Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach

Vincent Lenders[*], Martin May and Bernhard Plattner
Swiss Federal Institute of Technology (ETH Zürich), Switzerland
Email: {lenders, may, plattner}@tik.ee.ethz.ch

Abstract

Service discovery in mobile ad hoc networks is challenging because of the absence of any central intelligence in the network. Traditional solutions as used in the Internet are hence not well suited for mobile ad hoc networks. In this paper, we present a novel decentralized service discovery mechanism for ad hoc networks. The basic idea is to distribute information about available services to the network neighborhood. We achieve this by using the analogy of an electrostatic field: A service is modeled by a (positive) point charge, and service request packets are seen as (negative) test charges which are attracted by the service instances. In our approach, we map the physical model to a mobile ad hoc network in a way where each network element calculates a potential value and routes service requests towards the neighbor with the highest potential, hence towards a service instance. Our approach allows for differentiation of service instances based on their capacity. We define the required protocols and methods which we implemented in a network simulator. Using extensive simulations, we evaluate the performance and robustness of the mechanisms. The results indicate good performance and convergence even in highly mobile environments. We believe that this technique can and should be further exploited, e.g., as a routing protocol in mobile networks.

1. Introduction

Wireless mobile ad hoc networking has recently gained a lot of attention in research. A Mobile Ad hoc NETwork (MANET) represents the ultimate scenario where the network is operated without any fixed infrastructure support at all. Such networks can be deployed very quickly and are inexpensive as they do not invoke basic infrastructure costs. MANET applications cover various areas, such as military or post-disaster rescue operations, temporary group collaboration at conferences or lectures, sensor networks, and many others. Due to the absence of any fixed infrastructure support in MANETs, the participating nodes must provide the basic communication primitives such as routing, address allocation, name resolution, or service discovery themselves. To provide a certain degree of flexibility, MANETs must configure and operate automatically without human intervention. Automatic network configuration is especially difficult in a MANET due to the very dynamic nature of the system. The dynamism arises from the fact that nodes may join or leave at any time, that nodes are expected to move, and that the properties of the wireless medium are time variant.

Past research efforts for MANETs have primarily focused on packet routing. In this paper, we focus on the issue of service discovery which is of fundamental importance. Network support for service discovery is required when a client application desires to access a service provided by a host or server. Applications scenarios for service discovery in MANETs are manifold:

- In MANETs, some of the connected hosts might have, in addition to the ad hoc network interface, an external connection to the Internet. Such nodes may announce this ability as a service to the participating ad hoc nodes. Using service discovery, members of the MANET are then able to use such a *gateway* service.
- In an electronic parking system, a service is defined differently. In such a scenario, implemented as a sensor network, each parking slot is equipped with a sensor. Whenever the slot is not occupied, the sensor announces a *parking service* and a guidance system able to route the car to the parking slot.
- Using their wireless hand-held device or notebook, participants in collaborative applications or distributed gaming environments need to discover application or game servers before participating in a session.

From the possible application scenarios of mobile ad hoc networks, we derive two major requirements for a service discovery system specific to MANETs:

1. *Optimal service selection.* If the same service is offered by multiple instances, “good” service selection greatly improves the overall system performance. On

[*] Partly founded by the Swiss National Science Foundation (SNF) under grant 200021-103578

one hand, selection of a close service, localizes communication and therefore minimizes inter-node communication and interference. At the same time, it increases the total network capacity. On the other hand, the quality of service perceived by the client can be augmented by selecting a "good" service with high service capacity. For example, a gateway service attached to the Internet with a 100 MBit/s link is preferable over a service with a 1 MBit/s link.

2. *Robustness faced to mobility.* The network is by nature very dynamic as nodes are free to join, leave or move at any time. The system performance must remain stable when frequent changes in the network topology occur.

Existing service discovery mechanisms are not well suited for wireless ad hoc networks since they address these issues only partially or not at all. In this paper, we propose a novel approach for service discovery in wireless mobile ad hoc networks that fulfills the aforementioned requirements. Due to the nature of ad hoc networks, our approach is implemented in a totally distributed way, without any central servers or infrastructure. We assume that every node participates in the service discovery process. When a service appears in the network, it advertises itself. Intermediate nodes store and exchange information about offered services. The discovery process is then initiated from a client by sending out a query message which specifies the desired service type. Such a query is forwarded towards a service instance matching the service type specified in the query. When a discovery message arrives at the service instance, the service sends back a reply to the client. If multiple service instances of the same type exist, the service instance discovered by the client is not arbitrary. The discovery system "selects", on behalf of the requesting client, a service instance based on two metrics, the network *distance* (number of hops) between client and service instance and the *capacity of service* (CoS). For example, the CoS of an Internet gateway service may indicate the link capacity of its Internet connection. In the same way, the CoS for a printer can be used to indicate the print speed. Alternatively, the CoS can be used to express an average load to perform load balancing.

The service selection algorithm is distributed and does not involve interaction with the client. Our approach to select a service and thus determine how to forward service queries, is inspired from the physics, specifically of test charges in an electric field. Any negative test charge moves along the field's flux line towards a positive charge. The direction of the flux line is determined by evaluating the gradient of the field. Thus, to draw the analogy, we associate a query with a negative test charge and the capacity of a service instance (CoS) with a positive point charge that creates a field. Figure 1 shows a simple example with two service instances and one client. The resulting potential from the two charges at the service instances is used to deliver a query from a client to a service. In this example, since both services have the same charge, the query is delivered to the closer service on the left side.

