First Experiences with Bluetooth in the *Smart-Its* Distributed Sensor Network*

Oliver Kasten, Marc Langheinrich
Swiss Federal Institute of Technology
Distributed Systems Group
8092 Zurich, Switzerland
{kasten|langhein}@inf.ethz.ch

Abstract

Future ubiquitous computing devices will need to communicate with other smart devices in an ad hoc fashion, using minimal power and possibly without the help of a central infrastructure. However, so far no single communication technology has established itself in the field of ubiquitous computing: many current wireless communication technologies seem to lack robustness, consume too much energy, or require an infrastructure to be viable candidates. In order to evaluate the suitability of the new and promising Bluetooth standard for such communication requirements, we have integrated a Bluetooth module into the first prototype of a distributed sensor network node, developed within the European Smart-Its *research project. While Bluetooth offers robust and convenient ad hoc communication, preliminary experiments suggest that the Bluetooth standard could benefit from improved support for symmetric communication establishment and slave-to-slave communication.*

1 Introduction

In the vision of ubiquitous computing, every-day objects are augmented with computation and communication capabilities in order to make them "smart." While such artifacts retain their original use and appearance, their augmentation can seamlessly enhance and extend their usage, opening up novel interaction patterns and applications.

The goal of the Smart-Its project [12] is to add smartness to real-world objects in a *post-hoc* fashion by attaching small, unobtrusive computing devices, so-called Smart-Its, to them. While a single Smart-It is able to perceive context information from its integrated sensors, a federation of ad hoc connected Smart-Its can gain *collective awareness* by sharing this information.

Sharing information requires a suitable communication technology, which preferrably should be wireless in order to be in line with the unobtrusive nature of the devices. Since there is no central authority in a Smart-Its sensor network, nodes within the network must also be able to communicate in an ad hoc fashion, i.e., without a priori knowledge of each other, and without the help of a background infrastructure (though they may utilize services when available). Moreover, the communication technology must be robust, scale well, and must efficiently use the limited energy of the autonomous device. Finally, the communication technology employed should adhere to a broadly-used standard to leverage from existing communication services in the environment. These needs have prompted a search for a suitable communication technology for the Smart-Its sensor network. After a brief survey of the existing technology, we decided to take a closer look at the emerging Bluetooth standard as a potential candidate.

Bluetooth [6] is an emerging communication standard that provides ad hoc configuration of master/slave piconets including eight active units at most. It supports spontaneous connections between devices without requiring a priori knowledge of each other. Bluetooth allows data transfers between units over distances of nominally up to 10 meters. The gross data rates of 1 Mbps is shared among all

*The Smart-Its project is partly funded by the European Commission (contract No IST-2000-25428) and the Swiss Federal Office for Education and Science (BBW No 00.0281).

participants of a piconet. Bluetooth operates in the license-free 2.4 GHz ISM spectrum (2.400–2.484 GHz) and uses frequency hopping spread spectrum (FHSS) to minimize interference problems. The technology is geared toward low energy consumption, and targets the consumer mass market with world-wide availability and low price.

In order to explore the practical use of this communication technology, and to evaluate its suitability for ubiquitous computing in general and the Smart-Its project in particular, we have built a small number of Smart-Its prototypes with Bluetooth communications. While the functional prototype allowed us to get a good first look at using Bluetooth in custom-built devices, the lack of support for many of the core features in our pre-series Bluetooth modules did not allow us to arrive at final results regarding its suitability. Instead, we are planning to conduct further experiments in the future, preferrably with more mature modules from a number of different vendors.

Section 2 below describes our first Smart-Its prototypes and lists its design considerations and technology options. It also gives an overview of the board layout and its system software. In section 3 we then discuss the Bluetooth standard in view of its projected use in the project, and report the initial experiences gained with our prototypes. Section 4 introduces related work and briefly describes other possible communication technologies, while section 5 summarizes our experiences and outlines possible future work.

2 The Smart-Its Prototype

The goal of the Smart-Its project [12] is to add "smartness" in a *post-hoc* fashion. It aims at embedding computation into real-world objects by attaching small, unobtrusive, and autonomous computing devices to them. These devices, the *Smart-Its*, integrate sensing, processing, and communication capabilities, which can be customized to the objects they are attached to.

While a single Smart-It is able to perceive context information from the integrated sensors, a federation of ad hoc connected Smart-Its can gain *collective awareness* by sharing this information. A federation of Smart-Its-augmented objects can thus establish a common context that can be exploited by applications and services located in the environment.

