WBAN implementation on Magnetic Induction Radio IC for medical remote monitoring

Wim Torfs, Peter De Cleyn, Chris Blondia
University of Antwerp - IBBT,
Middelheimlaan 1, B-2020 Antwerp, Belgium
Email: {wim.torfs, peter.decleyn, chris.blondia}@ua.ac.be

Abstract—**The medical service environment is changing. There is a need for automatization of collecting medical data remotely from the patient. Telecare is the continuous remote monitoring of the health of the patient. The IM3 (Interactive Mobile Medical Monitoring) project focuses on developing and implementing such end-to-end service. We implemented the WBAN (Wireless Body Area Network) protocol on the Magnetic Induction Radio IC (Integrated Circuit), which is designed by NXP Semiconductors. We show in this paper that we have a working setup, where nodes form autonomously the network and send ECG data to the sink.**

I. Introduction

The medical service environment is changing. The social security system is being restructured due to changing demographics and the related economic aspects. This should result in rationalizing and cost cutting efforts. Telecare could aid cutting costs to the social security system [2]. It is the continuous remote monitoring of the health of the patient, who is living independent [3]. This means that a patient who in the past would have been admitted in the hospital to keep him under observation, can go home wearing monitoring devices, which send regularly the health status of the patient to the treating physician.

The goal of the IM3 (Interactive Mobile Medical Monitoring) project [1], was to design and implement a telecare and telemedecine service. The new medical service entails wireless monitoring of vital signs at any time and any place. The project is divided in four elements, the wireless body area network (WBAN), the medical hub, the backend server and the biodata viewer. The WBAN collects all the data and sends it to the medical hub. The data are forwarded to the backend server, which stores and processes the data with customized algorithms. The treating physician can use the bioviewer to inspect the patients vitals. This paper covers the implementation of the WBAN element of the project.

The NXP magnetic induction IC, which is based on the NXP Coolflux Digital Signal Processor (DSP) core, forms the core of the sensor nodes. This IC has been chosen because of its very low power consumption (1mW while transmitting) and its magnetic induction radio. The magnetic induction radiation does not harm the patient carrying the radio. The standard development boards, in combination with sensor boards of NXP are used to implement the proposed protocol on. The sensor boards can collect ECG data and accelerometer data. ECG data is used a lot as a data source in WBANs, but once a network has been built, it can be used to carry any sort of data.

The proposed protocol that provides MAC and routing functionality for the WBAN is Cicada [4], a Body Area Network (BAN) tree based scheduling protocol. However, the protocol was designed and validated by means of the ns-2 network simulator [6]. As the authors of [5] mentioned, designers can anticipate some of the challenges of a real deployment. However, some of the problems are only discovered by doing it. We did indeed discover problems when implementing the protocol. To implement Cicada, we had to make some adjustments to the original protocol. Sometimes it made too much assumptions or needed more detailed information in order to implement it.

In the next section, we describe the hardware used to implement the protocol. Section III elaborates on the theoretical protocol which we implemented. The *gap* between the reality and the theoretical model is explained in Section IV. The implementation of the protocol is described in Section V. The proof of concept setup is given in Section VI and finally we provide our conclusions in Section VII.

II. Magnetic Induction Radio IC

The Magnetic Induction Radio IC is a very low power, small footprint IC developed by NXP semiconductors, with the CoolFlux DSP (Digital Signal Processor) as the processor core. It can operate with an external crystal that has a value between 24.734 MHz and 38.144 MHz. However, it can also operate based on the internal RC oscillator, which is of course less precise, but consumes less power and does not require an extra component. The IC contains 33KB SRAM and 6Kwords flash memory. There are a number of peripherals, such as a DMA controller, an I2C controller, I2S, two USART interfaces, 4 timers and a radio component.

The radio provides the physical interface, which is a magnetic induction (MI) radio, which uses Continuous Phase Frequency Shift Keying (CPFSK) modulation. The carrier frequency can range from 7MHz to 15MHz and a maximum raw bandwidth of 300Kbit/s is supported. The distance that can be covered depends on the transmission output power and the coil used as antenna. Ranges in the order of meters can be achieved, hence the need to create a multihop sensor network.

