Adapting Distributed Hash Tables for Mobile Ad Hoc Networks

Tobias Heer, Stefan Götz, Simon Rieche, Klaus Wehrle

{heer,goetzs,rieche,wehrle}@informatik.uni-tuebingen.de

Protocol Engineering and Distributed Systems Group
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen

Abstract

While unstructured P2P systems have been embraced widely in mobile ad-hoc networks (MANETs), the applicability of structured approaches like Distributed Hash Tables (DHTs) to such settings remains controversial. Existing research delivers promising empirical results addressing the concerns about performance, complexity, and reliability, but does not analyze the principles of combining DHTs and MANETs. This paper identifies and discusses the fundamental implications of non-infrastructural networks for DHTs and analyzes solutions to these challenges.

1. Introduction

Mobile ad hoc networks (MANETs) have attracted significant research attention, resulting in their applicability to a wide range of scenarios, such as support for disaster and emergency response teams, information and services in campus and mass event settings, or military use. Each of these settings centers around the basic demand for managing and exchanging data between participants without relying on an infrastructure of central components. Thus, the peer-to-peer approach to data handling and communication appears as a natural choice due to such properties as full decentralization, scalability, and fault tolerance.

MANETs form a challenging environment for managing data in structured P2P systems, and Distributed Hash Tables (DHTs) in particular. In comparison to unstructured approaches, these systems rely on more rigid constraints on the relationships between nodes and data items. These are difficult to achieve or enforce under the dynamic and volatile conditions of MANETs, reducing the stability and efficiency of DHTs. Despite these challenges, DHTs remain attractive in these settings due to the hash-based data model, their lookup efficiency, and where DHT-based applications are to be reused in MANETs.

The use of DHTs in MANETs instead of infrastructural networks has already been explored to some extent in existing literature [4]. Several different DHT approaches were successfully devised for or adapted to this environment, mostly based on optimized parameterizations and integration of underlay and overlay routing. This work has delivered promising results which demonstrate the feasibility of the approach. However, the proposed solutions in many cases apply to specific DHT systems and are based on observation through simulation.

To the best of our knowledge, the fundamental implications of the Internet-to-MANET transition for DHTs have not been thoroughly analyzed, leaving parts of the problem space and possible solutions untouched. This paper attempts such an analysis on the example of Chord [10], addressing it from three points of view: overlay networks as a whole (Section 2), the level of managing individual nodes (Section 3), and the implications on the routing level including cross-layer concerns (Section 4). We have chosen to focus on the Chord algorithm for its performance and flexibility [2] which allows various modifications that are essential for the use in mobile networks. Section 5 discusses related work and Section 6 concludes.

2. Overlay-level Adaptations

Figure 1 illustrates frequent failure scenarios in MANETs, in particular how individual nodes and groups of nodes can become unreachable. Although DHT algo-

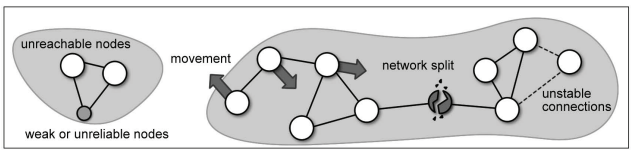


Figure 1. Connectivity issues prevalent in mobile ad hoc networks.

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)
0-7695-2520-2/06 $20.00 © 2006 IEEE

rithms are designed to handle node failure gracefully, they typically assume significantly more stable conditions of infrastructural networks. Thus, they focus on the failure of individual nodes, whereas MANETs can experience partitioning and merging frequently. This section discusses how such large-scale network splits can be handled at the overlay level.

2.1. DHT Discovery and Joining

A node which wants to join a DHT must establish a connection to a node which is already a member of the DHT-structure. Most DHT protocols solve this problem by publishing lists of participating nodes at well-known rendezvous points. While these do not exist in MANETs, broadcasts are inherently available and used by routing protocols. For this reason, limited broadcasts offer a good way to find a DHT in the same domain. If there is just a single type of DHT running on all nodes in the network and if every node is willing to participate in the DHT, a time-to-live (TTL) value of one hop suffices to discover a DHT because every node is either a DHT member or a potential member of a new DHT. In inhomogeneous networks, a higher TTL value is required to find DHTs. If several DHTs are discovered, the node needs to decide which one to join. This decision can either be based on the size of the DHTs or on the connectivity to the bootstrap node. Choosing the bigger DHT leads to fewer but larger DHT structures, whereas preferring a DHT with better connectivity will cause less network load. Assuming that the DHT node IDs are distributed uniformly, the number of nodes in a DHT can be estimated by examining the size of the segments managed by known nodes. The larger the segments are, the fewer nodes are contained in the DHT. A node which knows the segment sizes s_i of i nodes can estimate the number of nodes in the DHT $n_{estimated}$ with the following estimation function:

