Performance Evaluation of a DHT-based Approach to Resource Discovery in Mobile Ad Hoc Networks

Thomas Zahn
INRIA
Rocquencourt, France

Jochen Schiller
Institute of Computer Science
Freie Universität Berlin, Germany

Abstract – **Recently, approaches to DHT-style key-based routing explicitly designed for the use in mobile ad hoc networks (MANETs) have been proposed. However, as DHTs are usually no applications as such, a very practical distributed application running on top of such mobile DHTs could be resource discovery.**

Therefore, in this paper, a DHT-based approach to resource discovery – MAPNaS – will be described in detail and thoroughly evaluated through simulations. We aim at gaining a first insight into the question under which network conditions it is meaningful to perform DHT-based resource discovery and when it would be more advisable to opt for a less complex – e.g. a simple broadcast-based – approach instead.

Index Terms – **DHT, MANETs, Peer-to-Peer, Service discovery**

I. Introduction

The invisible omnipresence of the Domain Name System (DNS) in the Internet shields one of the most fundamental challenges from network applications and users: How to bind a resource, for example a file or a service, to a specific network address. It is, in fact, essential for a network application to resolve a given resource (e.g. by its name) to the concrete network address of the node where the desired resource actually resides.

In order to cope with scalability issues associated with static DNS servers, a number of peer-to-peer based name services have been proposed recently for the domain of the Internet: [13, 3, 4, 19, 1]. Instead of the static and hierarchical DNS infrastructure, these approaches use structured P2P networks, also known as DHTs ([14, 17, 15, 23]) to efficiently distribute and discover resources in the network.

MANETs are highly dynamic and self-organizing networks that are formed among wireless mobile devices. Due to this lack of a fixed infrastructure, there are no dedicated resource directories available in MANETs. Obviously MANETs and P2P networks share a good number of key characteristics, hence, it would be intuitive to deploy the P2P-based name services mentioned above in MANETs to provide resource discovery. However, those approaches rely on DHTs designed for the Internet that are ill-suited for the use in MANETs [21].

Therefore, in the short paper [22], the concepts of a DHT-based approach explicitly designed for resource discovery in MANETs – MAPNaS – were briefly outlined. In this paper, MAPNaS will be described in detail and its performance will be evaluated thoroughly in various network environments. We will try to answer the question under which conditions it is worthwhile to perform DHT-based resource discovery and when it might be more advisable to opt for a less complex (for example a broadcast-based approach with practical no maintenance overhead) approach.

The remainder of this paper is organized as follows. Section II provides a brief overview of the mobile DHT substrate MADPastry on which MAPNaS is based. Section III describes in detail the concept and architecture of MAPNaS. In Section IV, extensive simulation results are presented for MAPNaS and a reference broadcast application. Section V discusses related work. Finally, Section VI concludes this paper and provides a brief outlook on our future work.

II. MADPastry – Brief Overview

A large body of work exists on direct routing in mobile ad hoc networks: [5, 7, 8, 11] to name but a few. These ad hoc routing protocols deliver a packet from a source node to a predefined destination node. However, indirect routing – or key-based routing – differs from direct routing in that packets are no longer routed based on the destination node's address but on a key instead. The packet is then to be delivered to the node that is responsible for the packet's key. In other words, the actual address of the final destination node is usually unknown to the sender. For this purpose, MADPastry (Mobile Ad Hoc Pastry) [21] has been proposed.

MADPastry is a DHT substrate particularly designed for mobile ad hoc networks. It combines AODV ad hoc routing [11] and Pastry overlay routing [15] at the network layer to provide an efficient primitive for key-based routing in MANETs.

Abiding by the concept of structured P2P overlays (DHTs), each node in a MADPastry network assigns itself a unique overlay ID (for example by hashing its IP address, etc.), which defines its logical position on the virtual overlay ID ring [15]. Furthermore, in MADPastry, a message's packet header contains a message key. MADPastry then routes the message to that node in the network that is currently responsible for the message key – i.e. to the node whose overlay ID is currently the numerically closest to the message key among all MADPastry nodes in the network. To avoid message broadcasts whenever possible (e.g. for route discovery), MADPastry explicitly considers physical locality in the construction of its routing tables.

