Cognitive Radio DAB MAC Protocol Performance using a CR Specific Simulator and Software Defined Radio

Coert J. Loubser
Dept. of Electrical and Electronic Engineering Science
University of Johannesburg
Johannesburg, South Africa
coertl@openserve.co.za

Theo G. Swart
Dept. of Electrical and Electronic Engineering Science
University of Johannesburg
Johannesburg, South Africa
tgswart@uj.ac.za

***Abstract*—With the constant advances in wireless technology, radio spectrum has become a very scarce resource. Cognitive Radio (CR) has emerged as a viable way to deal with our inefficient use of the radio spectrum by utilizing unused spectrum holes or white spaces, as they are referred to. Using Software Defined Radio (SDR) we are able to realize CRs and their unique properties. Most of the research that has been done on CR protocols has been based on analytical assessments and simulations using non-CR specific network simulators. In this paper we code and compare two existing CR specific Medium Access Control (MAC) protocols using a CR specific simulator. We then prototype the chosen protocols using the Universal Software Radio Peripheral (USRP). This allows us to see how close the simulated performance results come to those actually achieved in a real prototype.**

***Index Terms*—Cognitive radio, MAC protocols, software defined radio**

I. Introduction

We are a data-hungry generation and with the requirement of more data, comes the requirement of more bandwidth. Worldwide, frequency regulators have statically defined spectrum usage to users. This means that fixed usage of frequency spectrum are allocated to entities, enabling them to have exclusive usage of certain frequency bands and channels. This static spectrum allocation policy has worked well in the past, but with a constant increase in bandwidth requirements we are running out of free radio spectrum. According to the Federal Communications Commission (FCC), the assigned spectrum usage varies between 15% and 85% of full capacity in certain geographical regions [1].

This has led researchers to try and find a viable a way to better utilize our scarce spectrum resource. Cognitive Radio (CR) has emerged as a possible solution. With CR we are able to utilize unused spectrum holes either by sensing them as they occur or by having prior knowledge through a database [2]. Once these unused frequency channels are detected, CRs are able to allow Secondary Users (SUs) to use these spectrum gaps without detrimentally affecting license-paying Primary Users (PUs) that occupy the current channels [3].

By using Software Defined Radio (SDR) we are able to realize the above mentioned functions of CRs. An SDR is a radio in which all or at least some of the physical layer properties and functions that are normally controlled electronically are instead controlled through software. Before SDR, radios were all hardware based. This limited radios to operating within a set frequency range and only have support for certain protocols and waveform standards [4].

As with any radio, a CR's operation is bound to certain protocols. Most of the issues concerning CRs can be dealt with on the Physical (PHY) and Medium Access Control (MAC) layers of the typical communications protocol. CR protocols have undergone some extensive research in recent years, especially with regard to the MAC layer [3].

In this paper we code and compare the performance of two existing CR MAC protocols using a CR specific simulator. The two protocols are then prototyped using three USRP SDRs to see how close the prototyped results come to the simulations. Section II covers some related previous work. Section III covers an overview of the chosen CR MAC protocols. In Section IV we discuss the workings of the CR specific simulator and SDR development toolkit. We then explain how the two protocols were coded on it and show the performance results achieved. We also compare the results achieved in the simulated environment to that achieved in the prototyped one and see how close the results come to one another. This is critical in validating the accuracy of the simulation platform and prototyped environment performance results.

II. Related Work

In [5] the authors show how to configure a cognitive engine using GNU Radio [6] and USRP1 [7]. They then consider a Cognitive Radio Network (CRN) consisting of a PU and two SUs to show how their configuration setup works both for the PHY and MAC layer operations.

Latency analyses utilizing GNU Radio were performed on USRP1, USRP2 and USRP E100 in [8]. The sources of

This work is based on research supported in part by the National Research Foundation of South Africa (UID 77596).

latency in these hardware platforms were identified and studied analytically and through experiments.

The authors in [9] utilized the USRP1 to set up an experimental CRN consisting of two PUs and two SUs to demonstrate the effects of dynamic spectrum allocation on legacy systems and coexistence of PUs and SUs. Their focus was on spectrum sensing and coexistence of PUs and SUs.

