# P2P-Based Semantic Service Management in Mobile Ad-hoc Networks*

Peter Baumung
*Institute of Telematics*
*University of Karlsruhe (TH)*
baumung@tm.uka.de

Stefan Penz
*Chair of Computer Science 4*
*RWTH Aachen University*
penz@cs.rwth-aachen.de

Michael Klein
*Institute for Pr. Str. & Data Org.*
*University of Karlsruhe (TH)*
kleinm@ipd.uka.de

## Abstract

*Service management in mobile ad-hoc networks has been a hot research topic in the past few years. With this demonstrator, we present the first service-oriented middleware that unifies semantic service description, service query management and Peer-to-Peer-based query dissemination. By adding a comfortable graphical interface on top of our middleware, we provide and demonstrate a fully functional software package that lets end-users elegantly manage and share application-level services on their wireless devices.*

## 1. Introduction

Mobile ad-hoc networks (MANETs) can exist wherever wireless devices cooperate without accessing any fixed infrastructure. As transmission ranges are limited, intermediate devices bridge distances between farther nodes and thus enable multi-hop communication. Compared to traditional infrastructure-based WLANs, MANETs feature an increased flexibility and ease of deployment, since they arise from fully self-organizing devices that require no additional infrastructure.

From an end-user's point of view, data exchange and the use of external functionality will be the main reasons for MANET-based communication. A MANET's usefulness will thus heavily depend on the data and services made available within the network. The latter's spontaneity however becomes problematic, as devices (and thus services) may appear or vanish from the network at any time. Therefore, service management (offering, discovering and using services) plays a key role in the operation of MANETs. By developing a service-oriented middleware, we face the problem of semantic service description as well as service query management and query dissemination. As a result, we provide and are able to demonstrate a fully functional real-world implementation that lets end-users flexibly manage and share application-level services on their wireless devices using a graphical interface put on top of our middleware. By basing the latter on a Peer-to-Peer (P2P) approach, our solution shows to be easily applicable and adaptable to many different scenarios.

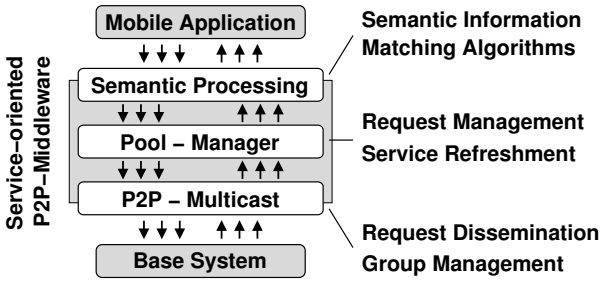


**Figure 1. System Architecture**

For demonstrating our middleware's capabilites, we choose a simple e-learning scenario. Students on a university campus use their mobile devices (e.g. PDAs or laptops) to spontaneously set up a MANET in order to share different services. The latter could e.g. comprise services for downloading exercise sheets or solutions as well as spell checking and dictionary services. Thus, our middleware on the one hand needs to include powerful service descriptions. Indeed, users must be able to precisely describe requests, in order to obtain only the most accurately matching results. On the other hand, the dynamic nature of MANETs imposes several requirements regarding service discovery and management. Due to a device's mobility, end-to-end network connections between service users and providers are significantly less stable than in fixed environments. Moreover, service providers may enter or leave the network at any time without any premonition. The following sections, that in detail present our demonstrator, shows how our solution copes with the arising problems. We first describe our middleware's architecture in Section 2. The demonstrator's hardware setup is discussed in Section 3. A brief list of our demonstrator's features and a short summary are given in the Sections 4 and 5.

*This work was funded by the German Research Foundation within the SPP 1140.