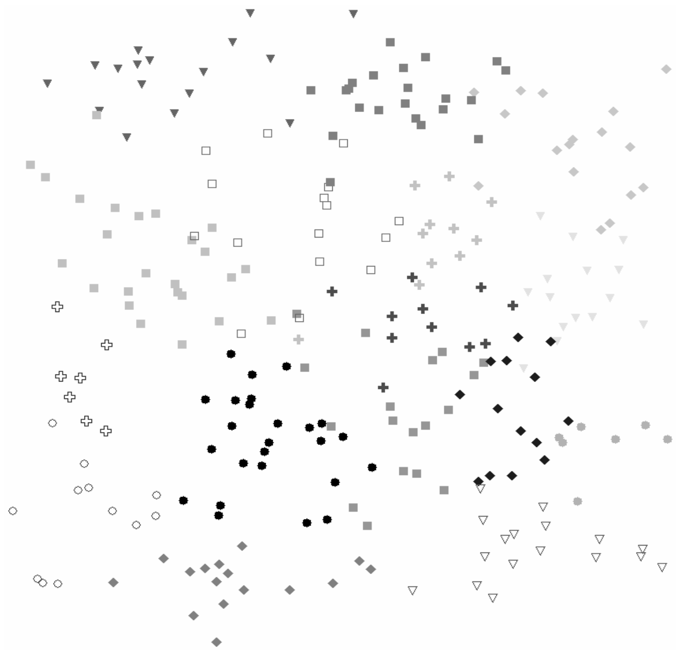
Figure 1. Spatial distribution of overlay ID prefixes

Clusters. Standard (Internet-based) DHTs are largely oblivious of the actual physical topology so that two overlay neighbors can be located arbitrarily far from each other in terms of the underlying physical network. This can lead to a large overlay stretch (i.e. the ratio between the length of the physical route traveled during an overlay key lookup compared to the direct physical path from the source to the eventual target node) as subsequent overlay hops can literally crisscross the physical network. Due to the volatile nature of physical routes in MANETs, this effect is especially prohibitive in such environments.

To exploit physical locality in the construction and maintenance of its overlay, MADPastry uses Random Landmarking [20]. Instead of having fixed landmark nodes – which simply are not available in MANETs – fixed *landmark keys* are used. These keys divide the logical overlay id space into equal sections (e.g. 16 keys with hexadecimal ids "0800…000", "1800…000", "2800…000", ... , "E800…000", "F800…000", etc.). The nodes whose overlay IDs are currently numerically closest to the landmark keys temporarily become landmark nodes and periodically issue beacon messages. Nodes overhear these beacon messages and periodically determine the physically closest temporary landmark node (e.g. in terms of hops). If need be, a node assigns itself a new overlay ID sharing the same prefix with the closest temporary landmark node. It would then (re-)join the network under its new ID. This leads to physically close nodes forming overlay regions, or clusters, with common id prefixes. In other words, nodes that are close to each other in the logical overlay ID space are also likely to be close to one another physically. This is demonstrated by Figure 1 which shows the spatial distribution of overlay ID prefixes in a 250 node MADPastry network. Equal symbols of equal shades represent equal overlay ID prefixes.

Routing Tables. MADPastry maintains three different

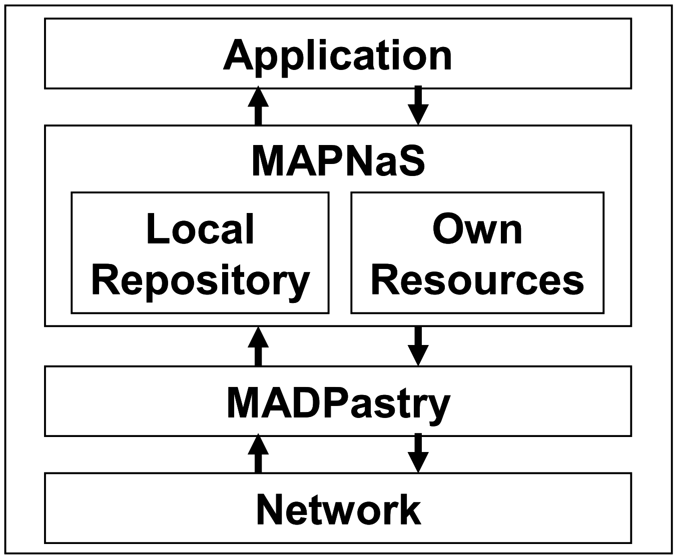

Figure 2. The MAPNaS architecture.

routing tables: a standard AODV routing table for physical routes from a node to specific target nodes, as well as a sparse Pastry routing table and a standard Pastry leaf set for indirect routing. The Pastry routing table only needs to contains as many entries as are necessary to keep a "finger" entry into each MADPastry cluster (i.e. one entry for each distinct cluster overlay prefix).

Routing Table Maintenance. To avoid the prohibitive overhead induced by routing table maintenance, the only proactive routing table maintenance that a MADPastry node performs is the periodic pinging of its "left" (i.e. the node who has the largest overlay ID smaller than the node's own) and "right" (i.e. the node who has the smallest overlay ID larger than the node's own) leaf as this is necessary to guarantee overlay routing convergence. All other routing entries are gained or updated implicitly by overhearing data packets.

