New Operating Modes for Bluetooth Networks in Distributed Process Control Systems

Lucia Lo Bello, Nunzio Torrisi, Orazio Mirabella
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania
{lucia.lobello, ntorrisi,omirabel}@diit.unict.it

Abstract - The Master/Slave Time Division Duplex approach implemented in the Bluetooth MAC sublayer simplifies management of the physical communication channel, but suffers from some limitations which prevent efficient bandwidth exploitation and may affect the overall throughput and message delays. These limitation have up to now prevented Bluetooth to be extensively and successfully used in areas which differ from the traditional target applications this protocol has been developed for. In this paper we address these limitations and propose alternative slave/slave operating modes which enable Bluetooth to support new kinds of applications, such as the ones typical of Distributed Process Control Systems.

1. Introduction

Bluetooth technology [1] enables low-cost on-demand connectivity among portable devices, such as notebook computers, mobile phones, PDAs, digital cameras, etc. For short-range wireless ad hoc interconnection, Bluetooth (BT) is also appealing in the industrial environment, where a number of applications, such as autonomous guided vehicles, remote control and diagnostics, process supervision, etc. can benefit from replacing traditional wired connections with wireless ones [2], [3]. An example of this kind of application is machine health monitoring. In some control systems, wireless access to diagnostic information in the encoders of large machinery, for example paper machines, is under investigation [4]. Log files containing information about internal errors which have occurred and the variables relevant to the encoders are usually retrieved by connecting laptops to the encoders by wire. Using wireless technology, this information could be more easily obtained. If the encoder is equipped with a BT interface, instead of climbing onto large machinery to connect a laptop or PDA by wire, it would be possible to stand on the factory floor and retrieve these values by connecting a laptop or a PDA over a BT link.

BT is also gaining ground among wireless systems in the context of Sensor Networks (SN) because of its resilience to interferences (notably in the 2.4 Ghz band), low energy consuption and software support (network self-assembly, multihop routing, in –network processing). Moreover, the mass production of BT modules guarantees robustness and decreasing costs thus pushing the development and usage of BT-based sensor networks. A key point in the way BT works is the scheduling of traffic in the MAC sublayer based on a Master/SlaveTime Division Duplex (TDD) approach. Although this approach simplifies management of the physical communication channel, it has various limits which prevent the bandwidth from being efficiently exploited, with consequences for the overall throughput and the delays affecting messages, thus limiting the areas in which it can be applied. In this paper we will discuss these limits and then propose alternative operating modes to improve the performance offered by the system and extend its use to applications hitherto not considered for BT. We will refer in particular to Distributed Process Control Systems (DPCSs), the particular features of which make them considerably different from applications supplying user services, for which BT was originally developed.

The inefficiency in the MAC layer is essentially due to the rigidity of the scheduling mechanism used, which affects the performance of both the piconets and scatternets. These limits will be discussed, analyzing the reasons which have designers to introduce them, the problems they cause, and possible alternative solutions.

We recall that transmission is based on the use of fixed-length time slots (625μs) which synchronize activities on the various nodes. This basic choice cannot be discussed, even though the constraints it imposes on minimum and maximum frame sizes are worthy of criticism. Use of the Spread Spectrum Frequency Hopping (SSFH) mechanism, in which the frequency of the various transmissions varies continuously, requires the presence of synchronization mechanisms that are certainly simplified in a TDD system with fixed-length time slots (or multiples of a basic slot). Even though the choice of a slot-based approach is not very efficient when the data involved is small in size (e.g. an application in a process control system in which a slot may be transporting a single 16-bit data variable), it is probably a good tradeoff between protocol simplicity and obtainable performance. Slot size will therefore not be discussed in the paper. The points we consider to be worthy of thorough investigation are basically the ways in which the Master manages the physical channel and the use of half-duplex communication in every piconet. The paper will address the limitations of the use of only Master/Slave

Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)
0742-1303/04 $ 20.00 IEEE

communications for DPCSs and propose an extension based on Slave/Slave communications that will enhance system performance. Some recent work recently addressed approaches to enable Slave/Slave communications. As an example, in [5] the idea of dynamic structure/role management was presented. According to this approach, when a slave needs to frequently communicate with another slave, a Master/Slave switching or a piconet partitioning is activated. An improvement of such a technique is presented in [6], where Pseudo Role Switching and Pseudo Partitioning techniques are proposed to reduce the switching bottleneck of the previous approach. In [7], a Time-Slot Leasing approach is presented, where temporary piconets *leasing* slots from the original piconet to support Slave/Slave communications are proposed. This approach permits the existence of multiple temporary piconets, thus enabling multiple communications at the same time. However, all the approaches referred above enable only communication between pairs of slaves, thus restricting the applicability of such techniques to file transfer or similar applications. On the contrary, in this paper we present an approach for Slave/Slave communication where slaves can communicate with any member of a group of slaves which have been configured for Slave/Slave communications, based on a virtual token passing among slaves. This feature makes the proposed approach very suitable for supporting the typical communication exchanges in DPCSs. In the following of the paper, we will first discuss the requirements of data exchange into DPCSs and show how our approach can improve the system performance. Then we will introduce the multi half-duplex mode and discuss the problems related to the presence of multiple parallel communications and the advantages offered by this operating mode.

2. From Master/Slave to Slave/Slave Communications

2.1 Remarks on link management in Bluetooth and on DPCSs.

Communication in a piconet is handled by the Master in a centralized Master/ Slave way. The Master establishes the sequence of transmission/reception frequencies, which will be known to all the piconet Slaves; they will calculate it autonomously, on the basis of the Master address and its clock. As all the nodes in the piconet agree on the next transmission/reception frequency, they can all take part in communications by synchronizing on the appropriate frequency.

Communication occurs on a Master/Slave basis, alternating a Master transmission with one by a designated Slave. As scheduling is handled by the Master, it addresses the Slave that has to reply to the Master. The approach does not allow for Slave/Slave communications. If a Slave wishes to communicate with another Slave it can only do so through the Master. The absence of Slave/Slave communications is a great limit on scheduling, which considerably increases communication delays, thus reducing the overall throughput. However, some features of BT, such as the great immunity to noise provided by frequency hopping and the presence of centralized approaches at the MAC layer, suggest its adoption in DPCSs [2][3][4]. The information flows produced by processes typical of process control environments can be divided into synchronous and asynchronous flows.

- **Synchronous Flow.** This is generated by periodic processes, that perform actions repeated with a constant frequency. An example is a sampling process that receives an analog signal from a sensor and produces a digital signal at the sampling rate. Each periodically produced data item is consumed by one or more consumer processes which are also periodic processes. Data has a lifetime which corresponds to the interval between the generation of two consecutive samples, as shown in Fig.1 and Fig.2.

An example is a consumer process reconstructing a transmitted signal to obtain the original analog signal. The consumer process has to receive each value within a maximum admissible interval which corresponds to the lifetime of the data. As a periodic information flow is linked to processes with known dynamics, the production and consumption periods and the jitter are known a priori.

The transmission of periodic data should in theory occur at fixed time instants; however, the real transmission instant is always located somewhere around the ideal instant, which is known as jitter. For a consumer process reconstructing an original digital signal, the maximum jitter allowed is represented by the sampling interval as shown in Fig.3.

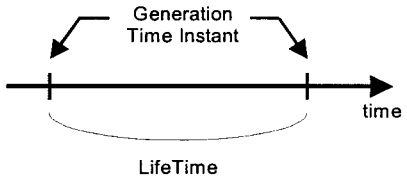

Fig.1: Lifetime of periodic traffic.

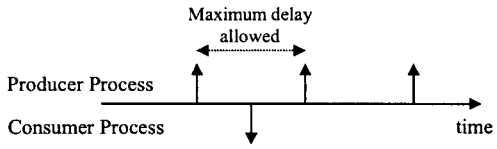

Fig.2: Time constraints of Producer and Consumer processes.

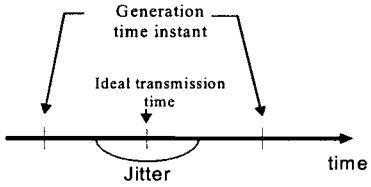

Fig.3: Jitter during the transmission of periodic data.

