

# A Highly Flexible Testbed for Studies of ad-hoc Network Behaviour

Hartmut Ritter, Min Tian, Thiemo Voigt, Jochen Schiller  
*Freie Universität Berlin, Germany*  
*Institut für Informatik*  
*{hritter, tian, voigt, schiller}@inf.fu-berlin.de*

## Abstract

Studies of mobile ad-hoc networks are often based on simulation and their underlying, necessarily simplified assumptions of physical reality. In order to analyse the practical problems we built our own hardware and software. The hardware consists of a core Motorola controller and different wired and wireless interfaces like a Bluetooth and a 433/868 MHz RF module. It allows therefore analysing different scenarios: First, the deployment of a pure ad-hoc network using Bluetooth or 433/868 MHz RF modules. Scenarios going beyond this cover the connection of Bluetooth piconets using the complementary RF technology. This overcomes the proximity requirements of Bluetooth scatternets. Finally, hybrid scenarios with some nodes connected to the Internet and providing Web access over a multihop ad-hoc network can be studied.

In this paper we present the testbed and solutions realized up to now. These include home automation scenarios as well as support for mobile ad-hoc games.

## 1. Introduction

The behaviour of wireless ad-hoc networks is strongly determined by physical limitations of radio communication. Integrating the underlying physical laws into a simulator increases complexity, so that either simplifications are used or the scope of the simulator has to be reduced to a radio-layer simulator. While simulation studies are useful, they should be complemented with real world experiments in order to reveal phenomena the simulation cannot cover. For example, doing real world experiments with IEEE 802.11b based wireless ad hoc networks, Lundgren et al. discovered the existence of so-called communication gray zones that previous simulations were not able to detect [6]. Furthermore, while network simulations usually model queuing and propagation delays very accurately, less emphasis is put on processor cost. If methods exist, they

are usually limited. For example, in NS-2 the standard way to model node delays is by using a timer and performing some actions on the expiration of the timer. In our testbed, the capabilities of the hardware are limited and therefore tasks such as packet processing requires much more time than in traditional routers. Real world experiments inherently include these increased processing costs.

For these reasons, we decided to build a testbed using fully-understood hard- and software that allows to perform measurements and to evaluate scenarios covering all layers beginning at the radio-layer and embracing all layers up to HTTP/TCP/IP and the application space. The hardware is an embedded device we call *Embedded Wireless Module* (EWM) consisting of a core controller (Motorola 68HC912) and I/O peripherals as well as different wireless and wired network connectors. The device is, for example, equipped with Bluetooth and a 433 MHz RF module.

In this paper, we demonstrate some experiments we have performed with this hardware. We present some general observations as well as some real world experiments covering different application domains such as ad-hoc gaming and home automation. While vertical handover times between Bluetooth and the 433 MHz RF modules are in the order of milliseconds, our measurements show that the handover time for a Bluetooth slave between two different piconets is too slow to enable highly interactive multi-user games in an ad-hoc gaming infrastructure we present in this paper. The presented home automation scenario evaluates the efficiency of service usage in ad-hoc networks. While demonstrating the feasibility of this approach further optimizations of some of the steps are necessary.

The remainder of this paper is outlined as follows: In the next section, we describe the hard- and software of this testbed. In Section 3, we discuss general observations found in our experiments and in Section 4 we present selected experiments. Finally we compare with related work and discuss the overall conclusions.

## 2. Description of the Testbed

In this part we give a description of the hardware used in the testbed. It was designed to be as flexible as possible. Thus, the testbed does not use any special hardware, all components are fully documented and even the hardware-related low-level program code is available.

### 2.1. Hardware of the Testbed

One of the hardware modules that builds our testbed is shown in Figure 1. The embedded module basically consists of a core controller (Motorola 68HC912) and I/O peripherals as well as different wireless and wired network connectors.

On the upper left hand side resides a 433 MHz RF module from Radiometrix [7] used for long-range interconnections between embedded wireless modules. On the opposite side of the display the class 2 Bluetooth module from CSR [1] can be seen. Hidden behind the display resides the core controller. It communicates over serial connections with the 433 MHz and the Bluetooth module. In addition there are analogue and digital I/O-port on the bottom, an Ethernet connector on the left side, CAN bus connectors on the right and a serial interface for debugging purposes.