The authors in [10] reported their experience on using two different software frameworks for SDR design of PHY-MAC layers. GNU Radio and Click [11] were used. They also discuss functionalities that they believe should be offloaded to an SDR device.

In [12] and [13] the authors all used NS-2 [14] to perform simulations on their proposed CR MAC protocols. All of these simulations were, however, designed from scratch in NS-2.

A simple Aloha transmit and acknowledge MAC protocol was implemented by the researchers in [15]. This example of implementation explains the operation of MAC layer protocols in Iris [16] as well as simple USRP implementation using Iris.

The authors in [17] propose a CSMA MAC protocol for SDR radios. Their idea is to introduce SDR researchers to MAC layer design. The MAC protocol operates on a host-based PC. The spectrum-sensing model has the capability to be modified or exchanged without the need to modify the MAC layer component.

III. CR MAC Protocols Considered

CR MAC protocols can be categorized into two main groups, namely: Direct Access Based (DAB) and Dynamic Spectrum Allocation (DSA). With DAB CR MAC protocols, each sender and receiver combination has its own goal that it aims to achieve and therefore does not pay regard to overall network optimization. DSA CR MAC protocols, on the other hand, try to optimize the overall network. Because of this, these protocols use more complex algorithms than DAB CR MAC protocols [18].

Our research focused exclusively on DAB CR MAC protocols since we are only trying to test the user and implementation experience of the CR specific simulator. In comparing only these protocols we will have set a good basis for future CR MAC protocol research, such as the more complex DSA CR MAC protocols using it. The CR MAC protocols we selected are: Opportunistic Spectrum Access MAC (OSA-MAC) and Opportunistic Multi-Channel MAC (OMC-MAC). We encourage the reader to further read references [19] and [20] for specifics on the operation of OSA-MAC and OMC-MAC, as we will only give a brief overview of their operations. Figs. 1 and 2 show the basic time frames of OSA-MAC and OMC-MAC respectively, for reference to our simulations and prototyping. Both protocols are contention-based DAB CR MAC protocols. They both operate in three phases in a set beacon period with periodic beacon transmissions on a common control channel, to allow the synchronization of SUs. Beacon transmissions follow the conventional IEEE 802.11 Distributed Coordination Function (DCF) mode.

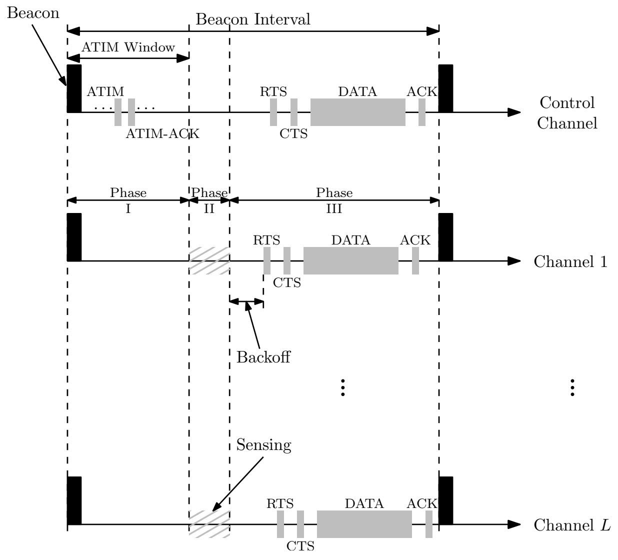

Fig. 1. OSA-MAC time frame [19]

