Synchronization in Wireless Sensor Networks Using Bluetooth

Roberto Casas[1], Héctor J. Gracia[2], Álvaro Marco[2], Jorge L. Falcó[2]

[1] Castelldefels School of Technology,
Technical University of Catalonia, Barcelona, Spain
rcasas@eel.upc.edu

[2]Superior Polytechnic Centre,
University of Zaragoza, Zaragoza, Spain
{hgracia,amarco,jfalco}@unizar.es

Abstract *— Wireless Sensor Networks (WSNs) can take advantage of versatility, completeness, and low prices of standard wireless protocols; Bluetooth as we will show later is a candidate suitable for WSNs. The fusion of data collected over a WSN is just an evident application of time synchronization. Bringing together these two issues, we find that synchronization using standard protocols poses an important drawback. In this paper, we present a simple method that allows clock synchronization in Bluetooth WSNs, down to few microseconds.*

1 Introduction

A Wireless Sensor Network is a particular case of a network composed by a large number of small devices with specific characteristics and requirements. These devices, frequently called nodes, take measurements, process them and communicate with the others coordinating their operations and collaborating to achieve a complex sensing task, named data fusion [1]. It illustrates a common need of synchronization in WSNs.

It is usual in WSNs to use non-deterministic communication channels characterized by variable, and sometimes relatively high, delaying times when transmitting information. This makes synchronization in WSNs is a major task. We can cluster the needs for synchronization in two groups: time scheduling when the nodes coordinate to perform tasks (collect or deliver data), and time stamping when a data processor aggregates information taking into account the collecting instant (use of redundant data to correct errors, reconstruction of system's state for control algorithms, off line analysis, data fusion algorithms and filters).

This paper introduces Broadcast Synchronization over Bluetooth (BSB); an accurate method improving the results of other specific sensornet protocols. First, we show related work in time synchronization in literature, give reasons for suitability of Bluetooth for WSNs and describe how the protocol and former works address synchronization specifically with Bluetooth networks. We also go deeply in the low level timing of the protocol relevant to understand BSB behaviour. Then, we explain how BSB is implemented in a WSN, analyze theoretically its advantages and describe the experimental setup and scenarios for its evaluation. After that, we show and analyze the results, and evaluate BSB performance comparing it with other methods. Finally, conclusions are explained.

2 State of the art

Time synchronization in distributed systems requiring a common temporal reference is easily achievable when there is a physical medium with known delays, both in systems with a dedicated cable for a clock reference and in those using data channel for synchronization with real time protocols as CAN [2]. Non real time data protocols, as Ethernet, using NTP (Network Time Protocol) may be suitable for computer synchronization on the Internet [3] but others show important limitations in sensor networks as energy and computation resources needed [1].

Most common alternative in wireless systems is to use radiofrequency (RF). GPS is a classical example providing high accuracy, better than 200 ns relative to UTC with commercial receivers [4]. This solution has important limitations for WSN: cost of the dedicated hardware, the settling time (up to several minutes) and need for a clear sky view.

Standard wireless communication protocols for WSNs eases many other considerations (channel noise, error management, connection, etc.). Creating a common temporal reference by using the nodes' wireless communication capabilities has been widely studied in bibliography [1]. Synchronization methods are analyzed keeping in mind the energy, cost and size limitations of the devices used in WSNs.

Time adjustment is a major issue. Attending to the strategy for time adjusting, we could group methods in a posteriori and a priori and synchronization [5]. First ones keep devices' clocks running free, gathering information between relative clocks and rearranging timestamps once the measurement process has finished. Second ones allow time stamping or scheduling with a common time reference (network global time) and requires regular clock corrections. Common time reference are best for WSNs, for two-way-message methods can overload the network (one return message for each device is needed to estimate the communication delay) and one-way-message methods are more energy-efficient but usually less accurate. TPSN (Timing-sync Protocol for Sensor Networks) achieves good time accuracy. It avoids the indeterminism typical with the high level protocols, because it works with the MAC layer to precisely timestamp messages at the exact moment they are sent [6].