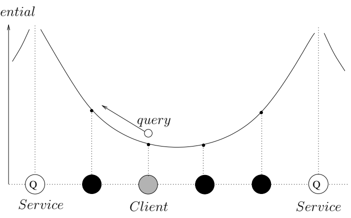


Figure 1. Service discovery with potentials.

The main contributions of this paper are as follows. We show how to map the concept of electric fields[1] to solve the service discovery problem in MANETs. The proposed solution supports service selection based on client-service distance and capacity of service. We show how to implement the solution in a distributed and efficient way and analyze the effect of node mobility on the system performance with simulations. Note that the proposed service discovery mechanisms are independent and even work in the absence of any underlying routing protocol.

The rest of the paper is organized as follows. The next section describes the details of our novel, field-based approach to service discovery. Our implementation and its evaluation is described in Sections 3 and 4. We then describe related work in Section 5 and conclude the paper in Section 6.

2. Service Discovery with Potentials

In our approach, scalar fields are defined on the network over which service queries are forwarded towards service instances. A scalar field is analogous to a potential φ in electrostatics resulting from electrical point charges. The potentials of point charges define a distribution with maxima at the point charges. Analogously, we consider the capacity of service (CoS) as a point charge Q, defining a scalar field on the network with peaks at nodes hosting service instances. Figure 2 depicts an example potential field comprising ten service instances (charges). The potential distribution results from discrete values defined at the network nodes.

The charges that contribute to a potential must be of the same service type. For example, printers contribute to the potential $\varphi^{(1)}$ of *service-type=printer*. However, a camera service contributes to the potential $\varphi^{(2)}$ that belongs to

[1] Throughout the rest of this paper, we always consider the potential resulting from point charges. The potential can be directly calculated from the electric field and vice versa.

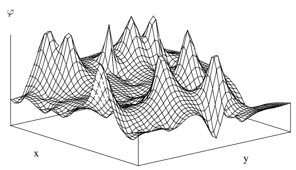


Figure 2. Service potential.

service-type=camera. As a consequence, multiple potentials are defined and co-exist on the network. When a client searches for a service, it specifies in the request the desired service type. The query is routed to a service instance based on the potential of that service type. Throughout the rest of this paper, we consider only one service potential and use φ to denote the potential of that service type.

2.1. Potentials

Consider a service instance at node n_j with a charge Q_j. The potential at any node n resulting from this charge is defined [2] as

$$\varphi_j(n) = c \cdot \frac{Q_j}{dist(n, n_j)} \tag{1}$$

where c is a constant and $dist(n, n_j)$ is the distance between node n and n_j. In physics, the distance between two nodes is defined as a geometric distance in meters. In a network however, the distance between two nodes is often reflected by the number of hops between them. Therefore, we define the potential as follows

$$\varphi_j(n) = \frac{Q_j}{\mid n - n_j \mid} \tag{2}$$

where $\mid n - n_j \mid$ is the shortest distance in hops from node n to n_j. For simplicity, we set the constant to $c = 1$ as it does not impact the discovery decisions. Note that in principle, other distance metrics for $dist(n, n_j)$ could be used including the transmission delay, link quality, etc. Throughout the rest of this paper, we use the shortest distance in hops for $dist(n, n_j)$ as defined in Equation (2).

Now consider N service instances of the same type (for example N printers). The resulting potential is calculated as

$$\varphi(n) = \sum_{j=1}^{N} \varphi_j(n) = \sum_{j=1}^{N} \frac{Q_j}{\mid n - n_j \mid} \tag{3}$$

which is simply a linear superposition of all potential terms. Note that the resulting potential value at nodes with a service instance is $\varphi \to \infty$.

2.2. Query Forwarding

With the use of this potential function, a service query packet is forwarded from a client to a service analogous to a test charge. The main difference from physics is that a charge moves along any path in an electric field, whereas query packets only move along the network links. In our approach, all Qs are positive. Thus, a negative test charge follows the direction of the steepest potential ascent. As a result, a node x forwards a query packet to its neighbor y_i that has the highest potential among its neighbors:

$$next\ hop(x) = y_i \,:\, \varphi(y_i) \geq \varphi(y_k) \wedge \varphi(y_i) > \varphi(x) \quad \forall y_k \in NB(x), y_k \neq y_i \tag{4}$$

where $NB(x)$ is the set of neighbors from node x. In cases where multiple neighbors of x have the same maximum potential value, the next hop is chosen arbitrarily. A service query packet has reached a service instance (its destination) when it arrives at a node with a potential value of $\varphi \to \infty$. If the potential function is monotonically increasing, query packets are guaranteed to eventually reach service instances. However, due to the constraint in our model that packets can travel only over links between nodes and not in any direction as in physics, it is possible that the potential distribution shows a local maximum at a node which does not provide a service:

$$\varphi(x) \geq \varphi(y_k) \qquad \forall y_k \in NB(x) \tag{5}$$

Note, that such local maxima emerge very rarely and only with specific topologies (for example star topologies with a large number of service instances at the edges). In all experiments we conducted, using random network topologies with random node motion, we never experienced local maxima in the potential distribution. To address the problem of local maxima, we propose the following solution. If ever a query reaches a node with a local maxima, the query changes the forwarding strategy from forwarding based on potential values and enforce "greedy" forwarding towards to closest service. Hence, the query will arrive at the closest service node.