Routing. MADPastry routes packets based on a key. When a node wants to send a packet to a specific key, it consults its Pastry routing and/or leaf set to determine the closest prefix match, as stipulated by standard Pastry. Next, it consults its AODV routing table for the physical route (or, rather, the next physical hop on the route) to execute this overlay hop. Intermediate nodes on the physical path of an overlay hop consult their AODV table for the corresponding next physical hop. When a packet thus reaches the destination of an overlay hop, that node again consults its Pastry routing table and/or leaf set to determine the next overlay hop. This process continues until the packet reaches the eventual target node that is responsible for the packet key – i.e. whose overlay ID is the numerically closest to the packet key.

III. The MAPNaS Name Service

In MAPNaS, a resource (e.g. a file, a service, etc) is identified by a unique resource key that is mapped into the logical MADPastry ID space. Due to the lack of a fixed network topology in MANETs, there are no dedicated resource directory servers. Instead, true to the P2P paradigm, every node functions both as a resource host (of its own files,

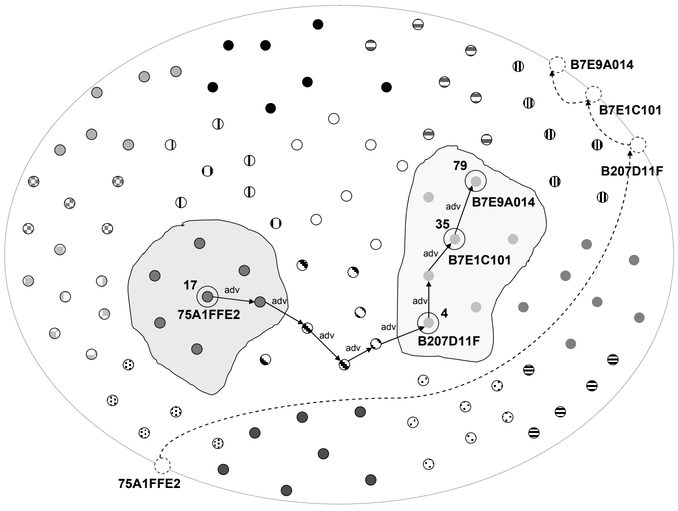

Figure 3. Indirect routing of a MAPNaS resource advertisement using MADPastry – the outer circle represents the overlay ID space.

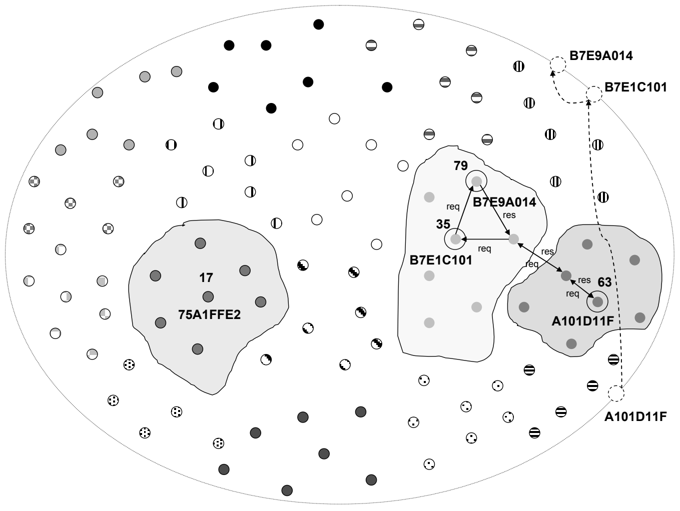

Figure 4. Resource discovery.

services, etc.) and as a resource directory for certain remote resources. As determined by MADPastry, every node keeps track of the network addresses of those resources whose resource keys it is responsible for. This design of MAPNaS is outlined in Figure 2. Nodes store the resource descriptors (the resource key along with the specific network address of the resource) they are responsible for in their local MAPNaS repository. Furthermore, every node advertises its own resources that it is willing to share through MAPNaS.

A. Resource Advertisement

When a node A in a MADPastry network wants to make a local resource (e.g. a service, a file, etc.) available to other nodes in the network, it needs to assign a hash key to that resource, e.g. by hashing the resource's name. Using that key, node A will then construct a resource descriptor consisting of the resource key and the physical network address (e.g. IP address) of the resource provider (in this case node A's address). Using MADPastry, the descriptor is routed to the node currently responsible for the resource key. That recipient node will then store the resource descriptor in its local repository.

