

Lanes – A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Networks*

Michael Klein, Birgitta König-Ries, Philipp Obreiter

IPD, Universität Karlsruhe, D-76128 Karlsruhe, Germany

{kleinm | koenig | obreiter}@ipd.uni-karlsruhe.de

Abstract

The ability to discover services offered in a mobile ad hoc network is the major prerequisite for effective usability of these networks. Unfortunately, existing approaches to service trading are not well suited for these highly dynamic topologies since they either rely on centralized servers or on resource-consuming query flooding. Application layer overlays seem to be a more promising approach. However, existing solutions like the Content-Addressable Network (CAN) are especially designed for internet based peer-to-peer networks yielding structural conditions that are far too complex for ad hoc networks. Therefore, in this paper, we propose a more lightweight overlay structure: lanes. We present algorithms to correct and optimize its structure in case of topology changes and show how it enables the trading of services specified by arbitrary descriptions.

1 Introduction

Over the last few years, much effort has been put into making multihop ad hoc networks (MANETs) a reality. Research has been focused on the development of appropriate routing protocols, methods for energy preservation, and other issues on the lower four ISO/OSI layers. While this work is still ongoing, by now, a sound technical basis for MANETs exists. In our opinion, it is thus time to start thinking about how to support applications based on MANETs. Just as the mere fact that computers were networked was not sufficient to allow for transparent access to distributed resources, the mere fact that it is technically possible to form a MANET is not sufficient to allow for effective usage of the resources contained within these networks. We believe that the main prerequisite for such usage is the ability to advertise and find services. Examples for services range from the ability to print documents, to the processing of database queries, to the delivery of documents or more generally information, and to the usage of specialized technical equipment. With this ability it becomes possible to use the

*A more detailed version of this paper can be found in [1]. This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG) within SPP 1140.

distributed knowledge, computing power, and capabilities spread throughout a MANET.

Unfortunately, existing mechanisms for service discovery are not well suited for MANETs, since they either rely on central directory servers or produce a huge message overhead. More sophisticated approaches analyze the content of the service requests to route them semantically. Typically, they support rather primitive service descriptions only. We believe that for an efficient usage of services in MANETs (as well as in any network), service discovery based on semantically rich, ontology based service descriptions needs to be supported.

In this paper, we propose a new method for service advertisement and discovery in MANETs. The basic idea is to define a two-dimensional overlay structure, called *lanes*, which is similar to, but less strict than the one used in the Content Addressable Network (CAN) [2]. One dimension of this overlay propagates service advertisements, the other one distributes service requests. This results in a fault-resilient and efficient structure, which can be used for semantics-based service discovery. While we are working on developing appropriate service descriptions, the description language is well beyond the scope of this paper. Furthermore, the approach described here is completely independent of the concrete service description used.

The remainder of the paper is organized as follows: In Section 2, we discuss existing approaches for service trading and explain why they are not usable in MANETs. Section 3 introduces CAN and discusses how its structure can be used for service trading, but also why it cannot be used “as is” in MANETs. Thus, we then introduce in Section 4 a more lightweight overlay structure, called lanes, by softening some of the conditions CAN is based on, and describe in detail how this overlay can be used for service discovery and advertisement. The paper ends with a conclusion and an outlook to future work in Section 5.

2 Related Work

Service discovery for distributed environments is often designed for internet based networks and thus does not take the characteristics of MANETs into account. As we have examined these techniques in detail in [1], we give a short overview here only. Generally, we can distinguish four main approaches: **Central service repositories** as used in CORBA’s Trading Object Service, Jini, Bluetooth, the Service Location Protocol SLP, in file sharing protocols like Napster, in the area of web services as UDDI registries, or agent environments like LEAP [3] and MircoFIPA-OS [4]. **Service discovery by flooding** like the Simple Service Discovery Protocol (SSDP) [5] and file sharing protocols like Gnutella and JXTA-Search. **Hash-based Approaches** like OceanStore [6], Globe [7], Chord [8], Freenet [9], Tapestry [10], and CAN [2]. **Semantic Routing**, i.e. intelligently choosing nodes to forward a request by inspecting its semantics, like the Intentional Naming Scheme (INS) [11], the agent-based discovery framework Allia [12], our Multi-layer Clusters [13] and Service Rings [14], NeuroGrid [15], or the Routing Indices approach [16].

