Radio Dust - Decentralized Spectrum Sensing for Cognitive Broadcasting

D. Robu[1], F. Sandu[1], T. Balan[1], M. Serban[1]

[1]Transilvania" University, Electronics and Computers Department
Bd. Eroilor nr 29A, 500036 Brasov, Romania
E-mail: robu.dan@unitbv.ro

Abstract: This paper proposes some models for optimizing the spectrum sensing needed for cognitive broadcasting in radio-congested environments by using different techniques in acquiring data from a "radio dust" - a network of inexpensive SDR devices. Practical solutions cover not only cost-effective SDR receivers but also local intelligence with a spectacular add-on: mobility.

Keywords: spectrum sensing, software defined radio, cognitive broadcasting

1. Introduction

The main objective of the research was to develop a demonstrator for improving multiple node configurations using Radio-Software (ex: USRP – Universal Software Radio Peripheral, RTL-SDR – low cost SDR based on RTL2832U chips or HackRF SDR). The configuration can be driven by National Instruments LabVIEW – for concept validation – with the next step of implementation in the low-power low-cost distributed intelligence platforms as Raspberry Pi, BeagleBone Black etc.

One scenario for the process of "spectrum sensing" is to add some receiver nodes distributed across the area of interest, in order to keep a list of occupied frequencies that can be queried by the "radio dust" SDR before making "spectrum sensing". This would increase their chance for faster finding of available frequencies.

Another scenario is to distribute this list even to the SDR nodes: when a transmitter SDR wants to perform a transmission, it can send a "broadcast" to other SDR-s with the frequency and duration of the intended transmission, thus each node can maintain and update an own list of frequencies announced to be occupied.

The *business management* of the decentralized cognitive broadcasting solutions should consider both CapEx & OpEx (Capital- / Operational- Expenditure).

The performance/cost ratio has more dependencies than for traditional radio.

The ratio of received and transmitted power P_r and P_t and can be computed with the well-known Friis transmission equation (1):

$$\frac{P_r}{P_t} = G_t \cdot G_r \cdot \left(\frac{\lambda}{4\pi R}\right)^2 \tag{1}$$

G_t and G_r are the (fixed or adjustable) gains at the transmitting and receiving side, λ is the wavelength, and R the distance between the transmitter and the receiver. G_t and/or G_r could include any imperfection of the antennas or the channel (-media –environment) absorbtion: The generic formula could be simply considered without these gains, and in isotropic (spherical) propagation conditions, it is only a ratio of equivalent surfaces.
The generic cost of the receiver should be proportional (~) with its sensitivity. One generic assessment of this sensitivity is proportional with the inverse of MDS, „Minimum discernable signal" [W].

For instance, if P_r = MDS, the Friis formula can give us the range R for an available P_t or the needed P_t *to cover a given range R.* Considering the generic cost of the transmitter as proportional with P_t *one could assess the CapEx* ~ (P_t / P_r).
Besides these costs for a traditional radio scheme, one should consider, for our cognitive broadcasting, extra factors like the adaption speed (or its reverse, the adaption time-constant $\tau_{adaptation}$). There is a generic relationship (2) between the most important time-constants (or their magnitude orders):

$$\tau_{sensing} < \tau_{signaling} < \tau_{adaptation} < \tau_{broadcasted\ events} \quad (2)$$

$\tau_{adaptation}$ should include all processing and data propagation delays in the system, a composite „benchmark depending on particular time constants like $\tau_{monitored\ process}$, $\tau_{hopping\ frequency\ transition}$ etc.

A possible "merit factor" of a cognitive radio broadcast solution could be then [(receiver sensibility x transmitter power) / (cost x adaption time constant)].
In terms of CapEx (but also on OpEx), the "overhead" brought by computational intelligence and signalling links should also be considered.
The intersections of these multi-dimensional "hyper-surfaces" of the merit factors of each solution *cathegory* are the *decision* limits for the choice of a family of solutions or another.

- a simple example at the receiver side: ETTUS USRP2 is 8 times more sensitive than the RTL2832U SDR "USB stick", but 100 times more expensive (2000 $ / 20 $).

