Towards Scalable Mobility in Distributed Hash Tables

Olaf Landsiedel, Stefan Götz, Klaus Wehrle
Distributed Systems Group
RWTH Aachen, Germany
firstname.lastname@cs.rwth-aachen.de

Abstract

For the use in the Internet domain, distributed hash tables (DHTs) have proven to be an efficient and scalable approach to distributed content storage and access. In this paper, we explore how DHTs and mobile ad-hoc networks (MANETs) fit together. We argue that both share key characteristics in terms of self organization, decentralization, redundancy requirements, and limited infrastructure. However, node mobility and the continually changing physical topology pose a special challenge to scalability and the design of a DHT for mobile ad-hoc networks.

In this paper, we show that with some local knowledge we can build a scalable and mobile structured peer-to-peer network, called Mobile Hash Table (MHT). Furthermore, we argue that with little global knowledge, such as a map of the city or whatever area the nodes move in, one can even further improve the scalability and reduce DHT maintenance overhead significantly, allowing MHT to scale up to several ten thousands of nodes.

1 Introduction

Peer-to-peer networking has changed the way to store data distributed in a network. Peer-to-peer networks are self maintaining, resilient, and only need limited infrastructure and control. The development of structured peer-to-peer networks, e.g. DHTs, extends these ideas to high scalability, increased resilience and flat hierarchies.

Structured peer-to-peer networking provides a number of key properties enabling efficient data access in mobile ad-hoc networks: (1) commonly, ad-hoc networks have limited or even no infrastructure. Thus, a fully distributed and hierarchyless substrate – such as a DHT – is required for efficient data storage and access. (2) The high scalability of DHTs enables large mobile crowds. (3) Furthermore, the fragile ad-hoc network can benefit strongly from the redundancy and resilience provided by structured peer-to-peer networks.

However, the random movement of nodes in a mobile ad-hoc network makes it challenging to deploy a structured peer-to-peer network. In this paper, we address these challenges and introduce Mobile Hash Tables (MHTs), as a substrate for scalable mobile peer-to-peer networking.

DHTs map data items, i.e. key-value pairs, on node IDs. Commonly, a key is computed via a hash function from a string describing the corresponding data item. And a node ID is derived from its IP-address [16, 12] or its geographic position [13]. MHT – in contrast – introduces semantics to the keys: it derives a geographic position, direction, and speed from the key of a data item and stores this item on the node which matches these properties best. Thus, a data item is assigned a path along which it moves by being stored on a node moving along a similar path.

Apart from the mobility-aware DHT, a scalable and low overhead routing protocol is required to form the base for an efficient peer-to-peer substrate. Due to their limited scalability, classic reactive and proactive routing protocols are not sufficient for the support of large systems. We use geographic routing, as its routing decisions are done locally resulting in the necessary scalability. Furthermore, MHT itself relies on position information.

The remaining paper is structured as follows: Section 2 discusses the limitations of current mobile peer-to-peer technologies. Section 3 introduces the initial MHT design, and section 4 discusses various design extensions and their impact on scalability. Section 5 evaluates the performance of MHTs and section 6 concludes.

2 Related Work

Peer-to-peer communication has had a large impact in the Internet research community. Various structured [1, 3, 12, 16] and unstructured protocols – such as the well known file sharing tools – have been presented. The peer-to-peer paradigm has been extended to ad-hoc networks and even sensor networks [13, 6, 15].

Although their high scalability, resilience, and flat hierarchies make DHTs an interesting substrate for mobile network-

Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)
0-7695-2679-9/06 $20.00 © 2006 IEEE

ing, only a very limited number of approaches base on this principle. In this section, we discuss mobile peer-to-peer systems and their shortcomings and compare our work to them.

Most mobile peer-to-peer [8, 9] approaches deploy an unstructured peer-to-peer network on top of a ad-hoc routing protocol, such as AODV [10] or DSR [4]. Thus, they suffer from the limited scalability of ad-hoc routing protocols and the limited scalability of unstructured networks as both layers strongly rely on information flooding. Additionally, these approaches do not use cross-layer optimization techniques so both layers flood the network without being aware of each other.