$$n_{estimated} = \frac{i}{s_1 + s_2 + \cdots + s_i} \cdot \alpha$$

α is the size of the DHTs address-space. The exactness of this estimation depends on number of known segments. Most DHT topologies require nodes to be only aware of the segment size of neighboring nodes. This leads to imprecise estimations due to a small number of samples. Additional segment size information of other nodes can be used in order to increase the number of samples. Sending this information piggy backed on DHT status messages reduces the additional overhead to a negligible amount. Moreover, a node can increase the precision of the estimation by comparing its results to those of neighboring nodes. Taking the average or median value of all known results leads to better results with a lower variance.

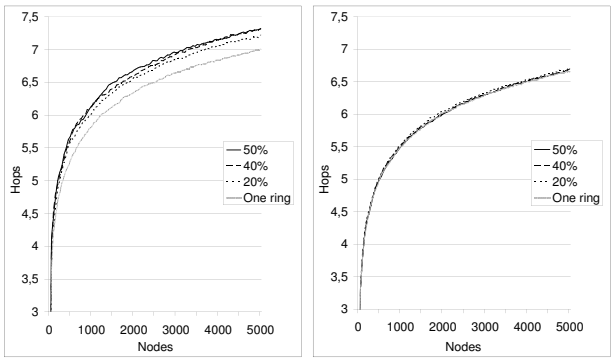
2.2. DHT Merging

Common DHT approaches assume that only one instance of a DHT—accessible to every node in a global network—exists. When DHTs are used in an ad hoc network, this assumption leads to the following problem: if two distinct DHTs (separated parts of one DHT or independent DHTs) meet in a network, suited methods—to either unite those structures or to enable communication between them—are necessary in order to make all available information accessible to every node. After a temporary network split, a merger is necessary to reunite the DHTs. Otherwise, further splits would create many small DHTs which are unable to handle requests efficiently. Since this is not a concern for existing DHT algorithms, approaches to merging DHTs must be developed to successfully use them in MANETs.

The simplest way of merging two DHT structures is to discard the smaller DHT-structure and let every node in this structure join the larger one. The size of both structures can be determined by using the estimation function proposed above. Simply discarding the smaller DHT structure imposes a vast amount of network traffic because all key-vale pairs of the smaller DHT need to be redistributed and all neighborhood connections must be re-established. The effectiveness of a merging scheme depends on the DHT topology. Therefore, different topologies require different merging schemes. As an example, the following merging scheme for the popular Chord protocol shows how ring-based DHTs can be merged efficiently. Despite its design for ring-based protocols like Chord [10] and Symphony [6], it can easily be adapted to other DHT protocols like CAN or Pastry, which operate on different topologies.

As shown in [10], every newly inserted node needs to establish $O(logN)$ connections and $O(logN)$ nodes must alter their routing tables which also requires $O(logN)$ connections to be established. Every new connection requires a lookup with a cost of $O(logN)$ hops. In reactive networks, every new connection causes route requests to be flooded in the network. In addition, every piece of information available in the smaller DHT must be redistributed. Depending of the applications which use the Chord ring, the number of key-value pairs managed by each node can vary between only few entries to thousands of pairs.

As an optimization, a node may keep its ID and all key-value pairs it manages. It also keeps all finger-table entries when it joins the new DHT. Only the successor list needs to be fixed in order to join the larger DHT. The key-value pairs that map to the preceding node need to be transmitted to the predecessor and the key-value pairs which are contained in the segment of the new node must be transferred from the successor. The main advantage of this approach is that the nodes do not need to establish new finger tables, nor do they need to redistribute all of their key-value pairs.



(a) One successor per node. (b) Four successors per node.