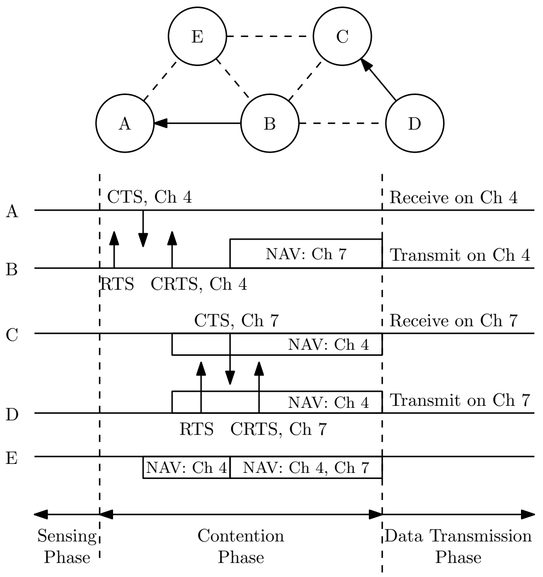

Fig. 2. OMC-MAC time frame [20]

In OSA-MAC, Phase 1 is used for the selection of channels between SUs. The transmitter sends an Ad-hoc Traffic Indication Message (ATIM) packet with the chosen PU channel embedded. On reception, the receiver responds with an ATIM-ACK packet, agreeing on the chosen channel. Availability of the chosen PU channel is then determined by the transmitting SU through channel sensing in Phase 2. Phase 3 is then used for data transmission. Transmitting SUs use random delays to resolve contention before transmitting a Request to Send (RTS) packet to its intended receiver. On reception the receiver responds with a Clear to Send (CTS) packet. The transmitter then sends a single DATA packet to which the receiver responds with an ACK packet to indicate that the data has been received successfully. Only a single packet is transmitted to reduce the chance of interference with the PUs

[19].

In OMC-MAC the first phase is used for sensing. Both the transmitting and receiving SU pair sense the allocated PU channels for availability and keep an index of the scanned channels. In the second phase, the transmitting SUs use random delays to resolve contention over the common control channel before transmitting an RTS packet to its intended receiver which contains the desired PU channel, randomly selected from the available scanned PU channels. On reception of the RTS packet, the receiver will compare the chosen channel with the list of available channels scanned by itself. Should the chosen channel also be available in the receiver's list, it will agree on the channel. If not, it will choose another available channel and respond with it included in a CTS packet. The transmitter then sends a Confirmed-RTS (CRTS) packet as a final confirmation. While all this is happening, other SU pairs monitor what channel has been chosen and set this channel as unavailable for selection. Should all channels have been chosen before a SU pair has made a choice, they will not transmit data for this Beacon Period (BP) and will attempt again in the next BP. Finally, the third phase is used for data transmission. The transmitting SU will transmit a DATA packet to its intended receiver, upon which the receiver will respond with an ACK packet as confirmation. This will continue for the rest of the BP unless an ACK packet is not received. In this case it will be assumed that a PU has become active on the channel and data transmission for the rest of the BP on the specific channel will end [20].

IV. Simulations, SDR implementation and comparison

A. *crSimulator*

crSimulator is a discrete event simulator based on OMNeT++ to allow simulation of CR ad-hoc networks. Users design their network topologies using OMNeT++ NED language and C++ code for all their relevant protocols. crSimulator provides support for all the layers of a typical protocol stack, allowing users to code each part of the stack according to their own needs. It works on OMNeT++ version 4.22 and above. It can run either in Linux or Windows, although Linux offers certain advantages [21].

Although most CR MAC simulations have been completed in NS-2, none of these used a CR specific simulator. Using a CR specific simulator would save a lot of time in coding future CR MAC protocols since all the required building blocks are already there. The only other CR specific simulator available is the Cognitive Radio Cognitive Network Simulator (CRCN) [22]. CRCN runs in NS-2 as an expansion package. It is only compatible with NS-2 version 2.31. Also, it only runs under a Linux environment. The overall user experience of CRCN simulator in NS-2 was found complex and buggy. Also, NS-2 is no longer supported making CRCN simulator undesirable for future use in CR MAC development. crSimulator in OMNeT++ on the other hand was easy to use and install. It runs on both Linux and Windows. We will therefore use crSimulator for all our CR MAC simulations. This will also help with the future extension of crSimulator especially with regard to the MAC layer.

B. *Iris*

Iris is a free open source software radio architecture used to build highly reconfigurable radio networks. It has formed the basis for various cognitive radio and dynamic spectrum access demonstrations at international conferences. It supports runtime reconfiguration, all layers of the network stack and provides connectivity to SDRs such as the USRP. The radio parameters are coded using XML while the physical modules used by the radio are coded in C++ [16].