Application scenarios of collective awareness of Smart-Its have been described in [10], for example an anti credit-card theft mode where a Smart-It-enabled credit card only functions if a sufficient number of Smart-It enabled personal artifacts such as clothes or car keys are around, rendering the card useless when lost or stolen.

The next sections will outline our requirements for a Smart-It node, describe the components integrated into our first prototype, and give an overview of the device's circuit board and system software.

2.1 Requirements

At the outset, the design of the first prototype should fulfill two main requirements: the device should be easy to use, program, and debug. At the same time, it should be small enough to serve as a demonstrator for a Smart-It sensor node, but large enough for easy handling.

In order to facilitate rapid prototyping, the first Smart-It unit should implement a limited functional core only. That is, it should include a processor, memory and, of course, Bluetooth communications. We decided to do without integrated sensing capabilities, but instead settled on providing a rather versatile external interface with analog and digital IO, allowing us to connect single sensors or a even a daughter board for sensing. This gives us not only the option to easily integrate commercially available sensors having very diverse interfaces (ranging from simple analog output over serial interfaces to bus systems, such as I^2C), but also leave us with more time to identify appropriate sensors and sensing algorithms for use within a Smart-Its environment. In addition we wanted an RS232 serial port, mainly for debugging purposes.

Keeping in line with the unobtrusive nature of the ubiquitous computing paradigm, the Smart-Its devices should be able to operate autonomously for extended periods of time. This includes consciously choosing the components regarding their power consumption, as well as providing a suitable power source. For easy handling, we decided to run the device from an externally attached rechargeable battery pack, rather than having a battery housing mounted on the device. This way we can have small batteries attached for normal operation but still operate the device using bulky but more powerful batteries for extended testing.

For easy layout of the circuit board, we opted for a single voltage plane and an overall low component count. Finally, the system should be in-circuit programmable in

order to minimize turn-around times.

2.2 Components

Commercial Bluetooth solutions are available as self-contained transceiver modules. They are shielded subsystems designed to be used as add-on peripherals. They feature an embedded CPU, different types of memory, as well as baseband and radio circuits. The modules offer a generic Host Controller Interface (HCI) to the lower layers of the Bluetooth protocol stack, while the higher layers of the protocol, as well as applications, must be implemented on the host system. Since the in-system CPU and memory are not available for installing user specific implementations, even a minimal standalone Bluetooth node thus needs an additional host CPU to execute applications and the corresponding higher layers of the Bluetooth protocol. Transport layers for communication between the Bluetooth module and the host system are standardized for UART, RS232, and USB. The only units available at the time (January 2001) were engineering samples of the Ericsson ROK 101 007 module [8].

To run the higher Bluetooth protocol layers and applications, we chose the Atmel ATMega103L microcontroller [4] as our host CPU. The unit is in-system programmable, and features an 8-bit RISC core with up to 4 MIPS at 4 MHz, a serial UART as well as several power modes. The embedded memory consists of 128 Kbytes Flash memory and 4 Kbytes of internal SRAM. The data memory can be extended up to 64 Kbytes, requiring only two external components, the SRAM and an address latch. The external memory is directly addressable by the 16-bit data-memory address bus, i.e., without paging. Even though a less powerful processor with less memory could have potentially delivered sensor data to the bluetooth module as well, we decided to use a more powerful system in order to allow more complex on-board pre-processing.

2.3 Device overview

Figure 1 shows a Smart-Its prototype on top of a battery pack (mostly hiddent behind the board) used for testing and evaluation. All components are mounted onto a 4 x 6 cm two-layer printed circuit board. The unit has several external interfaces. A serial UART port (at the top) is available for data transfer and debugging at speeds up to 57.6 Kbps. There are two 8-bit general-purpose I/O ports, eight 10-bit

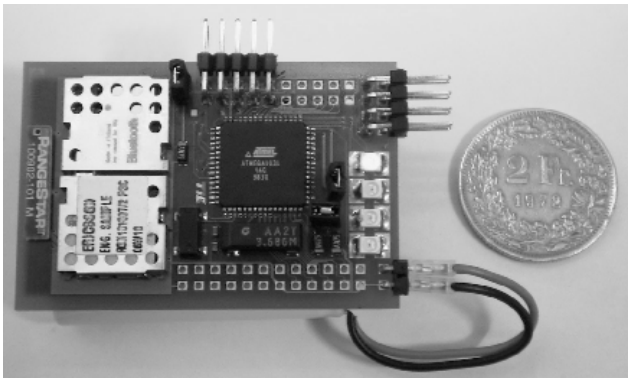

Figure 1: Smart-Its prototype

analog to digital converters, and two edge or level triggered interrupt lines to interface external sensors or other components (none of the interface pins are connected in the picture). Four LEDs (on the right) can be used for debugging and status information. For example, we use one LED to flash a heartbeat signal when the unit is operational. A voltage regulator is used to supply the necessary operating voltage of 3.3V from the battery pack.