2. Architecture

We propose the following models:

- a broadcast transmission model, where the intelligence is distributed at each node;
- a centralized model based on dweets, where the "radio dust" (inexpensive SDR receivers) is sending updates to a central server with the occupied frequencies.

2.1. The broadcast model

This broadcast model needs a multiple access communication medium like in the 802.11 specifications. In this way, by using special broadcast messages or multicast messages each node learns the information relevant for the used channels. The ETHERTYPE value of the Ethernet frames can be set to an available value [1]. Each SDR will send such a message if it needs to use a free channel including the time it needs to use this channel. So the broadcast or multicast message will include the

information about the channel/frequency used and the time it will use it. The frame format is derived from the ARP message format [1].

The information is stored in each node in a list. If the broadcast message contains information about the frequencies used by the SDR, then it is added to a list in the memory. The list is periodically cleared of the expired items.

This information is used when a free channel is needed for a transmission, so that the channels present in the table are not checked if they are marked as used. This reduces processing times since costly "spectrum sensing" procedures are avoided in many cases.

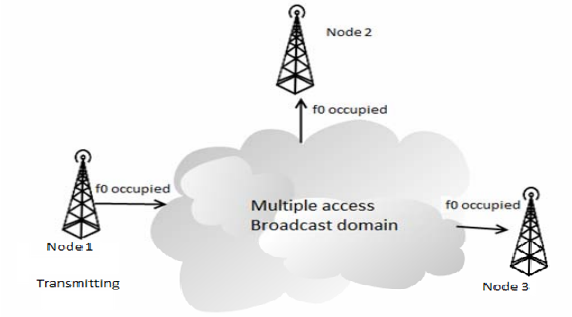

Figure 1. Multiple SDR nodes sharing information about frequencies used via broadcasts

Hardware type	Protocol Type	Hardware address length	Protocol address length	Operation (add or remove)	Sender hardware address	Sender protocol address	Target hardware address	Target protocol address

Figure 2. Proposed frame format for the broadcast messages

2.2. A centralized model based on dweets

The second model that we propose is to use the dweets ("data tweets") infrastructure which is a "Twitter for things" as it is presented on the homepage http://dweet.io [2]

Each transmitting node (SDR) publishes the information about the frequency(-ies) occupied using a dweet. The nodes that might use the same frequency(-ies) subscribe to the same channels, so that they can keep a list with the channels that are already used. When a second SDR wants also to make a transmission, it already knows which channels are for sure occupied so that there is no spectrum sensing for these channels.

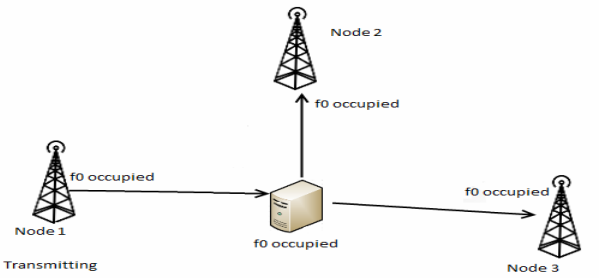

Figure 3. Multiple SDR nodes sharing information about frequencies used via "dweets"

3. Hierarchy and implementation

In the upper hierarchical level, we implemented concentrated computational intelligence plus advanced digital modulation at the transmitter side, in the configuration of Fig.4.

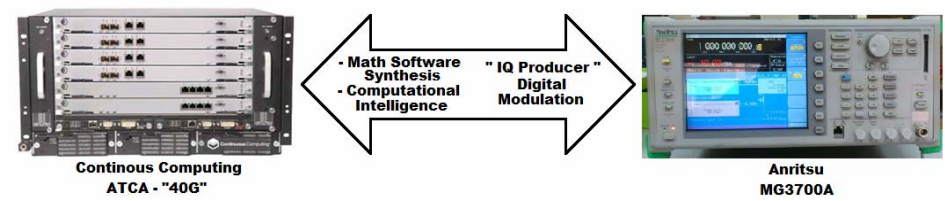

Figure 4. Computational intelligence @ ATCA level plus vector signal generator

The intermediate hierarchical level is using of general purpose computational intelligence and complete USRP (Universal Software Radio peripherals) – Fig. 5

