ANALYSIS OF PRIMARY USER MODELING ACTIVITY ON COGNITIVE RADIO WIRELESS NETWORKS

[1]DANILO LOPEZ SARMIENTO, [2]JOHANA HERNANDEZ VIVEROS

[1,2,3]Distrital University, International University of La Rioja

Abstract— Cognitive radio networks (CRN) are an important paradigm aimed at solving the problems related to efficient and dynamic management of the usable spectrum on wireless networks. Objective: This article attempts to deploy the algorithms presented in [1] (flowchart of the clustering engine and flowchart of the modeling engine) in order to model the activity of a primary user applying these algorithms to traffic flows on a cognitive radio wireless network. Results: It was found that for WiFi networks, the PUs characterization is not sufficiently valid.

Index Terms— Cognitive Radio, Modeling, Wireless, Primary User.

I. INTRODUCTION

This document explains how the algorithm in the article "Primary User Activity First-Difference Modeling Using Clustering and Correlation Filter in Cognitive Radio Networks" [1] has been implemented. This article posits a new model to parameterize main user traffic. The proposed model performs a primary user signal sensing, separating it into clusters using a filter of first differences and checking their correlations. The second part of the article calculates the model performance. These two modules presented in [1] were deployed at the software level. It is important to highlight that because some items in the article were not clearly specified, we explain how these setbacks were overcome based on the assumption of a series of events that could have affected the results in the article. An execution example is shown and finally an analysis is made to explain why the solution presented in [1] is not practical to solve the problem of modeling Wi-Fi type wireless networks flows.

II. COGNITIVE RADIO

Traditionally, the RF spectrum is managed by regulatory agencies by assigning fixed portions of spectrum to individual users as renewable licenses. Although this regulatory approach ensures interference-free communications between radio terminals, it is affected by inefficient use of the spectrum. Recently, the CR has received considerable attention from the scientific community as a technology that enables efficient management of the radio spectrum, and its success will depend on how fast and efficient the process of dynamic spectrum access is [2]. Structurally the CR is supported in software defined radio (SDR), but with ability to learn from its operating environment and adapt to statistical variations according to input stimuli searching for efficient use of network components [3]. Technologically CR appears as a potential real solution to the concept of dynamic spectrum access; which evidences the existence of a clear distinction between these two concepts where DSA relates to standards or methodologies that aim to change the way the spectrum is managed (currently distributed statically), to a more efficient mode by means of a dynamic allocation; on the other hand the CR is considered the technology able to carry the paradigm posed by DSA to reality; and for that the state of the art raises three different mechanisms (or transmission policies) intended to ensure the proper deployment of cognitive radio, minimizing the risk of possible collision or interference generated from SU to PU: the underlying spectrum (spectrum underlay), spectral overlap (overlay spectrum) and spectrum interweave[4], where the common goal for all three cases is the use of opportunistic access without interfering PU transmissions [4]. In [5], Mitola claims that a CR-based network is defined as a complex structure in which devices are able to adapt to the environment. Among the adaptability features is the ability to use the spectrum opportunistically, using its intelligence and autonomy. In general, a CR should be able to perform four tasks efficiently: Spectrum Sensing. Spectrum decision. Spectral sharing and Spectrum Mobility.

Spectrum sensing sweeps frequencies in the area of interest to identify the blank gaps most likely to be used in a given period of time, frequency and power within a specific geographic region.

Spectrum decision identifies with the selected channel or group of channels according to two factors:

1) the features available in the environment;
2) the needs requested by the SU for data transport [6].

Spectrum sharing is properly managing frequency bands maximizing use without causing discomfort in the PUs and other CR users [5], [6].

Spectrum mobility is the capacity of the CR to leave part of the busy frequency spectrum when it starts to be used by a PU and also to search for suitable empty space for communication [5].

