Cognitive Radio Simulation based on OMNeT++/MiXiM

José Marinho and Edmundo Monteiro

***Abstract*—Cognitive Radio (CR) is a recent paradigm which aims at increasing the efficiency of spectrum usage. Basically, this goal is achieved by opening the entire radio spectrum to unlicensed accesses as long as any harmful interference to licensed users is avoided. One of the possible approaches consists in enabling the unlicensed users to opportunistically access vacant spectrum bands (i.e., through dynamic spectrum access). This is the designated CR overlay approach. Currently, as we are engaged in research activities on overlay-based CR, we developed a generic CR simulation framework. Its main goal is to reduce the effort which will be necessary to develop the simulation models for our proposals and, therefore, simplify their validation and evaluation. OMNeT++/MiXiM was chosen as the developing platform, mainly due to its open source nature, its well-organized modular architecture, the existing documentation, and the provided IDE (Integrated Development Environment). This paper aims at describing, through a tutorial approach, the developed CR framework in terms of its most relevant characteristics, as well as a first experience in using it. Concretely, two CR medium access control protocols were successfully modeled. From this first experience, it can also be concluded that the presented CR framework definitively meets the objectives it was developed for.**

***Index Terms*—Cognitive Radio; Medium Access Control; OMNeT++; MiXiM; Simulation**

I. Introduction

Currently, the radio spectrum is ruled by inflexible policies which divide it in licensed and unlicensed frequency bands. The unlicensed spectrum is becoming more crowded. This is especially true concerning the ISM (Industrial, Scientific and Medical) bands, which are shared by a large number of technologies such as high speed wireless local area networks and cordless phones, in densely populated areas. On the other side, several measurements taken over the world show that licensed frequency bands are often underutilized, creating temporally available spectrum opportunities that are variable in time and space [1]. This obviously results in what is considered an inefficient usage of the spectrum. The Cognitive Radio (CR) area has emerged as a means to address this spectrum inefficiency. According to the designated CR overlay approach, unlicensed users (i.e., secondary users) are allowed to opportunistically use the referred spectrum opportunities, often designated as spectrum holes or white spaces, in order to increase their performance, as long as they preserve primary users (i.e., licensed users) from any harmful interference. Therefore, the operating spectrum band and other transmission parameters (e.g., transmission power and modulation scheme) are dynamically and intelligently chosen by secondary users (SUs). This makes CR highly dependent on the availability of a kind of radio that can be dynamically reconfigured by software (i.e., software defined radios). Additionally, it must be noted that the issues of CR can span all the layers of the communication protocol stack, but its basics are mostly limited to the physical (PHY) and medium access control (MAC) layers. CR has been and still is an active research area. Besides, due to its highly interdisciplinary nature, CR research activities have been concerned with distinct engineering and computer science disciplines such as signal processing, communication protocols, and machine learning [2].

We are currently engaged in research activities concerning optimized and practical overlay-based CR MAC protocols. Most existing CR MAC proposals have been validated and evaluated through simulation. This common practice is especially due to time and other resource restrictions. Therefore, we have also decided to follow this approach, and chose OMNET++/MiXiM [3][4] as our developing platform. OMNET++ [3] (Objective Modular Network Testbed in C++) is an object-oriented modular discrete network simulator. MiXiM [4] (Mixed Simulator) is a simulation framework for wireless and mobile networks which was developed on top of OMNeT++. Several reasons dictated our selection. However, among the most relevant, we can refer the open source nature of the framework, its well-organized and modular architecture, the existing documentation, and the provided integrated development environment (IDE). The latter effectively simplifies debugging and tracing through powerful tools, especially when the modeled network environments are based on complex protocols and interactions.

Most existing CR MAC proposals are based on SUs which are equipped with two or more radios [5][6]. Usually, one of them is dedicated to a common control channel, which is used for signaling operations, and the others, which are dynamically

Manuscript received November 9, 2011.

José Marinho is with the Engineering Institute of Coimbra, Polytechnic Institute of Coimbra, Coimbra, Portugal, and with the Centre for Informatics and Systems of the University of Coimbra (CISUC), Coimbra, Portugal (e-mail: fafe@isec.pt).