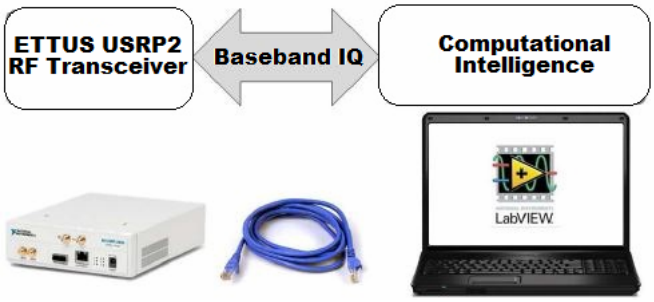

Figure 5. Computational intelligence @ SDR level ETTUS USRP2

The decentralized hierarchical level, with *cost optimization at the receiving side* of SDR-based broadcasting is presented in Fig.6.

The industrial information environment we used, LabVIEW, was updated for the future use in the Cloud – as much as possible, connections are done in the same way,

even if local, not only remote. Thus, towards the unified all-IP controllability, interfacing is controlled as TCP (even if it is USB at lower levels, like in the case of RTL-SDR).

Most data transfers would be generic HTTP GET/POST (port 80 is always opened, compared to dedicated ports for special solutions) [5].

In the VI diagram of Fig.7, LabView drives the RTL-SDR "dongle" via a TCP connexion [6]. This is addressing an important cost reduction (compared with the LabView & USRP solution), with the advantage of maintaining the same level of controllability (with virtual instrumentation, in this case).

In the 1st part of the diagram of Fig.7 it is opened a network connexion to the RTL_TCP embedded server, via the host and the inter-connecting port.

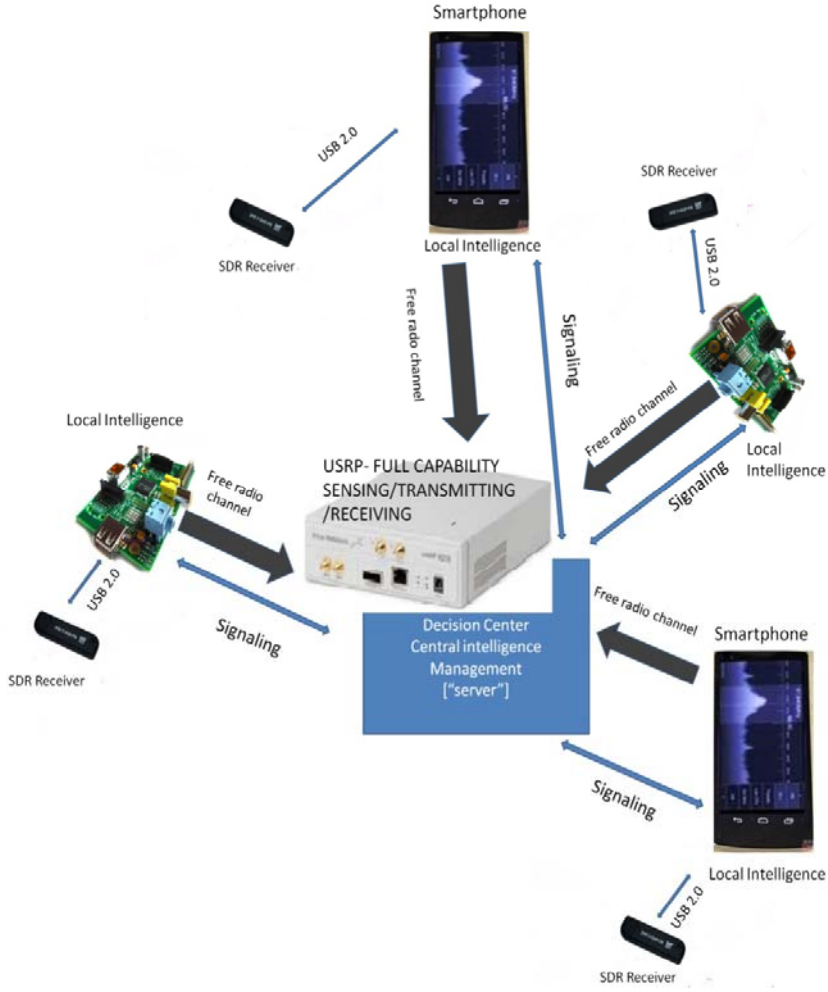

Figure 6. Local computational intelligence in a distributed configuration

In the 2nd part of the diagram, there are set the RTL-SDR parameters (gain, central frequency, sampling rate etc). This controls, in the time-domain, the signal receiving – but, as the investigation of available channels should be done in the frequency-domain, a FFT calculation is performed, giving the signal spectrum [7].