Ekta [11] maps a DHT on an ad-hoc source routing protocol, requiring the underlying DSR to frequently set up and maintain routes to all entries in the DHT routing table. Furthermore, the DHT routing is not aware of the underlying topology, resulting in high routing and maintenance overhead. Additionally, all of today's DHTs require frequent discovery messages to test whether their routing entries are still valid. In comparison, our mobile hash table does not depend on this mechanism.

Although not built for node mobility, the geographic hash table (GHT) [13] is probably the concept most similar to our approach. GHT maps a key associated with a data item to a geographic location. Geographic routing is used to store and retrieve a the data item at its location. We generalize the ideas presented in this work to support mobile nodes.

The multi-level peer index (MPI) [7] extends the GHT approach from providing a specific geographic location to a spatial area. However, this approach requires location updates to be distributed in the entire network which severely limits the scalability of this approach.

3 Introducing Mobile Hash Tables

First, we describe the basic design of mobile hash tables. Later in Section 4, we present additional design optimizations, which increase scalability and performance significantly.

The main challenge for structured mobile peer-to-peer networking is to apply a structure to the unstructured and seemingly random node movements. Assuming that each node knows its position, speed, and direction, we show how a structured DHT can be set up.

To map data onto the moving nodes, we propose the following scheme: for each data item we derive a path from its key. A data item moves along its path and is stored on the node which moves on the most similar path. A path consists of two points, between which a data item moves back and forth, and speed information. As the path of each data item is derived from its key and the path is a loop, one can compute the position of a data item at every point in time. Thus, queries can determine a data item's position and be routed to it. In this paper we use a simple path consisting of two points, e.g. we derive the x and y coordinates of these two points from the key. The approach also allows for more complex mapping functions, e.g. a rectangular path, were the points of the rectangle are derived from the key. Alternatively, a key can denote multiple points in space from which a spline curve can be derived.

By comparing position, direction, and speed, it is determined which node carries a data item. The scalability of the proposed approach bases on the following observation: the more nodes are in an environment, the higher is the probability that a node with a path and speed similar to the path of the data item exists. Thus, the more nodes, the better matches exist. As result, data needs to be moved between the carrying nodes less frequently.

3.1 Routing in a MHT

Mobile hash tables are built on top of GPSR [5], a geographic routing algorithm for multi-hop wireless networks. We now briefly discuss design features of GPSR relevant for our work and then propose a minor extension to GPSR to integrate the mobile hash tables.

GPSR is highly scalable, as its routing only depends on local knowledge. In GPSR, packets are routed geographically, i.e. based on physical positions of packets and nodes. Packets to be routed are marked with their destination. Furthermore, each node knows its own position and those of its immediate neighbors. GPSR uses this local knowledge to route packets to their final destination. In its default operation mode, GPSR forwards packets greedily. Greedy forwarding fails, when a node has no neighbor closer to the final destination: the packet has reached a local maximum, e.g. a void. In this case, GPSR switches to perimeter forwarding and routes packets with the right-hand-rule around network voids. GPSR returns from perimeter routing to greedy forwarding when it reaches a node closer to the destination than the one at which it switched to perimeter routing (its position was stored in the packet).

3.2 Extension of GPSR

In GPSR, each node knows the position of its neighbors in one-hop distance and uses this information for its local routing decisions. In practice, each node regularly announces its position to the surrounding nodes with local broadcast messages. We extend this announcement by the current speed and the direction the node moves at. These values can be easily derived from GPS position samples. In MHT, we use this extended information to find a node which has a position, speed, and direction similar to the path of a data item.

