Bluetooth-Based Sensor Networks

Philippe Bonnet, Allan Beaufour, Mads Bondo Dydensborg, Martin Leopold
Department of Computer Science
University of Copenhagen
bonnet@diku.dk, beaufour@diku.dk, madsdyd@diku.dk, leopold@diku.dk

Abstract

It is neither desirable nor possible to abstract sensor network software from the characteristics of the underlying hardware components. In particular the radio has a major impact on higher level software. In this paper, we review the lessons we learnt using Bluetooth radios in the context of sensor networks. These lessons are relevant for (a) application designers choosing the best radio given a set of requirements and for (b) researchers in the data management community who need to formulate assumptions about underlying sensor networks.

1 Introduction

Sensor networks are emerging as the platform of choice for a new generation of monitoring applications. Unlike traditional computing platforms, sensor networks are application specific [4]. An important part of developing a sensor network infrastructure is thus to choose, or possibly design, the hardware and software components that will fit the application requirements.

A key component in a sensor network is the radio: it impacts not only performance and energy consumption but also the network topology and software design. The radio actually impacts the assumptions underlying the design of software components such as data dissemination or in-network processing services. In this paper, we report on the lessons we learnt using Bluetooth radios in the context of monitoring applications.

Let us first motivate our focus on Bluetooth. It was designed as a cable replacement technology with an emphasis on interconnecting devices from different vendors. So why consider it in the context of sensor networks? Bluetooth is a spread spectrum radio, like the radios used in the WINS prototypes from UCLA [14] or the new IEEE 802.15.4 standard [16]. Such radios are resilient to interferences, which is a desirable property for many applications (notably in the free 2.4 GHz band). Moreover, the mass production of Bluetooth components ensures both robustness and decreasing costs. Bluetooth is thus a reasonable option when designing a sensor network, but:

- What is the impact of Bluetooth on the network topology?
- What is the impact of Bluetooth on the application software?
- What is the impact of Bluetooth on performance and power consumption?

These questions are relevant for any kind of radio. They are important when designing a sensor network in the context of a given application: Given the application requirements, what is the best radio available? Today, Bluetooth is an alternative to the broadcast radio embedded in the Berkeley motes. Tomorrow, radios following the 802.15.4 standard will be another option. Conversely, these questions are relevant when developing (application independant) data management services for sensor networks: Given a radio component, what is a consistent set of assumptions for the data management service? Indeed, it is neither desirable (from an resource usage point of view) nor possible to abstract software components from the characteristics of the underlying hardware components.

We focused on Bluetooth-based monitoring applications in the context of the Manatee project at University of Copenhagen [13]. Our goals were (i) to assess the strengths and limitations of Bluetooth in the context of sensor networks, and (ii) to characterize the monitoring applications for which Bluetooth is well suited. We chose a pragmatic approach based on experiments with actual Bluetooth devices. We give a short description of the relevant aspects of Bluetooth and of the devices we used in Section 2.

This paper makes the following contributions, based on the lessons learnt from the Manatee Project ([9], [2], [8]):

- We describe the features of Bluetooth radios that are relevant for sensor networks, both from a hardware and a software viewpoint;
- We describe the impact of Bluetooth on sensor networks, both from a functionality and a performance viewpoint;
- We present the classes of applications for which Bluetooth is well suited.

2 Bluetooth Radio Component

A Bluetooth radio component is composed of hardware (the radio front-end together with a microcontroller) as well as software (the Bluetooth stack). In this Section, we describe the Bluetooth radios we used for our experiments. We then discuss the features of Bluetooth that are relevant for sensor networks.

2.1 Bluetooth Hardware and Software

We experimented with devices representing two different design options for the Bluetooth hardware [1]:

- **Single-chip**: The Bluetags developped by a danish start-up [11], are equipped with Bluecore: a single-chip from Cambridge Silicon Radio (CSR) integrating an RF transceiver, baseband circuitry and a proprietary 16-bit microcontroller (XAP2) [12] running the entire Bluetooth stack. Each device has 2 KiB[2] EEPROM memory for data storage and 512 Kib flash memory for embedded applications (the bluetooth stack and possibly additional software components).
- **Separate chips**: The BTnodes developped by ETH Zurich in the context of the Smart-Its project [15] are based on (i) a microcontroller (Atmel ATmega128L – an 8 bit microcontroller clocked at 7.4 MHz, with 4 KiB on chip memory and an external memory chip of up to 64 KiB) connected to (ii) a Bluetooth module (Ericsson ROK 101 007 – including both the RF transceiver, baseband circuitry and a microcontroller running the lower layers of the Bluetooth stack).