Figure 2. Average hop count after two DHTs have been merged using the proposed merge scheme. The smaller DHTs contained 50%, 40%, and 20% of all nodes. The values for a single ring are plotted for reference.

Because Chord tolerates moderate deviations from the optimal composition of routing tables, the resulting variance is acceptable as long as the sizes of both DHTs do not differ drastically.

Figure 2 shows the average hop count after a merger of two simulated Chord rings. Eight different configurations have been simulated. Figure 2(a) shows the average hop count for a ring merged from two rings with only one successor and Figure 2(b) shows the results with four successors. In both figures, two rings with 50 to 5000 nodes have been merged. The smaller DHTs contained 50%, 40%, and 20% of all nodes. A single ring containing all nodes was simulated for reference. The figure shows the hop count difference is negligible if more than 4 successors are used. The proposed merge scheme offers an efficient and simple way to merge two DHTs with low effort. It requires only one lookup to find the successor. The IDs of other successors can be provided by the new successor. However, it is still necessary to establish links to the new successors.

Figure 3 illustrates the procedure. Node 5 takes its new place between node 2 and node 17. It keeps its finger list and exchanges the key-value pairs with both neighbors. Its fingers point to node 40, which is located in the same part of the network.

3. Node-level Adaptations

A high churn rate, i.e., the rate at which nodes fail, join, and leave a system, is characteristic of MANETs. This section discusses how DHT management and repair mechanisms can be adapted to such conditions atypical of infrastructural networks.

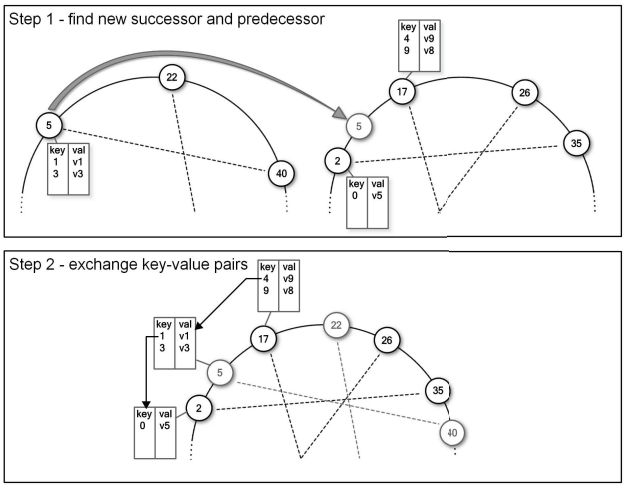


Figure 3. Simple merging scheme for Chord

3.1. Churn and Node Instability

In mobile ad hoc networks, node failures are caused by many reasons. Nodes may get separated from the network by node movement, environmental influences, or the failure of other nodes. In all these cases, the node is still operational and willing to join the network. This separation can be permanent or just temporary. Node 5 is not connected to the network at all time, but its connection is strong enough to have contact occasionally. Common DHT protocols initiate recovery mechanisms on the detection of a failed node. These mechanisms reach from simply removing the node from the routing tables to complex recovery measures like finding new entries for finger table and successor lists and copying stored backup information. As soon as the node reenters the network, the join procedure—which must be invoked in order to re-integrate the node—causes costly operations like topological adaptations, redistribution of key-value pairs and the exchange of information with the succeeding node. In a worst case scenario, a fluctuating node, which is partly available and unavailable over a longer period of time, can stress the whole network because of numerous join and leave procedures. If the topological structure allows the DHT protocol to delay recovery mechanisms without losing its routing capability, these costly recovery measures can be avoided by delayed recovery mechanisms. An increased level of redundancy can be used to achieve this goal. By replicating data items to successor nodes, these successors can answer queries for the absent node. Recovery mechanisms must only be initiated if the node does not reenter the network for a certain period of time. Especially when multiple nodes are fluctuating due to connectivity problems of a single node a lot of unnecessary overhead can be avoided.

If a link bridging two parts of the network is weak or un-

reliable, it is better not to use this link, but to divide the networks into two logical parts instead. Dividing the network into two logical parts will create two independent DHTs which need not deal with the symptoms of temporal partitions anymore. From a global viewpoint, there is no node which could decide whether a group of nodes is frequently separated from the network. Therefore, a distributed algorithm is needed to split the network. Due to the fact that the DHT has little knowledge of the actual underlying network structure, the recognition mechanisms must be located on the ad hoc network routing level.