C. *Simulations*

We performed simulations based on system saturation throughput and the effect of PU arrival rate on system throughput. Fig. 4 shows the basic network setup with one PU and one SU flow used in both protocols. A SU flow is the communication between a SU pair. This communication takes place between the same two SUs for the entire duration of the simulation. A simulation time limit of 40 s was used and an average of 10 runs were taken. A data packet size of 2000 bytes was used over a 2.04 Mbps channel with the other packets having normal IEEE 802.11 sizes. For a full list of all the parameters, including the ones used for the implementations in the next sections, see Tables I and II.

D. *Iris and USRP testbed setup*

Iris was installed alongside the Universal Hardware Driver (UHD) on Ubuntu 14.04. UHD is required to allow a USRP to access the required drivers on the PC [8]. We used the simple implementation of an Aloha MAC protocol coded by the authors in [15] as a starting point for implementing both OSA-MAC and OMC-MAC on Iris. This example showed us how MAC layer functionality is incorporated in Iris and provided a good base to build on. Both protocols operate using two threads running simultaneously in a C++ environment.

TABLE I
Difference in OSA-MAC simulation and USRP parameters

Parameter	Simulation value	USRP value	Factor
aSlotTime	20 μs	20 μs	
DIFS	50 μs	9 ms	$\times$180
SIFS	10 μs	3 ms	$\times$300
PU Sensing Duration	50 μs	10 ms	$\times$200
Channel Switch Delay	224 μs	2 ms	$\times$9
Beacon Period	11–32 ms	150 ms	$\times$5–14
W	64	8	$\times$0.125
Data Packet Size	2000 bytes	1535 bytes	$\times$0.76
Beacon Size	56 bytes	52 bytes	$\times$0.92
ATIM Size	21 bytes	34 bytes	$\times$1.62
ATIM-ACK Size	15 bytes	34 bytes	$\times$2.27
RTS Size	20 bytes	32 bytes	$\times$1.6
CTS Size	14 bytes	32 bytes	$\times$2.3
ACK Size	14 bytes	32 bytes	$\times$2.3

TABLE II
DIFFERENCE IN OMC-MAC SIMULATION AND USRP PARAMETERS

Parameter	Simulation value	USRP value	Factor
aSlotTime	20 μs	20 μs	
DIFS	50 μs	9 ms	$\times$180
SIFS	10 μs	4 ms	$\times$400
PU Sensing Duration	50 μs	10 ms	$\times$200
Channel Switch Delay	224 μs	2 ms	$\times$9
Beacon Period	100 ms	300 ms	$\times$3
W	64	8	$\times$0.125
Data Packet Size	2000 bytes	1535 bytes	$\times$0.76
Beacon Size	56 bytes	52 bytes	$\times$0.92
CRTS Size	14 bytes	32 bytes	$\times$2.3
RTS Size	20 bytes	32 bytes	$\times$1.6
CTS Size	14 bytes	32 bytes	$\times$2.3
ACK Size	14 bytes	32 bytes	$\times$2.3

These threads are the transmit and receive threads and we use two due to the fact that we are operating in a half-duplex mode. Whenever we transmit data from the transmitter we lock the execution of the thread until an acknowledgement packet is received from the receiver, hence the two separate threads.

Another important part of the code to take note of is the processing functions used within Iris. These are the *processMessageFromAbove* and *processMessageFromBelow* functions. These functions are built into Iris and deals specifically with operations between different component layers. They are implemented whenever a new message arrives from either the component above (i.e. application layer) or the component below (i.e. physical layer) the MAC layer component. The message is then passed to either the transmit or receive queue of the two independent threads. The queues operate on a simple first-in first-out principle.