Edmundo Monteiro is with the Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal (e-mail: edmundo@dei.uc.pt).

reconfigurable, are used for data transmission. Therefore, in the context of our current research activity on CR MAC protocols, we assume that the architecture of the SUs follows the multi-radio model which is illustrated in Fig. 1. With this model, the MAC entity of each radio implements a spectrum access protocol which defines how the sending and receiving of data frames are processed. Consequently, distinct radios can follow different access strategies. Concerning the CR MAC protocol, it runs on top of those radios and concretely implements the CR access strategy. Negotiation, coordination, and cooperation among SUs are performed by the CR MAC entity, as well as possibly other usual MAC-level management operations (e.g., timeouts, frame retransmissions). If a dedicated common control channel is used, control/signaling frames are exchanged via the corresponding radio. The scanner module, which is dedicated to sensing the activity of primary users (PUs), is optional as sensing can also be performed by the reconfigurable radios. This is the case for most existing CR MAC proposals. Finally, the cognitive radio engine module is dedicated to learning and decision making. For instance, it decides which channel should be selected as a candidate for transmission.

At the best of our knowledge, there was no known support for the referred model in OMNET++/MiXiM or any other network simulation framework. Besides, in the context of our research activity, the availability of a generic and modular implementation of the model in Fig. 1 would definitively contribute for simplifying the implementation of our proposals and, therefore, reducing time to validate and evaluate them. Basically, with such CR simulation framework, CR MAC proposals could be implemented as extensions (i.e., subclasses) of the provided CR MAC module, and proposals about learning and decision making as extensions of the CR Engine module. This is what we did and describe in this paper. It must be noted that the simulation framework which was developed also enables modeling the activity of PUs.

The remainder of the paper is organized as follows. Section II provides a brief introduction to relevant aspects of OMNeT++ and MiXiM. Then, section III describes the CR simulation framework which was developed based on the multi-radio model of Fig. 1. Section IV describes the implementation of two concrete CR MAC protocols and concludes about the effectiveness of the provided framework. Final conclusions are drawn in section V.

II. BACKGROUND

This section aims at providing relevant background on OMNeT++/MiXiM in order for the readers to fully understand the description of the CR framework which is provided in section III.

A. Architecture of simulation models in OMNeT++

In terms of the architecture of simulation models, OMNeT++ follows a modular approach in which modules are intended to be really reusable. It consists of compound and simple modules which are hierarchically organized (see Fig. 2). Compound modules include several modules, which in turn can include other modules, and so on. There is no limit in the number of hierarchy levels. Simple modules are at the bottom of the hierarchy of modules and, therefore, do not include any further modules. In fact, they are the active components of OMNeT++, being their behaviors defined through the C++ programming language. Fig. 2 illustrates a possible module hierarchy for a network with two wireless nodes. It is inspired on the MiXiM base node model which includes a base network module and no specific transport module. The structure and characteristics of the modules are defined through the NED language (see Fig. 3). Modules can have an arbitrary number of gates (see Fig. 2 and Fig. 3). Besides, connections can be established between the gates of modules which are sub-modules of the same module (i.e., which are in the same hierarchical level) or between a sub-module and the module

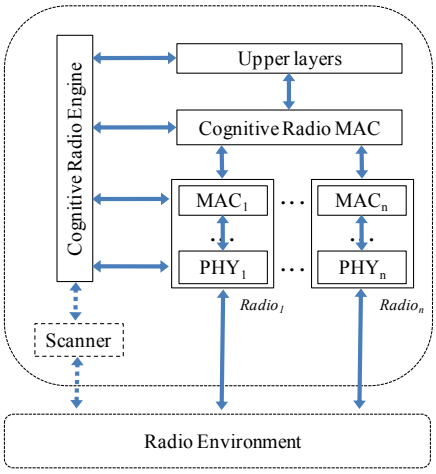

Fig. 1. A secondary user architecture based on a multi-radio model.