The radio also provides support for the MAC layer. The radio is inherently TDMA based, it creates a TDMA frame of a certain duration in which slots can be assigned. The assignment of slots is actually the scheduling of a *TX* or *RX* command, relative towards the superframe counter. Upon each transmission a header is appended to the data, containing a preamble and a synchronization word. The preamble is used by the receiver to fine tune the bit clock, while the synchronization word is used to detect whether the receiver is intended to receive a message from this transmitter. A checksum or parity is appended to the end of the data transmission, giving the opportunity to the receiver to detect or correct a corrupted message.

III. Cicada Algorithm

The Cicada protocol [4] is a cross-layer algorithm developed for wireless Body Area Networks with a single sink which has been simulated and analyzed in ns-2 [6]. The goal of this protocol is to create a wireless multihop network by means of a spanning tree that is set up autonomously. Routing is thus reduced to merely following the branches of the tree.

The idea of the protocol is to have a single cycle during which all data, sent by leaf nodes, arrives at the sink. In order to do this, Cicada defines two subcycles: the control subcycle and the data subcycle. During the control subcycle a parent sends a message downward to its children, indicating which slots can be used by which child. Each node is assigned one control slot. During the data subcycle, the nodes send data messages upward to their parent in the data slot assigned to them, specified by the message in the control subcycle. In such a way it is possible to traverse the whole tree downward and upward again in a single cycle.

During the data transmission, the node also informs its parent of the bandwidth it needs for the next cycle. This way slots can be assigned dynamically, depending on the need for bandwidth. This dynamic characteristic can also be found in the expansion of the network. New nodes have a contention slot to their disposal to send a registration request. The actual size of a cycle is flexible. Depending on the number of scheduled transmissions, the cycle length is adjusted. This should allow the network to expand dynamically. According to the authors, the network will be constructed in such a way that nodes can only hear their parent, their children and their siblings.

When a node has no data to transmit, it needs to send a hello message to its parent, in order to keep the parent informed about the link quality or mobility of its children. When the parent does not receive a message in five consecutive cycles, the parent assumes that the child has moved away or encountered a hardware failure and it is no longer part of the network.

IV. Implementation conflicts

Since Cicada has been developed independently from a specific hardware platform, the implementation of the protocol poses some issues. Some design changes are required, as is explained in this section.

Cicada defines slot usage within a certain cycle. This cycle does not need to be of static size, but its format stays the same, there is always a control subcycle after which the data subcycle is executed. This can be considered as being a TDMA protocol with a dynamic TDMA frame length. For a TDMA protocol to work, all nodes need to be synchronized in some way. Otherwise it is impossible to specify in which slot a node needs to send.

When the nodes are synchronized, they can send a registration request to their parent. The Cicada protocol provides a contention slot in which new nodes can try to send a registration request. The node, which receives the registration request is considered to be its parent and needs to allocate an extra slot for this node. However, Cicada requires that the tree is built in such way that nodes in a different branch do not hear each other. This requirement can not be fulfilled. What if the registration message arrives at two nodes? The protocol does not describe in detail how this should be handled.

Cicada uses a flexible TDMA frame, allowing dynamically the expansion of the frame when needed. So when a lot of nodes try to send at the same time, the TDMA frame will become larger, allowing all nodes to send their data. This is an interesting approach, however it has some practical issues. When the synchronization is done on the TDMA frames, this requires that all nodes change their TDMA frame size at the same time. This implies that a notification needs to be sent to all nodes during one TDMA frame and all nodes should acknowledge this message and then adjust their TDMA frame length. This results in a quite complicated mechanism which is not error proof.

However, when the synchronization is done on slot level, this dynamic expansion would be easier. Synchronization on slot level can be achieved by listening to the control slot and syncing to that particular slot. This implies however that every node needs to receive this control message in each cycle, otherwise the node is out of sync. Since we are dealing with a wireless medium, messages will get lost. When not implementing this mechanism with care, this could result in an unstable solution.

