QoS support for a MAC with a TDMA tree topology on the Magnetic Induction Radio IC

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

***Abstract*—In our previous work, we implemented a TDMA tree based protocol on the Magnetic Induction Radio IC. The protocol was designed for Body Area Networks (BANs). The result was a protocol that worked for that specific situation, but could not be extended to other applications. The protocol needed some mechanism that would provide extra flexibility to the network. In this work, we add QoS to the protocol, which results in the protocol that can be used in several kinds of applications. The resulting protocol has the capability to be configured at runtime for the kind of sensors that are being used, in order to optimize the throughput and efficiency.**

I. Introduction

Wireless Sensor Networks (WSNs) have become a popular area in many fields. For instance, we can monitor the heart remotely, the body temperature or movement. We can detect an intruder who has entered our premises or get information about hostile movements. WSNs can be used in a wide application area. If we classify these applications, we can say that we have data gathering applications and monitoring applications. The data gathering applications do nothing specific with the data, they simply collect and tranfer the data, e.g. the gathering of wheather information. On the other hand, the monitoring applications react when a certain threshold has been crossed, e.g. a heart monitoring application needs to inform the medicians when a heart failure has occured.

In the case of monitoring applications, it is important that the data arrives in real-time. If an intruder enters your house, you want to know it from the moment he puts his foot in your home and not when he already loaded every posession you have in his car. We want to have the most recent information that is available. This leads to the minimization of buffering. Buffers are often used to prevent data losses when a failure occurred in the network or when a node receives more data than it can relay. Network failures will always occur, since it is a wireless network. But a network can be configured in such a way that all nodes have enough bandwidth to relay all data they receive. When designing a protocol for a certain type of network, we take into account how much data is expected to be sent from each of the nodes and we make the design as optimal as possible for this network. For each kind of network, we need to reconsider the expected bandwidth requirements and redesign the network. By using QoS, in the sense that we allocate more bandwidth to the nodes that have a higher data throughput, we would not need to redesign and reconfigure the protocol according to the new parameters. The network will auto configure itself in the most optimal configuration. This is why QoS is important in such networks.

Our current work is based on previous experiences, where we have noticed that the protocol could perform much better if it would incorporate QoS. The network was designed to cope with the highest possible data throughput. Sensors with a lower throughput used more bandwidth than they needed and sensors with a higher bandwidth were not capable of sending all their data. Numerous papers describe the problem of QoS in TDMA based MACs, which are not referred to because of the lack of space. Almost all of them handle this problem by allocating extra slots to the nodes that need it the most. However, none of them considered that in certain environments, such as the human body, sensors have a constant bitrate. They are sampled at a certain rate and either accumulate their data, or send their data immediately to their neighbors. This implies that sensors send data with a constant bitrate, which is not going to change during the operation, unless they break down. Taking into consideration that we would like to have real time data, it would be more beneficial to configure the network first in such way that there is enough bandwidth available in the complete path from the node towards the sink. When a nodes moves to another location, this path could change, implying that the node would need to renogotiate its bandwidth requirements.

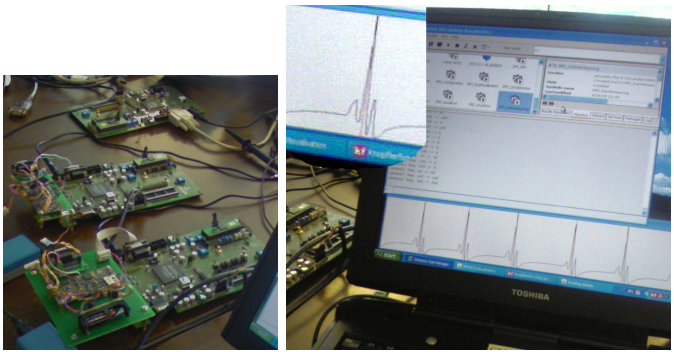
In the next section, we elaborate on our previous work [1]. Section III explains the changes we introduce. The last section gives a conclusion.

II. Previous work

In our previous work [1], we have implemented a tree based TDMA protocol, Cicada [2], on the Magnetic Induction Radio IC (developed by NXP semiconductors) with the CoolFlux DSP (Digital Signal Processor) as the processor core. The protocol was designed specifically for Body Area Networks (BANs), where we wanted to relay the ECG data and accelerometer data through the sensor network towards a hub.

The algorithm is a cross-layer solution, which includes both a TDMA MAC part and a routing part. Since the network topology is a tree, the routing consists only of following the branches to the root node. The protocol was not designed for any specific hardware, which resulted that some of the design choices could not be implemented on our hardware. Therefore we redesigned the original protocol, taking the principles of the original protocol in mind where possible. The PoC (Proof

of Concept) setup proves that nodes are capable to discover neighbors, autoconfigure the network topology, handle link failures and send ECG data in real time (see Fig. 1).