The single chip approach is very attractive as it promises reduced cost of production and reduced energy consumption. The two chip approach is flexible; it allows experimenting with changing generations of radio hardware and it allowed us to focus on the Bluetooth features that are relevant for sensor networks. While the lower layers of the Bluetooth stack focus on the spread spectrum radio, the higher layers provide a set of abstractions that make it possible to interconnect different devices from different vendors [6]. This is not a property of paramount importance in sensor networks tailored to operate together as one unit. Indeed resources are constrained and should not be wasted on unnecessary features. It was not possible to redefine the Bluetooth stack on the single-chip system from CSR. On the BTnodes, we defined a stripped down version of the Bluetooth stack: *tinyBT* integrated in the tinyOS operating system [5].
TinyBT provides access to a subset of the Bluetooth standard (its essential features of Bluetooth described in the next section), which results in flexibility for application development (there is no need to conform to high level abstractions tailored for interoperability) and reduced footprint (3 KiB for tinyBT as opposed to hundreds of KiB for the complete stack).

[1]See Xilinx's tutorial [18] for a complete description of the design space

[2]EIC standard 60027-2 defines KiB as 1024 bytes [3]

2.2 Bluetooth Features

Bluetooth operates in the 2.4GHz royalty free ISM band. It uses a Frequency Hopping Spread Spectrum (FHSS) scheme. Its essential features in the context of sensor networks can be summarized as follows:

- *Device Discovery.* In order for two devices to discover each other, they must be in two complementary states at the same time: *inquiry* and *inquiry scan.* The inquiring device continuously sends out "is anybody out there" messages hoping that these messages (know as ID packets) will collide with a device performing an inquiry scan. The inquiring device repeatedly sends short messages on different frequencies, while the device performing inquiry scan listens to one frequence for a while, before switching to another.
- *Connections.* Two devices need to be connected before they can exchange data. When two devices have discovered each other they are able to connect (they are in the *pageable* state). One device is then denoted as the master and the other as the slave (a slave is following the frequency hopping sequence dictated by its master).
- *Data Exchange.* A master and a slave exchange data as follows. A slave is only allowed to transmit to the master once the master has contacted it. Data is transmitted in slots of a fixed size of 625 μs : master and slave send alternatively.

- *Star Topology.* There are at most 7 slaves connected to a same master. There have been announcements about Bluetooth support for multihop network topologies where nodes can act both as masters and slaves (scatternets). But today, there is no product providing such support. We experimented with Bluetooth-based multihop networks using a dual-radio approach [9]: we equipped the BTnodes with two Bluetooth radios.

- *Power Saving Modes.* Bluetooth supports two power saving modes. In *park mode*, a slave has its radio turned off. It needs to reconnect to the master in order to exchange data. Alternatively a connection can be set in *sniff mode*: the connection between slave and master remains open and both devices can turn off their radio as long as they synchronize at given points in time.

We refer the interested reader to [6] for a complete description of Bluetooth.

3 Impact of Bluetooth

Both the features and the actual performances of Bluetooth radio components have an impact on application design. We review in this section the lessons we learnt in the Manatee project. Note that performance results only reflect the current generation of Bluetooth hardware. We emphasize the areas where we expect better hardware engineering to provide significant improvements.

3.1 Features

Let us first discuss how the Bluetooth features described in Section 2.2 impact application design.

Separated Channels

Bluetooth's frequency hopping scheme guarantees channel separation as opposed to the shared channel of broadcast radio components. Once two sensor nodes are connected, they exchange data without interferences from neighbor nodes. Separated channels enable thus very dense deployments. It also allows application designers to design their software without the unknown of interferences. This is particularly relevant in the context of aggressive power saving strategies where the sensor node is only awake for short periods of times during which interferences could have unpredictable consequences.

Channel separation requires sensor nodes to establish a connection before exchanging data. Many sensor network algorithms assume that a sensor node has some knowledge about its neighbors. Getting this information does not come for free using Bluetooth. It first requires discovering neighbor nodes, then establishing connections to each of them and then sending and receiving data. An improvement would consist in exchanging data during the device discovery phase, but this is not allowed by the Bluetooth standard.