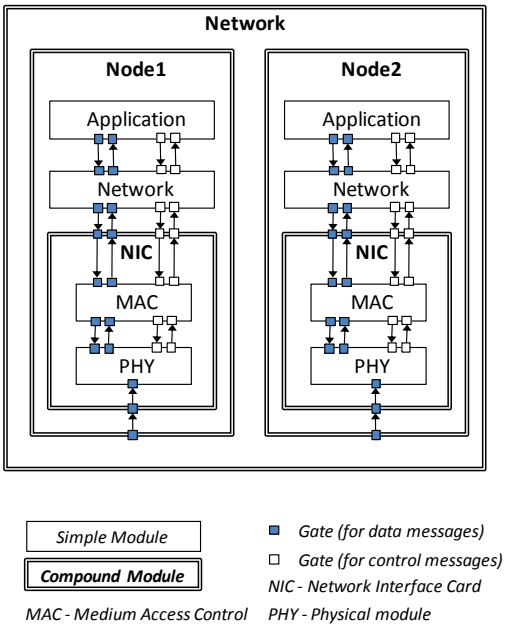

Fig. 2. Module hierarchy for a network with two wireless devices.

```
module Nic80211 like INic
{
    parameters:
         string connectionManagerName = default("");
         @display("i=block/ifcard");

    gates:
         input upperGateIn; // to upper layers
         output upperGateOut; // from upper layers
         output upperControlOut; // control information
         input upperControlIn; // control information
         input radioIn; // radioIn gate for sendDirect

    submodules:
         mac: Mac80211 {
              @display("p=96,87;i=block/layer");
         }

         phy: PhyLayer {
               @display("p=96,158;i=block/process;is=n);
         }

    connections:
         mac.upperGateOut --> upperGateOut;
         mac.upperGateIn <-- upperGateIn;
         mac.upperControlOut --> upperControlOut;
         mac.upperControlIn <-- upperControlIn;
         phy.upperGateOut --> mac.lowerGateIn;
         phy.upperGateIn <-- mac.lowerGateOut;
         phy.upperControlOut --> mac.lowerControlIn;
         phy.upperControlIn <-- mac.lowerControlOut;

        radioIn --> phy.radioIn;
    }
```

Fig. 3. Partial example of a module definition (compound module).

which includes it (see Fig. 3). These connections are used by the modules to exchange messages between them. In Fig. 2, most connections between modules are achieved through two pairs of gates. One of the pairs is used for exchanging data messages and the other for exchanging control messages (i.e., out-of-band signaling). The parameters which characterize the modules are defined in NED files. Their values can be specified in the designated *omnetpp.ini* file (see Fig. 4), by default in NED files (see the *connectionMangerName* parameter in Fig. 3), or prompted at runtime.

B. Physical models in MiXiM

The model of a wireless device always includes a physical module which actuates according to some defined physical models. In MiXiM, these models are specified through a XML (eXtensible Markup Language) file (see Fig. 4 and Fig. 5) and their implementations are achieved through the C++ programming language. These models include one or more analog models and a designated decider. The analog models specify the way the attenuation which is suffered by a signal is calculated. If more than one analog model is defined, the final attenuation is the combined effect of all of them. The decider aims at deciding if an incoming packet is to be received or not, i.e., passed to the upper layer (usually the MAC sub-layer). For instance, it can take the decision based on a SNR (Signal to Noise Ratio) threshold and on a defined bit error rate. MiXiM already includes a few analog models and deciders. However, new ones can be defined. This requires extending

```
         **.mac.queueLength = 14
         **.mac.bitrate = 11E+6bps # in bits/second

         **.phy.useThermalNoise = true
         **.phy.analogueModels = xmldoc("config.xml")
         **.phy.decider = xmldoc("config.xml")
```

Fig. 4. Partial example of a configuration file (omnetpp.ini).

```
<AnalogueModel type="SimplePathlossModel">

   <parameter name="alpha" type="double" value="3.0"/>
   <parameter name="carrierFrequency" type="double"
                                        value="2.412e+9"/>
</AnalogueModel>

<Decider type="Decider80211">

   <!-- SNR threshold [NOT dB]-->
   <parameter name="threshold" type="double"
                               value="0.12589254117942"/>

   <!-- The center frequency on which the phy listens-->
   <parameter name="centerFrequency" type="double"
                                       value="2.412e9"/>
</Decider>
```

Fig. 5. Specification of the physical models.

the original PHY module as it does not know how to use them a priori.

C. Programming paradigm in OMNeT++

The simulation library which is provided by OMNeT++ is based on the C++ language. Therefore, a significant amount of the effort which is necessary to develop new simulation models consists in programming the simple modules in C++. Despite the open source nature of OMNeT++ and MiXiM, modifications in the source code must be avoided, and the known good practices, in terms of object oriented programming, must be followed (e.g., code reuse, sub-classing, overriding of methods).

In OMNeT++, the programming model is event driven, i.e., the simple modules actuate upon the reception of messages. Basically, the C++ source code of any simple module includes a message receiving function (i.e., *handleMessage(cMessage* msg)*) which is asynchronously invoked any time a message is received. A module can receive messages through connected input gates (i.e., messages which are sent by its connected modules). It can also receive self-messages, i.e., messages which are sent by itself and scheduled to be received at a specified simulation time. Self-messages are of great importance in OMNeT++ (e.g., they are used to implement timers and to simulate time durations).