(a) hardware setup (b) ECG output visualization

Fig. 1. PoC setup

A. operational description

As depicted in Fig. 2, we used three types of slots during one TDMA frame, *synchronization slots*, *identification slots* and *data slots*. The synchronization slots are used to send synchronization messages to synchronize the nodes with each other. These messages always flow downwards, from the root of the tree to the leafs. The identification slots are used to send identification messages. Nodes that would like to enter the network use these messages to identify themselves and request a data slot. These messages always flow towards the root node. The last type of slots, the data slots, are used to send the sensor data upwards to the root node. Each node gets a data slot assigned upon registration. Through this slot, nodes can send their data to their parent, which relays the data to its parent untill it reaches the root node.

The registration of nodes happens in three different states. The first state is the synchronization state, where the new node needs to listen to synchronization messages in its neighborhood. During this state, the node gathers information about the occupation of the synchronization slots. As soon as it has heard a synchronization message from the same node during two consecutive TDMA frames, it selects this node as its parent. Then, the node must synchronize its TDMA frame to that of its parent. It uses the arrival time of the synchronization message in combination with the expected arrival time of the synchronization message, which is embedded in the message itself. As soon as the node has confirmed that it is in *sync* with its parent, it passes on to the next state, the identification state.

The identification state is the second state in the registration process. The new node sends its unique identification (id) to its parent by making use of the identification slots. The parent relays this message to its parent until the message reaches the root node. The root node assigns a free data slot, of fixed size, to this id and embeds this information in the next synchronization message it will send. This way, the new node receives the information about which data slot it can use. At this point, the node enters the last state.

The last state, the participation state, is where nodes send synchronization messages on their own. They listen to new children by listening to identification messages and they send sensor data to their parent when appropriate.

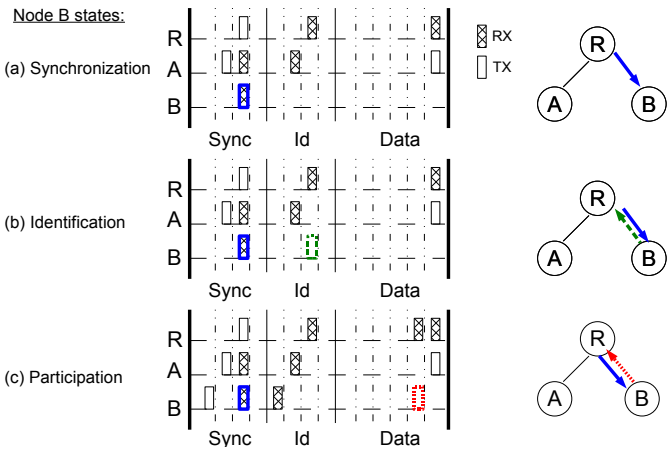


Fig. 2. State visualization of the implemented protocol

The synchronization and data messages form an important part of the protocol. The children can determine whether their parent has moved by listening to the synchronization message its parent should send each TDMA frame. If a child does not hear a synchronization message from its parent during five consecutive frames, it assumes that the link with its parent has broken and searches for a different parent. The parent requires that its children send a message in the data slots allocated to them, even if they have no data to be sent. If it does not hear a message during five consecutive frames, it assumes that the link with its child is broken and removes the child from its child list.

The resulting protocol results in a huge waste of bandwidth. Every node receives only one data slot of fixed size. If a node has n children, then either only one child can send continuously, or all children can send $1/n$ times their data. Not all nodes can use the full bandwidth that is assigned to them. They can send it to their parent, but their parent can not relay all the data it has received. The data slot it can use to send data to its parent is as big as the slot that its children use. At this stage we noticed that we needed some kind of QoS, which would ensure that sensors with a higher data throughput get more bandwidth.

III. QoS Algorithm

A. Introduction

The protocol that we are working on, still uses data slots of fixed size, since this is easier to use with the hardware we are working on. By means of Pulse Width Modulation (PWM), we can distribute the available bandwidth over the requesting nodes.

Fig. 3 depicts the principle of PWM. If we send half the time, twice as much data, it is the same as sending one unit of data

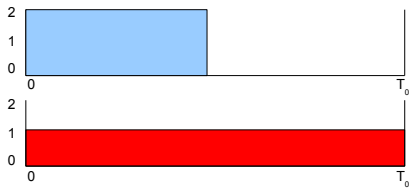


Fig. 3. Pulse width Modulation principle

the whole time. Instead of giving a node the exact amount of bandwidth it needs, we let the nodes use the full bandwidth, but only for a fraction of the time. The result is that the average bandwidth is the full bandwidth multiplied by the fraction of time that the bandwidth was used. A parent node distributes its available bandwidth between its children by letting them to send in the same data slot, but in different TDMA frames. This behaviour is repetitive, thus we have a cycle of *m* TDMA frames, where *n* frames of the cycle are used by a node.