The rest of the code is specific to the implemented protocol. All future half-duplex CR MAC protocols can therefore use the above code architecture and then modify the rest of the code specific to the implemented protocol. Tuning the USRP between the common control channel (CCC) and the PU channels was accomplished using a custom controller in Iris. Whenever the radio needed two switch channels, an event was triggered from within the MAC protocol to activate the controller responsible for the USRP frequency change. A switch event was called for both the transmit and receive chain even though we are operating in half-duplex mode. The reason for this is that both transmit and receive local oscillators operate independently. Automatic transmit-receive logic switches in the FPGA switch between the transmit and receive circuit as soon as transmit streaming comes to an end [23].

The USRP N210 with the SBX daughterboard and the VERT900 antenna were used in the experimental setup seen in Fig. 3. We only had access to 3 USRPs, therefore our experimental setup will be identical to the simulation scenario in Fig. 4. This small prototyping environment still allows us to gain understanding of the workings of CR MAC protocols in a SDR environment on Iris and allows us to effectively compare the results of the simulations to that of the prototyping to see how close the two come to one another. For comparison sake the simulation results included will also only be based on this network scenario, although a much bigger network can be simulated.

A gigabit Ethernet switch was used between one of the SUs and the PU, as the Ethernet port on the host PCs did not have gigabit Ethernet capabilities. Because we are prototyping using a host PC, we have to account for extra delays caused by the host PC, Ethernet cables and the gigabit Ethernet switch. For this reason some of the protocol parameters used in the prototyping such as the BP, DCF Interframe Space (DIFS), Short Interframe Space (SIFS), and PU sensing duration were larger than what was used in the simulations. For this reason the simulations were also re-run with the same parameters used in the SDR prototyping so that appropriate comparisons between simulation and SDR prototyping could be made.

E. OSA-MAC Prototyping

Table I shows the parameters used in the prototyping and how the value for each differed between the simulations and

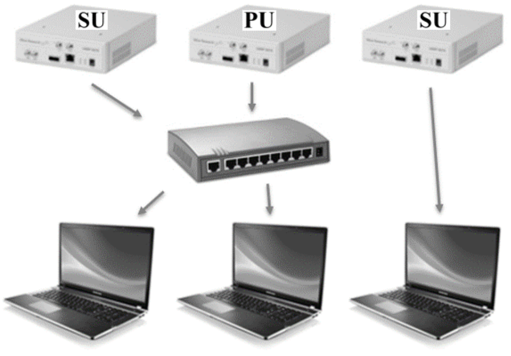

Fig. 3. Complete USRP network setup

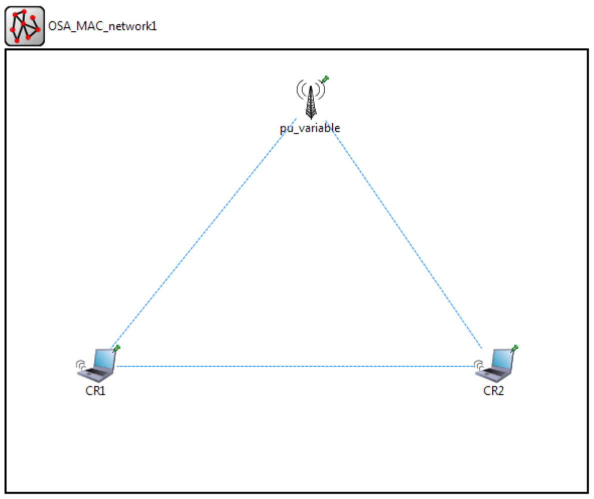

Fig. 4. Network setup for one secondary flow with a PU present

the implementations.

We analyzed the system saturation throughput with no PU presence first. Table III shows the results achieved in the SDR prototype as well as the simulation re-run. It is clear that the throughput stayed the same for all channel cases and that the SDR prototyping matches with what is obtained using simulations in OMNeT++. With a difference of 1.02% between prototyping and simulations, we have a near-perfect match between the two.

With the PU present, we considered the 2 and 5 channel cases and compared the results obtained from the SDR prototyping and the OMNeT++ simulations re-run. We varried the PU arrival rate from 0 to 1. A PU arrival rate of 0 means it never becomes active for the entire simulation time, while an arrival rate of 1 means it is active all the time. Fig. 5 shows the results achieved.