CR devices should therefore have, the ability to detect, recognize and adapt to the specific features offered by the environment. One of the most important properties of CRN is the ability to access the spectrum dynamically. However, Cognitive Radio Users (CRU) may also be able to recognize the occupation patterns [7] to reduce the use of energy used in detecting, signaling and transmitting. Because of this, CR has been considered as an alternative capable of reducing power consumption in radio communications and has also been selected as a solution to so-called wireless spectrum shortage [5], [8].

III. METHODOLOGY

The methodology used in developing this research is shown in Fig. 1.

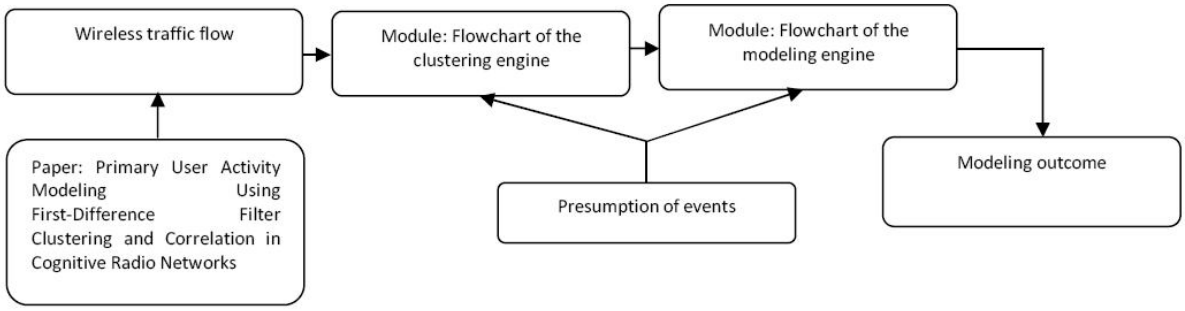


Fig. 1. Modules that make up the modeling analysis of PUs in wireless flows.

The implementation of modules shown in Fig. 1 was performed on the Matlab [9] event simulator specified in a file named filtroCluster.m. This file contains a function that can be called with the command: FiltroCluster (timeSeries, fi). Time Series, corresponds to the temporary series created directly from the simulator. Fi, is the variable that is defined in the Article with the same name. Importantly, we have implemented only the sensing part of the article corresponding to "Clustering-Modeling Module", which also includes Appendix A and B [1] with modifications. These modifications include calculating the delta (it is noted that instead of using the equation $\delta_m \approx r_m - r_{m-1} \quad \forall m \in [1,2,\ldots,p]$ a mean of D) values is done, calculate $|p(m)|$ (where instead of using the equation $|p(m)| = \left[\left|\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(3)}]}{\sigma_{x^{(3)}}} * \frac{r_{m+1-i} - E[r^{(3)}]}{\sigma_{r^{(3)}}}\right)\right|\right]$ a mean is made of values extracted by the formula $|P(x^{(3)}, r^{(3)})| = \left[\left|\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(3)}]}{\sigma_{x^{(3)}}} * \frac{r_{m+1-i} - E[r^{(3)}]}{\sigma_{r^{(3)}}}\right)\right|\right]$; and in the cluster classification the equation has been simplified, since much of the calculations were made to figure out model performance.

The following explains how the models have been implemented, highlighting the fact that the function developed in Matlab implements both defined flow diagrams [1].

The flow diagram (Fig. 2) shows the process as described in [1]. We start by calculating First Difference Filter. For each item in the time series from 1 to p, if the first difference filter is greater than a set data (value specified later in this paper), that point is added to a new cluster, separating it from the previous item. If the first difference filter is smaller than the set data, it will become part of a new cluster if correlations at that point ($|P(x^{(3)}, r^{(3)})|$) are greater than an established correlation. If these assumptions are not met, the item is included in the cluster of which the immediately preceding item is part. Thus at the end of the execution we will have a set of clusters whose members are always consecutive items, where each cluster represents a change in the dynamics of the signal.