In a nutshell, they are not suitable for an application in highly dynamic networks (in the case of server based architectures) or assume the existence of large bandwidths (in the case of service discovery by flooding), which are not available in most mobile devices because of their limited resources. Hash-based architectures, on the other hand, only support semantically weak service descriptions and therefore are not able to fulfill our needs. Semantic routing is a step in the right direction, but the existing approaches are not constructed for our demands.

3 Using the CAN Overlay Structure for Service Trading

3.1 Basic Idea of CAN

When examining existing approaches to service discovery, we notice that they only insufficiently mediate between the user's requested functionality (i.e. efficient, semantic service trading) and the characteristics of a mobile ad hoc network (highly dynamic topology and weak device capabilities). Thus, it seems reasonable to insert an additional layer above the transport layer (Layer 4) that bridges these two parts. To achieve this, the new layer should build, maintain, and offer an overlay structure that (1) is constructed to optimally serve the mechanisms of service trading (i.e., announcement and discovery of service descriptions) and (2) that can be efficiently adapted to the constantly changing topology of the underlying network. The structure used in the *Content Addressable Network* (CAN) [2] offers a good starting point for an overlay structure that can be used to gain the above-mentioned properties.

3.2 Application of a 2-dimensional CAN Structure to Service Discovery

In this section, we show how CAN's overlay structure can be used for service trading based on a semantically rich service description. However, the rich semantics prevents us from simply using CAN's hashing mechanism to determine the storage node for a service description because the employed hash functions only preserve semantic similarity in case of very simple descriptions (e.g., keywords combined by Boolean operators, see [2]). Nonetheless, if we want to support complex service descriptions, which might even be based on ontology languages, it is not possible to use their semantic content for determining the storage location. Therefore, we have to separate the overlay structure from the concrete description and build it by using the fundamental semantics of service trading, only, i.e. there are two orthogonal dimensions: one for announcing offered services and one for searching suitable services. These two aspects can be directly described by a 2-dimensional CAN overlay. The first dimension (in the following the y axis) defines the direction and the devices where service offers have to be stored, the second dimension (in the following the x axis) defines where services have to be searched. On average, this algorithm distributes the descriptions to \sqrt{n} nodes where n is the total number of nodes in the network.

3.3 Guaranteeing the Structural Conditions in MANETs

RATSANY et al. also present techniques for guaranteeing the integrity of the CAN overlay even in case of networks changes in the underlying layer. These algorithms are specially designed for the characteristics of peer-to-peer networks; it is questionable whether they can be transferred to MANETs without change. One problem arises, for example, in case of node entrance: The algorithm to log into the CAN structure does not try to find an optimal neighborhood for the new node, but chooses arbitrary neighbors resulting in the split of the hypercuboid of a randomly chosen node in the network. Moreover, this neighborhood is not optimized later leading to increasingly inefficient links that are not aligned with the physical network topology. In [2], the authors present a method to attenuate these effects by introducing commonly known landmark nodes. The tuple of distances to these landmarks indicates an area of similarly located nodes. Nodes from this area should be preferably chosen as the new node's neighbors resulting in a higher adaption of the CAN structure to the network topology. However, this method is not applicable to ad hoc networks because of the lack of fixed, commonly-known nodes. Another problem arises from the detection of unreachable nodes. In CAN, this is achieved by relatively extensive ping messages, which each node sends to all of its neighbors (normally at least four). For ad hoc networks, this is far too expensive, as it requires too much of the rare bandwidth.

Generally, the strict grid structure of CAN seems to hamper the transference to mobile ad hoc networks. Thus, the idea is to weaken the structural conditions of the CAN overlay so that they a) are applicable to MANETs and b) still offer the possibility to efficiently trade services. Such a structure is presented in the next section.

