Bluetooth Smart Nodes for Mobile Ad-hoc Networks

Jan Beutel, Oliver Kasten, Matthias Ringwald, Frank Siegemund, Lothar Thiele*
TIK-Report No: 167
Computer Engineering and Networks Lab
Swiss Federal Institute of Technology (ETH) Zurich
8092 Zurich, Switzerland
beutel@tik.ee.ethz.ch

Abstract

In this paper, we describe the deployment of large mobile ad-hoc networks using standard components. The design, implementation and operation of distributed, self-organized, large scale, mobile communication and information systems poses many interesting research problems. While a lot of questions devoted to algorithmic and architectural aspects are already being pursued, few have actually deployed such systems to the extents envisioned. We present Bluetooth Smart Nodes, each of which can store information, compute and communicate using standard wireless interfaces on a limited resource platform. These wireless enabled small devices can interact in a heterogeneous environment consisting of different types of networking nodes as well as with other wireless enabled appliances. Important requirements and design tradeoffs to be able to support multiple communication interfaces, handle limited resources, power aware operation and efficient testbed deployment are discussed. The BTnodes are integrated into our MANET application and networking framework. Demo applications give an insight into usage scenarios envisioned for future architectural explorations.

1 Introduction

Mobile ad-hoc networks have many attractive applications as computer systems and communication devices become truly ubiquitous. The past years have seen quite a few technologies and devices appearing and maturing. Handheld devices are no longer emerging technology but widespread. Especially mobile communication devices have opened new opportunities with applications like the short message service generating billions of interactions per year between independent devices.

The ongoing trends in miniaturization, price, performance and ever higher levels of system integration are pushing electronic devices into regimes that exceed visions [1] and expectations alike [2]. In contrast to the user centric approach of PC's, and infrastructure based systems like cellular telephony in the past years, the class of networked sensors are becoming a reality today that are actually empowering the environment to become "smart". As the Internet becomes ubiquitous, not the devices themselves will be dominating the field, but the network, applications and interactions. Interoperation between the most heterogeneous devices is thus an absolute necessity.

By using standardized communication interfaces, these networking nodes can interact with everyday appliances, peripheral devices, sensors and actors. A user can gain access to resources in the network and to sensors and actors attached to nodes of the network at its perimeter to the physical world. According to [2, 3] services in the network are the dominating factor for future growth. Fostering this interaction are established userinterfaces on

*The work presented in this paper was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

already common devices such as PDA's and cellular phones that make it possible to reach out into the digital representation of everyday objects and interactions.

This requires infrastructure that can provide peer to peer services across many hierarchies of devices and in a distributed and heterogeneous world. In order to scale to populations magnitudes higher than present today, flexibility, compatibility, self assembly and a timely deployment are as important as low power and small size.

Nevertheless, to date only very select deployments of these ideas have been realized [3]. Only very limited and actually few solutions for the implementation of ad-hoc networking system today are available. They are usually restricted in flexibility and the number of nodes in a single deployment that can actually interact with each other. Heterogeneity in existing solutions usually is limited to a maximum of a mobile unit and a base station. Interaction between hierarchies of devices can only be made possible if certain commitments are made and compatible standards are used, albeit their known deficiencies.

In order to be able to implement and analyze algorithms, networking principles, architecture tradeoffs and applications on a great number of mobile autonomous nodes the wireless prototyping testbed based on the miniaturized Bluetooth Smart Nodes (BTnodes) discussed in this paper has been developed based on standardized technology and prefabricated OEM macro modules. This makes it possible to focus on the higher levels of abstraction such as network, transport and session layer as well as applications and services. The utility of ubiquitous technology like Bluetooth and Wireless LAN is dominating clearly over the negative aspects such as the unacceptably high power consumption, considerable bulk and complexity of the implementations available today. To achieve the goal of large heterogeneous MANETs with all types of "smart" devices valuable insights gained from the deployment are going to have to drive the future development issues in lower level radio and baseband hardware.

1.1 Ad-hoc Networking Design Considerations

In order to support large ad-hoc network configurations of hundreds and thousands of nodes certain requirements have to be considered. Mobility, scalability and most important self-organization are the most dominant of these requirements, while flexibility, low power consumption and small size are optimization goals in the design space.

Networking nodes each need to contain storage, computation and communication resources to account for the necessary flexibility and scalability in such an environment. Communication should not only be possible through one single type of interface with one frontend per node but over multiple frontends per node enabling the exploration of network topologies at a higher degree of freedom without topology limitations, interference and performance degradation.

Achieving low power operation is crucial when scaling to very large and dense populations since increases in power densities and energy scavenging techniques are not keeping up with the rapid pace of semiconductor development. Coupled to low power consumption, but also due to the nature of many MANET applications, that are often embedded applications, stringent resource constraints apply. An optimal use of the available real estate in respect to system size, CPU cycles, wireless communication bandwidth and battery capacity efficiently is crucial.

The use of standardized communication systems such as Wireless LAN and Bluetooth allows testbed components to communicate with other wireless enabled appliances such as cell phones, headsets, PDAs, laptops, keyboards, game controllers, printers, video equipment, mobile storage, media players, cameras and many others through well defined APIs. This is crucial for the timely deployment of such a system as development time is reduced and off the shelf components can be integrated. Moreover as the Internet becomes ubiquitous, wireless access will be available virtually everywhere either through broad coverage of Wireless LAN, cellular services or occasional hot spots. Without connection to the outside world, the use of such a heterogeneous networked system would be very questionable and most tedious, the possible deployment limited to an artificial scenario at a single lab location.

