DynMAC: a resistant MAC protocol to coexistence in wireless sensor networks

Luiz H. A. Correia[a,b,1,2,*], Thanh-Dien Tran[b,2], Vasco N. S. S. Pereira[b,2], João C. Giacomin[a,1], Jorge M. Sá Silva[b,2]

[a] *Computer Science Department – Lavras - MG, Brasil*
[b] *Department of Informatics Engineering – Coimbra, Portugal*

Abstract

The growth of mobile and ubiquitous computing has increased the demand for wireless communications, which in turn raises interference levels and spectrum pollution, causing problems of network coexistence. The coexistence assurance between these devices and wireless sensor networks is a big challenge. This paper proposes a new medium access protocol, DynMAC (Dynamic MAC), which uses mechanisms of dynamic channel reconfiguration, recovery from lost links and reconfiguration of transmission parameters based on the properties of the cognitive radios, to deal with this problem. Simulations and experiments using a real WSN testbed, were performed to validate our protocol. Results show that the proposed mechanisms solve the WSN configuration problems, in noisy and interference environments, and enable the coexistence with different networks and devices operating in the same frequency spectrum, while maintaining application requirements in critical deployment scenarios.

Keywords: Wireless Sensor Network, Dynamic Channel Reconfiguration, Coexistence, Cognitive Radio.

*Corresponding author

Email addresses: lcorreia@dcc.ufla.br (Luiz H. A. Correia), than@dei.uc.pt (Thanh-Dien Tran), vasco@dei.uc.pt (Vasco N. S. S. Pereira), giacomin@dcc.ufla.br (João C. Giacomin), sasilva@dei.uc.pt (Jorge M. Sá Silva)

[1] Federal University of Lavras.

[2] University of Coimbra.

1. Introduction

The demand for ubiquitous and wireless devices has grown exponentially in recent years as more and more applications were created. However, the extensive use of these devices in the same location causes some problems as most of them use the same available unlicensed radio spectrum known as ISM (Industrial, Scientific and Medical) bands. Furthermore, other devices such as microwave ovens, remote controls, cordless phones, bluetooth communications, Hi-Fi and video systems also use ISM bands. Although most of these devices have small ranges and use signals with low amplitude, their interference in the ISM spectrum is not negligible.

In industrial scenarios, however, the effects induced by common ubiquitous mobile devices are lower due to the strict control imposed on these environments. Nevertheless, the interference problem still arises because of the multiple wireless devices working in the ISM spectrum such as sensor and actuator devices. Furthermore, as industrial applications demand very strict requirements regarding packet delay and packet loss, every interference may lower the expected quality of service (QoS). In the case of Wireless Sensor Networks (WSNs), which are more and more commonly used to replace old cabled monitoring and actuator systems, the coexistence of different networks and devices results in several problems that span from communication failures to inadmissible response times. These problems are especially serious in critical systems.

Wireless interference has been a widely research area. Studies in [1] showed that at some places the 2,400 MHz frequency spectrum, which is used by several WSNs, has an occupation of 90%. In addition, [2] predicted that in the near future of 5-10 years the growth of wireless communication networks using the ISM bands would suffer overlapping problems, which may extensively affect WSNs. The coexistence of different networks and devices that operate in the same frequency, or in adjacent frequencies, may lead to harmful interference, which in turn limits the capabilities of the applications and, in some cases, results in the complete shutdown of those networks [3].

In trying to solve the problem of WSN coexistence, a vast amount of research was done and some standard recommendations were produced [4, 5]. However, current devices do not yet support most of these recommendations. The new IEEE 802.19 Wireless Coexistence Working Group was formed to deal with coexistence between unlicensed wireless networks and devices. The objective of this Working Group is to develop standards for guaranteeing the coexistence of future wireless devices (CA – *Coexistence Assurance*), i.e., to guarantee that wireless devices can

operate in the same location without causing significant interference to each other.

Some proposals found in literature, targeting cellular networks and other wireless communication systems, use solutions based in Dynamic Spectrum Allocation (DSA). The mechanisms used in DSA include spectrum sensing, choosing the best channel/frequency available and dynamically reconfigure the device radio. These mechanisms have been used in cognitive or intelligent radios. Akan et al. [6] showed that these same techniques may be applied to WSN, mitigating coexistence problems.

To provide a reliable WSN for industrial environment with performance assurances, several studies and projects have been done. Among them is GINSENG project - Performance control in Wireless Sensor Networks [7], which tried to provide a solution for reliable and timely communication, while achieving energy efficient control, targeting WSN at critical industrial environments. The requirements that were found more relevant in these environments were packet delay and loss rate, which had to comply to the very strict performance boundaries. The GINSENG solution to these requirements was to develop a new reliable medium access protocol called GinMAC [8]. This protocol uses Time Division Multiple Access (TDMA) and contains specific mechanism to improve reliability and assure maximum delays. Also, it uses a specific topology control mechanism and implements message routing.

To validate the GINSENG proposal, a real testbed was installed at the oil refinery at Sines, Portugal. In spite of having good overall results, it was found that some problems existed due to the dynamic noise and interferences from the refinery environment, as stated in [9]. This misbehaviour of the WSN existed even in a very controlled area with strict access of personal and wireless devices, within the oil refinery. The cause of the failures was the pollution of the 2,400 MHz spectrum and the electromagnetic noise caused by engines and other machinery. In order to solve this type of problem it was necessary to manually monitor all the available channels under 2,400 MHz frequency and identify the least interference channel to be used for the sensor network. While in most of the current deployments, the channel used by a WSN is manually set before the deployment, remaining static during the all network operation, it became clear that, in the refinery scenario, the manual solution to choose the best channel did not guarantee the network operation. The reason is that in this environment the noise and interference pattern may change with time, resulting in the unreliability of the network during its normal operation lifetime.

Employing techniques of cognitive radio in WSN is still a big challenge, aggravated by the fact that sensor nodes have resource constraints in terms of com-

munication and processing capabilities. Thus, a cognitive based MAC protocol for WSN must take into account all these hardware limitations and still consider the critical time response essential in the industrial application scenarios. Moreover, the protocol should also be able to sense the wireless spectrum, choose the best channel and share that decision with neighbour nodes, while not increasing the packet loss, the energy consumption, or altering the time constraints of the communication.

This paper proposes a new medium access protocol, DynMAC (Dynamic MAC), which uses dynamic channel reconfiguration mechanisms in order to choose the best communication channel for WSN, while supporting the performance restrictions of critical systems. It has the same properties found in cognitive radios (sensing, decision, sharing and mobility of spectrum) and embeds characteristics of the GinMAC protocol developed under the GINSENG project. Furthermore, to maintain the network resilience, a mechanism was developed to automatically recover the nodes from connection losses.

While, in general, cognitive radios use the primary frequencies (licensed bands) on opportunistic mode to be used by the secondary users, our approach is different. In this paper, the main focus is the coexistence of WSN and devices that only use the ISM bands. In order to accomplish this objective, DynMAC protocol implements methods of classification and reconfiguration of channels. It uses the same techniques of cognitive radios but only uses one channel at a time. Therefore, the process of decision employed in DynMAC can be considered as part of a cognitive process, although this concept is still controversial for some authors. In order to evaluate the viability of DynMAC in critical scenarios, experiments were done using simulation and a real WSN testbed. Results showed that the mechanisms proposed in this paper can dynamically solve reconfiguration problems in WSNs operating in noisy and interference environments, while maintaining the application requirements. Furthermore, it enables the coexistence between the WSNs and wireless devices operating in the the same frequency spectrum.

Based on the results it is expected that the DynMAC protocol can be used for any applications that require critical time boundaries and resilience. As WSN are expanding to a broader set of applications and scenarios, where QoS is essential, the importance of guaranteeing that the performance expected initially is maintained over the life of the network is of utmost importance. Although DynMAC was developed mainly to industrial environments there are other examples of scenarios where applications could benefit from the mechanisms studied for DynMAC such as health status monitoring of patients, smart environments, control and monitoring of electrical power, fire system monitoring, gas leaks, industrial

plants, and any other general system that require responses in real time.

This paper is organized as follows. Section 2 presents the related works. The GinMAC protocol developed by GINSENG Project is described in Section 3. The characteristics and functionalities of DynMAC are presented in Section 4. Section 5 shows the experimental results using simulation and the real testbed. Finally, Section 6 presents the conclusions and future work.

2. Related Work

WSNs deployed in industrial environments require strict performance control especially in regards to packet delay and loss. Most of the problems arise during packet transmission and relate to the transmission medium (e.g. signal path-loss, noise and interference) and poor hardware. Controlling the transmission power, improving the antennas and carefully placing transmitters and receivers can reduce the effects of noise, path-loss and interference. However, finding the optimal position in which to put the nodes and antennas is a complex procedure.

Interference occurs due to the existence of more than one network or by the spectral spread in some wireless standards. Solutions to mitigate or avoid interference include selective licensing of spectrum by area, use of hardware standards and use of signal coding. However, in a non-licensed frequency spectrum such as the ISM, there is no guarantee of no interference. Consequently, in this frequency bands, any interference must be tolerated.

There are two different types of interference in WSNs: co-channel and adjacent channel [10]. Co-channel interference is defined as the use of the same frequency by more that one network or device in the same area . This interference cannot be solved by increasing the signal power, as that would increase the interference and the communication problems in neighbour networks. Adjacent channel interference occurs when consecutive channels are used by different devices in the same area, which leads to the degradation of both signals. This interference type can be minimized through careful filtering and channel assignments. However, it may not be sufficient because the filters and receivers used by radios are imperfect (as in the WSN) and do not cancel all the interference.

Because IEEE 802.15.4 based WSNs [11] operate in the same frequency band as that of WLAN (IEEE 802.11 b/g/n), a very used standard, interference will occur if they are deployed in the same area. As shown Figure 1 these two standards divide frequency band into channels of 5MHz and 22MHz respectively and they overlap in almost all extension of the spectrum.

