A Bluetooth-based Sensor Node for Low-Power Ad Hoc Networks

Jens Eliasson, Per Lindgren, Jerker Delsing
Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
Luleå, Sweden
Email: {jens.eliasson, per.lindgren, jerker.delsing}@ltu.se

***Abstract*— TCP/IP has recently taken promising steps toward being a viable communication architecture for networked sensor nodes. Furthermore, the use of Bluetooth can enable a wide range of new applications, and in this article, an overview of the performance and characteristics of a networked sensor node based on TCP/IP and Bluetooth is presented. The number of Bluetooth-enabled consumer devices on the market is increasing, which gives Bluetooth an advantage compared to other radio technologies from an interoperability point of view. However, this excellent ability to communicate introduces disadvantages since neither TCP/IP nor Bluetooth were designed with resource-constrained sensor nodes in mind. We, however, argue that the constraints imposed by general purpose protocols and technologies can be greatly reduced by exploiting characteristics of the communication scheme in use and efficient and extensive use of available low-power modes. Furthermore, we claim that a Bluetooth-enabled networked sensor node can achieve an operating lifetime in the range of years using a total volume of less than 10 cm^3. The Mulle Embedded Internet System (EIS), along with its advanced power management architecture, is presented as a case-study to support the claims.**

***Index Terms*— Mulle, EIS, Bluetooth, TCP/IP, sensor networks, PAN, power consumption, low-power design, motes.**

I. INTRODUCTION

The first Bluetooth [1] specification was developed in 1994 by Ericsson Mobile Platforms. It was initially designed to be a cable replacement technology, capable of allowing devices to communicate wirelessly without the need for time-consuming configurations. The Bluetooth Special Interest Group (SIG) [2], which was created in 1998, allowed a large number of companies to work together in order to make the Bluetooth specification an open and widely spread standard. The first specification versions (1.0 and 1.0b) had many problems with interoperability, making it difficult to create connections between devices from different manufacturers. Following versions (1.1 and 1.2) resolved many of these issues. Consumers now enjoy the benefits of owning products, from a wide range of different manufacturers, that can communicate with each other. Mobile phones and laptops were the first product types where Bluetooth became the de-facto standard for low bandwidth wireless communication. Today, the current specification has reached version 2.1 [3] with Enhanced Data Rate (EDR). The maximum bandwidth is 2.1 MBit/s and the price has dropped below $3 for a Bluetooth chip. This has contributed to its popularity and Bluetooth can now be found in a large number of consumer electronic products. Recently, Bluetooth also adopted the Wibree [4] specification from Nokia, enabling Bluetooth to be used as a wireless media for sensors located in cheap accessories for sports, entertainment, and health-care.

Shortly after Bluetooth started to increase in popularity as a technology for cable replacement and began to provide a base for wireless Personal Area Networks (PANs), interest also emerged in the sensor networks research community [5]. The frequency hopping approach makes Bluetooth relatively immune to electromagnetic interference [6], and the large number of Bluetooth-enabled consumer electronic devices provides an infrastructure that can be utilized by sensor nodes to achieve global Internet connectivity.

Bonnet et al. reported in 2003 [7] that Bluetooth has some advantages, such as high throughput and resilience against interference, but also a number of disadvantages:

- *Device discovery is slow*, meaning that it can take a considerable amount of time and energy for devices to discover each other.
- *Keeping connections open is expensive*, meaning that Bluetooth has a higher power consumption while communicating than do other similar radio technologies.
- *A layered stack* prohibits fine-granularity time synchronization and restricts cross-layer optimizations.

However, Negri et al. proposed in [8] that the use of power-management policies can make Bluetooth suitable for a wider range of sensor networking applications than what could otherwise be achieved.

Networked sensor nodes, or *motes*, based on Bluetooth can be divided into two main categories: The first category only uses the Bluetooth radio hardware to set up connections. Data communication is performed using proprietary protocols directly on top of the radio layer. This approach requires only a fraction of the Bluetooth specification to be supported and gives increased capabilities to adjust parameters for a specific application. The second category adheres to the Bluetooth protocol stack specification, which requires a much more advanced stack. However, this increase in stack complexity and size allows nodes to communicate with virtually any Bluetooth-enabled device, e.g. computers and mobile phones. The use of TCP/IP can further increase the communication possibilities by allowing a sensor node

to transmit data directly to the Internet without the need of customized gateways or middle-ware applications.

In this paper, we present arguments concerning power consumption, accessibility, mobility and interoperability, showing the advantages of using TCP/IP over Bluetooth as a communication suite for networked sensors in Mobile Ad-hoc Networks (MANETs). The Mulle, a Bluetooth-enabled Embedded Internet System (EIS), [9] is used to demonstrate the performance and feasibility of the combination of Bluetooth and TCP/IP in sensor networking applications. We chose a pragmatic research approach, where claims and hypotheses are tested using real-world experiments.

The following Section presents Bluetooth in sensor networks and provides a brief overview of related work in the research area of wireless sensor networks. Section III provides a brief overview of issues concerning TCP/IP in sensor networks, followed by Section IV, which gives an in-depth discussion on the characteristics of Bluetooth-based networked sensor nodes. A detailed description of the Mulle embedded system, with its communication infrastructure and low-power architecture, is presented in Sections V and VI. Section VII presents experimental results from real-world measurements and finally, Sections VIII and IX outline future work and conclusions, respectively.

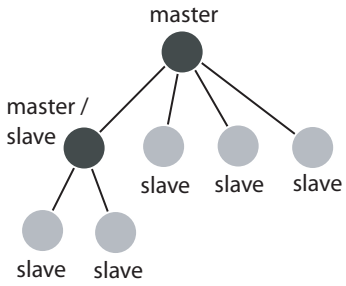

Figure 1. Connection-oriented network (Bluetooth)

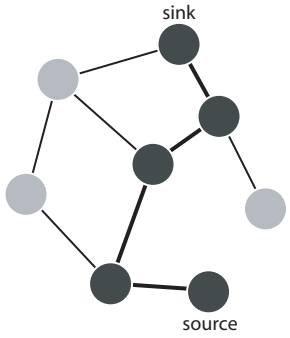

Figure 2. Broadcast multi-hop network (WSN)

II. Bluetooth for Networked Sensors

Sensor networks are emerging as a technology that can address a large number of application scenarios where it is crucial to perform measurements and take actions, such as sending an alarm or updating a control loop. Sensor networks can be divided into two categories: broadcast-based and connection-oriented. Traditionally, most sensor networks have been built using broadcast-enabled radios, but Bluetooth, a connection-oriented link topology, is growing in popularity as a design choice for smaller networks used in the vicinity of human users, e.g. for home-automation and health-care applications [10], [11].

