Track-Based Hybrid Service Discovery in Mobile Ad-hoc Networks

Zhuoqun Li, Lingfen Sun and Emmanuel C. Ifeachor
School of Computing, Communications and Electronics
University of Plymouth
Drake Circus, Plymouth, PL4 8AA, United Kingdom
{Zhuoqun.Li, L.Sun, E.Ifeachor}@plymouth.ac.uk

***Abstract*— Service discovery is a challenging task in mobile ad-hoc networks which are characterized by their dynamic, infrastructure-less and bandwidth-limited nature. The goal of this paper is to develop a bandwidth efficient approach for service discovery in mobile ad-hoc networks. The main contribution of this paper is a novel *Track*-based hybrid service discovery scheme. The proposed scheme performs service discovery based on the one dimensional proactive structure (e.g. tracks), instead of the existing two dimensional proactive structures (e.g. zones). This scheme can lead to reduced proactive traffic due to its light-weight organization of the hybrid structure. The density of proactive tracks is adapted to network traffic pattern dynamically for less proactive traffic. We simulated the *Track*-based approach in ns-2 network simulator with various network scales and mobility, and compared the performance with pure reactive (e.g. flooding) and Zone-based hybrid (e.g. bordercasting) protocols. Preliminary results show that, the *Track*-based approach achieved better efficient trade-offs between control packet overhead and querying success rate with minimal proactive costs among all the approaches compared.**

I. Introduction

Service discovery, which allows devices to transparently and seamlessly locate available software/hardware entities throughout the heterogeneous communication networks, is a critical component for on-demand communications and collaborations in pervasive computing environments. In the basic paradigm of service discovery, service consumers advertise requests containing attributes representing the services they need. A few service providers who want to share their resources listen on a specified interface for service requests and reply to those matching the services they hold.

Mobile ad-hoc networks are multi-hop networks formed by battery-powered mobile devices for instant networking needs, i.e. in case of eHealthcare emergency or crisis management. Service discovery is a challenging task in mobile ad-hoc networks due to lack of stable infrastructures as well as adverse conditions of the wireless channel. For example, conventional directory based service discovery standards such as Jini [1] and UDDI [2] can not be implemented straightforwardly as they require the support of a stable infrastructure. Related research activities are in the areas of routing/resource discovery for mobile ad-hoc networks. Most of these can be broadly classified as: virtual backbone [3] [4], pure reactive [5] [6] and hybrid [7] [8] [9] [10].

Approaches in the category of virtual backbone [3] [4] try to maintain a virtual hierarchical architecture in the ad-hoc network to support directory based service discovery. However, the virtual hierarchy requires constant management overhead and can be susceptible to single node failures.

Pure reactive service discovery approaches, e.g. EDSR [5], successfully waive periodical updates by advertising service requests on-demand. When a node needs to know about the location of specific resources/services, it floods service requests throughout the ad-hoc network . Response messages will be sent back by nodes which can provide the locations of the resources requested. Although the pure reactive schemes do not require constant exchange of control packets, flooding the whole network for every query is inefficient, especially when the service could have been easily located at the node's neighbourhood.

Hybrid approaches such as CARD [7], ZRP [8] and its variants [9] [10] try to benefit from the advantages of both proactive and reactive strategies. These protocols limit periodic exchanges of route updates to a node's neighbourhood, i.e. zones or vicinities of nodes several hops away. Resources inside the proactive area will be ready on demand. For resources not reachable in a node's own zone or vicinity, these protocols would dynamically initiate service discoveries in a reactive way. They also try to reduce excess traffic flooding by unicasting queries to remote 'contacts' [7] or bordercasting to neighbouring zones [8] [9] [10]. However, recursively unicasting queries to remote areas are likely to incur long discovery delays, whereas bordercasting is vulnerable to query failures due to out-of-date topological knowledge of the complicated proactive zones. A common drawback of these approaches is that substantial bandwidth overhead needs to be spent on maintaining the proactive area even in the absence of service queries or data traffic.

The limitations of current approaches motivated us to develop a more efficient and flexible hybrid service discovery protocol, which should be able to minimize the control packet overhead with light-weight proactive structure and self-adaptability for different traffic patterns.