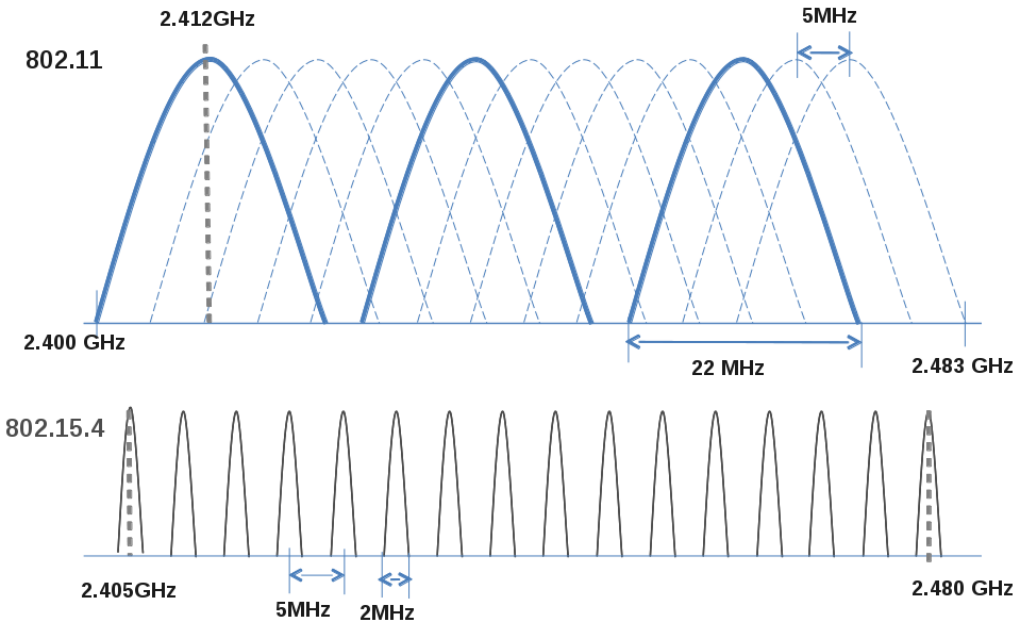

Figure 1: Allocation frequencies to standards IEEE 802.15.4 and IEEE 802.11.

Several studies were done to understand how the interference from these two types of networks affect each other. In [12] the authors investigated the low-power transceivers used in WSNs and analysed the models proposed for low-power wireless communications, considering both noise floor and interference. Results revealed inaccuracy in the models and in the packet reception algorithms used in the existing simulators. Thus, this indicates the importance of conducting real experiments with WSNs to evaluate the effects of interference.

The studies in [13, 14, 15] showed that by deploying ZigBee/IEEE 802.15.4 and WLAN networks in the same location, their performance in term of packet loss was significantly affected by interference. Authors in [16] found that with the interference of 802.11 networks, the packet error rate (PER) of 802.15.4 can be up to 95% when the interferer is in the distance of 1.5 meters. Inversely, the throughput of a 802.11 network can be reduced up to 30% when there exists an 802.15.4 based network in a short distance. In addition, Toscano and Bello [17, 18] tested environments only consisting of WSNs and found that interference still exists. Using 802.15.4 networks, their results showed that interference depends on the transmission power, on the distance between nodes and on the duty-cycle.

Zhou et al [2] prove experimentally how electronic devices operating at 2,400 MHz can cause interference and even hinder the operation of WSNs based on the 802.15.4 standard. To avoid this, the authors proposed a multi-channel approach for WSNs, which have multi-frequency radios. They also proposed a middleware between the physical and MAC layers in order to support multiple channels. However, this middleware uses only the CCA (Clear Channel Assessment) method to choose the best channel, not taking into account the different local noises in each node. Moreover, an evaluation of the middleware, either by simulations or practical experiments, was not found.

In order to guarantee the coexistence of different networks in the same area using non-licensed frequency spectrum, in the year 2000 the IEEE created the IEEE 802.15.2 Task Group - TG2 to study the interactions between WLANs and WPANs, resulting in the "Recommended Practice for Information Technology - Coexistence of Wireless Personal Area Networks With Other Wireless Devices Operating in Unlicensed Frequency Bands" [4]. These recommendations are based in mechanisms targeting the physical and link layers. However, the group TG-2 was hibernated. Recently, the work group IEEE 802.19 was created to continue the previous recommendations and to develop a standard for coexistence between wireless devices operating in unlicensed band [5]. However, current devices do not support these recommendations yet.

The proposed solutions for the coexistence problem of wireless communication can be divided into two main different classes regarding their mechanisms: collaborative and non-collaborative. Collaborative mechanisms are based in the exchange of information between all networks (inter-networking) in an effort to minimize the mutual interference by negotiation (e.g. networks synchronization and mapping of users in the same area). When the exchange of information among the networks is not possible, non-collaborative methods are used. In this case, the nodes only exchange information about the intra-networking conditions (e.g. Dynamic Spectrum Access (DSA), Software Defined Radio (SDR) and Cognitive Radios (CR)).

A coexistence collaborative mechanism is presented in [19], which explores the synchronization capabilities of WSNs. Channels management is obtained by exchange of messages between a central entity and the WSN coordinators. It is assumed that all WSN coordinators have wireless and wired interfaces and that the communication to the central entity is made using an Ethernet network. Results showed that the nodes' lifetime were increased and that coexistence was possible. Nevertheless, no mechanism for dynamic reallocation of channels was designed.

A Dynamic Spectrum Access Protocol (DSAP) [20] is a collaborative mecha-

nism for coexistence with central coordination. This protocol aimed at mitigating the congestion and interference between networks by adjusting the channels used by the nodes. To accomplish this, the central coordinators must have previous knowledge of the channels in use in the region. These coordinator nodes use two interfaces; one is used to communicate with its client nodes (by common channel) and other to access the informations about the channels' usage in the region. Results show that the DSAP reduces interference and improves the throughput. However, a wrong choice of the common channel or the effects of dynamic noise in the communication environment may disrupt the communication between nodes.

According to [14] technological advances in Cognitive Radios (CR) enable the application of DSA models to WSN to efficiently use the frequency spectrum and to support the coexistence of networks. To accomplish that, CR explores the frequency spectrum using an opportunistic mode, adjusting the transmission parameters (frequency, transmission power and codification) based on the information gathered from the environment [21]. Other approaches using CR are discussed in [22, 23] and proposes that CR, besides the programmable reconfiguration, may also learn to adapt to the surrounding environment by reconfiguring, in an intelligent way, all the transmission parameters.

In [24], two spectrum decision protocols for Cognitive Radio Sensor Networks (CRSN) are proposed, which provide distributed mechanisms to select the best wireless channel based on the application's QoS requirements. Simulations of low, medium and high noise scenarios have shown that the protocols improve the delivery rate while keeping the delay and energy consumption unchanged. However, the protocols are based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) whose response time is not deterministic and therefore are not suitable for industrial environments. In addition, these protocols were not yet validated using a real testbed.

Frequency Hopping Spread Spectrum (FHSS) has been used to provide for WSN coexistence. According to [25], channel hopping can reduce interference and increase resilience between neighbour networks. However, this mechanism uses a pseudo-random sequence to make the frequency hops and does not take into account the channel quality. If interference exists in more than one channel, FHSS technique may result in loss of packets and intermittent disruptions.

To reduce the number of collisions and to improve the reliability of WSNs, TDMA mechanisms were applied in MAC protocols for WSNs. In [26], several multi-channel MAC protocols are classified and discussed, as Y-MAC, HyMAC, TMCP, and MMSN. However, the authors presented evidences that these protocols have problems, as increasing the contention and collisions on the network.

Moreover, they proposed the MC-LMAC (Multi-Channel LMAC) protocol [26]. This protocol is based in the L-MAC (Lightweight Medium Access), that is an energy-efficient protocol that minimizes the overhead on the physical layer by reducing the number of on-off transitions in the transceivers. The MC-LMAC extends the LMAC protocol to the multi-channel domain. It was designed with the objective of maximizing the throughput of WSN by coordinating transmissions over multiple channels. Other protocol that was designed to improve the throughput of WSNs is the TreeMAC [27]. It allows nodes to obtain the bandwidth proportional to their demand, but it was designed for data collection, i.e., it only supports an unidirectional traffic model from the nodes to the sink. Also, none of these three protocols has a mechanism to deal with noise and interference environments.

Despite the efforts being made by IEEE and many other researchers, the coexistence problem of WSNs in the non-licensed frequency spectrum is still not resolved yet. The proposed MAC protocol, DynMAC, presented in this paper employs the cognitive radio principles to enable the dynamic choice of the best channel and adds mechanisms to improve the resilience and reliability for WSNs. Because the current version of DynMAC is based on GinMAC, which was developed by GINSENG project, the next section describes an overview of this project and presents the detail of the protocol.

3. GINSENG project and GinMAC

The aim of GINSENG project is to propose, develop and deploy a solution for performance controlled WSNs that guarantee reliable and timely data delivery [7]. The targets of this project are critical environments such as the oil refinery in which the time response, packet loss and reliability are bounded by some constraints. In order to fulfill these requirements, the GinMAC protocol was developed to provide a reliable and energy efficient control for wireless sensor networks [8]. GinMAC assures a time-critical network by using a TDMA technique and also by implementing a topology control mechanism that ensures the reliability of time-critical data delivery. As a case study, the GINSENG project implemented two WSNs in the oil refinery at Sines, Portugal.

DynMAC protocol adds an additional reliability mechanism to the existing GinMAC protocol. As its implementation is based on GinMAC, next subsections presents the details of GinMAC, namely the topology control mechanism and the TDMA frame structure.

3.1. Topology control mechanism

The network topology proposed in the GinMAC is a tree structure in which the sink node is the root of the tree and the other nodes, consisting of sensors and actuators, are the child nodes. The topology of GinMAC based sensor network is defined by two main parameters: maximum hop distance H and fan-out degree $O_h(0 \leq h \leq H)$ at each tree level h . The O_h specifies the maximum number of children associated with each node at level h. It is important to note that the fan-out degree at the root node is always defined as $O_0 = 1$, because there is only one root node in a tree.

From the above parameters, the maximum number of nodes (N^{max}) that this topology can accommodate is calculated by Equation 1 as:

$$N^{max} = 1 + \sum_{n=1}^{H} \prod_{h=1}^{n} O_h. \tag{1}$$

The parameter *n* in Equation 1 is the index variable. Its value indicates the level at which the maximum number of nodes is calculated. The inner part of the Equation 1 is used to calculate the maximum number of nodes at level *n*. In the GINSENG project the maximum number of nodes (N^{max}) was limited to 25 nodes per tree, due to the time restrictions used in industrial application.