- **Asynchronous Flow.** This is produced by processes that evolve in time in a way that cannot be foreseen a priori. The term asynchronous is used to highlight the absolute independence between activity on the communication channel (marked by a system clock) and that of the processes producing the traffic. In an industrial process control system there are numerous examples of asynchronous flows, for example alarms generated by a supervision system, operations to *download* intelligent *controller* configurations, or queries to centralized or distributed databases.

The presence of a Master handling access to the physical channel in BT represents an added value for periodic flows with hard real-time constraints as it allows communication delays to be kept under control. Unlike systems where access to the channel is completely distributed and subject to collisions (e.g. 802.11-based networks), the centralized approach in BT allows communications to be scheduled and once the traffic constraints are known, transmissions can be organized in such a way as to meet these constraints.

Centralized link management is not optimal for asynchronous flows because they are not known a priori. The Master can only operate by polling the various Slaves and this approach requires two slots. A more efficient approach for asynchronous traffic is one based on a virtual token passing, in which only one slot is used to address Slave nodes instead of the two slots required by the polling mechanism.

Implementation of a virtual token passing approach in BT implies the potential for Slave/Slave communications, which are currently not provided by the protocol.

The need for Slave/Slave communications is not only linked to the need to support asynchronous information flows, but also to an efficient exploitation of the bandwidth for periodic flows, which are usually the most prevalent type of traffic in DPCSs.

Before presenting an example highlighting the limits of the mechanisms currently present in BT, we will point out three key remarks about the Master/Slave approach on which the following discussion will focus:

Remark 1: -Transmission scheduling is fixed by the Master, which thus uses an algorithm to establish a transmission sequence that satisfies the requirements of the distributed system.

Remark 2:- The Master transmits in odd slots, the Slaves in even slots.

Remark 3:- The Slaves know in which time slot they are to reply to the Master (i.e., after having been addressed by the Master).

In the operating mode illustrated in Remarks 1,2 and 3, Slaves cannot communicate directly without passing through a Master, which in the best hypothesis means at least 4 time slots for information to get from one Slave to another.

In a process control system this makes it necessary, for example, to connect a Programmable Logic Controller (PLC) to the Master node to enable it to acquire information from the

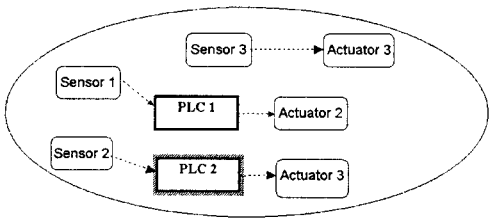

Fig.4: A Process Control Application Scenario.

relative sensors (Slaves) and activate appropriate actuators (Slaves). Direct communication between a sensor and an actuator (both Slaves) is not possible, thus limiting the realization of a control loop (which is common in modern control systems based on intelligent devices) based on a sensor and an actuator only.

A piconet can only have one PLC (which has to be allocated to the Master node) that can only handle one group of variables and execute a single control algorithm. Let us refer to Fig. 4, which represents a portion of a DPCS. If we allocate PLC1 to the piconet Master, the other communication exchanges, that is, the one between Sensor3 and Actuator3 (which realize an autonomous regulation loop) and the one between PLC2, Sensor2 and Actuator2, have necessarily to go through the Master, with a bandwidth waste that impacts on the dynamics of the system being controlled.

It is therefore evident the advantage of enabling Slave/Slave communications. This would obviously require some modification of the communication approach currently used in BT. It should, however, be pointed out that the implementation of Slave/Slave communications still requires Remarks 1, 2 and 3 to be respected, for two reasons:

- To ensure compatibility with devices operating in a traditional mode. The addition of new operating modes must not preclude the presence of traditional devices.
- Ad-hoc hardware must not be required, considering the significant amount of research and design effort devoted in the last few years to developing devices that implement BT . The only variations must be confined to the software.

We will therefore discuss Remarks 1,2 and 3, pointing out the modifications that can be made while maintaining compatibility with the standard.