Figure 1. Embedded Wireless Module

In most scenarios the core Motorola controller will maintain the state of all these peripherals and coordinates the communication and probably routing over the different network interfaces. The fixed Ethernet jacket serves in some wired scenarios as Internet connection of a Web Server realised on the embedded module.

### 2.2. Software

From the software point of view full access to nearly all communication layers is possible.

The **433 MHz RF module** basically offers a bitstream interface and converts all bits sent to it over the serial interface into an AM-modulated signal at 433 MHz frequency. Similarly, data received over the air is passed to the serial interface as an unmodified bitstream. As the module can not send and receive simultaneously, it defaults to the receive mode and switches to the send mode only if data is passed to it over the serial interface. Functions to achieve long-term parity between zero and one-bits as well as all kinds of framing and error checks have to be implemented in software running on the Motorola controller.

The situation differs a bit with the **Bluetooth module**. As it would be too much overhead to realize all layers of the Bluetooth protocol stack on the Motorola controller, we take advantage of the fact that the Bluetooth chip itself can take over the lower layers of protocol handling. Beyond that, the Bluetooth module even allows to download and deploy small programs that can access all provided Bluetooth functionality like inquiry, service discovery, and data access on the RFCOMM level and above. These programs run on the Bluetooth module, thus relieving the Motorola controller from a lot of time-critical protocol handling. The air interface of the Bluetooth module follows the Bluetooth standards, providing for example serial port emulation or Internet access using BNEP, whereas the interface to the Motorola controller is a serial interface. Depending on the scenario, communication between the Motorola core controller and the Bluetooth module takes place either on the level of a simple ASCII protocol or by exchanging Ethernet frames over the serial line between both.

We do not use an operating system as this might consume too much resources but instead deploy a simple loop that handles all tasks and that can be interrupted by events like expired timers and incoming data packets. However, as all direct hardware access is encapsulated in C functions, application programming does not require deep knowledge of the hardware while on the other hand keeping the full flexibility.

## 3. General Observations

In this section, we present some results that are not specific for a certain scenario but results that we consider general for the kind of technologies used.

### 3.1. Media Access

The 433 MHz RF module used for our EWM modules incorporates hardware providing carrier sense. To evaluate if it is necessary to implement a complete MAC layer with carrier sense and collision detection protocols we performed several experiments. In an example experiment, one EWM module sends a broadcast which is received by two other EWM modules. Upon the reception of this broadcast, these two EWM modules immediately perform a rebroadcast. Performing this experiment we noticed very few collisions. The reason for this is that even though the EWM modules receive the first broadcast almost contemporaneously, they do not rebroadcast it exactly at the same time. With the cycle time of the EWM being 200 ns, the EWM modules do not perform the rebroadcast at exactly the same time and the carrier sense of the radio module is fast enough to note that the “faster” EWM has already started retransmitting the broadcast. The transmission of the packet is delayed until the “faster” EWM has transmitted its packet. Therefore, very few collisions occur. Since we only have a limited number of EWMs this allowed us to abstain from implementing a complete MAC layer. Of course, the situation is different in dense networks.

### 3.2. Bluetooth Connection Termination

When making some preliminary experiments with Bluetooth, it turned out that the Bluetooth chip on the EWM modules needs several seconds until it detects that a slave is out of range. We experienced the same behaviour for connections between different Bluetooth-equipped entities, namely both for the Bluetooth chip on other EWM modules as well as for Bluetooth-equipped IPAQ 3970 PDAs.

Since this behaviour is undesirable in many scenarios, we implemented another approach in which the Bluetooth chip on the EWM modules continuously measures the link quality of the connections to the slaves. If the link quality falls beyond a certain threshold, the master breaks the connection to the slave and the core controller is informed. One additional advantage of this approach is that it allows to shape the range of the Bluetooth piconet and to adapt it to the physical topology.

## 4. Scenarios

The following four scenarios show the clear advantage of real-world implementations compared to simulations because they exhibit special hardware characteristics.

### 4.1. Vertical Handover Bluetooth to 433 MHz

In this scenario, we demonstrate a vertical handover from Bluetooth to the 433 MHz technology. Bluetooth is a mature technology that provides several features such as device and service discovery, media access control and connection management. While these features facilitate ad hoc communication, the indoor transmission range of Bluetooth is only about 10 meters. On the other hand, the range of our 433 MHz modules is about 50 meters indoors, but 433 MHz does not provide any higher layers.