Jumpers providing access to each of the main component's individual power-supply lines allows for exact monitoring of power consumption and duty cycles. The connector on the upper right is the in-circuit system programming interface (SPI) of the MCU. The Bluetooth module and an external 2.4 GHz antenna are mounted on top of a ground

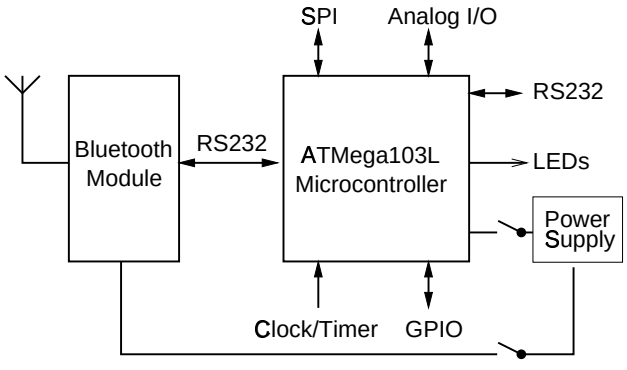

Figure 2: System overview

plane (on the left), to shield the system from RF interference.

The Bluetooth module is attached to the microcontroller unit by an UART implemented in software (as shown in Figure 2). We decided against using the hardware UART provided by the microcontroller, as that would have required additional circuitry for multiplexing pins shared between the UART and the in-circuit programming ports. Therefore we implemented a second software UART in C. Timing constraints prohibit data transfers exceeding 9.6 Kbps, effectively limiting the gross data rate of Bluetooth. In this first design we valued low component count and low circuit complexity over higher data rates. This will change in future designs.

2.4 System software

The system software is implemented in C, providing low-level drivers, a simple scheduler (which supports event-driven scheduling of application tasks) and the host portion of the Bluetooth protocol stack. There are system dependent drivers for both UART ports, analog to digital converters, general purpose IO, random number generator, system clock, and sensors.

At the time of the project start (January 2001), one open source [5] and several commercial implementations of the host portion of the Bluetooth stack are available. The commercially available software stacks posed very high requirements on the system, both in terms of required operating system features (particularly multi-threading) as well as program and data memory provisions. The open source implementation was targeted toward Linux environments and also did not take microcontroller requirements into account. However, previous experiences with the software had shown that about 2 Kbytes of data memory would suffice for a minimal implementation, most of which is used as buffer space. Since all alternatives were equally suitable (or rather unsuitable) we decided to use the open source implementation, last not least due to its immediate availability.

We ported the host portion of the Bluetooth protocol stack from the open source Linux implementation to our microcontroller environment. Supported layers are HCI and the Logical Link Control and Adaptation Protocol (L2CAP). The Linux version of the Bluetooth stack required multi-threading capabilities and access to the serial port. On our system, these functions are taken care of by the scheduler and the low-level drivers. The main obstacle in porting was the limited memory capacity of the microcontroller.

3 Discussion

How well is Bluetooth suited for ad hoc networking today? Bluetooth is the first de-facto standard for ad hoc networking, brought about in a joint effort of many different companies. It was originally conceived as a cable replacement technology and may serve well in that application domain. However, its particular design makes it less suited for other applications in the domain of ad hoc networking. Furthermore, the industry seems to have problems supplying modules in the quantity and quality desired by the market. First products were announced for late 1999, which only appeared late 2000. Even in the beginning of 2001, when we started our project, vendors continued to ship pre-releases of their Bluetooth modules that were still not fully built to the specification.

In this section we discuss some of the idiosyncrasies that hamper the use of Bluetooth technology in distributed sensor networks and describe our efforts integrating the technology into our Smart-Its infrastructure.

3.1 Piconets and scatternets

Bluetooth has been optimized to support a large number of communications to take place in the same area at once. It organizes all communications in piconets, each serving up to eight participants. Multiple piconets with overlapping coverage areas are referred to as a scatternet. It is possible to interconnect piconets by means of units participating in different piconets on a time-division multiplex basis. However, since the radio can only tune to a singe piconet carrier at any instant in time, a unit can only communicate in one piconet at a time.

Piconets are managed by a single *master* that implements centralized control over channel access. All other participants in a piconet are designated *slaves*. Communication is strictly slave to master (or vice versa), but can never be slave to slave. During the existence of a piconet, master and slave roles can be switched. This is desirable, for example, when a slave wants to fully take over an existing piconet. Likewise, a slave in an existing piconet may want to set up a new piconet, establishing itself as its mas-

ter and the current piconet master as slave. The latter case implies a double role of the original piconet master; it becomes a slave in the new piconet while still maintaining the original piconet as master. A unit can be slave in two piconets or be master in one, and slave in another piconet.