The GinMAC protocol uses "offline dimensioning" mechanisms for defining the network topology before deployment. This means that based on its ID, the node is aware of its position in the tree and of the time slots assigned to it. In addition, it also knows its parent and children. Therefore, each node can detect whether or not it can communicate with its parent and children. At the same time, the offline dimensioning allows for the assurance of a pre-determined level of performance. To be able to tolerate network dynamics the GINSENG project includes a Dynamic Topology Control (DTC) module. Whenever problems or possible optimizations are detected, the DTC may choose to change the network topology or setup, incrementing the operational efficiency of the network. The tree is constructed in a distributed manner by using Layer 2 signalling (through GinMAC). The only drawback of the GinMAC mechanism is the scalability because an increasing number of nodes will correspond to a bigger frame, and to increasing times for the nodes to access the network.

3.2. GinMAC time slot allocation

GinMAC uses TDMA slots exclusively, i.e., a slot allocated to one node cannot be reused by other nodes in the network. Each sensor or actuator has a dif-

ferent time slot to transmit and receive data to/from the sink node. The GinMAC super-frame, F, contains a number of basic slots that allow each sensor to send one message to the sink and the sink can transmit a command to every actuator. In addition, it also contains additional time slots to improve the transmission reliability. Moreover, GinMAC frame can also contains unused slots to improve energy consumption.

The GinMAC basic slots accommodate two different types of traffic: the upstream and downstream. The upstream is the traffic flow from sensor nodes to the sink. Each leaf node requires one time slot within F to communicate data to its parent node. The parent node requires a slot for every child node plus one for its own data to forward to its parent. Therefore, the number of upstream slots for each node at level h is calculated by Equation 2:

$$S_h^{up} = O_{h+1} \times S_{h+1}^{up} + 1,\ where\ \ S_H^{up} = 1. \tag{2}$$

Because there are $\prod_{i=1}^{k} O_i$ nodes at level k, the total number of basic upstream slot for this level is $S_k^{up} \prod_{i=1}^{k} O_i$. Consequently, the number of basic upstream slots in the entire network is calculated as in the Equation 3:

$$S^{up} = \sum_{h=1}^{H} S_h^{up} \times \prod_{i=1}^{h} O_i. \tag{3}$$

On the other hand, the downstream is the traffic from the sink to the actuators. The sink should be able to send a data packet to each actuator within a frame. Therefore, the sink requires at least as many basic slots as the number of actuators. In addition, the number of slots allocated for nodes at level h is the minimum between the maximum actuators in the network and the number of nodes below this level. Assuming that the maximum number of actuator nodes in the network is N_a^{max} , the number of basic downstream slots for each node at level h is calculated by Equation 4 as:

$$S_h^{down} = min\left(N_a^{max}, \sum_{i=h+1}^{H} \prod_{j=h+1}^{i} O_j\right), with\ \ S_H^{down} = 0. \tag{4}$$

The number of basic slots for downstream is computed using the Equation 5:

$$S^{down} = \sum_{h=0}^{H-1} S_h^{down} \prod_{i=0}^{h} O_i. \quad (5)$$

Thus, the total of time slots is $S_{total} = S_{up} + S_{down}$.

As an example consider a network topology where $H = 3$, $O_1 = 3$, $O_2 = 1$ and $O_3 = 2$. Applying the Equation 3 we have $S^{up} = 27$. Assuming $N_a^{max} = 2$, from equations 4 and 5, the total number of basic downstream slots is $S^{down} = 14$. The basic slot allocation of the GinMAC frame for this example network is shown in Figure 2. This example uses the same network topology that was used in the DynMAC testbed (Figure 3 – Section 4).

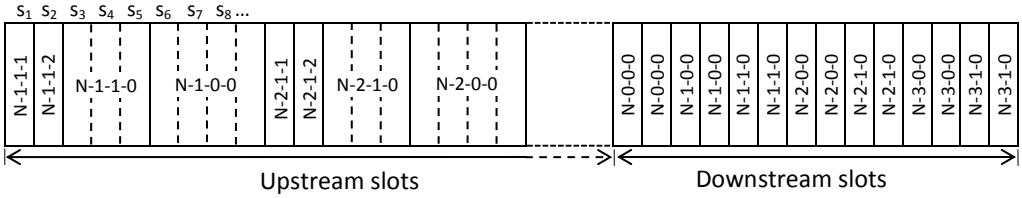

Figure 2: GinMAC TDMA frame structure.

4. DynMAC: dynamic channel allocation protocol for WSN

The goal of the DynMAC protocol is to reduce the effects of adjacent and co-channel interference in WSNs installed in environments in which other wireless networks and devices exist. A model of noise that reflects the attenuation and interference imposed by the environment in the signals and packets transmitted between nodes, is shown in [28]. Generally, in these environments the frequency spectrum of WLAN may be very polluted and the level of noise and interference are high. Although the original protocol GinMAC has mechanisms to reduce the problem of loss packets and to improve the delivery rate, it does not have an efficient mechanism to automatically avoid interference and allow the coexistence of networks.

One of the mechanisms used in GinMAC relied in the native implementation of CCA by the CC2420 radios used in the nodes. This mechanism detects if the

medium is busy before transmitting any data, avoiding collisions and possible interferences, improving the reliability of the channel. Unfortunately, this mechanism is not sufficient when there is interference in adjacent channel or when the duty cycle of the nodes is high.

In general, WSN radios are equipped with mechanisms to indicate the quality of the received signal, such as the Received Signal Strength Indicator (RSSI), Signal Noise Ratio (SNR) or the Link Quality Indication (LQI) [29]. In this paper, the RSSI is used to detect the energy of the received signal. As the SNR is highly based in the RSSI value, it is not considered in our experiments. The LQI is a characterization of the quality of a received signal, normally combining the RSSI value with a correlation of the expected and received data. However, LQI estimation method varies with its implementation. Also, according to [30], MAC protocols performance can vary over different channels with RSSI values being asymmetric in the links. In this way, we opted for a simple RSSI mechanism that is available in the radio of WSN nodes and that reflects the strength of the signal received by each node. This RSSI value is obtained periodically by each node offering a good approximation of the channel conditions.

DynMAC adds an additional reliable level to the GinMAC protocol by including the dynamic evaluation of the spectrum, the recovery from lost links and the reconfiguration of transmission parameters, techniques similar to those used in cognitive radio. The implementation of this protocol followed the same cognitive phases, described in the frameworks proposed by [6, 24].

The design of DynMAC is based on the techniques of non-cooperative coexistence. Thus, there is no centralized coordination between networks or mapping users and channels in the same region. This protocol takes into account the industrial environments and has the same operating restrictions as those assumed by GinMAC. In order to ensure network requirements such as delay and delivery rate, a time-slotted technique (TDMA) is used, so the sink node knows all the nodes in the network, as well as their hierarchical position, and the best channel decision is computed at the sink node. The only information used by DynMAC to choose the global best channel is the one exchanged between nodes in the tree, each providing its local information. There is no mechanism for exchanging information, of the channel conditions or of the nodes location, with nodes outside the tree. Furthermore, to ensure the resilience of network, the common control channel (CCC) is not used in the implementation.

The cognitive characteristics of DynMAC protocol are in the monitoring of environment conditions, assessment of channels (good and bad), decision of local and global best channel, sharing of informations with network nodes and in

mobility of channels. This cognitive capacity of the protocol allows the dynamic and periodic reconfiguration of channels, taking into account the variations of the environment noise and interference. Furthermore, to ensure the resilience of the network, two mechanisms were employed in DynMAC. Firstly, every nodes (except the sink) monitors the Packet Error Rate (PER) over a specified period of time. If the PER of the last time period is exceeded a threshold, e.g., 5%, the node will send an alert to the sink. Based on the alerts received from the nodes, the sink will decide when a re-evaluation of the global best channel is needed. For instance, the sink will start the re-evaluation of the global best channel when it received more than 3 alerts from the normal nodes (nodes that are not the sink). Secondly, the node verifies if a lost link occurs and tries to recover the connection by forcing the node to rescan for the channel on which the sensor network is working.

The DynMAC protocol is based on TDMA and relies on the assumption that the sink node knows both all nodes in the network and their hierarchical position. The nodes can only communicate with its direct parent or children. It means that to send a packet to the sink, a node has to send it to its direct parent node, which forwards the packet to the upper level in the tree. This process is repeated until the packet reaches the sink. In addition, the broadcast message from the sink is also transmitted using the multiple hops method. The synchronization is assured by the sink node, which creates the super-frame and designates the time-slots to each node in the network. Thus, specific time-slots are allocated for every node to transmit, receive and acknowledge data. It is assumed that the network topology is static and a list of possible nodes, i.e., nodes with specific identifications, and their positions in the tree are predefined. Figure 3 is an example of the tree topology implemented in the DynMAC testbed, and used for doing experiments for this paper.

The solution for providing reliability for WSNs proposed in this paper consists of four main phases: (1) network synchronization, (2) global best channel selection, (3) periodically re-evaluation of best channel, (4) recovery from lost link, which are presented next.

4.1. Network Synchronization

This phase is only run once to form the WSN. The processes running on the sink and on normal nodes are different. In order to avoid problems with the choice of channel, the process on the sink node does not use the CCC because a wrong choice of the channel may lead to unstable network operation. Instead, the sink

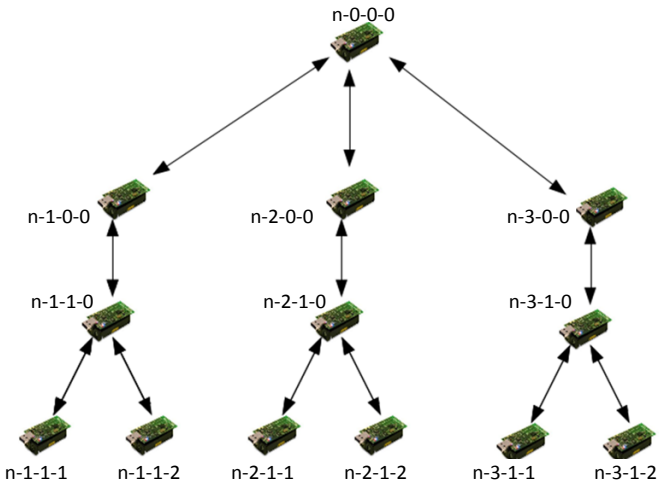

Figure 3: Tree topology of the network.

node chooses the local best channel based on RSSI and then broadcasts the channel information to the network. On the other hand, the process running on the child nodes scans the different channels until it receives a message from a specific sink node. These two processes are implemented at MAC layer.