3.3 Joining and Leaving an MHT

MHT bases on wireless communication so it benefits from its local broadcast properties. Thus, there is no need to find

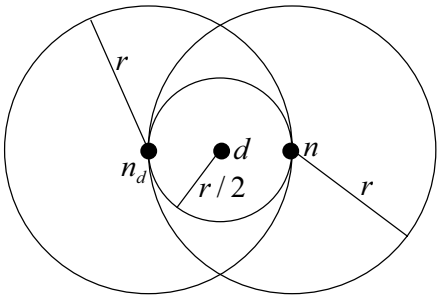


Figure 1. As long as the data item d is stored not farther away from its position than $r/2$, e.g. node n_d in the figure, a node n can reach node n_d when it comes into $r/2$ of the data d.

the ID space a node is responsible for via a search in the DHT as commonly in Internet-based systems. For mobile hash tables, it is sufficient that a joining node starts the regular local broadcast of its position, speed, and direction. Thus, its surrounding nodes recognize its existence and will consider this node for their routing and storage decisions.

To leave the system, the leaving node stops sending regular local broadcasts. Consequently, its surrounding nodes stop using it for routing or storage. In such a passive leave, the node does not announce its departure to its surroundings. However, the DHT's routing tables stays consistent after such a passive leave, as all decisions are based on local knowledge. No repair algorithm as in most Internet based DHTs – such as fixing finger tables in Chord – is necessary.

However, nodes may still try to route data via this node until their knowledge about this nodes times out. Furthermore, all data stored on this node and all messages it might have been forwarding at this moment are lost. Thus, next to the passive leave MHT supports a so-called active leave, the node announces its leave to its neighbors. This ensures that the leaving node is not considered for routing and storage anymore. Furthermore, it forwards all pending messages to their next hop and stores all data on the now best matching participant.

3.4 Data Placement

Before discussing lookups and data placement, we explain data movement in MHTs. Thus, for now we assume that a data item d is stored on a node n_d and we discuss how and when it is moved to another node n_{next}. Furthermore, we discuss how this node n_{next} is selected.

For simplicity's sake, we assume a circular communication range, e.g. a unit disk model, and that all nodes have the same communication range[1]. Let r be the communication range of a node, then a data item d needs to be stored on a node n_d not farther away than $r/2$ from the data item's position p_d – the position p_d is determined by the key which describes the data item d. This ensures that a node n in $r/2$ distance from p_d can communicate to n_d and retrieve the data item d (see figure 1). Let p_{n_d} be the position of the node n_d. Thus, when $|p_d - p_{n_d}| > r/2$, the data item d needs to be moved to another node to ensure that queries can successfully find the data item.

As the data item d has to be stored not farther away than $r/2$ from its position p_d, n_d selects the node n_{next} from its surrounding nodes. Since all nodes frequently announce their positions, directions, and speeds, to their neighbors, no explicit communication is necessary; n_d does a lookup in its neighborhood table. Among its neighbors, it selects the one node n_{next}, for which $|p_d - p_{n_{next}}| > r/2$ holds for the longest time. It uses the current speed and direction of data and nodes to predict future positions. When no node in range fullfills these requirements the data item is stored on the node which is closest to the data item's position. Thus, it maybe temporarily not reachable when this node is farther away than $r/2$ from the data item's position. In section 5 we evaluate the probability that a data item is our of place and thereby temporarily not reachable.

After discussing how the node which stores a data item is selected, data placement in the MHT is straightforward. A new data item is forwarded from its source to a node which is not farther away than $r/2$ from the data item's position. This node then determines the node to store the data item the same way that a new node for storage is selected.

3.5 Data Lookup

Data lookup, i.e. queries, use the same technique as the above described data placement. Knowing the data items key, the query compute the item's position. Thus, the query is forwarded from its source to a node which is not farther away than $r/2$ from the data item's position. A local broadcast from this node reaches the node carrying the requested data item, as the item itself is always on a node in $r/2$ or less distance from its position.