Bluetooth is a radio technology which has been used in a number of sensor nodes. As mentioned earlier, Bluetooth-based sensor nodes can be roughly divided into two main categories: A) The first category only uses Bluetooth (IEEE 802.15.1) as a radio link technology, without utilizing the standardized higher layers of the Bluetooth protocol stack. This approach keeps the communication code footprint small, reduces software development complexity and allows fine-tuning of the behavior of the radio to suite a certain application. The drawback of utilizing proprietary protocols is that only node-to-node and node-to-gateway communication is possible. In this category, we find sensor nodes such as the BTnode [12] from ETH Zurich [13] and the iMote [14] from Intel [15]. To enable sensor networks based on proprietary protocols to communicate with IP-based networks, such as the Internet, a gateway is required to convert the proprietary protocol used within the sensor network to TCP/IP. The approach of using special gateways, or in some cases middle-ware applications, is widely used. The approach of not using Bluetooth Profiles and higher protocol stack layers also enables formation of true multi-hop networks, which are currently not supported by the Bluetooth specification.

B) Nodes in the second category, such as the Mulle node [9], conform to standardized protocols and Bluetooth profiles. This gives improved interoperability, since sensor nodes can communicate with Bluetooth-enabled consumer devices, e.g. computers, PDAs and mobile phones directly. However, the code footprint can increase substantially and thus imposes higher memory and resource requirements. This may prohibit the use of Bluetooth on low-end microcontrollers. However, many modern Bluetooth modules with built-in microcontrollers can provide a full Bluetooth stack. This reduces the amount of additional code required, and shortens software development time. Drawbacks with this design approach are that low-level access to the protocol stack layers can be limited. Less control over the communication system may result in higher power consumption and also prohibits link-level time synchronization.

The Bluetooth specification for Scatternets, a form of multi-hop network, is not fully developed to date, resulting in point-to-point network support only. This restricts the usability when creating sensor networks built upon standardized protocols, and proprietary solutions are often used to address this issue. Bluetooth, as a connection-oriented technology (Fig. 1), has some limitations in terms of scalability [7], [16]. Furthermore, even Scatternet-based networks can only support a relatively small number of nodes compared to what other, more specialized technologies such as 433-915 MHz broadcast radios or IEEE 802.15.4 can manage. Such dense networks, which are often referred to as Wireless Sensor Networks (WSNs) (Fig. 2),

have distinct characteristics compared to sensor networks based on Bluetooth. Research is performed on wireless sensor networks by a number of institutions [17], [18] and companies [19], [20].

III. IP Sensor Networks

IP as a communication base for sensor networks has had an increase in popularity during the last years. It was long believed that TCP/IP on resource constrained sensor nodes was not feasible, but recent research and development [21]–[23] have shown that TCP/IP can be utilized in a wireless sensor network. IP-based sensor networks enable a new range of applications where traditional proprietary networks fail to meet the requirements. A detailed description of issues concerning TCP/IP in sensor networks can be found in [24]. Below are some advantages and disadvantages of IP-based sensor networks:

A. Advantages

The Internet has, as of today, around 400 million hosts, and the possibility to integrate sensors directly to it may prove to be beneficial in a number of application areas. Essentially, there are two different ways of achieving this: the first approach is to have a dedicated gateway that converts IP-traffic to the proprietary protocol used by the sensor nodes. The other approach is based on IP-all-the-way, where sensor nodes in the network perform all communication using the TCP/IP protocol suite. This eliminates the need for a gateway, but can increase requirements on the nodes in terms of processing power and memory consumption. In MANET applications, where no infrastructure exists, sensor nodes must be able to communicate with the (mobile) devices that wish to obtain sensor data, without using a gateway.

Below are short descriptions of the most two important benefits of using TCP/IP in a sensor network:

1) Interoperability: This is one of the key points of using the IP protocol suite. The nodes can directly access the Internet using commercially available gateways, e.g. mobile phones, WiFi routers and computers. Mobile users can also access sensor data using standard PDAs, laptops or smart phones.

2) Mobility: Sensor nodes, using, for example, Bluetooth, can use mobile phones with GPRS/UMTS as access points. Since mobile phones are commonly used, the nodes can be deployed easily without the need to first build an expensive communication infrastructure. Mobile phones equipped with GPRS or UMTS currently provide better radio coverage, even in rural areas, than any other technology. The combination of Internet access using GPRS roaming allow sensor nodes to be deployed virtually anywhere, while still allowing users all over the world to access them directly, given that the appropriate mechanisms for node localization are in place. Such mechanisms can, for example, be static IP address assignment, static hostname combined with dynamic DNS updates, or by using a proxy with known URL to forward data from sensors to users.

B. Disadvantages

TCP/IP was designed to primarily be used by computers and not small sensor and actuator nodes. Sensor nodes, with very limited memory and processing resources, may not always be capable of using memory consuming protocols such as TCP. Below are the most significant drawbacks and issues concerning TCP/IP in sensor networking applications.

- *Code overhead:* Implementations of TCP/IP have shown that it is possible to use IP with only 6.3 kB of added code space, and less than 1 kB of RAM on an 8-bit microcontroller platform [25]. This may of course be a substantial part of the total resources available on the low-end microcontrollers often found on motes. More complex software also increases the processing load on the sensor node, thus increasing the power consumption as well.
- *Communication overhead:* Communication overhead is defined as the number of bytes (or percent) that the communication system adds to the payload when transmitting a packet. If a single byte is transmitted using TCP, the actual packet size will be 41 bytes, consisting of a 20 byte IP header [26] followed by a 20 byte TCP header [27]. This will result in a packet where less than 3 percent is useful data and the remaining 97% is overhead. One solution to this problem is data aggregation, where as much sensor data as possible is sent as a single packet, thus reducing the overhead. However, the end-to-end delay will increase. Another solution is to apply header compression techniques [28]. TCP has also been shown to be ill-suited for wireless networks [29], with bandwidth limitations, energy consuming re-transmissions and high memory requirements.