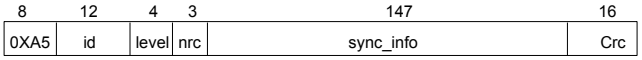
B. detailed description

Like in our previous work, the nodes need to register in order to be able to participate in the network. The same registration technique (3-state registration method) is used to achieve this goal. Once the nodes are synchronized with the other nodes, they send an identification message to their parent. This identification message does not only contain the *id* of the new node, such as in our previous work, but also the bandwidth it requires.

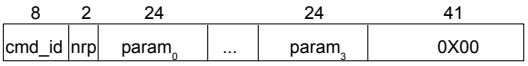
The parent which receives this identification request, checks whether it has enough bandwidth to support the bandwith requests of all its children. The parents needs to keep a list of its children and the amount of bandwidth they need. The fraction of the requested bandwidth, divided by the bandwidth that a single slot can provide will be reduced to its lowest terms by means of the Greatest Common Divisor (gcd). This fraction indicates how many slots per frame the child needs for sending its data. By combining the fractions of all children, a parent can determine the size of a cycle. A cycle is the number of frames that are needed to allocate sufficient data slots to the children. Each child sends during a number of frames of that cycle, which is proportional to the fraction of bandwidth it needs.

In our previous work, the number of nodes was limited by the number of data slots. The protocol needed precise information concerning the expected number of nodes in the application. The protocol we are working on, does not have this restriction. A node can use multiple data slots when needed. Since the synchronization slots and the identification slots are fixed in size, it is more efficient to make the frames as large as possible. This way, we have more space to send actual data. Considering that the hardware clock has a drift of 1 bit per 400 ms (by using a crystal), we use frames of 400 ms. This way, nodes resynchronize each 400 ms.

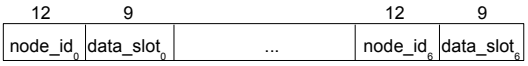
As we already mentioned, the synchronization technique stays the same. We keep 4 synchronization slots in which nodes can send their synchronization messages. However, the content of these messages differs from the synchronization messages of our previous work. Fig.4a depicts the structure of the synchronization message. The *id* field and *level* field stay the same. The *id* field contains the unique identification of the sending node and the *level* field, the level in the tree where the sending node is situated. The *nrc* field and the *sync_info* field have been added. The *nrc* field contains the number of children the sending node has. The *sync_info* field is multiplexed in its use. If the *nrc* field contains 0, the *sync_info* field is used as a command field, otherwise a schedule is given which indicates which child should use which slot.



(a) Sync message format (field size is expressed in bits)



(b) Sync_info format, nrc = 0 (field size is expressed in bits)



(c) Sync_info format, nrc = 1 (field size is expressed in bits)

Fig. 4. Format of the synchronization message

The format of the command structure is given in Fig.4b. The *cmd_id* field indicates which command is given. The *nrp* field gives the number of extra parameters and the *params* field holds the extra parameters.

The schedule structure is given in Fig.4c. Each childs id is followed by the slot number in which it can send its data during this frame. If no schedule is sent, the child assumes that the previous schedule can be used.

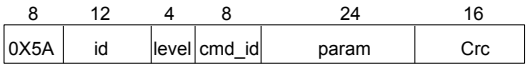


Fig. 5. Format of the identification message

Also the identification message is adjusted, which is depicted in Fig. 5. The *id* field holds the unique identification of the sender and the *level* field contains the level at which the sender is located. These fields are the same as in our previous work. We added the *cmd_id* field and the *param* field. The *cmd_id* field indicates which command is given, the *param* field provides space to pass a parameter with the command.

The changes we made provides a message structure which is flexible enough to allow runtime configuration of the network. The disadvantage is that the messages have become larger, which results in a larger absolute overhead. However, the relative overhead has decreased due to the changes that have been made to the data transmission part, which allows for a frame that has an interval of 400ms. Hence, the synchronization and identification messages are only a fraction of the frame time (the synchronization and identification slots use only 1.49% of the total number of bits that are available in a complete TDMA frame).