In designing a test and demo system for mobile ad-hoc networks one of the largely unknown is the exact application and it's final implementation that these systems will serve. This implies, that one has to over-provision certain resources in order to cater for the development overhead and to be able to detect possible tradeoffs, architectural and structural changes. A well understood standard operating system (Linux) and ready to use development tools (gcc, gdb and Java) certainly are desirable when researchers that are not experts in embedded architectures want to profit from such a testbed for their own purposes. Here, where MANET applications, mechanisms in the network, transport layer and above as well as network services are the intended application of such a framework, it is important to provide this flexibility.

1.2 Related Work

Many projects on integrated wireless sensor networks have triggered a large amount of work in the area in the last years. Although most of them have system centric approaches, they often focus on understanding and developing low level radio and baseband technology suitable for very limited, single chip implementations [4, 5, 6, 7].

The most prominent of all network sensor implementations today are surely the UC Berkeley motes [8] that are very similar to the BTnodes but have less IO and memory resources and a different radio frontend. In order to operate this frontend all the medium access layer and baseband processing has to be done on the host CPU itself, requiring a carefully designed and fine-grained real time operating system (TinyOS) to meet all timing constraints of the radio. This also requires a meticulous design process for any application implemented, that makes use of the host CPU, which implies knowledge about RT systems and training. The greatest drawback of such a full custom solution is, that it cannot interface with other wireless enabled devices, other than motes.

PDA's such as the iPAQ that are also part of our testbed scenario have been widely used as a ready to use, well supported solution when size constraints are given and systems with more than two or three laptops are desirable [9, 10, 11]. Recently, commercially available motes have been used in conjunction with PDA's to be able to increase the number of available nodes [12, 13, 14]. The great success of the motes and of TinyOS as a prototyping platform shows, that a demand for such systems is clearly available and further research necessary. Other prototyping systems are based on FPGAs, StrongARM processors etc. [15, 4].

The Pushpin nodes developed by the MIT Amorphous Computing experiment [16] are confined to specific locations and rely on infrastructure to operate. Custom infrared and RF communication interfaces have been developed here as well.

IrDA has been used in the past [17] for very early work on ubiquitous interaction devices but has proven to be unreliable and heavily dependent on the line of sight [18].

Issues concerning the power consumption are targeted by all these projects alike. Most notable are the strategies to use a secondary low power signaling radio link to wake up a remote system [9] and the online approaches for dynamic power management by [19, 20, 21, 22, 23].

In the following section we give an overview of the testbed components and on the technologies involved and their limitations. In section 3 the Bluetooth Smart Nodes are presented in detail with a focus on the hardware features. Concepts of energy aware operation are outlined in both sections with a detailed power consumption breakdown. Section 4 deals with the system software of the BTnodes. We conclude with two demo examples that have been implemented on our platform to demonstrate the interaction of multiple heterogeneous devices.

2 Overview on MANET Testbed Components

The wireless testbed described consists of three different classes of networking nodes: A Maxi class based on a laptop PC, a Mini class based on a PDA and a Micro class based on an embedded microcontroller, the BTnode. These networking nodes each feature multiple communication interfaces as well as computing and storage resources equivalent to their class specification.

Class	**Cost** [$]	**Size** [cm^3]	**CPU Type**	**Frequency** [MHz]	**Storage** [Mbyte]	**Memory** [Mbyte]	**Bandwidth** [Mbit/sec]
Maxi	3000	3200	Pentium4	2000	40000	500	108
Mini	1000	300	XScale	400	1000	64	11
Micro	110	12	ATmega128l	7.3	0.128	0.064	0.768

Table 1. Testbed component features

Depending on the applications and the scenarios surely the one or the other platform has it's advantages. In selecting such a heterogeneous group we are able to investigate into the interplay of the very small, numerous and resource limited devices such as smart tags and wireless sensor network nodes with larger already common but more bulky equipment incorporating a host of resources, extensive monitoring, debugging and development support and sophisticated user interfaces such as PCs and PDAs.

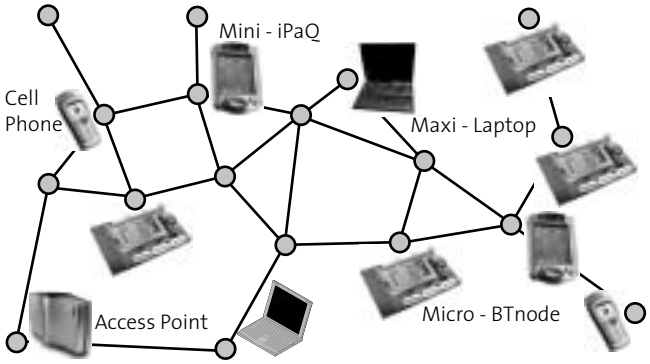

Figure 1. Different scales of heterogeneous devices that can interact with each other

The three classes of networking nodes can be distinguished according to cost, size, power and resource features. An overview of the features and classification is given in table 1. Furthermore, all classes have compatible interfaces, both to the user and to the developer.

2.1 Wireless Interfaces and Resources

Wireless networking systems available today range from Wireless LAN to cellular and low rate messaging and control technologies. Ideally such a medium would be required to be able to send a lot of data, very far, very fast, to many separate users and all at once [3]. Unfortunately to achieve all of these objectives simultaneously is prohibited by laws of physics so a compromise has to be made in many cases.

Major differences can be identified in the spatial capacity, network topology, resource consumption, latency and throughput. Significant trends have been identified and are expected to sustain into the future: I. A growing demand for wireless data in portable devices, II. the crowding of spectrum, III. the growth of high speed wired Internet access, and IV. the shrinking semiconductor cost and power consumption for signal processing [24, 4, 5, 6].

The wireless communication interfaces used in this prototyping platform are standard interfaces. This allows for flexible interaction with other devices that incorporate these standards. The availability of standard interfaces and drivers is necessary for a fast deployment of the whole system to potential users.