The main contribution of this study is a novel *Track*-based hybrid service discovery scheme. Unlike conventional Zone-based reactive/proactive hybrid schemes [7] [8] [9] [10], the proposed scheme deploys one dimensional tracks connecting service consumers and providers as the proactive vicinities. By

doing so, it requires much less proactive overhead than those with two dimensional proactive areas (e.g. circular zones). The proposed approach is also flexible in maintaining proactive areas. On one hand the tracks are built on existing service consumer to service provider associations. The busier the network, the more proactive the system behaviour becomes. On the other hand, the efforts in taking care of the proactive areas can be merged with those required for maintaining communications between service consumers and providers in the post-discovery stage. The *Track*-based scheme is designed to be integrated with existing routing/networking protocols to achieve better efficiency than application layer protocols (e.g. Lanes [11]or Chord [12]).

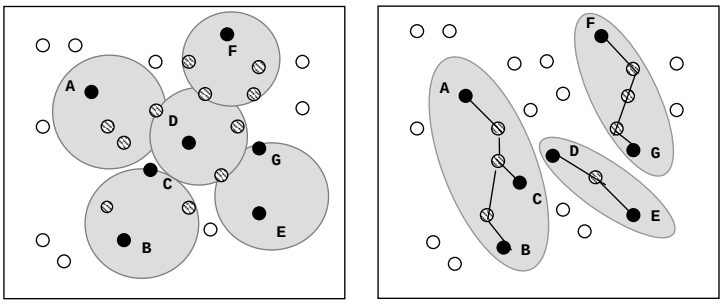
We have implemented the *Track*-based service discovery scheme in ns-2 network simulator [13]. We compared its performance with existing pure reactive (i.e. *flooding* [6]) and Zone-based (i.e. *bordercasting* [8]) approaches. Preliminary results show that the Track-based scheme achieved more efficient trade-offs between control packet overhead and success rate with minimal proactive overhead among all the approaches compared.

The remainder of this paper is organized as follows. In Section II, we describe the details of the *Track*-based service discovery protocol. The simulation model and settings are given in Section III. The performance measures and simulation results are illustrated in Section IV. Section 5 concludes the paper.

II. The *Track*-based Scheme for Service Discovery

A. Protocol description

The *Track*-based scheme is comprised of a proactive component and a reactive component for service discovery. The difference between the proposed protocol and the Zone-based hybrid approaches [8] [9] [10] is illustrated in Fig. 1, where grey areas represent proactive areas. Dark nodes correspond to service consumers (e.g. *B*, *C*, *E* and *G*) or providers (e.g. *A*, *D* and *F*). Shadow denotes nodes involved in proactive areas, and white for the remaining individual nodes. As illustrated in Fig. 1 (a), proactive areas in the Zone-based service discovery are circular zones consisting of nodes several hops away. Maintaining such a two dimensional topology requires a lot of proactive overhead, especially when node density is high. On the contrary, as depicted in Fig. 1 (b) the one dimensional track-like structure of proactive areas in the *Track*-based scheme would require minimal maintenance overhead. As each node in a track only needs to keep trace of its predecessor and its successor in the same track. The *Track*-based scheme is also flexible in adapting the number of tracks when the traffic pattern changes. The Zone-based strategies require every node to maintain its proactive zone to support its inter-zone bordercasting mechanism [8] [14], whereas in the *Track*-based scheme, proactive tracks are built from the associations between service consumer and service provider. Thus, the density of tracks grows or decreases with the number of these associations.



(a) (b)

Fig. 1. Instances of proactive areas. (a)the Zone-based service discovery,(b) the *Track*-based service discovery.

B. Proactive service discovery component

The proactive component of the proposed scheme is responsible for establishing and updating topological information of tracks through periodical validation messages. It also proactively maintains the states of available resources in a track.

When a service consumer successfully associated itself with its service provider a track is created between them at the same time. Nodes involved in a track will advertise their services to the whole track. As illustrated in Fig. 2, proactive areas are built on consumer-to-provider associations (e.g. $B \rightarrow A$, $C \rightarrow A$, $E \rightarrow D$ and $G \rightarrow F$). The lifetime of a track will be terminated after a certain idle period. This approach eliminates unnecessary proactive efforts for relay nodes never used.

Periodical validation messages are emitted from a service consumer to keep tracing its service provider and to detect broken connections in the track structure. Each relay node in a track forwards the validation message to its successor in the same track and replies to its predecessor with a validation acknowledgement. If the validation is not acknowledged for a certain time, the node assumes the connection to its successor is broken and invokes the local recovery procedure to repair the broken link.

