Université de Neuchâtel
Institut d'Informatique

**Rapport de Recherche — RR-I-06-05.1**

# On the Feasibility of DHT Lookup in Ad-Hoc Networks

Raphaël Kummer, Peter Kropf, Pascal Felber

May 1, 2006

# On the Feasibility of DHT Lookup in Ad-Hoc Networks

Raphaël Kummer
raphael.kummer@unine.ch

Peter Kropf
peter.kropf@unine.ch

Pascal Felber
pascal.felber@unine.ch

Computer science department
University of Neuchâtel
Emile-Argand 11, CP 158,CH-2009 Neuchâtel, Switzerland

## Abstract

*Various peer-to-peer (P2P) architectures for ad-hoc networks have been proposed over the last few years. Most of them are unstructured and use some form of flooding to locate content, because the physical constraints of the underlying network make the construction of arbitrary application-layer overlays impractical.*

*In this paper, we study the problem of applying distributed hash tables (DHT) for mobile ad-hoc networks. Our approach to efficiently lookup content in such networks exploits physical proximity of peers when establishing and maintaining the DHT based routing tables. The efficiency of our method is demonstrated by simulation of large networks.*

## 1 Introduction

Peer-to-peer (P2P) systems, in which peer nodes form a cooperative network and share their resources (storage, CPU, bandwidth), have attracted a lot of interest lately. Roughly speaking, P2P networks can be classified as either structured or unstructured. Unstructured P2P networks (e.g., the Gnutella [1] and KaZaA [2] file sharing systems) have no precise control over the file placement and generally use "flooding" search protocols, which typically generate a large number of query messages. In contrast, structured P2P networks (e.g., Chord [14], CAN [12], Pastry [13], P-Grid [3]) use specialized placement algorithms to assign responsibility for each file to specific peers, as well as "directed search" protocols to efficiently locate files.

Structured P2P networks using deterministic algorithms are based on the distributed hash table (DHT) paradigm. Each data item is identified by a key (data identifier). The DHT maps keys to the nodes of an overlay network and provides facilities for locating the current peer node responsible for a given key. DHT designs differ mostly by the way they maintain the network and perform lookups: there is a fundamental trade-off between the degree of the network (the number of neighbors per node, i.e., the size of the routing tables) and its diameter (the average number of hops required per lookup) [15].

Directed search protocols are particularly efficient, because they accurately route queries toward the peers responsible for a given file. They require few communication stages generating only little traffic, and do not produce false negatives (i.e., the search fails only if there is no matching file in the system). Therefore, they are particularly attractive for systems with limited resources, where the cost of sending a message is non-negligible (e.g., in terms of energy) and flooding is considered prohibitive.

Mobile ad-hoc networks could greatly benefit from the efficiency of a directed search protocol to locate content. They do, however, suffer from limitations that make the maintenance of a DHT impractical. Most notably, a node in an ad-hoc network can communicate directly only with its physical neighbors, i.e., with the nodes that are within its (radio) communication range, and communication between remote nodes must traverse multiple intermediate nodes. DHTs assume that the peers form a fully connected network and that the neighborhood of a node can be chosen according to its location in a "logical" overlay, rather than to "physical" considerations. A simple mapping of a DHT design to an ad-hoc network would be unrealistic, as every node would need to keep track and communicate with a set of "logical" neighbors many steps away in the physical network. There is a fundamental gap between the logical proximity defined by the DHT overlay and the physical proximity imposed by the underlying ad-hoc infrastructure.

In this paper, we study the feasibility of DHT lookup in mobile ad-hoc networks. We propose a DHT design that combines a minimalist logical overlay structure together with adaptive routing mechanisms to quickly locate content. Nodes are organized in a logical ring, similarly to Chord [14] or Pastry [13], but we do not use logical long-

range neighbors whose maintenance costs would be prohibitive. Instead, we rely on the physical neighbors of the nodes traversed by requests routed through the physical network to quickly converge toward the destination in the logical overlay. We propose additional extensions, in which we consider an extra level of visibility in the physical neighborhood (neighbors of neighbors) and maintain a history of previous requests to dynamically identify and exploit possible shortcuts.