4.1.1. The local best channel

This process is run by the sink to evaluate the best channel. The best channel in this case is called the local best channel because it is only based on the local information collected by the sink node. The primary information currently used to evaluate the channel quality is the RSSI. It is important to note that the RSSI value is associated with a received packet and indicates the signal strength. Therefore, the higher RSSI value is, the better is the quality of signal. However, the RSSI values collected when no nodes are transmitting indicate the noise and interference. Thus, the meaning of RSSI values is reversed, i.e. the higher RSSI values correspond to the channels with lower quality, being the bigger value indicative of the worst channel of all. Accordingly, the best channel is the channel that has the least accumulated RSSI. Additionally, another metric, which can be used in evaluating the best channel, is the incidence of the RSSI samples that are greater than the CCA threshold values. Algorithm 1 employs both these metrics in calculating the cost of each channel.

```
1  Procedure sink_local_best_channel
2  for (i in 1 to N) do
       /* N: number of times the spectrum is sensed  */
       /* Scan of channels 11 to 26 (16 channels)    */
3      for (c in 11 to 26) do
           /* Set c as the current channel           */
4          set_channel(c)
5          current_rssi = get_current_rssi()
6          rssi_values[c-11]+= current_rssi
           /* Detect RSSI values above threshold value
              */
7          if (current_rssi > CCA_THRESHOLD) then
8              rssi_above_threshold[c-11]++
   /* Sorting accumulated rssi in ascending order   */
9  acc_channel_index = index_ sort(rssi_values)
   /* Calculate channels cost and get the best
      channel                                        */
10 channel_index = cost_computing(acc_channel_index,rssi_above_threshold)
11 best_channel = channel_index[0] + 11
12 return (best_channel)
```

Algorithm 1: Sink node local best channel selection.

Initially, during the spectrum sensing, the sink node collects N samples (N is a empirical value) of local RSSI values for each of the 16 channels, from 11 to 26 (lines 2–6). For each channel, the accumulated RSSI value is collected and stored in the *rssi_values* array (line 6). In addition, the incidence of the RSSI values that is greater than the CCA threshold is also counted and stored in the array *rssi_above_threshold* (lines 7–8). After finishing the spectrum sensing, the channels are sorted in ascending order of the accumulated RSSI values (line 9). The index sorting is used to make the process of selection of best channel easier (the element at the first index (0) is the best channel). In addition, because sensor nodes can only work with integer values, the usage of the average of RSSI values of the each channel may produce an incorrect order. Therefore, the sum of RSSI values is used as one of the metric to select the best channel instead of average. Then, the cost of each channel is calculated and sorted by calling *cost_computing* (line

10). The process of computing the cost for each channel is shown in Algorithm 2.

Because channels are sorted ascendantly by its cost, the best channel is the one at the first position (line 11). The current best channel found is also called Local Best Channel since it is only the result of the spectrum sensing at the sink's location.

```
1 Procedure cost_computing
2 cost=1
3 for (i in 0 to15) do
      /* Get the channel at the index i                    */
4     channel = acc_channel_index[i]
5     channel_cost[channel]=cost
      /* Add the frequency as additional cost              */
6     channel_cost[channel]+=rssi_above_threshold[channel]
      /* Computing the cost based on accumulated rssi
         for next channel                                  */
7     if ((i<15) & (rssi_values[acc_channel_index[i+1]] >
      rssi_values[channel])) then
8         cost++;
9 return index_sort(channel_cost)
```

Algorithm 2: Computing Cost for Channels.

The cost of each channel is computed based on two pieces of information: (1) the accumulated RSSI value and (2) the frequency that the collected RSSI values are greater than the CCA threshold. To make it easy to discuss, we consider the first element as the RSSI cost and the second one as the threshold cost. As shown in Algorithm 2, the initial RSSI cost for the smallest accumulated RSSI value is one (line 2). For each channel, its cost is first assigned to that of the RSSI cost (line 5). Then, the threshold cost for that channel is added (line 6). It is important to note that channels having the same accumulated RSSI value will have the same RSSI cost. In addition, when the accumulated RSSI changes, the RSSI cost is increased by one unit (lines 7-8). The final result of this algorithm is the index sorted list (from lowest to highest cost) of the channel (line 9).

After choosing the best channel, the sink node sets it as its communication channel. Finally, the sink node must inform all other nodes in the network of the channel it is using. Thus, it builds a control frame with the network's GID (Group Identification) and broadcasts it to the network.

4.1.2. Network joining

The child nodes have to search the communication channel used by the sink to join the network. The process used by child nodes to detect the channel on which the sink node is running is described by Algorithm 3.

```
1  Procedure ScanningChannel
2  C_i = 11
3  while (true) do
4      set_channel(C_i)
5      if (is_exist_frame( )) then
           /* If a frame exists in channel          */
6          get_frame( )
7          if (is_valid_frame( )) then
               /* Frame ∈ sink's GID                */
8              break
9      if (C_i < 26) then
10         C_i ++
11         else
12         C_i = 11
13 return (C_i)
```

Algorithm 3: Detecting the communication channel of the sink.

During the initialization phase, the child node repeatedly scans the 16 channels in order to find the communication channel of the sink node. This process starts by setting the node on channel 11 (line 2 and 4). At each channel, the node checks whether there is some frame waiting to be received or not (line 5). If there is one, it will get and analyse the frame to see if it is from a valid sink, i.e., if the GID in the frame is equal to that of the node (line 7). If a valid frame is found, the scanning process is stopped (lines 7–8) and the node finishes its booting process. On the other hand, if there is no frame, or the frame is not from a valid sink, the node will sense the next channel in the sequence (lines 9-12). This process is repeated until the node detects the communication channel of the sink.

After detecting the sink's channel, the node will join the network by setting its radio channel to that of the sink and starts exchanging a joining frame with the sink. After receiving the joining frame from the node, the sink saves information about this new joining node and sends an ACK message back to the node. The

ACK message informs the node that the joining phase is completed.

Because the best channel selected by the sink is based on its local information, it may not be a good channel for the whole network. This is because the noise and interference at the child nodes might be different from those at the sink node. Thus, in order to choose the global best channel, a second phase is needed.

4.2. Global best channel selection

Because the best channel chosen by the sink node only uses its local information, it may be not the good channel for its child nodes. Consequently, it is necessary that all nodes in the network contribute to the process of choosing the best channel. This process should be started when all the nodes join the network. However, in a real environment the full network may never be reached for many reasons (e.g. some nodes are not available). Therefore, we define that a network is in a *stable* state if there is at least one node different from the sink that joins the network for long enough. When the network reaches the *stable* state, the global best channel process is started. The steps of this process are shown in Figure 4 and described below.

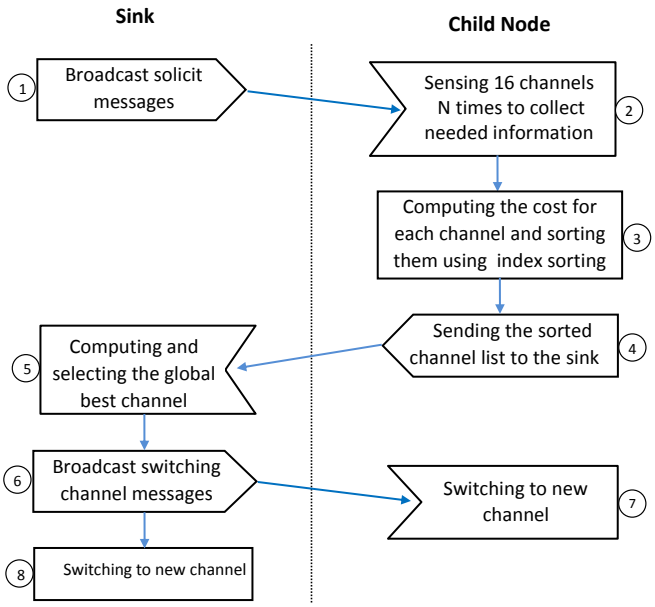

Figure 4: Choice of the global best channel.

(1) The sink node broadcasts solicit messages to collect the channel information of the other nodes. The solicit message is sent 3 times to make sure that every node receives at least one request.

(2) When receiving the broadcast messages from the sink, the normal node will forward those messages to its children. It then samples RSSI values on every channel a number of times. The process of sampling of RSSI values is the same as the one done at the sink in the local best channel selection process (Algorithm 1).

(3) Each node uses Algorithm 2 to compute the cost of each channel and to sort them in ascending order of the cost.

(4) The channel list ordered by cost, computed by each node (that is not the sink) in step 3, is then sent to the sink node.

(5) When the sink node has enough information from network nodes, it will decide which channel is the best one. The process of choosing the global best channel is described in Algorithm 4.

(6) After the sink node chooses the best channel, if the best channel is different from the current working channel, the sink will broadcast three switching channel messages to ask other nodes to switch to the new channel.

(7) When receiving the switching channel message from the sink, every child node will forward it to its own children. To ensure that at least one of these broadcast messages reaches the leaf node, all intermediate nodes will wait for the three switching channel messages or for a time-out of predetermined duration. After the waiting condition pass, it will switch to new channel.

(8) The sink node will switch to new channel after sending the last switching channel message.

As the sink receives an ordered list of channels (ordered from the best one to the worst) from each node in the network, Algorithm 4 is employed to decide which channel is the best for the network.

