Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks

Christian Frank
chfrank@inf.ethz.ch
Distributed Systems Group
ETH Zurich, Switzerland

Holger Karl
hkarl@ieee.org
Telecommunication Networks Group
Technische Universität Berlin, Germany

ABSTRACT

Emerging "urban" ad hoc networks resulting from a large number of individual WLAN users challenge the way users could explore and interact with their physical surroundings. Robust and efficient service discovery and routing protocols in such networks are a necessary ingredient. Although a lightweight service discovery proposal integrated with ad hoc routing exists, an implementation and performance evaluation with respect to overhead and correctness have so far been missing.

Moreover, the different service providers in an urban scenario, which (more or less) frequently and actively change their status, demand more flexible handling of cached information on neighboring providers than what is currently proposed. We therefore contribute mechanisms that maintain cache consistency and show that explicitly removing cache entries on existing neighboring providers is well invested effort. We finally evaluate whether ad hoc network latency can implicitly help our protocol in retrieving the physically closest provider, using an 802.11 model. Additionally, we provide a general architectural framework for enabling lightweight service discovery on top of most reactive routing protocols.

Categories and Subject Descriptors: C.2.1 [Computer Communication Networks]: Network Architecture and Design; I.6.6 [Simulation and Modeling]: Simulation Output Analysis

General Terms: Algorithms, Design, Performance

Keywords: Service Discovery, Ad Hoc Networks, Ubiquitous Computing

1. INTRODUCTION

The popularity of mobile computing and WLAN interfaces enables a new paradigm of communication: Mobile wireless network participants communicate amongst each other without the use of any infrastructure, forming mobile ad hoc networks (*manets*). With the growing density of WLAN devices large networks could emerge, which cover a significant part of a downtown city area. Such a network may be used to retrieve information on the surrounding physical world. Such information often takes on the form of a "Where" question: Where can I find something, where is a specific service provided? Consider a pedestrian user, standing at a downtown street crossing, wishing to call a taxi cab. Or a user in a car who wants to locate a subsidiary of his bank in an unknown neighborhood. In such scenarios, retrieving the *closest* such entity is most valuable to the client.

We contend that such functionality, which is currently realized in a centralized fashion, could easily and much less costly be addressed by a service discovery protocol within the city *manet*. The sought entities (e.g., taxi cabs, stores, gas stations or also Internet access points) publish their physical service as a network service, which may then be discovered by mobile clients. We claim that because of the inherent correlation between ad hoc network topology and physical proximity, approximating the physically closest provider without employing any localization system is feasible. The protocol would naturally integrate locality, mobility, and scalability and avoid single points of failure.

The required functionality of a service discovery component is simple: The client application issues a request to the node's lower layers. In the case that a provider is found, a reply is issued containing the provider's network address. Additionally, the network shall provide end to end routing for communication between client and provider once the provider has been discovered: In the taxi example, the client might wish to "hail" a taxi. In the case that no provider is available or is unreachable within a given scope from the client, no reply will be issued by the network. In this case the client will have to wait for a timeout. The functionality for provider applications is also straightforward: These shall be able to register and deregister their services at the service discovery layer.

The detailed mechanisms of such a system are unclear, however. We have to consider, e.g., where information about service providers is available: If available only at the providers themselves, a high overhead in searching for this information is incurred. If information is cached at other points in the network, this raises questions regarding cache lifetime and mechanisms for (more or less aggressively) maintaining cache consistency. In addition, the volatility of service provider availability has to be taken into account when designing such a system (this aspect is especially valid in the inspected usage scenarios, as provider availability could change both frequently and intentionally as in the taxi example). We study these aspects using a detailed simulation model for the urban ad hoc environment including a number of relevant participant types.

The following Section 2 overviews existing approaches for their suitability and selects one basic mechanism integrating service discovery and ad hoc routing as a starting point. Section 3 will then describe the chosen service discovery mechanism and introduce a number of enhancements to it that enable the protocol to be efficient in high load scenarios with frequent changes in provider availability. Based on our implementation, we also include a general con-

MSWiM'04, October 4-6, 2004, Venezia, Italy.
Copyright 2004 ACM 1-58113-953-5/04/0010 ...$5.00.

cept of possible integration of ad hoc routing and service discovery. We will then present detailed models used for the urban ad hoc environment in Section 4 and provide simulation results of the basic protocol and our extensions in Section 5.

2. RELATED WORK

2.1 Service discovery

Service discovery protocols provide features to spontaneously look up services with a low overhead for network configuration. Prominent among those are the *Service Location Protocol* [9], *Salutation* [20], *Jini* [6], *UPnP* [23] and the service directory service *UDDI* [22]. These systems were mostly designed for an administrated network context and are either based on aggregating information into central directories (like UDDI or Jini) or on maintaining a network-wide multicast tree used for periodic service advertisements (like UPnP, Salutation or SLP, when operated without directory agents). We believe that both approaches are not designed or suitable for the large and mobile ad hoc networks envisioned in this paper.

Among other approaches specifically targeted at ad hoc networks, reference [3] also use multicast trees, adding extensions to the On-Demand Multicast Routing Protocol (ODMRP) [25] to provide service discovery. Services/resources are advertised to a known multicast address which is joined by nodes interested in advertisements. We assume that multicast group management constitutes too much overhead for the localized interactions required in our scenario of a large and inherently mobile network.

A service discovery approach in which *proactive* routing and announcements are performed within a small neighborhood area is presented in [12]. Discovery outside the neighborhood is initiated reactively through remote *contact* nodes which then reply the query from their own neighborhood or forward it to their contacts. This work applies well to the described usage scenarios, yet we would like to avoid the routing overhead within the neighborhood areas in the case that network traffic is very infrequent. Also, it is unclear how a hop distance could be obtained for discovered resources, which would enable us to exploit network proximity for retrieving the closest provider.

In an Internet Draft [10], Koodli and Perkins describe ongoing work to integrate service discovery and a reactive routing protocol. This approach appears promising as it is a lightweight extension on top of a proven ad hoc routing protocol and may benefit from existing routing information.