4 A Lightweight Overlay for Service Discovery in Ad Hoc Networks

As we have seen in the previous section, the structural conditions of CAN cause some severe problems in highly dynamic MANETs because of the strict grid organization, in which each node (except for marginal nodes) has to maintain at least 4 links to neighboring devices (in case of two dimensions). For MANETs, it is therefore imperative to weaken these conditions. This could be possible because of the fact that for service discovery the query only needs to be passed to a set of devices that cover the complete x axis, their y coordinate does not matter. Thus, request messages do not necessarily have to be in line with their searching device. This allows to abolish the fixed assignment of nodes in x direction, i.e. there are only *lanes* of nodes, which are loosely coupled. Within a lane, there is still a strict relationship of predecessors and successors and also the lanes are arranged in a well defined order. From the outside, all nodes within one lane can be treated equally, if sending a service request to that lane – any arbitrary node in it has full information to handle the request. As a consequence, *anycast routing* can be used to send messages from lane to lane

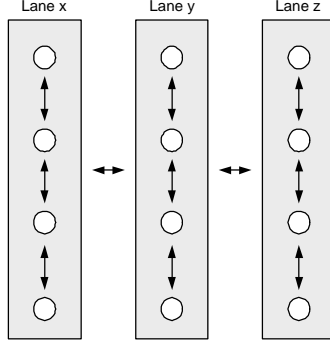


Figure 1: Within a lane, nodes are fixedly ordered and each of them knows its predecessor and its successor. The lanes themselves are loosely coupled: Nodes in the same lane share the same anycast addresses, which allows to use anycast routing for sending messages from lane to lane.

when nodes belonging to the same lane share the same *anycast address*. Figure 1 gives an outline of this idea.

In the following section, the formal conditions of the lane structure are introduced. After that, we will show the mechanisms to trade services on top of such a structure. The algorithms that guarantee the correctness of the structure are presented in the subsequent section.

4.1 Formal Conditions of Lanes

Definition 1 [Valid Lanes Overlay]

Let $D = D_x \times D_y = [d_{x0}, d_{x1}] \times [d_{y0}, d_{y1}]$ be a rectangle with $d_{x0}, d_{x1}, d_{y0}, d_{y1} \in \mathbb{N}$, i.e. its boundaries are natural numbers. A network forms a valid Lanes overlay if the following conditions are fulfilled:

1. Each node N in the network “owns” one nonempty rectangle $d^N = d_x^N \times d_y^N = [d_{x0}^N, d_{x1}^N] \times [d_{y0}^N, d_{y1}^N]$ with $d_x^N \subseteq D_x$ and $d_y^N \subseteq D_y$ and $d_{x0}^N, d_{x1}^N, d_{y0}^N, d_{y1}^N \in \mathbb{N}$.
2. For each two nodes N and K we have: d_x^N and d_x^K are equal (so the two nodes are in the same lane) or disjoint (so they are in different lanes).
3. All these distributed rectangles d^N are disjoint and their union exactly yields the complete space D .
4. If d^N and d^K are in the same lane and adjacent in y direction (i.e. $d_{y1}^N = d_{y0}^K$ or $d_{y0}^N = d_{y1}^K$), then node N knows the address of node K . We denote N 's upper neighbor with $N.T$ and its lower one with $N.B$, if existent.
5. Each node N owning a rectangle with $[d_{x0}^N, d_{x1}^N]$ is addressable by the two anycast addresses d_{x0}^N and $d_{x1}^N - 1$. Therefore, neighboring lanes can be addressed by $d_{x0}^N - 1$ (if $\neq d_{x0} - 1$) and d_{x1}^N (if $\neq d_{x1}$).

An example of such a lane structure is depicted in Figure 2. Note that the virtual rectangles directly (in case of the anycast addresses) or indirectly (in case

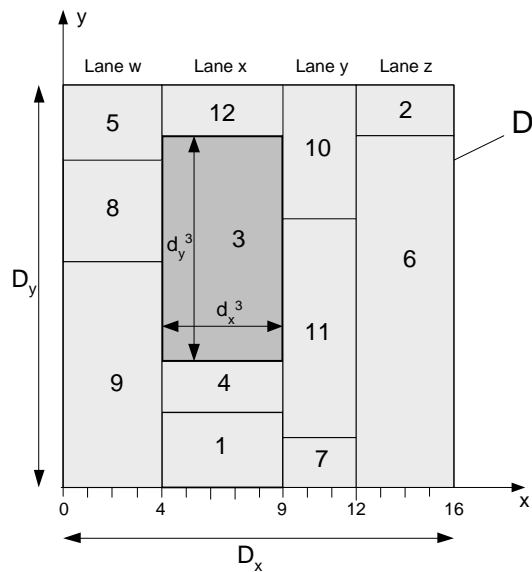


Figure 2: A correct lane structure. Node 3 owning the highlighted rectangle has to know the addresses of Node 4 and 12 because their rectangles are adjacent in y direction. Like any other node in Lane x , Node 3 has the two anycast addresses 4 and 8 resulting from its interval $d_x^3 = [4, 9)$.

of the unicast addresses within a lane) define the addresses a node has and also has to know. Therefore, analogously to CAN, algorithms changing the overlay structure only need to operate on this virtual space – the corresponding address adaptations are automatically defined by this.