Remark1: Slave/Slave transmission can take place by means of scheduling handled by the Master, which has to previously configure a group of Slaves as belonging to a certain logical ring (specifying the order of the various Slaves in the Ring: 5, 4, 2, 3, 4 in the example shown in Fig.5.) via a Broadcast message and then transfer to these stations the right to transmit in sequential order, specifying the starting slot and the number of times the sequence is to be repeated. This can all be done in a single Broadcast message. As can be seen in Fig.5A, the Master communicates with some nodes (1, 2, 3 and 4 in the example) in sequence (according to its scheduling) and then authorizes

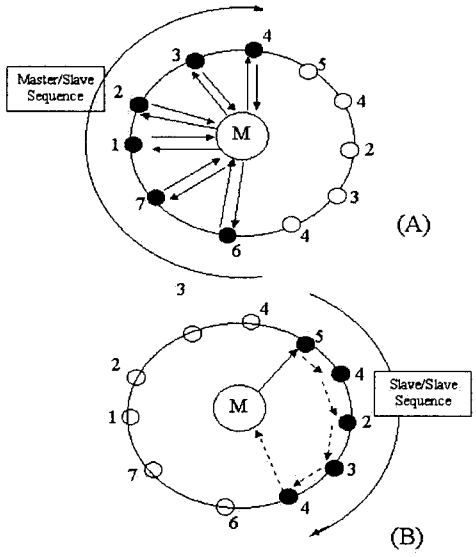

Fig.5 – Master/Slave communication (A), and Slave/Slave Communications (B)

communication between nodes 5, 4, 2, 3 and 4, which have previously been configured as a logical ring (see Fig.5B).

The approach is thus a token passing one, based on a virtual token, i.e. the token does not need to be passed physically because each station knows its position in the logical ring and therefore its turn to transmit. Similar approaches are adopted in some Fieldbuses, such as, for example, the Fieldbus Foundation [8] and P-Net [9]. The use of a fixed-length slot maintains synchronization between the various nodes and prevents overlapping between two consecutive transmissions. Time lags between subsequent transmissions are not needed, because the time sequence of the slots is known to all the nodes (Master and Slaves). Each station can address a message to any (active) Slave according to local scheduling. The presence of a token ring between Slaves enables asynchronous traffic being served more efficiently than by polling (i.e. requiring only half the number of slots), and also accelerates the exchange of information between two Slave nodes (for example Sensor3 and Actuator3 in Fig.4).

Remark 2: When the token is returned to the Master by the last Slave configured in the logical ring, the Master resumes transmissions based on its own scheduling in an odd slot. The Slaves in the ring, on the other hand, transmit in both odd and even slots. If a Slave gives control back to the Master during an odd slot, the Master will have to wait for the next odd slot before resuming transmissions. This means wasting a slot. It is therefore appropriate to organize the logical ring in such a way that the last Slave will complete transmissions in an even slot. Rotation of the token between the various Slaves can be performed once or several times, according to the scheduling established by the Master.

Remark 3: Slaves belonging to the logical ring transmit in the slot in which they possess the virtual token, i.e. when it is their turn according to the scheduling the Master has broadcast. A Slave recognizing itself as being the addressee of a message receives and copies it, but does not reply in the slot immediately following (as would happen in normal Master/Slave communications), unless this is contemplated in the scheduling of the virtual token ring configured by the Master.

In this way it is possible to extend the Master/Slave operating mode to Slave/Slave communications. In any case control of scheduling remains assigned to the Master, which has necessarily to acquire the information needed for correct bandwidth allocation. This information is simple to acquire in a process control system featuring well-defined, repetitive information exchanges (with mostly periodic producer/consumer traffic). The Master could be configured by an operator or acquire a configuration file from a database. As the Master has acquired the time requirements of all traffic, it can perform a scheduling analysis and compute a suitable scheduling table, containing the time slots assigned to each transmission [10]. The transmission sequence is repeated after a time called a *macrocycle*, defined on the basis of the periods of synchronous messages. Finally, it should be pointed out that, due to the need to build a static schedule, the approach described is very suitable for fixed (or rarely mobile) systems in which the piconet nodes do not change and scheduling remains valid for long periods of time. The non-mobility constraint is not, however, a real drawback in DPCSs, where devices are rarely moved once allocated.