The global best channel is selected based on the total cost of channels. In order to compute the total cost for channels, it is necessary to assign a cost for each position of the ordered lists obtaining from the normal nodes. For simplicity, the cost of the channel is also its position in the sorted list, i.e., the cost of the

```
1  Procedure GlobalBestChannel( )
   /* Compute the cost for each channel                    */
2  for (node_id ∈ nodeList) do
3      for (pos in 0..15) do
           /* Get the channel at pos                       */
4          channel = Channel_index_list[node_id][pos]
5          channel_cost[channel]+=pos+1
           /* Store the worst channel list                 */
6          if (channel_index_list[node_id][channel] == 15) then
7              Bad_channel[channel]+=1
   /* Sorts channels by cost and store in
      channel_index_global                                 */
8  Sort_channel(channel_cost, channel_index_global)
   /* Choosing the best channel                            */
9  best_channel = channel_index_global[0]
   /* Reselect the best channel if the current best
      channel exists in bad_channel_list                   */
10 c = 0
11 while ( (c < 16)&&(bad_channel[c − 1] > 0) ) do
12     c++
13     if (c < 16) then
14         best_channel = channel_index_global[c]
15         break
16 return best_channel
```

Algorithm 4: Choosing the global best channel.

first position is 1, the second position is 2, etc. Consequently, the total cost of each channel is calculated as described in lines 2–5. Because the value of variable *pos* starts at 0, the cost for each channel should be calculated as *pos*+1. To avoid choosing the worst channel of some nodes in the network, i.e., the channel located in the last position of the ordered channels, the sink node also stores a list of the worst channels (lines 6–7). The list of costs of the channels, is then sorted, in ascending order by cost, and stored in the *channel_index_global* array (line 8). The global channel best is the channel in the lowest position in the channel cost list that does not appear in the worst channel list (lines 9–14). In case there is no

such channel, the channel in the first position of the channel cost list is chosen as best channel.

4.3. Periodically re-evaluation of the best channel

Because the quality of the channel varies with time, the best channel at one period may become the worst (e.g., high lost rate) during the operation of the network. As a result it is necessary to periodically re-evaluate the quality of the channel. Two possible approaches are: (1) the sink node periodically sends a request to the normal nodes to solicit the channel information and re-elect the best channel; (2) the normal nodes periodically evaluate the quality of the current channel based on different metrics, such as error rate of the sent packets or on the number of lost links, and send these information to the sink.

In DynMAC, the second approach is taken using the PER (Packet Error Rate) as the evaluation metric. Periodically every normal node checks its PER and if it is higher than a threshold (e.g., 1%), the node will send a message to the sink node stating this information. When the sink node receives the information, it will decide whether there is a need to re-select the global best channel. Currently, the sink makes the decision based on the number of nodes that send the re-evaluation request. However, because sensor networks usually operate for a long time, if the pure accumulated PER is used for evaluating the current condition of the channel, then it will take very long to detect dynamic noise and interference. Consequently, only the PER obtained in the last interval of time (that contains measurements of the most recent conditions) is employed to evaluate the channel condition. Particularly, in our implementation, the PER of the last 5 minutes is used as the evaluation metric. The best channel re-evaluation process is the same as the process of the global best channel selection described in Section 4.2.

4.4. Recovery from lost link

During network operation any node may lose the connection to its parent for different reasons. One example of such scenario is that the node did not receive a switching channel message from the sink. In this case, the sink switches to the new channel, leaving the node working in the old channel. Therefore, it is necessary to have a mechanism for the node to recover from such a situation. To resolve this issue, the node is forced to re-scan the communication channel of the sink if it cannot communicate with its parent for a specific duration. In this case the child node uses the Algorithm 3 to scan the sink's channel.

4.5. Energy consumption issues in DynMAC

Energy consumption is one of the main concerns in WSN, and a trade-off between functionalities and the energy wasted to implement them, must always be studied. In this paper no specific tests on energy issues were done as DynMAC relies on the basis of the GinMAC protocol, which limits by design the energy wasted. The primary goal of GinMAC is to improve reliability and message delivery delay. However, it does not neglect energy efficiency. Because a node in a network has knowledge of its allocated time slots, it will sleep during the time slots of other nodes. To the topology depicted in 8 corresponds a super-frame of 50 time-slots, resulting in a duty cycle of 2% for each leaf node. Similarly, the maximum duty cycle of nodes at Level 2 is of 20% and that of nodes at Level 1 is of 42%. In addition, the super-frame can include extra slots in which all nodes in network will go to sleep to improve the network lifetime.

During the DynMAC operation there is one situation where an evaluation of the trade-off between wireless medium scanning and packet loss rate is more relevant, in the periodic re-evaluation of the best channel. The initial set-up phase is only done once and the recovery from a lost link is a trade-off between not having any communication and the effort wasted in reconnecting. However, even in the periodic re-evaluation of the best channel, the calculations are resumed to the PER and only if it exceeds a previous defined threshold, a message is generated. In the worst case, each node sends a message every 5 minutes (that can be configurable) for evaluation by the sink (using also information from other nodes), and only then a re-selection process may take place (Section 4.3).

5. Experiments and Results

In order to evaluate the mechanisms implemented in the DynMAC protocol, the Contiki operating system, together with TelosB sensor nodes was used [31]. As a first step, DynMAC was evaluated using the COOJA simulator provided together with ContikiOS [32]. COOJA simulator was used to make a first evaluation of the settings that would be used later in the real testbed. The structure of the network used for experiments is shown in Figure 3.

The topology of the WSN is a tree with one sink and 3 levels. The distance used between levels was of 5 meters and the communication range was set to 25m. These values aim to prove that the proposed protocol can avoid internal interference, i.e., the interference created by nodes inside the WSN. However, in real scenarios there is no constraint on the distance between levels, which depends

only on the radio coverage area. The unit disk graph radio model is used in simulation. Within this network, the total number of slots in the super-frame was set to 100. Therefore, each super-frame has 1000 ms (10 ms per slot). For testing the scanning and network coverage time, the experiments were repeated 300 times. To measure the packet loss within the testbed, for each channel, each node was set to send 1000 packets to the sink, with an interval between packets of 5 seconds.

5.1. Cooja Simulation

Cooja simulator was used to test the functionality of the proposed algorithms. Specifically, the measurements of the scanning and coverage time of the network, as well as the handoff time, were tested. For the simulation only internal interferences caused by extra nodes were considered.

5.1.1. Scanning and network coverage time

The first simulation was done to measure the scanning time of the nodes and the network coverage time. For the scanning time, it measured how long it took the nodes located at different levels to detect the channel on which the sink node was running. For the network coverage time, it measured how long it took for all nodes to successfully join the network. In these tests, the time was always measured from the moment when the sink node finished booting. Results of the experiments concerning the scanning time are shown in Table 1 and in the Figure 5.

Statistic	**Level 1 Scanning time (ms)**	**Level 1 Finish booting time (ms)**	**Level 2 Scanning time (ms)**	**Level 2 Finish booting time (ms)**	**Level 3 Scanning time (ms)**	**Level 3 Finish booting time (ms)**
Minimum	112	160	24	72	24	136
Q1	2228	2652	784	1170	822	1074
Median	3798	4054.5	2104	2352	2236	2468
Q3	4860	5462	2483.75	2753.25	4264	4554
Maximum	8648	8920	8872	8984	14264	14856
Average	3842.68	4175.21	2309.26	2569.97	3056.15	3332.00

Table 1: Simulation – scanning and booting time of nodes located at different levels.

As shown in Table 1, during the booting process, the nodes at Level 1 could immediately detect the channel on which the sink was running, i.e., on the first super-frame (112ms). In addition, it took a maximum of 9 super-frames (8648

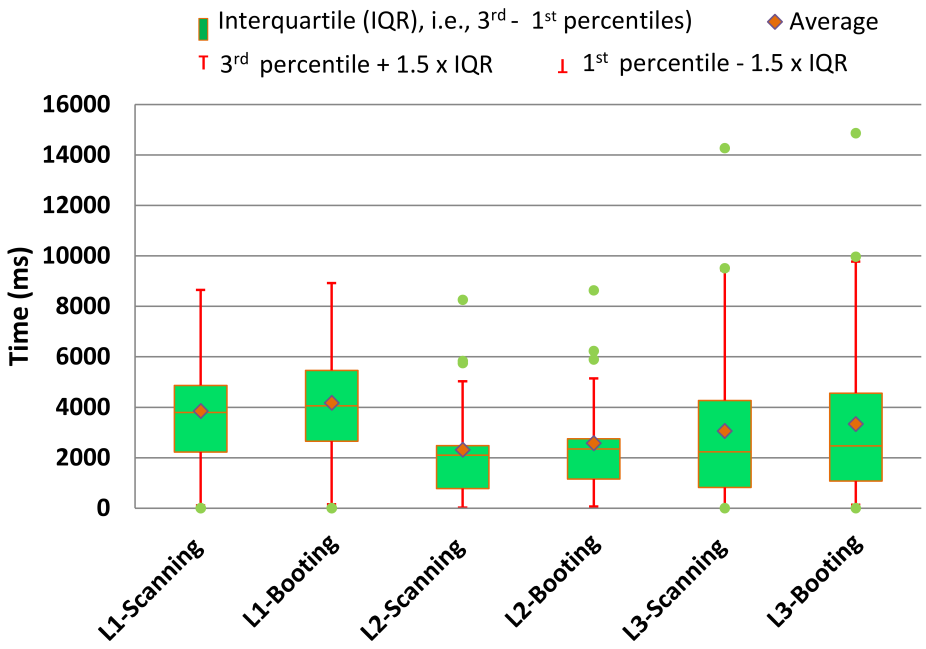

Figure 5: Simulation – scanning and booting time of nodes located at different levels.

ms), and an average of about 4 super-frames (3842.68 ms) for the nodes at Level 1 to scan the communication channel of the sink. Nodes at Levels 2 and 3 also needed a similar amount of time to detect the channel of the sink (or of their parent). This means that the node could detect its parent on the first super-frame. Furthermore, most of the nodes (75 percent) could detect their parents during the first 6 super-frames. However, for the node at Level 3, sometimes it took a little bit longer, up to 15 super-frames (14264 ms).

In addition to the scanning time, the nodes also need time to initialize the parameters for their environment to finish the booting process. However, in most cases, it took less than 600ms. In our simulations, after detection of the parent channel, the node needs a maximum of 729 ms.

From Table 1, the coverage time of the network, i.e., the maximum time for all nodes in the network to detect and join the network, can be estimated. In this case, the coverage time was less than 33 seconds (8920ms + 8984ms + 14856ms = 32760ms). This experiment was repeated a numerous times to measure the coverage time and in all the experiments it took less than 17 seconds (maximum 16920ms).

5.1.2. Handoff time

Handoff time is the time that a node (e.g., mobile nodes) takes to rejoin the network, that is, resynchronize itself with its parent after having left the network. From the simulation results, it took from 3ms to 384ms for a node to resynchronize with its parent node. If this node is not a leaf, its children also have to resynchronize with it and with the sink. From the experiment results, it can be seen that it was very fast for other nodes to resynchronize with their own parent (maximum 678ms).

