# Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices [⋆]

Frank Siegemund and Michael Rohs

Distributed Systems Group, Institute for Pervasive Computing, Swiss Federal Institute of Technology (ETH) Zurich, 8092 Zurich, Switzerland
{siegemun|rohs}@inf.ethz.ch

**Abstract.** Communication platforms for ubiquitous computing need to be flexible, self-organizing, highly scalable and energy efficient, because in the envisioned scenarios a large number of autonomous entities communicate in potentially unpredictable ways. Short-range wireless technologies form the basis of such communication platforms. In this paper we investigate device discovery in Bluetooth, a candidate wireless technology for ubiquitous computing. Detecting new devices accounts for a significant portion of the total energy consumption in Bluetooth. It is argued that the standard Bluetooth rendezvous protocols for device detection are not well suited for ubiquitous computing scenarios, because they do not scale to a large number of devices, take too long to complete, and consume too much energy. Based on theoretical considerations, practical experiments and simulation results, recommendations for choosing inquiry parameters are given that optimize discovery performance. We propose an adaptive rendezvous protocol that significantly increases the performance of the inquiry procedure by implementing cooperative device discovery. Also higher level methods to optimize discovery performance, specifically the use of sensory data and context information, are considered.

## 1 Introduction

Ubiquitous computing [10, 11] envisions that information technology is present throughout the physical environment, integrated in a broad range of everyday objects. Thereby, information technology becomes omnipresent but at the same time also invisible to users. Everyday items are augmented with self-awareness and awareness of their surroundings in order to provide new functionality and novel interaction patterns.

A first step towards this vision is to attach small computing devices to everyday objects. Smart things sense their surroundings and cooperate with one another. Information processing takes place autonomously in the background, unsupervised by human beings. To collect information about their surroundings,

[⋆] Part of this work was conducted in the Smart-Its project, which is funded by the European Commission (contract No. IST-2000-25428) and the Swiss Federal Office for Education and Science (BBW No. 00.0281).

smart artifacts need to be equipped with sensors for various physical parameters. To cooperate with other entities, e.g. to distribute collected sensor data or to use services offered by other entities, smart artifacts need to be able to communicate.

The communication of smart objects poses several challenging problems: the communication technology must be unobtrusive; the scarce radio resources must be used effectively in order to achieve scalability; communication must happen without mediation, spontaneously and without administration; previously unknown devices have to be discovered automatically; a wide range of communication patterns and traffic volumes must be accommodated for; and the least energy possible must be used.

In the Smart-Its project [17] small computing devices – so-called *Smart-Its* – were developed that are attached to everyday items providing them with collective awareness and supporting intelligent collaborative behavior. As a communication platform we investigate low-power fixed-frequency modules as well as Bluetooth [5], which is a frequency-hopping system. Fig. 1 shows a Smart-It equipped with a Bluetooth module and an attached sensor board [13].

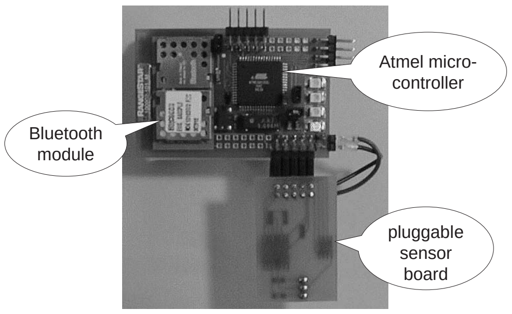


**Fig. 1.** A Bluetooth-enabled Smart-Its prototype

One reason for using Bluetooth in the Smart-Its project is that frequency-hopping as a spread-spectrum technique offers higher robustness and scalability than fixed-frequency systems. Smart-Its are designed to operate in areas with dozens of devices in range and are going to be equipped not only with standard sensors for temperature and acceleration but also with more data intensive sensors such as low resolution cameras. Hence, scalability in terms of number of devices in communication range and volume of data traffic is crucial.

The issue we focus on in this paper is the discovery of new devices, which is a necessary task for each device in an ad hoc network. Device detection is an essential part of the rendezvous layer. The challenge is to find all potential communication partners present in communication range using the shortest time and the least amount of energy possible. This issue is critical if a huge number of devices are present as in the scenarios envisioned. Although Bluetooth seems to be a promising technology for ubiquitous computing, the insufficient scalability and high energy consumption of its rendezvous layer limit its applicability. While investigating Bluetooth, we found that the Bluetooth modes for device detection – INQUIRY and INQUIRY SCAN – consume significantly more energy than normal receive and transmit modes. For the modules used in the Smart-Its project, energy consumption in INQUIRY mode is approximately twice as high as in transmit mode [12, 14]. Therefore, our goal in this paper is to reduce the energy consumption of Bluetooth's rendezvous layer during device discovery, while at the same time increasing its scalability. This is achieved through appropriate settings for the inquiry parameters, an adaptive protocol for cooperative device discovery, and the utilization of context information.

Due to a limited number of available Bluetooth modules and their restricted functionality, the performance evaluation of Bluetooth's rendezvous layer and of the proposed adaptions are based on simulation results with the Network Simulator (ns-2) [15] and BlueHoc [16], an open-source Bluetooth simulator provided by IBM. Considerable extensions of BlueHoc were necessary to carry out the simulation experiments described in this paper.

The remainder of the paper is structured as follows: Section 2 motivates the need for a rendezvous layer in ad-hoc networks in general. Section 3 introduces the Bluetooth inquiry procedure in particular, while section 4 evaluates its performance in terms of time to complete, energy consumption, and scalability. Section 5 discusses how to set the Bluetooth inquiry parameters in order to optimize performance. In section 6 we present an adaptive rendezvous layer protocol that optimizes discovery performance in settings with many devices present. In section 7 several possibilities for the utilization of context information in device discovery are explored. We conclude with a general judgement of the Bluetooth discovery process and give some suggestions for improvements.

## 2 The Rendezvous Layer

In mobile ad hoc environments of smart devices, units initially posses no information about nearby devices, and no centralized instance exists where devices can acquire information about their environment. Therefore, protocols are needed that provide energy-efficient means for detecting new devices and enable peer communications in mobile environments. The rendezvous layer contains such protocols. A rendezvous layer for fixed-frequency systems introduced in [4] provides a mechanism for node discovery using a beaconing approach and implements power saving meachnisms that allow units to be put in sleep modes between communication periods. Our approach to the rendezvous layer is different in that we concentrate on Bluetooth's device discovery and try to minimize power consumption by minimizing the time units have to stay in power-consuming device detection modes. Scheduled rendezvous are not an issue of this paper.