Using broadcast messages to establish common time reference gets rid of the main error sources. It assumes that all devices listening to the broadcast get the message at the same time, eliminating time uncertainty introduced the sender and setting a temporal reference shared by all the nodes [5, 7]. Later, the entity collecting all data can translate each timestamp to the global network time or each device can compare its clock with the others by means of algorithms such the ones described in [8].

When talking about Bluetooth to get synchronization, the standard defines synchronous connection-oriented (SCO). This is a channel with reserved communication intervals, which can be considered as a circuit-switched connection between the master and the slave. Clock synchronization could be achieved. However, practical aspects impede it in WSNs: it is a point to point connection allowing just three slaves in each piconet, with low accuracy and high power consumption.

According to the standard, physical channels are defined by a pseudo-random RF channel hopping sequence. Each hop occurs every 625 μs and corresponds to a different time slot. In a master-multislave communication, TDD (time division duplex) scheme is used: all the even slots are assigned to the master and the odd ones are apportioned to the slaves. Each slot is numbered from 0 to 2^{27}-1. To achieve that, all the

Bluetooth units have a 28 bits free running clock ticking at 3.2 kHz, and all the members of a piconet know their own offset to the master's clock: the piconet's heart beat. Since clocks are free-running with their own drifts (standard delimits +/-20 ppm), offsets have to be updated regularly. Every successful packet received carries information for it. Moreover, to overcome the misalignments between resynchronizations, an uncertainty window is considered when starting each slot: in active modes it is +/-10 μs, but in low power modes it can be greater due to increased time lapses between offset updates. In practice, the slot start instant determination is implemented in low level layer so applications using Bluetooth modules can not have access to it.

Resuming, Bluetooth has a thick grained ticking (312.5 μs) to keep the slot count accessible by an offset variable, which will not be enough in many cases. An uncertainty window (down to +/-10 μs) allows synchronizing emission and reception slots; lamentably this timing reference is not available for applications.

3 Is Bluetooth a suitable protocol for WSNs?

Some years ago Bluetooth was positioned as short distance cable substitution alternative interfering with 802.11b [9], contrasting in coverage, data rate, power consumption and computation resources. Recently Bluetooth concept has evolved to a protocol suitable to supporting more complex ad hoc networks with specific requirements, especially WSNs. In [10], advantages and drawbacks of the usage of Bluetooth in sensornets are analyzed, concluding that it is a good option for applications with infrequent data transferring, but at high rates. However, in [11], an exhaustive analysis of its applicability for large scale WSNs is carried out. By simulating the lower layers of the protocol stack, from baseband to BNEP (Bluetooth Network Encapsulation Protocol) and analyzing power consumption in interference resilient environments, they conclude that Bluetooth is an efficient protocol suitable for WSNs.

Several key issues (being continuously improved) support the consideration of suitability in sensornets. Data rate goes about 2 Mbps maintaining the radiofrequency modulation and dividing by two the symbol period (version 2.0+EDR). Besides doubling the data rate without big increase in power consumption, this implies a reduction of the energy necessary for transmitting the same amount of data as fewer packets or shorter payloads are necessary [12]. Chip manufacturers also work in proprietary low power modes [13]. Although having good performance for many applications, still it is far from WSN specific devices in terms of energy saving for low data rate applications: class 2 Bluetooth module from CSR has 180 μA average current consumption in a low power mode (able to receive messages every 1.28 s), around 17 mA in active mode transmitting at 115.2 kbps and 30 μA in sleep mode with no RF connection. Berkeley Motes have several times wider coverage, they consume less than 1 μA in sleep mode and 25 mA while sending data at 38.4 kbps. At higher levels in the stack, there is a proposal of a new protocol, based in Bluetooth, specifically designed for energy-efficient data collection in sensor networks [14].