Connection Graph

In a multihop network, the fact that sensor nodes need to establish a connection before they can send data means that a *connection graph* has to be established and possibly maintained. Such a connection graph is not fully connected. It is a restriction on the set of possible network routes, but it actually provides a form of topology control. Recent work has shown that topology control was an efficient power management strategy [19]. Topology control is also a necessary feature when managing a routing tree (as in TinyDB for exemple). In [9], we presented an algorithm for establishing a connection graph in a multihop network of dual-radio devices. There is however a need for efficient topology control algorithms on top of separated channel radios.

A question is whether a connection graph can be established and then maintained over the lifetime of the sensor network or whether connections should be limited to a few data exchanges. This depends on how efficient connection establishment and connection maintenances are on actual Bluetooth devices. We address this issue in the performance sub-section.

Power Management Strategies

The fact that connections are established between sensor nodes can also be used to define an aggressive radio driven power management strategy where a sensor node is turned off as long as there is no data available on any connection. We described the impact of such a power management strategy on the tinyDB data management service in [9]. Compared to the native application-driven strategy of tinyDB (where the application defines the sleep period of the sensor nodes), the radio driven approach guarantees that no message will be lost because the sleep period of neighbor sensor nodes overlap (in addition channel separation guarantees that no messages will be lost because of interferences).

Such a power saving strategy can be implemented using the sniff mode of the Bluetooth radio. However, Bluetooth's frequency hopping scheme requires a master and a slave to synhronize frequently. In sniff mode, the longest period during which devices can sleep is 40 seconds. This means that the sniff mode is not optimal for scenarios where devices could sleep for minutes or even hours.

Layered Stack

It would be desirable for applications to have access to the timing information managed by Bluetooth's baseband circuitry [3]. Also, applications could use information related to link management (acknowledgements of message transmission or information about message retransmissions). However, the layered structure of the Bluetooth stack prevents such cross layers optimizations.

3.2 Performances

We conducted experiments with the Bluetags and BTnodes described in Section 2.1. Here are the main lessons we learnt.

Device Discovery is Slow

Figure 1 shows the distribution of inquiry time obtained with a BlueTag device. The experiment duplicates the conditions of the experiments created by Kasten and Langheinrich [7]: two bluetooth devices are separated by one meter; 1500 inquiries are performed (10 seconds max) and we measure the time it takes for the devices to discover each other. The graph traces the number of times a device was discovered as a function of time. The result we obtain with the BlueTag is similar to the ones we have obtained with various bluetooth devices [8].

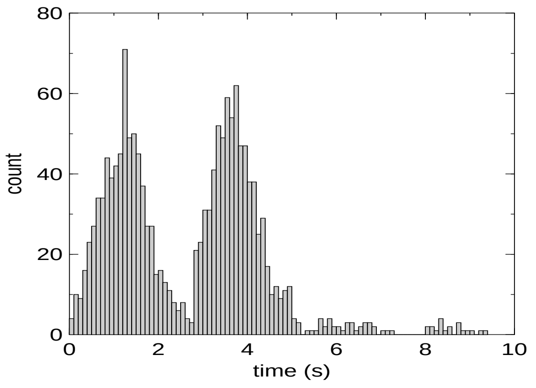

Figure 1: Inquiry Time Distribution (BlueTag)

The specification recommends a total inquiry period of 10.24 s to discover all devices in the neighborhood. Our study shows that most devices can be discovered within 5 seconds. This is too slow to perform device discovery whenever data has to be transmitted. This is also means that managing the connection graph takes a significant time (as new neighbor nodes must be discovered). We showed, in [2], that the distribution of inquiry time is dictated by the Bluetooth standard.

[3]The baseband circuitry manipulates timing information as it is responsible for synchronizing frequency hopping

Additional experiments, documented in [8], showed that inquiry time remains constant as the distance between two devices increase. As a result we can expect devices to discover each other at a distance of at least 20 meters.

Keeping Connections Opened is Expensive

We measured current draw and voltage[4] for different regimes of the BTnodes. Figure 2 summarizes our results.

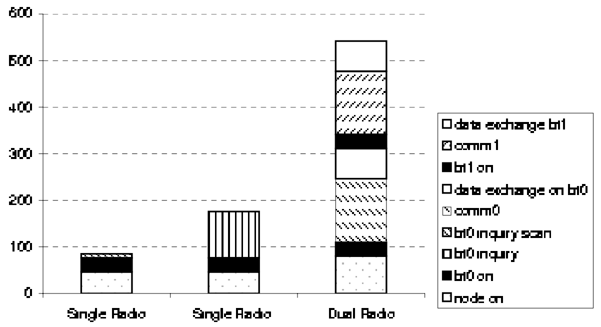

Figure 2: **Energy usage breakdown**. The graph presents sensor nodes with one radio (bt0) performing inquiry and inquiry scan as well as a dual-radio node (with bt0 and bt1). In the legend, comm0 and comm1 denote connections established on bt0 and bt1 respectively.