IV. Bluetooth-based Sensor Nodes

This Section presents an overview of, and comparison between, several Bluetooth-based sensor nodes. Other nodes, not included in the previous list, are the WISA [30] platform from ABB, which uses a modified IEEE 802.15.1 transceiver optimized for industrial control applications, and the Sanjay [31] mote. The WISA node is omitted due to the fact that it does not conform to the Bluetooth specification. Technical details on the Sanjay platform are, as of today, not available. Another interesting technology for wireless sensors is Wibree [4], which was recently adopted as a part of the Bluetooth specification. Wibree may further establish Bluetooth as a viable technology for wireless sensor networking applications.

A. ETH Zurich BTnode

The BTnode [12], [32] is a prototyping platform for ad-hoc networks. It was developed at ETH Zurich in a joint cooperation by the Computer Engineering and Networks Laboratory (TIK) department and Research Group for Distributed Systems. The BTnode, shown in Fig. 3, is composed of an Atmel ATmega128 microcontroller and two separate radios. The first radio is a low power Chipcon

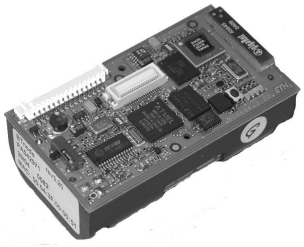

Figure 3. ETH Zurich BTnode

Figure 4. Intel mote

Figure 5. LTU EISLAB Mulle

CC1000 [33] ISM-band broadcast radio, the same as on Berkeley MICA2 [34] motes. This enables the BTnode to form multi-hop networks. The second radio system is a Zeevo ZV4002 [35] Bluetooth module. The BTnode has been used in a number of projects and is also used by universities for educational purposes. The BTnode software architecture is based on either Nut/OS [36] or TinyOS [37]. Wireless communication can be performed by the two radios independently. The Chipcon radio operates in the 433-915 MHz frequency range, and uses proprietary protocols. The Zeevo Bluetooth radio can, on the other hand, be operated using parts of the Bluetooth protocol stack. The BTnode Bluetooth stack supports the HCI, L2CAP and RFCOMM layers, enabling some interoperability with other Bluetooth devices. However, no support exists for the Bluetooth Service Discovery Protocol (SDP). This requires users to manually configure all connections and hence places the BTnode in between standards- based and proprietary architectures. Researchers at ETH Zurich have successfully created scatternets consisting of more than 70 BTnode rev3 devices, indicating that the scalability issues with the Bluetooth technology can be circumvented. However, the Bluetooth specification concerning scatternets is still unfortunately vague. There are still issues that are considered vendor-specific, which make is difficult to create networks with heterogeneous nodes.

B. Intel Mote

Intel has developed the Intel mote [38], or iMote, together with UCLA [39]. The iMote, shown in Fig. 4, is based on the Xscale microcontroller and uses a Zeevo TC2001 Bluetooth module. The platform is stackable, enabling a multitude of different sensors and power supplies to be attached to it. The operating system is TinyOS. No standard Bluetooth profiles are supported; instead, customized protocol layers written for TinyOS are used. These layers provide support for topology establishment and formation of both single and multi-hop networks. The use of TinyOS simplifies code reuse from other (non Bluetooth) mote platforms. More information about the iMote can be found in [40].

C. LTU EISLAB Mulle

The Mulle, shown in Fig. 5, is a Bluetooth-enabled sensor node [9]. It was originally developed at EISLAB, Luleå University of Technology (LTU) [41] but is now a commercial product from EISTEC AB [42]. The Mulle is based on a Renesas M16C/62 [43] microcontroller with 31 kB of RAM and 384 kB of flash memory, and its communication architecture (Fig. 6) supports the most commonly used Bluetooth protocols, e.g. HCI, L2CAP, SDP, BNEP, RFCOMM and PPP. This enables the Mulle to communicate with a large variety of devices, ranging from mobile phones and access points to computers and PDAs. The use of TCP/IP further increases interoperability since the Mulle can be connected to any IP-based network, such as the Internet, and hence allow users anywhere in the world to retrieve sensor data directly without the need of customized gateways or middle-ware applications. The lwBT Bluetooth stack [44] is developed in-house and is designed to have a small code and RAM footprint. The supported Bluetooth Profiles are: LAN Access Point (LAP), Dial-up Networking (DUN), Personal Area Networking (PAN) with all three roles: NAP, GN, PANU, and the Serial Port Profile (SPP). This extensive support for different Bluetooth Profiles enables a multitude of different network configurations to be created.

IP-support is provided by the lwIP [45] lightweight IP stack developed at SICS [46]. Support for the Multicast DNS and Service Discovery (mDNS-SD) protocol [47] provides automatic device and service discovery, and IP address allocation is normally performed by DHCP. A more detailed description can be found in [9], [48], and the Mulle Service Discovery architecture is summarized in [49].

D. Comparison of Bluetooth Sensor Nodes

This section contains a brief summary of the three Bluetooth sensor nodes mentioned previously. Each node has its own approach of using the Bluetooth technology for sensor networking purposes. The BTnode implements a dual radio approach, where the Bluetooth radio is complemented with a broadcast radio. Thus, it is capable of handling a wide

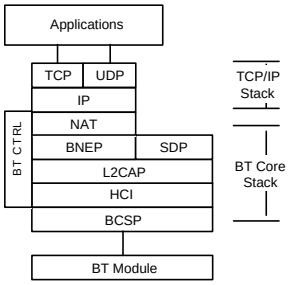

Figure 6. Mulle communication architecture

TABLE I.
COMPARISON OF NODES

	Mulle v2	BTnode	iMote
MCU frequency	10.0 MHz	7.37 MHz	12 MHz
RAM	31 kB	256 kB	64 kB
size	23.2x23.3 mm	58.2x32.5 mm	30x30 mm
CPU sleep, BT off	0.009 mW	9.9 mW	9 mW
CPU on, BT off	25.1 mW	39.6 mW	27 mW
CPU on, BT listen	28.4 mW	92.4 mW	62.1 mW

range of applications. The iMote uses a custom layer on top of Bluetooth radio links, which limits interoperability with consumer devices but enables formation of multi-hop networks. Finally, the Mulle uses only standardized protocols and Bluetooth Profiles, which prevents its use as a true WSN node, but enables communication with standard consumer devices. This makes the Mulle suitable for MANET sensor networking applications. See Table I for a brief overview on some interesting characteristics of each node. Mulle characteristics are taken from [50], while data for the BTnode and the iMote are taken from the Sensor Network Museum [51].