To sum up, we use a combination of a proactive structure within one lane (allowing to use unicasts to well-known predecessors and successors) and a reactive structure between the lanes (leading to anycasts to reach an arbitrary node in neighboring lanes). The following section will show how these different techniques map to the different characteristics of service trading.

4.2 Service Announcement and Discovery in Lanes

The algorithms for service trading on top of a lanes structure are similar to the ones for a general CAN structure (see Section 3.2). As the announcement of service descriptions is an event whose long-term effects (i.e. the storage of the description on several devices in the network) have to be maintained consecutively by the network (with the help of leases for instance), it seems reasonable to use the proactive, inner lane structure as communication path to distribute the offer descriptions. On the other hand, searching for services is a one-time action, which does not lead to a change that needs to be persisted. Thus, flexible anycast communication between the lanes seems to be a suitable direction of communication. With that knowledge, we can develop the trading algorithms:

Algorithm 1 [Service Announcement]

Node N wants to offer service s .

1. N sends a **ServiceOffer** message containing a description of s and N 's address to $N.T$ and $N.B$ (if existent).
2. Each Node X receiving a **ServiceOffer** from its upper (lower) node $X.T$ ($X.B$), stores it and forwards it to its opposite neighbor $X.B$ ($X.T$) (if existent).

Algorithm 2 [Service Search]

Node N wants to search for a device offering s or a similar service.

1. N checks whether it has stored an appropriate service description on its own device (stemming from the own lane). If not, N sends a **ServiceRequest** message containing the description of s and N 's address to the neighboring lanes using anycast routing to the addresses $d_{x0}^N - 1$ (if not $d_{x0} - 1$) and d_{x1}^N (if not d_{x1}).
2. Each node receiving a **ServiceRequest** checks its service description memory for suitable services in the own lane. If one of the descriptions fits, a **ServiceFound** response containing the unicast address of the service offerer is directly sent back to the requestor. If no description fits, the request message is forwarded to the opposite neighboring lane (using anycast addresses like in 1.). If such a lane is not existent, a **ServiceFailure** message directly addressed to the service requestor is sent back.

4.3 Asserting the Structural Conditions in Lanes

In MANETs, similar situations as in peer-to-peer networks can arise that result in violating the structural conditions of the overlay. Due to their high dynamics, in general, ad hoc networks face these problems more often and more severely. Therefore, the correction algorithms have to be more flexible. We consider intended login and logoff of a device as well as broken overlay links. In the following, we show that the flexible lanes structure is well suited for dealing with these problems in ad hoc networks by presenting possible correction algorithms.

4.3.1 Intended Node Login/Logoff

Logging into the network is not very difficult if new nodes are only permitted to join *exactly one existing* lane. Then, the entrance of a node neither results in a new lane¹ nor merges existing lanes. However, this might happen in a possible correction step when a large lane consisting of too many nodes is divided into two neighboring lanes or when two short lanes are combined to one new lane (see [1] for more details on these algorithms). Having said this, we can develop the login algorithm:

¹One exception is the first node: it initiates the first lane by claiming the entire space D .

Algorithm 3 [Login]

Node N wants to join the network.

1. N broadcasts a **LoginRequest** containing its own address to all nodes it can reach within a single hop.
2. Each node X receiving a **LoginRequest** sends a **LoginOffer** containing its address, the address of its upper neighbor $X.T$, and the length of its lane (if known) back to the requestor.
3. N collects these offers and chooses one of them. It prefers offers of a node X if $X.T$ has also sent an offer (resulting in efficient one-hop connections for N) or X 's lane length is short (helping to avoid lane splittings).
4. N sends **LoginAccept** messages to the chosen X and $X.T$.
5. X halves its rectangle horizontally and sends a **LoginConfirm** message to N . This message contains the upper half of X 's rectangle as well as X 's stored service descriptions. By that, X and $X.T$ have to update their neighbors to integrate N into the lane.
6. When N receives the confirmation, it stores the rectangle and the descriptions and is able to use the benefits of the network structure by offering and/or searching for services.