2.2 Anycasting

Service discovery in ad hoc networks using *Anycasting* [14] is a promising approach to integrate routing and discovery: Providers of a given service type also advertise routes to an additional *virtual node* representing the service type. Routing will then forward routing queries for the *virtual node* correctly to *one* of the service providers. Anycast extensions are described for various routing types. Among those, for the described large mobile network scenario, only the reactive AODV [17] and potentially *link reversal* may apply. However, AODV uses sequence numbers for each node to avoid loops, and it is unclear how *virtual nodes* could issue coherent sequence numbers [14].

Link-reversal routing algorithms [8] maintain a valid route to a destination proactively. This is done by local repairs that always maintain a forwarding link towards the destination. We considered a service discovery protocol such that nodes maintain a valid route to a given service type at all times (at least in a given scope around the provider or in areas with high network load). Whenever the last link to a provider of any service type fails, nodes will take proactive measures to ensure at least one outbound link (and therefore at least one route) is present. By treating service types on the same level as node addresses, one could use a variant of TORA [15] to do service discovery as proposed in [13]. Although interesting, we believe that the proactive link reversal protocols are not quite suitable for the requested functionality: While nodes will maintain a route to a matching provider whenever possible, nodes do not incorporate new information: If a new provider becomes available and a route to another (possibly quite remote) provider already exists, the node will never learn of the new provider, as no state change of the node is required.

2.3 Internet connectivity for mobile ad hoc networks

Support of Mobile IP [19] in ad hoc networks also includes discovery of *foreign agents* when a mobile host has reached a new network. Prominent work in this area is done by Sun et al. [21]. The focus is on integrating AODV [17] and Mobile IP protocols. Proactive foreign agent advertisements are flooded periodically by intermediate nodes. The analysis focusses on varying mobility and announcement intervals with one or two foreign agents. Ratanchandani and Kravets [18] describe a hybrid approach between reactive discovery and periodical advertisements of foreign agents. The reactive discovery is integrated into AODV route discovery. Agent advertisements are limited in scope and also establish routes. Our work is different as it provides support for a variety of provider types among which some may frequently change their availability.

Summing up, we believe that the most promising mechanisms are based on the integration of service discovery functionality into ad hoc routing protocols as exemplified by reference [10], especially because both routing and discovery must be available in such a network anyway. However, it appears reasonable to extend it further with functionality that reflects the dynamic nature of *service availability*, which is independent of whether a service provider is *reachable* from a pure routing point of view – the main problem is that cached information on service availability might be outdated or staled inside the network while routing information is still accurate. Hence, we choose reference [10] as our starting point and extend it with proactive extensions that reflect these service dynamics and are efficient in both high-load and inactive phases of network operation. The following section will describe these mechanisms in more detail.

3. PROTOCOL DETAILS

Section 3.1 describes our current specification and implementation of reference [10]. Some clarifications and additional specifications were needed (even though they might have introduced some slight differences to [10]). Additional specifications for our purpose can be found in subsections 3.1.3 and 3.1.4. Section 3.2 will motivate additional proactive enhancements to the protocol in order to find a balance between areas of lower and higher request rates; it will analyze different proactive elements that may be employed and will summarize the elements chosen for simulations.

3.1 Service discovery in on-demand ad hoc networks

The main idea of [10] is to attach a service discovery header to control packets used by routing.

3.1.1 Service request and reply packets

In [10] the authors assume that the routing component follows a request/reply cycle: A route request packet containing the sought destination is broadcast and is replied by a node knowing a route to the destination. This sequence is part of most reactive routing protocols.

A service request packet (SREQ) is shown in Figure 1. It comprises a route request packet (RREQ) plus an additional header describing the sought service, the so-called *Service Request Extension* or SRE. The SRE contains information to identify a given service type (*service selector*). In referece [10] this information may be either a URL or a port number; in our context it is modeled by an integer referring to a predefined service type. If the search should be enhanced with semantic attributes (which is not implemented), these attributes would also be located in the SRE. Similarly, a service reply packet (SREP) will be a route reply packet including additional information on the service found, accordingly termed a *Service Reply Extension* or SRpE. It includes the service lifetime, which may be different for various providers.

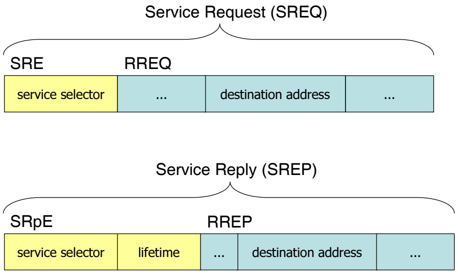


Figure 1: An additional header is attached in front of routing control messages

The depicted *destination address* field in the RREQ and RREP messages refers to the sought destination for which route discovery is performed. Service discovery will use it to store the provider address, see below.

3.1.2 Service and route resolution

At the time a client issues a request, it creates a route request packet and adds an SRE with the sought service type. At this time the destination address of the RREQ packet is unknown and will be set to zero (other fields may need to be set compatibly, e.g. if using AODV routing the destination sequence number must be set to *unknown*). The destination address of the RREQ will later contain a matching provider address.

The SREQ will propagate through the network like an ordinary RREQ. A node which receives an SREQ will fill-in the missing destination address if it knows a matching provider or if it is a provider itself. Each node will store so-called *service bindings*, a binding associating a service type to addresses of provider nodes. The provider node will update its bindings locally. Other nodes will cache service information from SREPs they forward. Note that multiple providers may be registered for a given *service type*. The table additionally stores a cache lifetime up to which the service information is valid.

The handling of an incoming SREQ is shown in Figure 2. The node at first checks whether the packet contains a destination (the destination field is non-zero). If the packet *does not* contain a destination, there are two possibilities:

1. If the node does not have a service binding for the given service type it will rebroadcast the SREQ unchanged.

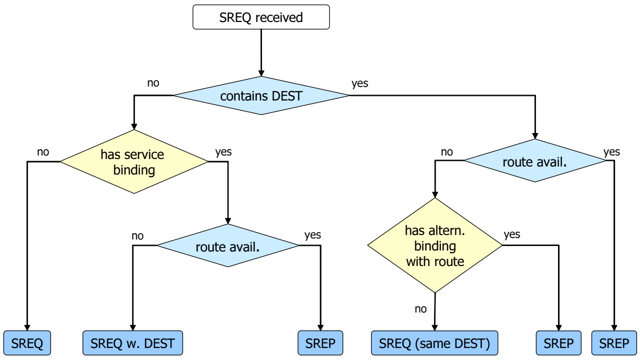


Figure 2: Handling an incoming SREQ

2. If the node has a service binding including a route, it will send an SREP packet back towards the client. If it has a service binding but no route, it will just fill-in the destination address of the binding with the highest lifetime and rebroadcast an SREQ with an updated destination field.