It must also be noted that, in OMNeT++, simulation time is different from real time and its update is only determined by the messages which are scheduled. In general terms, when a message is sent/scheduled, it is tagged with an arrival time (e.g., the current simulation time plus a duration time). Then, when the scheduled message with the most recent arrival time is processed, simulation time is updated accordingly. For instance, to simulate the time which is necessary for a given

computation task to complete and, therefore, update the simulation time accordingly, a self-message must be used. Therefore, in OMNeT++, simulations terminate when there are no more scheduled messages, as no more events will occur.

III. CR Simulation Framework

This section presents the main characteristics of the CR simulation framework which has motivated the writing of this paper. As referred previously, this framework aims at supporting and simplifying the development of simulation models for overlay-based CR proposals, in the context of our research activity.

As illustrated in Fig. 2, a typical wireless device includes a single network interface card (NIC), i.e. a single radio. In MiXiM, a radio is modeled as an OMNeT++ compound module which, in turn, includes two simple modules. These are the PHY and MAC modules. Therefore, in order to preserve the same structuring approach, all the radios and the CR MAC entity of the multi-radio model (see Fig.1) have been included in a single compound module. This is what we designate as the CR NIC module (see Fig. 6). Concerning the SUs, they are also modeled as compound modules which include the referred CR NIC module, as well as the CR engine and the CR scanner simple modules (see Fig. 2 and Fig. 6). It must be noted that in Fig. 2 and Fig. 6 three MiXiM utility simple modules, which are not relevant in the context of the current discussion, have been omitted for simplicity. The gates and the connections have also been omitted in Fig. 6 for simplicity. Besides the modeling of SUs, the developed framework also includes a simple module which aims at modeling the activity of a primary user. Additional details about the modules of the CR framework are presented in the next subsections.

A. CR NIC module

As referred previously, the CR NIC compound module includes a simple module which models the CR MAC entity, a compound module which models a non-reconfigurable radio, and an array of compound modules which model the reconfigurable radios (see Fig. 6). The two types of radio modules are connected to the CR MAC module through input and output control gates, which are used to exchange control messages, and through input and output data gates, which are used to send and receive data frames. The non-reconfigurable radio is intended to be used as a common control channel. However, any CR MAC module is free to ignore it or use it for another purpose. It depends on the specificities of the implemented CR MAC protocol. Concerning the reconfigurable radios, they aim at enabling the dynamic access to the spectrum, and their number is defined as a parameter in the configuration file (i.e., *omnetpp.ini*). The non-reconfigurable and reconfigurable radio modules include their corresponding MAC and PHY simple modules.

The MAC module of the non-reconfigurable radio is based on the MiXiM implementation of IEEE 802.11, which accesses the spectrum through a random approach. This is the currently most used protocol for wireless networks. It is

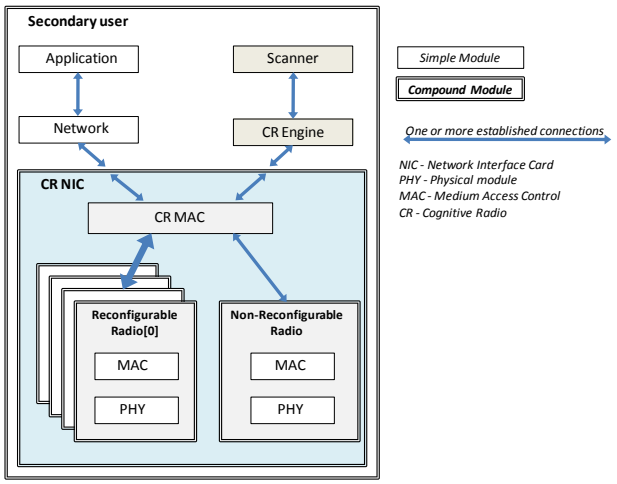

Fig. 6. SU compound module structure.

expectable that several management issues, which are usually addressed by IEEE 802.11 or any other legacy wireless MAC protocol, be performed by the CR MAC protocol. Actually, the latter is the main MAC entity in a SU. Therefore, the provided IEEE 802.11-based MAC module can be configured by the CR MAC module in order to avoid the retransmission of unicast frames or to access the spectrum without any contention procedure. To inhibit the RTS/CTS (Request To Send/Clear To Send) handshake of IEEE 802.11, a high enough RTS/CTS threshold (i.e., equal or above the maximum transfer unit) must be specified in the configuration file. The IEEE 802.11-based MAC module also includes additional control messages, such as *TX_STARTED* (transmission started) and *TX_OVER* (transmission over), which are sent to the CR MAC module. This enables the CR MAC protocol to use the provided additional information for its internal operations, whenever and if it is necessary. The provided MAC module also supports the optional definition of a maximum coverage area in the configuration file (e.g., ***.mac.maxDistance = 100m*). In this case, the transmission power is determined on startup based on the analog model which is used by default in MiXiM and in our framework also (i.e., the simple path loss model).