One way to handle such temporary network partitions is to not use unstable network connections to route DHT messages. Therefore, unstable ad hoc network routes must be marked as impassable for DHT messages. Whenever a DHT message is about to be routed over such an unstable route, a stable route must be discovered for it instead. Otherwise, the source node must be informed that the destination is unreachable.

Every node can classify its direct neighborhood connections as stable or unstable by monitoring them for a while. A new connection may be classified as stable when the node has answered all pings for a certain time. If the connection is lost frequently, the route must be classified as unstable and therefore impassable. As a consequence, unstable network layer connections can not be used by the DHT and the DHT will experience a situation similar to a permanent network partition. Other applications which use the network may still use the weak connections as only DHT messages will not be routed across them. This approach implies a modification of the ad hoc routing protocol. Multi-hop routes which use unstable path segments need to be marked to avoid that DHT messages are routed towards a dead end.

This approach ensures that no DHT connections can be established over unstable routes. While there are stable routes between two nodes which are also connected over an unstable route, the unstable route may be used as long as the stable route is still present for fall-back purposes. A negative aspect of this approach is that not all information contained in the network is available, though it accessible over unstable links.

3.2. Node Ranking

There is a large variety of mobile devices which are capable of wireless ad hoc communication. Also, the resources available in smartphones, personal digital assistants PDAs, sub-notebooks, and notebooks vary significantly. But not only the different hardware of devices but also the position of a node in an ad hoc network topology has impact on its connectivity and reliability. These differences lead to heterogeneous networks, consisting of strong and weak nodes. Node failures reduce the stability of the overlay and trigger recovery mechanisms, imposing potentially significant load on the network. An early discovery of weak nodes enables proactive recovery measures which help to diminish the costs of sudden node failures. Indicators for such weak nodes are the duration and reliability of their network connection, their battery life, connection quality, or mobility. If the decision whether a node may join the DHT is based on these criterions, an increased stability can be achieved due to a DHT structure consisting of relatively reliable nodes. If free riding is not an issue or if measures can be taken against free riders, nodes which have been rejected for their low reliability could still use the DHT without being involved in the routing process. That way, even weak nodes can benefit from the DHT without impacting its stability.

The indicators described above can also be used to predict node failures caused by low energy and weak connections. Nodes which are likely to fail can inform their neighboring nodes in the DHT about their likely failure. Thus, costly recovery mechanisms can be avoided which leads to lower overhead of DHT maintenance.

This approach is similar to the supernode or superpeer approach used in many Peer-to-Peer topologies. The difference between the proposed approach and the supernode approach is that most of the nodes in a supernode network are connected through supernodes while only weak nodes are connected through other nodes.

In dense networks—where the transmission ranges of many nodes overlap—it is beneficial to not only avoid weak peers but to reduce the number of nodes participating in a DHT in a region. That way, the sparse bandwidth is not wasted by too many nodes performing DHT maintenance operations. Instead, a node can connect to the DHT through a supernode within range. By using a supernode approach, the used bandwidth can be reduced and interferences and retransmissions can be avoided.

4. Routing-level Adaptations

Structured overlay networks can be thought of as routing systems. Hence, they duplicate the functionality of the underlying ad hoc routing protocols to a certain extent. Without modifications, both routing instances act unaware of each other, resulting in an inefficient overall service due to overhead in terms of node and network resources. These aspects and possible solutions are discussed in this section.

4.1. Maintenance of Routing Tables

Keeping the routing tables of each node up to date is one of the main tasks of a DHT system. Efficient routing depends on routing information being current and consistent. Invalid entries cause unnecessary overhead because of misrouted messages and suboptimal routing. To avoid these inconsistencies, DHT protocols employ maintenance mechanisms to keep the routing tables up to date. Typically, nodes probe their neighboring nodes via periodic ping request and

response messages to learn whether they are still available or not.