Being able to link only the eight nodes of a piconet will in most cases be inadequate to set up densely connected sensor networks. Wanting to communicate with more than eight nodes at the same time will require some sort of time multiplexing, where additional nodes have to be parked and unparked repeatedly. Setting up additional piconets instead will still require gateway nodes to alternate between their respective piconets, since Bluetooth only supports units being in one active piconet at a time.

Also, applications will most likely need slave-to-slave communication, which is not provided in the Bluetooth standard. One obvious workaround is to channel all slave-to-slave traffic through the master, thus increasing both traffic and energy consumption. Alternatively, one of the slaves could switch roles with the current master, or even set up an additional piconet altogether. Both solutions incur substantial communication and configuration overhead. Future experiments will need to show if and how clever communication protocols can alleviate some of this overhead, and more precisely measure its impact on communication and power usage.

3.2 Power consumption issues

The default state of a Bluetooth unit is *standby*. In this state, the unit is in a low-power mode, with all components but the internal clock shut off. In standby there can be no connections open.

When there is an active connection to a Bluetooth unit, it is said to be in *connect* state. In connect state, Bluetooth knows four different power modes: *active*, *sniff*, *hold*, and *park*. In active mode, the Bluetooth unit actively participates on the channel. Data transmission can start almost instantaneously, but at the expense of increased power consumption (compared to the remaining three modes).

When low-power operation is favored over short response times, units can make use of one of the three power-saving modes sniff, hold, and park. All low-power modes reduce the duty cycle of different units within a piconet. In sniff mode, slave units only listen in on the channel at specified times, agreed upon with the master. Hence, transmissions can only start at these times. The connections of a piconet can also be put on hold. In hold mode every participant (including the master) can take some time off for sleeping. Prior to entering hold mode, master and slaves agree on a time when to return to active mode again. The time off can also be used for conducting other business, such as attending other piconets, or scanning for other units.

The park mode is a special mode for slaves that do not need to participate in a piconet, but nevertheless want to remain connected (in order to avoid going through the connection establishment procedure again). Parked slaves do not count as regular (i.e., active) piconet members. In addition to the up to eight active members there may be up to 255 parked slaves within a piconet.

Low-power modes are a trade-off between power consumption and response time. Increasing sleep time reduces power consumption but prolongs time before access can be made, and vice versa. Low-power modes are a powerful tool offering a range of options to applications when the transmission pattern is known beforehand. When data traffic commences at a regular schedule, the sniff and park modes seem to be appropriate. For example, a Smart-Its node may want to dispense its sensor readings every 10 seconds to a node in the background infrastructure (implementing the master). That node would set itself up for sniff mode with a 10 seconds sleep cycle. Similarly, the hold mode serves applications communicating on a more irregular, yet predictable schedule.

If, however, time-critical data transmissions start spontaneously, an application has no other option than keeping the Bluetooth module in active mode. The example here may be a Smart-It node using a background service for processing audio clues. Transmission of the data would need to start immediately after observing the audio clue, since buffering on the local Smart-Its device is not feasible.

The system power consumption for different operating modes, as measured with our prototype, is given in table 1. Table 1 shows that the dominant component in a Smart-It, with respect to power consumption, is the Bluetooth module. Since our Bluetooth modules are engineering samples that do not implement low-power modes, no data is available on how these would alleviate the significant power usage of the modules in active mode.

We expect, however, that improved Bluetooth products will eventually reduce power consumption considerably. Our system design allows for easy replacement of the Bluetooth transceiver module, once improved modules are

Table 1: System power consumption at 3.3 V

CPU powered down, Bluetooth detached	< 11 mW
CPU running, Bluetooth detached	29 mW
CPU running, Bluetooth standby	50 mW
CPU running, Bluetooth inquiry-scan mode	100 mW
CPU running, Bluetooth inquiry mode	200 mW
Bluetooth transmit mode[1]	94 mW
Bluetooth receive mode[1]	94 mW

available on the market.

3.3 Inquiry and connection establishment

Bluetooth supports the paradigm of spontaneous networking, where nodes can engage in communications without a priori knowledge of each other. A procedure termed *inquiry* can be used to discover which other Bluetooth units are within communication range. Connections are then established based on information exchanged during inquiry. Once a unit has discovered another unit, connection establishment is very fast, since information exchanged in the inquiry procedure can be exploited.