In the 2 channel case both the results showed a drop of around 50% in overall system throughput when the PU arrival rate reached one. This was expected since channels are chosen at random, so there was a 50/50 chance that the PU channel would be selected. A maximum difference in throughput between the two results was experienced at a PU arrival rate of 0.7, where the throughput differed by 7.4624 kbps or 13.26%. There could be various reasons for the difference at this point. It is, however, believed that the main factor for the difference in throughput at various PU arrival rates between the simulated and prototyped results is that the PU becomes active at different start times during prototyping compared to during the simulations. The PU always became active at the same time as the SU pair during simulations, while the PU was left on during SDR prototyping iterations. Furthermore, demodulation errors could also have contributed to the drop in throughput.

The 5 channel case showed similar results, though with a less detrimental drop in throughput since the probability of choosing the PU channel decreased with an increase in channels.

F. OMC-MAC Prototyping

Table II shows the parameters used in the prototyping and how the values for each differed between the simulations and the implementations.

We again analyzed the system saturation throughput with no PU presence first. Table IV shows the results achieved.

We see that the throughput decreased by around 3.37% from the 1 channel case to the 2 and 5 channel scenarios. It was noted that all the switching between the CC and the channels to be scanned and transmitted on resulted in the demodulator losing some frames, since the USRP had not switched over to the correct frequency before transmission occurred.

With the PU present, we again considered the 2 and 5 channel cases and compared these to the results obtained from the SDR prototyping and the OMNeT++ simulations re-run. Fig. 6 shows the results achieved. We started the PU transmission at random times between 0 s and 0.5 s during the simulation re-run to ensure that we achieved the randomness of the PU start time experienced in the SDR prototyping. We did not do this during the prior simulations.

While we see some differences between the simulations and SDR prototyping, they do seem very similar in many ways. With the random start time now included in the OMNeT++ simulations we see a drop in throughput with varying PU

TABLE III
OSA-MAC SYSTEM THROUGHPUT COMPARISON WITH NO PU PRESENCE FOR SDR PROTOTYPING AND OMNeT++ SIMULATIONS

	System Throughput (kbps)		
	1 Channel	2 Channels	5 Channels
SDR Prototype	80.77	80.31	80.37
Simulation Rerun	81.60	81.63	81.54

TABLE IV
OMC-MAC SYSTEM THROUGHPUT COMPARISON WITH NO PU PRESENCE FOR SDR PROTOTYPING AND OMNeT++ SIMULATIONS

	System Throughput (kbps)		
	1 Channel	2 Channels	5 Channels
SDR Prototype	202.93	196.08	196.02
Simulation Rerun	202.62	202.62	202.62