5.2. Experiments in real testbed

In addition to simulation, a testbed was also set up using TelosB motes. The objective of the experiments using a testbed was to validate the mechanisms proposed in DynMAC. Interferences were added by using different Wi-Fi networks in range. More important, the experiments with the testbed proved that the proposed solution is feasible for a real sensor network.

5.2.1. Evaluation of the quality of different channels

The first experiment was done to evaluate the quality of different IEEE 802.15.4 channels and employed the Algorithm 1, described in Section 4.1 to identify the best one. The test was repeated 1500 times and the number of times that each channel was chosen as the best channel, as well as the worst, was counted.

The results of these experiments are described in Table 2 and in Figure 6.

Out of 1500 tests, the channel 26 has appeared as the best channel for 833 times (55.53%). Channels 25 and 26 have high probability of being chosen as the best channel because these channels are out of the IEEE 802.11 WLAN spectrum. Other channels that also appeared as good channels with a high incidence were 25 (9.07%), 16 (6.4 %), 17 (6.07%), and 13 (5.06 %). On the other hand, channel 23 appeared as the worst one with a rate of 42.8 %). The second and the third worst channels were 24 (21.13%) and 11 (13.73%) respectively.

As shown in Figure 6, there were several cases in which a channel was sometimes identified as the best channel at a time and identified as the worst in other time. One possible reason for this is that the noise and interference are dynamic and the wireless communications among devices might also have gap intervals when no traffic is generated. Consequently, if the sink scans a channel during a gap interval, it is more likely that this channel is identified as the best channel.

One important point to note is that the result of this experiment correctly reflected the current environmental conditions. It can be seen from the Figure 6

Channels	Incidence as the best channel	Incidence as the worst channel
11	50	206
12	18	86
13	76	23
14	26	37
15	69	13
16	96	8
17	91	12
18	13	24
19	19	19
20	5	30
21	53	43
22	7	31
23	5	642
24	3	317
25	136	2
26	833	7

Table 2: Real testbed – experiments to choose the local best channel.

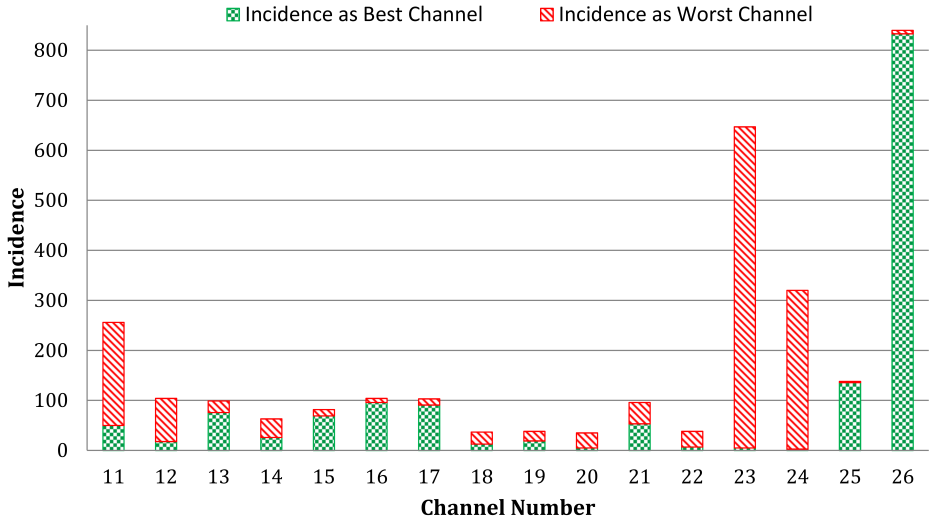

Figure 6: Real testbed – Local Best Channel Incidence at sink nodes.

that most of the time, channels 23 and 24 were identified the worst ones. The reason is that there is a wireless Access Point (AP) working on the channel 11 of IEEE 802.11g, positioned very near the sink node (1 m). Consequently, this AP

significantly affects the quality of the channels of IEEE 802.15.4 based WSNs. Similarly, the experimental results also showed that channels 25 and 26 were selected as the best with a very high incidence. This result is justified by the fact that, in the experimental testbed location, these channels were not interfered with other wireless networks. Other channels suffer interferences from the many Wi-Fi networks operating on the channels 1, 6 and 11, and their quality depends on the traffic rate of those networks. In fact, as shown in Figure 6, there are still some good channels that can be used for WSNs such as channels 16 and 17.

To evaluate the reliability of the selected best and worst channel, numerous experiments were done to measure their packet loss rate. For these experiments, the three highest incidence best channels (26, 25, and 16) and the three worst ones (23, 24, and 11) were selected. The test was done using two MAC protocols: X-MAC [33] and DynMAC. The objective of this test was not to compare the performance between these two protocols but to see how each performed on different quality channels. Because the X-MAC protocol is based in CSMA/CA, a testbed was set with only two nodes: one for sending packets and the other for collecting packets and calculating the packet loss rate. The reason for this simple testbed was to avoid the inner interference among the nodes in the WSN. The testbed for the latter case has the same topology described in Figure 3. The experiments to measure the packet loss of both X-MAC and DynMAC were repeated 20 times, using 1 hour for each channel. The result of these experiments is shown in Figure 7, where the bars indicate the average packet loss and the markers represent the 95 percentile.

It can be seen from Figure 7 that there is a significant difference in loss rate between the best and worst channels. By using a simple two node testbed with X-MAC, the packet loss rate of the worst channel was more than 6%, while using the best channel the rate dropped to less than 3%. For the DynMAC, because it is a TDMA protocol with acknowledgement and retransmission mechanisms, the loss rate was significantly reduced. However, there is still a noticeable difference between the best and the worst channels. For the worst channel, the loss rate could be up to more than 3% while that rate for the best one was about 0.25%. It could be concluded that, by selecting the right communication channel, a significant improvement of the WSN performance can be achieved.

5.2.2. Scanning and network coverage time

The second experiment done measured the scanning time of the nodes and the network coverage time. In the scanning time test, the time it took for the nodes located at different levels to detect the channel on which the sink was running

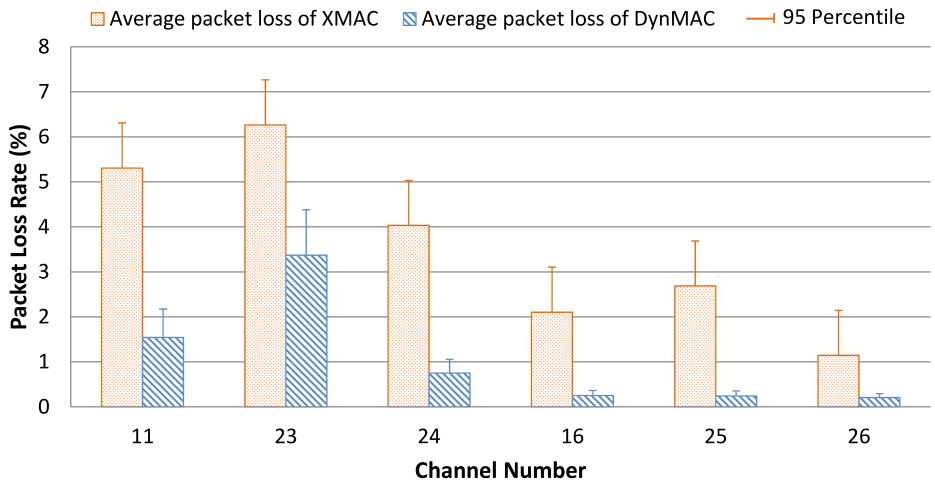

Figure 7: Real testbed – Loss Rate of Best vs Worst Channels.

was measured. For the network coverage time, the time it took for all nodes to successfully join the network was measured. In these tests, the time was measured from the moment when the sink node finished booting. The experimental results concerning the scanning time using simulation are shown in Table 3 and Figure 8.

Statistic	**Level 1 Scanning time (ms)**	**Level 1 Finish booting time (ms)**	**Level 2 Scanning time (ms)**	**Level 2 Finish booting time (ms)**	**Level 3 Scanning time (ms)**	**Level 3 Finish booting time (ms)**
Minimum	16	168	8	88	8	88
Q1	120	432	160	416	60	376
Median	424	832	392	728	336	668
Q3	720	1056	664	1076	658	954
Maximum	984	1352	1328	1720	2120	3496
Average	432.35	745.41	459.63	794.35	477.24	761.41

Table 3: Real testbed – scanning and booting time of nodes in different levels.

As shown in Table 3, nodes scan and detect the radio channel of the sink node very quickly. In most of the cases the nodes only take 1 super-frame (less than 1 second) to find out on which channel the sink is running. However, in some cases, it takes up to 3 super-frames (2120 ms) to scan the operation channel of the sink node, a situation that occurs mainly from nodes at a level far away from the sink.

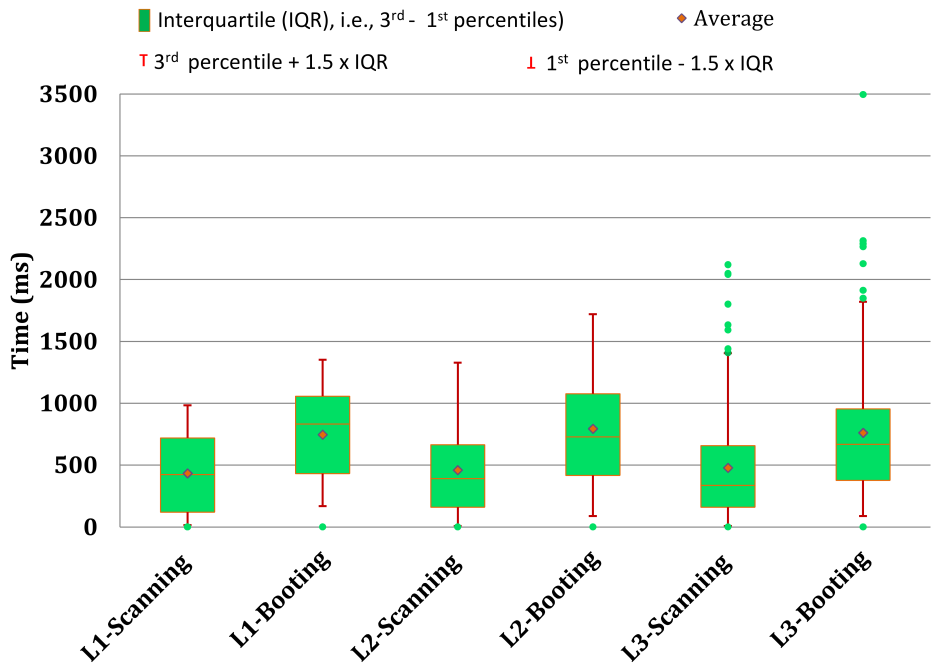

Figure 8: Real testbed – scanning and booting time average of nodes located at different levels.