From Fig. 2 the algorithm was defined and implemented as follows:

1. For the entire temporary series the D array differences are calculated (equation: $D_m = r_m - r_{m-1} \quad \forall m \in [1,2,\ldots,p]$) It corresponds to the "First difference Filter" in the previous flow chart diagram:

$$D_i = r_i - r_{i-1}$$

2. Since that article does not specify how to calculate the delta ($\delta_m \approx r_m - r_{m-1} \quad \forall m \in [1,2,\ldots,p]$),It is calculated by averaging values multiplied by fi. This step is not shown in the article flowchart diagram.

$$delta = media(D) * fi$$

3. To calculate p correlations (Linear Pearson Product-Moment Correlation), the equation is used as it is defined in the article:

($|P(x^{(3)}, r^{(3)})| = \left[\left|\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(3)}]}{\sigma_{x^{(3)}}} * \frac{r_{m+1-i} - E[r^{(3)}]}{\sigma_{r^{(3)}}}\right)\right|\right]$).

$x^{(3)} = [1,2,3]$, therefore they are fixed values for each p element.

$$R_m = |P(x^{(2)}, r^{(2)})| = \left[\left|\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(2)}]}{\sigma_{x^{(2)}}} * \frac{r_{m+1-i} - E[r^{(2)}]}{\sigma_{r^{(2)}}}\right)\right|\right]$$

fi since the article does not clearly define how to do the math.

$$\forall\, m \;\rightarrow\; |p(m)| = pMean = media(P) * fi$$

4. The formula:

($|p(m)| = \left[\left|\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(3)}]}{\sigma_{x^{(3)}}} * \frac{r_{m+1-i} - E[r^{(3)}]}{\sigma_{r^{(3)}}}\right)\right|\right]$), is calculated by averaging the values of p multiplying by

Note that both items three and four are used to evaluate the formula $|P(x^{(2)}, r^{(2)})| \geq |p(m)|$.