Wireless LAN has shown to perform very poorly when used with single interfaces in multihop scenarios [25, 24]. Multiple networking interfaces per nodes allow to overcome limitations imposed by standardized wireless systems at the cost of additional channels and of resources. Using multiple instances of both Bluetooth and Wireless LANs in a node allows to adapt to the necessary cell sizes (overlay networks) and to scale the available bandwidth in multiples of 0.768 (Bluetooth), 11 (IEEE 802.11b) and 54 (IEEE 802.11a), up to 108 Mbit/sec (proprietary IEEE 802.11a). Additionally, multiple Bluetooth systems per networking node make it possible to actively route across the very limited piconet borders, whereas Bluetooth and Wireless LAN in one node allow to bridge between the different standards.

2.2 Communicating with other Wireless Enabled Devices

A connection from the testbed components to other networks, especially to the Internet is very important for interacting in applications and for monitoring and control purposes. It can be achieved through Access Points, at medium and high data rates as well as by using cellular services, such as GPRS (see figure 1). Nodes supporting multiple types of interfaces simultaneously can act as a bridge between different systems. For example an iPAQ equipped with Wireless LAN and Bluetooth can provide IP connectivity to a Bluetooth piconet through WLAN access points, whereas a Bluetooth enabled phone can provide the same for a wireless ad-hoc network when out of range of the access point. By using BTnodes in such an environment sensors and actors can instantaneously become part of the always-on, always-connected Internet.

GSM phones are widely spread and can serve as good user terminals due to their familiar user interface. They can send and receive short messages, exchange data objects, provide low rate IP access. Recently uploading applications

and remote execution of these is also becoming a comodity. Other devices, mostly from the multimedia domain are slowly becoming available, although far from a commodity. They can be used for various interactions with the testbed components described (see the application example in section 5.2).

A drawback is that the standardized protocols like Bluetooth are often very specialized and quite clumsy. Almost every interaction imaginable is defined by a profile that needs top be implemented on the devices.

2.3 MANET Testbed Classes

The Maxi class consists of laptops equipped with different wireless interfaces ranging from low rate, low range Bluetooth at 768 kbit/sec to high rate IEEE 802.11a at 108 Mbit/sec. Support for multiple interfaces is only limited by the overall size, mobility requirements and power consumption. Characteristics of this class are flexibility, a feature rich user interface and a wealth of support and debugging possibilities. The use of the Linux operating system gives the system the most transparency possible for development and monitoring work.

The Mini class are iPAQ type devices that are already a common handheld platform used by many researchers. The benefit of this device is the considerable reduction in form factor while still supporting a familiar and powerful operating environment, integrated Bluetooth and Wireless LAN and the broad availability of peripherals and documentation. The networking interfaces that can be supported simultaneously are somewhat more limited than on a laptop because only one expansion pack can be added at a time and the USB port is solely available to be used as a slave device.

At about 1/10 the cost and the size of a laptop it can be used in higher mobility settings with possibly more units than with PC based systems. The ratio of the operating frequencies, memory and network bandwidth are comparableat around 1/10, only the permanent storage capability is considerably reduced by 1/40. Most deployments today are known to have on the order of tens of these devices [10, 12, 9].

The characteristics of the Micro class devices are a competition-less small form factor and low component count while maintaining a standardized wireless interface. A detailed description of the system implementation is available in sections 3 and 4.

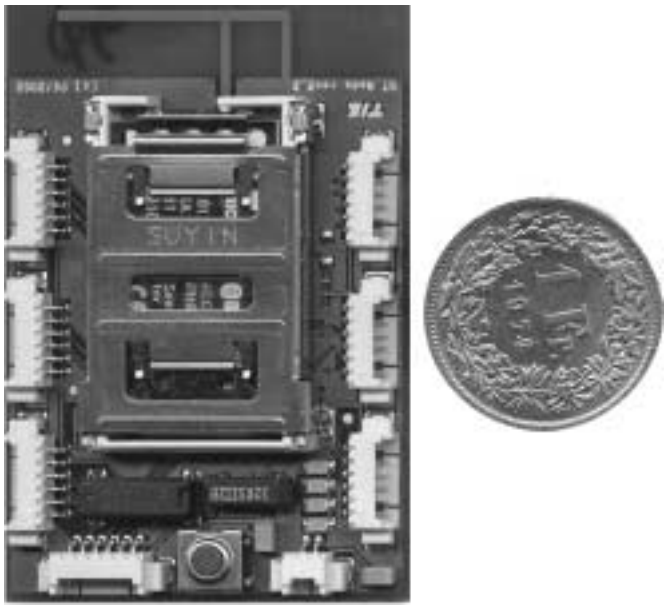

Figure 2. The Micro class: Bluetooth Smart Node with microcontroller, sensor interface and Bluetooth

To fulfill the requirement of ubiquitous deployment of MANET capable devices, many more devices than feasible using the Maxi and Mini class need to be considered. When comparing the Micro class devices to the other two device classes the magnitude of reductions in cost, size, operating frequency and communication bandwidth are comparable, at about 1/50 for the Mini class and about 1/100 for the Maxi class resulting in a similar distribution

of resources among these devices. The storage and memory is reduced drastically by factors of many thousandsfor nonvolatile storage and for system memory. This shows clearly that these devices are not suited for resident applications but more for networking and communication applications.

Class	Operating System	API
Maxi	Multitasking OS	Java JDK/JRE
Mini	Downsized OS	Simple Java JRE
Micro	Limited OS	Simple drivers

Table 2. Software APIs

For each networking interface the appropriate drivers are provided as part of the operating system. At a minimum a connection/disconnection state machine and means for exchanging addressing information are necessary in such a driver. Applications then can make use of the interfaces through these drivers.