Figure 3 shows an example of a resource advertisement. Node 17, whose current overlay ID is `75A1FFE2`, wants to advertise its resource with the name "file123". Hashing that file name yields the hash key `B7E9A578`. Node 17 now constructs a resource descriptor containing the resource key and the network address of the host: {`B7E9A578`, 17}. This advertisement packet will then be routed to the responsible node using MADPastry. At node 17, the closest entry in its MADPastry routing table is node 4 with overlay ID `B207D11F`. This first overlay hop (as indicated by dotted arrow) takes 5 physical hops (as indicated by the solid black arrows) to be completed and delivers the packet from the source cluster already to the target cluster (as indicated by the two shaded regions). Node 4 will then consult its MADPastry routing table to determine the next node to forward the advertisement packet to – in this case node 35 with overlay ID `B7E1C101`. This second overlay hop consists of two physical hops. For the final overlay hop, node 35 consults its MADPastry leaf set to forward the packet to node 79 (overlay ID `B7E9A014`) who is responsible for the resource key and who will store the resource descriptor.

B. Resource Discovery

Resource discovery with MAPNaS works analogous to the resource advertisement process. When a node needs to look up a resource, it will simply hash the resource name and send a request to the node currently responsible for that hash key. This request is routed using MADPastry in the same indirect manner as described above for the resource advertisements. The eventual destination node will check its local repository and send back the matching resource descriptor.

Figure 4 illustrates the resource discovery process with MAPNaS. Following up the example from above, suppose now node 63 (overlay ID `A101D11F`) is interested in the resource "file123" that is provided by node 17. Unfortunately, node 63 has no idea which node provides the desired resource. Therefore, node 63 hashes the name of the file, which yields the hash key `B7E9A578` as it did for node 17 for its advertisement. Next, node 63 simply sends a request for a matching resource descriptor towards the hash key using MADPastry. Thus, in the first overlay hop, the request will be delivered to node 35 with overlay ID `B7E1C101`. Node 35 will then forward the request to node 79 with overlay ID `B7E9A014` who, as before with the resource advertisement, is responsible for the given hash key. Upon reception of the request, node 79 will check its local repository and send a response containing the resource descriptor {`B7E9A578`, 17} back to the requester, node 63.

C. Local Replications

For the scalability and feasibility of a MANET, it is essential to restrict network traffic to local regions as much as possible [6, 10]. Therefore, MAPNaS makes use of MADPastry's clusters to store local replications of resource descriptors.

When a node intends to advertise a resource, it will now insert the resource descriptor under two different keys. The first key is the regular hash key and the resource descriptor is inserted into the network as described in A. To obtain the second key under which the resource descriptor will be stored, the regular resource key is altered to make sure the descriptor will be stored in the resource host's own MADPastry cluster. For this purpose, the resource key's prefix is replaced with the host's own cluster prefix. In a MADPastry network with 16 landmark keys (i.e. 16 prefix-based clusters), node 17 from the previous example would store the descriptor for its resource "file123" first under its key **B**7E9A578 somewhere in the network. Additionally, it would also insert the resource under the local key **7**7E9A578 into its own MADPastry cluster.

As described in Section II, MADPastry clusters are made up of nodes that share a common overlay ID prefix so that they are close to each other in the overlay ID space. These overlay neighbors are also likely to be close to one another in the physical network. Hence, intra-cluster communication can be expected to travel only short physical paths. Therefore, when a node needs to look up the address of a certain resource, it will generate the resource key (by hashing the resource name, etc) as described above. However, before engaging in a potentially cross-network indirect routing process to find the corresponding resource descriptor, the node will first replace the descriptor key's prefix with its own cluster prefix. To restrict the lookup process – if possible – to nodes in its physical vicinity, the lookup request is then first routed to the appropriate local cluster member to see whether a matching descriptor can already be found in the local cluster. This might, for example, be the case with popular files or standard services that are hosted by multiple nodes. Only if this local lookup provides no answer, will the node engage in a regular network-wide lookup process.

D. Handovers

When a MADPastry node moves from one cluster to another, it will eventually join the new cluster by assigning itself a new overlay ID that shares a common prefix with its cluster members. Therefore, when a MADPastry node running MAPNaS changes its cluster membership, it needs to pass the resource descriptors that are in its local repository to its old "left" and "right" leaf set members as those two nodes will now be numerically closest to the corresponding resource descriptor keys. Furthermore, when the rejoin process under its new overlay ID is completed, it needs to acquire from its new "left" and "right" leaf set members those resource descriptors whose keys it has now become responsible for. This handover process does not require indirect routing, though, since the network address of the corresponding leaf set members must be known (as these leaf set members are proactively maintained by MADPastry).

Since a handover packet could be lost – e.g. due to collision, etc. – a node can potentially end up having some resource descriptors in its local repository that it is actually not responsible for (any longer). To take care of such incidences, each node periodically checks its local repository for such descriptors and hands them over, if need be, to the best candidates as proposed by its Pastry leaf set or routing table.