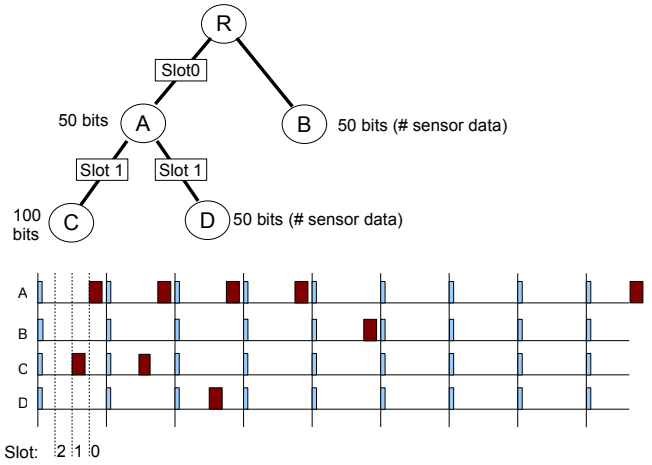


Fig. 6. Slot usage, with a slot capacity of 400 bits

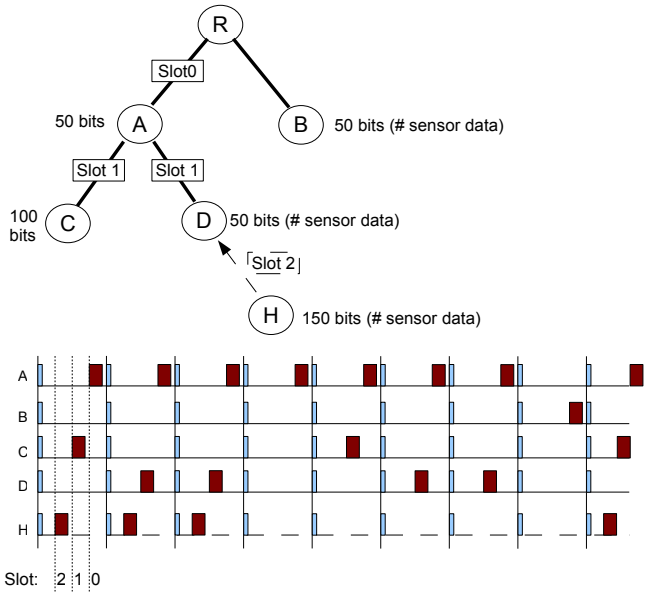


Fig. 7. Slot usage, extra node, with a slot capacity of 400 bits

C. Slot allocation example

Fig. 6 depicts the slot utilization for a network of only five nodes. For simplicity, the frames have only three data slots and each frame lasts one second. Each data slot has a size of 400 bits. The figure shows that node A has a sensor attached which samples at 50 bits per second. The children of node A, node C and D have a sensor attached that sample at 100 bits and 50 bits per second respectively. This implies that node A needs 200 bits per second in total. Node B requires only 50 bits per second, hence node A and B can send in the same data slot, since their total bandwidth is 250 bits per second, while the available bandwidth of one data slot is 400 bits per second. Node R provides only one data slot to both node A and B. The bandwidth of node B is 1/8th of the total bandwidth, the bandwidth of node A is 1/2th of the total bandwidth. Combining these two fractions, leads to the cycle should last 8 frames and node A occupies 4 of these frames, while node B occupies only 1 frame.

Node A also provides only one slot, which is shared between its children. Node C requires 1/4th of this bandwidth, while node D requires 1/8th of the bandwidth. This also results in a cycle of 8 frames, of which node C occupies 2 frames, and node D occupies only 1 frame.

If we add an extra node to the network, node H (Fig. 7), the bandwidth required for node D increases to 200 bits per second, since it needs to send its own data and relay the data of node H. Node A, which relays the data coming from node D, also needs more bandwidth. The cycle of node C and D lasts only 4 frames, since node C requires 1/4th of the bandwith and node D requires 1/2th. Node C occupies only 1 frame, node D occupies 2 frames.

IV. Conclusion

We noticed that our previous work was not efficient and flexible enough. Therefore we are currently working on a solution where QoS solves the unnecessary waste of bandwidth, while providing the flexibility to handle different bandwidth requirements of sensors. This new approach allows that the TDMA frame structure can be enlarged and thus reducing the overhead of the synchronization and identification slots. Moreover, the data slots can be enlarged, thereby reducing the overhead of the physical layer, which appends a header to each message it sends.

This protocol also provides the possibility to configure the nodes during runtime operation. The network can be optimized, depending on the sensors in the network. When the network has a lot of sensors which send only a small amount of data, the data slot size can be reduced, which implies that the nodes do not need to wait a lot of time to gather enough data to send. This would reduce the delay of the sensors that do not have high bandwidth requirements. In such network, we can still have sensors that send a lot of data at once. They use multiple small data slots at the same time. On the other hand, when using larger data slots, the overhead of the physical layer is reduced. So in a network where most nodes have high bandwidth requirements, all nodes can be configured to use the maximum data slot size. By means of the flexible command structure, all kinds of configurations are possible.

In our future work, we will further implement this protocol and gather statistics about its efficiency, stability, flexibility and throughput.

References

[1] W. Torfs, P. De Cleyn, C. Blondia, “Implementation of TDMA communication on Magnetic Induction Radio IC for multihop WSNs”, *Submitted to 6th European Conference on Wireless Sensor Networks (EWSN), 2009*. Cork, Ireland.

[2] 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.