2.4 Energy Aware Operation

Many developments concerning lowering the power consumption of very specific parts [20, 19, 22] or of whole systems [21, 26] have been made. In principle, there are two ideas most noteworthy to our testbed approach: I. Wireless communication strongly dominates digital computation [4, 8, 6] and II. If you really want to save power you have to turn off systems or parts thereof [11, 9, 23]. In the sense of a whole architecture, to turn off a system really means to power off, not just to trigger a disable pin or go to a different power mode.

The well known numbers on the bitwise energy consumption for wireless communication and computation [6, 4] are important indicators. But they cater primarily to the development of semiconductor circuits. From a system perspective it is not just important to achieve a single most efficient power mode, but an overall optimized duty cycle for the system power consumption.

This means that one has to provide flexible methods for controlling all the resources and their modes of operation available on a networking node. Where shutdown or disable is not implemented in the hardware or in cases where these are not sufficient to reach a power dissipation close to 0 when unused, a separate software controllable power supply should be implemented. The result achieved through this method is comparable to clock gating with a remaining power dissipation of only the standby current of the power supply which is typically < 6 µW. A good example for power control are the PCMCIA specifications that have added power-management in a well-defined interface. Insertion and removal, suspend and resume as well as the time most appropriate to change to a new power state are detected by the drivers. When disabled, the driver interface can disconnect the card from it's power source completely.

Standby modes are not really an option when the goal is to investigate networking applications. Without improved energy saving mechanisms and standard rechargeable batteries the Maxi and Mini classes can be operated autonomously for only short periods. The Micro platform shows a tenfold increase in the operating time (see table 3).

Class	Supply [Wh]	Power Drain [W]	Lifetime [h]
Maxi	53.28	13.3-26.2	2-2
Mini	5.04	0.39-2.02	2.5-13
Micro	3.02	0.067-0.25	12-45

Table 3. Power consumption under load

3 Bluetooth Smart Nodes - The Embedded Micro Platform in Detail

The BTnode is an autonomous wireless communication and computing platform based on a Bluetooth radio module and a microcontroller. The benefit of this platform is the small form factor while still maintaining a

standard wireless interface. With its general purpose interfaces the BTnode can be used with many peripherals, such as sensors, DSPs, serial devices (like GPS receivers, RFID readers, etc.) and user interface components.

Having networks comprising hundreds of autonomous nodes in mind, the following target features were considered in the design of the BTnodes:

- **In-circuit programmable Bluetooth platform:** Simple reprogramming of the system software as well as drivers is crucial to any development work. Although a very standard requirement a surprising amount of devices do not yet support this feature.
- **Remote update of system software:** When it comes to the deployment of the BTnodes there must be a way to reprogram the nodes over the network as they will be distributed in many different locations and not necessarily attached to a PC.
- **Low component count:** Cost effective, timely deployment and the reduction of design iterations both demand a fairly simple design consisting of standard components.
- **Overall system size:** One of the most apparent arguments for not deploying laptops and PDAs in large quantities besides the cost is the system size. They would interfere unduly with the environments that are intended for the use of our testbed components and sometimes cannot be placed exactly where necessary.
- **Simple debugging capability:** An experimental platform must support an easy to handle standard way of debugging. Many changes to the software and extensive monitoring are the main applications for this feature.
- **Sensor and user interface:** The BTnodes are targeted to be attached to sensor and UI peripherals in many usage cases. A standard interface is thus necessary.
- **Single voltage design with power management:** Supporting multiple operating voltages creates a much more complex design and limits the possible interactions with peripherals.

Most commercial Bluetooth solutions are available as fully self-contained transceiver modules that implement a the lower layers of the protocol from RF frontend to the generic Host Controller Interface (HCI). Representing a classical embedded system they contain an embedded CPU, different types of memory, baseband and radio circuits. They are designed to be used as add-on peripherals with the higher layers of the Bluetooth protocol and applications implemented on a host system. Due to system partitioning and commercial interests the in-system CPU and memory are usually not available to developers for user specific implementations.

Thus, a minimal stand alone Bluetooth node needs an additional host CPU to execute applications and the corresponding higher layers of the Bluetooth protocol like the connection state machine in the logical link layer and adaption protocol (L2CAP).

3.1 BTnode Hardware Overview

The BTnode hardware (see figure 3) consists of an Atmel ATmega128l microcontroller with on-chip memory and peripherals. The microcontroller features an 8-bit RISC core delivering up to 8 Mips at a maximum of 8 MHz. The on chip memory consist of 128 kbytes of in-system programmable Flash memory, 4 kbytes of SRAM and 4 kbytes of EEPROM. There are numerous peripherals integrated as well: JTAG for debugging, timers, counters, pulse width modulation, 10-bit analog digital converter, I2C bus, two hardware UARTs. An external low power SRAM adds another 60 kbytes totaling in 64 kbyte SRAM available on the system to be used as data memory.

A real time clock is driven by an external quartz oscillator to support timing updates while the device is in low power sleep mode. The system clock is generated from an external 7.3728 MHz crystal oscillator.

The single input power supply is designed to be used on a 3 cell rechargeable battery. The power management is performed by a low dropout linear voltage regulator that provides the system voltage of 3.3 V. Although parts of the BTnode system could be operated at lower voltages this option was not used since 3.3 V is the one most commonly used and it would make it more difficult to interface with other devices and sensor peripherals. A final production version would surely prefer a lower operating voltage over a prototype.