We have extensively studied our algorithm by the means of simulations in static deployment scenarios. We did not take into account mobility nor churn as we are mostly interested in evaluating the feasibility of DHT lookup in ad-hoc networks. Results demonstrate that our approach is indeed efficient and scales well to large peer populations. It represents a promising alternative to existing solutions based on unstructured P2P networks.

The remainder of this paper is organized as follows. Section 2 discusses related work. We detail our algorithm in Section 3 and present results from experimantal evaluation in Section 4. Finally, Section 5 concludes.

## 2 Related work

Peer-to-peer overlays have emerged from file sharing applications on top of the Internet, leveraging from its routing infrastructure and the intrinsic peer-to-peer property of the IP protocol. As already discussed in Section 1, the situation is different in the case an mobile ad-hoc networks. In particular, the DHT paradigm with its notion of regular topology (often a ring) and the shortcuts (fingers) introduced at the overlay layer, makes a direct mapping to ad-hoc networks difficult. Various approaches are known from literature to achieve such mappings.

While GRACE (Global Replication And Consistency) [5] has not been specificcaly designed with ad-hoc networks in mind, it enables for mobile collaboration by combining DHT properties with a layered architecture. GRACE supports mobility in wide-area networks. The different layers or consitency levels are interconnected through "consistency neighbors" that are logically close to each other. Requests are routed along these neighbors. The lookup algorithm of the system is based on Pastry [13]. This approach still relies on the standard Internet infrastructure.

Pucha et al. [11] implement Pastry on top of the Manet routing protocol DSR (Dynamic Source Routing)[6]. Three modifications are proposed as compared to the implementation on the Internet: (1) the node join precedure is modified by expanding the ring search for locating distinguished bootstrap nodes in charge of arrivals; (2) the Pastry ping metric is replaced by a distance metric to reduce network load; and (3) the DSR protocol is modified to inquire about the proximity used in the adapted Pastry routing.

Ekta [10] and MADPastry [16, 17] integrate the DHT paradigm with ad-hoc network routing. Both approaches introduce the necessary functions at the network routing layer. The principal idea of Ekta is to move the DHT protocol from the overlay level to the network layer of the Manet : a one-to-one mapping between IP-adresses and logical (DHT) node IDs is applied. MAPastry is built on top of the AODV protocol (Ad-hoc on-demand vector routing) [9]. This protocol aims to avoid full broadcasts as much as possible because these are costly in ad-hoc networks when targeted to the entire network. MADPastry creates clusters composed of physically close nodes sharing, at the same time, a common overlay prefix. The nodes in a cluster are thus physically and logically close. Routing is then based on the logial overlay node IDs.

All the aproaches above suffer from the size of the routing tables and the complexity of setting up and managing connections with all the nodes that they contain. In constrast, in our design, we only need to keep track of two remote nodes, as explained in the next section.

Finally, Cell Hash Routing (CHR) [4] is a specialised ad-hoc DHT. CHR uses position information to construct a DHT of clusters instead of organizing individual nodes in the overlay. This approach groups nodes according to their physical location. The routing between the clusters is done by position-based routing with the GPSR [7] routing algorithm. A major limitation of this approach is that nodes are not individually addressable, but only via the clusters.

## 3 The Ad-hoc Lookup Algorithm

A DHT system maps keys to nodes in a peer-to-peer infrastructure. Any node can use the DHT substrate to determine the current live node that is responsible for a given key. Every node and key has a specific position in a logical identifier space, and the mapping between keys and nodes is determined according to a proximity metric in the logical space. DHT nodes typically rely upon "long-range neighbors" to quickly route messages to remote locations in the identifier space. For instance, Chord [14] organizes the nodes in a logical ring. Each node is connected to its closest neighbors (successor and predecessor) in the identifier space. These connexions are necessary to guarantee successful routing: it is always possible to traverse the whole ring by following successor links, albeit at a very high cost. Each node additionally has a number of long-range neighbors, called fingers, that refer to nodes located at exponentially increasing distance in the logical space. Using these links, a node can quickly reach a remote location: the expected path length (number of hops) of a lookup is in $O(logN)$, where $N$ is the number of nodes in the system. Long-range neighbors are not necessary for the safety and reliability of the system (successor links are sufficient),

but they provide liveness properties that allow for efficient lookup.