An example of the advantages of Slave/Slave scheduling is given by comparing Figs. 6 and 7, which depict the duration of a Scan cycle in a process control system comprising 1 PLC, n −1 sensors (with n<= 7) and 1 actuator. The PLC is allocated on the Master node, while the sensors and actuators are allocated to Slave nodes. As can be seen in Fig.6, with the Master/Slave approach the scan cycle is equal to 2n slots as each data exchange is always preceded by authorization by the Master.

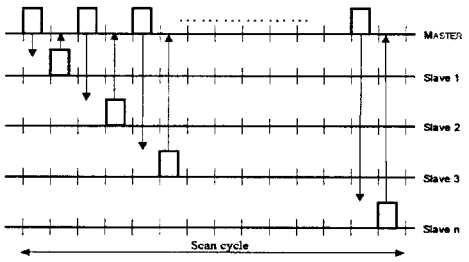

Fig.6: Scan cycle in the Master/Slave approach.

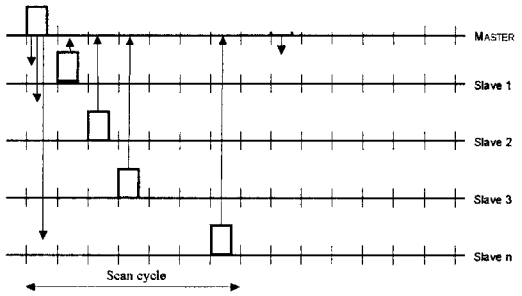

Fig.7: Scan cycle in the Slave/Slave approach.

In the Slave/Slave approach shown in Fig. 7 the scan cycle is n+1, where the first slot is used by the Master to activate the Slave/Slave sequence. If the cycle, once activated, is repeated several times, the scan cycle lasts n slots. The Slave/Slave mode obviously does not prevent data from being sent by Slaves to the Master: on the contrary, sequences of transmissions towards the Master can take place in consecutive slots. This obviously leads to shorter scan cycles and makes it possible to support faster processes.

2.2 Synchronization Problems.

BT requires accurate synchronization between all the elements in a piconet. According to the BT baseband specifications [1] each BT station has to have a clock to mark the system time and calculate the hopping frequencies. This native clock is never reset and is always active until the device is switched off. The clock in each device is a counter and has no relation with absolute time: it can be initialized to any value. The time and hopping frequencies in a piconet are established by the Master clock. After formation of the piconet, the Master clock is transmitted to all the Slaves, thus distributing the reference clock for all transmissions on the piconet. To refer its clock to that of the Master, each Slave adds an offset to its native clock. As the native clock in each device cannot be regulated, the offset is updated periodically. The logic behind the transmission of the Master clock to the Slaves is that if a Slave is connected to the piconet, the offset between its clock and that of the Master must not exceed a certain maximum limit (10 microseconds in the standard).

The native clock, which is also said to be *free-running* as it is never reset, is the reference for all the other clocks. In the Connection State the native clock is marked by a crystal oscillator with a worst-case inaccuracy of ±20ppm.

As each local oscillator has a certain drift d_i which represents the delay or advance on the initial instant of the rising edge of the slot, it is important to keep the max drift of the Slave clocks with respect to the Master under control. Its value never exceeds 0.002% of the nominal length of a slot (625 μs), so after each slot it will be $|d_i| \leq 0.0125\mu s$.

For each packet transmitted by the Master, each active Slave corrects its clock offset, so an increase in the distance between two subsequent Master synchronizations corresponds to an increase in the clock drift between the Master and the Slaves.

Both Masters and Slaves have to provide a tolerance window for the reception of packets such that it is able to intercept early or late packets up to a maximum of 10μs, reaching this early or late threshold represents the worst case. It means, in fact, that each Slave will not only lose the time reference for the transmission and reception of the slots in the piconet, but also be unable to resynchronize via given recovery mechanisms.