If the packet *does* contain a destination there are also two cases:

1. If the node knows a route to the destination, it will send an SREP packet including the service type and lifetime in its service binding table.

2. If the node does not know a route to the destination, it will look up whether it has alternative service bindings (including a route to the provider address) for the given service type. If so, it sends an SREP including service time and the lifetime stored in its table, if not it just rebroadcasts the SREQ.

An SREP will be unicast back along the spanning tree established by the SREQ flood. The intermediate node will add or update its own service binding if the information in the message is more recent than its own. In the opposite case, the lifetime of the message is updated to the node's own service binding entry. The RREP part of the message will always be processed by the corresponding routing component and thus routes will be constructed via the message as well. If the SREP is directed at the local node, a service reply is indicated to the requesting application and the message is not forwarded.

3.1.3 Choice of service bindings

The cited Internet Draft [10] does not make any statements on the choice of service bindings when multiple bindings are available. In our work the *closest* provider – measured in hops – is chosen according to the current node's route table. If service bindings are cached at intermediate nodes, this is only an approximate choice which will be evaluated in the results section.

This decision does weigh (potential) closeness of a provider heavier than the freshness of provider information. As a consequence of cache inconsistencies, some requests may be replied incorrectly. This could be avoided when using the most recent service binding (with the largest lifetime) instead, but this would heavily compromise the goal of hop-wise optimality of the replies. The performance evaluation section will show the consequences of this choice.

3.1.4 Post-discovery connectivity

A route from the client to the provider is always implicitly constructed by the SREQ/SREP pair. Because it is assumed that subsequent communication between provider and client will be necessary, the route back *from* provider *to* client is also important. This route is also implicitly constructed if the SREQ message reaches the provider itself (as the reply travels this route as part of the constructed spanning tree). Yet this route is *not* available if the reply is issued by an intermediate node.

Having used AODV as the routing component, we achieve this end-to-end connectivity by setting the "gratuitous" flag in the route request (RREQ) part of the message. The flag causes a gratuitous route reply (GRREP) to be generated at the intermediate node and be unicast towards the provider as shown in Figure 3. The fields of a GRREP are set as if the provider node had requested a route to the client and the GRREP were the reply. This functionality is part of the AODV protocol ([17], Sec. 6.7).

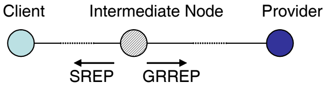


Figure 3: Intermediate node constructing routes in both directions

If the client node should be able to reply the service request locally, i.e. the client node holds a service binding and a route and does not send out an SREQ message at all, service discovery will explicitly initiate a GRREP at the AODV module.

3.2 Proactive enhancements

The route and service search by [10] described in the previous Section 3.1 and its integration with AODV routing contribute the valuable concept of simultaneously discovering routes and services. It is not clear, however, how these mechanisms behave especially under heavy load on the network and under frequently changing provider availability.

We evaluated possible proactive enhancements that are able to decouple protocol effort from the number of requests in areas of high traffic. Naturally, the proposed caching of service bindings is able to improve performance in high load scenarios. It is unclear, however, what a suitable caching lifetime would be in such scenarios. Although caching may reduce protocol effort, we expect that it yields a significant amount of incorrect replies, especially when providers become unavailable and in presence of many requests. Also, caching may compromise our goal of hop-wise optimality, when a choice among several providers is made. These aspects will be studied in the performance evaluation section.

Generally, proactivity is activated by a provider becoming either newly available or unavailable. In both cases, the time during which the network stores inconsistent information should be as short as possible. In the reactive case (with caching), a node learns about a new provider only after it has overheard or processed a request/reply pair to this node. This happens soon at high request frequencies. Periodic announcements, on the other hand, limit the time until the new provider is known to the announcement interval. To match the reactive protocol's performance in high-load cases, frequent announcements would be necessary, which are just wasted overhead in other cases. Additionally, the period of inconsistency for providers that have become unavailable remains. Hence, the benefits of periodic announcements would depend on a careful tuning of announcement intervals; therefore, periodic announcements are questionable and will not be considered further.

Yet, what other enhancements can be made at high load, which do match our scenario and also maintain correct protocol behaviour?

3.2.1 *Negative announcements*

When a provider becomes unavailable, every cached entry on this provider may cause incorrect replies. Here, the disadvantages of caching are clear, both for reactive and for periodically announcing systems. To limit incorrect answers, short cache lifetimes would be necessary, nullifying its effectiveness. Therefore, explicitly removing cached entries by negative announcements would allow to maintain caching benefits without, hopefully, imposing too large an administrative burden. Such negative announcements are easy to implement for providers *actively* switching state. This active transition to unavailable status triggered by the service application is a crucial aspect, in which our scenario is different from scenarios used in most related work.

In the case that the provider becomes inadvertently disconnected through route changes, the case is different: The first request will result in a route error, yet the provider address is still valid and useful information: If no other service provider is known at this node, a new route to the provider will have to be discovered by the routing component. The route error becomes relevant only when choosing among several providers: Other providers with available routes will be preferred choices.

Once inconsistencies are dealt with, the possibility to maintain longer service lifetimes is reintroduced, as the entry will be erased when the provider actively changes state and routes will be kept up to date by the routing protocol.

Similarly, positive announcements upon provider state changes may also be beneficial, but preliminary evaluations showed that they increase inconsistency for providers which frequently change state: The effort required to delete information spread via positive announcements plus the additional effort to send the announcement in the first place is larger than their benefit. Due to space constraints, we largely omit their discussion (for details see [7]).

In summary, the remaining mechanisms to consider are reactive with or without caching and caching with negative announcements. Service announcements are implemented as follows.