V. MULLE COMMUNICATION INFRASTRUCTURE

The Mulle communication architecture is based on a combination of Bluetooth and TCP/IP. A Mulle can communicate with: Bluetooth access points, mobile phones, PDAs and computers. An example of a typical Mulle network is shown in Fig. 8. Any Mulle that discovers a mobile phone will use the DUN profile to gain Internet access. When an Internet connection is established, the Mulle initiates its own PAN-NAP service, thereby enabling other Mulles and users to connect to it. In the figure, we see how one Mulle provides Internet access for Users #1 and #2 as well as for other Mulles. Experiments performed at EISLAB show that a Mulle PAN network can consist of at least 15 devices, and work is in progress to create Mulle networks with more than two times as many nodes. The Park mode is utilized for creating Piconets consisting of more than seven slaves, as well as for reducing the power consumption. When a Bluetooth slave enters Park mode, it releases its Piconet active member (AM) address. The seven AM addresses that are available can be shared by a large number of nodes in a time-division multiplexed fashion. Once a Mulle has established a network connection, it uses the Multicast DNS with Service Discovery (mDNS-SD) [47] service discovery protocol, to locate available services in its vicinity. In most

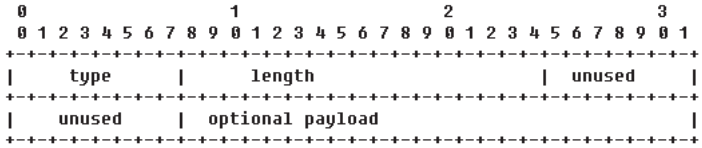

Figure 7. Ostmark Link Protocol

usage scenarios, the Mulle will scan the network for NTP servers and Mulle database servers. NTP is used by the Mulle to (re-)synchronize its Real-time Clock. When a Mulle discovers a database server, it will attempt to establish a TCP connection to it. Sensor data is then transmitted from the Mulle to the server where it is stored. Users can also log in to the server and view sensor data in real-time. The following protocols are supported by the Mulle communication architecture: IP, TCP, UDP, ICMP, DHCP, DNS, mDNS-SD, NAT-PMP, IGMP, NTP and HTTP. The Mulle also supports dynamic DNS updates, which together with the NAT Port Mapping Protocol (NAT-PMP), enables it to advertise available services globally on the Internet even when operating behind a firewall.

The Mulle uses the Ostmark Link Protocol (OLP) when transmitting sensor data over TCP. OLP (Fig. 7) is a simple protocol developed at EISLAB and consists of a 5 byte header with an optional payload. The *type* field, bits 0 to 7, indicates the type of packet, e.g. LOGIN or SENSOR_DATA. Bits 8 to 23 forms the *length* field in little endian, which tells the number of bytes of payload that are appended to the 5 byte header. Bits 24 to 31 and 32 to 39 are reserved for future use and could be used for encryption purposes. This simple packet format is used for all Mulle sensor communication, i.e. Mulle-to-Mulle and Mulle-to-user.

VI. MULLE LOW-POWER ARCHITECTURE

The Mulle software is based on a few major components: Bluetooth- and IP-stacks, a Hardware Abstraction Layer (HAL) which encapsulates low-level hardware behavior, an optional Real-time Operating System (RTOS), RTXC [52], and the sensor system. To resolve the issue of having multiple software components trying to utilize different low-power modes simultaneously, a new type of power management was developed for the Mulle: the energy-aware Task Manager [53]. The Task Manager, as shown in Fig. 9, coordinates all subsystems in order to efficiently conserve energy resources, both fixed and renewable. Solar cell support is successfully implemented and tested in real-world trials. The Task Manager consists of an API in the C programming language, which is to be used by all energy-aware software components and a context- and energy-aware scheduler. The scheduler uses a set of rules, which consists of alarm levels and various operations' energy consumption, an activation schedule, and status information from the main components to dynamically change the MCU clock frequency, and the usage of *stop_mode()*. This mode stops all MCU clocks and enables the Mulle to reduce

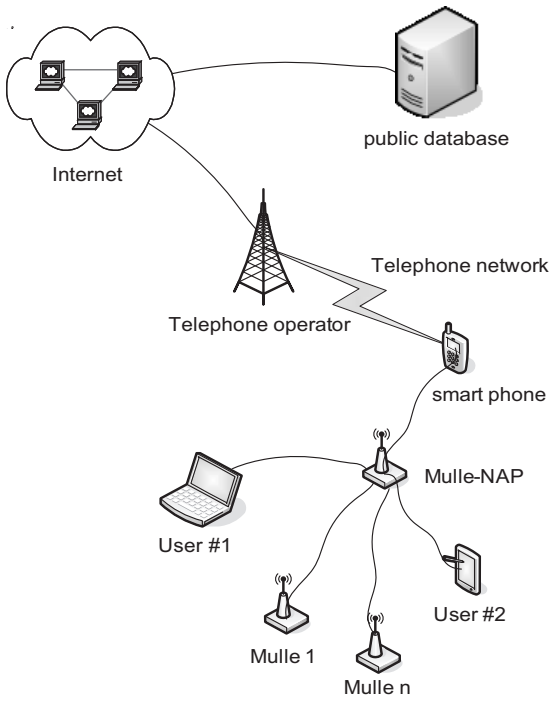

Figure 8. Mulle communication infrastructure

its power consumption by three orders of magnitude. The activation schedule controls the duty-cycle based operation of the Bluetooth subsystem. In other words, it controls how often the Bluetooth module is powered on and the sampling interval for the sensor system.