When the number of slots in which the Slave clock does not resynchronize grows, the clock drift between the Slave and the Master also increases. Whereas in normal BT operations a Master can remain inactive for a maximum of 5 slots (corresponding to the maximum duration of a transmission by a Slave), it may happen in the Slave/Slave approach introduced here that the virtual token passing mechanism leaves the Master inactive for a considerable length of time, such as to cause a great loss of Slave synchronization.

Let us define t_{outMAX} as the maximum number of consecutive slots in which the Master is absent, after which there is no guarantee that a node will recover temporal alignment in the uncertainty window. The value of t_{outMAX} is equal to the ratio between the maximum length of the uncertainty window and the maximum drift per slot expressed in microseconds.

$$t_{out_{MAX}} = \frac{\text{UncertaintyW indow}}{d_i}$$

Substituting the reference value indicated in the current BT specifications we get:

$$t_{out_{MAX}} = \frac{10\mu s}{0.0125\mu s} = 800 \quad slot$$

This value represents the maximum duration of a Slave/Slave sequence that can be activated before a loss of synchronization with the Master occurs. The worst case occurs when the reception tolerance windows of the Slave and the Master only overlap at the two ends. This happens when the clocks of both devices drift in different directions. As can be seen, it is possible to generate long Slave/Slave transmission sequences without loss of synchronization with the Master.

3. Multi half -duplex mode

The approach described in Section 2 allows for other modifications which will further enhance the performance obtainable. As it is known, transmission in each piconet is half-duplex. As discussed in Section 2.1, for some classes of applications it may be useful to introduce a Slave/Slave transmission periodically to exploit the bandwidth more efficiently. Another step which can be taken is to activate a Slave/Slave communication at the same time as the classical Master/Slave scheduling, so as to have two communications in parallel in a single piconet, thus considerably increasing

bandwidth exploitation. An example will clarify the concept. Let us assume that we have a piconet with one Master and 7 Slaves. At a certain point the Master activates a Slave/Slave sequence, broadcasting a configuration message containing the Slave/Slave sequence 2-4-7-3 and indicating that it is to be repeated 5 times. In the following slot the sequence will start with transmission by station 2 followed by 4, 7 and 3 in the subsequent slots. As the sequence is to be repeated 5 times, it will take a total of 20 slots. During this interval the Slaves do not synchronize with the Master, but maintain their synchronization as described in Section 2.2. In the meanwhile the Master is inactive. The question is whether it is possible for the Master to continue to transmit to Slaves not belonging to the logical ring using the classical Master/Slave approach. In a wired system this would not be possible (with the exception of broadband systems using different channels for data), but in a wireless system it is possible as long as transmission is on a different frequency from the one used by Slaves in the logical ring during the Slave/Slave sequence. As there is only one frequency hopping sequence which all the Slaves (and the Master) know, to prevent Master/Slave and Slave/Slave transmissions from using the same frequency, the simplest solution is to introduce a shift (one position is sufficient for each ring) in the hop sequences used in the two approaches. In this way the two transmissions are guaranteed to take place without interference.

As shown in Fig.8, since the hopping sequences are known to all the nodes (they are computed once and stored into a suitable data structure in the baseband level) it is possible for all stations in the piconet, to agree on a strategy to avoid frequency overlapping (and thus inevitable collisions) between nodes. A shift of one position in the sequence allows the two groups of nodes, Master/Slave and Slave/Slave to operate at the same time without frequency collisions. The Master/Slave group will maintain the standard sequence, while the Slave/Slave group can use the shifted sequence (upon configuration by the Master which enables the Slave/Slave group to operate).

It should be pointed out that in order to avoid collisions between the Master/Slave and Slave/Slave sequences, Slave/Slave transmissions all have to be single-slot. In this way a Slave/Slave transmission will not keep the same frequency for more than one slot and will not collide with a Master/Slave transmission (which, we recall, uses the same frequency sequence shifted back by one position). The same limit does not apply to Master/Slave communications, which can continue in the standard mode, because a multiple-slot transmission will never overlap with a Slave/Slave communication. In

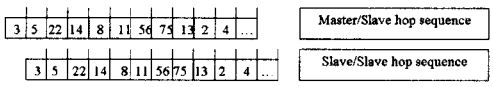

Fig.8: Frequency Hopping shift in two simultaneous trasmission in the same piconet.