V. Implementation

Since the Cicada protocol can not be implemented straightforwardly, as argumented in the previous section, we implemented a BAN protocol on the Magnetic Induction Radio IC honoring Cicada principles where possible, but tailored to the specific hardware. First of all, we need to add a synchronization mechanism. The nodes need to know when they should listen or send. Without synchronization, collisions will happen.

As depicted in Fig. 1, we keep the concept of a control stage, a data stage and a contention window in one TDMA frame as in the Cicada protocol. The control stage (called *synchronization stage* (the Sync slots in Fig. 1)) is used to provide this synchronization. Unlike in the Cicada protocol, the *synchronization stage* is bounded in the number of slots (4 slots), where each node sends in a slot which is not used

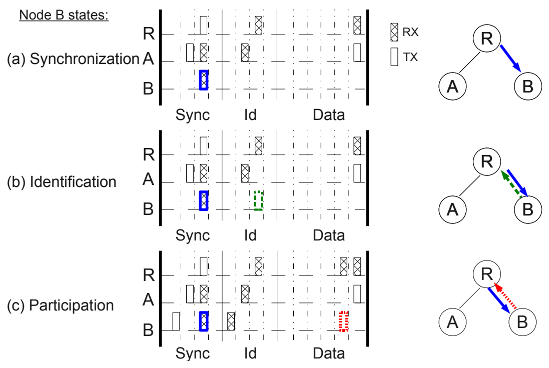


Fig. 1. State visualization of the implemented protocol

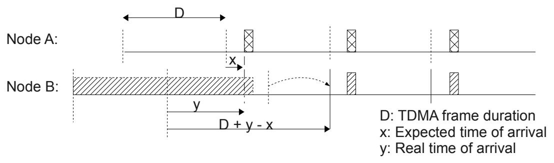


Fig. 2. Synchronization process

by one of its neighbors.
The *identification stage* (the Id slots in Fig. 1), to which Cicada refers to as the contention slot, allows to register new nodes. Requests are forwarded to the root, as in the Cicada protocol. However, the *identification slot* is coupled to the *synchronization slot* and is not a single slot, such as in Cicada. By means of the slots in the data stage (the Data slots in Fig. 1), all data will be forwarded to the root node. We do not support dynamic slot allocation in order to prevent too much delay. We use only four synchronization slots, which implies that the information sent from the root node will not reach all nodes during one frame. We do however support the *hello messages*.
The registration of new nodes happens in three states, the *synchronization*, *identification* and *participation state*. During the *synchronization state* nodes that are not yet part of the network will listen to synchronization messages of their neighbors and synchronize their TDMA frame with their neighbors. Node B in Fig. 1a receives a synchronization message from node R during the synchronization state. In the following state, the *identification state* (Fig. 1b), nodes that are synchronized will send their *id*. The parent will relay this message until it has reached the root node. The root node will assign a data slot to the id of the new node and embed this information in the next synchronization message. When the nodes do not receive this information they send their *id* again. As soon as they receive the data slot information, they enter the *participation state* (Fig. 1c), where they start sending synchronization messages on their own, listen to identification messages and they send data messages.

A. Synchronization

All nodes that are part of the network, send a sync message each TDMA frame in their synchronization slot. Nodes that are not yet part of the network can listen and try to synchronize to these messages. Fig. 2 depicts the process of the synchronization, where node A is sending a synchronization message each TDMA frame in the same slot. Node B is trying to synchronize to node A.
The hardware has a built-in mechanism that allows us to synchronize the TDMA frames. As already mentioned in Section II, a radio transmission consists of a header and the actual data, followed optionally by some bytes for error correction or error detection. This header contains a preamble and a synchronization word. As soon as the receiving radio notices this synchronization word, it takes a snapshot of the current time in the TDMA frame (time y in Fig. 2). If we know when the synchronization word is expected to arrive (time x in the figure), the receiving node can relate the start of the TDMA frame of the sender to its own start of the TDMA frame. The receiver will adjust its TDMA frame length during one TDMA frame, in order to be synchronized to the sender. In this fashion, all nodes can be synchronized with an accuracy of 3,3μs or better (1 bit offset in the worst case).
We provide four synchronization slots for the synchronization messages. Nodes need to listen to their neighbors during the synchronization state and pick a free synchronization slot in which they can send their own synchronization messages without interfering the other nodes. If a node hears a message in all synchronization slots, it will not send synchronization messages to prevent collisions. Nodes in its neighborhood need to synchronize to one of its neighbors. This is a flexible solution which does not limit the number of nodes in the network and has a relative small overhead compared to assigning a synchronization slot to each node.
Nodes that are not anymore in the *synchronization state*, need to keep listening to the synchronization messages they receive from their parents. In case of some kind of network failure, which results in the missing of synchronization messages during five consecutive TDMA frames, the nodes are brought back to the *synchronization state* and start scanning the network again.