Our first goal was to measure energy consumption for an idle BTnode (in black on the figure). According to the manufacturer [1], the 8 MHz microcontroller can use up to 12 mA at 5 V (60 mW) in idle sleep mode. With a slightly lower clock frequency, we observe a lower energy consumption (about 46 mW). In the dual radio case, the base energy consumption is 80 mW. This is mostly due to the (unused but turned on) led on the additional Bluetooth radio, which consumes about 22 mW, as well as the voltage regulator and the serial voltage level converter.

Turning a Bluetooth radio on consumes about 30 mW extra (in idle mode). Performing an inquiry scan requires an additional 9 mW. Performing an inquiry costs approximately 100 mW: an order of magnitude more than performing an inquiry scan.

Nodes use about 136 mW to maintain connections. Additional experiments showed that putting a connection in sniff mode saves a marginal 5 mW. This suggests that some optimizations might have been missed in the design of the Bluetooth module.

[4]We used an input voltage of approximately 5V. We observed that voltage varied slightly during the experiments. We thus decided not to assume constant voltage to compute the energy consumption.

Once a connection is established sending or receiving data consumes an additional 65 mW when transferring at 6 KiB/sec when using a single radio and 5 mW for each radio when transferring 10 packets per second.

Throughput is high
We measured throughput on point-to-point connection between master and slave using the BTnodes.
For payloads of 20 bytes, we achieve a throughput of about 5 KiB/sec. For the maximum payload of 668 bytes, throughput is significantly higher: it varies from 6 to 35 KiB/sec. Here, the number of slots used for transmission and the resistance to noise have a significant impact.
Note that the throughput we achieve is far from the theoretical max, of up to 90 KiB/sec. First, the Tiny Bluetooth stack we used for these experiments accesses the Bluetooth module via the Btnode's UART: the maximum throughput supported by the UART is approximately 45 KiB/sec. Second, the Bluetooth module generates superfluous messages that take up bandwidth on the UART interface (hardcoded Ericsson string events). These can be optimized away on future generations of the Bluetooth modules.
The master is responsible for allocating the channel bandwidth for each slave in a multipoint connection. We setup the master to connect to a number of slaves and send packets in round-robin order on each connection. We measured the bandwidth on each connection. It would be expected that the total bandwidth stays the same and that each slave receives a fair share. Table 1 shows that the aggregate bandwidth drops considerably but that each slave receives a fair share of that bandwidth. It is peculiar that the aggregate bandwidth drops as much as 50 %—the lower layers of Bluetooth allow the master to switch between slaves without any additional overhead. The Bluetooth modules wastes in-air slots by not sending meaningful data.

	1	2	3
Aggregate	38.1 KiB/s	25.4 KiB/s	19.3 KiB/s
Pr. slave	38.1 KiB/s	12.7 KiB/s	6.4 KiB/s

Table 1: Throughput for an multipoint asymmetric DM3 connection, aggregate and individual bandwidth

Madden et al. [10] write that it is reasonable to expect that the Berkeley motes (equipped with the same microcontroller as the BTnode) transmit approximately 500 bytes per second. The BTnodes thus achieve a throughput, which is between one and two orders of magnitude higher for point-to-point connections and at the very least a factor of two higher for multipoint connections.

4 Bluetooth-Based Monitoring Applications

Our work shows that Bluetooth is well suited for applications with the following characteristics:

- **Network Topology**. Bluetooth is best suited for point-to-point connections. In case the application requires a sensor network composed of large clusters of sensor nodes, it is necessary to adopt a dual-radio approach to construct a Bluetooth multi-hop network.

- **Lifetime** All in all, a single-radio BTnode uses approximately 50 mW when idle and 285 mW when communicating, while these numbers are up to 80 and 450 mW for a dual-radio BTnode. If we compare with the Mica motes from UC Berkeley, these numbers are high. Indeed, Madden et al. [10] report 10 mW for for an idle node and 60 mW when communicating.

  The BTnodes consume five times more energy than the Mica motes doing nothing! This is due to the fact that the microcontrollers are placed in different sleep modes. The Mica motes favour power saving strategies where the applications manage themselves the time they spend in sleep mode. This way, the microcontroller can be put to sleep in *power save* mode, where only the external clock can send wake-up signals, i.e., the motes do not get data from sensors or from other nodes while the MCU is in sleep mode. The *power save* sleep mode is the most energy efficient.