The network structure is also a crucial point. Wide coverage networks with Bluetooth are possible by merging piconets, the so called scatternets [15, 16]. Class 1 devices can also extend the link distance up to 100 m but increasing the transmitting power. In active (communicating, high consumption) mode, each master on a piconet is limited to connecting simultaneously with up to seven slaves, which sets an important limitation for wide coverage. In low power mode (park state) up to 255 can be

connected and all receive broadcast messages from the master and share an adjusting clock that allows coordinated wake up for data collection. As WSNs commonly do not require very frequent data collection a good compromise is switching the nodes to park state while not collecting data, improving consumption and allowing many devices connection [17]. It also allows synchronization through broadcast.

Common applications for WSNs need a big amount of low demanding nodes, which makes the node cost a key issue. Big market penetration of Bluetooth offers chips with excellent price vs. computation resources relationship.

Dedicated protocols (Berkeley Motes, ZigBee) have better overall characteristics for WSNs. Nevertheless, based on the previous arguments, successful implementations of many sensornet applications [17, 18, 19], and synchronization results we have found, we can affirm that Bluetooth is an appropriate protocol to be used at certain WSNs.

4 Proposed synchronization method

Traditional synchronization systems where an entity sends its time reference by messages to remote devices, experience a big problem when dealing with standard communication protocols: delay between the time stamping and the instant when the message is processed by the remote device is not constant. This makes inaccurate the propagation of the time reference. When time stamping can be done directly in the MAC layer [6], high accuracy is achieved with this sender-receiver synchronization scheme as it drastically reduces the sender's uncertainty.

If access to the low layers is not possible, the accuracy is raised when synchronizing using broadcast messages [7, 5]. The main reason is the elimination of the uncertainty time in the sender. Some works defend the fact that the reception moment of a broadcast message is tight [7, 21]. Indeed, in [5] is presented, after an exhaustive characterization analysis, that time difference between reception instants of broadcast messages follows a Gaussian distribution for the devices tested (Berkeley Motes).

Our method takes advantage of the elimination of the delay of the sender and checks that reception instants also follow a Gaussian distribution for Bluetooth.

4.1 Description and implementation

BSB method uses the moment of reception of a Bluetooth broadcast message (no matter the information in it) as synchronization reference for each node. It is based in the small difference found in the delays of the notifying messages that warn the Bluetooth host (node) about the reception of a broadcast. This notification is done via HCI, a part of the standard implemented in modules from different vendors [20]. We found this hardware configuration to be generic enough, allowing full control of the protocol.

We implemented a WSN with a master and 4 nodes to test the behaviour of the method under different conditions. Each node is controlled by a microcontroller with a 200 ns clock resolution to which sensors are connected. It also manages the Bluetooth module using a simplified Host Controller Interface (HCI@115 kbps) and raises a GPIO pin when a broadcast message has arrived; these devices will act as the slaves of a Bluetooth network. The broadcast message is sent by the master, a PC with another BT module controlled via HCI, we use to manage the piconet. For better comprehension, we show in figure 1 the system modules and the processes from the

generation the broadcast message to the evidence of the delivery through the GPIO port.