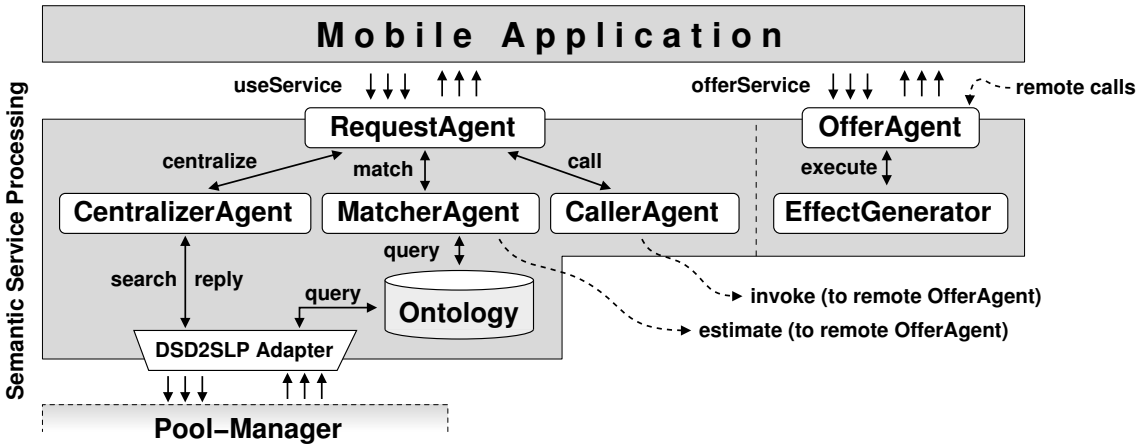


**Figure 2. The Semantic Service Processing Component**

## 2. System Architecture

The architecture of our service-oriented middleware is depicted in Figure 1. As can be seen, a mobile application (that includes a graphical user interface) runs on top of our middleware. The latter comprises three sublayers: The top sublayer (presented in Section 2.1) provides a semantic service description and appropriate matching algorithms to enable a service usage that is driven by the application and transparent to the user. To find appropriate offers matching a given service request, it relies on the second sublayer (Section 2.2), which provides techniques based on SLP and multicast for managing service requests as well as collecting and caching service offers. The third sublayer (Section 2.3) integrates the efficient dissemination of service requests using P2P-multicast techniques. Needed by every P2P-application, a base system (Section 2.4) yielding network access resides on bottom of Figure 1.

### 2.1. Service Description and Matching

The top sublayer of our middleware is the *Semantic Service Processing*. Its goal is to enable transparent service usage by automatically matching, selecting and invoking services. The basis for this is the semantic *DIANE Service Description Language* (DSD) [4]. It provides unambiguous and ontology-based service descriptions. Services are not represented by their information flow but regarded as sets of possible state changes, which can be selected and configured by the service requestor. DSD separates between descriptions for offered and requested services and allows to completely integrate user preferences and context restrictions into request descriptions.

Figure 2 depicts the components of the Semantic Service Processing. They are split into two parts representing two roles: the user can offer a new service, which is handled by the *OfferAgent* running on his device, or he can access an existing service. This is handled by the *RequestAgent* on his device. The procedure of such a service usage is also presented in Figure 2: At first, the user chooses to use a certain functionality of the mobile application, which sends an appropriate service request description to its *RequestAgent*. The RequestAgent handles this request autonomously on behalf of the application and processes it in three steps: First, it contacts its *CentralizerAgent* to gather possibly appropriate service offers from the network. In our implementation, this is done by transforming the high level, semantic DSD description into a lower level, syntactic SLP description and instructing the Service Management on the second sublayer to perform the service discovery with a rough syntactic pre-selection. The incoming results are processed by the *MatcherAgent*, which performs a finer and semantic matching based on DSD. To do this, it queries additional information about the affected real world from an (distributed) ontology or contacts the service provider directly to get additional up-to-date information about the service by performing a remote estimation call to the respective OfferAgent. In addition to the appropriateness of a service provider, the MatcherAgent also determines the messages that have to be sent to the provider in order to achieve the requested results. This information is passed on to the *CallerAgent*, which invokes the service provider by exchanging messages with the service provider's OfferAgent. The OfferAgent on the server side realizes the service by executing the underlying *EffectGenerator* (i.e. the implementation of the service).