Logging off from the network is simple too, thus the algorithm is omitted. The only important aspect to mention is that all offered services of the leaving node are removed with the help of a **OfferRemove** message that is forwarded through the lane. Note that no extensive lease mechanism is needed as each invalid service description is explicitly removed.

4.3.2 Broken Connections Between Nodes

To repair broken links within one lane, we first have to detect them. This is done by periodical **Ping** messages that every node N sends to its upper neighbor $N.T$ containing N 's and $N.B$'s address. Generally, if such a message is missing for a certain time, a Node X assumes a broken network connection to its lower neighbor $X.B$ (because $X.B$ unintentionally left the lane or a network partition has occurred). In this case, X tries to contact $X.B.B$ (known from previous ping messages) in order to inform it with a **LaneBroken** message about the broken connection to $X.B$. If this is possible $X.B$ seems to have vanished from the network unexpectedly and the lane is repaired by connecting $X.B.B$ and X . Moreover, the descriptions of $X.B$'s service offers are explicitly removed from all devices by forwarding a **OfferRemove** message through the lane. If, on the other hand, X cannot reach $X.B.B$ either, the network is probably partitioned and the lane is split into two parts. In this case, X (and $X.B$ in the other half) inform their remaining lane members about the partition, which invalidate service descriptions they have stored stemming from offerers in the other half of the lane. Furthermore, X adds the complete rectangle of the lower split-off part to its own rectangle resulting in the rectangle $[d_{x0}^X, d_{x1}^X] \times [d_{y0}, d_{y1}^X]$. In the same

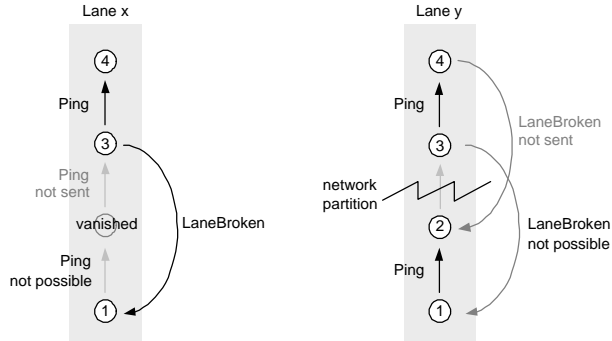


Figure 3: Algorithms for detecting broken links. On the left hand side, Node 2 has unexpectedly vanished from Lane x , which can be recognized by Node 1 and 3 because of problems with sending or receiving **Ping** messages. As the **LaneBroken** message can still be sent, the Lane can be repaired by connecting Node 1 and 3. On the right hand side, the network has been partitioned between Node 2 and 3, which can also be recognized because of **Ping** message failures. In this case, no **LaneBroken** message is received for different reasons resulting in two separated lanes.

way, $X.B$'s rectangle is augmented by the complete rectangle of the upper part of the lane.

The sender of a **Ping** message detects faults in a similar way: if N cannot send its **Ping** message to $N.T$, it assumes that $N.T$ has left unintendedly. As $N.T.T$ will also recognize this problem, N waits for the appropriate **LaneBroken** message. If it arrives within a given time period, indeed, $N.T$ has vanished and the lane is repaired by connecting N and $N.T.T$, otherwise N assumes a network partition and proceeds as described above.

Note that each node in the above-mentioned algorithms is responsible for pinging *one* lane member only. In contrast to that, in CAN, each non-border node has to ensure the validity of at least four neighboring nodes, which would consume much of the available bandwidth in mobile networks. An example for applying this algorithm can be found in Figure 3.

4.3.3 Network Partition and Reintegration

Network partition and reunion pose difficult problems for the consistency of overlay structures. In the case of lanes, network partitions can be detected with the methods from the previous section leading to two split lanes. Nevertheless, service trading can continue separately in the two parts. Furthermore, the overlay structures of the partitions can change independently.