2.3. Illustrative Examples

We now illustrate, based on simple examples, the basic properties of our approach for service discovery. We start looking at a potential which is defined by a single service instance (charge). We show that in this case, a client query is forwarded to the service over the shortest path. When

[2] In analogy to physics, the electrical potential at position $\vec{r}$ which relates to a point charge Q_j located at $\vec{r_j}$ is $\varphi_j(\vec{r}) = \frac{1}{4\pi\varepsilon} \frac{Q_j}{|\vec{r} - \vec{r_j}|}$. Note that this function is continuous whereas in our definition, the potential function has discrete values at the network nodes.

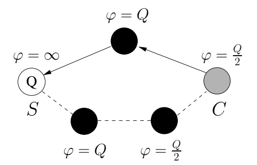


Figure 3. One service instance.

two service instances define the potential field, we distinguish two cases: (i) If both charges are equal, the query message is delivered on the shortest path to the closest service; and (ii) if the charge values are different, there is a tradeoff between proximity and intensity. We show that, with multiple service instances, service query packets are directed towards regions in the network with high service density.

2.3.1. Potential Function Resulting from a Single Charge

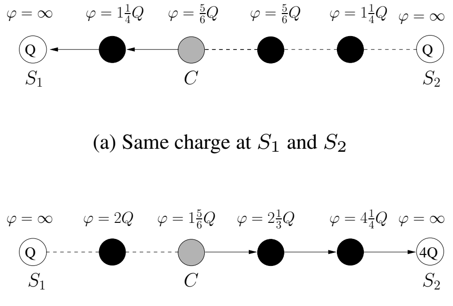
Consider a scenario with only one service instance S (see Figure 3). A charge Q is assigned to S. Client C sends a service query packet and we illustrate the traversed path (marked with arrows) of this packet. Note that there are two paths from C to S with a length of two and three hops. To calculate the potential value at each node, we must first determine the distance from any node to S. Using this distance and Equation (3), we derive the potential value at each node as given in Figure 3. The potential at node S is $\varphi \rightarrow \infty$. The potential of node C is $\varphi = \frac{Q}{2}$ because the shortest path from C to S is two hops. According to Equation (4), a node which forwards a query packet, forwards it to the neighbor node with the highest potential value. In this case, node C has two neighbors with potential $\varphi = Q$ and $\varphi = \frac{Q}{2}$, respectively. The query packet is therefore forwarded to the node with potential $\varphi = Q$. Next, the packet is forwarded to S since it has the highest potential value $\varphi \rightarrow \infty$. S is the final destination, namely the service instance. Note that the service query packet is forwarded to the service along the shortest path. Generalizing this observation, we claim the following:

Theorem 1 *If just one service instance of a given type exists, a service query packet from any client node in the network is directed to the service node along the shortest path.*

Proof: Let n_c be a client node that sends a service query packet to a service node n_s. Then, assume that node n_x is the next hop from the shortest path of n_c to n_s. Consider a neighbor node n_y of n_c such that $n_x \neq n_y$. Since node n_x is on the shortest path we claim that

$$\mid n_s - n_c \mid = 1 + \mid n_s - n_x \mid \leq 1 + \mid n_s - n_y \mid \qquad (6)$$

This implies that $\mid n_s - n_x \mid \leq \mid n_s - n_y \mid$. If the service at node n_s is the only service, the potential at node n_x is $\varphi(n_x) = \frac{Q}{|n_s - n_x|}$ and $\varphi(n_y) = \frac{Q}{|n_s - n_y|}$ at node n_y. Therefore, $\varphi(n_x) \geq \varphi(n_y)$. According to the forwarding rule (Equation (4)), a service query packet from n_c is forwarded to the node with the highest potential. Therefore, the packet is forwarded to n_x which is the next hop on the shortest path to n_s. ■



(a) Same charge at S_1 and S_2

(b) Different charge at S_1 and S_2

Figure 4. Two service instances.

2.3.2. Potential Function Resulting from two Charges

In general, there are many reasons for a client to select a close service instance. For example, localized communication generally reduces end-to-end delays or the probability of route failure due to mobility. It also reduces internode interference, which in turn increases network capacity. However, in the presence of more distant services with high CoS, it is, to some extent, reasonable to access more distant service instances to increase the quality of service perceived by a client. For example, consider two Internet gateway services in a MANET with a $100KBit/s$ and $100MBit/s$ connection link to the Internet. Unless a client is really much closer to the $100KBit/s$ service, it is reasonable to use the much faster $100MBit/s$ service to access the Internet. The next two scenarios show the properties of service discovery when exactly two service instances of the same type exist.

In Figure 4(a), a client node C is two hops away from S_1 and three hops away from S_2. If S_1 and S_2 both have a charge Q, we calculate the potential at all nodes using equation (3) as given in the picture. For example, the potential at C is equal to $\varphi = \frac{Q}{2} + \frac{Q}{3} = \frac{5}{6}Q$. Client C sends a service query packet to the left neighbor since its potential $\varphi = 1\frac{1}{4}Q$ is larger than the potential $\varphi = \frac{5}{6}Q$ of the right neighbor. Henceforth, C discovers service S_1. Note that in this case, C discovers service S_1 because it is closer, in terms of hops, than service S_2.

Now consider the scenario illustrated in Figure 4(b). The only difference is that a charge of $4Q$ is assigned to service

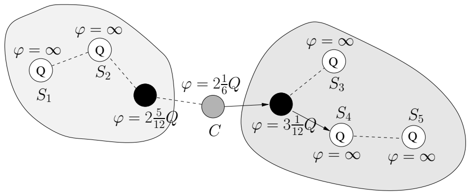


Figure 5. Multiple service instances.