The rendezvous layer enables devices to communicate with each other by helping them to find potential communication partners. The actual data traffic after connection establishment, however, does not flow through the rendezvous layer. The term "layer" might therefore be misleading, but it emphasizes that

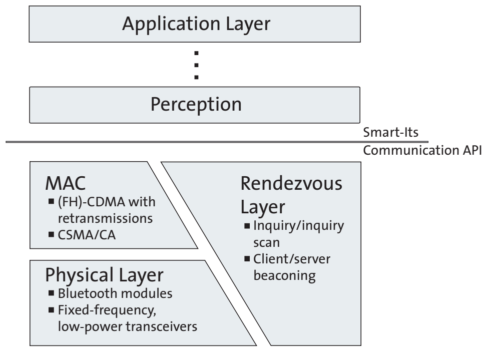


**Fig. 2.** Communication platform architecture for smart devices

the results of rendezvous layer protocols are a precondition for communication and that every mobile node generally needs to use rendezvous protocols to be able to connect to other devices.

Fig. 2 shows a possible communication platform architecture for smart devices. The actual position of the rendezvous layer strongly depends on the concrete design. It is possible that it reaches down to the hardware layer, e.g. when low-power RF detection circuits are used to detect other devices. For fixed-frequency systems, [4] distinguishes between client and server beaconing. In the envisioned ubiquitous computing application scenarios there are no fixed client/server roles. Hence, it seems advantageous to distinguish between dynamically assigned roles such as service provider and service consumer. In general each device acts both as service provider and service consumer.

The rendezvous layer for frequency hopping systems is more complicated than for fixed-frequency solutions. This is mainly due to an initial frequency discrepancy between devices. Frequency hopping systems also result in a much higher energy consumption of the rendezvous layer compared to fixed-frequency systems. In Bluetooth, the rendezvous layer mainly consists of the Inquiry and Inquiry Scan procedures.

## 3 Bluetooth's Inquiry Procedure

The Bluetooth standard introduces an Inquiry procedure for device detection and a Page procedure for connection establishment. Both are asymmetric processes initiated by the unit that wants to collect device information or create a connection. The initiating unit spends significantly more energy than the unit that is inquired or paged, because it stays in Inquiry or Page mode for a long time whereas the other device enters a scanning mode only periodically for short time intervals. The Page and Inquiry procedures resemble each other in that they both have to overcome an initial frequency discrepancy between devices. However, a paging unit already has an estimate of the scanning unit's current clock which it acquired during a preceding inquiry.

During the inquiry process, the unit that wants to find devices in communication range periodically enters the INQUIRY state. Devices that want to advertise their presence and thereby agree to be found by other devices enter the INQUIRY SCAN state regularly. Typically, in the envisioned application scenarios devices enter both states, INQUIRY and INQUIRY SCAN, in certain time intervals. But in order to ensure that two devices find each other, one has to be in INQUIRY and the other in INQUIRY SCAN state simultaneously. To prevent devices from synchronizing their inquiry states, the time between the start of two consecutive inquiries, $T_{inquiry}$, has to be randomly distributed in an interval $[T^{min}_{inquiry}, T^{max}_{inquiry}]$. Fig. 3 and Tab. 1 show the parameters influencing Bluetooth's inquiry procedure.

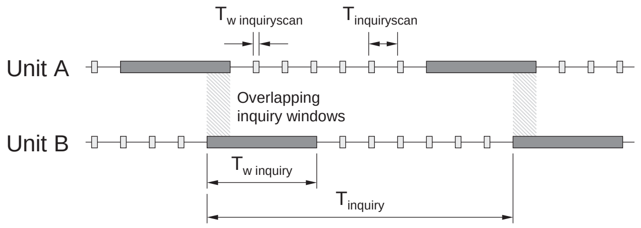


**Fig. 3.** Important inquiry parameters

The device in INQUIRY state broadcasts ID packets on different frequencies at twice the usual hopping rate. That is, it sends two ID packets in a 625 $\mu s$ wide slot, and afterwards listens for $625 \mu s$ for responses from other devices. This is repeated for the duration of the entire inquiry window, $T_{w\,inquiry}$, which is typically in the range of several seconds. There exists a unique inquiry hopping sequence comprising 32 frequencies[1] on which an inquirer sends out ID packets. This sequence is the same for all devices, only the phase within the sequence is determined by the native clock $CLKN$ of the inquiring unit and therefore specific for each device. Furthermore, for each 1.28 $s$ the inquiry hopping sequence is divided into two disjunct, consecutive trains $A$ and $B$, each containing 16 frequencies. The inquirer needs $T_{train} = 10\ ms$ to both send on all frequencies in a single train and check for potential responses. According to the Bluetooth specification [5], the frequencies in "a single train must be repeated for at least $N_{inquiry} = 256$ times before a new train is used". The phase $X_p$ in the inquiry hopping sequence that determines the frequency at which ID packets are transmitted is calculated as follows:

$$X_p = [CLKN_{16-12} + k_{\text{offset}} + (CLKN_{4-2,0} - CLKN_{16-12})\, mod\, 16]\, mod\, 32 \quad (1)$$

[1] This paper concentrates on Bluetooth's 79 hop system because it is applied in the vast majority of countries in the European Union and in the USA. In case of the reduced hop system, the inquiry hopping sequence contains only 16 frequencies.

In equation 1, $CLKN_{x-y,z}$ denotes bits $x$ to $y$ and bit $z$ of the inquiring unit's native clock. $k_{\text{offset}} \in \{24, 8\}$ selects the active train $A$ or $B$ of the inquirer. $k_{\text{offset}}$ is changed after a single train is repeated $N_{inquiry}$ times. The frequencies within each train are shifted by one phase every 1.28 $s$, since after this time $CLKN_{16-12}$ changes. CLKN has a resolution of 312.5 $\mu s$. $(CLKN_{4-2,0} - CLKN_{16-12}) \, mod \, 16$ determines the phase within each train. The expression $CLKN_{16-12}$ is necessary to avoid omitting a frequency when $CLKN_{16-12}$ changes, since this could lead to a repetitive mismatch between inquiring and scanning unit.