In an ad-hoc network, nodes can directly communicate only with their physical neighbors, i.e., nodes that are within their communication range. Messages sent to remote nodes require multiple steps[1]. In the following, we assume that (1) the ad-hoc network forms a connected graph and (2) there is an underlying ad-hoc routing protocol that allows us to route messages between any two nodes. We do not make any assumption on the ad-hoc routing protocol, except that it always succeeds and that it is possible to take actions at intermediate nodes on the routing path (e.g., to change the course of a message). One may note, that when a DHT overlay is implemented over a classical Internet structure, the IP routing also takes care of delivering messages over multiple physical steps for one logical link (hop).

Mapping a DHT design such as Chord, Pastry, or P-Grid on an ad-hoc network would place nodes that are physically close at arbitrary locations in the logical space. Therefore, successor, predecessor, and long-range links would refer to remote nodes that can only be reached by multiple physical steps. Besides being inefficient in terms of physical path length, such an approach is impractical because each node would have to maintain accurate routing information for $O(logN)$ long-range neighbors. This can become a major problem when the size of the network grows or if nodes often become unreachable (due to disconnections or failures). Moreover, mobility and churn, though not considered in this study, make a direct mapping of a DHT design to ad-hoc networks unrealistic.

The intuition underlying our design is to only maintain a minimalist logical overlay without longe-range neighbors, that is responsible for safety only. For efficient lookup, we rather rely on the physical neighborhoods of the nodes traversed during lookup to spontaneously find long-range links in the logical space; therefore, we have a probabilistic form of liveness property because long-range links may be encountered on a random basis. Our conjecture is that lookup requests can be routed more efficiently and with much lower management costs than when deterministically maintaining $O(logN)$ long-range neighbors.

Similar to Chord or Pastry, we organize the nodes in a logical ring, as shown in Figure 1. The node responsible for a given key is the closest one in the identifier space. Each peer $n$ only needs to keep track at all times of its successor $succ(n)$ and predecessor $pred(n)$ on the ring, which imposes limited management overhead, only. Additional robustness can obviously be gained by considering several successors and predecessors, but we do not consider failures in this paper.

Each node is assigned a random identifier in the logical space, e.g., by hashing the node's IP address using a cryptographic hash function such as SHA-1. Therefore, the physical neighbors of a given node are expected to be randomly distributed in the logical space (see Figure 1). This diversity property is important for the efficiency of our lookup algorithm.

[1] In this paper, we denote by *hop* a logical link on the DHT overlay, and by *step* a physical link of the ad-hoc network.

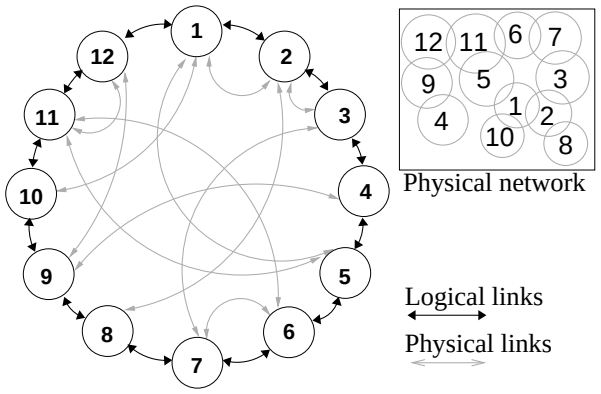


**Figure 1. Illustration of the DHT model for ad-hoc network.**

### 3.1 Basic lookup

We first describe the basic version of our algorithm, before discussing improvements to enhance lookup performance. The pseudo-code is shown in Algorithm 1.