S_2. The potential values at all nodes change from the previous scenario. The potential at C is now equal to $\varphi = 1\frac{5}{6}Q$. However, in this case, the potential $\varphi = 2\frac{1}{3}Q$ of the right neighbor is higher than the potential $\varphi = 2Q$ of the left neighbor. Therefore, a service query packet is sent via the right neighbor to service S_2. Note that this time, service S_2 is discovered which is more distant (in number of hops) than service S_1 because its charge intensity is higher.

We conclude that when two service instances have the same charge, a client discovers the service instance which is closest to him. However, when the service instances use different charges, the tradeoff between proximity and charge intensity has to be handled during the discovery process. Recall that the charge is a value to quantify the capacity of service (CoS) as for example, the print speed or link capacity. Thus, applying field theory to service discovery allows to exploit the natural proximity/intensity tradeoff resulting from the potential to select the appropriate service instance. We evaluate this tradeoff in more detail in Section 4.

2.3.3. Potential Resulting from Multiple Charges

An example scenario with five services (S_1 - S_5) of the same type is illustrated in Figure 5. An identical charge Q is assigned to all services. A service query packet from client C is sent to its right neighbor with potential $\varphi = 3\frac{1}{12}Q$ which is larger than the potential $\varphi = 2\frac{5}{12}Q$ of its left neighbor. The right neighbor of C then forwards the packet to either S_3 or S_4 as drawn in the picture because they both have an infinite potential value.

In this example, client C is equidistant from S_2, S_3, and S_4 which all have the same CoS. If we based our decision on simple distance and capacity metrics instead of using potentials to forward service queries, all three service instances would be considered "equally" optimal. However, with our approach, a query message from C will always be forwarded to S_3 or S_4. This is due to the summation of CoS and the consequential higher potential in the direction of the "service instance cloud". This example illustrates the benefit of using our potential function to forward queries. Assume that C sends a query packet towards S_2. If S_2 moves away or just disappears before the field values can be updated by the protocol, the query packet will be dropped at the relaying node. Now assume that C sends its query towards S_3 which in turn disappears. An intermediate node between C and S_3 is now able to react and forward the query to an alternate node which in this case is S_4 and successfully deliver the query. In other words, query packets are directed to network spots with large charge density. A large charge density may be the result of many small charges or few high charges close together. More generally, we claim that our approach adds a *probability of successful service delivery*. Hence, we increase the robustness of the system.

3. Implementation

In this section, we describe our implementation of the potential-based approach for service discovery in mobile networks. We present a mechanism to establish potential values at nodes and how to react to failures due to node mobility. We also describe an optimization that significantly reduces the control overhead of the protocol and we show that this is achieved without sacrificing the service discovery performance. The discovery mechanism is entirely based on local communication and does not rely on any underlying routing protocol. The only communication service required from the network layer, is the ability to send a packet to one (one hop unicast) or all local neighbors (broadcast). We assume that all nodes in the network are mobile and that all wireless links are bi-directional, i.e. if node s is able to transmit to node r, then node r can also transmit to node s.

3.1. General Overview

Our implementation is based on the soft state principle. We consider it unrealistic to expect service instances to de-register their profiles in a wireless ad hoc network. Service instances periodically advertise the service type or types they offer. These advertisements are flooded through the network within a limited scope. Each node temporarily stores recently received advertisements and calculates its potential value for each service type. After a timeout, advertisements simply expire if they are not updated. In addition, neighboring nodes periodically exchange their local potential values for all service types.

When a client searches for a service, it creates a service query message. This query message contains the service type of the desired service. We assume that clients and services share a common ontology to express the service types. Intermediate nodes have to relay this query message according to their potential value and the value of their neighbors (see Equation (4)). If a local maximum is detected (Equation (5)), a query is forwarded to the closest service instance. When a service instance receives a query message with the service type it provides, it replies to the client with a query reply message. The discovery process is terminated as soon as the client receives the query reply message which contains the network address of the service.

3.2. Protocol Messages

Four different message types are required to 1) advertise service profiles, to 2) exchange potential information between neighbors, to 3) send service queries, and to 4) reply to those queries.

3.2.1. Service Advertisements

Service instances need to periodically advertise the service they offer. A service advertisement contains the following items:

- *Service Type:* This item defines the type of service (e.g. *printer*).
- *CoS:* The capacity of service which is analogous to the charge Q. Therefore, all CoS values are positive. In practice, a common quantification guideline for the CoS will be required per service type. For example, we can define the Internet gateway service capacity as follows. A service with a $100 KBit/s$ connection has a CoS of 2 and a Internet gateway service with $10 MBit/s$ a CoS of 15. It is also possible to adjust the CoS value depending on the momentary load of a service. A 10 MBit/s gateway could start decreasing its CoS when advertising its service as soon as its traffic load increases.
- *Hop Count:* The hop count field is initially set to zero by the service. It is incremented by one at each hop when forwarded. Thus, it is the distance from the receiving node to the service (in hops), as used in Equation (3) to calculate the potential of nodes.
- *Service ID:* An identifier which uniquely identifies the service instance. This ID is generated locally. Different mechanisms exist to achieve global uniqueness using for example part of the MAC address [1] or a time value [2].
- *Sequence Number:* A number which is incremented each time the service instance re-advertises the service it provides. This item is required to detect if the advertisement is new, obsolete, or is a duplicate which has been delivered over an alternate path.
- *Maximum Advertisement Lifetime:* A value set by the service which specifies when the advertisement expires. When an advertisement expires, its contribution to the potential must be removed.
- *TTL:* This value is set by the service provider to limit the flooding scope of an advertisement.