Concerning the MAC module of the reconfigurable radios, it is an extension of the module which was described in the previous paragraph for the non-reconfigurable radio. Basically, it enables the operating center frequency to be dynamically changed. It must be noted that this operation is more complex than just changing an attribute in the MAC module. For instance, a center frequency switch takes some amount of time and incoming signals must be ignored meanwhile, it can only start after any ongoing request and all the pending requests are processed, and so on. Therefore, the CR MAC entity only considers that a frequency switch request has been completed after it asynchronously receives a control message from the reconfigurable MAC. When a maximum coverage area is defined in the configuration file, the transmission power is also adjusted during a frequency switch operation. This is due to a strong dependency between the coverage area and the operating frequency (i.e., the higher the frequency the shorter

the coverage area and vice-versa). Finally, it must be noted that other MAC modules which are not based on the ones we provided in the framework can be developped as long as they preserve some key methods and control messages.

Concerning the PHY modules, we first had to develop a subclass of the MiXiM decider for IEEE 802.11 in order to enable it to operate correctely when distinct transmissions take place simultaneously on different frequency bands. Basically a few minor and hard to locate/track implementation incorrections were fixed. Then, the referred fixed IEEE 802.11 decider was extended in order to support dynamic switches of the operating center frequency and transmission power. The simple path loss analog model of MiXiM was also extended for the same reason. Finally, a subclass of the MiXiM *PhyLayer* class was developed in order know how to use the provided reconfigurable analog model and decider.

B. CR MAC module

The provided base CR MAC module only forwards messages between the data gate of the upper network module and the data gate of the first reconfigurable radio, and between the control gates of the same modules. In fact, it does not aim at providing a useful CR MAC protocol. Its main objective is to be extended and, therefore, it essentially provides useful initializations, attributes and base methods which aim at simplifying the development of subclasses (i.e., concrete CR MAC proposals). As expected, subclasses must override some of the base methods (e.g., *initialize*, *handleUpperMsg*, *handleSwitchFreqOver*), besides defining new ones. The provided base CR MAC module can also be configured to fragment packets with lengths over a specified maximum transfer unit size (e.g., ***.crMac.doFragmentation = true and **.crMac.mtu = 10000bit* in the configuration file).

C. CR engine module

The framework includes a base CR engine which is implemented as a simple module in each host. The main function of this module is to enable the CR MAC protocol to select a channel. It can be configured, through the configuration file, to return the first channel or a random channel from a set of channels which are considered available (e.g., ***.crengine.useRandomChannelSelection = true*), and to try to preserve the previously selected channel. The provided base CR engine also includes a method which enables the CR MAC module to report its experience in accessing the selected channels (i.e., successful or unsuccessful access). The current implementation of this method performs no operations upon the reception of an access report. More advanced selection schemes can be implemented by overriding it (e.g., in order to collect and maintain statistics about past history), as well as other methods which are related to channel selection. Finally, it can also be noted that the provided CR Engine can be configured in order to have a perfect knowledge about which channels are sensed free of PU activity at a given moment. This is not a realistic assumption, but might be useful for some simulation experiments (e.g., for establishing a comparison base).

D. Scanner module

The base scanner module in Fig. 6 is also implemented as a simple module which is connected to the CR Engine, and has a dual role in the context of the simulation framework. Firstly, it can act as the scanner entity of Fig. 2. In this case, scanning requests are made by the CR Engine through the sending of messages which identify the targeted frequency band. Then, when scanning is over, results are asynchronously sent back through messages. The Base CR engine keeps the conclusions of these sensing requests in an internal data structure. Therefore, CR engines which are based on extensions of the base CR engine can take advantage of these facilities (i.e., scanning requests and the internal data structure). Time to scan a channel is defined in the configuration file. The scanner module can also be extended in order to model less ideal scanners (e.g., with missed detection and false alarm probabilities).