Inquiry is an asymmetric procedure, in which the inquiring unit and the inquired unit need to be in complementary modes, called *inquiry* and *inquiry-scan*. When a Bluetooth unit has been set to inquiry mode, it continuously sends out inquiry messages to probe for other units. Inquiry mode continues for a previously specified time, until a previously specified number of units have been discovered, or until stopped explicitly. Likewise, other Bluetooth units only listen (and reply) to inquiry messages when they have been explicitly set into inquiry-scan mode. In a unit, inquiry and inquiry-scan modes are mutual exclusive at any time.

When a unit in inquiry-scan mode recognizes an inquiry message, it replies to the inquirer. Thus, the complete inquiry procedure requires one broadcast message to be sent from the inquirer, and one message from every inquired unit back to the inquirer.

If an inquiry is initiated periodically, then the interval between two inquiry instances must be determined randomly, to avoid two Bluetooth units synchronizing their inquiry procedures in lock step. In a scenario where units are peers, i.e., when there is no dedicated inquirer, application software carries the burden of breaking the symmetry.

As can be seen from table 1, power consumption increases considerably during inquiry. This is due to the asymmetric nature of the Bluetooth inquiry procedure, where the burden of expending power is mostly placed on the unit conducting the inquiry.

In order to save power, a unit in inquiry-scan mode does not continuously listen to inquiry messages. Instead, it only listens for a very short period of time (11.25 ms by default), which, under regular conditions, suffices for the inquiry message to get through with sufficiently high probability. Then the unit enters idle mode for a much longer interval (typically 1.28 s). However, the inquiring unit needs to send inquiry messages (and alternately listen for potential replies) during the entire interval, since it cannot know when the target unit is actually listening.

According to Salonidis et al. [11], the expected delay for link formation (i.e., inquiry plus connection establishment) of peer units is 1 s when both units alternate between inquiry and inquiry-scan modes following uniform distribution. In the link-formation delay, device discovery is by far the dominating factor. However, we came to different, much higher results for device discovery, both in theory as well as in experiments. Firstly, the figures given in [11] neglect that units in inquiry-scan mode only pay attention to inquiry messages for 11.25 ms out of 1.28 s, less than one percent of the time! Therefore, we would need to add $(1.28\ s - 11.25\ ms)/2$ to the expected delay. Secondly, the figures are based on the assumption of an ideal, error-free environment, where messages are never lost.

Indeed, our experiments show that device discovery is much slower in real-live settings and often takes several

[1] Values taken from [8].