When the radio contact between separated partitions is reestablished later, we must assure that now again services can be offered and/or found in the previously unconnected parts. Thus, a reintegration of both overlay structures is inevitable. Generally, we face two major problems: a) the reunion must be detected and b) the overlay structures of the separated parts must be combined to one new, regular overlay structure. In the following, we will analyse these

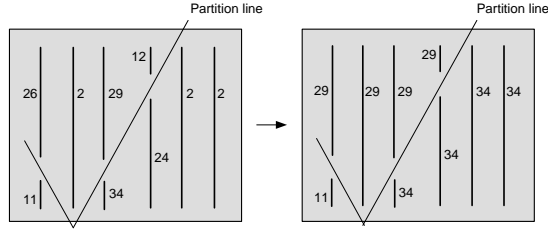


Figure 4: Propagation of partition numbers in case of network partition. Originally, every lane in the network had the partition number 2 (not shown). At a certain time, the network has been partitioned by the marked line. As three lanes are affected by this, six new partition numbers are assigned to the remaining lane parts: 26 and 11, 29 and 34, and 12 and 24 (left picture). After propagating these numbers in the own partitions, only three numbers remain leaving three uniquely denoted partitions: 11, 29 and 34 (right picture).

problems and sketch solutions for solving them in a lanes overlay.

To detect network reunion, it is sensible to assign different partition numbers to different partitions. Thus, when detecting network partitions by the algorithms in Section 4.3.2, each of both lane parts adds a random value taken from the set $\{1, 2, \dots, s\}$ to its partition number resulting in a new, higher partition number². After that, the two parts check, whether their originally neighboring lanes are still reachable (via their known anycast addresses). If yes, the lane with the lower partition number overrides it with the higher one and propagates this higher value to the remaining lanes in opposite direction. On the other hand, if no previously neighbored lane can be found anymore, the lane should try to reach other neighborless lanes by broadcasting a message in the own partition. In any case, after this process, each network partition has its own unique id. Figure 4 shows an example of this idea. To detect a network reunion, **Ping** messages are extended by the sender's partition number. Generally, we have the following rule: If Lane L_1 overhears a **Ping** message of L_2 that contains a different partition number than the own one, L_1 has detected a network reunion and reintegration steps have to be taken. As **Ping** messages are sent periodically by all nodes, network reunions are detected quickly and reliably.

The reintegration of two lane structures is achieved by connecting the separated lanes. To do that, L_1 prompts L_2 to establish a connection between their end nodes. In this case, the complete rectangle of Lane L_1 is horizontally halved and redistributed equally amongst the nodes of L_1 and L_2 . As the old space of L_2 is overridden, L_2 loses its old anycast addresses and adopts the two addresses of L_1 . Furthermore, the stored service descriptions are shared between L_1 and L_2 . In case that L_1 and L_2 have already been connected earlier, this last step can be omitted by reactivating previously just invalidated service descriptions. In any case, L_2 informs its originally neighbored lanes to have them connected to its new neighbored lanes, too.

²If s is large enough, we can assume that each lane part chooses a unique value. To avoid an overflow of the strictly increasing partition numbers, the number of a partition is set back to a random number between 1 and s in times of low network changes.

4.4 Optimizing the Lanes Structure

A structure following the rules of being a correct lanes overlay does not necessarily yield an efficient overlay. On the one hand, the logical links constantly need to be adapted to the changing physical network conditions, on the other hand, the overall structure must fit to the usage profile. In [1], we propose algorithms for tightening inefficient inner lane connections as well as for splitting oversized or merging undersized lanes.

4.5 Advantages of Lanes

Lanes have several advantages over existing approaches. This is particularly true when taking into account the characteristics of ad hoc networks and complex, semantic service descriptions. The main reason for this lies in its lightweight structure. Compared to CAN's strict hypercube structure, lanes require the fulfillment of much fewer and weaker structural conditions. This makes them better suited for highly dynamic network topologies. For instance, the detection of node failures, network partitions and reintegration can be achieved with just one periodic ping message per node, whereas in the CAN structure each node needs to constantly keep track of all of its neighbors resulting in at least four periodic ping messages for inner nodes.

