This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

Sliding-Window Algorithm for Asynchronous Cooperative Sensing in Wireless Cognitive Networks

Chengqi Song, Qian Zhang,
Department of Computer Science and Engineering, Hong Kong University of Science and Technology
{lars, qianzh}@ust.hk

***Abstract*—To improve the performance of spectrum sensing, cooperation among Cognitive Radios (CRs) has been proposed recently as an effective solution. Most existing works require either time synchronization or extra infrastructure support, which are not always practical. This paper proposes an asynchronous spectrum sensing framework which is of high accuracy, short response time and ease to implement. Within such a framework, each node uses our proposed Sliding-Window algorithm to analyze the spectrum status with the sensing results received from its neighbors. This algorithm maintains a minimal and sufficient number of sensing results using Sliding-Window method, and applies Likelihood Ratio Test (LRT) on them to make a decision. This algorithm is evaluated both in theory and by simulations. The results proved that our proposed cooperative sensing solution can show good performance in various situations.**

Keywords-cognitive radio, spectrum sensing, sliding window

I. INTRODUCTION

Recently, Opportunistic Spectrum Sharing (OSS) has attracted a lot of attention. It allows secondary users to operate in licensed spectrum bands without harmful interference to primary users so that the utilization of radio spectrum resource can be significantly improved. Cognitive Radios (CRs) are the essential technology to implement OSS because they are able to sense the spectrum environment and make use of unused spectrum bands.

To sense unused spectrum and avoid interference to primary users, CRs need to carry out accurate spectrum sensing and respond to the change of spectrum as soon as possible. However, that is not easy for individual radios because of limitation of hardware, noise etc; therefore cooperative sensing has been introduced recently. In cooperative sensing, secondary users receive sensing results from neighbors to determine the spectrum status. Thus a common and crucial problem is how to fuse multiple results together. It has been discussed in several former works. In [3] and [4], the authors explore the fact that by adding up sensing result from different radios, signal-noise ratio (SNR) can be improved. This method is far from the best because it only considers the noise. In [6], weighted sequential probability test (WSRPT) is used to fuse reports. It performs higher accuracy than [3] and [4]. However, all these methods require strict synchronization of spectrum sensing, which may impose a big overhead.

In this paper, we propose a novel asynchronous cooperative spectrum sensing framework. In this framework, secondary users share their sensing results to the neighbors without time synchronization requirement. And each secondary user makes local decision basing on sequentially arrived reports from nearby nodes including it self, which makes the decision more difficult to make but the system more lightweight and easier to implement. Besides, such a scheme can significantly reduce secondary users' response time for the variation of spectrum status. It can also satisfy users' requirements of accuracy adaptively.

Such a system design has some challenges we have to deal with. First, without synchronization, sensing results come sequentially instead of in a batch. This means most reports are outdated, and we have to make decisions out of the outdated reports. Secondly, in order to make the scheme be easily used in various CR networks, we need to keep the computational overhead as low as possible. To address these problems, we propose a Sliding-Window based cooperative sensing scheme. At first, we use a sliding window method to minimize the number of reports we need to consider, otherwise it's too complicated to process all received reports. Secondly, we apply Likelihood Ratio Test (LRT) on reports to analyze the spectrum status, which is a robust statistical test without complex computation. This paper makes three contributions on the cooperative sensing in CR networks:

1. Totally different with existing works, our cooperative sensing framework works without synchronization and can be easily implemented more widely in various CR networks.
2. Besides high sensing accuracy, Sliding-Window algorithm also contributes on improving response speed to variation of spectrum.

The rest of the paper is organized as follows: in Section II, we investigate existing works. Section III introduces the overview framework of this work: the system model, critical issues and brief description of the solutions. In Section IV, we describe Sliding-Window algorithm in detail. The performance of our algorithm is evaluated with both theoretical analysis and numeral results in Section V. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Since sensing the existence of primary user is a crucial problem in CR networks, it has drawn great attention in recent years. To address this problem, cooperative sensing is proved to be an effective way.

In [3], the authors point out that the noise on secondary users is independent. So SNR can be improved by simply adding

The research was support in part by grants from RGC under the contracts CERG 622407 and N_HKUST609/07, by grant from National Basic Research Program of China (973 Program) under the contract No. 2006CB303100, the NSFC oversea Young Investigator grant under grant no. 60629203, National 863 Program of China under Grant No. 2006AA01Z228.

sensing results from different users. In [4], the same method is used; moreover, the authors also discuss the impact of malicious nodes on their proposed scheme. However, these two works consider the difference of each nodes and possible malicious behaviors. Therefore, they can not achieve good performance. In [6], the authors focus on how to fuse the sensing reports from nearby secondary users, and propose a weighted sequential probability ratio test (WSPRT) scheme, which can achieve high detection rate and low false alarm rate when there are malicious nodes. However it requires synchronization among nearby nodes, which is not always practical. Moreover, another result is that the reaction time to primary users' activities is improved; if neighbors can send results at different time, the nodes may detect the change before their period expires.