IV. Simulation Results

To evaluate the performance of MAPNaS, we implemented a MAPNaS reference application running on top of a MADPastry routing agent in ns-2. All simulations that we carried out modeled wireless networks over the course of one (simulated) hour. Nodes are always moving around according to the Random Waypoint model with 0s pause time (constant movement). For data transmission, nodes are using the 802.11 communication standard with a transmission range of 250m. The node density in the networks that we investigate is always 100 nodes/km^2.

The question to be answered in MANETs when dealing with more complex approaches such as MAPNaS running on top of MADPastry is whether the effort of maintaining the data structures is really worthwhile. In the case of MAPNaS, is there really anything to be gained from going through the process of advertising resources, handing over resource descriptors, maintaining MADPastry's routing tables, etc.? Or, would it be perfectly sufficient if nodes did not advertise their resources and if resource discovery requests were simply broadcast through the network? For this purpose, we also implemented a reference application in ns-2 where nodes do not advertise their own resources and resource discovery requests are simply broadcast (already forwarded requests will not be forwarded a second time). Every receiving node checks its own resources and, if there is a match, it sends back a direct response using AODV. For such unicasts, we are using the AODV-UU implementation 0.9.1 for ns-2.

For all simulations, each node provided 5 resources. Equally, each node periodically issued a discovery request for a randomly selected resource every 10 seconds. Furthermore, MAPNaS nodes cache overheard resource descriptors (e.g. from handover or response packets) for a duration of 30s.

The following two metrics are analyzed:

Success Rate. This figure represents the percentage of random lookups that eventually deliver a response containing the correct resource descriptor back to the requesting node. In other words, this is the round-trip (request + response) success rate of all lookups.

Overall Traffic. Since many different packet types (e.g. AODV route requests, MADPastry packets, resource advertisements, handover packets, etc.) of various packet lengths are transmitted during a simulation run, we are not evaluating the total packet count. Instead, we are considering the total network traffic (in kilo bytes) that is created during the simulated hour. Whenever a node forwards a packet, this figure is increased by the packet size. Again, this figure includes *all* routing and application level packet types (AODV, MADPastry and MAPNaS packets).

A. Network Size

In a first set of simulations, we are comparing MAPNaS on top of MADPastry against the simple broadcast approach with AODV for varying network sizes (50, 100, 150, 200, and 250 nodes). For all network sizes, nodes were always moving at a

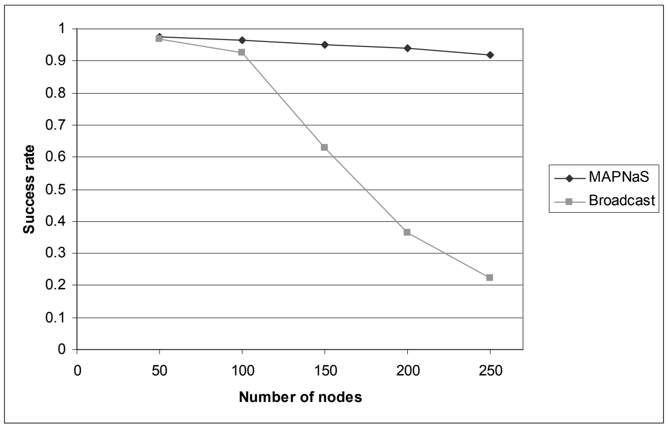

Figure 5. Success rates vs. number of nodes.

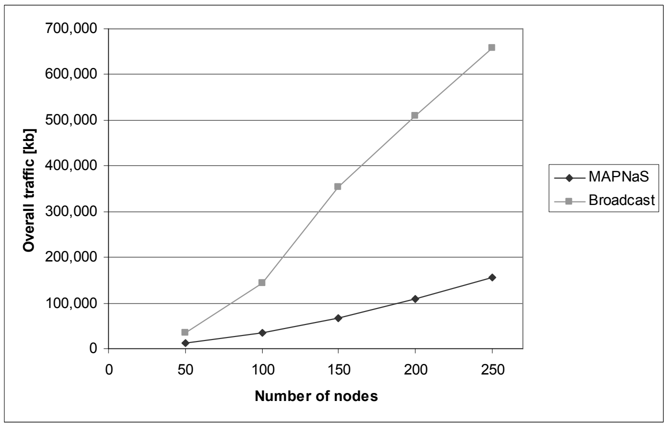

Figure 6. Overall traffic vs. number of nodes

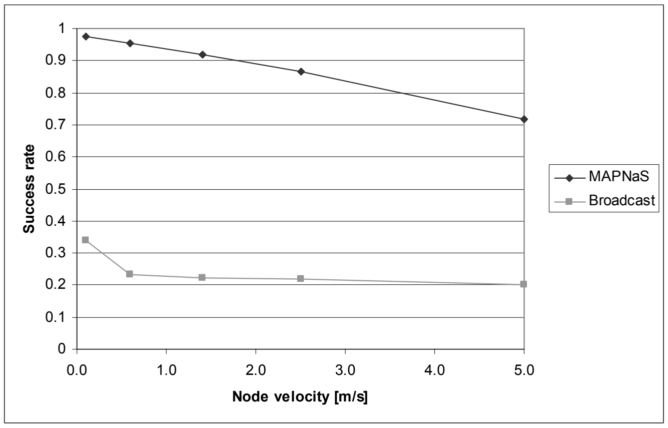

Figure 7. Success rate vs. node velocity.