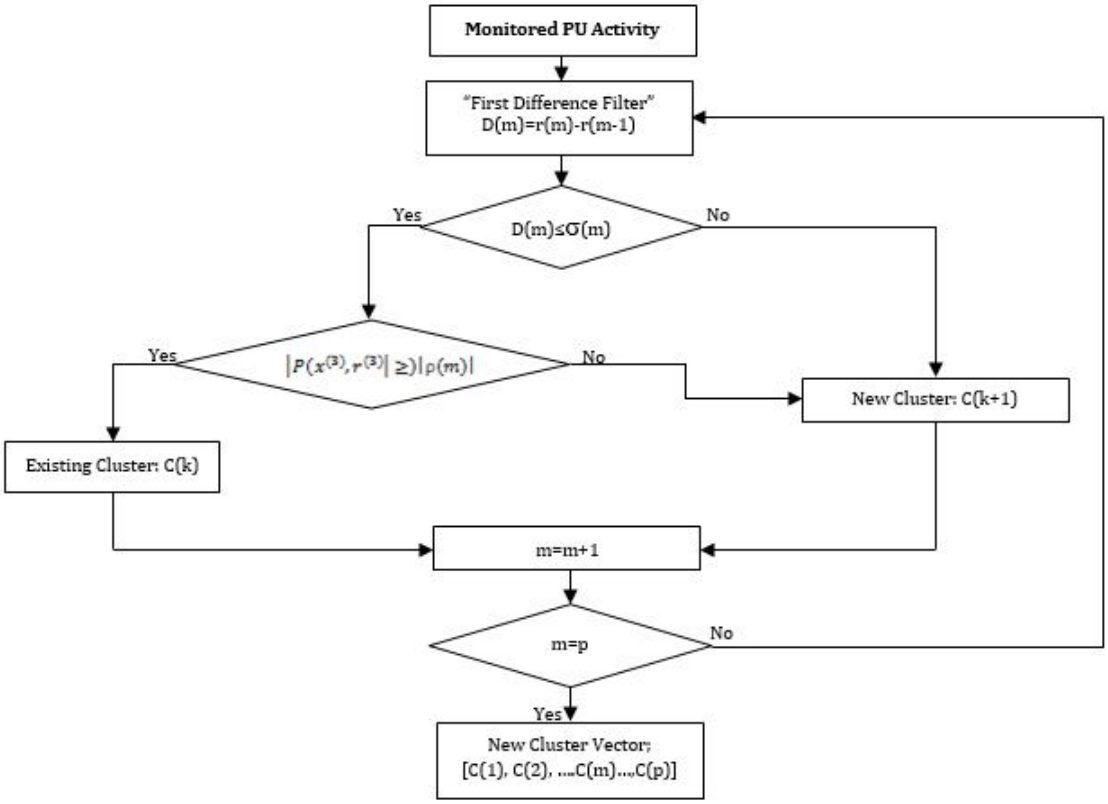


Fig. 2. Flowchart of the clustering engine (modified 1).

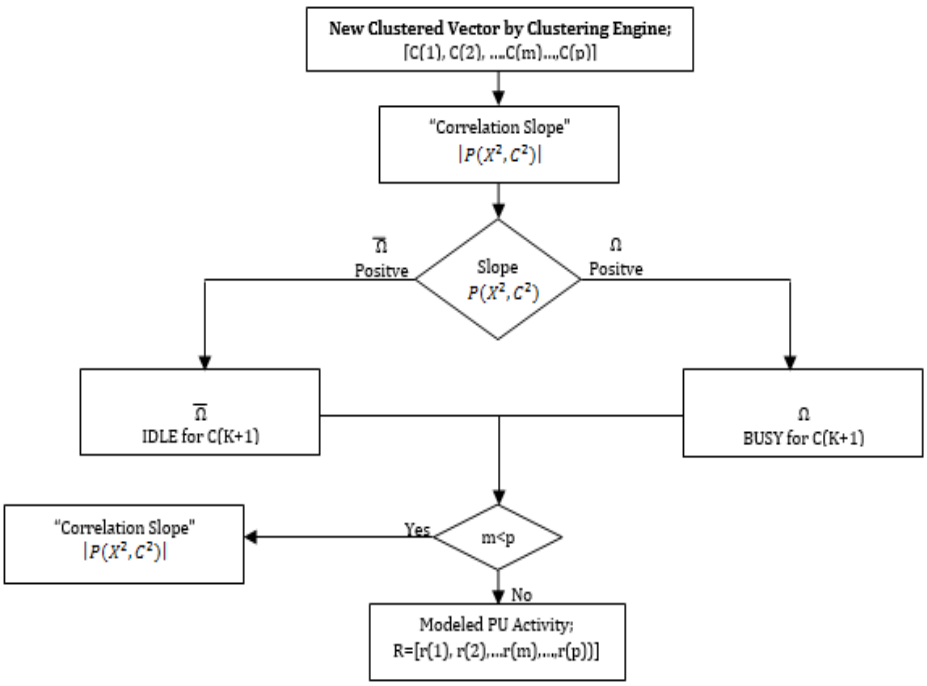


Fig. 3. Flowchart of the clustering engine [modified 1].

1. The correlation (Pearson Product-Moment Linear Correlation) between consecutive clusters is calculated using equation:

$$|P(x^{(2)}, c^{(2)})| = \left|\left[\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(2)}]}{\sigma_{x^{(2)}}} * \frac{c_{m+1-i} - E[c^{(2)}]}{\sigma_{c^{(2)}}}\right)\right]\right|$$

$x^{(2)} = [1,2]$, therefore they are fixed values for all Slope elements:

$$Slope = |P(x^{(2)}, c^{(2)})| = \left|\left[\frac{1}{2}\sum_{i=1}^{2}\left(\frac{x_i - E[x^{(2)}]}{\sigma_{x^{(2)}}} * \frac{c_{m+1-i} - E[c^{(2)}]}{\sigma_{c^{(2)}}}\right)\right]\right|$$

This variable only returns -1 or +1 depending on whether the value of c(x) is larger or smaller than c(x-1), So that the formula could be simplified, although this change has not been applied in order to remain as close as possible to the stipulations in [1].