Compared to the approaches from Section 2, lanes do not need dedicated nodes for any tasks and yet it is not necessary to flood the network to find or announce services. Since lanes offer a possibility for semantic service search without using semantic information to form the overlay (other than the fact that the two dimensions stand for service announcement and search, respectively), they are independent of the service description, which enables complex description languages. However, the assignment of service announcements to the inner lane structure and of service requests to inter lane communication is not arbitrary: For instance, the fixed structure within a lane allows to forego periodic refreshments of service announcements via a lease concept, since changes in the set of service offers are propagated immediately within the lane.

To summarize, lanes are the ideal compromise between weakly structured approaches, which are easily adapted to network characteristics but typically scale poorly, and highly structured approaches with optimal adaptability to user profiles at the cost of highly inefficient maintenance in dynamic network topologies.

5 Conclusions and Future Work

In this paper, we provided an overview of existing service discovery approaches in ad hoc networks. In order to overcome their drawbacks, we identified CAN as a promising basis for service discovery and introduced an application that copes with semantically rich service descriptions. The mismatch of CAN based service discovery and MANET's profile led us to ease CAN's structural restrictions by conceiving Lanes. We proposed protocols that maintain Lanes in the presence

of diverse events that are typical of ad hoc networks. Lastly, we pointed out the benefits of Lanes compared to existing service discovery approaches.

In future, we intend to specify algorithms for automatically tuning network parameter (like lane lengths or time between Ping messages) to the profile of the network and the user. Furthermore, we are currently examining the advantages of hierarchies of lanes, which are formed by aggregating complete network regions into clusters that act as “nodes” in a higher lane structure. Currently, we are implementing all presented protocols for testing in our emulator DIANEmu [17].

References

- [1] Klein, M., König-Ries, B., Obreiter, P.: Lanes – a lightweight overlay for service discovery in mobile ad hoc networks. Technical Report 2003/6, Universität Karlsruhe, Faculty of Informatics (2003)
- [2] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. In: Proceedings of ACM SIGCOMM 2001. (2001) 161–172
- [3] LEAP: Lightweight extensible agent platform. (<http://leap.crm-paris.com>)
- [4] Tarkoma, S., Laukkanen, M.: Supporting software agents on small devices. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems. (2002) 565–566
- [5] Internet Engineering Task Force: Simple service discovery protocol - internet draft v1.03. (http://www.upnp.org/download/draft_cai_ssdv1.03.txt)
- [6] Kubiawicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of ACM ASPLOS, ACM (2000)
- [7] Bakker, A., Amade, E., Ballintijn, G., Kuz, I., Verkaik, P., van der Wijk, I., van Steen, M., Tanenbaum, A.S.: The globe distribution network. (In: Proceedings of the USENIX Annual Conference) 141–152
- [8] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, ACM Press (2001) 149–160
- [9] Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science **2009** (2001) 46+
- [10] Zhao, B.Y., Kubiawicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley (2001)
- [11] Adje-Winoto, W., Schwartz, E., Balakrishnan, H., Lilley, J.: The design and implementation of an intentional naming system. In: 17th ACM Symposium on Operating System Principles (SOSP 99). (1999)
- [12] Ratsimor, O., Chakraborty, D., Tolia, S., Kushraj, D., Kunjithapatham, A., Gupta, G., Joshi, A., Finin, T.: Allia: Alliance-based service discovery for ad-hoc environments. In: ACM Mobile Commerce Workshop. (2002)
- [13] Klein, M., König-Ries, B.: Multi-layer clusters in ad-hoc networks - an approach to service discovery. In: Proceedings of the First International Workshop on Peer-to-Peer Computing (Co-Located with Networking 2002), Pisa, Italy. (2002) 187–201
- [14] Klein, M., König-Ries, B., Obreiter, P.: Service rings – a semantical overlay for service discovery in ad hoc networks. In: International Workshop on Network-Based Information Systems (NBIS2003), Workshop at DEXA 2003, Prague, Czech Republic. (2003)
- [15] Joseph, S.: NeuroGrid: Semantically routing queries in peer-to-peer networks. In: Intl. Workshop on Peer-to-Peer Computing (at Networking 2002), Pisa, Italy. (2002) 202–214
- [16] Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proceedings of the International Conference on Distributed Computing Systems (ICDCS). (2002)
- [17] Klein, M.: DIANEmu – a java-based generic simulation environment for distributed protocols. Technical Report 2003/7, Universität Karlsruhe, Faculty of Informatics (2003)