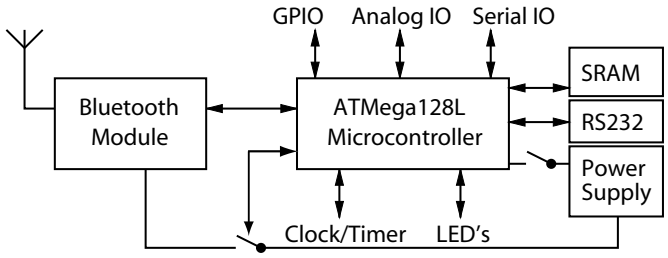

Figure 3. The BTnode system overview

A Bluetooth module is connected to one of the serial ports of the microcontroller using a detachable module carrier and to a planar inverted F antenna (PIFA) that is integrated into the board to reduce the critical components. The operating and power modes of the Bluetooth module can be controlled by the MCU.

Four LEDs were directly integrated, mostly for the convenience of debugging and monitoring although they could easily be optional external add-ons. One analog line is connected to the battery input to be able to monitor the battery status. A reset button was added to ease software development. The other peripherals are directly accessible on external connectors. These connectors all have 4 signal lines and additional power and ground lines to be able to flexibly power and control add on sensor peripherals. The supply voltage for MCU, memory and Bluetooth is fed through 0 Ohm resistors. This is a common way to enable exact in-situ power consumption profiling for each component. Simply detach the resistor and add a connection to a current meter.

In cases where additional networking capability is necessary, a second Bluetooth module can be attached to the second serial port of the MCU as an external peripheral.

3.2 Designing for Power Aware Operation

When pursuing an overall strategy to reduce the system power consumption, the interplay of the system components is most important.

Different power saving modes are available for both the microcontroller and the Bluetooth module. Furthermore the microcontroller can be run at different operating frequencies controlled by the software. Clock division is supported from 2-129 resulting in an operating range from 7.3728 MHz down to 57 kHz (see figure 4 for the resulting power consumption of the MCU). Of course changing the system clock frequency in operation affects most of the peripherals as well and caution must be exercised when using devices like UARTs and timers that depend on the system clock frequency. Therefore it is suggested to only use this feature to adjust the optimal point of operation after an application has been defined and not to implement dynamic frequency scaling to achieve optimal power savings.

The processor supports six different sleep modes that use clock gating techniques to disable certain modules in the MCU, thereby saving power: Idle, ADC Noise Reduction, Power-down, Power-save, Standby and Extended Standby. The modes are controlled by a register value and activated upon execution of the `sleep()` instruction and last until an interrupt occurs. Table 4 shows the power consumption for every mode.

- **Idle Mode:** This sleep mode basically halts *clkCPU* and *clkFLASH*, while allowing the other clocks to run.
- **ADC Noise Reduction:** This sleep additionally halts *clkI/O*, while allowing the other clocks to run. This improves the noise environment for the ADC, enabling higher resolution measurements from sensors.
- **Power-down Mode:** In this mode, the external oscillator is stopped. When waking up from Power-down Mode, there is a delay from the wake-up condition.
- **Power-save Mode:** This mode is identical to Power-down, with the exception of *TimerCounter0* that will continue to run during sleep.
- **Standby Mode:** This mode is identical to Power-down with the exception that the oscillator is kept running.

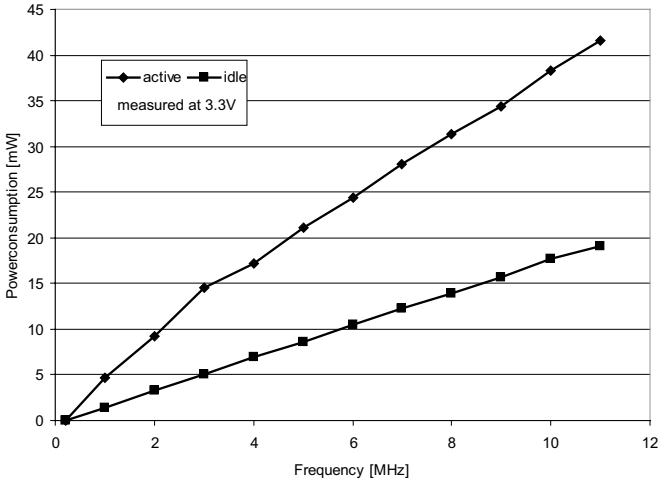

Figure 4. Processor power consumption

- **Extended Standby:** This mode is identical to Power-save mode with the exception that the oscillator is kept running. From both standby modes, the device wakes up in 6 clock cycles.

Component	**Power Consumption** [mW]						
	Active	**Idle**	**ADC**	**P-down**	**P-save**	**Standby**	**E-Standby**
MCU ATmega128l	29.04	15.71	6.99	0.04	0.05	0.53	0.52
	Active			**Dynamic Access**		**Standby**	
External SRAM	1.65			2.44		0.09	
	Off	**On**	**Init**	**Visible**	**Inquiry**	**Connect**	**Tx**	**Rx**
ROK 101 008 R3A	17.46	96.53	69.96	71.45	162.92	102.20	134.54	138.34
ROK 101 007 P4D	17.46	174.64	134.05	136.16	216.25	157.61	164.27	197.80
ROK 101 007 R1A UART	16.24	109.43	37.69	89.96	151.37	138.27	170.08	137.94
ROK 101 007 R1A USB	1.29	104.08	128.34	146.49	205.82	165.76	226.68	157.91

Table 4. Component power consumption breakdown for different modes of operation

The power control of the Bluetooth subsystem follows a twofold strategy. The Bluetooth module supports different power modes that can be activated either by an external line (enable) or in software. By large, the power consumption depends on the operating mode of the radio. This means, that given an optimal communication strategy and in relation to the consumption of the rest of the components power management must only be considered while not communicating.