2. Then a verification is made from 1 to m where m <p, if the Slope is negative or positive. If negative, we conclude that we are not transmitting anything in the time instants included in the cluster. If Slope is positive, at time points contained in the cluster, it is decided that data is being transmitted.

Finally, it is important to mention that calculations are simplified, since the point is not to calculate the ck cluster state but, the ck+1 cluster since this ck cluster was already calculated in the previous iteration. Looking at Table III in [1] and only the cluster decision being necessary, we can see that only the information of Ω is needed to know if the cluster is in IDLE or BUSY, implying that Ω corresponds to whether we have negative or positive Slope in Fig.4 (shown later in this paper); thus the equations can be again simplified.

IV. CODE IMPLEMENTED AND APPLICATION EXECUTION

The code sequence implemented in Matlab for the simulation (flowchart diagrams in Figs. 2 and 3) is shown in Fig. 4.

Application execution starts by creating a simulated and processed signal (ie, captured with Wireshark and processed); upon completing satisfactory execution of wave simulator (either loading or creating a signal), the following code is executed in order to execute the algorithms discussed above:

```
sensing = FiltroCluster (signal, fi);
```

Where fi is a decimal value between 0 and 1. As an example a possible data sequence would be:

```
sensing = FiltroCluster(signal, 0.45);
```

To display the entire signal the following function must be called:

```
PrintSensing (signal, sensing);
```

If you wish to show a fragment of the signal you could run the following line:

```
PrintSensing(signal(x:y,:), sensing(x,y));
```

```
p = abs(p);
pMean = mean(p)*fi;

cluster = zeros(length(timeSeries(:,2))-2,4);

k = 1;
startingCluster = 1;


for m = 3: length(timeSeries(:,2))


    if ( D(m-1) < delta)

        cluster(k,:) = [startingCluster,m-1,mean(timeSeries(startingCluster:m-1,2)),0];
        startingCluster = m;
        k = k+1;
    else
        if (p(m-2) < pMean)

            cluster(k,:) = [startingCluster,m-1,mean(timeSeries(startingCluster:m-1,2)),0];
            startingCluster = m;
            k = k+1;
        end
    end
end

cluster(k,:) = [startingCluster,m,mean(timeSeries(startingCluster:m,2)),0];
cluster = cluster(1:k,:);

p = abs(p);
pMean = mean(p)*fi;

cluster = zeros(length(timeSeries(:,2))-2,4);

k = 1;
startingCluster = 1;


for m = 3: length(timeSeries(:,2))


    if ( D(m-1) < delta)

        cluster(k,:) = [startingCluster,m-1,mean(timeSeries(startingCluster:m-1,2)),0];
        startingCluster = m;
        k = k+1;
    else
        if (p(m-2) < pMean)

            cluster(k,:) = [startingCluster,m-1,mean(timeSeries(startingCluster:m-1,2)),0];
            startingCluster = m;
            k = k+1;
        end
    end
end

cluster(k,:) = [startingCluster,m,mean(timeSeries(startingCluster:m,2)),0];
cluster = cluster(1:k,:);
p = zeros(length(cluster)-2,1);

for m = 1:length(cluster)-2
    ans = 0;
    media = mean(cluster(m:m+2,3));
    desviacionTipica = std(cluster(m:m+2,3));
    for i=1:3
        ans = ans + (i-2)*(((cluster(m+3-i,3)-media))/desviacionTipica);
    end
    p(m) = ans;
end


p = abs(p);
pMean = mean(p)

fprintf('Terminada tercera fase\n');


p2 = zeros(length(cluster)-2,1);
xmean = mean([1 2]);
xstd = std([1 2]);
x1 = (1-xmean)/xstd;
x2 = (2-xmean)/xstd;
for m = 1:length(cluster)-2
    media = mean(cluster(m:m+1,3));
    desviacionTipica = std(cluster(m:m+1,3));
        ans = (x2*((cluster(m,3)-media)/desviacionTipica)) + (x1*((cluster(m+1,3)-media)/desviacionTipica));
    p2(m) = ans;
end
% Implementación de figura 4
m=1;
k=1;
% Primer punto aleatorio
cluster(1,4) = round(rand(1));
% m representa el puntero para recorrer la señal monitorizada
% Implementación de figura 4
for m = 1: length(cluster)-2
    % Slope aqui hay que dividir p /|p| pero es lo mismo que ver si es
    % positivo o negativo.
    if ( p2(m) < 0) % negativo omega
        if (p(m) > pMean) % positivo psi
        else % negativo psi
        end
        % Next cluster busy
        cluster(m+1,4)=1;
    else % positivo omega
        if (p(m) > pMean) % positivo psi
        else % negativo psi
        end
        % Next cluster idle
        cluster(m+1,4)=0;
    end
end

% Poner valor a los clusters en función de si emiten o no
output = zeros(length(timeSeries(:,2)),1);
for m = 1: length(cluster)
```