This is reasonable because nodes at higher levels must wait for their parents to joint the network before they can detect them.

Similarly to the simulation results, after knowing the channel of the sink node, the nodes also need some time to finish booting. From the observation of the testbed, it can be concluded that the results obtained are similar to those obtained from simulation. However, in some cases, it takes a little longer for the node to finish booting. In the experiments results, there is a case where it took a node 1696 ms to finish booting after it detected the channel. As shown in Table 3, the coverage time of the network can be estimated to be less than 7 seconds: (1352ms maximum booting time at level 1 + 1720ms maximum booting time at level 2 + 3496ms maximum booting time at level 3 = 6568ms). Observing the results of the experiments on network coverage time, it is noticeable that most of the time it took about 3 seconds to cover the entire network. The maximum network coverage time was measured as 6040ms.

5.2.3. Dynamic Interference Detection

Besides the ability to select the best channel automatically, DynMAC is also able to deal with dynamic noise and interference during the sensor network's operation. The mechanism used by DynMAC to accomplish this function is described in Section 4.3. To prove that the proposed mechanism is applicable and workable solution to a real scenario, numerous experiments were done.

In the experiments, the duration for computing the latest PER was set to 5 minutes and the threshold was defined as 1%. It means that if the PER of the last 5 minutes was equal or greater than 1%, then the normal node would send an alert message to the sink. In addition, if the sink received 3 or more alert messages from the nodes it would start the best channel re-evaluation process.

In order to evaluate this functionality of DynMAC, during the normal operation of the WSN, some interference was introduced on its current communication channel. This was done by making 4 sensor nodes continuously broadcast on the working channel of the sensor network. The interference introduced by these 4 sensor nodes affected the PER of some nodes in the network. Consequently, after about 10-15 minutes, the sink started the best channel re-evaluation process. The reason for the delay is because a normal node usually has to wait at least one interval (e.g., 5 minutes) to calculate the PER. In addition, the sink has to receive at least 3 messages to start the re-evaluation process. In this case, there may be one or two normal nodes affected. Thus, the sink has to wait at least 2 intervals to start the process.

Similarly, we also used an IEEE 802.11g network with one AP and two laptops as interferers to test the capability to dynamically detect interference of DynMAC. The experiments showed that the DynMAC could detect this noise source and reevaluate the quality of the channel, switching to a better channel.

5.2.4. Recovery from Connection Loss

Another experiment to evaluate the ability of a node to recover from a connection loss. It is important to note that the connection loss, in this context, mainly concerns the case in which one or more normal nodes cannot receive any switching channel request message from the sink. The result of such a scenario is that all the sensor network nodes switch to another communication channel, with the exception of some specific nodes. Consequently, these latter nodes have to search for the channel used by the network.

In order to test this functionality, during the switching channel process a node was moved to a place in which it could not communicate with its parent. Then, after the sensor network successfully switched to new channel, that node was moved

to its original position. The experiments were done using two types of normal nodes: leaf and intermediate nodes. The experimental results showed that these nodes detected their parent very quickly and re-establish their communication. In most of the cases, it required a node approximately 3 seconds, with a maximum of less than 10 seconds, to resynchronize with its parent (Figure 9).

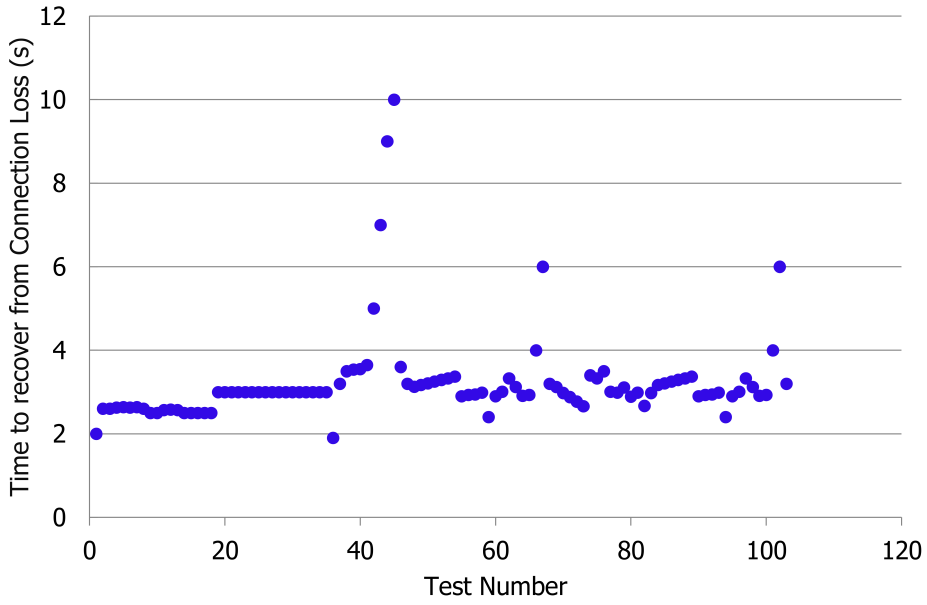

Figure 9: Real testbed – recovery from connection loss.

Another experiment to test this function is to use an IEEE 802.11g network with one AP and two laptops as the interferer between two sensor nodes (one parent and one child). During the switching channel process, two laptops continue broadcasting messages to corrupt the communication between sensor nodes. Of ten experiments, three of them successfully corrupt the communication between two sensor nodes, i.e., the child node cannot receive the switching message from its parent. Similar to previous experiments, after the network switched to the new channel, it required approximately 3 seconds for the child node to resynchronize with its parent. Similar tests were made in [34] that had similar results.

5.3. Comparison to other protocols

The experiment setup is similar to that used the GINSENG project [35]. For the performance evaluation of DynMAC, we compared its packet loss rate and

delay with those of LMAC [36] and MC-LMAC [26] protocols. These two protocols are TDMA based MAC protocols for WSNs. To evaluate the performance of LMAC and MC-LMAC, a simulation using OMNeT++ [37] was set up, using the network topology depicted in Figure 3. The distance between nodes is 5m and the transmission range of each node is around 25m. The transmission power was set to 0dBm. With MC-LMAC the number of channels was varied from 3 to 7. In these simulations all nodes transmit the packets to the sink and each node generates a packet every 5 seconds. To create a similar environment to the real testbed, three nodes operating respectively in the IEEE 802.11g channels 1, 6, and 11 were introduced in the network. For each protocol, 100 simulations were run. In each simulation, a node generated 1000 packets.

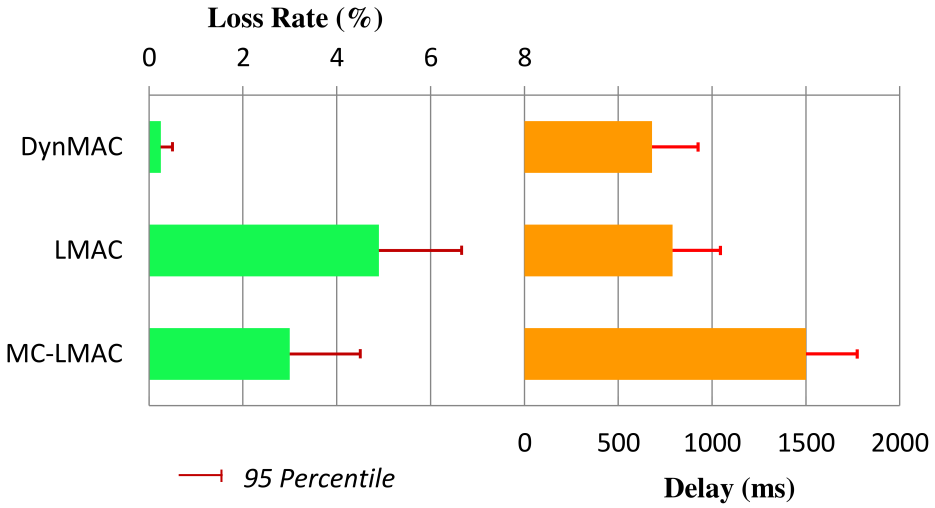

Figure 10: Comparison of loss rate and delay between protocols

The results of these experiments are shown in Figure 10. With a network similar to the one depicted in Figure 3, the simulation results showed that the packet loss rate of LMAC was of 4.9% and that of MC-LMAC was 3%, with more than 5 channels. For end-to-end delay, i.e., the time it takes a node to send data to the sink, the simulation results showed that LMAC had an average delay of 790 ms. The delay of MC-LMAC was higher than the other two (1500ms). Regarding DynMAC, as shown in previous section, with the best identified channel, it is possible to achieve a packet loss rate of 0.24%. In addition, the experimental results showed that the average delay of leaf nodes was around 680 ms and for

the nodes near the sink was around 430 ms. An important point to note is that the delay of DynMAC can be guaranteed and, in most cases, it is less than the duration of one super-frame.

5.4. Energy consumption issues in DynMAC

Energy consumption is one of the main concerns in WSN, and a trade-off between functionalities and the energy wasted to implement them, must always be studied. In this paper no specific tests on energy issues were done, as DynMAC relies on the basis of the GinMAC protocol, which limits by design the energy wasted. The primary goal of GinMAC is to improve reliability and message delivery delay. However, it does not neglect energy efficiency. Because a node in a network has knowledge of its allocated time slots, it will sleep during the time slots of other nodes. To the topology depicted in Figure 3 corresponds a super-frame of 100 time-slots, resulting in a duty cycle of 2% for each leaf node. Similarly, the maximum duty cycle of nodes at Level 2 is of 20% and that of nodes at Level 1 is of 42%. In addition, the super-frame can include extra slots in which all nodes in network will go to sleep to improve the network lifetime.

During the DynMAC operation there is one situation where an evaluation of the trade-off between wireless medium scanning and packet loss rate is more relevant, in the Periodic re-evaluation of the best channel. Regarding the other two situations, the initial set-up phase is only done once and the recovery from a lost link is a trade-off between not having any communication and the effort wasted in reconnecting. However, even in the periodic re-evaluation of the best channel, the calculations are resumed to the PER and only if it exceeds a previous defined threshold, a message is generated. In the worst case, each node sends a message every 5 minutes (that can be configurable) for evaluation by the sink (using also information from other nodes), and only then a re-selection process may take place (Section 4.3).