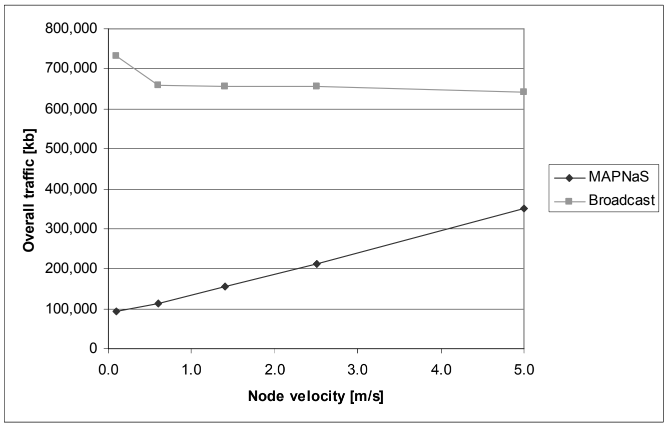

Figure 8. Overall traffic vs. node velocity.

constant speed (Random Waypoint, 0s pause time) of 1.4 m/s – a quick walking pace. To measure the basic results of both systems, MAPNaS nodes always used the global resource keys and did not first query their own cluster for possible replications.

Figure 5 compares the success rates of MAPNaS against the broadcast approach. As can be observed, for all considered network sizes, MAPNaS achieves success rates of well above 90%. The broadcast-based approach, on the other hand, can only keep up with MAPNaS in small networks of 50 and 100 nodes. For larger networks, its success rate deteriorates drastically.

There are two main reasons for MAPNaS's much better performance. First of all the broadcast application produces significantly more traffic (roughly 4 times as much) than MAPNaS over MADPastry does, as Figure 6 demonstrates. Because of this higher traffic, there are clearly more packet losses in the broadcast approach due to collisions and interference. Thus request and responses are often dropped before they reach the appropriate destination. Secondly, the overall traffic of the broadcast application is entirely made up of broadcast requests, AODV packets, and response messages, all of which usually affect the entire network. With MAPNaS on the other hand, a sizable portion of the overall traffic stems from MADPastry and is thus often restricted to certain clusters.

B. *Node Velocity*

Another parameter that certainly influences the performance of applications in MANETs is the node velocity. Therefore, in the next set of simulations, we considered a 250-node network with varying node velocities (0.1, 0.6, 1.4, 2.5, and 5.0 m/s). Again, all nodes provided 5 resources and issued requests for randomly selected resources every 10 seconds.

Figure 7 shows the success rates of the two resource discovery approaches. Not surprisingly, the broadcast-based approach is scarcely affected by the node velocity as its network-wide discovery requests do not rely on any established routes that could break. Of course, the AODV-unicast replies are affected by different node velocities, but since they constitute only a small fraction of the overall traffic (success rates around 20%), the effect really only becomes noticeable in almost stable networks (node velocity 0.1 m/s).

MAPNaS can maintain success rates of above 90% up to a node velocity of 1.4 m/s. For higher node velocities, it becomes more and more difficult for MADPastry to maintain its overlay clusters. This results in high numbers of cluster membership changes, which, in turn, trigger an ever growing amount of necessary handover traffic – as becomes visible in Figure 8, which, again, depicts the overall traffic produced by the respective resource discovery approaches.

C. *Local Replications*

Next, we examine the impact of local replications on the

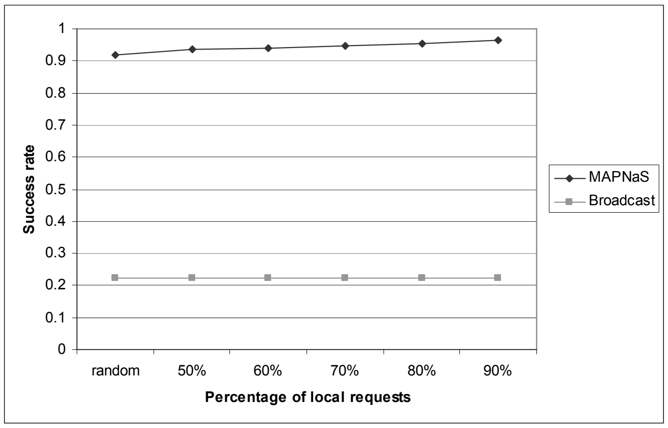

Figure 9. Success rate vs. locality rate.