In our approach, DHT lookup is closely integrated with the routing of messages through the ad-hoc network: when a message is sent to a remote node multiple steps away, it might diverge from its path at some intermediate if it finds a shorter way to its final destination.

Every lookup message contains the identifier of the key being searched (denoted by $k$ in the pseudo-code). In addition, when a lookup message is routed to a remote node across the physical network, it contains the identifier of its destination ($n_d$ in the pseudo-code).

Upon receiving a lookup message, a node $n_i$ proceeds as follows. First, $n_i$ searches among its physical and logical neighbors, as well as itself and $n_d$, which node is closest to the key (line 2). If $n_i$ is closest, then it is responsible for the key and it replies to the originator of the request (line 4).

If the request is on its way to a remote node $n_d$ closer to the key than any of $n_i$'s neighbors, we simply forward the message to the next node on the multi-step path to $n_d$ (lines 6–7).

Otherwise, the node $n_j$ closest to the key is part of $n_i$'s physical or logical neighborhood and will become the new destination. There are two cases to consider: if $n_j$ is a logical neighbor, we issue a multi-step request toward $n_j$

**Algorithm 1** Basic DHT lookup algorithm at node $n_i$ for key $k$

$\delta(k_1, k_2)$: distance between keys $k_1$ and $k_2$
$id(n)$: logical identifier (key) of node $n$
$V_i$: physical neighbors of $n_i$
$L_i = \{pred(n_i), succ(n_i)\}$: logical neighbors of $n_i$

```
1:  procedure LOOKUP(k, n_d)
    k : looked up key
    n_d : next logical hop
2:      n_j = arg min_{n ∈ V_i ∪ L_i ∪ {n_i, n_d}} δ(id(n), k)
3:      if n_j = n_i then            { We are responsible for k }
4:          return n_i
5:      else if n_j = n_d then       { Continue to n_d }
6:          n_k ← next step on physical path to n_d;
7:          send LOOKUP(k, n_d) to n_k;
8:      else if n_j ∈ L_i then       { Go to logical neighbor }
9:          n_k ← next step on physical path to n_j;
10:         send LOOKUP(k, n_j) to n_k;
11:     else                         { Go to physical neighbor }
12:         send LOOKUP(k, n_j) to n_j;
13:     end if
14: end procedure
```

(lines 9–10); otherwise, we directly send the request to that node (line 12). In both cases, we bypass the regular routing process and take a shortcut toward the destination key.

### 3.2 Convergence and termination

We shall now study the convergence and termination of our algorithm, under the assumption that the system is in a stable state with no node joining or leaving.

Let us consider node $n_i$ receiving a lookup request for key $k$, possibly on its way to a remote node $n_d$. To guarantee convergence, our algorithm tries to always reduce the logical distance to $k$. To that end, it looks among its neighbors, $n_d$, and itself, the node $n_j$ closest to the key.

First observe that, if $n_i$ is closest to the key, then it *must* be responsible for the key. Otherwise, either its predecessor or successor would be closer.

There are four cases to consider:

- If $n_i$ is closest to the key, then it is responsible for the key and the lookup ends.
- If a physical neighbor is closest to the key, then it directly receives the request.
- If a logical neighbor is closest to the key, then it becomes the new target $n_d$ of the request.
- If $n_d$ is closest to the key, then the algorithm does not change the destination of the request.

In the first three cases, the distance to the key decreases, either because we reach the node responsible for the key, or because we send the request to a node closer to the key. In the last case, we do not decrease the distance to the key but we proceed toward $n_d$ and, if the request reaches $n_j$, one of the first three cases will apply. The assumptions on the underlying ad-hoc routing protocol guarantees that a request sent to a node will eventually reach that node; thus, the last case cannot apply forever. It follows that our algorithm converges and terminates in a finite number of steps.