In table 4 the detailed power consumption breakdown is given for different versions of the same Bluetooth subsystem in the relevant operating modes. The difference of pre-series (ROK 101 007 P4D) to the certified product and of the point-to-point (ROK 101 008) to point-to-multipoint (ROK 101 007) are quite plausible and can be explained by improvements in hard- and software. The most significant difference exists in the case of the same hardware while using alternative interfaces (ROK 101 007 R1A). This shows, that a USB interface results in a much higher overall power consumption whereas a UART system is highly ineffective in the idle states.

A major drawback is the large power consumption while the module is disabled ($\approx$ 18 mW). This is more than the MCU consumption. For long idle periods on the communication link a separate power supply controlled by the microcontroller is implemented to turn off the Bluetooth supply current resulting in a consumption of only the shutdown power consumption of the additional voltage regulator shown in figure 3 ($\leq$ 2 µW).

4 System Software for the BTnodes

The BTnode system software is a lightweight operating system made up of low level drivers that are interrupt driven and a simple dispatcher for scheduling multiple threads. This OS is well suited for the applications of such small scale networking devices that will consist mostly of of simple IO and monitoring tasks and communication.

There are two programming models: A sequential model and an event-driven model with cooperative multitasking. Different system software exist for the two models. The sequential model allows tight control of all resources but leaves more tasks to the application programmer, while the event-driven model provides convenient functions for resource management but is more restrictive when accessing hardware resources. This is explained in the following.

4.1 The Dispatcher

The dispatcher implements cooperative multitasking. Only one task (event handler) can be active at a time. Events are processed in the order they appear. Every event handler is always completely executed until the next is scheduled. So every event handler depends on the previous event handler to terminate in time.

A software component, such as a driver, can generate an event to notify other components of the occurrence of a change in state that requires further action. For example a byte is received on the serial port. The serial driver now needs to notify an interested software component (e.g. an application or another driver) about this event so that it can read the byte received. This is done by inserting the corresponding event into the dispatchers event queue. The dispatcher has a FIFO queue to hold events that have been triggered but not yet handled by the dispatcher:

```
event_queue (fifo queue):
  | ... | ...       |
  | ev1 | call_data |
  | ev2 | call_data |
  | ev3 | call_data |
```

Execution of event handlers can be delayed by other tasks. In that queue, however, there might be several other events already waiting for their event handlers being executed. So the delay by other tasks depends on the system load (tasks waiting for execution) and the time these tasks may take.

A callback table stores information relevant to an event. That is, the callback function `cb` to handle an event and a data structure `cb_data` that is passed to the callback:

```
cb_table:
   1: ev1 | cb | cb_data |
   2: ev2 | cb | cb_data |
   3: ... | .. | ...     |
```

4.2 Reduced Bluetooth Protocol Stack

The Bluetooth radio module offer a generic host controller interface (HCI) to the lower layers of the Bluetooth protocol through a serial port while the higher layers of the protocol such as link and networking layer must be implemented on the host system. HCI commands and events are asynchronous which means that a state machine must be available to handle all these correctly.

The relevant commands and data structures from the HCI and L2CAP layers (see figure 5) have been implemented to support a basic connection manager at a very low level. The higher Bluetooth protocol layers required to provide connectivity to consumer devices are now part of the applications and are to be integrated in future system software.

Compared to TinyOS our operating system is quite a bit larger. Most of this is accounted for by the Bluetooth stack ($\approx$ 30 kbytes) that has to support a substantial amount of commands and events in `hci.o` and also more sophisticated drivers using handshake and buffers. The core components account for $\approx$ 1710 kbytes with a variable number of drivers, Bluetooth and debugging support as can be seen in table 5. For the support of the BTnodes a

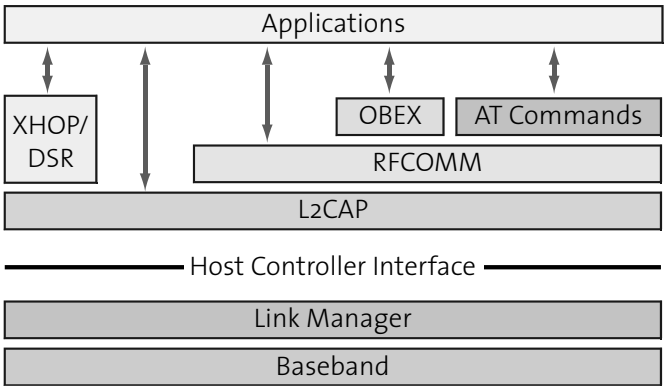

Figure 5. Reduced Bluetooth protocol stack

few more features and therefore more code are actually a benefit because it eases development through debugging support and flexibility.

Component	**Code** [bytes]	**Data** [bytes]	**BSS** [bytes]
`avr128_init.o`	12	0	0
`dispatcher.o`	774	4	462
`timeouts.o`	702	0	0
`share_random.o`	232	0	0
`avr128_rtc.o`	778	0	5
`avr128_power.o`	84	0	0
`avr128_uart0.o`	1320	0	260
`avr128_uart1.o`	1278	0	260
`avr128_led.o`	132	0	0
`hci.o`	15857	0	10
`bt_hci_in.o`	3388	3	16
`bt_stack.o`	295	0	0
`connect_manager.o`	1342	8	0
`l2cap.o`	9358	0	5
`avr128_printf.o`	1484	0	0
`debug.o`	1577	0	0

Table 5. Code, data and buffer size breakdown of the BTnode OS (not optimized)

4.3 Development Tools

A software kit consisting of a build environment (avr-gcc cross compiler and libraries), source libraries, debugging support, demo examples and documentation has been assembled for the BTnodes and is available for download. A support mailing list and repository are also available to developers.

In order to support fast prototyping and debugging without having to download to the embedded target on every design iteration, a separate build tree on Linux with the required interface and hardware emulation has been developed.

5 Applications and Deployment of the Prototyping Platform

In the following we give some insight into applications that have been implemented using the testbed components and especially the BTnodes.