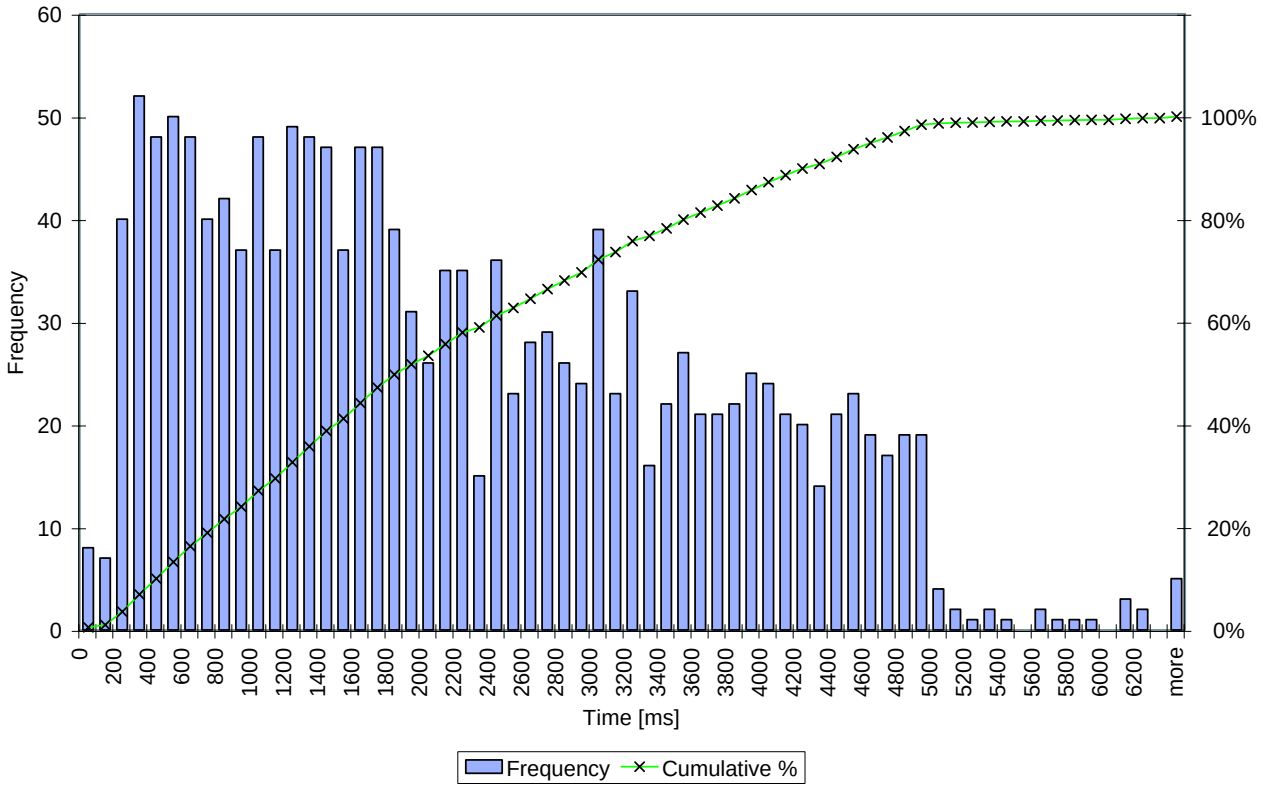

Figure 3: Average time to establish a connection with a device that is in inquiry-scan mode

seconds to complete. Figure 3 shows the distribution of the discovery delay for an inquired device, based on 1500 tests.

The experiment setup consisted of two immobile Bluetooth evaluation boards, using the same ROK 100 007 modules as our Smart-It prototypes, which were placed at a distance of about one meter. One unit was constantly set to inquiry-scan mode, while the other unit was dedicated to inquiry, both using Bluetooth default settings.

Instead of using two of our Smart-It units directly, we decided to use the evaluation boards (where the host portion of the inquiring unit's stack was run on a Linux machine) since it offered us a much finer timer granularity than what would have been possible using the Smart-Its prototype. In every test, the dedicated inquirer was conducting inquiry for exactly 12.8 seconds, even if the target device was discovered in less than 12.8 seconds. Prior to carrying out the next test the inquirer went back to standby for a time uniformly distributed between 0 and 12.8 seconds to avoid synchronization artifacts. The experiment was set in a typical office environment with little traffic from an IEEE 802.11 wireless LAN and no Bluetooth traffic.

The results of the experiments are:

1. the average inquiry delay is 2221 ms
2. after 1910 ms, 4728 ms, and 5449 ms, the target unit had been found in 50, 95, and 99 percent of all tests, respectively.

A possible reason for the high discovery delay may be that inquiry messages and replies to inquiry messages are lost or are not being recognized as such. Differences in the local clocks and the frequency hopping scheme may be reasons for the latter case.

The Bluetooth inquiry model in general seems to be geared toward settings where a dedicated unit is responsible

for discovering a set of other units, e.g. a laptop computer periodically scanning for periphery. It seems less appropriate for truly symmetrical nodes. Also, in the laptop setting described above, a delay of several seconds for connection establishment would be tolerable. In distributed sensor networks such as the Smart-Its network, however, we expect nodes to be mobile. An example would be sports gear augmented with Smart-Its, e.g. bikes, skateboards, or a football. Based on the experienced mean discovery delay of 2221 ms, two Bluetooth devices traveling at a relative speed of 12.5 km/h (4.5 m/s) could already barley set up a connection before moving out of communication range again. The lengthy connection establishment effectively prevents the use of Bluetooth in fast-moving settings.

The inquiry message broadcast by an inquiring unit does not contain any information about the source. Instead, the inquired unit gives away information required for connection establishment, such as the unique device id, in the inquiry response. Thus the inquired unit must reveal information about itself without knowing who is inquiring. This inquiry scheme may become a privacy concern in our project, where personal belongings such as children's toys may be augmented with Smart-Its.

Finally, because of power consumption, Bluetooth's inquiry is probably less suited for low-power nodes that will frequently have to scan their surroundings to discover new nodes or background services. On the other hand, this poses no problem to more powerful devices such as laptop computers: Placing the power burden on the inquiring unit may be a desired feature in an asymmetrical communication setting, where it would relieve mobile low-power periphery.

3.4 Pre-series Bluetooth modules

In their current state, many commercial Bluetooth modules do not offer the full functionality of the specification. The Ericsson ROK 101 007 modules at our disposal have several features missing.

Most importantly, the units do not implement point-to-multipoint connections. Therefore, piconets are limited to just two devices and interconnected piconets cannot be established. Consequently, broadcast and master-slave role switching has not been implemented either, since it only makes sense for piconets of three or more participants. When in connect state, our units cannot actively inquire other devices, nor can they be inquired, regardless of any traffic over that connection. In addition to the very high power consumption of the module, none of the low power modes (hold, sniff, and park) are supported.