The base scanner module is also a utility module. It keeps track of primary user activity changes (i.e., active to inactive, and vice-versa). If the SU it belongs to is in the interference area of the PU which has changed its activity state, an update message is sent to the corresponding CR engine. The interference area is determined based on the coverage areas of the SUs and PUs, which are modeled as perfect circles, and on their positions in the simulation playground. The radiuses of the coverage areas of the provided base PUs are fixed and defined in the configuration file (see next subsection). In terms of the SUs, the coverage areas are calculated based on the transmission power and on the analog models in usage (in the current implementation, only the simple path model is supported). The base scanner module can naturally be extended in order to consider other propagation models. The update messages which are sent include the activity status of the corresponding PU, its frequency band, and if the SU is in its coverage range, i.e., if it can detect it through local sensing. This information is primarily intended to be used by the CR engine to collect statistics about interferences to PUs and missed detections. This enables the user of the CR framework to analyze the effectiveness of its CR proposals in terms of PU protection. In this context, it must be referred that the CR MAC module also informs the base CR Engine module any time it decides to access a given channel. This enables to track interferences to PUs. Additionally, any time the base CR engine is notified by the scanner that a PU activity has started in the interference area of the SU it belongs to, it also notifies the CR MAC module. Then, if there is any ongoing interfering transmission, the CR engine is notified back. This enables the collecting of statistics about interferences to PUs which appear after transmission was started.

The information about PU activity which is provided by the base scanner module is also used by the CR engine for channel selection when it is configured to assume a perfect knowledge about which channels are sensed free of PU activities at a

given moment (see the discussion at the final of the previous subsection).

E. Modeling of primary users

In our CR framework, base primary users are implemented as simple modules, and their activity modeled as alternating idle and busy periods. In terms of configuration parameters, a PU is characterized by its position in the simulation framework, the radius of its coverage area, its operating center frequency, the bandwidth it uses, and the durations of the busy and idle periods (see Fig. 7). Probabilistic distributions can be used in OMNeT++ configuration files for defining the values of the parameters. For instance, in Fig. 7 the durations of the busy and idle periods are defined as being exponentially distributed, such as in the work of Issariyakul, Pillutla, and Krishnamurthy [9], and the positions of the PUs are randomly distributed.

IV. Validation of the CR framework

Up to know, we have already implemented two concrete CR MAC protocols on top of the presented CR framework. A specific paper, which has already been submitted, describes these protocols in detail. Therefore, this section, which aims at describing our first experience in using the presented CR simulation framework, does not go into much detail. Based on this first experience, we can also conclude about the effectiveness of the developed CR framework and, therefore, validate it.

A. Implementation of the CR MAC protocols

The first CR MAC protocol we implemented (CrMac1) is based on the work of Jia and Zhang [8] which follows a three-way handshake approach for channel negotiation and reservation. The second implemented CR MAC protocol (CrMac2) is an innovative proposal of ours. Basically, it extends the work of Jia and Zhang [8] with a cooperative sensing phase which is also based on a three-way handshake scheme. In this proposal, practicality, low-complexity, scalability, increase of performance which is delivered to SUs, and protection of PUs in multi-hop CR networks were our major concerns.

Both CR MAC protocols have been implemented as extensions of the base CR MAC module in the provided CR simulation framework. They also require the SUs to be equipped with two radios: one dedicated for operating the common control channel (CCC); and the other, which is dynamically reconfigurable, responsible for data transmission. It must be noted that local sensing is achieved by the reconfigurable radios and not by the scanner of Fig. 1. The developed CR MAC protocols are also strongly based on the overhearing of frames by neighbors. Therefore, a control CR MAC frame, which includes the intended CR MAC destination address, is broadcasted over the CCC in order to be delivered to the CR MAC modules of all the neighbors of the source SU.

```
**.puNode[0].posX = uniform(1,50)
**.puNode[0].posY = uniform(1,50)
**.puNode[0].posZ = uniform(1,50)
**.puNode[0].coverageArea = 500m
**.puNode[0].centerFrequency = 700e+6Hz
**.puNode[0].bandwidth = 110e+6Hz
**.puNode[0].activityArrivalRate = exponential(0.001s)
**.puNode[0].activityDuration = exponential(0.001s)
```

Fig. 7. Configuration of a primary user.