The local recovery mechanism is designed to help tracks

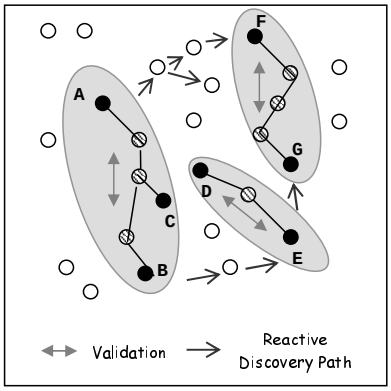


Fig. 2. General view of the service discovery path between service consumer *B* to service provider *F*

survive frequent link failures with acceptable recovering overhead. Generally, the missing successor of a node could just move away when the link broken is detected. And this successor may still be accessible by other one-hop away neighbours. It is also possible that some one-hop neighbours have obtained alternative paths towards the destination. Based on these considerations, we use a two-hop broadcast method to recover the broken connections of a track. This method enables an efficient local (two-hop away) discovery of alternative paths of broken connections without initiating global path discoveries.

The bandwidth overhead (i.e. transmission and reception of control packets) required to maintain a track can be estimated as $2\sigma(L-1)+2\mu(\Delta+\Delta^2)$ packets (emitted and received) per track, where σ is the validation rate for an L hops long track, μ is the link breakage rate and Δ the average degree of a node. It is a sum of validation and local recovery overhead where $2\sigma(L-1)$ represents the former and $2\mu(\Delta+\Delta^2)$ the latter.

C. Reactive service discovery component

The reactive component in the *Track*-based scheme is employed to disseminate service requests that can not be matched locally. Fig. 2 illustrates the reactive discovery path between consumer *B* and service provider *F*. When node *B* needs a specific type of resource or service, it first checks the availability of such service in its own service table which also comprises of service information of track $B \rightarrow A$. If the required service can not be matched at the local service table, the reactive component of node *B* creates and pushes a service request to nodes in track $B \rightarrow A$, which would then help to relay the request to neighbouring nodes or tracks.

To efficiently relay requests among mixed tracks and individual nodes (as not every node is involved in a track), we make the proactive tracks as entities to disseminate service requests by broadcasting just as a node. However the broadcasting points in a track are limited at nodes with higher degree so as to efficiently cover most of the neighbouring nodes. The number of nodes that a track broadcast to in relaying a request are denoted as the *effective degree* Δ_E. A higher effective degree Δ_E of a track indicates more of discovery routes that it can make. As show in Fig. 2, track $B \rightarrow A$ made more discovery routes than track $E \rightarrow D$ due to its higher Δ_E.

Upon receiving a service request, a reply is sent back if the receiving node knows the location of the services requested. Otherwise, individual nodes would rebroadcast the request to its neighbours. For nodes belonging to tracks (e.g. node *E* and *G* in Fig. 2), the request is pushed to the whole track before it is relayed to the neighbouring areas. The same request received by any node in this track next time is considered redundant and will be discarded. As shown in Fig. 2, although track $G \rightarrow F$ received the same requests from different nodes, only the first copy would be processed.

As the density of proactive tracks is growing with the traffic in the *Track*-based scheme, few tracks would be built up in an idle traffic pattern. In this case, the performance of the proposed scheme is similar to that of a reactive approach, i.e. the Ad-hoc On-demand Distance Vector Routing (AODV) [6]. Querying traffic (transmissions and receptions) at this stage is expected to be $2M_n + f_{rep}$ packets per query, where M_n is the number of links in an ad-hoc network with size n and f_{rep} represents the number of service reply packets unicasted to the requesting source. The value of f_{rep} can be roughly estimated according to the distance between the replying nodes and the requesting source [15]. With *t* tracks existing in the *n* nodes network, the reactive overhead of this approach can be estimated as $2(M'_n + \sum_{i=0}^{t} \Delta_{Ei}) + f_{rep}$, where M'_n is the number of links excluding those connected to the t tracks and Δ_{Ei} is the effective degree of track *i*. As M_n is growing faster than the network size, reduce M_n to M'_n contributes to the reduction of reactive packets overhead per request.

III. Simulation Model

In this section, we briefly present the parameters and settings in simulating the behaviours of the *Track*-based scheme as well as the protocols used for performance comparison, namely, flooding and bordercasting. We implemented these service discovery mechanisms in the ns-2 network simulator. The link layer and physical interface were set to approximate the Lucent WaveLAN^{TM} card with 250m nominal propagation range. IEEE 802.11b was utilized as the MAC protocol operating at a data rate of 2 MBit/s.