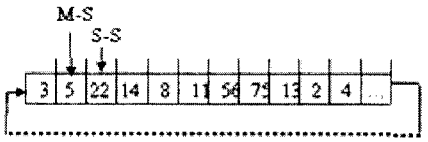

Fig.9: The frequency hopping shift in M-S and S-S communications.

fact, as shown in Fig.9 if a Master/Slave communication mantain the same frequency for 3 or 5 time slots, this does not produce a frequency overlapping between Master/Slave and Slave/Slave communications. If, on the contrary, a Slave/Slave communication use the same frequency for more than one slot, the Master/Slave pointer can reach the Slave/Slave communication frquency. In principle this approach can be taken even further by activating several transmissions at the same time, for example between pairs of nodes, thus notably increasing the overall throughput. In the application scenario shown in Fig.4 it would be possible to configure communication between PLC1, Sensor1 and Actuator2 as a Master/Slave communication and at the same time configure that between Sensor3 and Actuator3, Sensor2, PLC2 and Actuator3 as a Slave/Slave communication. In this way it is possible to manage complex DPCSs requiring several regulation loops in a single piconet.

In any case the Master must perform an accurate analysis of the traffic requirements in order to find a suitable schedule. It starts from a single ring and if the bandwidth is not enough to satisfy the requirements it doubles the rings and moves part of traffic on the second ring as Slave/Slave traffic. If the schedule is not satisfactory, the Master try to accomodate part of asynchronous traffic (if any) on the empty slots of the Master/Slave ring. Otherwise, it splits again the ring into smaller rings. In theory, up to 4 rings can be provided. Fig.10 shows a flow diagram explaining this approach.

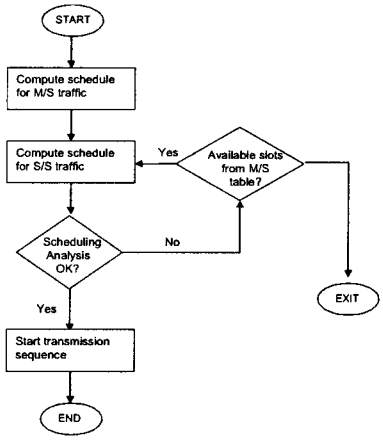

Fig.10: Flow diagram of the scheduling sequence in the multi half duplex approach

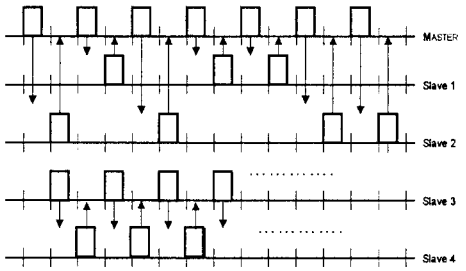

Fig.11: Schedule in a piconet with two logical rings.

One point that must be highlighted is that this maintains compatibility with stations operating in traditional modes: they can operate in the Master/Slave mode without realizing that other stations have activated the Slave/Slave mode. This is extremely important because it does not require stations already present in a network to be replaced: they can be integrated with other stations having a superset of functions that can operate in both the classical Master/Slave and the Slave/Slave mode. Fig.11 shows scheduling in a piconet in which the Master operates in the traditional mode with Slaves 1 and 2, while at the same time Slaves 3 and 4 are operating in the Slave /Slave mode.

3.1 Some implementation remarks.

Although the management of the logical ring we described in the previous sections does not require any modification to the current BT specifications, some additional functionalities are needed, which require a strict integration with processes performing link management in the baseband layer.

Unfortunately, the communication functionalities available between the lower part of BT stack (UART, USB, PCMI) are not fast enough to allow synchonization between scheduling processes located in different parts of the stack, so we are forced to move some functionalities to the baseband layer.