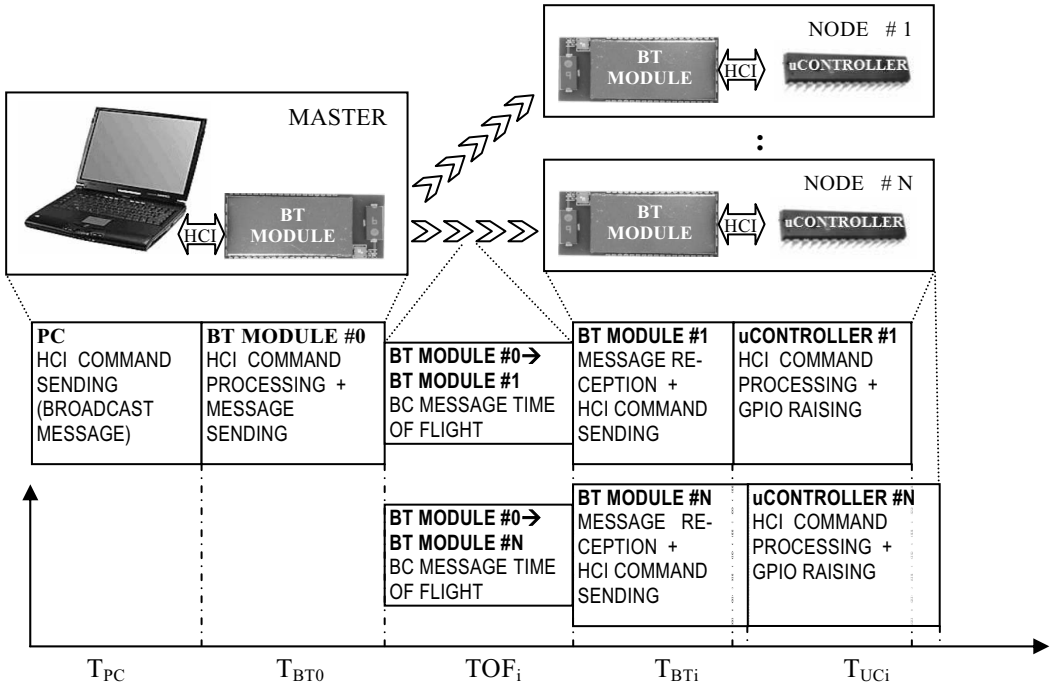

Figure 1: BSB timing in WSN implemented

A brief timing analysis gives us the clues of the advantage of BSB method. In classical methods where the master's time stamp is propagated to the devices to be synchronized, all time intervals represented in the figure 1 have to be taken into account. In this case T_{PC} and T_{BT0} would represent the time in the clock gathering, message creation and protocol processing by the module, access time to the channel and physical transmission. These times are usually the biggest contribution to the total delaying time in most networks. Moreover, they are normally complicated to quantize due to their nondeterministic nature. This is precisely our case; since the PC sends the HCI command to the BT module, until it get access to the medium and really sends the message can pass several milliseconds. When using broadcast synchronization methods, the contribution of those times does not have influence in the technique accuracy [5]. Given the speed of the radiofrequency propagation in the air (the speed of light) we can assume that the time of flight from the master to the different slaves (TOF_i) will be the same: negligible [5].

In BSB time of arrival differences among nodes may only be due to delays in each node: receive time T_{BTi} and T_{UCi} are the only critical ones. We can say differences in instant of notification are just due to the variation between devices in the Bluetooth message processing, HCI warning and later handling by the microcontroller.

4.2 Experimental setup and scenarios

We have registered the four GPIO pins of the nodes with a four channel digital oscilloscope. Taking one node as reference, time differences among these signals are meas-

ured with a 100 ns precision. From the several operating modes in Bluetooth standard we have selected for evaluation of synchronization those most useful in WSNs: active and park states.

Active mode is useful in small WSNs, with large data exchange and without power consumption limitations: higher data rates can be achieved and all master and nodes can emit and receive data at the cost of higher power consumption, with a limitation of seven active slaves in a piconet at any given time.

The standard defines three low power modes: sniff, hold and park. First two work respectively reducing the slave's activity by alternating active mode and temporary disconnection from the piconet. We do not consider them because when they may receive broadcast messages there is no difference with active mode.

Park mode is proposed when a slave does not need to participate emitting on the piconet channel, but still wants to remain connected and keeps receiving capacity. In this state the slave enters in the lowest power consumption mode, leaving it at regular intervals to re-synchronize and to receive broadcast messages. It also admits up to 255 connected devices (or even more depending on the hardware implementation). To transmit data when WSN needs data collection, nodes need to swap to active mode. Depending on the application, cycles of sleep and "broadcast receiving" will vary; thus, we have evaluated several intervals: 20 ms, 650 ms and 2.5 s.

To validate the study in more real/adverse environment, we have also tested the system with two simultaneous interfering communications: another Bluetooth piconet transferring large files over FTP at 50 kbps and an 802.11b network also transferring files at 18 Mbps in the same area.

4.3 Results

a) Gaussian distribution of delays

The first result is the characterization of the reception time difference when four nodes listen to a broadcast message. The resulting histogram of 300 samples, grouping the measurements into 1μs buckets, is showed in figure 2.