3.2.2 *Service announcement format*

A service announcement message (SANM) is shown in Figure 4. Its header consists of the service discovery header, which is also used for service replies, and a flag indicating whether it is a positive or a negative announcement. The message tail is a regular route reply with the provider as the destination and may be used to construct routes to the provider: E.g. in our implementation the tail is processed by the AODV routing component. It will mark the predecessor (which forwarded the announcement) as a next hop to the provider. Note that the *originator address* of the RREP is set to broadcast: The RREP is not unicast back to any requesting (originator) node, but is simply broadcast to all neighbors up to a specified number of hops. As this is not required AODV behavior, some functionality needs to be added; Section 3.3 describes the required functionality at the routing component and its implementation for the existing AODV protocol.

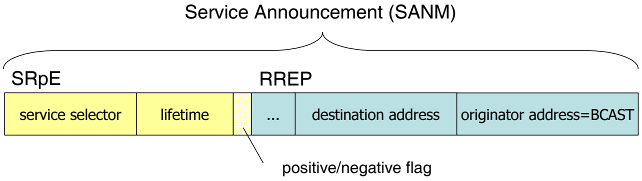


Figure 4: Service announcement (SANM)

3.2.3 Limiting Announcement Forwarding

Service announcements will be forwarded only within a given parameterized scope. Service bindings will be deleted while forwarding negative announcements. Additionally, several heuristics are used in order to limit the protocol effort imposed by announcement floods; essentially, a node does not forward the announcement if it does not affect the node's own provider choice:

1. A node forwards negative announcements *only* in the case that the binding which shall be deleted by the announcement is present in its own service binding table.

2. A node does not forward negative announcements in the case that another – hop-wise closer – provider is already present in its service binding table. This is because the old information to be deleted by the negative announcement would not be part of replies issued by this node, anyway.

In summary, if the provider has changed to unavailable state it will send out a negative announcement. Upon reception of a negative announcement nodes delete their corresponding service binding. Forwarding is limited by a parameterized scope and the above heuristics. Preliminary simulations have shown that the above heuristics significantly reduced the number of messages while maintaining consistency and correctness levels.

3.3 Interdependency of Routing and Discovery

The previous section has described the possibilities for a service discovery layer built on top of an ad hoc routing protocol, especially AODV. In principle, however, the service discovery functionality should be independent of the concrete routing protocol, even though it clearly has to rely upon it. To enable a flexible combination of service discovery and routing protocol, we describe here a separation of routing and service discovery functionality that enables different ad hoc routing protocols to be used as a plug-in, as long as they satisfy a given interface.

Service discovery functionality will be realized in the network layer, the service discovery component is placed, perhaps unconventionally, adjacent to and *below* the routing component as seen in Figure 5.

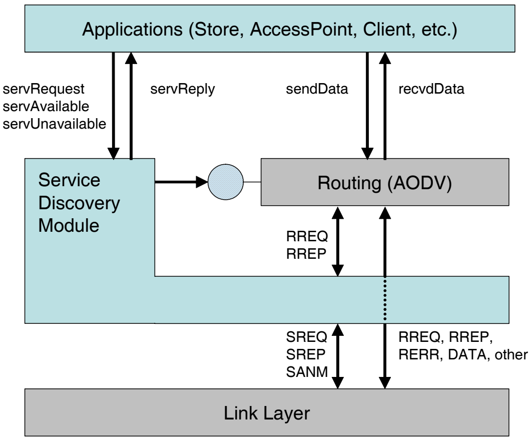


Figure 5: Network layer service discovery

The top part of the figure summarizes the application primitives: An application may request and (un)register services (at the service discovery component) as well as send and receive data messages (at the routing component).

Being set below routing, the service discovery layer has the possibility to intercept all routing messages. This constitutes its independence of routing: It is able to attach the described service discovery headers while passing messages downward and detach and process the headers from routing messages passed upward. Note that SREQs, SREPs and SANMs, are only service discovery headers in front of RREQ and RREP messages, respectively (SANMs also enclose a RREP, as seen in Section 3.2.2). Service discovery at first removes the header, passes the regular routing message upward and reattaches the header on its way out.

This structure makes it possible to achieve the required functionality (both the reactive part derived from [10] and the announcements) while keeping as much independency as possible from the routing component.

Nevertheless, the protocol requires additional access to information of the routing component, especially information in the routing tables. One could retrieve this information from the intercepted routing messages, but this is tedious and requires code that is heavily customized to a given routing protocol. Hence, an employed routing protocol is required to support the following interface:

createRREQforSd Shall create a RREQ message according to the message format used by routing and initialize all message fields such that they are compatible to other RREQs issued (e.g. in AODV the route request id must be unique for this node). The destination field shall have a symbolic address that indicates an unknown target address. The routing protocol must support such a symbolic address for which it does not hold any routes.

routeAvailable Shall look up a given destination in the route table and return the number of hops or that none is available.

setDest Shall change the destination field of a given RREQ to the new address (the service discovery module itself does not explicitly access routing message fields in order to maintain the exchangeability of routing modules). Additionally, it must perform all other required changes to keep the message fields consistent with the new destination. In the present case of AODV routing the corresponding destination sequence number must be set, which is also held in the AODV routing tables. Other similar simple access methods are also required (e.g., getDest etc.).

belongsTo(rrep, rreq) In order to correctly reattach the service discovery header to a RREP message, the service discovery layer needs to be able to relate RREPs to RREQs that caused them (and have been previously passed upward).

For the proactive enhancements it is beneficial to create routes while (negatively or positively) announcing service providers. This improves performance (and is implemented in the experiments of Section 5), but is not mandatory for the functionality of the protocol. Announcements are service discovery headers attached in front of route reply messages according to Section 3.2.2. For this case, an additional interface method to routing is needed to create a correctly formatted broadcast RREP message:

createRREPforSd Shall create a RREP message with the fields set such that routing components in neighboring nodes will forward the RREP up to a given number of given hops (announcement scope). If using AODV this would encompass setting the *originator address*, which usually holds the request initiating node and the addressee of the RREP, to broadcast and the *ttl* field to the given scope.

Summing up, the entanglement between service discovery and routing is significant. But the areas of interaction between the two can be well defined and it is possible to exchange one routing protocol against another while maintaining the various optimizations of the service discovery component described here (caching, proactive negative announcements, smart forwarding).