Besides important features not being implemented, a number of other difficulties arose during the development of the first Smart-It prototypes. The Bluetooth modules were hard to obtain and came at a rather high price (USD 80 per piece). Secondly, product information, such as unimplemented features, and mechanical and electrical specifications were unavailable until the modules were shipped. Also, for the assembly of the Smart-Its devices a placement machine was required. Whereas all other components could be soldered manually, the ball-grid array of the SMD-packaged Bluetooth unit could not. This meant that the assembly of only a few Smart-Its already required production-scale facilities solely due to the packaging of the Bluetooth units, thus greatly increasing cost.

Although this is not a particular problem of using Bluetooth in general, it indicates that experimentation and development for researchers without direct access to production-scale facilities is made difficult at this point in time.

4 Related Work

Several aspects of our Smart-Its project with respect to its form factor and communication technology are also being investigated in other research projects.

In the Smart-Dust [13] project at Berkeley, similar-sized prototypes have already been built. However, instead of aiming for a sticker-sized form factor, smart dust is ultimately aiming at much smaller size of only a single cubic millimeter, i.e., "dust-sized". Also, instead of radio communication, smart dust pursues active and passive optical communications for ultra-low power consumption. While passive communication is very power conservative, it requires a central authority (base station transceiver, BTS) that initiates communication with a modulated beam laser. Individual nodes reflect a constant laser beam from the BTS and use a deflectable mirror built in MEMS technology to modulate the reply onto the beam. Active communication scenarios using a built-in laser are also investigated. However, it is not yet clear how two such devices can locate and communicate to one another without initially knowing the location of the other device.

Also, Smart-Its are envisioned to provide a substantial

amount of pre-processing, for example in the area of audio and video sensor data, not only on a single unit basis, but particularly as a collective, distributed processor made up of a smart-its *federation*. Consequently, smart-its aim at providing complex *context* information, while smart-dust focuses on relaying direct sensor data to a more powerful central processor.

Within the Smart-Dust project, a range of prototypes have also been built that use different communication technologies apart from optical communication. Most recently, the "weC Mote" uses a custom protocol over a 916.5 MHz transceiver with a range of 20 meters and transmission rates of up to 5 Kbps. It runs a custom micro-threaded operating system called TinyOS [9], features various different onboard sensors and is already being used in the UCLA habitat monitoring project [7] as part of a tiered environmental monitoring system. While one of our project partners within the Smart-Its project, the University of Karlsruhe, is also building Smart-It prototypes using a fixed frequency custom radio transceiver in the 915 MHz ISM band [10], we purposefully wanted to investigate the suitability of the Bluetooth standard in order to allow Smart-Its to communicate with other Bluetooth enabled commercial devices.

Next to the Bluetooth technology employed in our first prototypes, a number of comparable communication technologies exist that support some or all of our required communication aspects.

Both the popular IEEE 802.11 for Wireless Local Area Networks (WLAN) and its competitor, HiperLAN/2 [1], offer ad hoc modes for peer-to-peer communication. Because 802.11 requires a dedicated access point (AP) for many features such as QoS or power saving, its ad hoc mode is very limited. In HiperLAN/2, mobile terminals take over the role of APs when being in ad hoc mode and thus can continue to support QoS and power saving. Since these technologies are mainly intended for scenarios where mobile clients communicate through base stations, their transmission power is considerably higher than that of Bluetooth (10-300 mW, compared to 1 mW in Bluetooth). Future WLAN devices that support transmit power control (TPC) might be a suitable alternative. Also, none of these devices are yet available in the desired form-factor.

A very interesting recent standardization process has been initiated by the IEEE 802.15 working group, which tries to define a Personal Area Network (PAN) standard [2]. Its first incarnation (802.15.1) is to be based on Bluetooth and should improve and extend the existing specification. 802.15.3 aims for high data rates of 20 Mbps or more, at low cost and low power consumption. 802.15.4 supplements a low data rate (10 Kbps) standard, but using ultra-low power, complexity and cost.

Recently, the XI Spike communication platform [3], developed by Eleven Engineering in Canada, has stirred interest as a viable competitor to Bluetooth, operating at both the 915 MHz and 2.4 GHz ISM bands using frequency hopping and direct sequence spread spectrum. Originally developed for the gaming industry (connecting game controllers to consoles), it offers multiple data rates of up to 844 Kbps using a transmission power of 0.75 mW; supports both peer-to-peer and broadcast communication; allows its embedded RISC processor to be used for user applications; and is said to come at a much lower price than any available Bluetooth module today (USD 6.25).