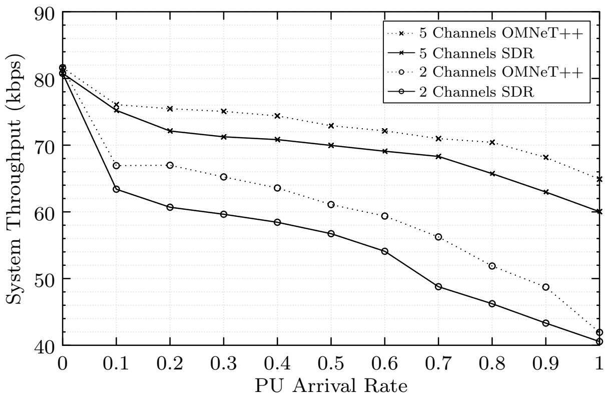

Fig. 5. OSA-MAC with PU present, SDR vs. OMNeT++ results for 2 and 5 channels

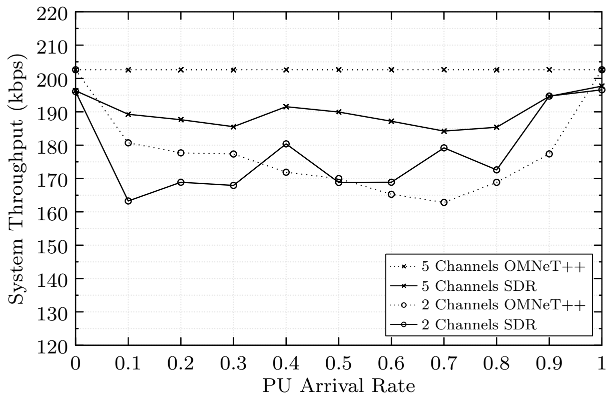

Fig. 6. OMC-MAC with PU present, SDR vs. OMNeT++ results for 2 and 5 channels

arrival rates in the OMNeT++ simulation case as well. This validated our hypothesis for the drop in throughput during prototyping, which was not previously seen in the simulations. The OMNeT++ simulation re-run dropped in throughput until it hit a low of 162.83 kbps at a PU arrival rate of 0.7 (19.88% lower than the maximum). After this point the throughput steadily increased back to the same level it started off with at a PU arrival rate of one. The SDR prototype had more erratic results, showing random increases and decreases, when the PU arrival rate was varied. This non-linear result could have been caused by PU sensing errors made by the SU pair. In the 5 channel case, the variable PU start time did not affect the OMNeT++ simulation results. The SDR prototyping still produced a small drop in throughput. However, it was significantly lower than what was seen in the 2 channel case. The probability of choosing the busy PU were lower, therefore the throughput decreased less.

V. Conclusions

We have successfully programmed and tested both our CR MAC protocols on crSimulator. We have thus showed that CR MAC layer programming can successfully be implemented within crSimulator in an OMNeT++ environment. Future CR MAC research can now build on what we have done. In our research we have only looked at DAB CR MAC protocols. These protocols do not pay regard to overall network optimization. Future work can be done to see if DSA protocols can be coded and tested within crSimulator. These protocols seek to optimize the overall network performance. Because of this DSA CR MAC protocols use more complex algorithms than DAB CR MAC protocols.

The access to only three USRP N210s during our SDR prototyping limited our implementation to a network of one SU pair and one PU. For this reason contention between SU pairs was not coded in both our SDR prototyped CR MAC protocols. Should one have access to 5 or more USRP units in the future, contention between SU pairs can be included in the SDR prototyping.

In our research, delays were included to simulate contention since there would not actually be any contention with only one SU pair. Adding code to deal with contention between SUs will require a more advanced coding effort since we have already seen that large delays were experienced with the current method of channel sensing used. A new channel sensing method will need to be developed to ensure that sensing delays are kept to a minimum not only for contention between SUs, but also for PU sensing.

Cognitive Radio using SDR is a very exciting and new field of study especially with regards to sharing spectrum with PUs in the currently statically defined frequency allocations. We are running out of free spectrum and better ways of utilizing unused spectrum holes need to be found to ensure that we use our scarce spectrum resource efficiently. Using simulations and SDR prototyping will form an important part of quantifying future proposed CR protocols, not only with regards to the MAC layer, but all layers of the protocol stack.

References

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006.

[2] Z. Gao, H. Zhu, Y. Liu, M. Li and Z. Cao, "Location privacy in database-driven cognitive radio networks: attacks and countermeasures," in *Proc. IEEE INFOCOM*, Turin, Italy, pp. 2751–2759, Apr. 2013.

[3] J. Marinho and E. Monteiro, "Cognitive radio: survey on communication protocols, spectrum decision issues, and future research directions," *Wireless Netw.*, vol. 18, no. 2, pp. 147–164, Feb. 2012.

[4] R. D. Raut and K. D. Kulat, "SDR design for cognitive radio," in *Proc. Int. Conf. Modeling, Simulation and Appl. Optimization*, Kuala Lumpur, Malaysia, pp. 1–8, Apr. 2011.

[5] W. Song, "Configure cognitive radio using GNU Radio and USRP," in *Proc. IEEE Int. Symp. Microw., Antenna, Propag. and EMC Technol. for Wireless Commun.*, Beijing, China, pp. 1123–1126, Oct. 2009.