Although the data item can be found easily, sending the reply back to its source is not as trivial because the source moves along its own path. To find the source, the query and its reply store the position, direction, speed and ID of the source. Thus, nodes forwarding the reply can determine the source position at any time. However, the source might change its direction or speed at any time. When this happens, the source places a new (temporary) data item – a buoy – in the MHT. It is placed at the current position of the source and moves with the old direction and old speed of the source. Furthermore, it stores the new direction and speed of the source. Thus, the

[1] Please note that by introducing a factor α to the communication range, we can easily model heterogeneous communication ranges and non unit-disk models.

reply reaches the buoy instead of the the source and retrieves the new direction and speed of the source. Multiple direction changes are handled by chains of buoy.

4 Design Improvements

After discussing the basic design of the mobile hash tables in section 3, we present more details and performance enhancements.

4.1 Realistic Mobility

MHT focuses on systems with a high degree of node mobility. In the real world, nodes move along a limited set of paths, e.g. the roads of city. Thus, instead of arbitrary paths, we derive paths along the roads of a city from the data item's keys. In our work, we use the Manhattan grid as an example. As a result, the space and direction where nodes and data items move become strongly correlated. Thus, the chance of matching data and node paths increases significantly and MHT provides better performance for lower numbers of nodes.

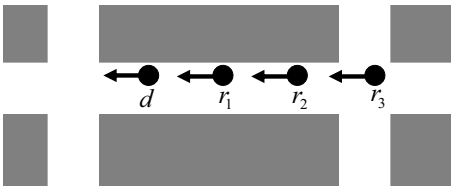
4.2 Traffic Adaptation

Furthermore, nodes in a city do not move at random speeds. For example, the cars on a road drive with similar speeds, as the they cannot pass each other arbitrarily. Thus, we now discuss a technique which describes how data items can adapt their speeds to the speed of the nodes surrounding it.

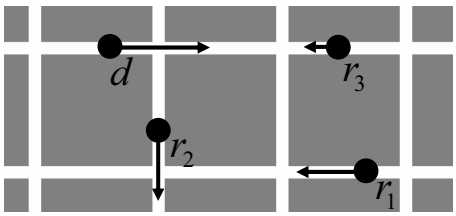
For this we change the information derived from the key. Instead of deriving the start and end points of a path, we derive a center point p, a direction dir, and a circulation time t for each data item d. Thus, from the center point p the data item moves in direction dir and turns around after $t/4$. It passes the center at $t/2$ and turns around the second time at $3/4 * t$. It determines its speed from the speed of the nodes on the road it currently moves on. Thus, while the circulation time is constant, the distance a data item moves is determined by the speed of the surrounding nodes. This approach reduces the need to move a data item to another node, as nodes and data items move with similar speeds.

Using traffic adaptation for data placement, queries also need to perform a similar traffic adaptation. From the circulation time, the query can predict on which side of the center a data items is and into which direction it moves. Furthermore, once the query reaches the street the item moves on, it can refine its prediction using the speed of the surrounding nodes. From this information the query can compute an "interception" course to find the data item and so resolve the query.

Concluding, this technique enables an adaption of the data movement to the movement of the surrounding nodes. This is very interesting for real world traffic situations because node speeds can vary highly. For example, a road accident can slow down or even stop traffic. The described traffic adaptation method allows data items to adapt their movement dynamically to changing situations. As a result, the occasions where a data items need to be moved from one node to another one are reduced. However, it becomes more complex to resolve a query, as the position of a data item needs to be predicted.



(a) Local Redundancy: adding replicas with the same path, but slightly different starting points.



(b) Global Redundancy: adding replicas with uncorrelated paths.

Figure 2. Local vs. global replication in the Manhattan grid.

4.3 Redundant Data Storage

In this section, we discuss two redundancy techniques to ensure the availability of data items: local and global redundancy.

To ensure local redundancy, MHT places replicas of each data item close their positions. We extend the information derived from the key by an offset for each replica. This offset is added to the start and end point – or the center point in case of traffic adaptation – of the data item. Thus, replicas and original data items move on the same path, i.e. the same direction and speed, just slightly apart from each other (see figure 2(a)).