Fig. 4. Code implemented in Matlab.

V. RESULT AND ANALYSIS

After completing the above steps, output or result obtained at the level of emission/no emission prediction is shown in Fig. 5. In the upper graph of this figure, the emission and no emission signal is shown while the graphic below shows the emission/no emission prediction level. It can be seen clearly that prediction varies constantly between emission and no emission when in the actual signal there is only a long emission interval between two no emission intervals. For a more detailed analysis and applying an extension to the above chart: "Print Sensing (signal(1900:2000,:), sensing(1900:2000));", where *x* is the beginning of the signal to be printed and y the end, Fig. 6 is generated. The upper figure represents the actual emission and no emission signal, while the lower (blue signal) represents prediction of emission and no emission, and the red signal represents signal energy at each time instant. Also you can see in Fig. 6 that there is an interval of time in which we are not emitting. However prediction detects constant variations in emission and non-emission. This is due to the nature of the model and that there is a sinusoidal signal on which the prediction is applied. The model detects, by creating clusters, changes in signal behavior, mainly drops or sudden power surges. These changes may be due to a high level of noise in the signal. In the simulated signal, each cluster is generated based on the radical changes in power over the last three timestamps for each unit of time. Then each cluster is evaluated based on its Slope with respect to previous Slopes and the total signal Slopes average. The result is a not very accurate detection of emission and no emission moments.

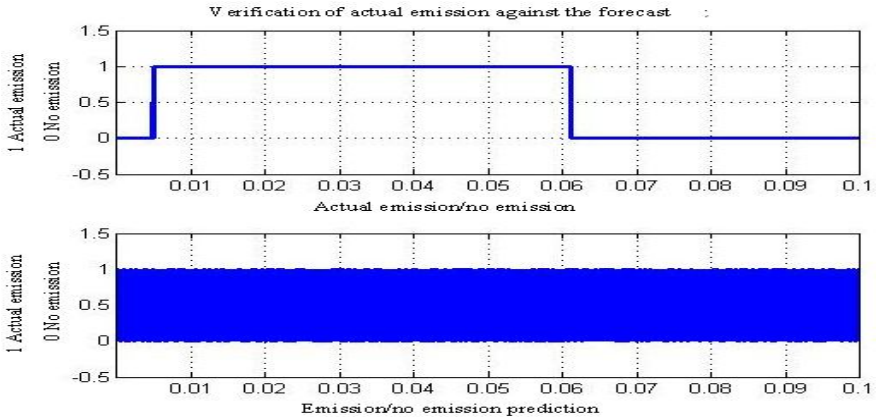


Fig. 5. Emission/No emission prediction.

From the above it can be concluded that using constant energy in emission instead of sine waves (including noise), we note that implementing the two algorithms as specified in this document (separating the signal in clusters using First Difference Filtering and Linear Pearson Product-Moment Correlation and checking each cluster for positive or negative slope), it is possible to reach the same results as those described in [1], setting a threshold to delimit emission and no emission energy values. With this threshold we could see if the cluster value is above or below and thus know whether or not we are emitting. Moreover, we could save us the calculations to generate clusters, and compare this threshold with values in each timestamp in the temporal series. Actually in the calculation of clusters what is being done is to soften the signal using a three unit wide window (correlation). The easiest way to soften with a three unit window is to calculate the average of the last three values.