4. MODEL

Several system aspects have to be modeled for a performance evaluation. The models of this section were implemented in C++ using the Omnet++ [24] discrete event simulator and its extensions for wireless networks [5].

4.1 Network model

For most of the scenarios, physical and link layers are not modeled in detail, a simple unit-disc graph model is used (mostly to keep simulation times manageable): Broadcast and unicast message delivery is provided. Message losses occur depending on a given bit error rate and message length. Parameters for the transmission range of the network interfaces are derived from off-the-shelf 802.11b Wireless LAN cards [4]. A bitrate of 2 MBit is assumed (the 802.11b standard allows up to 11 MBit within a smaller radius; as messages are fairly short, higher bitrates are not considered necessary and not modelled).

Transmission duration is computed from the bitrate and message length. Hosts transmit with a nominal power of 15 dBm (about 32 mW) and a simplified path loss model is used. Assuming a path loss factor for a semi-open environment yields a maximal transmission range of 158 meters. The semi-open environment roughly describes inner-city or airport hallway/lounge environments.

To study how well the nearest provider can be found, channel conditions of the ad hoc network are very relevant. For these investigations, we used an IEEE 802.11b model with DSSS PHY and the Distributed Coordination Function (DCF). The simulation parameters were chosen to be comparable with [1] but with a higher path loss factor resulting in a 310 m interference range and a 140 m transmission range. The data rate is 2 MBps; the RTS threshold is set such that the RTS procedure is used for data messages (used e.g. in access point sessions or service invocations) but not for the fairly short control messages.

An AODV routing component is used with fairly standard parameters [17].

4.2 Participant model

The urban scenario will be modeled with a variety of participant types: clients and three types of service providers. The availability of each provider behaves differently to client requests. All participant types provide routing and service discovery functionality. For mobile providers a random waypoint mobility model [2] with a few adaptations is used: In this model, node speeds are not uniformly distributed, but drawn from a normal distribution with a mean chosen according to the participant type. To avoid problems with nodes “starving” on paths with very low speeds, a positive minimum speed is adopted as proposed in [26], which skews the distribution towards the right. The actual mobility parameter values differ with each host type. We modelled the following participant types:

`Client` These participants model a pedestrian (or by other means mobile) network user. Instances of this type will follow the random waypoint mobility model. Their main parameters are summarized in Table 1(a). In this model, users do not provide services of their own and are the only participants that request services. Inter-request times are drawn from an exponential distribution.

Table 1: Parameters for different participant types

(a) `client`

Parameter	**Value**
mean time to next request	120 s
mean speed	4 m/s
speed variance	1 $(m/s)^2$
mean pause time	20 s

(b) `cab`

Parameter	**Value**
mean time until avail.	20 s
mean speed	8 m/s
speed variance	3 $(m/s)^2$
mean pause time	3 s

(c) `access point`

Parameter	**Value**
mean session length	120 s
session packet rate	1.2 pkt. / s
mean speed	stationary

`Store` This simplest provider type represents any stationary service provider with a constant service configuration: The typical example is a store, which offers a range of goods and represents them in the network by advertising its products as services to the network. The provider is always available.

`Cab` This type represents mobile service providers with volatile service configurations. Examples of such providers are taxi drivers or bike couriers. Instances of this type will follow the random waypoint mobility model. The service configuration, i.e. is the service available or not at a given time, is dynamic: A taxi cab changes its status of availability each time it is hailed or when it completes a ride. The simulation includes an implementation of taxi cabs: The time for completing a taxi ride will be modeled by a gamma distributed variable [11]. The cab will become unavailable again after it has been found by a client and hailed (service invocation).

An overview of the parameters used for providers of type `cab` is given in Table 1(b). These providers, used with stationary mobility parameters, may also model other examples like free parking spots.

`Access Point` Represents a stationary provider which exhibits a semi-stable service configuration: Initially an access point is available and accepts client sessions. Upon each service invocation a new client session begins. Session durations are drawn from a gamma distribution. The access point becomes unavailable when it has reached a threshold number of clients (full capacity). After having reached full capacity, it will become available again only when at least one client session has ended, or, to avoid oscillations, when the available capacity has fallen below, say, 90% of nominal capacity.

Additionally, the access point communicates with its clients during a session. The effect is that the route between client and provider is needed and will be maintained (data messages cause route requests at the routing protocol) in spite of route changes throughout the session. To model this, the access point sends messages at the *session packet rate* to each of its clients, with appropriately chosen, exponentially distributed inter-message times. We focussed on the route maintenance effect, as it most predominantly influences our metrics. Different packet sizes and heavier data traffic remain to be studied. Table 1(c) shows the main parameters.

The `access point` type, if turning the background traffic feature off, also models other physical services with a given maximum capacity like parking lots, garages or a taxi stand.

Passive Participant To be able to increase the request frequency without affecting node density the model will also include passive participants. These nodes include the routing and service discovery components but do not request or provide services themselves. Mobility parameters are equivalent to client hosts.

5. RESULTS

This section studies the behavior of the different protocol variants. We concentrate in this paper on the reactive protocol without and with caching as well as on the hybrid protocol with negative announcements. Adding positive announcements turned out to be quite undesirable and is not discussed here. The fact that routes are constructed with positive announcements does not make up for increased message traffic and increased inconsistency levels.

As metrics, we used the messages sent per request, counting overheads (as routes are discovered along with discovery requests, messages for route resolution to providers are also counted). Later we will evaluate the distance optimality metric: The distance between the physically closest and the provider actually included in the reply.

All simulations are conducted in a square area of 800x800m. *Akaroa* [16] was used to remove initial transient periods of the simulation. Once in steady state, samples were collected until means could be estimated with a confidence interval width smaller than 5% or narrower than 0.05 in absolute units, whichever is higher, at a 95% confidence level.

5.1 Varying cache lifetimes

The first question in context with caching is the proper cache lifetime. Here, the different characteristics of `store`, `cab`, and `access point` providers should require different lifetimes. The cache lifetimes were varied from 1s to 600s.

The simplest case are the `store` providers. We simulated 9 stationary `store` providers placed on a 3x3 grid surrounded by 50 mobile hosts. Because of their stable service configuration, cached information is always consistent and the caching lifetime can be arbitrarily large. Figure 6 shows the messages per request (note the logarithmic x axis); there are no incorrect replies for this provider type. Negative announcements would never be sent in this scenario as services never become unavailable.