To address the issues mentioned in Section II, the Mulle uses a combination of different techniques:

- *Device discovery*: Bluetooth specifications 1.2 and newer introduced two faster and more efficient device discovery mechanisms: *enhanced inquiry* and *interlaced inquiry scan*. These two new features reduce the maximum inquiry time from 10 to 5 seconds and can perform connection establishment in half the time compared to Bluetooth 1.1 and older. The Mulle is therefore equipped with a Bluetooth 2.0 compatible module from Mitsumi. The Mulle also caches Bluetooth addresses from devices, e.g. access points, in its vicinity and thus enabling the number of required inquiry scans to be minimized.
- *Keeping connections*: The Mulle makes aggressive use of sniff and park low-power modes when connected. It can also operate the Bluetooth module with the MCU in sleep mode by using the BCSP protocol [54]. The largest reduction of energy consumption can be achieved by powering down the Bluetooth system when it is not used, i.e. duty-cycling.
- *Layered stack*: The use of BCSP allows the host to interface different layers directly, which in turn simplifies low-power management. BCSP also enables the MCU to enter sleep mode while the Bluetooth module is connected. This is possible because the packets that are sent over the UART may be lost, due to the time it takes for the MCU to wake up. They are then retransmitted. The Bluetooth module is also programmed to generate an interrupt which will wake the MCU up from sleep whenever data is received over the radio link.

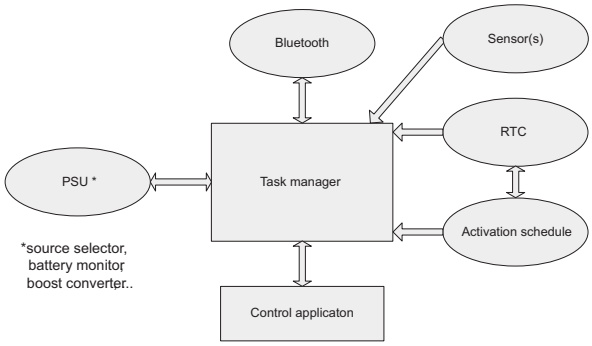

Figure 9. Mulle low-power architecture

The Mulle communication low-power architecture uses three different levels of synchronization granularity in order to minimize delay while still allowing energy-efficient operation. The three levels of granularity are *fine*, *medium* and *course*. Their respective power consumption and communication delay can be seen in Table II.

- *Fine:* Bluetooth supports bandwidths up to 3 MBit/s, but the maximum bandwidth a Mulle can deliver is 57.6 Kbit/s due to baud rate limitations in the UART that connects the Bluetooth module and the microcontroller. This allows the Mulle to use the *Sniff mode* to reduce the Bluetooth bandwidth and thus conserve energy efficiently.
- *Medium:* When only sporadic transmissions are required, but the Mulle still needs to be connected, it uses the *Park state*. This mode enables a Piconet to have more than 7 clients and can reduce the power consumption drastically.
- *Course:* When the Mulle is not required to maintain a connection, it can disconnect and turn off the power to the Bluetooth module, thus reducing the power consumption by three orders of magnitude.

To support a large variety of applications and different low-power requirements, the Mulle has three different operational modes. Each mode models a certain application class. Classes are separated by a) how the Mulle should sample its sensors and b) the requirements on the communication system. Where WSN nodes typically are optimized for node-to-node and node-to-gateway communication, the Mulle is designed to be used by human users. This fact reflects how the Mulle normally operates [48], with a user-oriented approach.

As shown in Fig. 10, the Task Manager API consists of four functions. The *pwrmgr_inform()* function is called whenever a subsystem changes state, e.g. when the Bluetooth system goes from Idle to Listen state. The Bluetooth subsystem then calls the *pwrmgr_inform()* function in order to tell the power manager that a change of state just occurred. The *pwrmgr_action()* function is called by the HAL

```
err_t pwrmgr_inform(uint8 *event);
err_t pwrmgr_action(uint8 clk);

err_t pwrmgr_setMCUspeed(uint8 clk_div);
uint8 pwrmgr_getMCUspeed(void);
```

Figure 10. Task Manager API

system, and triggers the Task Manager to see if the system can enter any low-power mode. This design approach keeps each subsystem unaware of what state other systems are in, while still being able to efficiently activate available system power-save modes. The *pwrmgr_setMCUspeed()* and *pwrmgr_getMCUspeed()* functions are used to indicate the minimum clock frequency a subsystem needs and read the current MCU clock frequency. Below are short descriptions of all available operating modes:

- *Passive mode* When a Mulle is in Passive mode, it has its Bluetooth transceiver on in listening mode. Unused components are powered down in order to conserve energy and the MCU is typically in *stop_mode*, while the Bluetooth module is in listening mode.
- *Active mode* In this mode, the Mulle initiates outgoing connections. Once a connection is established, the Mulle starts to stream sensor data to a user or database server. The power consumption usually depends on the specific type of sensor(s) attached to the Mulle.
- *Time-synchronous mode* This mode, which combines the two previous modes using an activation schedule, is a form of distributed duty-cycling and allows the Mulle to conserve a considerate amount of energy. The activation schedule can be modified dynamically, and allows users to make trade-offs between system lifetime and end-to-end delay. The Mulle spends most of its time, typically 95-99 %, in sleep mode where it consumes less than 10 μW. Periodically it wakes up to either: listen for incoming connections or establish its own outgoing connections.

The decision of which of these modes to use is highly application specific and must always be decided at compile time.

VII. Experimental Results

In order to evaluate the performance of the proposed low-power techniques, a series of real-world tests were performed. These tests were constructed in a way that reflects how a sensor node behaves during normal operation, i.e. periodically waking up from sleep mode in order to sample its sensor or to transmit sensor data. Table II provides a short summary of a Mulle's power consumption and system delay in various modes. High-efficiency voltage regulators provide the system with 3.3 V. In all measurements, a Real-time Clock was used to wake the MCU every 1000 ms in order to measure the system current consumption using a Dallas DS2782 high-precision battery monitor chip, which measures the voltage drop over a small (0.020 Ohm) series resistor with a sampling frequency of 18 kHz. In order to

TABLE II.
Mulle v2 power consumption

Mode	Delay	Power
All systems sleep	-	0.0089 mW
MCU 10.0 MHz, BT off	-	25.1 mW
MCU 5.0 MHz, BT off	-	16.8 mW
MCU 2.5 MHz, BT off	-	10.1 mW
MCU 1.25 MHz, BT off	-	7.3 mW
MCU sleep, BT listen	2-12 s.	3.3 mW
MCU sleep, BT active	-	132.9 mW
MCU sleep, BT sniff (210 slots)	131 ms.	27.8 mW
MCU sleep, BT sniff (2010 slots)	1256 ms.	9.1 mW
MCU sleep, BT parked (18 slots)	13 ms.	24.9 mW
MCU sleep, BT parked (200 slots)	130 ms.	8.8 mW
MCU sleep, BT parked (4094 slots)	2560 ms.	6.0 mW
MCU sleep, BT parked (8094 slots)	5000 ms.	6.0 mW