### 3.3 Increasing visibility

The visibility of a node, in terms of physical neighborhood, is limited by the communication range. As nodes can communicate directly with their physical neighbors, a simple improvement is to have nodes exchange information about their neighborhood. That way, visibility gets extended to the "neighbors of neighbors" (NoN).

This extension has a limited cost in terms of message overhead, because NoN lists are sent only upon change and require a single broadcast. Yet, it has the potential to significantly improve lookup efficiency because it increases the probability of finding a suitable long-range link while routing requests across the ad-hoc network.

### 3.4 Exploiting request history

Ad-hoc networks are characterized by the fact that every node must participate in request routing and forwarding. Therefore, a node sees traffic for which it is not the target. We can thus exploit information about past requests to acquire knowledge of remote nodes and improve lookup efficiency.

Each node maintains a cache that stores, for each observed request, the searched key $k$ and the destination of the message $n_d$. Thereafter, the cache entry $(k, n_d)$ can act as a long-range neighbor: when a node receives a request for a key closer to $k$ than the node that would be selected by Algorithm 1, the request is redirected toward $n_d$ instead.

This extension does not require any extra message to be sent. Further, as all the physical neighbors of a node can listen to its messages (radio communications are broadcast), information about past requests can be gathered passively. The cache uses a least-recently used replacement policy and we have limited its size to 256 entries in our implementation.

# 4 Evaluation

## 4.1 Experimental setup

An experimental system has been implemented to analyze the efficiency of the ad-hoc lookup protocol and to demonstrate the feasibility of exploiting the known performance of the DHT lookup paradigm when applied to ad-hoc networks.

The simulated system is an overlay network of a fixed number of nodes arranged in a rectangular area as physical space. Recall that our study neither targets to investigate mobility nor churn. Therefore, the experimental setup considers only static scenarios.

The nodes are randomly distributed in the physical space and are randomly assigned their identifiers in the DHT space. To take into account the limited radio range of a node in the ad-hoc network, two nodes are only considered to be connected if their separating distance (relative to the dimensions of the rectangular area) is smaller than a maximum range limit. A density parameter allows to control the average number of connections of a node in the system.

To forward a request to the next hop in the logical space, appropriate routing in the physical space needs to be simulated. The implemented simulation routing strategy is similar to ad-hoc routing algorithms such as LAR [8]. The positions of all nodes are known to the simulator, which allows to refrain from implementing routing tables. When a node receives a request to be forwarded to some destination node at the logical level, it chooses as the next step the node in its physical neighborhood that is the closest to the position of the the (logical) destination node. The network generation described above allows for degenerated situations where this routing process does converge to the destination (e.g., when reaching a "dead-end"). To cope with this situation and avoid the complexity of simulating a complete ad-hoc routing protocol, a bypass method is used in which the request is directly delivered to the destination node. For such a communication, the current average number of steps for the corresponding logical link is accounted for in the statistical analysis.

The experiments have been conducted for different network sizes and the different variations of the ad-hoc lookup algorithm.

**A. Simulation setup**

**Network size :** 1,000, 10,000, 100,000.

**Connectivity :** the average number of physical connections of a node (network density) varies between 13 and 16.

**Lookup requests :** for each experiment, the paths of at least 2,000 randomly generated requests are statistically evaluated.

**Steady state :** to evaluate the variants of the ad-hoc lookup algorithm using the caching mechanism, the simulation runs a *warm-up* phase during 2,000 (2K), 50,000 (50K), or 100,000 (100K) requests in order to reach a steady state before the statistical information is collected for the analysis.

**B. Simulation experiments**

**Basic :** the basic DHT lookup algorithm

**Neighbors-of-neighbors (NoN) :** the basic algorithm evaluating the physical neighbors and their physical neighbors to choose the next step.

**Cache (C) :** the NoN algorithm using a cache memorizing previous forwarding choices.

**Warm-up (Wup) :** the simulation system executes a warm-up phase to reach a steady state prior to issuing the analyzed requests.

The data used for the analysis is collected by the requests themselves along their itinerary through the ad-hoc network. This allows us to also analyze the paths through the physical network for individual requests and hence compute average values.