### 2.2 Service Management and Request Caching

The second sublayer of our middleware deals with the service management. It receives service requests from the top sublayer and discovers appropriate service providers

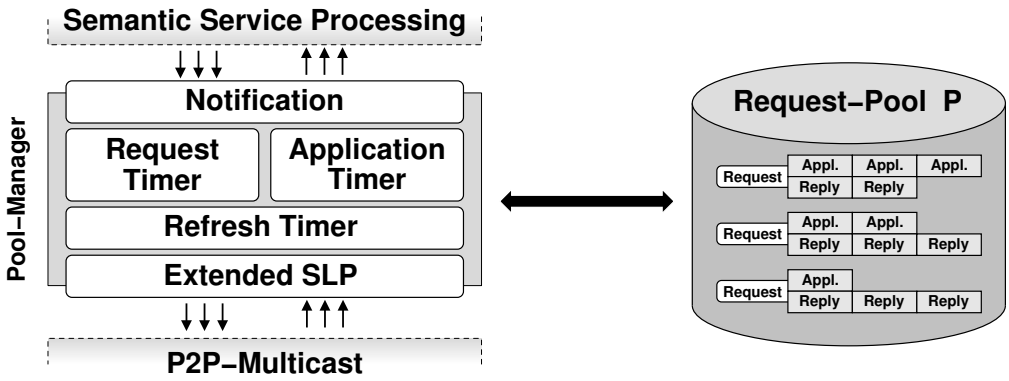


**Figure 3. The Pool Manager Component**

spontaneously. As network access is very expensive in a MANET and applications use a service request more than one time in different configurations, it is reasonable for the system to be up-to-date with the current provider situation all the time. Therefore, we designed and integrated a service discovery and caching mechanism that maintains a pool of current service provider information [5]. This information is updated periodically and registered applications are notified about changes. This mechanism allows the user application to quickly select a service provider in case of a repetitive query. This middleware sublayer not only manages service access information, but also – possibly dynamic – service property information so that it is possible to detect the most convenient service providers in this moment or for this context.

The components of the service management sublayer are depicted in Figure 3. The main entity of this sublayer is a system service called *Pool-Manager* residing at the service user's device. It maintains a local database called *Request-Pool* that stores a set of different requests issued by the DSD-Middleware. A list of corresponding replies is associated with each request as well as a list of references to applications interested in this information.

When the Pool-Manager receives a service request from the DSD-Middleware, it first searches the Pool for matching service information. If a matching offer is found, the corresponding replies are immediately returned to the DSD-Middleware and a reference to the requesting application is added to the list associated with the request. If the request cannot be answered from Pool information, it is added to the Pool as well as the application reference and the SLP service discovery mechanism (see below) is called to retrieve current provider information from the network.

The Pool-Manager utilizes several timers for the maintenance of the Pool contents. The *Refresh Timer* periodically calls the SLP service discovery in order to update the stored provider information. Upon reception of the corresponding replies, the notification component informs all registered applications about the discovered service providers and their properties. On the basis of this information the applications can prearrange the selection of an alternative provider. *Application Timers* and *Request Timers* remove outdated application references and requests in order to minimize the amount of network traffic imposed by unnecessary service discovery messages. If an application wants to be kept informed about available providers for a longer period of time, it has to be aware of this timer and re-issue the request before the application timer expires.

The service discovery mechanism used by the Pool-Manager is based on the IETF *Service Location Protocol* (SLP, [2]) which offers efficient and robust service discovery procedures suitable for dynamic ad-hoc networks. We extended the SLP system to support dynamic service attributes so that the user is able to select appropriate providers also on the basis of transient provider information (e.g. service load). Additionally, we optimized SLP for the efficient usage of sparse resources of wireless networks.

In order to enable robust service discovery even in the absence of central directories, SLP relies on multicast request dissemination. Every provider is member of a predefined multicast group. Requests can be issued to this group and matching providers answer with the requested information. Therefore, an efficient multicast routing mechanism is necessary, which will be described in the following subsection.

### 2.3. P2P-Based Request Dissemination

For the efficient multicasting of service requests, we chose a P2P approach that uses an overlay network for data dissemination. Overlays proved to be an ideal approach for organizing groups of users in MANETs, since

- Overlays feature an increased degree of robustness because of their decentralized design that overcomes the necessity of central components.