The average of the 10 lowest spectral peaks is done in order to compute the Threshold, then, using this channel limit value, the sub-VI EnergyDetect.vi is used to measure power in the channel [8].

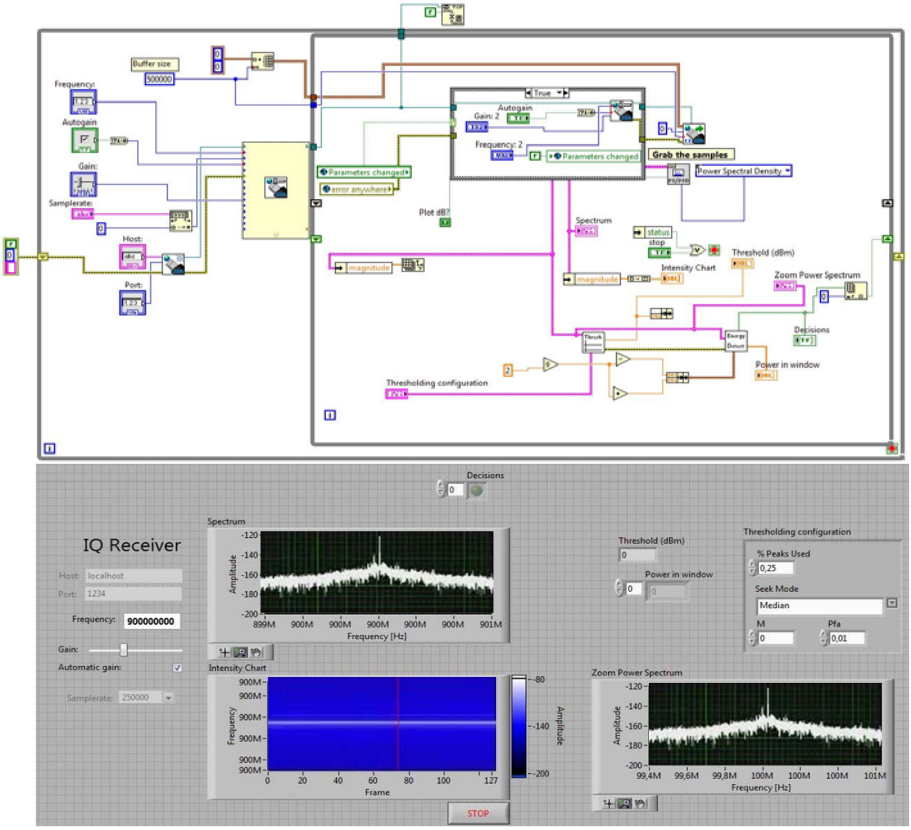

Figure 7. LabVIEW integration of the RTL2832U SDR receiver via TCP

One of our original contributions, that has the additional advantage of *mobility* is presented in Fig.8 – a spectacular solution with computational intelligence at Smartphone level / SDR receiver RTL2832U.

Figure 8. Computational intelligence @ Smartphone level / SDR receiver RTL2832U

3.1. Implementation of the centralized model based on dweets

For this model, the application used on the mobile platform is cURL, which can retrieve and post the information on dweet.io [9].

All models are implemented using low-cost platforms like Raspberry Pi or standard smartphones. The smartphones of today have the distinct advantage of the multiple interfaces available and quite a lot of processing power. These are processing the samples received from a simple IQ ADC present in commercial receivers – Fig.9.

Figure 9. Example of sending/receiving a dweet with cURL for a used frequency

For the simple spectrum sensing some applications can be compiled for the mobile device. They use as an input the data provided by the SDR stick (in fact from its ADC). The driver sends the data over TCP/IP so that it can be used locally or remotely (however around 20Mbps will be used, so extra care is needed to use a transmission medium that can transport this traffic).