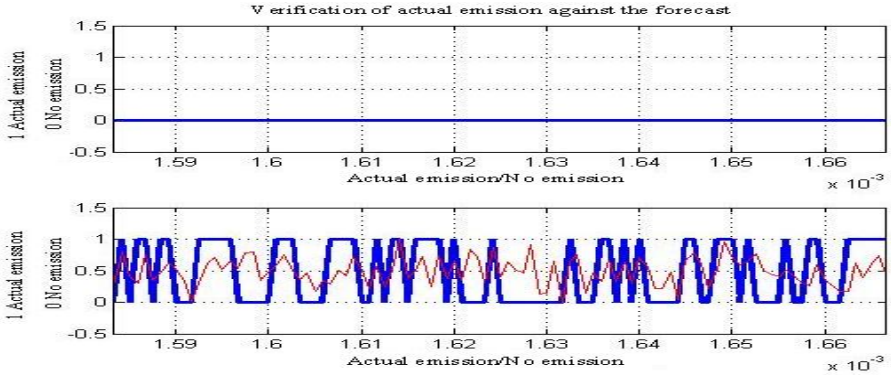


Fig. 6. Emission/no emission prediction zoom.

CONCLUSIONS

The working environment is defined in [1] specified in the following sentence: "…the CR user transmission C_{SU} under hypothesis H_0 is constant with 20 DB". In this passage it tells us clearly that while we are emitting, we do so with a constant energy value. Any real environment uses different modulations based on sine waves, which is why the assumption in the article could eventually invalidate its use for sensing and modeling in a real WIFI network environment level as shown in Fig. 6. Specifically, the implemented simulator uses sine waves which have varying values during data emission. Although it is true that in an idyllic setting during no emission there is no energy in the signal but, during emission, the sine wave minimums have zero energy too. If white noise is added, always present in real environments, we cannot use sensing by detecting energy because we would detect emission false negatives

IMPORTANT ENTRY

It should be noted that this article takes as important fundamental reference the paper: [1], since its analysis and construction is generated from what was developed there both qualitatively and quantitatively in algorithms and approach.

REFERENCES

[1] C. Berk, I. Akyildiz, S. Oktug, "Primary user activity modeling using first-difference filter clustering and correlation in cognitive radio networks," IEEE / ACM Transactions on Networking. Vol. 19, pp. 170-183, 2011.

[2] Q. Zhao, L. Tong, A. Swami, "Decentralized cognitive MAC for dynamic spectrum access", in Proc. 1st IEEE Int. Symposium for New Frontiers in DySPAN, Reston, Virginia, USA, Sep. 26–35, 2005.

[3] S. Haykin, "Cognitive radio: brain-empowered wireless communications", IEEE J. Sel. Areas Commun. Vol. 23, pp. 201–220, 2005.

[4] L. Vinuesa, "Spectrum management in cognitive radio networks", Politecnica of Catalunya University, 2008.

[5] J. Mitola, Q, Maguire, "Cognitive radio: making software radios more personal", IEEE Pers. Commun. Vol. 6, pp. 13–18, 1999.

[6] I. Akyildiz, L, Won-Yeol, M, Vuran, S. Mohanty, "A survey on spectrum management in cognitive radio networks", IEEE Communications Society, Communications Magazine. Pp. 40-48, 2008

[7] N. Bolivar, "Medium access control messaging scheme for cognitive radio networks", Department of Computing, Architecture and Technology, Girona University, Spain, 2012.

[8] A. Goldsmith, S. Jafar, I. Maric, S. Srinivasa, "Breaking spectrum gridlock with cognitive radios: An information theoretic perspective", Proc. IEEE. Vol. 97, pp. 894–914, 2009.

[9] Mathworks. [Online], retrieved on www.mathworks.com/index.html?s_tid=gn_logo.

★★★