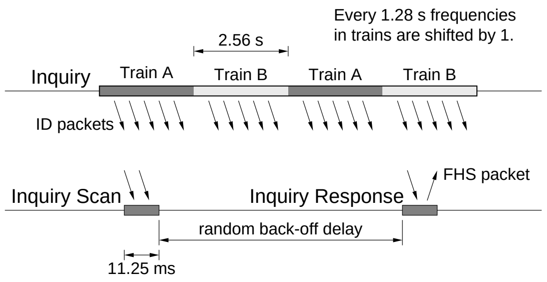


**Fig. 4.** Overview of the inquiry procedure

The device that agrees to be found enters the Inquiry Scan state periodically. The time between two consecutive inquiry scans is determined by the inquiry scan interval $T_{inqscan}$. The inquiry scan window, $T_{w\,inqscan}$, specifies the time a unit stays in Inquiry Scan mode. During that time the unit listens at a single frequency in the inquiry hopping sequence for ID packets from the inquirer. The current phase in the inquiry hopping sequence is determined by its native clock [5]:

$$X_p = CLKN_{16-12}. \tag{2}$$

The Bluetooth standard defines $T_{w\,inqscan} \geq T_{train} = 10 \; ms$ in order to ensure that a frequency synchronization between inquiring and scanning unit takes place when the scanning frequency is in the currently active train of the inquirer. Also, the condition $T_{inqscan} \leq 2.56 \; s$ must hold. When the unit in Inquiry Scan mode receives an ID packet, it leaves the Inquiry Scan mode for a random backoff delay which is evenly distributed between $[0, \ldots, 639.375] \, ms$. This reduces the probability that units simultaneously transmit response packets on the same frequency. Afterwards, the unit enters Inquiry Response state and again listens for ID packets of the inquiring unit. When the unit in Inquiry Response state achieves frequency synchronization, it transmits a packet containing device information such as its current clock timing and its Bluetooth device address to the inquirer.

## 4 Performance of Bluetooth's Inquiry Procedure

A characteristic feature of ubiquitous computing settings is the presence of many highly autonomous, mobile devices with distinctive resource restrictions in a relatively small area. The aspects of scalability, energy consumption, and device detection delay are therefore crucial when evaluating Bluetooth's rendezvous protocols.

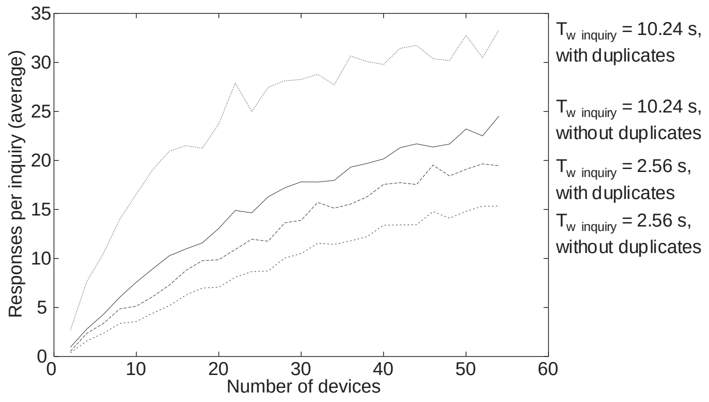


**Fig. 5.** Responses during inquiry subject to the number of devices in range considering different inquiry windows

According to datasheets [14] and experimental measurements [12], the energy consumption in Inquiry and Inquiry Scan state is approximately twice as high as in normal receive and transmit modes. The Bluetooth standard suggests that devices could enter Inquiry mode for 10.24 $s$ every minute. This means that Bluetooth devices would spend approximately 17% of their lifetime in Inquiry mode. Besides an unacceptably high energy consumption, this also leads to poor performance if many devices are present. Above all, in Inquiry mode a Bluetooth unit cannot actively exchange application data with other devices. Fig. 5 shows the average number of responses during inquiry subject to the number of potential communication partners in range. It indicates that in the presence of only a limited number of devices practically all units are found, when $T_{w\,inquiry}$ is chosen as suggested in the standard. It also indicates that performance deteriorates as the number of devices grows, in that a smaller and smaller portion of devices are discovered. The reasons are manifold and are discussed in more detail in the following sections:

- Because of a long random backoff delay, $T_{w\,inqscan}$ must be very large when the scanning unit is supposed to answer more than once during one scan window. Large scan windows are undesirable because they are repeatedly blocking the device.

– Since the relative clock differences of Bluetooth units remain unchanged, a unit tends to answer the same device in consecutive Inquiry Scan intervals (cf. equations 1 and 2). Fig. 5 shows that the overall number of responses is sufficiently high, but the same device often answers the same inquirer in consecutive inquiry scan intervals. This prevents the device from responding to other devices. Only in Inquiry Response state an offset is added to the scanning frequency after each response. However, multiple responses during a single inquiry scan window are only possible when the window is relatively large, which is undesirable in the envisioned application scenarios.
– Large $\frac{T_{w\,inquiry}}{T_{inquiry}}$ values lead to high numbers of overlapping inquiry windows where the devices cannot find each other.

Regarding scalability and energy consumption the rendezvous layer must be designed to support inquiry parameter settings such that

– the overall time a unit has to stay in Inquiry and Inquiry Scan modes is minimized,
– the probability for overlapping Inquiry and Inquiry Scan states is maximized,
– the probability for overlapping Inquiry modes in different units is reduced, and
– the time for the frequency synchronization delay between inquiring and inquired device is as low as possible.

Decreasing the value of $\frac{T_{w\,inquiry}}{T_{inquiry}}$ leads to fewer overlapping inquiry intervals. Since $T_{inquiry}$ cannot be predetermined but generally depends on sensory input and application restrictions, one purpose of the rendezvous layer is to decrease $T_{w\,inquiry}$, the inquiry window. Small $T_{w\,inquiry}$ values reduce the energy consumption and decrease the number of duplicate responses and overlapping inquiry intervals. However, $T_{w\,inquiry}$ cannot be decreased arbitrarily. Fig. 5 shows the average number of devices found during a 2.56 $s$ compared to a 10.24 $s$ inquiry window subject to the number of devices in range. In 2.56 $s$ the inquirer can probe only at a single train, which limits the number of responses. But compared to $T_{winquiry} = 10.24$ $s$, in the presence of many devices fewer duplicate responses are received, much less energy is consumed, and proportionally more devices are found. In settings with many devices, increasing $T_{w\,inquiry}$ will not result in the discovery of significantly more devices, because the devices will block each other. Even with large inquiry windows, in settings with many devices not all devices in range are found.