## 4.2 Discussion and Results

Each request is evaluated at each node independently of whether the node is an intermediate node of the logical path or an intermediate node for reaching a node on the logical path. In either case, the algorithm tries to forward the request to the node approaching the most the destination. Logical paths may thus be altered by the algorithm. Table 1 shows the percentages of altered paths (or shortcuts taken) determined during the different simulation scenarios. The versions using the cache use shortcuts for more than 50% of the requests. For large networks, however, this percentage decreases. This is not surprising, because the density (average number of physical connections) is the same for all networks. In case of large networks, there are hence less possibilities for profitable shortcuts.

One observes that the average values for networks of size 1,000 and 10,000 are roughly the same when applying the warm-up phase. This suggests that our lookup algorithm provides a stable lookup time in average after some running time (warm-up phase in the simulation), and up to some network size.

| Network size | Basic | NoN | C Wup-0 | C Wup-2K | C Wup-50K | C Wup-100K |
|---|---|---|---|---|---|---|
| 1,000 | 38% | 37% | 51% | 54% | 54% | 54% |
| 10,000 | 39% | 41% | 49% | 56% | 59% | 59% |
| 100,000 | 35% | 37% | 42% | 46% | 53% | N/A |

**Table 1. Average use of logical shortcuts.**

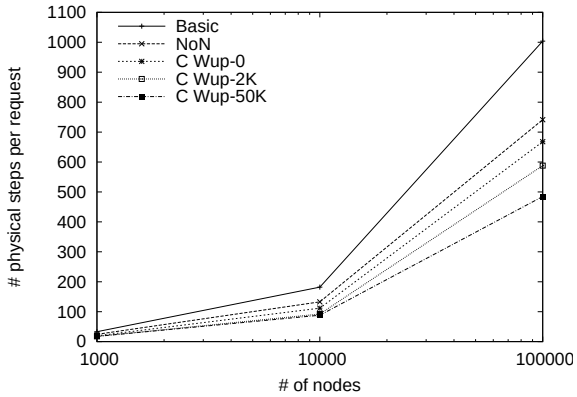


**Figure 2. Average number of physical steps to achieve a request**

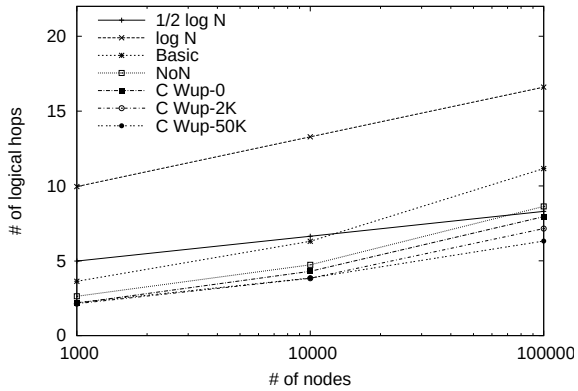


**Figure 3. Average number of logical hops per lookup**

Figure 2 shows the average number of physical steps required to serve a request. The use of the NoN and cache with warm-up strategies reduces the number of physical steps by $\sim 20\%$ respectively $\sim 50\%$. The introduced logical shortcuts can thus be considered as very effective.

| | Basic | NoN | C Wup-0 | C Wup-50K |
|---|---|---|---|---|
| 1,000 | 9.07 | 9.22 | 8.77 | 7.84 |
| 10,000 | 28.45 | 28.06 | 25.94 | 24.09 |
| 100,000 | 89.89 | 85.80 | 83.95 | 82.16 |

**Table 2. Average cost of a logical hop in terms of physical steps**

The average costs of one logical hop in terms of physical steps are roughly the same for all variants of the algorithm (see Table 2). The performance of the lookup algorithm increases thus when the number of logical hops for a request decreases. Figure 3 demonstrates that our ad-hoc lookup algorithm indeed reduces the number of logical hops to serve a request.