In order to maintain a constant node density (an average of 8 for Δ), mobile nodes are randomly distributed in an area growing from 700m$\times$700m to 1800m$\times$1800m, while the number of nodes ranges from 20 to 120 to reflect realistic scenarios. In our simulations, nodes move according to the random-waypoint mobility model as described in [5] [9]. Maximum velocities range between 0 m/s and 20 m/s, while the pause time is kept at 0 seconds to cause continuous movement. The default maximum velocity is 5m/s for varying simulation topologies. While we changing the maximum velocity, the default network size is 40 nodes and the default simulation area is 1000m$\times$1000m.

Among the service discovery protocols simulated, *flooding* and the reactive component of the *Track*-based scheme are extensions to AODV, while *bordercasting* is derived from ZRP with 2-hop zone radius, and is configured with query detection and early termination as described in [5]. To simulate the procedure of service discovery, we generally specify 3 pairs of service providers for 3 categories of services resulting in a total of 6 providers. We also vary the number of service consumers and query rate in traffic patterns noted as T(*C*, *Q*), i.e. each of *C* service consumers initiates *Q* queries every second for each kind of service in turn. If not stated, the default traffic pattern is T (12, 1). While we varying the traffic patterns, the network size is fixed at 40 and maximum speed fixed at 5m/s.

IV. Simulation Results and Evaluation

The measures deployed for performance evaluation and the results obtained from the ns-2 simulation are described in this section. We measured the system's performance in terms of

control packets overhead and first attempt success rate. In the results given by the following figures, each data point corresponds to a mean of 100 repeated measurements with different random seeds. We also plot the 95% confidence interval as error bars on these figures.

A. Control packets overhead

The control packets overhead consists of querying overhead and proactive overhead. Querying overhead gives a quantitative view on traffic produced for a service query, while proactive overhead represents those required to maintain proactive areas of an ad-hoc network averaged over a certain period of time. HELLO beacons are not included as their quantity is independent from the discovery schemes used. We try to illustrate that how the *Track*-based scheme) adapt to various traffic patterns in Fig 3. We can see from Fig. 3 that the reactive traffic decreases when the network becomes busier. This is the natural cause of increased track density. Higher track density makes the behaviour of the *Track*-based scheme become more proactive, as also indicated by the curve of proactive traffic.

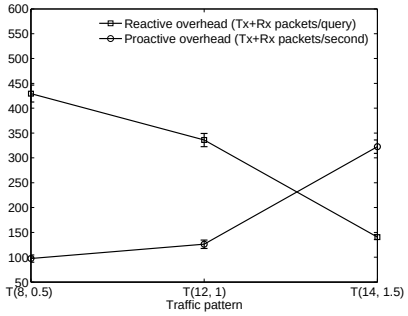


Fig. 3. Control packets overhead vs. Traffic Pattern

Figs. 4 and 5 presents a comparison of the querying overhead among the three service discovery protocols against varying network size and mobility. The overhead measured includes transmission as well as reception of control packets in each operation. From these two figures, we can see that *Track* achieved at least 30% less traffic than *flooding* in transmitting requests and replies in all the scenarios simulated. From Fig. 4 it seems that *bordercasting* outperforms the rest with its efficient inter-zone bordercasting techniques. However, Fig. 5 shows that the performance of *bordercasting* degrades dramatically when the maximum speed is increased. This result is due to that *bordercasting* delivers requests according to its bordercast tree topology that may be inadequate in frequent link failures or collisions caused by high mobility.

Figs. 6 and 7 gives the overall control packets overhead comparison with indications of the contributions of both reactive and proactive overhead. The control packets overhead measured are quantified as packets per second when the reactive traffic is generated from 1 query per second. As shown in Figs. 6 and 7, proactive traffic constitutes of 80-90% of the overall overhead of *bordercasting* in all scenarios. As the two dimensional proactive zone of *bordercasting* requires frequent exchanging of control packets (e.g. link state packets) to sense topology reconfigurations. Such maintenance expenses are very sensitive to the size or mobility of the ad-hoc network. On the contrary, reactive traffic constitutes of 60-90% of the overall overhead of *Track*. For *flooding*, all of the overall overhead is produced only in reactive queries. However, with such a low query rate, the proactive traffic of the *Track*-based scheme does not lead to worse performance than *flooding*. In fact, Figs. 6 and 7 show that the overall traffic for *Track* is about 25% less than that of *flooding* in large network size or high mobility scenarios. There are several reasons why the *Track*-based scheme can achieve such a significant reduction in proactive traffic. Firstly, the one dimensional structure of tracks requires very light traffic for maintenance or local recovery. Secondly, the number of tracks are adapted to fit the traffic pattern of the mobile ad-hoc networks, whereas in *bordercasting* the proactive zones are independent of the pattern of traffic.