Therefore, it makes sense to use Bluetooth as the primary communication technology between EWM modules while these are in transmission range but to change to 433 MHz when Bluetooth communication is not possible, for example, when the EWM modules move out of range. Therefore, our system must be able to perform a handover between Bluetooth and the 433 MHz technology. This scenario shows that such a vertical handover is feasible with our EWM modules.

In this scenario, we use two EWM modules: one immobile and one mobile. In the beginning, both EWM modules are within Bluetooth transmission range and therefore they establish a Bluetooth connection with the immobile EWM being the master and the mobile EWM being the slave<sup>1</sup>. The immobile EWM continuously measures the signal strength. As the mobile EWM is moving away, the signal strength falls beyond a predefined threshold. The BT chip therefore breaks the connection to the mobile EWM and informs the core controller about the teardown of the connection. Knowing that the Bluetooth connection has been broken, the core controller issues a broadcast over the 433 MHz module searching for the mobile EWM. As an identifier the Bluetooth address is used since these are guaranteed to be unique and the EWM already knows the Bluetooth address of the mobile peer. After the reception of the search request, the mobile EWM replies with a unicast message and the two EWM modules are ready to communicate using their 433 MHz modules. The whole procedure from the teardown of the Bluetooth connection until the establishment of the “connection” using 433 MHz takes only a few milliseconds.

While in this scenario we use 433 MHz to extend the communication range between EWM beyond the meagre transmission range offered by Bluetooth, 433 MHz can also be seen a second wireless technology providing

---

<sup>1</sup> We say that the EWM performs actions but note that here the BT chip performs these actions.

fault-tolerance in case Bluetooth for some reason does not work.

## 4.2. Gaming

In this scenario, we describe our experiences with an infrastructure for ad-hoc multiplayer games that uses EWM modules as the main components. Our goal is to develop an architecture that supports ad-hoc multi-user games on mobile devices such as PDAs and mobile telephones. The architecture should require a small and cheap infrastructure only and it should be possible to set up the infrastructure very quickly and in an ad-hoc fashion. We envision for example a place like a university, a subway station and other places where people meet and possibly wait for transportation, the beginning of a lecture etc. In these scenarios, spontaneous group formation will take place if it is supported by some technical means. For example, the system could provide high-scores, a gamer's archive allowing persistent nicknames, activity control or cheating prevention.

Based on the capabilities of our hardware we developed a system for gaming support that will be described in this section.

**4.2.1. System Overview** For the envisioned kind of ad-hoc games, Bluetooth is very attractive since building small ad-hoc Bluetooth networks is straightforward. Furthermore, most modern PDAs and mobile telephones provide Bluetooth support. However, the transmission range of Bluetooth is about 10 meters and thus quite short. As argued in section 2.2, Bluetooth scatternet support has some major drawbacks. Therefore, we choose a different approach and set up what can be roughly defined as a roaming infrastructure: In order to overcome the problems of short-range Bluetooth piconets, we distribute several EWM modules across the field.

We use the 433 MHz RF technology on the EWM module (with transmission ranges of up to 300 meters) as a second wireless technology. Bluetooth is used to transport gaming data, whereas the 433 MHz technology is used to maintain a global view of the game, to track the players and to exchange gaming data between the piconets. This way, the full bandwidth that Bluetooth offers can be used to transport gaming data. Only data that has to be routed to another piconet passes the 433 MHz RF link. Another usage of the 433 MHz technology is to find out whether players have left the game.

The Bluetooth chips on the Embedded Web Server modules act as the masters of the Bluetooth piconets around them. As described in Section 3.2, the BT chip continuously measures the link quality to the slave. When the link quality of a connection passes a threshold value, the Bluetooth chip breaks the connection to the slave and sends a message to the core controller. Note that though

the EWM module could also inform its neighbours when the Bluetooth chip breaks the connection to a slave, it turned out to be simpler to inform neighbours when a new slave (player) is found only.

When the connection has been broken the slave (player) has left the piconet. When the player is within coverage of the next EWM module, the Bluetooth device of the player is inquired by the Bluetooth chip of this EWM module. The latter informs the adjacent EWM modules about the newly found player using the 433 MHz technology. If an EWM module loses a player and neither finds her again nor does it receive a notification that an adjacent EWM module has found that player, it can assume that the player no longer participates in the game. Obviously she has either left the game or turned off her Bluetooth device.