[6] GNU Radio, "GNU Radio, The free and open software radio ecosystem," GNU Radio, Jan. 2017. [Online]. Available: http://gnuradio.org

[7] Ettus Research, "USRP1," Mar. 2014. [Online]. Available: https://www.ettus.com/product/details/USRPPKG

[8] N. B. Truong, Y.-J. Suh and C. Yu, "Latency analysis in GNU Radio/USRP-based software radio platforms," in *Proc. IEEE Military Commun. Conf.*, San Diego, USA, pp. 305–310, Nov. 2013.

[9] Z. Yan, Z. Ma, H. Cao, G. Li and W. Wang, "Spectrum sensing, access and coexistence testbed for cognitive radio using USRP," in *Proc. IEEE Int. Conf. on Circuits and Syst. for Commun.*, Shanghai, China, pp. 270–274, May 2008.

[10] R. Dhar, G. George, A. Malani and P. Steenkiste, "Supporting integrated MAC and PHY software development for the USRP SDR," in *Proc. IEEE Workshop Netw. Technol. for Software Defined Radio Netw.*, Reston, USA, pp. 68–77, Sep. 2006

[11] E. Kohler, R. Morris, B. Chen, J. Jannotti and M. F. Kaashock, "The Click modular router," in *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[12] J. Jia and Q. Zhang, "Hardware-constrained Multi-Channel Cognitive MAC," in *IEEE Global Telecommun. Conf.*, Washington DC, USA pp. 4653–4658, Nov. 2007.

[13] C. Zhang and Y. Xiao, "Performance analysis of cognitive MAC under saturation condition using Statistical Channel Allocation," in *Proc. IEEE Int. Conf. Signal Process.*, Beijing, China, pp. 1369–1372, Oct. 2012.

[14] NS-2, "The Network Simulator - ns-2," NS-2, Jan. 2017. [Online]. Available: http://www.isi.edu/nsnam/ns/

[15] A. Puschmann, "Developing MAC Components with Iris," www.puschmann.net, Feb. 2013. [Online]. Available: http://www.puschmann.net/page/

[16] P. D. Sutton, J. Lotze, H. Lahlou, S. A. Fahmy, K. E. Nolan, B. Ozgul, T. W. Rondeau, J. Noguera and L. E. Doyle, "Iris: an architecture for cognitive radio networking testbeds," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 114–122, Sep. 2010.

[17] A. Puschmann, M. A. Kalil and A. Mitschele-Thiel, "A Flexible CSMA based MAC Protocol for Software Defined Radios," *Frequenz*, vol. 6, no. 9-10, pp. 261–268, Feb. 2012.

[18] A. De Domenico, E. C. Strinati and M.-G. Di Benedetto, "A survey on MAC strategies for cognitive radio networks," *IEEE Commun. Surveys & Tutorials*, vol. 14, no. 1, pp. 21–44, Feb. 2012.

[19] L. Le and E. Hossain, "A MAC Protocol for Opportunistic Spectrum Access in Cognitive Radio Networks," in *IEEE Wireless Commun. Netw. Conf.*, Las Vegas, NV, pp. 1426–1430, Apr. 2008.

[20] S. C. Jha, M. M. Rashid, V. K. Bhargava and C. Despins, "OMC-MAC: An opportunistic multichannel MAC for cognitive radio networks," in *Proc. IEEE Veh. Technol. Conf. Fall*, Anchorage, USA, pp. 1–5, Sep. 2009.

[21] S. N. Khan, M. A. Kalil and A. Mitschele-Thiel, "crSimulator: A discrete simulation model for cognitive radio ad hoc networks in OMNeT++," in *Proc. Joint IFIP Wireless and Mobile Netw. Conf.*, Dubai, UAE, pp. 1–7, Apr. 2013.

[22] T. Chigan, "Cognitive Radio Cognitive Network Simulator (NS2 Based)," July 2014. [Online]. Available: http://faculty.uml.edu/Tricia_Chigan/Research/CRCN_Simulator.htm#_Introduction_to_CRCN

[23] Ettus Research, "Application Note Frontends, Sub-Device Specifications, and Antenna Port Selection," Nov. 2015. [Online]. Available: http://www.ettus.com/content/files/kb/application_note_frontends_subdevices_antenna_ports.pdf