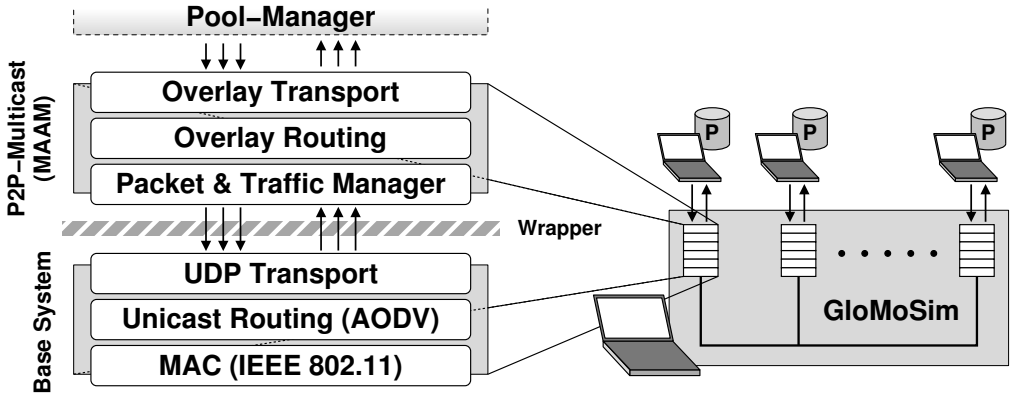


**Figure 4. The P2P-Multicast Component and the Base System's Layers**

- Overlays easily allow the differentiation and coexistence of user groups sharing diverse interests.
- Overlays affect third party nodes neither with additional protocol requirements nor with the management of complex state information.
- Overlay-based protocols thereby feature an extreme adaptability in respect to a specific group's needs.

In contrast to classic multicast routing protocols that reside on a device's network layer, overlay multicast leaves packet duplication and routing entirely to a multicast group's members. To do so, a virtual network structure (the overlay) is formed in a first step, by interconnecting group members using common transport links. By propagating the established overlay topology between the group members in a second step, multicast routing on the application layer is eventually enabled.

As has been proven in previous studies, TCP in its standard implementation shows to be inadequate for multi-hop communication in MANETs [3]. We thus decided to use unreliable UDP for establishing the overlay's transport links. This, however, requires us to face the problem of reliable data forwarding on our own [1].

The way reliability is best provided for a specific application heavily depends on the actual application's requirements and its emitted traffic. As optimal performance can only be achieved by meeting these requirements, reliability needs to be precisely configurable. To attain this level of configurability, we developed a Modular Architecture for Application-layer Multicast (MAAM). The MAAM, which in a simplified version is shown on the left side of Figure 4, decomposes an P2P-multicast service into single modules, such as e.g. overlay transport, overlay routing and packet queueing (traffic management). By defining fixed interfaces between modules, the latter become arbitrarily interchangeable. Applications, on the one hand, are thus enabled to dynamically set up multicast services by linking different modules reflecting their requirements. On the other hand, modules might be designed in order to provide mechanisms tailored to the specific characteristics of an application. Since modules not only implement reliability mechanisms (overlay transport) but also overlay routing (topology construction), the MAAM allows for a flexible composition of multicast services.

While we have a variety of modules available to implement different reliability mechanisms and overlay topologies, in this demonstrator we choose an unreliable transport module for showing the pool-manager's request caching robustness. Moreover, we use an extremely lightweight tree-based overlay module (for clear visualization purposes) that we consider ideal for service request dissemination.

### 2.4. Base System

As depicted on the left side of Figure 4, overlay multicasting approaches require a base system yielding lower layer functionality such as UDP, unicast routing and medium access (MAC). In order to facilitate simulative evaluation and real-world operation of protocols and MAAM-modules, the MAAM-Stack (cf. Section 2.3) abstracts from a specific base system by relying on an additional wrapper layer. A wrapper provides a generic interface that enables the MAAM-Stack to operate on top of arbitrary base systems. As the latter become interchangeable, identic implementations of modules can be operated in simulative as well as in real-world environments.

## 3. Demonstrator Setup

Our demonstrator consists of three palmtop computers that represent single mobile nodes in a large MANET on a university campus. These nodes run a mobile graphical e-learning application (cf. Sections 2) that allows students

to offer and use study-related services using their mobile devices. Students e.g. can provide file transfer services to share study material such as exercise sheets or papers.