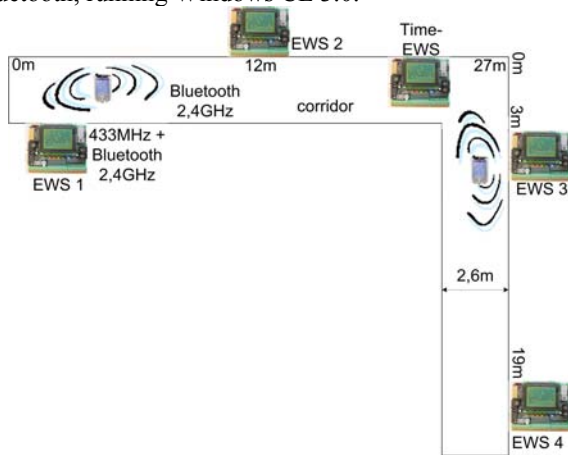
For communication between the 433 MHz RF modules we use a very simple ASCII-based protocol. The messages contain a header consisting of the sender and receiver ID, a message type, the packet length and the payload. The protocol implementation supports both reliable and unreliable messages. For our purposes we have defined several message types (e.g. FOUND when a new player is found). Thus, it is sufficient that the payload contains the Bluetooth address of the affected player.

In our current implementation the EWM modules are numbered and have a priori knowledge about their neighbours which is used for direct neighbour-to-neighbour communication. We leave the definition of a true zero configuration protocol as future work. Note that this is not a real problem. When players place the EWM modules in the field, they just have to make sure that they place the modules according to their consecutive numbering (Figure 2 shows an example).

**4.2.2. Evaluation** The infrastructure supporting ad-hoc multiplayer games described so far was fully implemented and evaluated in the context of our lab building. This allows to get practical experiences and to uncover difficulties. We see it as a platform for future research of different kinds of games.

Our testbed depicted in Figure 2 consists of four EWM modules that are placed inside the offices of our lab members. During the experiments the affected four office doors are closed. The distances between adjacent offices are 12 m (EWM 1 in office 1 to EWM 2), 15.3 m (EWM 2 to EWM 3) and 16 meters (EWM 3 to EWM 4). Fortunately, this is more than the transmission range of Bluetooth but less than the double of the transmission range. As expected, 20 meters is in the transmission range of the 433 MHz. The baud rate of the 433 MHz transmission is set to 19200 in our experiments. As client

device we use an iPAQ 3970 PDA equipped with Bluetooth, running Windows CE 3.0.



**Figure 2. Testbed**

In order to collect distributed timing information, we use a fifth EWM module, called time-EWM, this time without a Bluetooth chip. It serves for fine-granular logging of the distributed system's behaviour. The other EWM modules inform the time-EWM by sending messages over 433 MHz RF technology. The message type depicts the nature of the event. Time differences between events can be computed by calculating how often the timer interrupt between two successive events is triggered. This timer interrupt is triggered by the hardware every 65 ms. Note, that the time-EWM is not needed at all for the proper functioning of the distributed gaming infrastructure, but serves only as a monitoring station.

In order to get data about the usability of the testbed for gaming, we measure the Bluetooth handover time, which we define for the purpose of this paper as the time difference between the teardown of the connection by the first EWM module and the moment the mobile client is found by the second EWM module. This handover time is of extreme importance to determine the types of games that can be played on our architecture. Before conducting this experiment, we validated that when a player moves from one EWM module to the next, it is in the transmission range of the second EWM module before the connection is disrupted by the first EWM module because the quality of the connection falls beyond the threshold as described in Section 3.2.

In this experiment, an EWM module notifies the time-EWM when it disrupts the connection to a client. When an EWM module finds a client, it also sends a message to the time-EWM but this time another type of message. The time-EWM can now compute the handover time, i.e. the time difference between the arrivals of the two messages. Note that a Bluetooth slave can belong to one piconet

only; thus, it cannot be found by another Bluetooth master before it has left its current piconet.