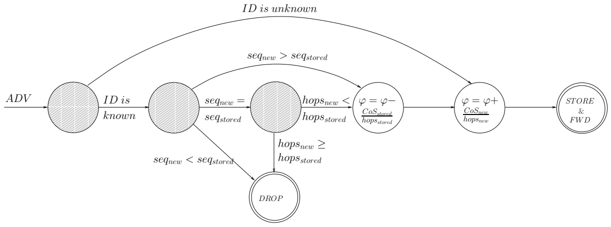

ID is unknown
$seq_{new} > seq_{stored}$
ADV
ID is known
$seq_{new} = seq_{stored}$
$hops_{new} < hops_{stored}$
$\varphi = \varphi - \frac{CoS_{stored}}{hops_{stored}}$
$\varphi = \varphi + \frac{CoS_{new}}{hops_{new}}$
STORE & FWD
$hops_{new} \geq hops_{stored}$
$seq_{new} < seq_{stored}$
DROP

Figure 6. Advertisement handling.

The way a node handles an incoming advertisement is pictured in Figure 6. This algorithm is required to determine if an advertisement is new, an update, or obsolete and consequently, adjust the potential value of the receiving node. When a node receives an advertisement, it first checks the *Service ID* to determine if a previous advertisement from the same originator has been received. If not, the advertisement is coming from a new service instance. In this case, the potential value for the advertised *Service Type* is created, the advertisement is stored, and then forwarded as a broadcast message to all neighbors. However, if a stored advertisement exists for that *Service ID*, the node must further look at the *sequence number* included in the advertisement. If the sequence number in the received advertisement (seq_{new}) is smaller than the sequence number from the stored advertisement (seq_{stored}), the received advertisement is obsolete and can be dropped. If seq_{new} is equal to seq_{stored}, the new advertisement is identical to the stored advertisement but must have travelled over an alternate path. The rule is to keep the advertisement which has travelled over the shortest path. Therefore, the packet is dropped if the *Hop Count* field of the new advertisement ($hops_{new}$) is larger than or equal to the *Hop Count* from the stored advertisement ($hops_{stored}$). If however, $hops_{new}$ is smaller than $hops_{stored}$ or seq_{new} is larger than seq_{stored}, the contribution of the stored advertisement is subtracted, and the contribution of the new advertisement is added to the potential value. Then, the new advertisement replaces the stored advertisement and is forwarded.

3.2.2. Local Exchange of Potential Values

Neighbors periodically exchange information about their local potential values for the different service types. These broadcast packets have two purposes. First, these packets are used as “hello”-messages to indicate the current neighborhood nodes; thus, if a node fails to receive a packet from a neighbor for a predefined amount of time, the neighbor is assumed to be gone. Second, these packets are used to exchange local potential values of the known service types with the neighbor nodes. A node always knows the potential values of all neighbors as required to forward queries.

3.2.3. Service Queries

A client that searches for a service of a specific type creates a service query message. Such service query messages contain the following fields.

- *Service Type:* The service type that a client is searching for.
- *Message ID:* The message ID serves to associate a reply message from a service with a request sent by a client.
- *Requester Address:* The network address of the client.
- *Forwarding Mode:* This field is used to specify whether the query is forwarded based on potential values or on proximity. In the latter case, the query is forwarded towards the closest service instance. The closest service instance is simply determined by comparing the *Hop Count* values from the recently received service advertisements that every node must store to calculate its own potential value.
- *TTL:* The time-to-live field is a hop count initially set by the client. It is reduced at each hop by one until it reaches zero. In that case, the query is not further forwarded. This field can be used by the client to restrict its discovery range and serves also to prevent queries to be caught in a loop (short-lived loops can only occur during the protocol update phase of potential values).

3.2.4. Query Replies

Upon reception of a service query, a service instance must reply to the client with a query reply. This query reply contains the actual network address of the service and a description field which is used to give additional information about the service to the client.

- *Service Type:* The service type of the service which replied to the query.
- *Message ID:* The ID from the query reply message is the same as the corresponding query message.
- *Service Address:* The network address of the service instance.
- *Description:* Additional information about the service. For example, the port number at which the service process is listening can be specified here.

A query reply is routed back to the client over the same path as the service query.

3.3. Handling Node Mobility and Failures

Nodes determine connectivity by listening for the periodic potential value update broadcast packets from their neighbors. If a node has not received an update packet from a neighbor for some timeout value, it assumes that the link to the neighbor is lost and removes this neighbor from its table. In addition, nodes detect if neighbors moved away or disappeared when sending unicast packets. With IEEE 802.11 [3], a node that moved away can be detected with an appropriate link layer notification (in the absence of a link layer ACK or failure to get a CTS after sending RTS). For example, when a node forwards a service query, it tries to forward the query to the neighbor with the highest potential. However, if this neighbor has disappeared, a notification from the link layer is triggered. In such cases, the node removes the neighbor with the highest potential from its neighbor list and retransmits the query to the neighbor with the next highest potential value.

Due to mobility, it is possible that the network becomes partitioned. In this case, nodes gradually delete advertisements which timeout over time. If two network partitions merge together, services from one partition will become visible to the other partition as soon as they re-advertise their service type.

3.4. Reducing Flooding of Advertisements

In our implementation, service providers must broadcast advertisements periodically because this information is stored in soft state. Since these advertisements are flooded, one can argue that scalability is an issue. We therefore describe a method specific to our approach to significantly reduce overhead traffic. Other methods to reduce flooding overhead, such as selective flooding (e.g. Multipoint relaying [4]), could also be considered to further improve the performance. However, we consider this as an orthogonal research issue and do not further put additional efforts in this direction to improve the performance.