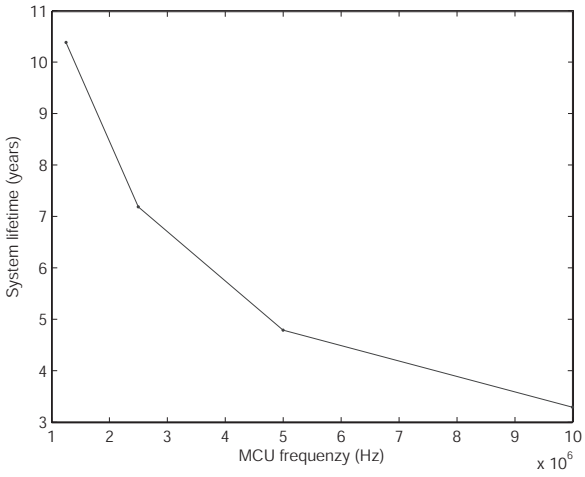

Figure 11. Extrapolated system lifetime vs. MCU frequency

get a good reading of the average current for each mode, the system was measured over a time period of approximately 20-30 minutes. The current measurement resolution was chosen to be 80 μA. One drawback to this solution is that the energy-consumption caused by the measurement is included in the total result. The communication delay is calculated from the number of time slots that are in between two consecutive timeslots in which the Bluetooth module has its radio active.

Fig. 11 shows the extrapolated node lifetime versus MCU clock frequency during active periods. A duty cycle of 1% for the MCU was chosen. We see here that by using a standard Saft LS14500 2250 mAh Lithium battery, the Mulle can utilize the MCU for 600 ms per minute and live 10.4 years when running at 1.25 MHz and 3.3 years when running at full speed. Note that these values exclude the power consumption for attached sensors, since some sensors, e.g. GPS-receivers, can consume more power than the sensor node itself. Internal leakage and aging of the battery cells are omitted from the calculations. Thus, the extrapolation is only indicative to actual lifetime of the system.

All measurements with the Bluetooth module in connected state, using Sniff or Park low-power modes, were performed with full TCP/IP communication. DHCP was

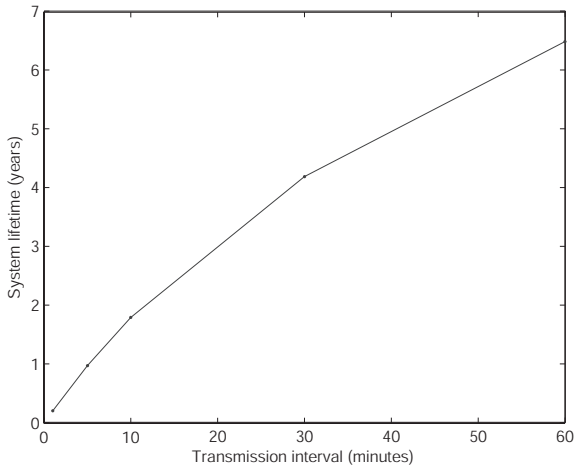

Figure 12. Extrapolated system lifetime vs. transmission interval

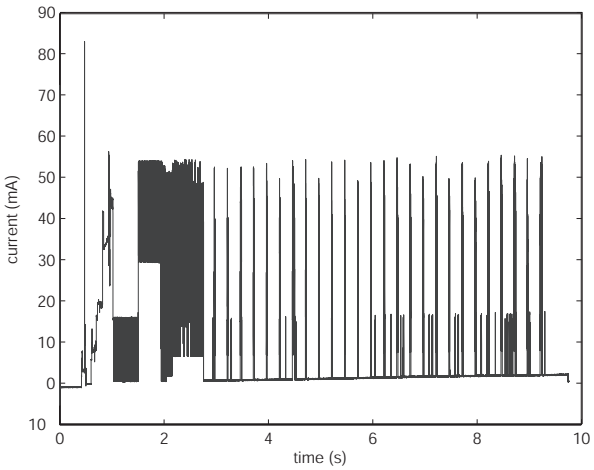

Figure 14. Transmission current consumption for optimized system

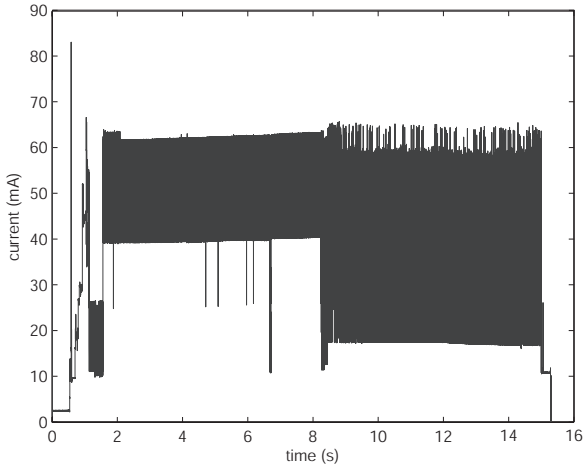

Figure 13. Transmission current consumption for un-optimized system

used for automatic IP address allocation and NTP for programming the on-board RTC with the correct date and time. Dynamic DNS updates were performed against a DNS server on the Internet, and the use of the mDNS-SD and NAT-PMP protocol enabled users to navigate to the Mulle using a standard web browser.

Fig. 12 shows extrapolated lifetimes for a Mulle using the Bluetooth module at different transmission intervals in order to send data. The cost of initiating a connection is measured to be 259 mWs, and then an extra 2.2 mWs for each additional 128 bytes of payload. The payload accumulates with 2 bytes/second and is the reason that the lifetime does not scale linearly with the transmission interval but instead is bounded by the amount of sensor data that need to be transmitted. Fig. 12 also shows that data aggregation extends the lifetime by reducing overhead since connection establishment dominates the power consumption for the lower intervals. A higher interval leads to higher efficiency even though the end-to-end delay increases.