Due to the modular structure of the developed CR simulation framework and, obviously, OMNeT++, time to implement the CR MAC protocols was considerably reduced. In fact, only sub-classes of the CR MAC module had to be defined. Besides, as most low-level and base details are already handled by the provided base classes and modules, most of the development effort was focused on the specific issues of the targeted CR MAC protocols. It must also be noted that coding such protocols in OMNeT++ is a rather simple and comfortable experience when they are previously specified as finite state machines, such as we did. The main reason is due to the asynchronous and message-driven nature of the OMNeT++ programming model.

B. Evaluation results

As referred previously, details about the implementation and evaluation of the two CR MAC protocols are provided in a specific paper which has already been submitted for approval. Therefore, this subsection only provides a brief and partial description of the simulation scenarios which were used, as well as some of the obtained results.

For performance comparison, in terms of the throughput versus load metric, both protocols and legacy IEEE 802.11 (i.e., non-CR nodes) where evaluated on the same simulation scenarios. These scenarios included one or several sender-receiver pairs, which were randomly located in a two dimensional simulation area. The sender-receiver pairs were all located in the range of each other, which is the worst possible scenario in terms of contention for the spectrum. It was also assumed that four non-overlapped channels were available. The bit rate, both on the CCC and on the data channels, was set to 2 Mbit/s, and the MTU (Maximum Transfer Unit) to 18432 bits, such as in IEEE 802.11. Most of the PHY and MAC simulation parameters were the same which are set by default in MiXiM. The base CR engine was configured to follow the uniform random channel selection scheme. Time to adjust the radio parameters when switching to a new channel was set to 10^{-4} seconds, such as in the work of Jia and Zhang [8]. When PU activity is considered, the parameters of the exponential distributions of the idle and busy periods were set to 0.1 and 0.01 seconds, respectively [9]. As the CR MAC proposal includes an additional sensing phase when compare to the work of Jia and Zhang [8], distinct local sensing times were considered in order to evaluate their impact on the overall performance. The selected values were chosen based on the fact that the 802.22 Working Group [7] specifies that fast sensing should be under 1 ms per channel.

Due to the really modular architecture of OMNeT++ and of the presented CR simulation framework, selecting the CR MAC protocol to be used only requires changing the value of the *crMacType* parameter in the configuration file (e.g., "CrMac1" or "CrMac2"). Some of the obtained results are presented in Fig. 8 and Fig. 9. From these figures, it can be concluded that the implemented CR MAC protocols bring obvious benefits when compared to legacy IEEE 802.11. This is due to their CR capabilities which enable distinct sender-receiver pairs to transmit in parallel on distinct channels. Contention among SUs only occurs when there are less available channels than sender-receiver pairs and results in an expected degradation of performance (see Fig. 8 when PU activity is considered and Fig. 9).

We also evaluated the benefits which can result from the cooperative sensing scheme of our innovative CR MAC proposal (CrMac2), in terms of the protection of PUs (i.e., reduction in the number of missed PU detections). The simulation scenario which is illustrated in Fig. 10 and has a fixed topology was used. It pretends to be representative in terms of secondary and primary hidden nodes (i.e., nodes which can interfere with each other but cannot sense/detect each other). Every PU uses a predefined channel and the SUs only have access to this set of four channels (i.e., there are less channels than sender-receiver pairs). The activity of PUs was modeled as defined previously, and the intermediate 0.5 ms sensing time was considered. Results in Table I aim at quantifying the decrease in the global number of missed PU detections when the idle neighbors of CrMac2 also participate in sensing. Different load conditions were considered. It must be noted that, with the proposed sensing approach, the probability of a SU to be idle is lower for higher loads and, therefore, its chances to help neighbors in sensing also.

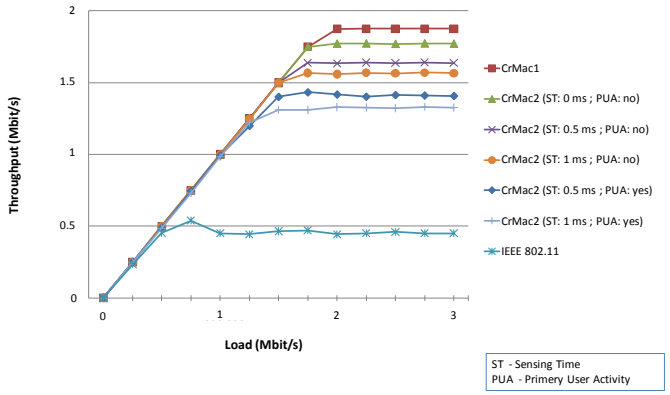

Fig. 8. Throughput vs. Load with four sender-receiver pairs.