5.1 Ad-hoc Dynamic Source Routing on Bluetooth

A reduced version of the CMU Dynamic Source Routing protocol [27] has been implemented as a packet forwarding mechanism on the wireless testbed. This XHOP/R-DSR protocol enables communication across piconet borders and hooks right into the L2CAP layer of the Bluetooth stack (see figure 5). It has been assumed that routes can be discovered and maintained separately from the packed forwarding mechanism. Due to the limitation of Bluetooth to master-slave piconets with at most 7 active slaves the capability of the multi-hop routing is reduced to a multi-hop message passing mechanism. Furthermore the long setup delays of Bluetooth are a drawback when standard BTnodes with one communication frontend are used. On every hop a connection is opened by the source (master) and an XHOP packet (see figure 6) is transmitted to the next destination (slave). This connection is then closed and the payload is evaluated. In order to forward a packet over further hops a new connection is opened and the XHOP packet is sent on until the final destination has been reached.

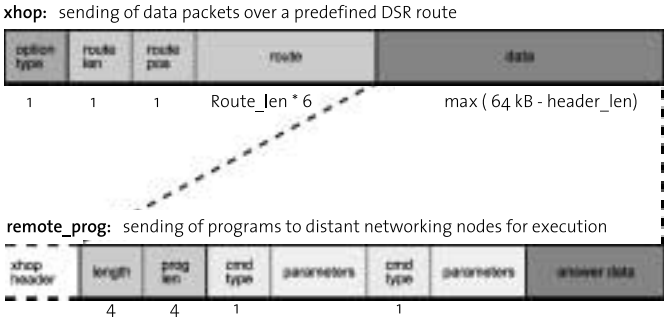

Figure 6. The format of XHOP/R-DSR packets

A simple script command language has been implemented in the payload of the XHOP packets to allow the execution of remote commands. For example a remote topology discovery at node E initiated at node A would consist of one DSR packet with the commands `xhop(ABCDE)`, `inquiry()`, `return_result()` and `xhop(reverse_route)` in the payload (see figure 7). The returned DSR packet would then consist of the Bluetooth addresses of the nodes discovered at node E, namely D, F and G in our example.

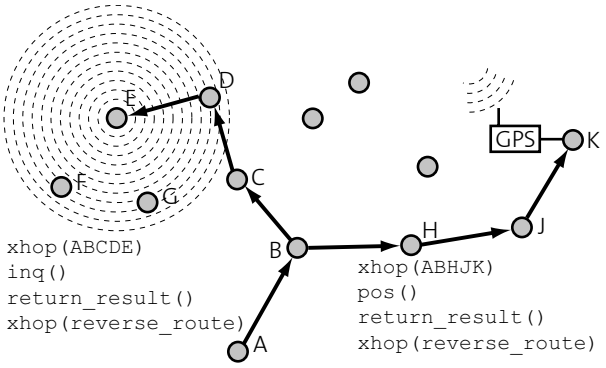

Figure 7. Remote topology discovery in a heterogeneous environment using XHOP/R-DSR

With an average setup time of up to 1 sec per link the performance of this multi-hop packet forwarding application for long routes is insufficient. An additional Bluetooth module connected to the second serial port of a BTnode

allows dually equipped nodes to each be a Master of a piconet. Using this extension makes it possible to keep a whole XHOP route open for subsequent packet transfers without having to open and close one link at a time. The average end to end delay was measured to be ≤ 40 msec per hop.

A second application of this topology discovery is an absolute positioning service using a Global Positioning System (GPS) module attached to a BTnode. These nodes can be queried in much the same manner as described in the first example. In figure 7 such a positioning beacon can be seen in node K. For example `xhop(ABHJK)`, `pos()`, `return_result()` and `xhop(reverse_route)` would return the GPS coordinates of node K along route KJHBA.

To simplify servicing a number of configuration and operation parameters of the BTnodes can configured remotely with the `config()` command. A remote bootcode update is possible using the bootloader feature of the MCU and the `programm()` command over the XHOP protocol. Of course this application is also available in the Mini and Maxi platform where control and monitoring of a collection of BTnodes most suitably takes place.

5.2 Spontaneous Interaction with Bluetooth Enabled Mobile Phones

Spontaneous interaction in ubiquitous computing settings requires a simple, well-understood means for users to communicate with devices in their nearby environment, some kind of equipment most people are familiar with and a mechanism to determine whether certain entities can interact with each other. Mobile phones are well-established means for people to communicate not only verbally, but also by the Short Message Service (SMS) [28].

Interaction of sensor events from the environment can be forwarded from BTnodes that are directly connected to the sensors via SMS using the GSM network. Here, an SMS template is pushed onto a mobile phone by a BTnode. The phone serves as an SMS gateway and can then send this message to it's final destination. There the received message can be used to formulate a reply to the origin.

Figure 8. Product monitoring using BTnodes as smart tags and SMS via mobile phones

Similarly, the Bluetooth OBEX profile can be used to exchange and store data from BTnodes with mobile phones. Figure 8 shows an example of a product monitoring application where a shock sensor has been attached to the BTnode to detect motion patterns.

5.3 Deployment

The small form factor and low component count allow for a cost effective deployment of large quantities of networking nodes. The first prototypes were designed and manufactured in under 2 months.

The BTnodes have been developed and distributed in cooperation of the NCCR-MICS with the Smart-Its Project, the latter being a part of the EU Disappearing Computer initiative. The low complexity and small bill of material of the BTnodes results in a unit cost of $ 110 for the initial deployment of currently 200 units that have been distributed among different European research groups. So far the feedback from these partners has been very promising and many students have been able to start their own projects successfully on our platform within a couple days. Many of them are now well under way in conducting their own research into ubiquitous and pervasive applications and networking.