  The BTnodes favour applications that are in sleep mode until (unpredicatable) events are received on the Bluetooth radio or the sensors. This precludes the use of the *power save* mode because events generated by the Bluetooth radio would be ignored. The microcontroller is best put in *idle* sleep mode until an event is received from the UART or from the clock. The *idle* sleep mode is however less energy efficient than the *power save* mode. We can estimate to 8 days the life expectancy of a BTnode in *idle* sleep mode with two standard AA batteries.

  Connection maintenance consumes a lot of energy, both on master and slave. Using the sniff mode does not make a significant difference.

These results suggest that connections should only be established for short periods of time.

- **Payload** A Bluetooth radio component enables a high throughput. As a consequence, it is possible to transmit large payloads (including audio and images).

These remarks suggest that the Bluetooth radio components are best suited for:

1. Applications involving a large number of mobile devices disseminate data between small clusters of fixed sensor nodes [2], e.g., hikers spreading sensor data across a national park (that cannot be covered by a single cluster) or zebras being monitored in the savanna [17].

2. Applications that are active over a limited time period, with few unpredictable bursts of very heavy network traffic. An example of such application could be a sensor network deployed to secure a building in a mounted operation in urban terrain. Such a network could have a life expectancy of up to a week, operating in sleep mode until individuals are detected in which case as much situational information as possible could be obtained (including possibly images or sound on a suite of point-to-point connections).

Note that progresses in the engineering of Bluetooth modules will probably not provide the orders of magniture improvements that would be needed to embed a Bluetooth radio component in a sensor network whose life expectancy is larger than a few months.

5 Conclusion

Our work shows that Bluetooth has some advantages and many limitations in the context of sensor networks. The main advantage of Bluetooth is its support for separated channels, which avoids interferences and enables radio-driven power management strategies. There is a need for efficient topology control algorithms on top of separated channel radios. Another very interesting aspect of Bluetooth is its integration in single-chip systems. Further work is needed to investigate the opportunities of such systems.
The Bluetooth standard is neither flexible (layered approach, no exchange of data during device discovery, no support for multihop networks), nor performant (device discovery is slow, frequency hopping requires frequent synchronizations between master and slaves). In addition, the current generation of Bluetooth radios does nor provide satisfactory performances (high cost for connections, throughput loss on multipoint connections). The 802.15.4 standard promises to address some of these issues. Our work provides a baseline to evaluate future 802.15.4-based sensor networks.

References

[1] Atmel home page. http://www.atmel.com/.

[2] Allan Beaufour, Martin Leopold, and Philippe Bonnet. Smart-tag based data dissemination. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*, pages 68–77, New York, September 28 2002. ACM Press.

[3] International Electrotechnical Commision. Letter symbols to be used in electrical technology - part 2: Telecommunications and electronics. IEC standard 60027-2, 2.nd edition, 2000.

[4] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking August 15 - 19, 1999, Seattle, WA USA*, pages 263–270, 1999.

[5] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS-00)*, pages 93–104, 2000.

[6] Charles F Struman Jennifer Bray. *Bluetooth Connect Without Cables*. Prentice Hall, 2001.

[7] Oliver Kasten and Marc Langheinrich. First experiences with bluetooth in the smart-its distributed sensor network. In *Workshop on Ubiquitous Computing and Communications, PACT*, 2001.

[8] Martin Leopold. Evaluation of bluetooth communication: Simulation and experiments. Technical report, DIKU 02/03, 2002.

[9] Martin Leopold, Mads Dydensborg, and Philippe Bonnet. Bluetooth and sensor networks: A reality check. In *1st ACM Conference on Sensor Networks*, 2003.

[10] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*, 2003.

[11] BlueTags Home Page. http://www.bluetags.com/.

[12] Cambridge Consultants Home Page. http://www.cambridgeconsultants.com/pd_xap_reduced.shtml.

[13] Manatee Project Home Page. http://www.distlab.dk/manatee/.

[14] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[15] Smart-its home page. http://www.smart-its.org.

[16] IEEE 802.15.4 Standard. http://www.ieee802.org/15/pub/tg4.html.

[17] http://www.ee.princeton.edu/ mrm/zebranet.html.

[18] Xilinx's Bluetooth Hardware Tutorial. http://www.xilinx.com/esp/bluetooth/system_design/.

[19] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *1st ACM Conference on Sensor Networks*, 2003.