A data item and its replicas need to exchange frequent "still alive" messages to test their availability. When an item is lost, a copy with the corresponding path information is created and placed in the hash table. As all items are close to each other, the "still alive" messages result in low overhead as they only need to be transmitted via a limited number of hops. For the same reason, updates to a data item have limited overhead. Local redundancy allows MHT to efficiently deal with sudden node death or departure. Furthermore, it allows some basic

load balancing as replicas and original data items can share the query load.

Nonetheless, local redundancy has some limitations. When a data item is very popular, queries can cause a high load on the routes to the data item. Global redundancy addresses this problem by placing a data item at various unrelated places in the network (see figure 2(b)). This ensures better load balancing and furthermore ensures that the network stays alive even when parts get disconnected. For global redundancy however, "still alive" messages need to be sent across large parts of the network and so increase the message overhead.

Concluding, both redundancy techniques have different trade-offs, which depend on the deployment scenario. It is beyond the scope of this paper to discuss these in more detail.

4.4 Data Locality

In some scenarios, it might be interesting to select the path of a data item manually when it is inserted into the network. For example, a restaurant might place its menu and other information on the road in front of it or in case of a traffic jam a warning message can be published in its area. Commonly, a data item's key is derived via a hash function from the string describing it. Alternatively, MHT allows the user to manually select a path for data items and to publish these properties in out-of-band media.

5 Evaluation

After discussing the initial MHT design and performance enhancing features, we evaluate the proposed technique. First we present our simulation setup and then discuss simulation results.

5.1 Simulation Model

We implemented the mobile hash table in the OmNet++ simulator [17] to evaluate the MHT performance and scalability. The mobility scenarios are based on the "random waypoint" model [4, 14]. When not denoted differently, between 1000 and 100000 nodes move with a speed uniformly distributed between 10 and 15 m/s in an area of 2000m x 2000m. The wireless radio has a transmission range of 100m and its propagation bases on the unit-disk model. The simulation duration is 1000s and results are averaged over three runs. Our simulation model ignores the capacity of, and the congestion in the network. Additionally, the model ignores packet losses and churn. While these assumptions are obviously unrealistic, they allow the simulator to scale to tens of thousands of nodes and us to evaluate the scalability of the proposed approach.

We evaluate the following performance metrics:

- Maintenance overhead: this metric evaluates how long a data item is stored on a node until their paths do not match anymore and the data item is moved to another node. This metric is the key metric of MHT, as it describes its scalability.
- Path length: this metric evaluates the hops it takes to resolve a query in the MHT.
- Data item out of place: when a data item needs to move to another node and there is no applicable node with similar path properties in range, the data item can move away from its path. Thus, queries fail until the data item is back on an appropriate route.

For the simulation results, we evaluate the performance of MHT in various scenarios: (1) node and data movement in the open space, (2) nodes and data move in the Manhattan grid, (3) nodes move in the Manhattan grid and traffic adaptation of the data items. Of the approaches for mobile peer-to-peer networking we are aware of only GHT scales up to several thousands of nodes. Thus, in the evaluation we compare our work to GHT.