On the basis of the temporal constraints of application processes, the Master node schedules the Slave/Slave communications to be activated. The choices are performed at the Application level, on top of the HCI, where processing

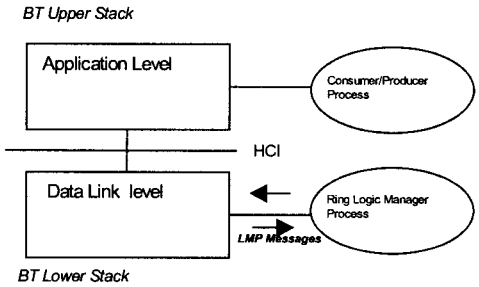

Fig.12: Functional architecture of a Slave/Slave node

resources are available for the computation of scheduling sequences. Then the schedules are transferred to the Data Link level (baseband) where an ad-hoc process will manage the logical ring (or rings). The hardware resources required for the logical ring management reduce to a memory register storing the current Slave/Slave scheduling and one pointer to shift the Master/Slave frequency hopping sequence (as described in Sect. 4). Some memory space is also required for storing the logical ring management process.

Fig. 12 shows the functional organization of the system. All the new functionalities required for Slave/Slave communication can be implemented as firmware in the BT chip, exploiting the unused resources usually available in most BT controllers devices.

4. Conclusions.

The paper has presented a proposal for modification of the BT operating mode to support Slave/Slave communications. After motivating this modification for DPCSs, the paper has shown that it is possible to introduce a multi half-duplex mode to allow for the simultaneous presence of several communications in a single piconet. This is extremely useful in both DPCSs and Sensor Networks, as it enables an increase of the number of devices communicating each other at the same time, thus improving total throughput and reducing the delay. To obtain multiple simultaneous communications in traditional BT systems, the alternative is to use scatternets, but these introduce great complications in handling the exchange of data between elements belonging to different piconets and wastes some bandwidth for the synchronization of the various piconets. In the approch presented in the paper, with careful scheduling, the Master can activate several, to a certain extent independent (but synchronized), communications in a single piconet, this way allowing simple, efficient data exchange between the various components. Each communication channel is based on a virtual ring that allows to define data flows in a flexible way. Through suitable scheduling, a Slave can be allowed to transmitt long sequences of data without any possibility of collisin with other data flows in the same piconet. Scheduling techniques to build transmission sequences able to meet the requirements of synchronous traffic while accommodating asynchronous transmissions are currently under investigation and will be presented in future works.

References

[1] Bluetooth SIG, *Specification of the Bluetooth System - Version 1.1B, Specification Vol. 1 &* 2, February 2001, 2001.

[2] U Bilstrup, P.A. Wiberg, "Bluetooth in Industrial Environment", Proc. IEEE WFCS'2000, Sept.2000, Porto, Portugal.

[3] U. Bilstrup, P-A. Wiberg. "Wireless Technology in Industry- Applications and User Scenarios", *in Proc ETFA2001, IEEE Conf. on Emerging Technologies and Factory Automation*, pp. 123-133, October, 2001.

[4] Leine&Linde, Wireless health monitoring systems for encoders, http://www.leinelinde.se, July 2001.
[5] W. Zhang, H. Zhu, G. Cao, "On Improving the Performance of Bluetooth Networks through Dynamic Role Management", Technical Report CSE-01-018, Pennsylvania State University.
[6] S. Abhyankar, R. Toshiwal, C. Cordeiro, D. Agrawal, "On the Application of Traffic Engineering over Bluetooth Ad Hoc Networks", MSWiM'03, Sept.19, 2003, San Diego (CA), ACM.
[7] W. Zhang, H. Zhu, G. Cao, "Improving Bluetooth Network Performance Through a Time-Slot Leasing Approach", Proc IEEE WCNC'2002.
[8] Foundation Fieldbus, http://www.foundationfieldbus.org
[9] Cenelec (1996). General Purpose Field Communication System EN50170, P-Net.
[10] S. Cavalieri, A. Di Stefano, O. Mirabella, "Pre Run-Time scheduling to reduce Schedule Length in the Fieldbus Environment", IEEE Transactions on Software Engineering, vol.21, 1995.