## 5 Inquiry and Inquiry Scan Settings

Device discovery in Bluetooth is performed in the Inquiry and Inquiry Scan procedures which are controlled by various parameters. These are shown in Fig. 3 and Tab. 1. $T_{inquiry}$ and $T_{inqscan}$ denote the interval between the start of two consecutive inquiries and inquiry scans, respectively. $T_{w\,inquiry}$ and $T_{w\,inqscan}$

specify the duration of a single inquiry and inquiry scan, respectively. $N^{max}_{devices}$ depends on the memory restrictions of a device in that it restricts the number of inquiry responses that are processed. As pointed out before, $T_{inquiry}$ depends on specific applications and sensory input. Therefore, the rendezvous layer does not influence $T_{inquiry}$ and $N^{max}_{devices}$ settings. The train repetition number $N_{inquiry}$ defines the number of times a single train is repeated by the inquirer before a new train is used.

**Table 1.** Inquiry and inquiry scan parameters

| **Parameter** | **Description** |
|---|---|
| $T_{inquiry}$ | inquiry interval |
| $T_{w\,inquiry}$ | inquiry window, $T_{w\,inquiry} \leq T_{inquiry}$ |
| $T_{inqscan}$ | inquiry scan interval, $T_{inqscan} \leq 2.56\ s$ [5] |
| $T_{w\,inqscan}$ | inquiry scan window, $T_{train} = 10\ ms^2 \leq T_{w\,inqscan} \leq T_{inqscan}$ |
| $N^{max}_{devices}$ | maximum number of responses processed in a single inquiry |
| $N_{inquiry}$ | train repetition number, $N_{inquiry} \geq 256$ (predefined, fixed) |

## 5.1 The Inquiry Scan Window

A suitable value for the inquiry scan window is the minimal setting $T_{w\,inqscan} = T_{train} = 10\ ms^2$. Here, $T_{train}$ is the time period for the inquirer to send at all $N_{train} = 16$ frequencies in the active train. $T_{w\,inqscan}$ should only be increased when $N_{inquiry}$ is noticeable smaller than 256 (which is not the case in Bluetooth), because the inquirer consecutively sends on more than $N_{train}$ frequencies only when it switches between different trains. This happens just every $T_{train} * N_{inquiry}$ seconds and would not justify the additional time a unit would have to spend in INQUIRY SCAN mode.

In an error-free environment, $T_{w\,inqscan} = T_{train}$ seems to be the best choice. However, in ubiquitous computing application scenarios where the probability of packet loss is relatively high, we suggest to choose $T_{w\,inqscan} = 2{*}T_{train} = 20\ ms$ to ensure that an inquirer can send ID packets at each frequency in its active train twice. If the ID packet that was sent at the scanning frequency gets lost, the inquirer can send it again at this frequency. Fig. 6 and 7 show the average number of devices found during inquiry considering inquiry scan windows of varying length in an error-free environment. Noticeably more devices are only found when $T_{w\,inqscan}$ is substantially increased, because in this case the probability that a train switch takes place during scanning is significantly higher. However, since all connections have to be suspended during scanning, substantially increasing $T_{w\,inqscan}$ is not recommendable. Instead, as Fig. 6 and 7 suggest, decreasing the

[2] The definition of the Write_Inquiry_Scan_Activity HCI command says that $T_{w\,inqscan} \geq 11.25\ ms$.

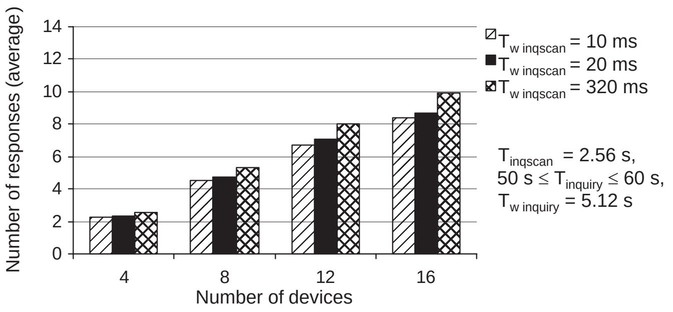


**Fig. 6.** Average number of devices found during inquiry subject to $T_{w\,inqscan}$ and number of devices in range, $T_{inqscan} = 2.56\ s$

inquiry scan interval is much more effective regarding both energy consumption and the number of devices found. Increasing $T_{w\,inqscan}$ by a factor of 32 from 10 ms to 320 ms is not as effective as lowering $T_{inqscan}$ from 2.56 s to 1 s regarding the number of devices that are found during inquiry and the energy consumed.

### 5.2 The Inquiry Scan Interval

The inquiry scan interval, $T_{inqscan}$, denotes the time between the start of two consecutive inquiry scans. The condition $T_{inqscan} \leq 2.56\ s$ must hold. $T_{inqscan}$ is chosen such that the overall energy consumption of the whole system of participating nodes is reduced. Consequently, the scan interval can be shortened when in return the inquiry window of other devices can be reduced. Since every unit generally attains both inquiry and inquiry scan modes regularly, this is beneficial for the whole system of smart devices as well as for single units. A module in continuous INQUIRY mode consumes significantly more energy than in periodic INQUIRY SCAN mode when the inquiry scan window is sufficiently small as suggested in section 5.1.

$T_{inqscan}$ can be used to control the accessibility of single devices. A short inquiry scan interval entails that the device can easily be found by other devices. On the other hand, a low value of $T_{inqscan}$ also means that the device might respond more often to the same device during consecutive inquiry windows.

In order to decrease the time a unit has to stay in continuous INQUIRY mode it is desirable that the first and second frequency synchronization before and after the random backoff delay (cf. Fig. 4) take place before a train switch in the inquiring unit occurs. Since a train switch takes place every $T_{train} * N_{inquiry}$ seconds, a first approximation for a suitable $T_{inqscan}$ is

$$T_{inqscan} \leq T_{train} * N_{inquiry} - RB_{max} - T_{train} \tag{3}$$

Here, $RB_{max} = 639.375\ ms$ is the maximum random backoff delay. For the standard settings and $N_{inquiry} = 256$ this results to $T_{inqscan} \leq 1910.625\ ms$.