Each palmtop is connected via Bluetooth to a laptop which visualizes the operations of the corresponding node's upper two sublayers of the middleware. The palmtop regularly exchanges status information with the associated laptop in order to synchronize the visualization with user interactions and internal events. The operations of the middleware can be stopped at significant points so that it is easy for a spectator to follow the steps of the demonstration.

A campus-scale MANET as mentioned above can hardly be shown in an artificial hardware-only demonstrator, since it requires an increased number of mobile devices. As depicted on the right side Figure 4, we therefore operate the MAAM on top of several virtual base systems within the GloMoSim simulation environment. This enables us to simulate complex network and node behavior (GloMoSim e.g. features a full IEEE 802.11b implementation as well as different radio propagation models) while in parallel visualizing the resulting scenario. Requiring a significant amount of CPU power for real-time MANET simulation, GloMoSim runs on a separate (fourth) laptop. By connecting all four laptops using a conventional Fast Ethernet network on the one hand and by appropriately configuring the routing layers on the other hand, the traffic generated by the three palmtop/laptop pairs can directly be injected in GloMoSim's MANET simulation. Three of the simulated nodes are thus directly associated to the real palmtop/laptop pairs and act as entry and exit points for the real traffic into the simulation and vice versa. GloMoSim's MANET simulation itself comprises an arbitrary number of mobile nodes (typically about 100-200 nodes on a 1000m$\times$1000m square area), of which some run the MAAM as described in Section 2.3.

A visualization process at the GloMoSim host graphically depicts the simulation environment and the operations of the P2P-multicast overlay. By interpreting the incoming and outgoing messages, it can even display limited service status information near the associated simulation nodes.

All in all, our setup allows us to demonstrate step-by-step a complete service access procedure. The latter ranges from the semantic description of desired services over the service information caching mechanism, the discovery procedure based on efficient P2P-multicast, the semantic matching of service information and provider selection to the access of the selected service provider.

## 4. Features

Much effort has been put in an appropriate visualization of the different algorithms of the implementation. When running the demonstrator, the following features are visible:

- a mobile graphical application running on a real PDA, which can be used by a spectator in order to configure, offer, or access services,
- a visualization of the interaction of the different software components trying to fulfill the service request,
- a step-by-step visualization of the matchmaking of the semantic service descriptions,
- a step-by-step visualization of the sending, receiving and caching process within the service discovery,
- a possibility to inspect the SLP requests and the corresponding results,
- a visualization of the moving participants of the MANET as well as their adaptation of the overlay links,
- a visualization of the message dissemination in the overlay network for multicast routing.

## 5. Summary

With this contribution, we demonstrate the first fully functional implementation of a P2P-based middleware for semantic service management in MANETs. By unifying semantic service description, service request caching and service request dissemination via P2P-multicast, we provide a practical and viable solution to the problem of efficient discovery and use of application-level services in MANETs. We demonstrate our solution by adding a graphical user interface on top of our middleware, that lets spectators initiate customized service offers or uses and follow the process of service discovery by watching our middleware's respective visualizations.

## References


[1] P. Baumung, M. Zitterbart, and K. Kutzner. Improving delivery ratios for application layer multicast in mobile ad-hoc networks. *Elsevier Special Issue on Computer Communications*, 28(14):1669–1679, 2005.

[2] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2 . RFC 2608 (Proposed Standard), June 1999.

[3] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of IEEE/ACM MOBICOM '99*, pages 219–230, Seattle, WA, USA, August 1999.

[4] M. Klein, B. König-Ries, and M. Müssig. What is needed for semantic service descriptions - a proposal for suitable language constructs. *International Journal on Web and Grid Services (IJWGS)*, 1(2), 2005.

[5] S. Penz. SLP-based Service Management for Dynamic Adhoc Networks. In *Proceedings of the 3rd Workshop on Middleware for Pervasive and Ad Hoc Computing (MPAC '05) co-located with ACM/IFIP/USENIX Middleware 2005*, Grenoble, France, Nov. 2005.