Our test runs resulted in an average handover time of 1.43 seconds. The results were very similar for the three sections of our setup. We also redid the same experiments using a laptop instead of the iPAQ and achieved similar results

Due to the large handover time, we do not assume that highly interactive multiplayer games are feasible using our architecture. Unless, of course, handovers take place during non time-critical moments that are part of many games, for example before the start of a racing game or when switching between game levels. Note that, since our EWM modules monitor the signal strength, we have the possibility of warning players when they move into a zone with weak connectivity and may thus trigger a handover. Further results can be found in [8].

In summary, this scenario realized a fully mobile infrastructure allowing cross-piconet coordination and communication using two wireless technologies available on the EWM modules.

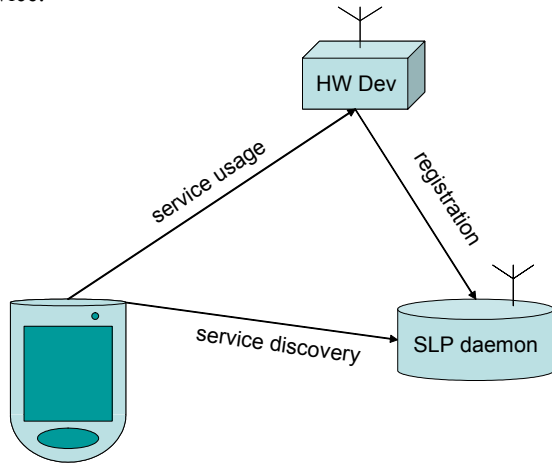
### 4.3. Home Automation

The home automation scenario focuses on the aspects of heterogeneous ad-hoc networks and demonstrates once more, that not only functionality of an approach matters, but also measurements that give an impression of the real world behaviour.

The main functionality that should be provided in the home automation scenario is provisioning of an interface for a mobile PDA user that allows the user to control and interact with devices in a room, like a coffee machine or even a light bulb. The basic interaction principle is shown in Figure 3: At first, a new hardware device (HW Dev) registers itself with a service repository. This is done using the service location protocol SLP [11]. A mobile user with a PDA entering a room would then perform SLP service discovery to find out what services are provided within the room. After receiving a complete service description, probably filtered according to its wishes, the user connects from the PDA via Bluetooth directly with the device. This third step comprises the exchange of an XML file that describes the capabilities and controls of the actual hardware device and the command issuing by the PDA user.

In order to realize this scenario within our testbed, we use the Bluetooth part of an embedded wireless module as prototype HW device that allows to switch on and off a LED directly connected with the Bluetooth device. The core Motorola controller of this module is not used, therefore the hardware could be stripped down in a real product to dimensions of 1.5 x 2 cm. A second embedded wireless device could be used for hosting the SLP

daemon. Nevertheless, a full SLP implementation would consume too much resources of both the Motorola controller and the Bluetooth module of the prototype HW Device.



**Figure 3. Home Automation Scenario**

Therefore, we modified the basic principle shown in Figure 3 by using a downstripped protocol that is translated into the SLP protocol format by a proxy running on a separate PC. We are currently porting this proxy from the PC to the Motorola controller on the embedded wireless module, so that also in full ad-hoc mode service usage will be possible. The proxy is announced via Bluetooth SDP as an additional service, so that the PDA and all HW devices have a fixed entry point for registration and service discovery respectively.

**Table 1. Interaction times**

	Action	Time needed	Comments
1	Service Registration of the HW device	5 s (excl. inquiry)	Needed only at first device usage
2	SLP proxy discovery by PDA	6 s	
3	Download of all parameters of a specific service to PDA	2-6 s	Could be cached on the PDA
4	Service usage (download of XML file, command transfer)	2.3 s	

For analysis of the behaviour besides the bare functionality it is crucial to perform measurements of the

timing behaviour of this setting. Our results using a Compaq iPAQ PDA, a Linux PC running SLP proxy and daemon and an embedded wireless module as HW device are presented in Table 1.

The results show the problems with long inquiry and service discovery times of Bluetooth. Nevertheless, actual service usage is in an acceptable timeframe of 2 to 3 seconds. This indicates that caching mechanisms and service discovery in the background based on user profiles are promising. The embedded wireless modules of the presented testbed allow real user experience that is needed for determination of acceptable interaction scenarios.