The technique to reduce flooding of advertisements we propose consists of caching and aggregating advertisements before relaying them. The first time a node receives an advertisement with a *service type* it has not seen before, it adds an entry to the service table and directly forwards the advertisements to its neighbors. However, when a node receives an advertisement with a *service type* it already knows, it is not mandatory to directly forward the advertisement since a potential is already defined on the network for this service type. Hence, the node may cache the advertisement for a while. During that time, the node collects additional advertisements from other services and then forwards the collected advertisements together in one single message. With this technique, the total number of advertisement messages can significantly be reduced. Nonetheless, the discovery performance is not degraded too much (see Section 4.5) since advertisements are only delayed for existing service types. Thus, when a client sends a request during the time an advertisement is cached at an intermediate node, the query still reaches a service.

4. Evaluation

In this section, we evaluate the performance of our implementation with a network simulator. Three main aspects are evaluated. We look at the performance and convergence with respect to mobility, the behavior of discovery when varying the CoS values at different service instances, and the control traffic overhead caused by the discovery protocol.

4.1. Simulation Model

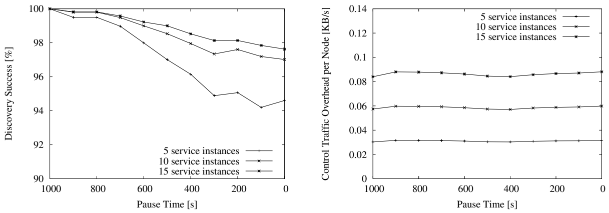
We use GloMoSim [5] as network simulator. At the MAC layer, we use the distributed coordination function (DCF) of the IEEE 802.11 [3] standard . The access scheme is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). The 802.11 DCF uses Request-to-send (RTS) and Clear-to-send (CTS) control packets for unicast data transmission to neighboring nodes. We use a free space propagation model with a threshold cutoff for the experiments. The radio propagation range for each node is set to 250 meters and the channel capacity has a nominal bit-rate of 2 Mb/s. We use the random waypoint model [6] as the mobility model: At the beginning of each simulation, nodes are placed randomly in a rectangular area. Nodes start moving with a randomly chosen speed between 0-20 m/s to a random destination. Once the destination is reached, the node waits for a pause time before another random destination is chosen. Note that with the random topologies used in the simulations, we never observed local maxima in the potential distribution. Thus, all query packets are always forwarded based on the steepest ascent of the potential and not based on proximity.

4.2. Simulation Parameters

The simulation duration for all experiments is set to 1000 seconds. At least twenty runs are performed for each point in the graph and the results of all runs are averaged together to produce the resulting graphs. The network size is limited to 100 nodes on a rectangular (1500m x 1300m) topology. According to our experience, we set the protocol parameters as follows: Service advertisements are broadcast every 5 seconds and have a lifetime of 21 seconds (somewhat more than four times the broadcast interval). If the flooding reduction technique is used, an advertisement is cached between 0 seconds and 5 seconds (depending on the arrival time) before forwarding. Neighbors periodically exchange their potential values every 5 seconds with broadcast packets.

4.3. Effects of Node Motion

Two key performance metrics are evaluated to assess the effects of node mobility and to show that the algorithm converges when nodes are moving. The *discovery success* is the ratio of service query packets that arrive at any service instance to the total number of query packets sent by all clients. The control traffic *overhead* is measured as the average sending rate of control traffic per node. Control overhead traffic encompasses all service advertisements and the periodic messages to exchange potential values between neighbors. With this definition, the control overhead traffic is purely pro-active and therefore, independent of client requests. We do not account the service query and reply messages in the control overhead which depends on the search activity of clients. These messages are not critical to the scalability of the system since they are unicast and not flooded.



(a) Discovery success

(b) Overhead

Figure 7. Effects of node mobility

The discovery success is a very important metric as it determines if a client discovers a service or not. The discovery success is plotted in Figure 7(a) with changing pause time. For this experiment we placed 5, 10, and 15 service instances of the same type on different nodes. Ten clients are constantly sending service request packets at a rate of four packets per second (This rate is much higher than we might expect in practice. We stress the network on purpose to capture the discovery performance at various moments when nodes are moving with high speed.). We conclude that for higher pause times (low mobility), the discovery success is almost perfect ($> 99\%$). For lower pause times (high mobility), the performance remains quite stable above around 95%. Note that the discovery success improves when the number of services instances increases because clients and services tend to get closer on average. We conclude that the algorithm converges even for high node mobility ($v_{max} = 20m/s$).

The control overhead traffic rate per node is plotted in Figure 7(b). For this experiment, we used the flooding reduction technique as proposed in Section 3.4. We will show the control overhead without this technique later (Section 4.5). The overhead is almost independent of the node mobility because control traffic is pro-active. However, the control overhead traffic rate depends linearly on the num-

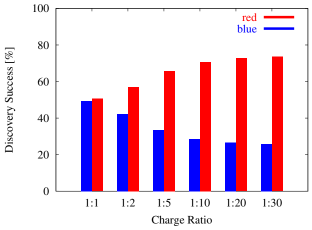


Figure 8. Effects of different CoS values.

ber of service instances.