4.1.1. Status Messages and Multicast

In many cases, periodic maintenance messages are sent to all nodes contained in the routing tables. Therefore, identical 'I am alive' messages are sent to a fixed group of nodes. As multicast communication is available in some ad hoc routing protocols like AODV or the On Demand Multicast Routing Protocol (ODMRP), these control messages can be transmitted via multicast in a more efficient way. Instead of sending numerous unicast messages, one multicast message can be used instead. Multicast messages consume less bandwidth and hence, less transmission energy. Multicast can make mobile DHTs more efficient because a smaller number of frequent control message transmissions are needed. However, the wide variety of MANET protocols makes a general statement on the use of multicast together with mobile DHTs impossible. Instead we will discuss the use of multicast for routing table maintenance on two examples: AODV and ODMRP.

Multicast groups need to be established and maintained while they are used. As a potentially very large number of multicast groups is necessary for DHT status message propagation—one group for every node in the DHT—the use of this approach must be considered carefully with respect to the characteristics of the underlying multicast protocol. AODV has a very efficient multicast management in which additional multicast groups only impose a small overhead. Multicast routes must be maintained and fixed whenever a topology change affects the distribution path. This overhead is mitigated by the need to also maintain the unicast connections for overlay routing. However, every node maintains a multicast routing table which is stored in the node's memory and must be searched for routing packets. As long as these additional memory and processing demands are not an issue, multicast messages can significantly decrease the maintenance overhead of the DHT in an AODV network.

ODMRPs multicast mechanism works differently. ODMRP source nodes (nodes which want to propagate multicast messages) send flooding-based broadcasts across the network to maintain the links to the receiving nodes. For this reason ODMRP does not scale well with the number of senders [5]. Due to the fact that using multicast for DHT protocol control messages implies a large number of source nodes, ODMRP is not suited for this purpose.

4.1.2. Status Information: Push vs. Pull

Periodic control messages like 'I am alive' messages can be triggered in two different ways. A node can actively request a status update from a distant node (Pull) or it can wait until the distant node sends such an update (Push).

The choice whether the pull or the push scheme is used, also determines if a DHT is suited for the use of multicast messages. Pushing messages can be done by using multicast messages as long as no acknowledgments for successful message deliveries are needed. Pulling messages requires a combination of multicast messages (pull request) and unicast messages (pull reply). As the pull reply is always a direct response to the request, it cannot be sent with multicast messages. For this reason the push method is better suited for the use in networks capable of multicast.

Many DHT protocols use the pull method but most can be easily modified to use the push method. The Chord protocol, for instance, requires the use of an additional data structure. In addition to the finger table and the successor list a references list becomes necessary. This list is used to store the addresses of all nodes which point to the respective node. Frequent heartbeat messages can be delivered to all nodes contained in this list by using the push instead of the pull method.

4.1.3. Route Discovery and Timing

Periodic maintenance traffic can stress the underlying network if the period between the messages is not chosen without consideration of the behavior of the underlying network. Reactive ad hoc networks like AODV build routes on demand. These paths are refreshed by messages which are routed over them. If a route is not used for a certain time, it is considered obsolete and hence is torn down. If the frequency of the DHT heartbeat messages is too low, routes become outdated before a new heartbeat message is sent. In order to transmit the heartbeat message, a new route finding request—which causes network flooding—is necessary. In the case of Chord, this situation can occur frequently due to the long time intervals between probing entries from the finger table in a round robin fashion.

There are two solutions for this problem. Either the DHT needs to reduce its update intervals or the routing protocol needs to keep routes longer. If the two intervals diverge too much, neither approach can lead to satisfying results. Either too many DHT messages are sent and thus the network experiences unnecessary stress, or many messages will be routed across faulty routes as a result of the long route timeout. If AODV is used, multicast routes can help to solve this problem. As multicast routes are maintained and refreshed, short timeouts are not an issue as long as the destination node is in the multicast group.

5. Related Work

Ekta [8] is an implementation of Pastry [9] which has been modified for the use in mobile ad hoc networks. The

authors present two approaches. The layered approach uses a combination of Dynamic Source Routing (DSR) [3] and a slightly modified Pastry, while the integrated approach removes the layering between Pastry and DSR. The integration of the DHT protocol and the route cache of DSR into one structure help to avoid overhead and increases the efficiency of both protocols in a mobile environment. IP addresses are directly mapped to pastry node IDs, enabling efficient routing to particular nodes. Ekta uses eavesdropping to learn about paths to close nodes. This knowledge is used to find close nodes within certain ID spaces for proximity routing and thus optimize the performance of Pastry. The work focuses on the efficiency of the routing procedure and does not address problems like network splits, mergers, and the existence of multiple DHTs. The simulation scenario which was chosen makes network splits impossible, and therefore does not allow making statements about the performance of Ekta in these extreme but common situations.