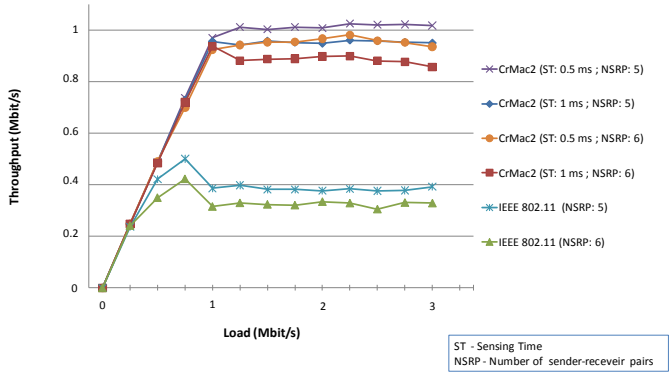

Fig. 9. Throughput vs. Load with five and six sender-receiver pairs.

V. Conclusions

This paper described a generic CR simulation framework which was developed in order to support and simplify the evaluation of dynamic spectrum access-based CR proposals. OMNeT++/MiXiM, which was also briefly introduced, was the chosen simulation network platform. The presented CR simulation framework has already been successfully used to implement and evaluate two CR MAC protocols. This first experience is briefly described in this paper, which is tutorial by nature, and enables us to conclude that the framework will effectively meet the objectives it was developed for.

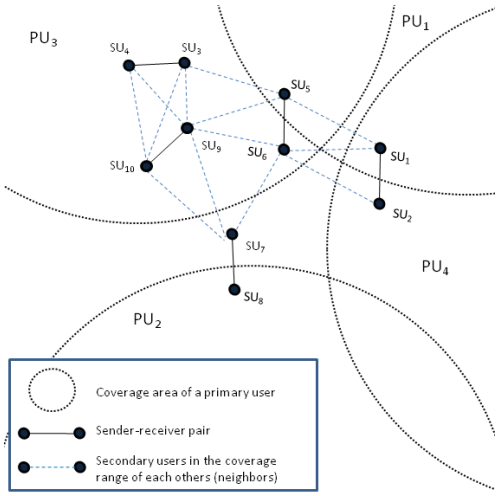

Fig. 10. A scenario with hidden nodes.

TABLE I
Contributions of Idle Neighbors in Cooperative Sensing (CrMac2)

Load	Variation in the number of missed PU detections[a]	
	Scenario in Fig. 10	Scenario in Fig. 10 with additional idle neighbors
1 Kbit/s	-39.3%	-53.6%
500 Kbit/s	-24.5%	-47,2%
2 Mbit/s	-4.4%	-43,7%

[a]When compared to the results which are obtained when the cooperation of idle neighbors is inhibited.

REFERENCES

[1] FCC Spectrum Policy Task Force, “Report of the spectrum efficiency working group,” Nov. 2002.

[2] M. Timmers, S. Pollin, A. Dejonghe, L. Van der Perre, and F. Catthoor, “A Distributed Multichannel MAC Protocol for Multihop Cognitive Radio Networks,” Vehicular Technology, IEEE Transactions on, vol. 59, 2010, pp. 446-459.

[3] “OMNeT++ Network Simulation Framework,” [Online]. Available: http://www.omnetpp.org/. [Acessed: Dec., 2010].

[4] “MiXiM,” [Online]. Available: http://mixim.sourceforge.net/. [Acessed: Dec., 2010].

[5] C. Cormio and K. Chowdhury, "A Survey on MAC Protocols for Cognitive Radio Networks," Ad Hoc Networks (Elsevier), Vol. 7, Issue 7, Sep. 2009, pp. 1315-1329.

[6] H. Wang, H. Qin, and L. Zhu, “A Survey on MAC Protocols for Opportunistic Spectrum Access in Cognitive Radio Networks,” Computer Science and Software Engineering, 2008 International Conference on, 2008, pp. 214-218.

[7] “IEEE 802.22 WRAN WG Website,” [Online]. Available: http://www.ieee802.org/22/. [Accessed: Dec. 2, 2009].

[8] J. Jia and Q. Zhang, “A Testbed Development Framework for Cognitive Radio Networks,” Communications, 2009. ICC '09. IEEE International Conference on, 2009, pp. 1-5.

[9] T. Issariyakul, L. Pillutla, and V. Krishnamurthy, “Tuning Radio resource in an Overlay Cognitive Radio Network for TCP: Greed Isn't Good,” IEEE Communications Magazine, Jul. 2009, pp. 57-63.