of the Pastry join protocol to handle node arrivals. Its join and leave procedures are lightweight and incur message overhead only to maintain leaf set consistency which is required for DHT convergence.

Ekta inherits all of the optimizations on route discovery and route maintenance used by the DSR protocol. In addition, Ekta updates its routing table and leaf set using routes snooped while forwarding and overhearing packets, thus constantly discovering fresh and low proximity routes for the leaf set and the routing table entries.

3 Evaluation

We use ns-2 to evaluate the performance of the two design approaches. The "random waypoint" model is used in which 50 nodes move at a speed uniformly distributed between 1-19 m/s in an area of 1500m x 300m. A wireless radio with 2 Mbps bit rate and 250m transmission range is used. The communication pattern consists of 40 traffic sources, each initiating packets at the rate of 3 packets/second. Each packet has a 48-byte message body, prepended with a 128-bit key generated from hashing the message body. Thus, the effective packet payload is 64 bytes. This communication pattern models the traffic in a DHT-based storage system such as PAST [2].

Figure 1 compares the routing overhead, packet delivery ratio (PDR), and average delay of Ekta and *Layered* for all mobilities. The higher overhead in *Layered* can be attributed to the following reasons: (i) *Layered* employs periodic routing table maintenance (every 250 seconds), which causes extra route discoveries. Ekta uses overhearing of routes from physically nearby nodes to maintain proximity of its routing table entries. (ii) *Layered* selects a node as the next overlay hop irrespective of whether there are routes to that node in the DSR route cache, which causes unnecessary route discoveries. In Ekta, a node that has a valid source route is given preference and only when no such node exists, a prefix-based route discovery is issued. (iii) *Layered* selects a node as the next hop regardless of the relative freshness of its DSR source route compared to other candidates, since Pastry can not tell, which leads to an increase in ROUTE ERRORS.

A consequence of the high routing overhead in *Layered* is that its PDR drops and its delay increases as the mobility increases. In contrast, the PDR and the routing delay for Ekta remain largely constant as mobility increases. These performance results suggest integrating the functionalities of the DHT into the routing layer is much more efficient than having two independent layers with minimal interactions.

4 Open Questions

We used DSR as the representative MANET routing protocol to integrate with Pastry because source routing fits well with prefix-based routing. It remains interesting to see how to integrate DHTs with other MANET routing protocols. The use of hop-by-hop routing as in AODV and DSDV would require that the all nodes along a route to a destination maintain this route. This implies that these nodes need to have a prefix match with the destination nodeId in order to contain the destination nodeId in their prefix-based routing structures. It is also interesting to see how to efficiently integrate other DHTs such as CAN and Chord with MANET routing protocols.

References

[1] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic, 1996.

[2] A. Rowstron and P. Druschel. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proc. of ACM SOSP*, October 2001.

[3] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, November 2001.