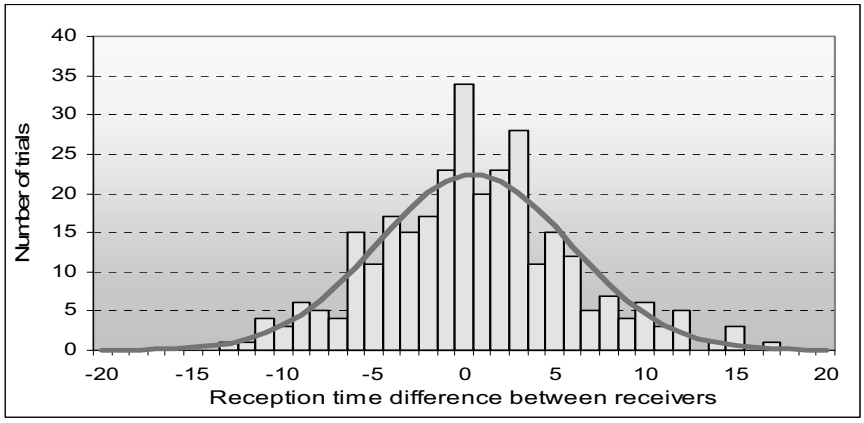

Figure 2: Histogram of reception time difference between receivers

An exploratory data analysis shows a skewness of 0.224 and a kurtosis of 0.244 indicating that the distribution appears Gaussian. It is also confirmed by a Chi squared

test. The difference between GPIO events has a mean of 0.5 μs with a standard deviation of 5.3 μs and a maximum of 17 μs.

b) Time precision in synchronization

Statistical post-facto treatment of the synchronization measurements is a common practice to improve clock offset and skew. This is just possible when dealing with well known error distributions. This normal distribution would enable improvement of the synchronization error statistically but, as seen before, this is not the aim of this paper: several measurements would be necessary, which would need more wireless communication and more energy consumption. In very low power applications and when needing last-minute synchronization, it is interesting to have a reliable method able to give good accuracy in one message. Thus, we evaluate ours without any *post-facto* operation. Statistics shown only characterize one message synchronization precision.

In table 1 we characterize the error in time among the GPIO pulses generated by the microcontrollers, at the arriving of broadcast messages. To calculate the statistics we use only the magnitude of the synchronization error and neglect the sign because it only indicates that the reference clock is ahead among the others. We present, for each scenario described before, the average error, standard deviation and maximum error from 75 measurements. We also indicate the percentage of synchronization events with an error below 10 μs. We choose this limit because it is the uncertainty window considered by the Bluetooth standard.

	Active	**Park (20 ms)**	**Park (620 ms)**	**Park (2.5 s)**	**Noise**
Average error (in μs)	3.70	6.13	5.74	2.67	6.30
Standard deviation (in μs)	2.30	3.63	4.28	1.73	4.23
Worst case error (in μs)	9.60	13.10	17.40	6.90	17.30
Percentage of samples with error below 10 μs	100%	80%	83%	100%	80%

Table 1: BSB statistics over different scenarios

Noise in the channel, as described in the set up, has been tested and does not affect precision in the data above.

It could seem logical that the more time between Bluetooth synchronizations, the more drift between clocks and the larger uncertainty window. On the contrary, as shown in table 1, the accuracy achieved does not appear to be related neither with the radio state (active or park) nor with the noise in the channel. The only opportunities every slave has to resynchronize are the transmissions it gets from the master (strictly, every successfully received packet header), so the reason for that precision could only be the good clock compensation implemented in the low level Bluetooth modules. We can say that this synchronization method is a way to access the accurate timing of the standard.

5 Evaluation

There are many requirements that should be kept in mind when designing a synchronization strategy for a system. [1] presents an exhaustive analysis we particularize when dealing with Bluetooth.

As told before, the **energy consumption** is one of the most important features needed for the WSN devices. We have seen that Bluetooth expenditure is low enough to be used in WSNs, and BSB is quite power efficient because it only needs one message to send a common temporal reference to all the nodes in range. It is important to observe that it is possible to perform synchronization keeping the nodes in low power modes (park).