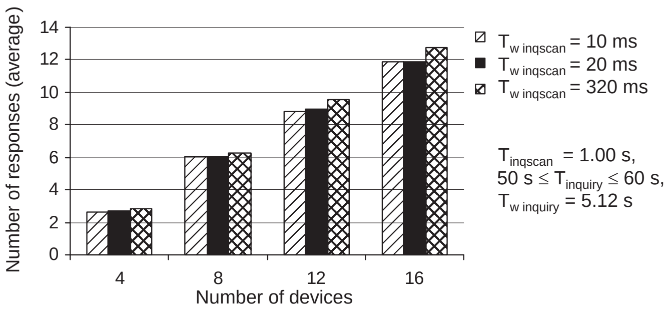


**Fig. 7.** Average number of devices found during inquiry considering a different value for $T_{inqscan}$ (1.00 $s$) compared to Fig. 6 (2.56 $s$)

The above settings ensure that when there are only two devices and one of them enters Inquiry mode, the inquirer has to stay in inquiry mode for only 5.12 $s$ (instead of 10.24 $s$) to find the other device with high probability. Especially in the presence of many devices it is worthwhile to decrease $T_{inqscan}$ further. A lower value than 1910.625 $ms$ leads to a higher energy consumption for scanning. But on the other hand, $T_{w\,inquiry}$ can be reduced to $N_{inquiry} * T_{train} + T_{inqscan} + RB_{max} + T_{train}$. Furthermore, a device with a short inquiry scan interval can respond to other devices more frequently.

## 5.3 The Inquiry Window

The inquiry window, $T_{w\,inquiry}$, denotes the time a unit continuously stays in Inquiry mode. Since Inquiry is a mode with very high energy consumption, $T_{w\,inquiry}$ should be as low as possible. In settings with a low number of Bluetooth devices, a unit might prolong the inquiry window until no new devices are found for a certain amount of time. However, as shown before (cf. section 4) in environments with a large number of devices, prolonged inquiry windows make the rendezvous layer inefficient because of overlapping inquiry windows, high energy consumption, and decreased accessibility of inquiring devices. Furthermore, even in settings with large inquiry windows it cannot be assured that all potential communication partners are found.

When equation 3 holds for all devices $d \in D$ in communication range, a good lower bound for the inquiry window parameter would be $T_{w\,inquiry} = \max_{d \in D}\{(\lfloor \frac{T_{train} * N_{inquiry}}{T_{inqscan}(d)} \rfloor + 1) * T_{inqscan}(d)\} + RB_{max}$, where $D$ is the set of devices in range and $T_{inqscan}(d)$ the inquiry scan interval of device $d$. This setting ensures that without overlapping inquiry windows and only a few devices in range all potential communication partners can be found with high probability.

A generally appropriate setting for $T_{w\,inquiry}$ is $T_{w\,inquiry} = 2 * N_{inquiry} * T_{train} = 5.12\ s$, when $T_{inqscan}$ and $T_{w\,inqscan}$ are selected as recommended in the previous sections. This enables the inquiring device to probe at frequencies in both trains for an equal amount of time and provides sufficient time to select responses. On the other hand the number of duplicate responses is relatively low and the energy consumption is much lower than for the suggested 10.24 $s$ in the Bluetooth standard.

However, $T_{w\,inquiry} = 5.12\ s$ might be a suboptimal choice for environments with only few devices and is still very energy consuming. The next section deals with an adaptive protocol for Bluetooth-enabled smart devices that performs well independently of the number of devices present and further reduces $T_{w\,inquiry}$ to save energy.

## 6 An adaptive rendezvous layer protocol for cooperative device discovery

The performance of the standard Bluetooth inquiry procedure is sufficient for settings with a limited number of devices in communication range. But the performance decreases significantly with a rising number of units. In such environments only a fraction of the potential communication partners are found – even when $T_{w\,inquiry}$ is high. Unfortunately, large values for $T_{w\,inquiry}$ result in many duplicate responses, overlapping inquiry windows, and poor overall performance (cf. section 4).

In settings with many devices, it is more appropriate to discover devices in a cooperative fashion. Cooperative device discovery splits up the task of finding communication partners between multiple units. One idea is to let only one or two units per piconet [5] handle rendezvous tasks on behalf of the whole piconet; another is to utilize inquiry results of other devices that responded during inquiry. The goal of such measures is to reduce the overall number of devices that take part in inquiry and the overall time units have to stay in INQUIRY mode in order to discover more devices in less time using less energy.

In the adaptive protocol proposed here, a unit starts inquiry for a certain time window and accumulates responses from other devices. Although units do not know how many devices are in range, they can estimate their number considering the number of devices that responded during the first seconds of an inquiry. When, after a given time interval, more devices than a predetermined threshold were discovered, it concludes that many devices are in range, stops the inquiry, builds up connections to some of those devices, and gets further discovery information from them. By selecting devices with appropriate clock values, this can be done in such a way that a large subset of the available devices is covered, as explained below.

The advantage of this approach is that it splits up the responsibility for inquiry between different nodes and, more importantly, that the time interval after which inquiry is canceled when a sufficient number of responses are accumulated can be very short. In fact, we suggest a value of only 2.56 $s$. During this interval an inquirer only inquires at frequencies in a single train since $T_{train} * N_{inquiry} \geq 2.56\ s$ (cf. section 3). That is, in the average case only 50% of devices in range can be found during the first phase of the protocol. However, the timing information transferred during inquiry responses enables the original inquirer to identify devices that during their inquiries discover a subset of devices not found by direct inquiry. It might seem that connecting to another device consumes the energy saved by a shorter inquiry for the paging process – which is also very energy intensive. But since the timing information of this device were transferred during a recent inquiry response, connection establishment is almost instantaneous. The simulation results show that the overall execution time for the adaptive protocol is only slightly longer than the interval after which the actual inquiry is stopped.