5.2 Performance Results

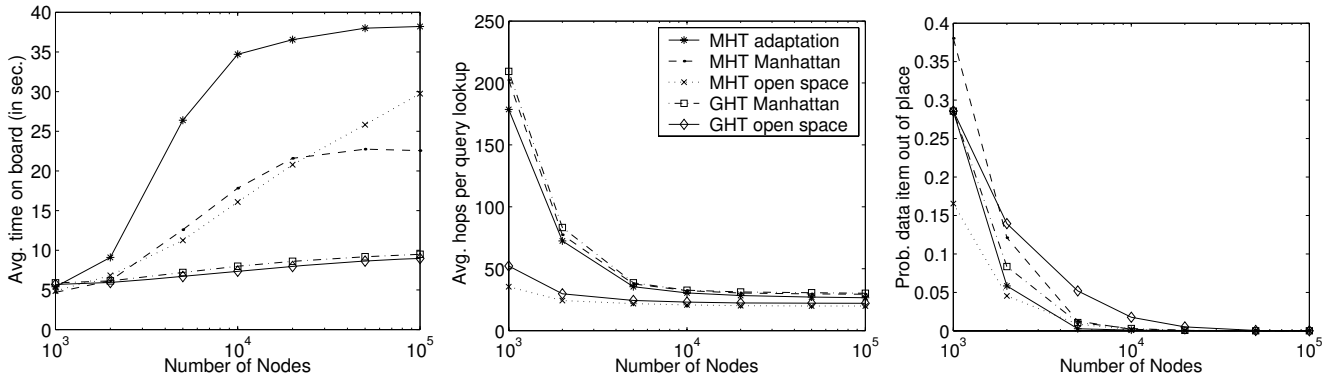
Varying Number of Nodes

The results for varying the number of nodes in the system are depicted in figure 3. With the raising number of nodes the node density increases. Thus, the probability increases that a data item finds a node with similar speed and direction to be stored on. As figure 3(a) depicts, the average time a MHT data item is stored on a node raises with the increasing number of nodes. Consequently, the more nodes participate in a system, the lower the maintenance overhead is. This is a very interesting system property, as MHT – in contrast to most other systems, including GHT – scales inverse with the number of nodes. GHT does not show these scaling properties as it does not benefit from node movement.

Furthermore, figure 3(b) shows that the MHT approach – and particularly not the adaptation mode – does not impact data lookup. The average number of hops to resolve a query of MHT is nearly equal to the hops of GHT. Figure 3(c) depicts the probability that a data item is stored farther away from its position than half the transmission range when queried and so cannot be retrieved via lookups. The figure shows that commonly for MHT this probability is lower than for GHT.

Varying Playground Size

To further evaluate the scalability, we varied the size of the playground from 1km x 1km to 10km x 10km. Figure 4(a) shows the average time a MHT data item is stored on a node. The figure shows interesting results, as MHT performs best on mid-sized playgrounds. For this we see two contributing factors. On the one hand, for small playgrounds, the average



(a) Maintenance overhead: average duration a data item is stored on a node.

(b) Query Lookup: number of hops to resolve a query.

(c) Probability that a data item is out of place when queried.

Figure 3. Evaluating the performance of MHT for varying node numbers. For comparison GHT performance is depicted, too. The playground has a size of 2000m x 2000m. Please note the logarithmic scale.

trip duration of a node is low and so a nodes changes its direction more often than on bigger playgrounds. Thus, the performance of MHT raises, when the playground gets bigger. On the other hand, the bigger the playground is the lower the node density is. This reduces the probability that a data items find a node with similar path properties. Additionally, figure 4(c) shows, that with increasing playground size the probability that a data is out of place when queried raises strongly for GHT, while it remains low for MHT.

Varying Node Speed

Figure 4(b) depicts MHT performance for various node speeds on a 7500m x 7500m playground. As the speed increases, the average node trip duration is decreased. Thus, nodes change their direction more often and so as stated above data items need to be moved to another node more often. However, MHT performs much better than GHT at all times.

Concluding the performance evaluation, the simulation shows that MHT performs about 5 to 10 times better than GHT depending on the simulation scenario.

6 Conclusion

Our work addresses two key problems of mobile peer-to-peer networking: efficient data lookup and scalable routing in a mobile environment. Our simulation results validate the scalability of the design – in a network with 100000 nodes it supports efficient data lookup and has low DHT maintenance overhead.

Certain additional problems remain to be addressed before deploying a MHT. One important problem is the design of a secure MHT which is resistant against denial of service attacks. Just like in the internet-based peer-to-peer network, nodes have two functionalities: they work as routers and servers. Thus, malicious nodes can attack the network at two structural points. We plan to evaluate how techniques already successfully deployed in internet peer-to-peer systems can be adapted to the mobile environment. However, it is interesting to notice that some threads, like the Sybil attack [2], are not existent in the MHT environment, as routing decisions and data storage are strongly bound to the physical network topology and node positions. Thus, a malicious node can only spawn identities at one geographic position in the network – its own physical location.