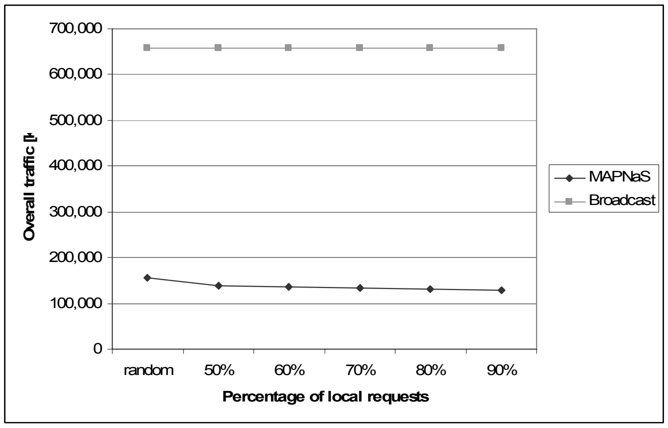

Figure 10. Overall traffic vs. locality rate.

overall performance. We employ a simple traffic pattern where 50%, 60%, 70%, 80%, or 90% of all resource requests can be satisfied locally. Otherwise, a 250-node network with a node velocity of 1.4 m/s is used.

As there are no overlay clusters in the broadcast application that would allow for a deterministic way of inserting local replicas and since requests are always broadcast throughout the network, it is hard to compare the performance of MAPNaS with local replications against the broadcast application in a fair and meaningful manner. For the sake of simplicity, we are merely providing the result of the broadcast-based approach from the previous section as a reference line.

Figure 9 and Figure 10 show the success rate and the overall network traffic in reference to the locality rate. As can be expected, a high locality rate furthers boosts the performance of MAPNaS, as shorter physical routes (intra-cluster communication) are not as vulnerable as cross-network routes. For MAPNaS, the success rate can be increased to over 96% for high locality rates.

V. RELATED WORK

Another approach that proposes the integration of a conventional DHT with an ad hoc routing protocol to provide indirect routing in MANETs is Ekta [12]. Ekta, like MADPastry, is based on Pastry [15], but it uses DSR [8] for its route discoveries. Also like MADPastry it is not a name service or resource discovery application on its own, but instead it is a DHT substrate for MANETs that could be used to build such applications.

The main difference to MADPastry is that Ekta does not explicitly consider physical proximity in its DHT routing table. Instead, it merely tries to optimize its DHT entries by overhearing packets and replacing physically remote entries by nearer ones. Ekta has no notion of overlay clusters of physically close nodes. Thus, the routes traveled during its overlay routing process may be expected to be less efficient than those in the cluster-based MADPastry. This should become even more pronounced as the network size increases.

Most recently, *Virtual Ring Routing* (VRR) [2] has been proposed. With VRR, each node maintains an AODV-style route to each of its *r* virtual neighbor nodes. When a node wants to send a packet with a given key, it selects from its routing table the entry numerically closest to the packet's key. The packet is then forwarded to that node using the next hop information from the routing table entry. This process continues until the packet reaches the node whose virtual identifier is numerically closest to the packet's key.

VRR also has some drawbacks. First of all, since nodes set up physical paths to new virtual neighbors via existing physical neighbors, the physical paths thus set up between virtual neighbors during the bootstrap process are likely not shortest physical paths. Also, there is no correlation between physical locality of nodes and their virtual neighbors. Thus, a key lookup can be expected to zigzag through the physical network. This is further aggravated by that fact that the source routes of the individual overlay hops are quite likely to be suboptimal – in other words: not only can a key lookup be expected to zigzag through the physical network, but the individual legs of the zigzag course will also quite likely be unnecessarily long.

Furthermore, a large number of approaches have been proposed for resource discovery in MANETs, for example [16, 9, 18]. A comprehensive summary of all existing approaches is clearly beyond the scope of this paper. In brief, existing approaches have been using all sorts of techniques such as centralized servers, multicasting, geographic routing, piggybacking, secondary service directory overlays, etc., etc. However, it is the purpose of MAPNaS to demonstrate how to implement resource discovery for MANETs based on a mobile DHT (MADPastry).

VI. CONCLUSION

In this paper, we have described in detail the architecture of MAPNaS, a peer-to-peer based resource discovery approach for MANETs. MAPNaS runs on top of MADPastry, a general-purpose DHT substrate for MANETs. Instead of having a number of dedicated directory servers, every MAPNaS node serves both as a resource directory for certain remote resources and as a host of its own resources.