In the following, the algorithm for the inquirer is depicted. The inquiry scan settings in participating Bluetooth units should be chosen as described in section 5. The protocol can be implemented on top of Bluetooth's Host Controller Interface (HCI) without changing lower layers of Bluetooth. An initialization for all Bluetooth-enabled smart devices should include enabling page and inquiry scans (HCI_Write_Scan_Enable) and setting the inquiry and page parameters (HCI_Write_Inquiry_Scan_Activity, HCI_Write_Page_Scan_Activity). Furthermore, the page timeout should be chosen as low as possible in order to prevent a device from paging a unit that left its communication range for a long time (HCI_Write_Page_Timeout). After sending inquiry responses, units should enter Page Scan state to ensure fast connection establishment.

```
**BRLP (Bluetooth Rendezvous Layer Protocol)**
**Input:**
    Inquiry settings for inquirer: $T_{inquiry} \in [T^{min}_{inquiry}, T^{max}_{inquiry}]$, $T_{w\,inquiry}$, $N^{max}_{devices}$
    Time interval for normal inquiry: BRLP_timeout
    Threshold for device responses: BRLP_size
    Number of devices to retrieve discovery information from: $N_{select}$
**Output:**
    Bluetooth device addresses and clock settings of devices

ensure($T^{min}_{inquiry} > T_{w\,inquiry}$)
inqtimer = random($T^{min}_{inquiry}$, $T^{max}_{inquiry}$)
do forever
    if time_over(inqtimer) then
        responses.delete()
        HCI_Inquiry(ALL_DEVICES, $T_{w\,inquiry}$, $N^{max}_{devices}$)
        inqtimer = random($T^{min}_{inquiry}$, $T^{max}_{inquiry}$)
        BRLP_timer = BRLP_timeout
    end if
end do

inquiry_response_event_handler(Inquiry_Response_Event e)
begin
    response = e.getResponse()
    responses.add(response)
        if not time_over(BRLP_timer) and responses.size > BRLP_size then
```

```
            HCI_Inquiry_Cancel()
            selected_responses = responses.select($N_{select}$)
            **for all** sr **in** selected_responses **do**
                HCI_Create_Connection(sr)[3]
            **end for**
        **end if**
**end**

connection_complete_event_handler(Connection_Complete_Event e)
**begin**
    get_assembled_devices(e.connection_handle);
**end**
```

The suggested protocol decreases the level of confidence in the obtained results, because they are partially retrieved indirectly from other devices and might refer to units outside the communication range. This is not a severe problem, however, because direct results might also be inaccurate – e.g. obsolete because of mobility – and the algorithm has to deal with uncertain results anyway. Section 7 shows how sensory input can be used to decrease this uncertainty. Also, in order to inhibit error propagation, a unit is only allowed to pass on discovery information that it learned from its own most recent inquiry procedure. A low value for BRLP_size means that only a few inquiry results are available to be transferred to other devices. This entails that this parameter should be adapted after each inquiry process. A variation of the described protocol is to carry out normal inquiry for a certain amount of time (for example 2.56 s) regardless of the number of devices found during this inquiry window. When more than a given threshold of units have been found after this period, connections to some of these devices are established and discovery information is requested.

To clarify the performance of the adaptive part of the algorithm, Fig. 8 shows the average number of devices found considering a very low value for BRLP_size. It considers only the cases in which after 2.56 s inquiry connections to other devices are established to request discovery information. From the set of inquiry results two devices are selected to retrieve further discovery information from ($N_{selected} = 2$). The selection criteria are explained below. The average total time for the initial 2.56 s inquiry, connection establishment, and transfer of discovery information from the two selected devices was 3.44 s. Compared to normal inquiry with $T_{winquiry} = 10.24\ s$ a better performance regarding the number of responses is achieved, and the time a unit has to stay in inquiry mode is reduced significantly. Therefore, the adaptive protocol results in substantial energy savings, enables units to enter energy-saving modes more frequently, and leaves more time for application specific tasks.

The selection of units to retrieve discovery information from is important for the performance of the adaptive algorithm. The retrieved discovery information

[3] See the definition of HCI_Create_Connection for the exact sequence of parameters.

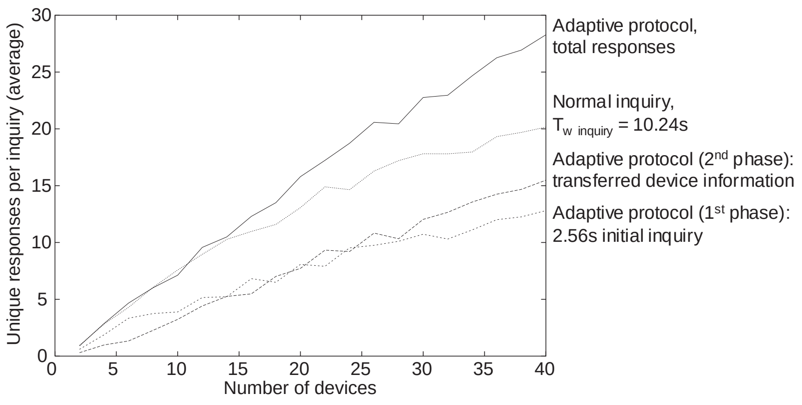
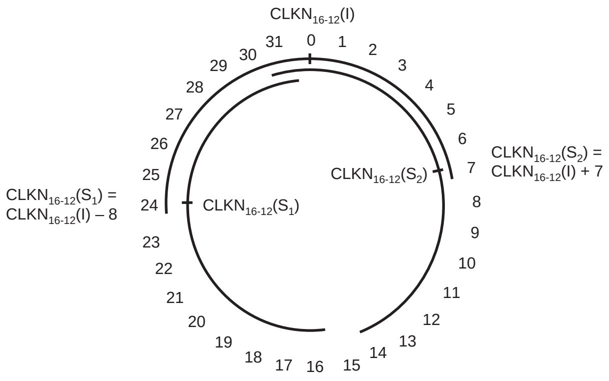


**Fig. 8.** Average number of discovery responses accumulated with the adaptive protocol ($N_{selected} = 2$)

is only useful if it contains inquiry results from devices not already found by direct inquiry. The probability for this is highest, when devices with appropriate clock offsets are selected. The clock offsets are part of the inquiry results.