With CrossROAD [1], Delmastro also proposes cross-layer modifications to Pastry and the routing protocol, which are in part similar to those found in Ekta and exhibit similar restrictions. The experimental results for an eight-node network under modestly dynamic conditions show that tight integration of layers can almost eliminate the overhead stemming from overlay routing and maintenance.

The Chord Ad hoc Routing Protocol (CARP) [7] uses a spanning tree to build a ring topology as the basis for a Chord ring. The interval of the ID space handled by a node is determined by its position in the spanning tree and thus by its position in the underlying network. This enables Chord routing without an underlying network routing protocol. Node movements change a node's position in the spanning tree and therefore a node constantly changes its Chord ID while moving. This causes a continuous hand-over of key-value pairs in networks with higher node mobility. Due to the coupling of node position and Chord ID, the protocol does not distribute the IDs evenly across the Chord ring. The lack of finger tables leads to an average routing latency significantly larger than $O(logN)$ hops in a network of size N. In the worst case, CARP has a lookup latency of $O(N)$.

6. Conclusions

In mobile ad hoc networks, Distributed Hash Tables are faced with several challenges, which we analyzed in this paper. One major aspect is the interaction between two routing systems: the ad hoc routing protocol and the DHT routing algorithms. Since both systems attempt to handle topology changes and node failures concurrently, their interactions often result in suboptimal routing at high costs in terms of consumed network and host resources. Another main aspect is the high churn DHTs experience in MANETs, leading to DHTs being separated and reunited—network behavior atypical of infrastructure-based networks and hence rarely addressed in the original DHT algorithms. These drastic implications of running DHTs on MANETs make it necessary to adapt DHTs for efficiency and stability.

Based on our analysis, we discussed possible solutions to the MANET challenges. The integration of the DHT and ad hoc routing layers is crucial to achieve efficiency, as it significantly reduces the duplication of routing information and maintenance efforts. Furthermore, DHTs can partition and merge again so frequently that repair mechanisms for such events need to be an integral part of the system. Since large-scale MANETs aggravate the presented issues, they appear as an unattractive platform for DHTs which depend on persistent long-distance links between nodes. However, an approach similar to supernodes could allow even such scenarios by finding a stable core of the MANET for serving less reliable nodes. The evaluation of the solutions sketched here through simulation and experimentation is future work.

References

[1] F. Delmastro. From Pastry to CrossROAD: CROSS-layer Ring Overlay for AD hoc networks. In *Proc. of Workshop on Mobile Peer-to-Peer Computing - MP2P '05*, Kauai, Hawaii, 2005.

[2] R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity, 2003.

[3] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic, 1996.

[4] W. Kellerer, R. Schollmeier, and K. Wehrle. *Peer-to-Peer Systems and Applications*, chapter Peer-to-Peer in Mobile Environments, pages 401–417. LNCS 3485, Springer, Sept. 2005.

[5] T. Kunz and E. Cheng. On-Demand Multicasting in Ad-Hoc Networks: Comparing AODV and ODMRP. In *Proc. of the 22nd International Conference on Distributed Computing Systems - ICDCS'02*, Vienna, Austria, 2002.

[6] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems - USITS'03*, Seattle, WA, 2003.

[7] G. Porter, K. Lai, I. Stoica, and J. Condit. The Chord Ad-hoc Routing Protocol. Technical report, UC Berkeley, 2002.

[8] H. Pucha, S. M. Das, and Y. C. Hu. How to Implement DHTs in Mobile Ad Hoc Networks? In *Proc. of the 10th Annual International Conference on Mobile Computing and Networking - MobiCom'04*, Philadelphia, PA, 2004.

[9] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of International Conference on Distributed Systems Platforms - Middleware'01*, Heidelberg, Germany, 2001.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM 2001 Conference*, San Diego, CA, 2001.