B. First attempt success rate

The first attempt success rate is the ratio of successful queries in one try over the total number of generated queries. This metric is more straightforward than those obtained in simulations of [7] [10]with up to 3 retries for failure queries. The performance of first attempt success rate try is illustrated in Figs. 8 and 9 against growing network size and mobility, respectively. Despite *bordercasting* can efficiently deliver requests among proactive zones and reducing querying traffic with its inter-zone bordercasting mechanism, whose accuracy heavily relies on its knowledge of the proactive zones' topology [10]. As more link state packets are lost in collisions or link breakages, it is more difficult to precisely maintain the topological information when the network size or mobility increase. Therefore, the first attempt success rate of *bordercasting* drops significantly when the network size or mobility increase, as shown by Figs. 8 and 9. Although collision-free solutions may help to release this problem, we can still observe the weakness of relying on proactively maintained topology to unicast service requests.

Also plotted in Figs. 8 and 9, the curves of *Track* and *flooding* do not show much sensitivity on changing network size or mobility. *Track* keeps the first attempt success rate over 90% in most cases. This value is generally decreasing with the network size. However, from Figs. 8 it is approaching that of *flooding* when we increase mobility. This is because more tracks are likely to be broken in higher mobility scenarios resulting in the more reactive behaviour of the *Track*-based scheme. From the two figures, we can conclude that the *Track*-based scheme does not trade much of the querying success rate off for the reduced querying or proactive overhead.

V. Conclusion

In this paper, we present the novel *Track*-based scheme for service discovery in mobile ad-hoc networks. The *Track*-based scheme is a combined proactive/reactive hybrid service discovery approach. It tries to minimize the proactive traffic