$CLKN_{16-12}(I)$
$CLKN_{16-12}(S_2) = CLKN_{16-12}(I) + 7$
$CLKN_{16-12}(S_1) = CLKN_{16-12}(I) - 8$
$CLKN_{16-12}(S_2)$
$CLKN_{16-12}(S_1)$

**Fig. 9.** Phases covered by different units relative to $CLKN^{I}_{16-12}$ during inquiry in train A

When the timeout for the initial inquiry, BRLP_timeout, is lower than or equal to $T_{train} * N_{inquiry} = 2.56\ s$ – which is desirable regarding energy consumption – only the frequencies of a single train are inquired. Let $I$ be the inquiring and $S$ a scanning unit that responded to $I$ during the initial inquiry window. $CLKN^{I}_{16-12}$ and $CLKN^{S}_{16-12}$ shall be bits 12 to 16 of the native clock of $I$ and $S$, respectively. The frequencies at which $I$ inquires and $S$ scans only depend on $CLKN^{I}_{16-12}$ and $CLKN^{S}_{16-12}$ (cf. equations 1 and 2). Let the first active train during inquiry be train A. Then, the phases of the frequencies in the active train are $[CLKN^I_{16-12} - 8, \ldots, CLKN^I_{16-12} + 7]$ (mod 32). When $CLKN^S_{16-12} \in [CLKN^I_{16-12} - 8, \ldots, CLKN^I_{16-12} + 7]$ during one inquiry scan interval this will also be the case during all successive inquiry scan intervals due to constant clock differences. This is important: because of given constant clock differences it does not matter when a device enters inquiry state. It will always find the same devices scanning at frequencies in the same train, because the frequencies in a train also depend on $CLKN_{16-12}$. This implies that $I$ should select a device $S$ to obtain discovery information from, such that $|CLKN^I_{16-12} - CLKN^S_{16-12}|$ is maximal.

Fig. 9 illustrates this in the light of a concrete example. The semicircles show the frequency phases covered by units $I$, $S_1$, and $S_2$ during inquiry in train A relative to $CLKN^I_{16-12}$. The inquiry of $I$ results in the discovery of two units, $S_1$ and $S_2$. The clock offset of $S_1$ is $CLKN^{S_1}_{16-12} - CLKN^I_{16-12} = -8$; that of $S_2$ is $CLKN^{S_2}_{16-12} - CLKN^I_{16-12} = 7$. These relative clock differences remain constant over a longer period of time, although the $CLKN_{16-12}$ change every 1.28 s, possibly at different times, rotating the halfcircles right. Fig. 9 shows that $S_1$ and $S_2$ in their inquiries cover frequency phases relative to $CLKN^I_{16-12}$ that are not traversed by $I$. Units scanning at these phases with a relative distance greater than 7 or less than $-8$ are never found by $I$, regardless of the current value of $CLKN_{16-12}$. Since the clock offsets of $S_1$ and $S_2$ are maximal, they cover the maximal number of phases relative to $CLKN^I_{16-12}$ not traversed by $I$. By transitively selecting devices from the discovery information of $S_1$ and $S_2$ it is possible for $I$ to choose devices that have optimal clock offsets in order to cover a large area of phases. In the same way that a device $S_{opt}$ is an optimal choice for $I$, $I$ vice versa is an optimal choice for $S_{opt}$ – the relationship is symmetric. Therefore in a setting with a large number of devices present, a few devices can form stable subgroups to cooperatively perform device discovery. They complement each other, and the whole system as well as individual units profit in terms of energy savings, number of devices discovered, and shorter discovery delays.

## 7 Using sensory input to improve rendezvous-layer protocol performance

When everyday items are augmented with information processing capabilities they will provide information about their environment to other devices, thus enabling collaborative perception of the environment. In the Smart-Its project, smart devices are equipped with a wide variety of different sensors for physical parameters like temperature, acceleration, pressure, etc. An interesting question is how sensory data that is accumulated independently from the communication platform can be used to improve rendezvous layer protocol performance. The idea to take advantage of context information – especially location – in communication protocols has already been applied to other protocol layers, e.g. routing protocols [8].

## 7.1 Context for Adapting Inquiry Parameters

If sensory input from acceleration or general location sensors are available, the inquiry parameter settings can be adapted when a device moves. Since it is more probable that a moving device enters a new environment, the inquiry window is prolonged or the inquiry interval is shortened in order to discover these devices more quickly. In this case only individual devices increase their inquiry window; this does not lead to a deterioration of the rendezvous layer protocol performance of the system as a whole. Alternatively, the inquiry scan interval is reduced to ensure that other units can access the device faster. In general, all sensory input that hints at an increased usage of the device in the future could result in such an adjustment of inquiry parameter settings.

Context information can also be used to implement the *select* routine in the adaptive rendezvous layer protocol. The purpose of the *select* statement is to choose such devices among all units that responded during inquiry, for which the uncertainty of transferring obsolete device information is lowest. When no sensory input is available, devices are selected as shown in the previous section or randomly from the set of devices that already responded during inquiry. But when context information is available, then it should be used to select devices that are as near as possible, that inquired at different frequencies, and that started inquiry most recently. In the Bluetooth 1.1 standard there exists a link manager protocol (LMP) command to determine the signal strength to other Bluetooth devices. This information could be used to evaluate whether the device is suitable to get inquiry information from.

## 7.2 Restricting Device Discovery to Symbolic Locations – Smart Doors

One of the main differences between typical pervasive computing settings and traditional distributed systems is that many smart devices such as the Bluetooth-enabled Smart-Its do not possess conventional input and output capabilities such as buttons, keyboards, or screens. Instead, sensors attached to smart devices, perceived sensory input, and context information derived collaboratively among different objects determine whether, when, and how entities are going to communicate with each other.

Based on the Bluetooth-enabled Smart-Its (cf. Fig. 1) we have built several context-aware applications in which history information as well as the context of devices and users determine whether two entities are going to exchange information with each other [19]. A critical property of these applications is that communication is often restricted to a certain symbolic location shared by several smart objects. For example, smart objects in a certain room often only need to communicate with other Bluetooth-enabled mobile devices that share the same symbolic location. As radio waves penetrate walls, the conventional Bluetooth inquiry procedure is not suitable to distinguish between devices in different rooms. Furthermore, as shown in the previous sections, especially in the presence of many devices, the Bluetooth inquiry procedure can become too slow for ad hoc interaction.

Therefore, we have developed the concept of *smart doors* that (1) determine whether two Bluetooth-enabled devices share a certain symbolic location and (2) considerably reduce the time in which new devices are found. Smart doors are appliances that improve the rendezvous layer protocol performance of Bluetooth-enabled smart devices by using passive RFID technology. Passive RFID labels are attached to users' mobile devices such as mobile phones and PDAs. RFID tags contain the access parameters of devices, i.e. information about how these devices can be reached. For example, an RFID tag attached to a Bluetooth-enabled mobile phone contains the phone's Bluetooth device address and the its phone number. An RFID reader attached to a Smart-Its, which is mounted to the entrance of an office (cf. Fig. 10) serves as the main sensor. When a user enters a room, the RFID tag attached to the mobile device is scanned by the RFID reader and the corresponding RFID data is processed by a Smart-Its. Smart objects know to what room they belong because the Smart-Its at the door connects to devices entering the room and transmits the current location information. Devices in the room have typical pull and push options: they can connect to the smart door each time they are interested in devices sharing their symbolic location. Alternatively, they can be notified by the smart door whenever a new device enters a room. The latter alternative is very efficient when subscribed devices belong to the piconet of the smart door as the information can then be transferred via Bluetooth piconet broadcast.