6. Conclusions and Future Work

This paper proposes a new MAC protocol, DynMAC, with mechanisms for dealing with the coexistence problem in WSNs. The mechanisms employed in this protocol are based on the cognitive radio functions such as spectrum sensing, analysis, decision, and sharing. The DynMAC was evaluated using both simulation and a real testbed. Experiments were done to test different functionalities of the proposed protocol. Specifically, tests were made to evaluate the ability of the DynMAC protocol for sensing and deciding the best channel, both by using

the sink alone and by using the cooperation between the sink and normal nodes. The results from these experiments correctly reflected the interference condition of the tested environment. In addition, the time needed to form the network was also evaluated. In the case where all the nodes booted at the same time, it took less than 7 seconds for all nodes to scan and join the network. Moreover, the normal nodes could very quickly detect the communication channel of the sink (less than 1 second). Furthermore, the dynamic noise and interference detection, and lost communication recovery mechanisms, were successfully evaluated using the real testbed.

Results from experiments indicate that DynMAC can solve the configuration problems of the WSNs in dynamic, noisy and interference environments, while maintaining the application requirements under critical scenarios. It is a resilient protocol that enables WSNs to coexist with other wireless networks and devices without a central coordinator.

As a future work, we will apply the mechanisms employed in DynMAC on a CSMA based MAC protocol for WSNs, such as X-MAC.

Acknowledgement

This work was supported by CAPES in the post-doctoral (BEX 6668/10-0) and had the financial support of agency CNPq (process 306869/2012-8). The work presented also was partially financed by the iCIS project (grant CENTRO-07-ST24-FEDER-002003) and by the IST FP7 0384239 GINSENG.

References

[1] M. A. McHenry, P. A. Tenhula, D. McCloskey, D. A. Roberson, C. S. Hood, Chicago spectrum occupancy measurements & analysis and a long-term studies proposal, in: Proceedings of the first international Workshop on Technology and policy for accessing spectrum (TAPAS), 2006, 2006, pp. 1–12.

[2] G. Zhou, J. A. Stankovic, S. H. Son, Crowded spectrum in wireless sensor networks, Workshop on Embedded Networked Sensors (2006).

[3] P. Ferrari, A. Flammini, D. Marioli, E. Sisinni, A. Taroni, Coexistence of wireless sensor networks in factory automation scenarios, Sensors Transducers Journal 90 (4) (2008) 48–60.

[4] IEEE Std. 802.15.2, Recommended Practice for Telecommunications and Information exchange between systems. Local and metropolitan area networks Specific Requirements – Part 15.2: Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Band (2003).

[5] T. Baykas, J. Wang, S. Filin, Standardization activities of IEEE 802.19 Task Group 1: wireless coexistence in the TV White Space (2011).

[6] O. B. Akan, O. B. Karli, O. Ergul, Cognitive radio sensor networks, Network Magazine of Global Internetworking. 23 (2009) 34–40. doi:http://dx.doi.org/10.1109/MNET.2009.5191144.

[7] FP7, Ginseng project, European Commission FP7, `http://www.ict-ginseng.eu/` (2010).

[8] P. Suriyachai, J. Brown, U. Roedig, Time-critical data delivery in wireless sensor networks, in: R. Rajaraman, T. Moscibroda, A. Dunkels, A. Scaglione (Eds.), Distributed Computing in Sensor Systems, Vol. 6131 of Lecture Notes in Computer Science, Springer Berlin – Heidelberg, 2010, pp. 216–229. doi:10.1007/978-3-642-13651-1_16.

[9] T.-D. Tran, R. Silva, D. Nunes, J. Silva, Characteristics of Channels of IEEE 802.15.4 Compliant Sensor Networks, Wireless Personal Communications (2011) 1–16. doi:10.1007/s11277-011-0395-3.

[10] T. Rappaport, Wireless Communications: Principles and Practice, 2nd Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[11] IEEE Std. 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) (2006).

[12] B. Dezfouli, M. Radi, S. A. Razak, T. Hwee-Pink, K. A. Bakar, Modeling low-power wireless communications, Journal of Network and Computer Applications. doi:10.1016/j.jnca.2014.02.009.

[13] W. Yuan, X. Wang, J.-P. Linnartz, A coexistence model of ieee 802.15.4 and ieee 802.11b/g, in: Communications and Vehicular Technology in the Benelux, 2007 14th IEEE Symposium on, 2007, pp. 1–5. doi:10.1109/SCVT.2007.4436237.

[14] D. Cavalcanti, S. Das, J. Wang, K. Challapali, Cognitive radio based wireless sensor networks, Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on (2008). doi:10.1109/ICCCN.2008.ECP.100.

[15] S. Nethi, J. Nieminen, R. Jantti, Exploitation of multi-channel communications in industrial wireless sensor applications: Avoiding interference and enabling coexistence, in: Wireless Communications and Networking Conference (WCNC), 2011 IEEE, 2011, pp. 345–350. doi:10.1109/WCNC.2011.5779155.

[16] S. Pollin, I. Tan, B. Hodge, C. Chun, A. Bahai, Harmful coexistence between 802.15.4 and 802.11: A measurement-based study, in: Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on, 2008, pp. 1–6. doi:10.1109/CROWNCOM.2008.4562460.

[17] E. Toscano, L. Lo Bello, Cross-channel interference in ieee 802.15.4 networks, in: Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on, 2008, pp. 139–148. doi:10.1109/WFCS.2008.4638731.

[18] L. Lo Bello, E. Toscano, Coexistence issues of multiple co-located ieee 802.15.4/zigbee networks running on adjacent radio channels in industrial environments, Industrial Informatics, IEEE Transactions on 5 (2) (2009) 157–167. doi:10.1109/TII.2009.2018541.

[19] P. Ferrari, A. Flammini, D. Marioli, E. Sisinni, A. Taroni, Synchronized wireless sensor networks for coexistence, in: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, 2008, pp. 656–663. doi:10.1109/ETFA.2008.4638466.

[20] V. Brik, E. Rozner, S. Banerjee, DSAP: a protocol for coordinated spectrum access, in IEEE DySPAN (November 2005). doi:10.1109/DYSPAN.2005.1542680.

[21] FCC, ET Docket No 03-322 notice of proposed rule making and order, `http://www.cs.ucdavis.edu/~liu/289I/Material/FCC-03-322A1.pdf` (2003).

[22] A. M. Wyglinski, M. Nekovee, Y. T. Hou, Cognitive Radio Communications and Networks: Principles and Practice, Academic Press, 2009.

[23] J. Marinho, E. Monteiro, in: Cognitive radio: survey on communication protocols, spectrum decision issues, and future research directions, Kluwer Academic Publishers, Hingham, MA, USA, 2012, pp. 147–164. doi:10.1007/s11276-011-0392-1.

[24] L. H. A. Correia, E. E. Oliveira, D. F. Macedo, P. M. Moura, A. A. Loureiro, J. S. Silva, A framework for cognitive radio wireless sensor networks, IEEE Symposium on Computers and Communications (ISCC), 2012 (July 2012). doi:10.1109/ISCC.2012.6249364.

[25] L. Stabellini, M. M. Parhizkar, Experimental comparison of frequency hopping techniques for 802.15.4-based sensor networks, UBICOMM 2010: The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (2010).

[26] O. D. Incel, L. van Hoesel, P. Jansen, P. Havinga, MC-LMAC: A Multi-channel MAC Protocol for Wireless Sensor Networks, Ad Hoc Netw. 9 (1) (2011) 73–94. doi:10.1016/j.adhoc.2010.05.003.

[27] W.-Z. Song, R. Huang, B. Shirazi, R. LaHusen, Treemac: Localized tdma mac protocol for real-time high-data-rate sensor networks, in: Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on, 2009, pp. 1–10. doi:10.1109/PERCOM.2009.4912757.

[28] H. Lee, A. Cerpa, P. Levis, Improving wireless simulation through noise modeling, in: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN'07, ACM, New York, NY, USA, 2007, pp. 21–30. doi:10.1145/1236360.1236364.

[29] Chipcon CC2420 Datasheet Texas Instruments, `http://focus.ti.com/lit/ds/symlink/cc2420.pdf` (2007).

[30] K. Srinivasan, P. Dutta, A. Tavakoli, P. Levis, An empirical study of low-power wireless, ACM Trans. Sen. Netw. 6 (2) (2010) 16:1–16:49. doi:10.1145/1689239.1689246.

[31] A. Dunkels, B. Gronvall, T. Voigt, Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors, in: Proceedings of the 29th

Annual IEEE International Conference on Local Computer Networks, Vol. 0 of LCN'04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 455–462. doi:10.1109/LCN.2004.38.

[32] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, P. J. Marrón, Cooja/mspsim: interoperability testing for wireless sensor networks, in: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools '09, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2009, pp. 27:1–27:7. doi:10.4108/ICST.SIMUTOOLS2009.5637.

[33] M. Buettner, G. V. Yee, E. Anderson, R. Han, X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks, in: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, ACM, New York, NY, USA, 2006, pp. 307–320. doi:10.1145/1182807.1182838.

[34] L. Angrisani, M. Bertocco, D. Fortin, A. Sona, Experimental study of coexistence issues between ieee 802.11b and ieee 802.15.4 wireless networks, Instrumentation and Measurement, IEEE Transactions on 57 (8) (2008) 1514–1523. doi:10.1109/TIM.2008.925346.

[35] T. O'donovan, J. Brown, F. Büsching, A. Cardoso, J. Cecílio, J. D. Ó, P. Furtado, P. Gil, A. Jugel, W.-B. Pöttner, U. Roedig, J. S. Silva, R. Silva, C. J. Sreenan, V. Vassiliou, T. Voigt, L. Wolf, Z. Zinonos, The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences, ACM Trans. Sen. Netw. 10 (1) (2013) 4:1–4:40. doi:10.1145/2529975.

[36] L. van Hoesel, P. Havinga, A lightweight medium access protocol (LMAC) for wireless sensor networks, in: Proceedings of the First International Conference on Networked Sensing Systems, 2004.

[37] Omnet++ discrete event simulation environment, available at `http://omnetpp.org/`.