The power consumption of a Mulle node performing a data transmission without any low-power techniques applied is shown in Fig. 13. The total energy consumption is measured to be 1920 mWs. The figure shows a boot of the Bluetooth module followed by an Inquiry scan. After the scan is successful, the Mulle performs an L2CAP connection, BNEP connection and a DHCP discover. When an IP address is allocated by DHCP, the NTP protocol is used to obtain the correct date and time. Finally, a TCP connection is performed against a Mulle database server. A Mulle performing the same TCP communication, but without an Inquiry scan or DHCP discover is shown in Fig. 14. By caching addresses to use, the connection setup time can be minimized. Sniff mode and Dynamic Frequency Scaling (DFS) are also used in order to reduce the power dissipation. The second connection only consumes 246 mWs, compared to the first which consumes 1920 mWs. The total energy consumption is reduced by over 87%. Measurements for Figures 13 and 14 were performed using a high-performance Tektronix TDS 7254 digital oscilloscope with a sample rate at 25 kHz using a 10-bit resolution. A small resistor of 5.00 ohms was connected in series with the Mulle, and the oscilloscope was measuring the voltage drop over the resistor. The current measurement resolution was set to 40 μA. A Keithley 6487 Picoammeter high-precision current measurement instrument was used to measure the static system current consumption in sleep mode. All data analysis were performed using MATLAB.

In 2003, Leopold et al. reported that the power consumption was 89 mW for a BTnode with the Bluetooth module in pagable and inquirable mode, and approximately 136 mW to maintain a connection [16]. This high power consumption was one of the reasons behind the conclusion that Bluetooth is ill-suited for sensor networking applications. However, a Mulle can perform the same operations today consuming only 3.4 mW and 7 mW, respectively. The next generation of the Mulle platform is designed with a projected power consumption of less than 0.5 mW when using Sniff mode

in a connected state.

VIII. Future Work

Even though Bluetooth-based networked sensor nodes can have the power consumption greatly reduced, there are still issues when it comes to scalability. Experiments performed have shown that it is feasible to create networks consisting of up to 15 nodes. However, many applications require more nodes than this and further research will be targeted to address this issue. Other limitations are long connection setup times, which restrict Bluetooth to be used in scenarios where nodes are traveling at high speeds, and the relatively high energy cost and long time delay of establishing a connection.

We also believe that, by extending the Bluetooth specification with support for broadcast channels, the issues of scalability, long connection setup times, and power consumption can be addressed much more efficiently.

IX. Conclusions

We conclude that Bluetooth, as a technology for ad-hoc networked sensor nodes, has made tremendous progress in terms of power consumption. Still, there are limitations on how large a Bluetooth network can be, but for moderate network sizes, we claim that Bluetooth is a feasible alternative for Personal Area (Sensor) Networks. In this article, we have demonstrated that the relatively high power consumption of Bluetooth-based sensor nodes, communicating using the TCP/IP protocol suite, can be reduced by orders of magnitude while still enabling interoperability with existing infrastructure. We claim that a system lifetime in the range of months to years is made feasible by exploiting characteristics of the communication scheme and efficient and extensive use of available low-power modes, requiring a total volume of less than 10 cm^3.

Acknowledgment

The authors would like to express their gratitude to Asanga Lokuyaddage for his support with electronics-related issues.

References

[1] “The Official Bluetooth Technology Info Site,” Sep 2007, http://www.bluetooth.com/.

[2] “Bluetooth Special Interest Group,” Bluetooth SIG, Sep 2007, http://www.bluetooth.org.

[3] “Bluetooth Core Specification v2.1,” Bluetooth SIG, Sep 2007, www.bluetooth.com/Bluetooth/Learn/Technology/Specifications.

[4] “Wibree,” http://www.wibree.com/, May 2007.

[5] L. Negri and L. Thiele, “Power management for bluetooth sensor networks,” in *EWSN*, 2006, pp. 196–211.

[6] J. Delsing, J. Ekman, J. Johansson, S. Sundberg, M. Backstrom, and T. Nilsson, “Susceptibility of sensor networks to intentional electromagnetic interference,” vol. 27, pp. 172–175, FEB 2006.

[7] P. Bonnet, A. Beaufour, M. B. Dydensborg, and M. Leopold, “Bluetooth-based sensor networks,” *SIGMOD Rec.*, vol. 32, no. 4, pp. 35–40, 2003.

[8] L. Negri and D. Zanetti, “Power/Performance Tradeoffs in Bluetooth Sensor Networks,” vol. 9, pp. 236b– 236b, JAN 2006.

[9] J. Johansson, M. Völker, J. Eliasson, Å. Östmark, P. Lindgren, and J. Delsing, “MULLE: A Minimal Sensor Networking Device - Implementation and Manufacturing Challenges,” in *IMAPS Nordic 2004*, Helsing”or, Denmark, September 2004, pp. 265–271.

[10] S. Moushumi, A. Shameem, S. Ahamed, M. Haque, and A. Khan, “Healthcare Aide: Towards a Virtual Assistant for Doctors Using Pervasive Middleware,” in *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW06)*, Baltimore, MD, USA, 2006.

[11] “Designing wireless interfaces for patient monitoring equipment,” Noel Baisa, Apr 2005, http://rfdesign.com/mag/radio_designing_wireless_interfaces/.

[12] “A Distributed Environment for Prototyping Ad Hoc Networks,” http://www.btnode.ethz.ch/, September 2006.

[13] “Swiss Federal Institute of Technology,” www.ethz.ch/, May 2007.

[14] “Intel motes,” www.intel.com/research/exploratory/motes.htm, Sept 2006.

[15] “Intel corporation,” 2007, http://www.intel.com/.

[16] M. Leopold, M. B. Dydensborg, and P. Bonnet, “Bluetooth and sensor networks: A reality check,” in *SenSys ’03*, Los Angeles, Californa, USA, November 2003.

[17] “Center for Embedded Networking Senses,” www.cens.ucla.edu/, September 2006.

[18] “Networked embedded systems,” Berkely, 2007, http://webs.cs.berkeley.edu/nest-index.html.

[19] “Crossbow Technology Inc.” 2007, http://www.xbow.com/.

[20] “Moteiv Corp.” 2007, http://www.moteiv.com/.

[21] A. Dunkels, T. Voigt, and J. Alonso, “Making TCP/IP Viable for Wireless Sensor Networks,” in *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, JAN 2004. [Online]. Available: http://www.sics.se/˜adam/ewsn2004.pdf

[22] A. Östmark, C. Öhult, J. Eriksson, P. Lindgren, and J. Delsing, “A wireless network of eis devices [sensor networks],” in *Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference*, vol. 2, May 2004, pp. 1199 – 1202.