We observe that our algorithm always achieves the lookup in less than $\log N$ hops or even less than $\frac{1}{2} \log N$ hops. This clearly demonstrates the efficiency of the approach and confirms that the introduction of the cache further improves performance. Moreover, the various warm-up phases applied show that the ad-hoc lookup algorithm stabilizes over time: the longer it runs, the better it performs. There exists of course a limit where no further improvements will be possible when running the system longer and longer. To gain insight to the complex question how, whether and under which conditions (network topology) this limit remains a difficult graph theoretic question. Our empirical investigation suggests only that our algorithm exposes a stable behavior over time. Moreover, it must be remembered that the average search costs using the logical links only has a performance of the order of $O(N)$. In comparison, Figure 3 clearly shows the superiority of exploiting the properties of the physical links in the algorithm.

The ad-hoc lookup algorithm finds logical shortcuts through the selection of physical nodes closest to the destination and with the help of the information gained from previous requests kept in the cache. Intuitively this suggests that shortcuts determined at the beginning of a logical path should approach the destination the most. Indeed, the probability that some node in the physical neighborhood is closer to the destination node than the next node on the logical path decreases with the number of logical hops traversed. In other words, the percentage of logical hops that terminate without a shortcut being taken increases when

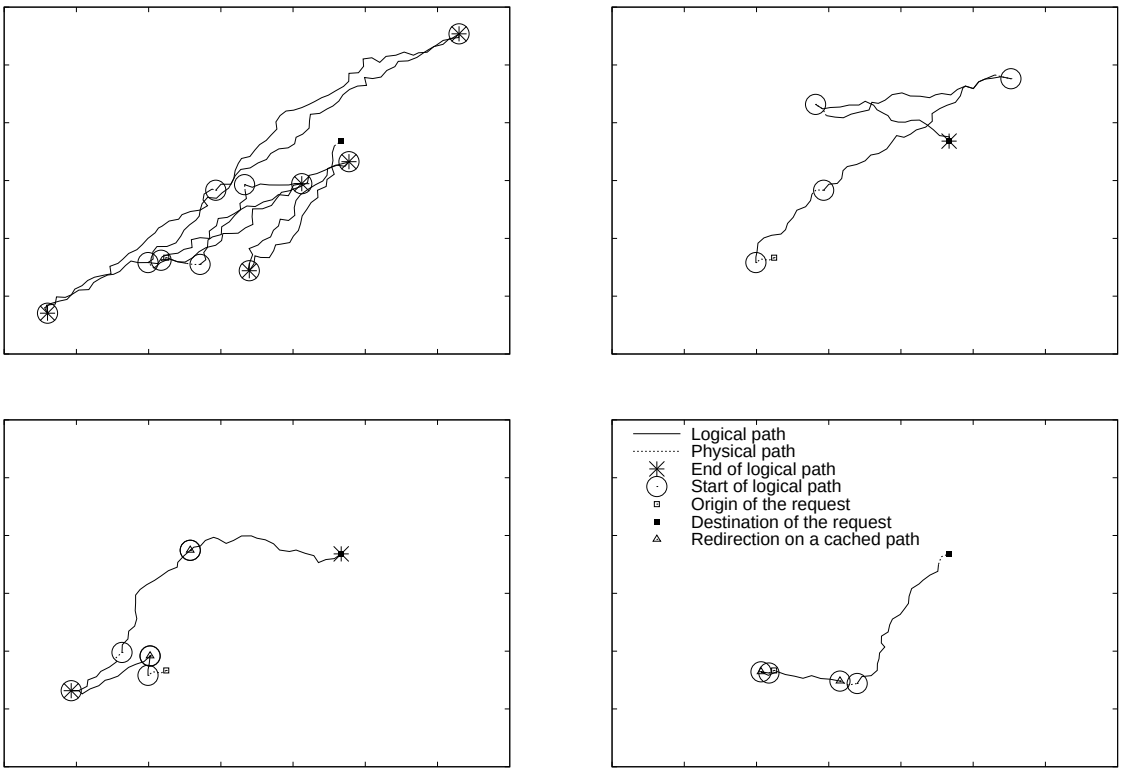


**Figure 5. Path of a request with Basic (top left), NoN (top right), C Wup-2K (bottom left), and C Wup-50K (bottom right).**

approaching the destination. This characteristic is demonstrated in Figure 4. The figure shows the percentage of terminated logical hops as a function of the distance from the node responsible for the searched key.