B. Identification

During the *identification state*, a node sends a *join-request* message (identification message). This message is sent to the node with which the requesting node is already synchronized. We assume that each node can be identified by some unique hardware dependent number which is contained in the identification message.
The root has the role of a centralized control organism. All identification messages arrive finally at the root, which assigns an available slot to the node. The notification about which slot to use is sent in the synchronization message. There is only room for one notification message. This is flexible since the message format, and thus the source code, do not need to be adjusted depending on the number of nodes in the network. The number of nodes is not limited by the number of bits in the synchronization message, but by the number of available

data slots. If there are multiple notifications to be sent, they need to be sent sequentially in subsequent TDMA frames.
In order to prevent collisions in identification slots, each identification slot is linked towards the synchronization slot that the sensor is synchronized to. This ensures that the identification slot will not be corrupted, as long as the synchronization slot is picked carefully. However, it could be that multiple children of the same parent try to enter the network at the same time, thereby causing collisions of the identification message at their common parent. One could resolve this issue by using a backoff algorithm for sending an identification request. As such, a node will need to wait a random number of frames before sending an identification message, when no notification has arrived.

C. *Participation*

In our solution, dynamic slot assignment is not supported, since slot requests need to be propagated to the root and back to the requester as soon as possible. If we want to perform dynamic requesting of data slots, we require as many synchronization slots and identification slots as the number of nodes that are in the network. Instead, we assign a fixed slot to the nodes upon their registration request (the identification message). The flow on the left of the top of the Fig. 3 depicts

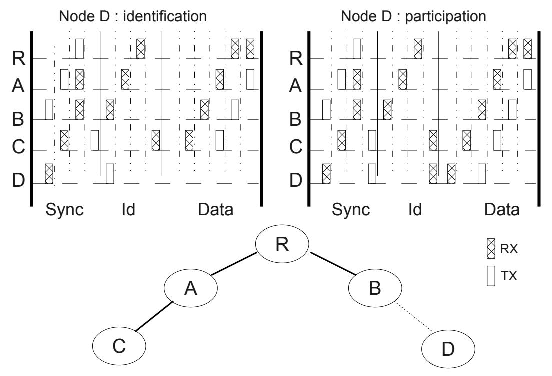
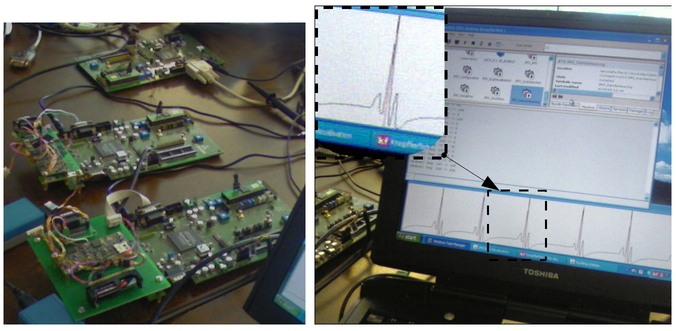


Fig. 3. Packet flow of the implemented protocol

the situation when node D is in the *identification state*. Node D is sending its *id* to its parent. The flow of the *participation state* is shown on the right of the top of the figure. It starts transmitting a sync message in a free synchronization slot. It also tries to receive incoming identification messages from possible future children in the identification slot.
If a node has children, it listens during the slots that have been assigned to its children. Either their data, or the gathered data from its own measurements will be forwarded in the following data transmission slot. If none of them is available, a 'hello' message will be transmitted that is discarded at the receiver side. Since every node has a transmission slot assigned to them, it is possible to send 'hello' messages if nothing else needs to be sent. If a node does not receive any message during five subsequent TDMA frames, it will consider the child to be lost. This applies too for the children, if they do not receive a synchronization message during five subsequent TDMA frames, they consider the link to be broken and start looking for a new parent by entering the *synchronization state* again.