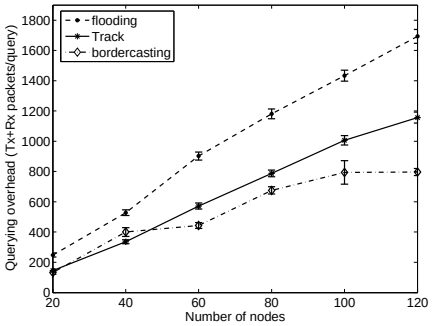


Fig. 4. Querying overhead vs. Network Size

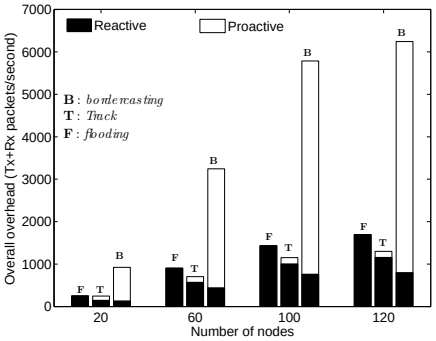


Fig. 6. Control packets overhead vs. Network Size

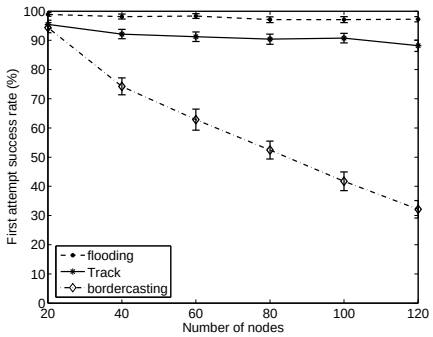


Fig. 8. First attempt success rate vs. Network Size

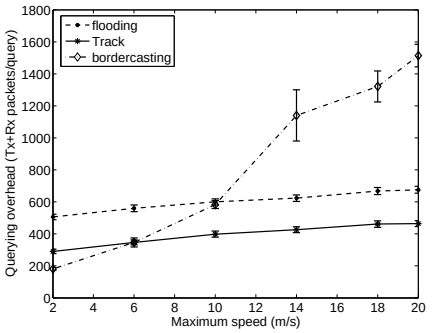


Fig. 5. Querying overhead vs. Mobility

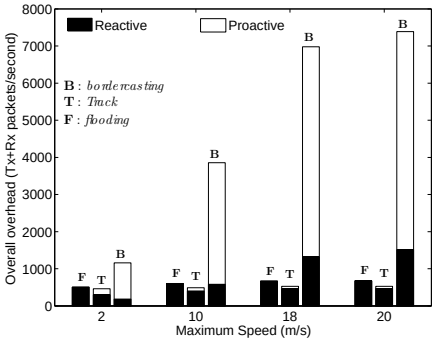


Fig. 7. Control packets overhead vs. Mobility

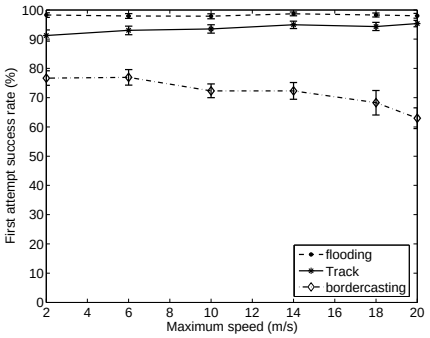


Fig. 9. First attempt success rate vs. Mobility

with its one dimensional track-like structure of proactive areas. It also adapts the density of proactive tracks to different traffic patterns for better efficiency. We simulated the proposed scheme in the ns-2 network simulator for a wide range of network scale and mobility scenarios. Its performance is compared with pure reactive (e.g. *flooding*) and Zone-based hybrid (e.g. *bordercasting*) protocols against several Quality of Service measures (e.g. control packets overhead, first attempt success rate). Preliminary results show that, the *Track*-based approach requires minimal overhead to maintain its light-weight hybrid framework and performs more efficiently in balancing the trade-offs between control packets overhead and other performance measures among the approaches compared. Future work involve more intelligent algorithms for the organization of the proactive track structure to achieve better Quality of Service for upper layer applications.

ACKNOWLEDGMENT

The work reported here is supported in part by the BIOPATTERN EU Network of Excellence (EU Contract 508803).

REFERENCES

[1] "JINI™ architectural overview, technical white paper," *Sun Microsystem*, January 1999.

[2] "UDDI version 3.0.2," *UDDI Spec Technical Committee Draft*, October 2004.

[3] U. C. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," *in Proc. of IEEE Infocom 2003, San Francisco, USA*, March 2003.

[4] J. C. Liu, Q. Zhang, W. Zhu, and B. Li, "Service locating for large-scale mobile ad hoc networks," *Kluwer International Journal of Wireless Information Networks*, 2002.

[5] I. Gruber, R. Schollmeier, and W. Kellerer, "Performance evaluation of the mobile peer-to-peer protocol," *Proc. of Fourth International Workshop on Global and Peer-to-Peer Computing*, April 2004.

[6] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad-hoc on-demand distance vector (AODV) routing," *RFC 3561*, July 2003.

[7] A. Helmy, S. Garg, P. Pamu, and N. Nahata, "Contact Based Architecture for Resource Discovery (CARD) in large scale MANETs," *Proc. of Int'l IEEE/ACM Parallel and Distributed Processing Symp.*, April 2003.

[8] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for ad hoc networks," *Internet Draft*, November 1997.

[9] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, "SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks," *in Proc. of ACM MobiHoc Conference 2003*, pp. 303–313, June 2003.

[10] L. Wang and S. Olariu, "A two-zone hybrid routing protocol for mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 12, December 2004.

[11] M. Klein, B. Konig-Ries, and P. Obreiter, "Lanes - a lightweight overlay for service discovery in mobile ad hoc networks," *in Proc. of 3rd Workshop on Applications and Services in Wireless Networks (ASWN2003)*, July 2003.

[12] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, February 2003.

[13] "The network simulator - ns-2," *available on line at http://www.isi.edu/nsnam/ns*.

[14] M. R. Pearlman and Z. J. Haas, "Determining the optimal configuration for the zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, August 1999.

[15] T. Clausen, P. Jacquet, and L. Viennot, "Analyzing control traffic overhead versus mobility and data traffic activity in mobile ad-hoc network protocols," *ACM Wireless Networks journal (Winet)*, vol. 10, no. 4, July 2004.