Two curves (1) and (2) are shown with 50 or 20 active clients, respectively. The larger benefit of caching in the face of many clients (1) can be explained in two ways: Firstly, more frequent requests enable better exploitation of cached information. Secondly, client requests also keep routes in between intermediate hosts up-to-date. Because the messages metric also counts SREQ messages that are only sent to discover routes (although the provider address is known), requests require less SREQ messages in face of existing routes.

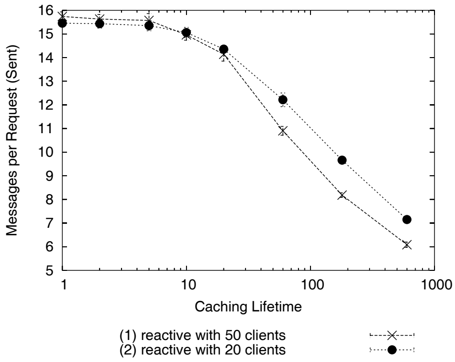
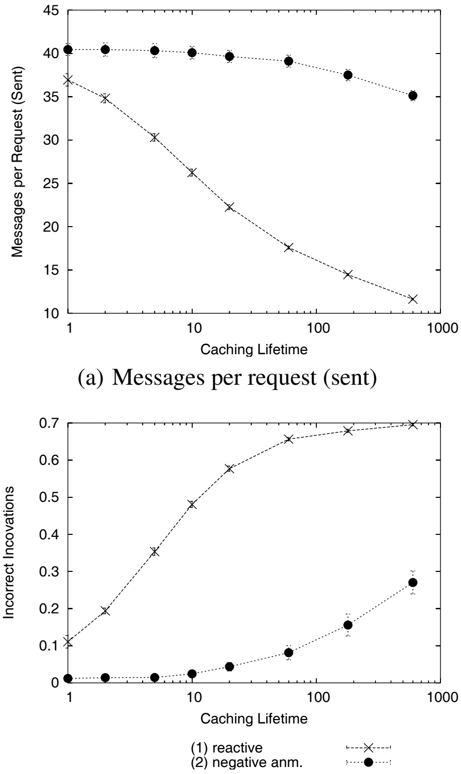


Figure 6: Effects of increasing cache lifetime, `store` providers



(a) Messages per request (sent)

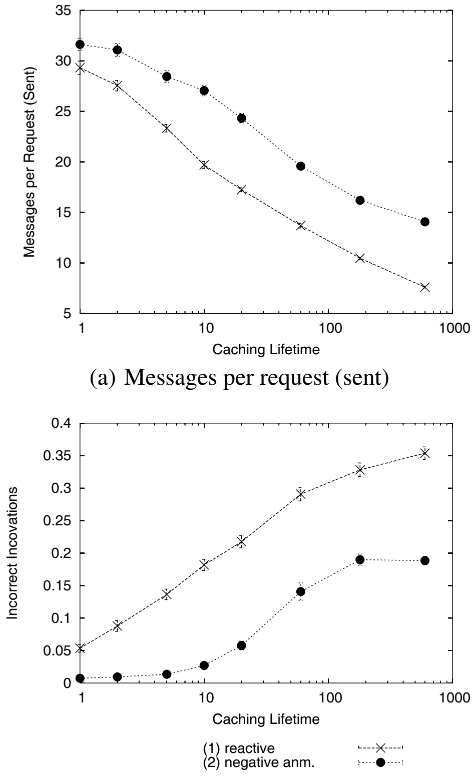
(b) Incorrectly invoked requests

Figure 7: Effects of increasing cache lifetime, `cab` providers

Services very frequently become unavailable for `cab` providers: Five `cab` providers and 50 clients requesting the service were simulated next. Figure 7 shows caching lifetime effects on number of messages per request and on the number of incorrect invocations for the `cab` providers, comparing a purely reactive protocol (1) and one with negative announcements (2).

While the messages per request in Figure 7(a) monotonously decrease with growing lifetimes, they must always be regarded jointly with the presented ratio of incorrectness: The reactive protocol suffers from a very high number of incorrect replies in Figure 7(b), even with moderate cache lifetimes.

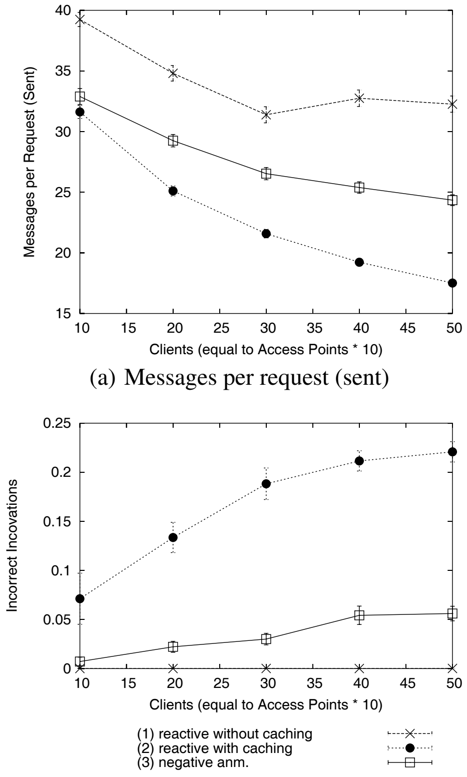
Negative announcements are able to keep the ratio of incorrect replies low for small to moderate cache lifetimes. When lifetimes become very large, more incorrect answers are produced: Any node



(a) Messages per request (sent)

(b) Incorrectly invoked requests

Figure 8: Effects of increasing cache lifetime, `access point` providers



(a) Messages per request (sent)

(b) Incorrectly invoked requests

Figure 9: Effects of increased network load

which is "missed" by the negative announcement process – which is limited in scope – re-injects the information into the network and causes high inconsistency. On the other hand, higher lifetimes cause the amount of messages used by negative announcement floods to increase: More and more inconsistent information is cached and due to smart forwarding, negative announcement floods must spread further.