#### 4.4. Hybrid Scenarios

The embedded wireless modules also allow Internet access using TCP/IP. Therefore, we ported the implementation of Dannenberg et al. for the Texas Instruments MSP 430 (described in [3]) to the Motorola controller. Based on this implementation an embedded wireless module can get an IP address using DHCP and offer Internet access over the different wireless links. We both implemented the transport of IP packets over the 433 MHz radio link and Internet access over Bluetooth. Multihop routing across embedded wireless modules is currently being implemented.

For Internet access over Bluetooth we implemented the PAN profile using BNEP on board of the Bluetooth chip. Thus, any Bluetooth device that also implements this profile can connect with the Internet over the Bluetooth chip and the Motorola controller. The Bluetooth module handles all Bluetooth-related stuff and just exchanges Ethernet packets with the Motorola controller. The latter handles the Bluetooth device as just another Ethernet device.

The throughput in this scenario is bounded to the rate of the serial line between Bluetooth device and Motorola controller, currently set to 115.2 kbit/s. We get delay times of up to 600 ms between an external Bluetooth device and the Motorola controller. These times include protocol handling on board of the Bluetooth chip and the Motorola controller, and the typical delay of the Bluetooth link layer.

#### 5. Related Work

Prototype devices built in the *smart-Its* project [9] are based on two different microcontroller platforms, Atmel and PIC. The Atmel platform is extended with an Ericsson's Bluetooth module, while the PIC-based platform is augmented with an 868MHz module for mobile communication. The goal of the Smart-Its projects is to attach small and unobtrusive computer modules with

wireless transmission capability to real world objects. These modules communicate with each other in an ad hoc fashion. In contrast to our EWM the Smart-Its devices are obviously targeted at networks of small, nearly invisible nodes such as sensor networks.

Hartwig et al. describe the possibilities for applying general-purpose, pluggable microservers running on WAP over the Bluetooth technology for remote control purposes [4]. Their goal is to connect electronic devices to the Web with inexpensive standard technologies. This comes close to our home automation scenario but requires a WAP server on each hardware device.

While these efforts have, precisely as the hardware described in this paper, produced artefacts that enable real world experimentation, the APE testbed [5] is designed to perform large-scale, reproducible experiments with IEEE 802.11b based ad hoc networks. APE aims at assessing several different routing protocols in a real-world environment instead of by simulation. The motivation for building such a testbed is that the accuracy of simulation results depends very much on the modelling of the physical layer [10]. Furthermore, simulations of one particular protocol implementation often lack the interactions with other layers [2].

## 6. Conclusions

In this paper we presented a testbed that enables real world evaluation of ad-hoc networks. The various scenarios highlight the flexibility of the embedded wireless modules. Our results show that testbeds are needed to uncover and deal with real-world performance problems in ad-hoc networking. Our scenarios demonstrate several performance flaws with Bluetooth. For example we identify the problem that the detection of a connection breakdown is strongly delayed. The testbed assisted us with finding and evaluating a proper solution, showing the clear advantage of real-world implementations compared to simulations.

## 7. References

- [1] Cambridge Silicon Radio. <http://www.csr.com>
- [2] D. Cavin, Y. Sasson, and A. Schiper. *On the accuracy of manet simulators*. Workshop on Principles of Mobile Computing (POMC'02), October 2002
- [3] Design&Elektronik Extra: Embedded Internet. September 2001. Available at: <http://www.design-elektronik.de/extraheft>.
- [4] S. Hartwig, J- Strömann, and P. Resch. *Wireless Microservers*. IEEE Pervasive computing, April-June, 2002.
- [5] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, C. Tschudin. *A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations*. IEEE Wireless Communications and Networking Conference, 2002.
- [6] H. Lundgren, E. Nordström and C. Tschudin. *Coping with communication gray zones in IEEE 802.11b based ad hoc networks*. 5th ACM international workshop on Wireless mobile multimedia, Atlanta, Georgia, USA, 2002.
- [7] Radiometrix. <http://www.radiometrix.com>
- [8] H. Ritter, T. Voigt, M. Tian and J.Schiller. *Experiences Using a Dual Wireless Technology Infrastructure to Support Ad-hoc Multiplayer Games*. NetGames 2003, Redwood City, California, USA, May 2003.
- [9] <http://www.smart-its.org>
- [10] M. Takai, J. Martin and R. Bagrodia. *Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks*. ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001), October 2001.
- [11] J. Veizades, E. Guttman, C. Perkins. *Service Location Protocol*. RFC 2165, June 1997.