Cost and **scalability** are also discussed in the "Is Bluetooth a suitable protocol for WSNs?" section. Scatternets are the solution to extend Bluetooth networks range, and its arrangement is similar to multihop WSNs. Thus, propagation of the time reference could be done in the same way as detailed in many works out of the scope of this publication.

Talking about **settling time**, the only time required to start with this synchronization is the one needed to establish the connection between the master and the slave nodes and send a broadcast message. Of course, it will depend on the number of devices, but the connection time of a single unit can be tens of ms.

The experiments showed a good **robustness**; we have a maximum error of 17.4 μs for 300 measurements in several scenarios including one with high noise on the communication channel.

The synchronization **lifetime** will depend on many factors. Firstly the frequency of the broadcast messages; the higher they are, the more references the nodes will have. Other issue will be the characteristics of the device attached to the module, mainly its clock drift and speed. This is important because, between broadcasts, it will be in charge of keeping the clock "in time". The post-facto algorithms used to estimate its drift and offset are also important.

One of the strongest points about using Bluetooth modules is their **ability to be configured**. We can park our devices with the needed synchronization period, optimizing this way the energy consumption without decreasing precision. This feature, added to the fact that synchronization error has no relation with the number of devices connected, (other protocols are heavily dependant [5] on the number of devices compounding the network) results in a network with a high grade of scalability.

The most significant aspect to be observed is the **precision** achievable. BSB can accomplish better results than other protocols specifically designed for its use in WSNs. This is shown in the table 2, where BSB accuracy is compared with two specific synchronization protocols implemented using Berkeley Motes: TPSN [6] and RBS [5].

Parameter	TPSN	RBS	BSB
Average error (us)	16.9	29.13	4.56
Worst case error (us)	44	93	17.4
Best case error (us)	0	0	0
Percentage of time error is less than or equal to average error	64%	53%	60%

Table 2: Comparison of error in synchronization methods

We can observe a considerable improvement in average error, less than 5 μs, which will be good for post-facto statistical treatment. Worst case error enhancement allows increasing the accuracy when using one single broadcast, we can be quite sure that all the nodes in a piconet will process it with a misalignment below 18 μs.

6 Conclusion

WSNs can take many advantages from using Bluetooth protocol. We have introduced and analyzed "Broadcast Synchronization over Bluetooth", resulting in a good WSN synchronization method that gives better precision results than other specific sensor-net protocols.

Due to complexity establishing a low error relationship between clocks in two devices sharing a highly structured protocol such as Bluetooth, conventional methods are not commonly well suited. This uncertainty is reduced by sending a broadcast message and synchronizing the sensing moment of all the nodes when they receive it. This way, every device listening to the broadcast message receives a time reference with very little difference in time.

We have implemented the experimental setup. Then, temporal delays involved have been analyzed and checked their improvement over other synchronization schemes. After that, we have proved that the statistical distribution of delays adjusts to a Gaussian function, which will allow statistical treatment. Finally, we have found the precision of BSB is about 4.5 μs in average and 17.4 μs in the worst case. This precision is maintained in low power modes of Bluetooth modules that allow up to 255 nodes and in noisy channel conditions.

Further analysis of results comparing with standard synchronization protocols for WSN give a superior performance of BSB.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Technology, under the CICYT Project numbers TIC-2001-1868-C03-01, TIC2003-07766 and the PROFIT Project number FIT-150200-2000-215. We would also like to thank Ana Garrido for her support with statistics and Alfredo Mata for his revision in Bluetooth aspects.

References

[1] F. Sivrikaya, B. Yener. Time synchronization in sensor networks: A survey. *IEEE Network, 4(18):* 45–50, July 2004.

[2] M. Gergeleit, and H. Streich. Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus. *1st international CAN-Conference 94*, Mainz, 13-14 Sep., CAN in Automation e.V., Erlangen 1994

[3] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[4] J. Mannermaa, K. Kalliomaki, T. Mansten, and S. Turunen. Timing performance of various GPS receivers. *In Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and the IEEE International Frequency Control Symposium*, pages 287–290, April 1999.