(a) hardware setup (b) ECG output visualization

Fig. 4. PoC setup

VI. Proof of Concept Setup

We created a PoC (Proof of Concept) setup (Fig. 4) that allowed us to evaluate this implementation. The protocol is implemented on the Magnetic Induction Radio IC. The sensor board is connected to the Magnetic Induction Radio IC through the uart. The data, which the sensor board sends, consists of ECG data at a rate of 500 bytes per second (250 samples/s). In this setup, we used a low output power and a small coil as an antenna for the magnetic radio, and thus we achieved a range of about 30cm.
When powering up the nodes, they autonomously form a multihop network and start transmitting the data they receive from their sensor. By moving a node farther away, the link deteriorates and the node tries to resynchronize to the network by listening to synchronization messages. As soon as it hears a synchronization message, it registers again to the network. At that moment, it is again able to transmit its ECG data. There is a local buffer in each of the nodes to store one second of ECG data. So, if the node manages to reconnect within one second, there is no interruption of the data.
In order to test whether the implementation can deliver the results we expect, we use the following scenario. The initial positioning of the nodes is depicted in top of Fig. 5, where the root node is connected directly to the ECG (electrocardiogram) sensor node. When the ECG sensor node moves away from the root node, the connection is broken (middle of Fig. 5). A third node is inserted between both of the nodes (bottom of Fig. 5), which connects to the root node and relays the data coming from the ECG node.
Testing with this scenario led to the observation that when a node enters the network, the connection seems to be made almost instantaneously. However, from measurements, we discovered that making a connection easily takes more than 10 TDMA frames. This delay is caused by the synchronization state in the first place. The new node needs to listen to synchronization messages until it has heard the same message in two consecutive frames. This takes a couple of TDMA