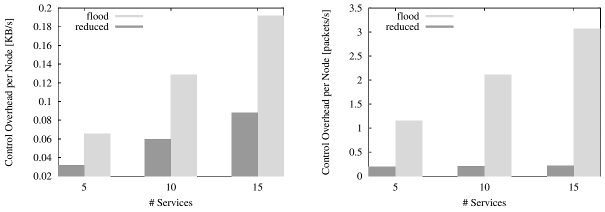
4.4. CoS-Distance Tradeoff

One of the interesting aspects of using field theory for doing service discovery, is the implicit tradeoff between distance and CoS for service selection. A close service instance is discovered by a client unless a better (higher CoS) service is available. To explore this behavior, we analyze our approach with different CoS values assigned to the service instances. The experiment was conducted without node mobility, using a static network where the nodes are placed randomly in the simulation area. We then divide the set of services in a simulation in two classes, the *red* services and *blue* services. Both, the red and blue services are of the same service type but have different charges (CoS). The charge value at each service is constant over the whole simulation.

The discovery success is plotted in Figure 8 for different charge ratios between the red and blue services. A ratio of $1:1$ means that the red and blue services have the same charge, whereas a ratio of $1:5$ means that the charge of a red service is 5 times higher than the charge of a blue service. When the charge ratio is $1:1$, we see as expected that the service discovery queries are evenly distributed to both classes. For a charge ratio of $1:2$, 57% of client queries during the simulation arrive at red services and 43% of the queries at blue services. When further increasing the charge ratio to $1:30$, the red services are discovered 74% of the time compared to only 26% for the blue services. Further intensifying the charge ratio does not much impact the distribution any more and we therefore conclude the following. When the charge ratio is $1:1$, clients discover services which are very close independent of their color. As the charge ratio increases, discovery packets start to drift in the direction of red services as the potential gradient gets steeper in that direction. Thus, a discovery packet can be forwarded to a red service even if a blue service is closer. Note that the charge ratio does not influence query packets from clients which are 1 hop away from a service instance because the potential value at a service is very large (infinite) and therefore, this service instance is always chosen as a next hop.

# services	flood	reduced
5	94.68%	94.61%
10	97.01%	96.91%
15	97.63%	97.61%

Table 1. Discovery success with (reduced) and without (flood) reduction technique.



(a) Average sending rate of control packets

(b) Average number of control packets sent

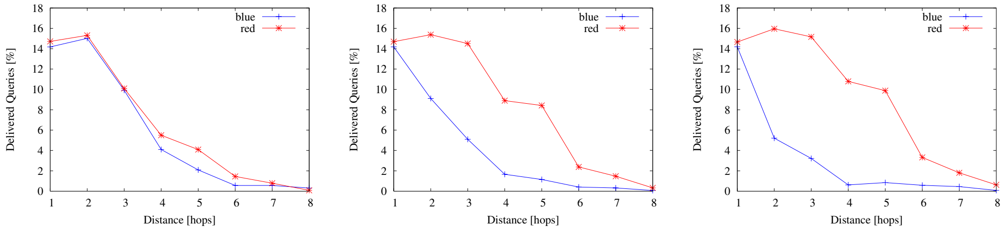
Figure 10. Control overhead per node.

The distance distribution between clients and discovered services is plotted for three different charge ratios in Figure 9. When the charge ratio is $1:1$ (Figure 9(a)), the distribution for the blue and red services is quasi identical (the red and blue curves are expected to converge when using an infinite number of runs). It is interesting to see the distribution behavior when increasing the charge ratio. In Figure 9(b), the distance distribution is plotted for a charge ratio of $1:5$. Blue services tend to only get discovered by close clients. This effect is even more pronounced when the charge ratio is increased to $1:30$ (see Figure 9(c)). We conclude that CoS is an effective mean to differentiate service instances.

4.5. Effects of Overhead Reduction Technique

We next investigate how much control overhead is saved when using the proposed flooding reduction technique (see Section 3.4). To determine how effective our optimization is, we first compare the discovery success with and without optimization for 5, 10, and 15 service instances and very high node mobility (pause time = 0 seconds, $v_{max} = 20m/s$). Then, we compare how much control overhead is caused with both approaches.

Table 1 shows the discovery success for different service numbers. In the different scenarios, we observe only a very small, acceptable performance loss of $\leq 0.1\%$. We now look at the effective control overhead gain with our reduction technique. In Figure 10(a), we see that up to 54% of the total traffic load per node was reduced. In addition to the average sending rate of control traffic per node, we also measured the average number of control packets sent



(a) Charge ratio 1:1

(b) Charge ratio 1:5

(c) Charge ratio 1:30

Figure 9. The distribution of the distance between client and discovered service

per second at each node. The results are shown in Figure 10(b). The number of control packets that were sent are reduced approximately by a factor of 5 (for 5 services) to a factor of 14 (for 15 services). We further observe that, using the flooding reduction technique, the number of control packets does not increase with the number of services. This is because advertisement packets from different service instances can be combined into a single advertisement packet.

5. Related Work

In this paper, we proposed the novel idea of using field theory to do service discovery in wireless ad hoc networks. The idea of using potentials for routing in the Internet has been proposed in [7]. Our approach mainly differentiates from this paper in the way potentials are used. In their approach, a unique potential is associated for all possible destinations in the network. However, we assign a potential to a service type. Therefore, our potential function might have more than one maximum/minimum. Furthermore, we apply the idea of potentials to dynamic networks such as wireless ad hoc networks whereas the authors of [7] mostly target static environments where the topology rarely changes.

Sun's Jini [2], Microsoft's UPnP [8] or IETF Service Location Protocol [9] have been proposed as service discovery standards. These systems were not designed for wireless mobile ad hoc networks and are therefore, not suited for dynamic and infrastructure-less networks.