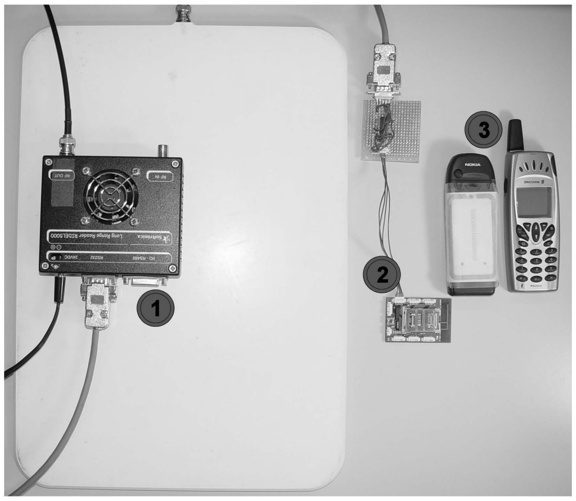


**Fig. 10.** Smart doors are equipped with an RFID reader and antenna (1) as well as a Bluetooth-enabled Smart-Its [18] (2). The RFID tags attached to Bluetooth-enabled mobile phones (3) are read out each time a users enters a room, and the RFID information is transmitted via Bluetooth broadcast to other smart devices in the room.

The main advantages of the smart door approach are that it (1) reduces the amount of interference by restricting communication to devices sharing the same symbolic location and (2) minimizes the time mobile, battery-powered devices have to perform the energy-consuming task of conventional Bluetooth inquiries.

## 8 Conclusion

The standard inquiry procedure of Bluetooth consumes much energy which is problematic for communicating smart devices in ubiquitous computing settings. This paper showed how the standard inquiry parameters can be adapted to decrease power consumption and increase the scalability of the inquiry procedure. Since typically smart devices perceive their environment through sensors, we also presented ways for using sensory input to improve the performance of Bluetooth's rendezvous layer.

Furthermore, it was pointed out that the scalability of Bluetooth's inquiry procedure is not sufficient if many devices are present. As a result from this observation, an adaptive protocol for cooperative device detection was introduced that reduces energy consumption and improves scalability for environments with many devices.

The properties of Bluetooth's rendezvous layer that have the strongest impact on device detection delay are a relatively high random backoff delay and the existence of two separate frequency trains. In terms of the rendezvous layer for general frequency hopping systems there should be only a single train comprising all 32 frequencies of the inquiry hopping sequence. Alternatively, $N_{inquiry}$ could be reduced to one, and the minimum inquiry scan window should be increased to 20 ms. It is important to note that even with these adaptations the majority of parameter selection rules and the adaptive rendezvous layer protocol presented in this paper are still applicable.

## References

1. Mattisson, Sven: *Low-Power Considerations in the Design of Bluetooth.* International Symposium on Low Power Electronics and Design, ISLPED 2000, Rapallo, Italy, 2000.
2. Salonidis, T.; Bhagwat, P.; Tassiulas, L.; LaMaire, R.: *Distributed topology construction of Bluetooth personal area networks.* In: Proceedings of INFOCOM 2001, IEEE, Volume: 3, 2001, pp. 1577-1586.
3. Salonidis, T.; Bhagwat, P.; Tassiulas, L.: *Proximity awareness and fast connection establishment in Bluetooth.* First Annual Workshop on Mobile and Ad Hoc Networking and Computing, MobiHOC, 2000.
4. Girling, Gray; Wa, Jennifer Li Kam; Osborn, Paul; Stefanova, Radina: *The Design and Implementation of a Low Power Ad Hoc Protocol Stack.* In: IEEE Personal Communications, Vol. 4, No. 5, pp. 8-15, October 1997.
5. Bluetooth Special Interest Group: *Specification of the Bluetooth System, Core.* Version 1.1, February 2001.

6. Haartsen, Jaap C.: *The Bluetooth Radio System.* In: IEEE Personal Communications, February 2000.
7. Gomie, N.; van Dyck, R. E.; Soltanian, A.: *Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation.* ACM MSWiM 2001, The Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems 2001, July 2001, Rome, Italy, 2001.
8. Ko, Young-Bae; Vaidya, Nitin H.: *Location-Aided Routing (LAR) in Mobile Ad Hoc Networks.* MOBICOM 98, Dallas, Texas, USA, 1998.
9. Feeney, Laura Marie; Nilsson, Martin: *Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment.* IEEE INFOCOM 2001, Anchorage, Alaska, April 2001.
10. Mattern, Friedemann: *The Vision and Technical Foundations of Ubiquitous Computing.* In: Upgrade European Online Magazine, pp. 5-8, October 2001.
11. Weiser, Mark: *The Computer for the Twenty-First Century.* In: Scientific American, pp. 94-100, September 1991.
12. Kasten, Oliver; Langheinrich, Marc: *First Experiences with Bluetooth in the Smart-Its Distributed Sensor Network.* Workshop on Ubiquitous Computing and Communications. In: Proceedings PACT 2001, October 2001.
13. Moser, Thomas; Karrer, Lukas: *The EventCollector Concept, Distributed Infrastructure for Event Generation & Dissemination in Ad Hoc Networks.* Diploma thesis, ETH Zurich, March 2001.
14. Ericsson Microelectronics: *ROK 101 007 Bluetooth Module Datasheet Rev. PA5.* April 2000.
15. The Network Simulator - ns-2: *http://www.isi.edu/nsnam/ns/.*
16. BlueHoc, An Open-Source Bluetooth Simulator:
*http://oss.software.ibm.com/developerworks/opensource/bluehoc/.*
17. The Smart-Its Project: *http://www.smart-its.org.*
18. Beutel, Jan; Kasten, Oliver: *A Minimal Bluetooth-Based Computing and Communication Platform.* Technical Note, May 2001.
19. Siegemund, Frank; Flörkemeier, Christian: *Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones.* To be published in the proceedings of PerCom 2003 (IEEE International Conference on Pervasive Computing and Communications), March 2003.