Through extensive simulations, we have observed that MAPNaS on top of MADPastry achieves significantly better discovery success rates than a simple broadcast-based reference application does for most network environments considered. Furthermore, MAPNaS achieves its better success rates while producing markedly lower amounts of network traffic. The results presented in this paper indicate that DHT-based resource discovery can be performed efficiently under many MANET conditions. As a "casual" rule-of-thumb, DHT-based resource discovery appears very promising in larger MANETs ($\geq$ 100 nodes) with medium node velocity (walking speeds). In smaller networks, one does not necessarily need to maintain DHT structures but could use less complex approaches as well. In very volatile networks (e.g. high node velocities), it becomes more and more challenging to maintain the DHT structures so that approaches with small structural overhead – such as broadcasting – might be preferable over DHT-based approaches.

It should be pointed out at this point that it is not the purpose of this paper to proclaim MAPNaS to be the resource discovery of choice for all conceivable MANET configurations. Instead, its purpose is to demonstrate under which network conditions it is feasible to build resource discovery efficiently on top of a DHT substrate such as MADPastry. The presented results look promising in this regard.

In the future, it would be interesting to see how other DHTs for MANETs such as Ekta or VRR would impact the performance of MAPNaS. An aspect of particular interest, here, would be the trade-off between considering physical locality but having handovers (MADPastry) and not considering physical locality without the need for handovers (Ekta, VRR). Additionally, it would be appealing to compare the performance of MAPNaS to other existing, more elaborate resource discovery approaches. We also plan to further investigate the impact of other traffic patterns. So far, we considered all resources equally popular. Other distributions, e.g. Zipf, would be interesting here.

REFERENCES

[1] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. "A Layered Naming Architecture for the Internet". In *Proc. of ACM SIGCOMM*, August 2004.

[2] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. "Virtual Ring Routing: Network routing inspired by DHTs". In *Proc. of ACM SIGCOMM'06*, September 2006.

[3] M. Castro, P. Druschel, A.-M. Kermerrec, and A. Rowstron. "One Ring to Rule them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks". In *Proc. of SIGOPS*, September 2002.

[4] R. Cox, A. Muthitacharoen, and R.T. Morris. "Serving DNS using a Peer-to-Peer Lookup Service". In *Proc. of IPTPS*, February 2002.

[5] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. "Optimized Link State Routing Protocol for Ad Hoc Networks". In *Proc. of IEEE INMIC*, December 2001.

[6] P. Gupta and P. R. Kumar. "The Capacity of Wireless Networks". In *IEEE Transactions on Information Theory, Vol. 46, No. 2*, March 2000

[7] Z. J. Haas, M.R. Pearlman, and P. Samar. "The Zone Routing Protocol (ZRP) for Ad Hoc Networks". In *Internet Draft RFC (http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt)*, July 2002.

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

[9] U.C. Kozat and L. Tassiulas. "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks". In *Proc. of IEEE INFOCOM*. March 2004.

[10] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. "Capacity of Ad Hoc Wireless Networks". In *Proc. of ACM SIGMOBILE*, July 2001.

[11] C. E. Perkins and E. M. Royer. "Ad hoc on-demand distance vector routing". In *Proc. of IEEE WMCSA*, February 1999.

[12] 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 IEEE WMCSA*. December 2004.

[13] V. Ramasubramanian and E.G. Sirer. "The Design and Implementation of a Next Generation Name Service for the Internet". In *Proc. of ACM SIGCOMM*, August 2004.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A Scalable Content-Addressable Network". In *Proc. of ACM SIGCOMM*, August 2001.

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

[16] F. Sailhan and V. Issarny. "Scalable Service Discovery for MANET". In *Proc. of PerCom*, March 2005.

[17] 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 SIGCOMM*, August 2001.

[18] J.B. Tchakarov and N.H. Vaidya. "Efficient Content Location in Mobile Ad hoc Networks". In *Proc. of MDM*, January 2004.

[19] M. Walfish, H. Balakrishnan, and S. Shenker. "Untangling the Web from DNS". In *Proc. of NSDI*, March 2004.

[20] R. Winter, T. Zahn, and J. Schiller. "Random Landmarking in Mobile, Topology-Aware Peer-to-Peer Networks". In *Proc. of FTDCS*, May 2004.

[21] T. Zahn and J. Schiller. " MADPastry: A DHT Substrate for Practicably Sized MANETs". In *Proc. of ASWN*, June 2005.

[22] T. Zahn and J. Schiller. "MAPNaS: A Lightweight, Locality-Aware Peer-to-Peer Based Name Service for MANETs". Short paper in *Proc. of LCN 2005*, November 2005.

[23] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-Resilient Wide-area Location and Routing". *UCB Tech. Report UCB/CSD-01-1141*, April 2001.