6 Conclusions and Future Work

In this paper we have reasoned that it is necessary to use standard wireless interfaces to be able to interact with wireless enabled consumer appliances. Moreover the design and implementation process of applications is simplified through the use of well defined standard APIs.

A prototyping testbed that is suitable for the deployment of large populations of heterogeneous distributed mobile communication and information systems to support a "smart" networked environment in everyday objects has been motivated and described. The centerpiece is the Bluetooth Smart Node, an embedded networking device that can store data, compute and communicate over a standard wireless interface. We believe that realizing strategies for the reduction of the overall system power consumption implies a good understanding and flexible control of all subsystems and respective operating modes involved. System centric power aware operation has been realized in the design of the hardware and the lean operating system.

The drawbacks of Bluetooth, namely the high power consumption and the rigid networking structure can be overcome by our approach of multiple frontends per node. They are outweighed by the fact that interaction can take place with almost any other wireless enabled device. The two examples and the successful deployment with different research groups show that our approach is feasible and extendable to larger systems and applications.

Future applications of the BTnode will be in the networking and ubiquitous computing application domain. Enabling technology like network topology management and control as well as positioning services will evolve to support applications

Thanks to Matthias Fries for simulating the antenna design and all our student workers for programming.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, September 1991.

[2] B. Raman, S. Agarwal, Y. Chen, M. Caesar, W. Cui, P. Johansson, K. Lai, T. Lavian, S. Machiraju, Z.M. Mao, G. Porter, T. Roscoe, M. Seshadri, J. Shih, K. Sklower, L. Subramanian, T. Suzuki, S. Zhuang, A.D. Joseph, R.H. Katz, and I. Stoica, "The SAHARA Model for Service Composition Across Multiple Providers," *Pervasive Computing*, August 2002.

[3] D.G. Leeper, "A Long-Term View of Short-Range Wireless," *IEEE Computer*, vol. 34, no. 6, pp. 39–44, June 2001.

[4] J. Rabaey, J. Ammer, J. da Silva Jr., D. Patel, and S. Roundy, "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking," *IEEE Computer*, vol. 33, no. 7, pp. 42–48, July 2000.

[5] J.M. Kahn, R.H. Katz, and K.S.J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 271–278.

[6] L. Doherty, B.A. Warneke, B.E. Boser, and K.S.J. Pister, "Energy and performance considerations for Smart Dust," *International Journal of Parallel and Distributed Systems and Networks*, vol. 4, no. 3, pp. 121–133, 2001.

[7] G. Asada et al., "Wireless integrated network sensors: Low power systems on a chip," in *Proceedings of ESSCIRC '98*, 1998.

[8] J. Hill, R. Szewczyk, A. Woo, S Hollar, D.E. Culler, and K.S.J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.

[9] E. Shih, P. Bahl, and M. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proceedings of ACM MobiCom 2002, Atlanta*, September 2002.

[10] J. Elson, L. Girod, and D. Estrin, "Short Paper: A Wireless Time-Synchronized COTS Sensor Platform, Part I: System Architecture," in *Proceedings of the IEEE CAS Workshop on Wireless Communications and Networking*, September 2002.

[11] M. Stemm, P. Gautier, D. Harada, and R. H. Katz, "Reducing power consumption of network interfaces in hand-held devices," in *International Workshop on Mobile Multimedia Communications (MoMUC-3), Princeton*, October 1996.

[12] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: application driver for wireless communications technology," *Computer Communication Review*, pp. 20–41, 2001.

[13] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, December 2002.

[14] L. Girod and D. Estrin, "Robust range estimation using acoustic and multimodal sensing," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, vol. 3, pp. 1312–1320.

[15] F. Bennett, D. Clarke, J.B. Evans, A. Hopper, A. Jones, and D. Leask, "Piconet - embedded mobile networking," *IEEE Personal Communications*, vol. 4, no. 5, pp. 8–15, October 1997.

[16] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T.F. Knight, R. Nagpal, E. Rauch, G.J. Sussman, and R. Weiss, "Amorphous computing," *Communications of the ACM*, vol. 43, no. 5, pp. 74–82, 2000.

[17] M. Weiser et al., "The ParkTab Ubiquitous Computing Experiment," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996.

[18] J. Tourrilhes, Magalhaes L., and C. Carter, "On-Demand TCP: Transparent peer to peer TCP/IP over IrDA," in *Proceedings of ICC 2002*, 2002.

[19] S. Irani, S. Shukla, and R. Gupta, "Competitive analysis of dynamic power management strategies for systems with multiple power saving states," in *Design Automation and Test Conference, DATE 2002*, 2002.

[20] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *IEEE International Solid-State Circuits Conference*, February 2000, pp. 294–295.

[21] T. Simunic, L. Benini, P.W. Glynn, and G. De Micheli, "Dynamic power management for portable systems," in *Mobile Computing and Networking*, 2000, pp. 11–19.

[22] M. Weiser, B. Welch, A.J. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Operating Systems Design and Implementation*, 1994, pp. 13–23.

[23] L.M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *IEEE INFOCOM*, 2001.

[24] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Mobile Computing and Networking*, 2001, pp. 61–69.

[25] M.K. Marina and S.R. Das, "Routing performance in the presence of unidirectional links in multihop wireless networks," in *Proceedings of the ACM MobiHoc 2002*, 2002.

[26] R. Brodersen, "A Multimedia Communication System Providing Wireless Access (Infopad)," Department of Electrical Engineering and Computer Sience, University of California at Berkeley, 1995.

[27] D.B. Johnson and D.A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996.

[28] F. Siegemund, "Spontaneous interaction in ubiquitous computing settings using mobile phones and short text messages," in *Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings, Ubicomp 2002*, September 2002.