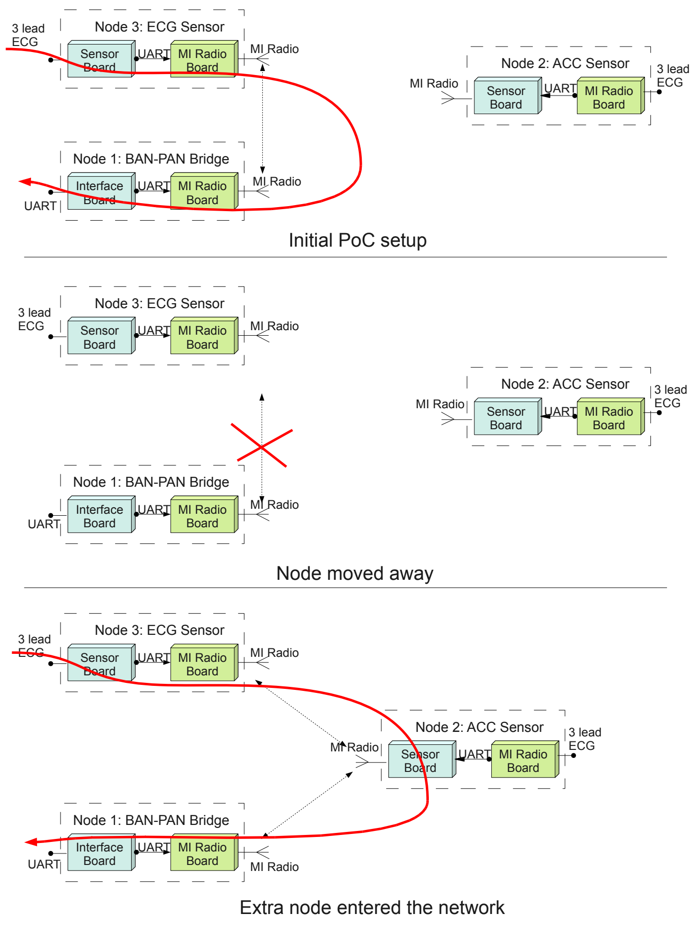


Fig. 5. PoC setup: extra node entered

frames. After selecting a synchronization message, it needs to synchronize to this message. This synchronization takes also several frames. This results in the synchronization phase which takes about 5 to 6 frames. Only then, the identification state can start. In theory, this state should only last one or two TDMA frames. However, in practice, it takes 4 to 5 frames. Sometimes the identification messages get lost or corrupted (when moving two nodes closer to each other), and as a result, the receiver does not hear the identification requests. Therefore, making a connection, which seems to occur almost instantaneous, takes in reality about 10 frames. In the PoC setup, we have only a limited number of nodes. This is why the frame duration is rather small (13ms), which explains why we could have the impression that making a connection is very fast. However, if you would like to have a network of 33 nodes with the same size of data slots, the TDMA frame would take about 100ms. This means that making a connection could take one second.

We also noticed that when moving a node away from its parent, the connection deteriorates first, causing now and then erroneous messages to arrive. The CRC check of these messages will fail. When the node is farther away, some of the messages do not reach their destination, which is detected as a lost message. As well the parent as the child assume that the link is broken when they do not receive a message during five consecutive frames.

The implementation is designed in such a way that the synchronization and identification overhead is a constant value. The more nodes the network can handle, the more data slots need to be available. This implies that there is more time available in the TDMA frame to send data, and thus the aggregate data throughput increases, resulting in a lower overhead. But this implies also that the throughput and delay of a single node is degraded, since it needs to wait until all the other nodes have sent their data.

VII. Conclusion

Thanks to the PoC we can conclude that we have a working solution. We can stream real-time ECG data over the network towards the medical hub. The setup proves that the network can recover autonomously when a node has moved away while an intermediate node enters the network.

The synchronization and identification technique gives the nodes the possibility to synchronize and register to the network. This part performs great and is quite robust. Even when the link is deteriorating, the connection will not be broken immediately. However, the fixed assignment of the data slots has its drawbacks. It implies that the data throughput is limited by the number of nodes that are directly connected to the root node. Another drawback is that the number of nodes in the network is limited by the number of available data slots. A network with a lot of nodes but a low data throughput needs a different configuration than a network with few nodes that send a lot of data. In our future work, we will address these matters by adding QoS to the implemented protocol. This way we can make sure that nodes that need more bandwidth can send their data, while not disturbing the nodes that have only a few bytes to send.

Acknowledgment

This work is funded by the IBBT and has been performed within the Interactive Mobile Medical Monitoring (IM3) project in cooperation with NXP Semiconductors, Onafhankelijke ziekenfondsen, UZ-Gent, Vitalsys and Ziekenhuisnetwerk Antwerpen, Universiteit Gent - IBCN, Universiteit Gent - WICA, Katholieke Universiteit Leuven - COSIC, Katholieke Universiteit Leuven - ICRI, Vrije Universiteit Brussel - SMIT, Universiteit Gent - MEDISIP-IPI.

References

[1] http://www.ibbt.be/project/im3
[2] http://www.bjhc.co.uk/telecare/2007/Presentation slides/Bill Behnke.pdf
[3] http://www.telecareaware.com/what-is-telecare/
[4] B. Latré, B. Braem, I. Moerman, C. Blondia, E. Reusens, W. Joseph, P.Demeester, "A low–delay protcol for Multihop Wireless Body Area Networks", *Mobile and Ubiquitous Systems: Networking & Services, 2007*. Philadelphia, Pennsylvania, USA.
[5] R. Szewczyk, J. Polastre, A. Mainwaring, D. Culler, "Lessons From a Sensor Network Expedition", *Proceedings of the First European Workshop on Sensor Networks (EWSN),2004*. Berlin, Germany.
[6] The Network Simulator NS: http://www.isi.edu/nsnam/ns