In this paper, we presented a scalable approach to structured peer-to-peer networking in mobile environments. We believe that mobile networks can strongly benefit from an efficient peer-to-peer substrate as it allows to store and retrieve data items without the need for additional infrastructure.

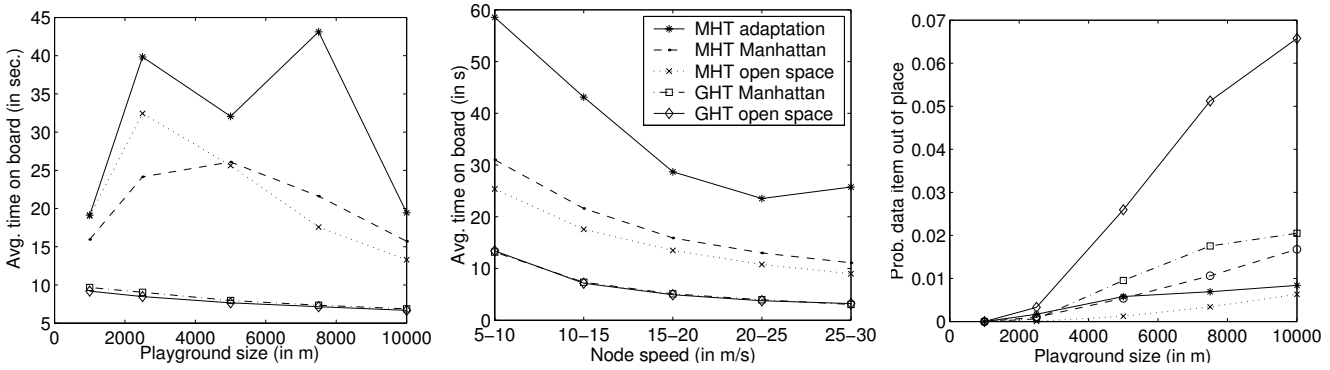
References

[1] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. 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*, November 2001.

[2] J. R. Douceur. The Sybil attack. In *Proc. of First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[3] N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proc. USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2003.

[4] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.



(a) Maintenance overhead for various playgrounds: average duration a data item is stored on a node.

(b) Maintenance overhead for various node speeds: average duration a data item is stored on a node.

(c) Probability that a data item is out of place when queried.

Figure 4. Evaluating the performance of MHT for varying playground sizes and different node speeds. For comparison GHT performance is depicted, too. The number of nodes is 100000. Due to the large number of nodes, we executed the simulations in this figure for 100s.

[5] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[6] O. Landsiedel, K. A. Lehmann, and K. Wehrle. T-DHT: Topology-Based Distributed Hash Tables. In *Proc. of 5th IEEE Conf. on Peer-to-Peer Computing (P2P)*, August 2005.

[7] M. Li, W.-C. Lee, and A. Sivasubramaniam. Efficient peer to peer information sharing over mobile ad hoc networks. In *Proc. of Second WWW Workshop on Emerging Applications for Wireless and Mobile Access (MobEA04)*, May 2004.

[8] C. Lindemann and O. Waldhorst. A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications. In *Proc. of 2nd IEEE Conf. on Peer-to-Peer Computing (P2P)*, September 2002.

[9] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proc. of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, October 2001.

[10] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, February 1999.

[11] H. Pucha, S. M. Das, and Y. C. Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proc. of 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, December 2004.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, September 2001.

[13] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets. In *Proc. of ACM Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.

[14] G. Resta and P. Santi. An analysis of the node spatial distribution of the random waypoint model for Ad Hoc networks. In *Proc. of ACM Workshop on Principles of Mobile Computing (POMC)*, October 2002.

[15] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-Centric Storage in Sensornets. In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, October 2002.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, August 2001.

[17] A. Varga. The OMNeT++ Discrete Event Simulation System. In *Proc. of the European Simulation Multiconference (ESM)*, June 2001.