For spectrum sensing after eliminating the occupied channel, applications developed using the *liquid dsp* library are used. This is an optimized library for small devices based on ARM architecture [10].

A simple way of viewing the spectrum is the Android application SDRTouch [11]:

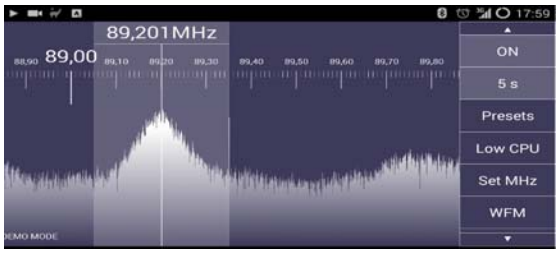

Figure 10. Example of graphical spectrum sense with SDRtouch application

When we want to start manually the capture, we start the RTL-SDR application in advanced mode:

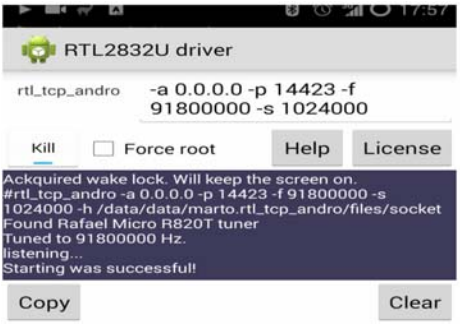

Figure 11. RTL-TCP utility for RTL2832U used for transmitting the IQ samples

4. Conclusions and further development

In our paper, we presented a range of mixed SDR-based models. Implementing any of these models can improve significantly the speed of finding a free broadcasting/transmission channel in the case of cognitive broadcasting, by avoiding unnecessary spectrum sensing. However, choosing the decentralized broadcast model over the centralized one depends on both the transmission delay between the nodes (with the greatest negative impact on the centralized model) and the available computing and network resources in the SDR node (the broadcast model needs a broadcast medium).

The smartphones implementing local computational intelligence offered also the possibility of mobility of the SDR nodes since the common communication channels are IP based – the add-on is a lot of flexibility.

As a further development, we propose another model, based on the use of DNS (Domain Name System) protocol if IP communication is available. This has the advantage that it can include the location information, besides the frequency and the TTL (Time To Live – caching time). The DNS domain and sub-domains will give the location area and the hostname will be the frequency that is "queried". This information will be cached in both clients and servers. At each transmission the SDR will update the information stored in the DNS server and the serial number of the record.

In this way the information is updated in the DNS servers and when a "spectrum sense" is performed, the information is available in the DNS servers based on the location of the SDR and the frequency that it wants to use.

References

[1] Tanenbaum, Andrew S., Wetherall, David J.: Computer Networks (5th Edition), Prentice Hall, 2010, ISBN-13: 978-0132126953

[2] Simple messaging for the Internet of Things - Bug Labs Inc., 2014 [Online]. Available: https://dweet.io

[3] Kamran Arshad, Klaus Moessner: *Robust spectrum sensing based on statistical test*, School of Engineering, University of Greenwich, Chatham, UK
[4] Weifang Wang: *Spectrum Sensing for Cognitive Radio*, iitaw, Third International Symposium on Intelligent Technology Application Workshops, pp.410-412, 2009
[5] National Instruments Corp. - LabVIEW - [Online]. Available: www.ni.com/labview
[6] National Instruments, *An introduction to Software Defined Radio,* Version 1.0-Q1 2013
[7] Thad B. Welch, Sam Shearman, *Teaching Software Defined Radio using the USRP and LabView*, 978-1-4673-0046-9/12/$26.00 ©2012 IEEE
[8] Sushobhan Nayak, *Spectrum Sensing in Cognitive Radio*, Indian Institute of Technology Kanpur
[9] *cURL – Command line tool for transferring data with URL syntax* [Online]. Available: http://curl.haxx.se
[10] Liquid DSP - Free and open-source signal processing library for software-defined radios written in C. [Online]. Available at: http://liquidsdr.org/
[11] SDRTouch - Affordable and portable software defined radio scanner. [Online]. Available at: http://sdrtouch.com/index.php