[23] A. Dunkels, “Programming Memory-Constrained Networked Embedded Systems,” Ph.D. dissertation, Swedish Institute of Computer Science, FEB 2007.

[24] ——, “Towards TCP/IP for Wireless Sensor Networks,” 2005. [Online]. Available: http://eprints.sics.se/72/01/dunkels05towards.pdf

[25] ——, “Full TCP/IP for 8 Bit Architectures,” in *Proceedings of the First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*. San Francisco: USENIX, MAY 2003. [Online]. Available: http://www.sics.se/˜adam/mobisys2003.pdf

[26] “RFC 791 - Internet Protocol,” http://www.ietf.org/rfc/rfc791.txt, 1981.

[27] “RFC 793 - Transmission Control Protocol,” http://www.ietf.org/rfc/rfc793.txt, 1981.

[28] M. Degermark, M. Engan, B. Nordgren, and S. Pink, “Low-loss tcp/ip header compression for wireless networks,” *Wirel. Netw.*, vol. 3, no. 5, pp. 375–387, 1997.

[29] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, “TCP Performance Issues over Wireless Links,” in *IEEE Communications Magazine*, vol. 39, July 2001, pp. 52–58.

[30] D. Dzung, J. Endresen, C. Apneseth, and J.-E. Frey, “Design and Implementation of a Real-Time Wireless Sensor/Actuator Communication System,” in *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005)*, SEP 2005.

[31] C. Dethe, D. Wakde, and C. Jaybhaye, "Bluetooth based sensor networks issues and techniques," in *Proceedings of the First Asia International Conference on Modelling & Simulation (AMS'07)*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 145–147.

[32] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald, "Next-generation prototyping of sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 291–292.

[33] "Chipcon cc1000," Chipcon, 2007, http://www.chipcon.com/.

[34] "MICA2 mote," Crossbow Tech. Inc., 2007, http://www.xbow.com/Products/productdetails.aspx?sid=174.

[35] "Zeevo ZV4002," Zeevo Inc., 2007, http://www.zeevo.com/.

[36] "Real-time Operating System," http://www.ethernut.de/en/software.html, June 2007.

[37] "Open-source Operating System," http://www.tinyos.net, January 2007.

[38] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel, "The Intel Mote platform: a Bluetooth-based sensor network for industrial monitoring," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 61.

[39] "University of Californa, Los Angeles," http://www.ucla.edu, January 2007.

[40] R. Kling, R. Adler, J. Huang, V. Hummel, and L. Nachman, "Intel Mote: using bluetooth in sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, Baltimore, MD, USA, 2004, pp. 318 – 318.

[41] "Luleå University of Technology, Embedded Internet System Laboratory," Sep 2007, http://www.csee.ltu.se/eislabfo.

[42] "Embedded Internet System technology bothnia AB," EISTEC AB, 2008, http://www.eistec.se/.

[43] "Microcontroller m16c/62m," http://www.renesas.com/eng/, June 2004, renesas Technology Corporation.

[44] C. Öhult, "lwBT - a lightweight Bluetooth stack," http://www.csee.ltu.se/~conny/lwBT/, February 2005.

[45] A. Dunkels, "lwIP - a lightweight TCP/IP stack," February 2005. [Online]. Available: http://www.sics.se/~adam/lwip/

[46] "Swedish Institute of Computer Science," http://www.sics.se, January 2007.

[47] "Multicast DNS," http://www.multicastdns.org/, September 2006.

[48] J. Eliasson, M. Lundberg, and P. Lindgren, "Time Synchronous Bluetooth Sensor Networks," in *IEEE Consumer Communications and Networking Conference (CCNC 2006)*, vol. 1, Las Vegas, Nevada, USA, January 2006, pp. 336–340.

[49] Å. Östmark, J. Eliasson, P. Lindgren, A. van Halteren, and L. Meppelink, "An Infrastructure for Service Oriented Sensor Networks," *Journal of Computers (JCP)*, pp. 20–29, August 2006. [Online]. Available: www.academypublisher.com/jcp/vol01/no05/jcp01052029.pdf

[50] "Mulle development information site," EISLAB, 2007, http://www.csee.ltu.se/~jench/mulle.html.

[51] "SNM - The Sensor Network Museum," 2007, http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum.

[52] "RTXC. Quadros Systems, Inc," http://www.quadros.com/, May 2005.

[53] Jens Eliasson and Per Lindgren and Jerker Delsing and Simon J. Thompson and Yi-Bing Chen, "A Power Management Architecture for Sensor Nodes," in *IEEE Wireless Communication and Networking Conference - WCNC 2007*, March 2007, pp. 3008–3013.

[54] "BCSP - BlueCore Serial Protocol," March 2005, http://www.csrsupport.com/index.html.

Jens Eliasson received his M.Sc. in Computer Science and Engineering at Luleå University of Technology (LTU), Sweden 2003. In the end of 2003, he joined the department of Computer Science and Electrical Engineering at LTU as a research engineer. Now he is persuing his PhD. degree at EISLAB, Luleå University of Technology, in the field of embedded systems and sensor networks. His main research area is low-power design of Embedded Internet Systems (EIS).

Per Lindgren holds the position of senior lecturer/assistant professor after defending his PhD thesis in January 2000, and is leading education and research in the area of Computer Engineering/Embedded Systems at EISLAB, Luleå University of Technology. In January 2006 he received the Docent degree and became associate professor. He is currently heading a group of Ph.D. students in the area of embedded system design, with a focus on real-time, low-power software and hardware architectures for Embedded Internet Systems.

Prof. Jerker Delsing received the M.Sc. in Engineering Physics at Lund Institute of Technology, Sweden 1982. In 1988 he received the PhD. degree in Electrical Measurement at the Lund University. During 1985 - 1988 he worked part time at Alfa-Laval - SattControl as senior sensor specialist, with development of sensors and measurement technology. In 1994 he got the docent degree (associate prof) in Heat and Power Engineering. Early 1995 he was appointed full professor in Industrial Electronics at Luleå University of Technology where he currently is working as the scientific head of EISLAB, http://www.ltu.se/eislab. For the period 2004-2006 he was Dean of engineering faculty at Lulea University of Technology. His present research profile can be entitled "Embedded Internet Systems", EIS, with applications both to industrial, medical and sport. Since 1999 he is president of ITF (Instrument Tekniska Foreningen/Instrument Society of Sweden).