The presented two provider types are extreme in the sense that they are insensitive to requests (`stores`), making caching very attractive, or become unavailable even after a single request (`cabs`), making caching quite questionable. In the second case, negative announcements have performed acceptably – the interesting question is what is their performance for providers where caching does make sense, but which do switch state from time to time? The `access point` providers are such an example.

Figure 8 shows the effects of various caching lifetimes when discovering access points. Here, 50 clients issue discovery requests and subsequently begin internet traffic sessions at one of the 5 access points. Access points accept up to 10 parallel sessions; the parameters are set such that they operate at their load limit (change their state frequently) but at least one of the providers is available most of the time. This provides a challenging scenario for service discovery. As expected, the negative announcement protocol can improve the ratio of incorrect replies to acceptable levels. The number of messages per request is higher but yet acceptable in face of the presented consistency improvements. Also, the utilized messages are always fewer than without caching.

The incorrectness stays low only up to certain lifetime levels; similarly to the `cab` provider type leftover inconsistencies may propagate in the network. The figures for `cab` and `access point` providers motivate lifetimes values of up to 20 seconds; these improve efficiency but still yield acceptable inconsistency levels.

5.2 Varying number of clients and providers

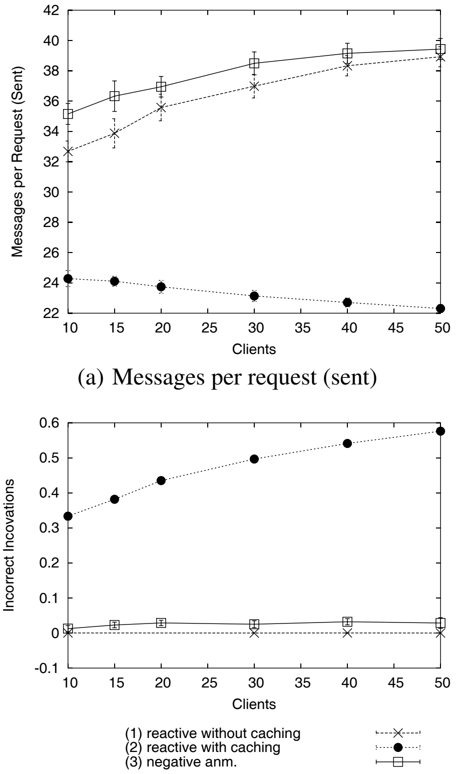
In addition to varying the caching lifetime, it is interesting, in particular for the `access point` providers, to look at the potential influence of high network load on performance. As provider availability influences the messages metric (expanding ring searches at maximal scope employ much more messages) we chose to keep the load on each access point constant but to increase the strain on the protocol. In Figure 9, among a constant number of 55 hosts access points and clients are added: The first data point corresponds to 10 clients and one access point, the second to 20 clients and two access points etc. This ratio also lets access points operate at the limit of parallel sessions. The figure shows the reactive protocol without and with caching, and a protocol with negative announcements.

The reactive protocol without caching (1) marks one side of the results range: While the amount of messages needed for each request is comparably high, naturally the amount of incorrect requests is constantly zero, as no inconsistent information is stored in intermediate nodes.

The reactive protocol with caching (2) marks the other side of the results range: While the amount of messages needed for each request is relatively low, a high portion of incorrect requests is measured.

Curve (3) shows the protocol including negative announcements: Negative announcements significantly reduce the amount of incorrect requests in Figure 9(b) even under high load and result in a good compromise between effort and correctness. This tradeoff is better than what could be achieved without negative announcements and only relying on short cache lifetimes.

Slightly different considerations apply to the `cab` provider type in Figure 10. While negative announcements do reduce incorrectness, the protocol without caching performs best, because of the `cab` application model (the provider becomes unavailable at the first discovery).



(a) Messages per request (sent)

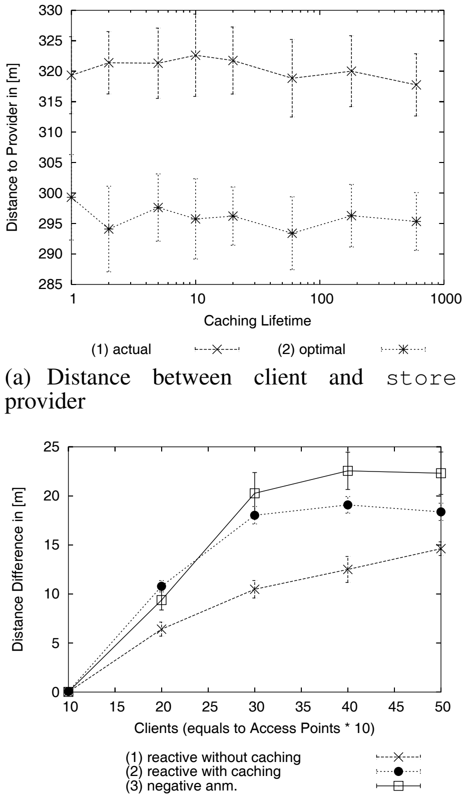
(b) Incorrectly invoked requests

Figure 10: Effects of an increased number of users in the `cab` scenario

5.3 Physical distance considerations

One of the basic questions that we set out to investigate was whether the implicit mapping of proximity to network neighbors is sufficient to find the (nearly) closest provider (out of several available ones) by choosing the first reply. This study was performed using the more detailed MAC model. Choosing the first reply is based on latency could be suboptimal due to high network load and interference, effects which are well represented by the above model.

Nevertheless, results are satisfactory: Figure 11(a) shows the optimal and the actual distance in meters between client and provider in a scenario of 50 clients sending requests at four store providers. The providers are set on a straight line near the bottom of the simulation area 200 m apart for two reasons: Firstly, the scenario results in longer routes between client and provider with a mean of around 3 hops and suboptimal replies are more frequent in longer routes as the provider choice is often made at (caching) intermediate nodes with incomplete information. Secondly, the provider positions ensure high penalty if a wrong provider is chosen.



(a) Distance between client and `store` provider

(b) Distance difference between replied and closest `ap`

Figure 11: Correlation with physical proximity

With varying cache lifetimes, the location error of choosing a suboptimal provider, averaged over many requests, stays within a very acceptable range of 30 m. Essentially, at high service binding lifetimes, this is equivalent to the optimality of the underlying AODV routing information: After a transient phase all four providers are cached at all nodes, and the remaining provider choice is made based on the timeliness of cached hop-count information and the symmetry with which unknown routes are discovered. Similar numbers were obtained when varying the number of clients and are omitted here.