Our approach is comparable to the Intentional Naming System (INS) as proposed in [10]. In that approach, client requests for a service are directly routed towards a matching service instance without an intermediate lookup to discover its address. To route these requests, a resolver network of dedicated INS resolvers is required which might not be available in an ad hoc network. However, it is imaginable to extend the INS approach and for example, delegate the task of INS resolver to each node or at least a subset of the participating nodes. The main difference between our approach and INS is how service instances with identical service type are discovered. We use a tradeoff between network proximity from a client and service capacity. This issue is, by design, not addressed in INS.

Kozat and Tassiulas [11] proposed a service discovery mechanism targeted at mobile ad hoc networks. A virtual backbone is constructed dynamically, assuring that all nodes are part of this backbone or at least one hop away. Here again, the service discovery system does not provide mechanisms for service selection when multiple service instances of the same type coexist. Konark [12] is a middleware designed to support service discovery and delivery in ad hoc networks. Services are expressed using XML. The service delivery itself is based on SOAP. Unlike our approach, Konark requires a multicast protocol for the actual discovery process.

Our approach can be viewed as a form of a publish / subscribe system. In a publish / subscribe system (e.g. TIB / RENDEZVOUS [13]), processes can subscribe to messages containing information on specific subjects, while other processes produce (i.e. publish) such messages. In our approach, the clients would publish requests while the service providers subscribe to those. The publish/subscribe systems so far have been researched and developed mostly in fixed networks. Our approach takes full advantage of the broadcast nature of wireless radio where traditional publish/subscribe systems are often built as overlays over IP.

Similar concepts have been proposed for sensor networks. For example, Estrin et al. proposed Directed Diffusion [14]. Data packets follow application-specific gradients to reach their destinations. These approaches are targeted at applications for collecting sensor data and is not very well suited for service discovery.

An alternative methodology to do service discovery has been proposed by Koodli and Perkins in [15]. The basic idea is to add service information in route request messages from on-demand ad hoc routing protocols such as AODV [16]. A drawback from this approach is that each client request generates a message which is flooded in the network.

6. Conclusions

This paper defines a novel approach towards efficient and robust service discovery in mobile ad hoc networks. As such, electric field based service discovery uses a simple mechanism to find the best route to the closest service instance: at each node, the request is routed towards the steepest gradient until it reaches the service instance. We have shown that the algorithm is stable, even when conditions are highly dynamic. In addition, we examined modifications of the algorithm to reduce control overhead without degradation of performance. The major advantage of this approach however, is its simplicity and clarity in design. We believe that our method to perform service discovery can easily be adapted to be used for additional tasks. The first application that comes in mind is packet routing in MANETs. Instead of forwarding requests to service instances only, the same mechanism can be used to establish communication between two devices as long as they are uniquely identified in the network. Another valuable property of this approach is its independence of the underlying network protocol. Indeed, it is not only independent, it works even in the absence of any underlying routing protocol.

In a next step, we will examine the impact of different distance functions ($dist()$ from Equation 1). The reach of each individual service instance gets smaller when using steeper distance functions. For the future, we also plan to enrich the design by assigning negative charges to clients. Thus, the position of clients will also influence the distribution of the service fields. This can be used to perform load balancing in the system. I.e., since clients are using negative charges, the potential of a service instance is reduced by neighboring clients. As a result, requests from other clients are more likely to be forwarded towards other service instances with higher CoS. This extension is of specific interest with regard to the parking system application or the Internet gateway example.

Acknowledgments

Polly Huang was involved in initial discussions of this project. We wish to thank Christian Bruns and Ulrich Fiedler for their valuable feedback on field theoretic aspects of the paper. We also would like to thank Matthias Bossardt for his comments on the paper and all anonymous reviewers.

References

[1] S. Deering and R. Hinden. IP Version 6 Addressing Architecture. IETF RFC 2373, July 1998.

[2] Sun Microsystems. Jini Architecture Specification. version 2.0, June 2003.

[3] IEEE Std 802.11-1997. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. Number ISBN 1-55937-935-9. 1997.

[4] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS'2001).*, 2001.

[5] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations (PADS '98)*, Banff, Alberta, Canada, May 1998.

[6] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imelinsky and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[7] A. Basu, A. Lin, and S. Ramanathan. Routing Using Potentials: A Dynamic Traffic-Aware Routing Algorithm. *SIGCOMM'03*, Karlsruhe, Germany, August 2003.

[8] Microsoft. The Universal Plug and Play (UPnP) Forum. http://www.upnp.org, 2003.

[9] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service Location Protocol. RFC 2165 (http://www.ietf.org/rfc/rfc2165.txt), June 1997.

[10] W. Adjie-Winoto, E. Schwartz, and J. Lilley. The Design and Implementation of an Intentional Naming System. In *Proceedings of the 17th Symposium on Operating Systems Principles (SOSP '99)*, pages 186–201, Charleston, SC, USA, 1999.

[11] U. C. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. *INFOCOM 03*, San Francisco, USA, April 2003.

[12] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - A Service Discovery and Delivery Protocol for Ad-Hoc Networks. In *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC)*, New Orleans, USA, March 2003.

[13] TIBCO Software Inc. TIB/Rendezvous Concepts. Technical Report Release 6.4, Palo Alto, CA, October 2000.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *MOBICOM '00)*, Boston, USA, 2000.

[15] R. Koodli and C. Perkins. Service Discovery in On-Demand Ad Hoc Networks. IETF Internet Draft draft-koodli-manet-servicediscovery-00.txt, October 2002.

[16] C. Perkins, E. Belding-Royer, and S. Das. Ad Hoc On-Demand Distance Vector (AODV) Routing. IETF RFC 3561, July 2003.