5 Conclusions

Several factors have contributed to the significant attention Bluetooth has received in recent months. As one of the first international standards available, it greatly simplifies ad hoc networking using the piconet communication paradigm. By using the freely available ISM band, Bluetooth devices can be used world-wide without alterations. Its frequency hopping technology makes transmissions robust against narrow-band interferences (which might be frequent within the ISM band). Even though Bluetooth modules are currently rather expensive, prices are expected to drop to about USD 5 per unit once mass-production is running full-scale.

Originally intended as a cable-replacement technology, future Bluetooth modules (i.e., built fully to spec) will be well suited for scenarios where a powerful master device (usually a laptop, PDA or mobile phone) connects seamlessly to a number of peripherals (e.g., a printer, keyboard, or mouse). With data rates of up to 1 Mbps, Bluetooth also offers more than enough bandwidth for ubiquitous computing applications such as simple sensor networks (as exemplified by our Smart-Its). However, scenarios involving a large number of identical low-power devices using ad hoc networking in a true peer-to-peer fashion (like our Smart-Its nodes) still face a number of obstacles when using Bluetooth as their communication technology:

- Asymmetrical communication setup: Finding new communication partners requires one node to be in in-

quiry mode and the other being in inquiry-scan mode at the same time.

- Master-slave communication paradigm: Communication in piconets must always be conducted between master and slave—two slaves must always involve the master node in order to communicate.
- Piconet concept: No more than seven slaves can be active in a piconet at any time—if more nodes need to be added, other active nodes must be put in park mode. In park mode however, nodes cannot actively communicate.
- Scatternet concept: Even though nodes can be in more than one piconet at a time, they can only be active in one of them at a time—meanwhile communications within other piconets must be suspended.
- Power consumption: Even if the power consumption of current pre-series modules can be cut significantly, centralized control of the piconet as well as the asymmetric nature of inquiry and connection establishment puts the burden of expending power onto a single device. Low-power modes may help but they do not apply to every situation.

Incorporating one of the early Bluetooth modules into our Smart-It prototype gave us a good idea on both the potential and the possible limitations of this technology. However, it is difficult to arrive at a final judgment regarding the suitability of Bluetooth for peer-to-peer communication in ubiquitous computing applications: Since our pre-series units lacked many of the standard Bluetooth features (e.g. no point-to-multipoint communication, no power saving modes), we were not able to assert their suitability for applications such as our Smart-Its sensor network.

Even though Bluetooth might not be optimally suited for our domain, it still seems to be the best readily available alternative at the moment. Using a custom radio solution might render communication sensitive to interference unless spread spectrum solutions such as frequency hopping is used. Also, error correction, transmission power adaptation, and fundamental quality of service options that are already part of the Bluetooth standard would need to be re-implemented when using a custom solution. Once the next generation of Bluetooth modules becomes available, we are planning to re-examine their usability as an ad hoc networking technology in our Smart-Its project. Also, we continue to explore other wireless communication standards such as IEEE 802.11, HiperLAN/2, IEEE 802.15, or XI Spike for use in true peer-to-peer, low-power sensor networks, as soon as we will be able to obtain corresponding modules in a matching form factor.

References

[1] ETSI HIPERLAN/2 Standard. http://www.etsi.org/technical activ/hiperlan2.htm.

[2] IEEE 802.15 Working Group for Wireless Personal Area Networks (WPANs) . http://www.ieee802.org/15/.

[3] Spike Homepage. http://www.spike-wireless.com/.

[4] Atmel Corp. *Atmel ATmega103/103L Datasheet*, 2000.

[5] Axis Communications. *Axis Bluetooth Driver Software*.

[6] Bluetooth Special Interest Group. *Specification of the Bluetooth System v1.1*, December 2000.

[7] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the First ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, San Jose, Costa Rica, April 2001.

[8] Ericsson Microelectronics. *ROK 101 007 Bluetooth Module Datasheet Rev. PA5*, April 2000.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS-IX*, pages 93–104, Cambridge MA, USA, November 2000.

[10] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proc. Ubicomp 2001*. Springer-Verlag, September 2001.

[11] Theodoros Salonidis, Pravin Bhagwat, and Leandros Tassiulas. Proximity awareness and fast connection establishment in bluetooth. In *Proceedings of the First Annual ACM Workshop on Mobile and Ad Hoc Networking and Computing, Boston, Massachusetts*, pages 141–142, August 2000.

[12] The Smart-Its Project. http://www.smart-its.org.

[13] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer*, 45(1):44–51, January 2001.