[5] J. Elson, L. Girod, and D. Estrin. Fine-Grained Time Synchronization using Reference Broadcasts. *Proc. 5th Symp. Op. Sys. Design and Implementation*, Boston, MA, Dec. 2002.

[6] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks. *ACM Conference on Embedded Networked Sensor Systems (SENSYS)* 2003.
[7] M. Mock, R. Frings, E. Nett and Spiro Trikaliotis. Continuous Clock Synchronization in Wireless Real-Time Applications. *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, p.125, October 16-18, 2000
[8] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. *Int'l. Parallel and Distrib. Processing Symp.*, *Wksp. Parallel and Distrib. Comp. Issues in Wireless Networks and Mobile Comp.*, San Francisco, CA, Apr. 2001.
[9] C.F. Chiasserini, R.R. Rao. A comparison between collaborative and non-collaborative coexistence mechanisms for interference mitigation in ISM bands. *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd* , *Volume: 3*, pages: 2187 - 2191, 6-9 May 2001.
[10] M. Leopold, M.B. Dydensborg and P. Bonnet. Conference On Embedded Networked Sensor Systems. *Proceedings of the 1st international conference on Embedded networked sensor systems table of contents*. Pages 103-113. Los Angeles, California, USA Sep.2003
[11] X. Zhang**,** G.F. Riley. Bluetooth Simulations for Wireless Sensor Networks using GTNetS. *12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Oct. 2004
[12] Bluetooth SIG. *BLUETOOTH SPECIFICATION Version 2.0 + EDR*. November 2004
[13] Cambridge Silicon Radio. *CSR application notes*.
[14] M. Handy, J. Blumenthal and D. Timmermann. Energy-Efficient Data Collection for Bluetooth-Based Sensor Networks. *IEEE Instrumentation and Measurement Technology Conference*, Vol. 1, pages 76-81, ISBN: 0-7803-8249-8, Como, Italy, 2004
[15] L. Yong, M.J Lee, T.N. Saadawi. A Bluetooth scatternet-route structure for multi-hop ad hoc networks. Selected Areas in Communications. *IEEE Journal,* Vol.21, pages 229-239, Issue: 2, Feb. 2003
[16] K. Persson, D. Manivannan, M. Singhal. Bluetooth scatternet formation: criteria, models and classification. *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, Pages:59 – 64, 5-8 Jan. 2004
[17] J.G. Castaño, J. Lonnblad, M. Svensson, A.G. Castano, M. Ekstrom and Y. Backlund. Steps towards a minimal mobile wireless Bluetooth/sup TM/ sensor. *Sensors for Industry Conference, 2004. Proceedings the ISA/IEEE , 2004,* pages 79-84. IEEE SICON 2004 CONFERENCE (New Orleans, Louissiana) 2004
[18] S.H. Choi, B.K. Kim, J. Park, C.H. Kang and D.S. Eom. An implementation of wireless sensor network for security system using Bluetooth. *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pages 236-244. Feb. 2004
[19] P. Ferrari, A. Flammini, E. Sisinni, D. Marioli and A. Taroni. A Bluetooth-Based Sensor Network with Web Interface. *IMTC03 IEEE Instrumentation and Measurement Technology Conference*, pages 892-896. Veil, Colorado, 21-23 May 2003.
[20] Z. Bradáč, R. Vrba, P. Fiedler and P. Cach. Wireless Communication in Automation. *In Proceedings on 10th IEEE International Conference on Electronics, Circuits and Systems. 10th IEEE International Conference on Electronics, Circuits and Systems (IEEE ICECS 03)*. Pages 659 - 1 320. Sharjah University, U.A.E., 2003.
[21] P. Veríssimo and L. Rodrigues. A posteriori Agreement for Fault-tolerant Clock Synchronization on Broadcast Networks. *22th Int. Symp. on Fault-Tolerant Computing*, Boston, July 1992