Let us now consider the more unstable case of `access point` providers: Here, the scenario brings about the discussed consistency challenges. Also, the network load is higher due to session data traffic. We chose the access point locations such that each provider "covers" a certain portion of the simulation area: e.g. the first in the middle, two on a diagonal (280 m apart), three in a triangle (over 400 m apart) and so forth.

Figure 11(b) shows distance differences between the closest access point and the one actually included in the reply. The network load was increased by simultaneously adding clients and access points, the providers were set on fixed locations, as described above. Naturally, all replies are optimal in presence of one provider. For more than one provider, be aware that the curves are not comparable because they incorporate a different subset of requests: The distance metric could only be computed for *correctly* replied requests (the optimal provider is always the closest *available* provider) and we must thus disregard replies with unavailable providers that might be even closer than the optimal one. But the ratio with which the replied provider is available depends on the protocol variant, e.g. the large ratio of incorrectly invoked requests in the reactive case (2) was not included in the metric; these requests were replied correctly in the negative announcement variant (3) and may account for its slightly worse numbers.

The important result is that all three curves yield an optimality gap which is surprisingly low (below 25 m) throughout the scenarios encompassing up 5 providers and 50 clients. From the correctly replied requests, all protocol variants yield satisfying results with respect to locality. While we are aware that wireless propagation in urban environments is more complex than our model, the MAC simulation effectively captures the effects on latency which is caused by interference and high network traffic. The numbers confirm that choosing the first reply of the network is a good indicator and not significantly influenced by the latency variance of multi-hop routes facing the load of the described models.

6. CONCLUSION

The basic conclusion to draw is that service discovery can indeed be efficiently integrated with route discovery; efficiency refers to message overhead, correctness of the delivered answers and even geographic optimality of a discovered service provider. But the detailed mechanisms depend on the type of the service provider – the diversity of service providers requires a corresponding diversity in service discovery protocols to work well in a mobile ad hoc network. Providers which are stable both in space and in their service offerings are best supported by an (existing) reactive protocol with large caching lifetimes in the network. For volatile providers which frequently change their availability, we have shown that their needs are essentially incompatible with caching and that simple reactive protocols work best. Most interesting is the case of semi-stable providers: For this type of providers, we have demonstrated that explicitly removing stale cache information is well invested effort, outperforming our base protocol. As this last protocol also performs acceptably in the other cases, it is a good candidate for actual deployment. It even achieves a reasonable closeness of discovered service provider and potentially closest one (which, in reality, would of course depend largely on radio and MAC characteristics).

The protocol described here opens a number of options for further investigations. In particular, using different cache lifetimes adapted to the type of service appears promising, as does adapting forwarding range of announcements based on mobility of different nodes. In addition, the locality of discovered providers should be improved by assuming that all nodes in fact know their positions (e.g., via GPS) and by using geographic routing instead of a pure ad hoc routing. We are currently implementing such a complementary approach to the problem and intend to compare these two solutions.

7. REFERENCES

[1] A. Acharya, A. Misra, and S. Bansal. A label-switching packet forwarding architecture for multi-hop wireless lans. Technical Report RC22512, IBM Research, June 2002.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking*, 2(5):483–502, 2002.

[3] L. Cheng. Service advertisement and discovery in mobile ad hoc networks. In *Proc. of CSCW 2002*, New Orleans, LA, USA, November 2002.

[4] Proxim Corporation. Orinoco 11b client PC card. Internet Datasheet, 2003.

[5] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl. A mobility framework for OMNeT++. 3rd International OMNeT++ Workshop, January 2003.

[6] K. Edwards and T. Rodden. *Jini: Example by Example*. Prentice Hall, June 2001.

[7] C. Frank. A hybrid service discovery protocol for mobile ad-hoc networks. Master's thesis, Technische Universität Berlin, Berlin, Germany, August 2003.

[8] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29, January 1981.

[9] E. Guttman, J. Veizades, C. Perkins, and M. Day. RFC 2608: Service location protocol, version 2, June 1999.

[10] R. Koodli and C. E. Perkins. Service discovery in on-demand ad hoc networks. IETF draft (work in progress), October 2002.

[11] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Inc., second edition, 1991.

[12] N. Nahata, P. Pamu, S. Garg, and A. Helmy. Efficient resource discovery for large scale ad hoc networks using contacts. *ACM SIGCOMM Computer Communications Review*, 32(3):32, July 2002.

[13] V. Park and J. Macker. Anycast routing for mobile networking. In *Proc. of MILCOM*, October 1999.

[14] V. Park and J. Macker. Anycast routing for mobile services. In *Proc. of Conference on Information Sciences and Systems (CISS)*, March 1999.

[15] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of INFOCOM*, pages 1405–1413, Kobe, Japan, 1997.

[16] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, pages 132–139, January 2002.

[17] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. IETF draft (work in progress), February 2003.

[18] P. Ratanchandani and R. Kravets. A hybrid approach for internet connectivity for mobile ad hoc networks. In *Proc. of WCNC*, New Orleans, LA, USA, March 2003.

[19] RFC 3344: IP mobility support for IPv4, August 2002.

[20] Salutation architecture specification v2.0c (part-1). The Salutation Consortium, June 1999.

[21] Y. Sun, E. M. Belding-Royer, and C. E. Perkins. Internet connectivity for ad hoc mobile networks. *Intl. J. of Wireless Information Networks (special Issue on Mobile Ad hoc Networks)*, 9(2), April 2002.

[22] UDDI version 3.0 published specification, July 2002.

[23] Universal plug and play device architecture. UPnP Forum, June 2000. Version 1.0.

[24] A. Varga. The OMNeT++ discrete event simulation system. In *Proc. of ESM*, Prague, Czech Republic, June 2001.

[25] Y. Yi and S. Lee. On-demand multicast routing protocol (ODMRP) for ad-hoc networks. IETF draft (work in progress), November 2002.

[26] J. Yoon, M. Liu, and B. D. Noble. Random waypoint considered harmful. In *Proc. of INFOCOM '03*, San Francisco, CA, USA, April 2003.