Finally, Figure 5 shows the path of the same request in the same network of 10,000 nodes but with different versions of our algorithm. Coordinates represent the geographical position of the nodes. Clearly, the best result is obtained when using the NoN version of the algorithm with caching.

## 5 Conclusion

Apart from very few known approaches, P2P systems for mobile ad-hoc networks have essentially used unstructured designs and flooding-based search protocols. It is indeed challenging to build a structured overlay with a directed search protocol on top of an ad-hoc network, because nodes have only limited communication capabilities: they can exchange messages only with the nodes that are within their (radio) communication range.

In this paper, we have studied the feasibility of building a DHT for mobile ad-hoc networks. The premise of such a design is to improve lookup performance and reduce the message overhead as compared to flooding-based protocols which yield linear or worse performance. In our approach, we only maintain a minimalist logical overlay structure but rely on the physical neighbors of the nodes traversed during ad-hoc routing to quickly reach the object being looked up. Our simulation results have demonstrated that a sub-linear lookup performance can be preserved when relying on a mobile ad-hoc network infrastructure rather than on the standard Internet. It must however be noted that, to the best of our knowledge, the observed sub-linear performance still has to be confirmed by an analytical study.

## References

[1] The Gnutella web site, www.gnutella.com, 2006.
[2] The KaZaA web site, www.kazaa.com, 2006.
[3] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Record*, 32(3):29–33, 2003.
[4] F. Araùjo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri. CHR: a distributed hash table for wireless ad hoc networks.

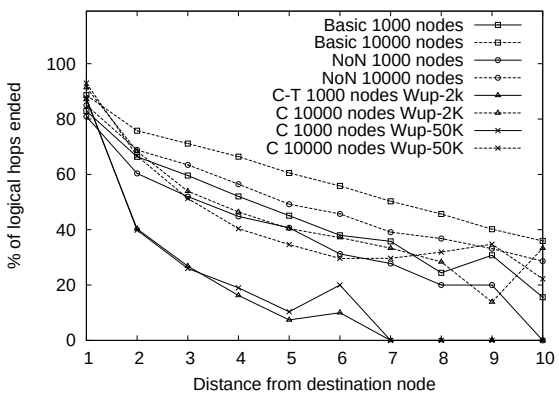


**Figure 4. Percentage of terminated logical hops**

pages 407–413. 25th IEEE International Conference on Distributed Computing Systems Workshops, June 2005.

[5] A.-M. Bosneag and M. Brockmeyer. GRACE: Enabling collaborations in wide-area distributed systems. pages 72–77. 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), 2005.

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

[7] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom'00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[8] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, 2000.

[9] C. E. Perkins and E. M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *WMCSA*, pages 90–100, 1999.

[10] H. Pucha, S. M. Das, and Y. C. Hu. EKTA: An efficient DHT substrate for distributed applications in mobile ad hoc networks. pages 163–173. 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004), December 2004.

[11] H. Pucha, S. M. Das, and Y. C. Hu. How to implement DHTs in mobile ad hoc networks? the 10th ACM International Conference on Mobile Computing and Network (MobiCom), September 2004.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[13] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[15] J. Xu, A. Kumar, and X. Yu. On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. *IEEE Journal on Selected Areas in Communications*, 22(1):151–163, January 2004.

[16] T. Zahn and J. H. Schiller. MADPastry: A DHT substrate for practicably sized MANETs. 5th Workshop on Applications and Services in Wireless Networks (ASWN2005), June 2005.

[17] T. Zahn and J. H. Schiller. DHT-based unicast for mobile ad hoc networks. pages 179–183. Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops(PerComW'06), March 2006.