Our Sliding Window Algorithm is different from the above solutions and achieves several improvements. At first, it's easy to implement because it doesn't need any synchronization or extra sensing. Secondly, it can adaptively satisfy users' different requirements of detection rate and false alarm rate. At last, it can improve reaction speed to change of spectrums because it can make a decision once it collects an enough number of results from neighbors without waiting for the expiration of a fixed sensing period. In the following sections, we introduce how it achieves these improvements.

III. Framework of Asynchronous Cooperative Sensing

In this section, the motivation of asynchronous sensing and the challenges to enable it are discussed, and then the framework of Sliding-Window scheme is introduced.

A. Asynchronous Cooperative Sensing

As shown in Fig. 1(a), in cooperative sensing schemes, a node collects sensing results from the neighbors to make decisions about spectrum status. The cooperation can be synchronous or asynchronous. In synchronous schemes, as shown in Fig. 1(b), all nearby nodes need to sense spectrum simultaneously to achieve better accuracy, but synchronization is a considerable overhead and sometimes not practical. On the other hand asynchronous schemes are more lightweight. As shown in Fig. 1(c), nodes sense at different patterns without any constraint. But such scheme has seldom been discussed, because there are some challenges associated with it:

i) In synchronized schemes, all nodes stop transmission and sense spectrum at the same time. So they do not disturb each other's sensing. Without synchronization, how to make sure that a node's sensing is not disturbed by other's transmission is a problem.

ii) In synchronized schemes, a node receives a batch of sensing reports of the latest spectrum status periodically, and such data is easy to process. Without synchronization, how to process sequential sensing reports?

For the first challenge mentioned above, a node can send a special packet to keep neighbors quiet for its sensing. For example, in the CR network testbed KNOWS [5] RTS packet is used to avoid inference to sensing. It is very easy to implement such RTS mechanism in practice.

In this paper we focus on the second problem and solve it with Sliding-Window algorithm. The framework of the algorithm is introduced in next sub section.

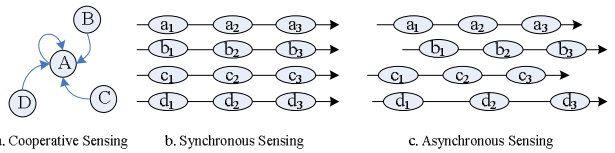

Fig 1. Synchronous and Asynchronous Sensing (A, B, C, D are 4 nearby users, a1 to a4, b1 to b4 etc. are their sensing result, arrows are the direction of time)

B. Sliding Window Algorithm

In asynchronous cooperative sensing, a node needs to detect the change of spectrum as soon as possible basing on received sensing reports. Because the reports are sequential, this is a typical *sequential change-point detection* problem in statistic theory: detect the change of the distribution of a set of random samples. General theoretical methods are introduced in [7-9]. However, they require analysis on the whole sample set and need complicated computation. Therefore, in our Sliding-Window algorithm, we make use of the special characteristics of CR networks and provide a more practical solution.

Instead of analyzing the distribution of all reports, we choose Likelihood Ratio Test (LRT) to make decisions. LRT is very effective when there are only two hypotheses to determine and in this problem we do have only two hypotheses: primary user is active or inactive. With LRT, a ratio is computed between the likelihoods of two different spectrum statuses under known sensing reports, so that a node can make decisions basing on this ratio by:

$$\begin{aligned} &\Lambda(\Omega) < \eta_0 && \text{say } S=0 \\ &\eta_0 < \Lambda(\Omega) < \eta_1 && \text{say need more reports} \\ &\eta_1 < \Lambda(\Omega) && \text{say } S=1 \end{aligned} \quad (1)$$

Where Ω is received reports, $\Lambda(\Omega)$ is likelihood ratio, η_0 and η_1 are two thresholds, and S is the decided spectrum status. $\Lambda(\Omega)$ is the ratio of likelihood $L(S=1|\Omega)$ and $L(S=0|\Omega)$,which are equal to $P(\Omega|\ S=1)$ and $P(\Omega|\ S=1)$.

The precondition to apply LRT is to calculate the likelihood, which is not easy because:

i) The number of received sensing reports can be very large, how can we apply LRT on the whole sequence? A practical method is to apply LRT only on the latest part of reports. Then another question is how many reports should be used?

ii) It's hard to calculate the likelihood even on a small set of reports. In asynchronous sensing, most of the reports are outdated and the spectrum status is changing all the time. How can we calculate the likelihood out of outdated reports?

These two problems are solved in Sliding-Window Algorithm by two methods: Sliding-Window method is developed to minimize the number of needed reports and Divide and Conquer method is developed to calculate likelihood.

1) Observation Window

In our scheme, we only observe latest reports, instead of the whole sequence. As shown in Fig. 2, a node has received a lot of reports, but only makes use of the latest 4 reports in the *Observation Window*. From (1), it shows that when using LRT, a node can not make decision without enough confidence. Relatively, the sliding window method works in this way: i) when we receive a new sensing report, add it to the observation window, then try to make a decision; ii) if there is no enough confidence, wait for next report, otherwise iii) make a decision, and try to remove the oldest useless reports, keep

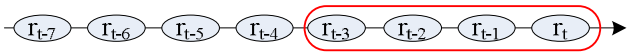

Fig. 2. Sliding Window

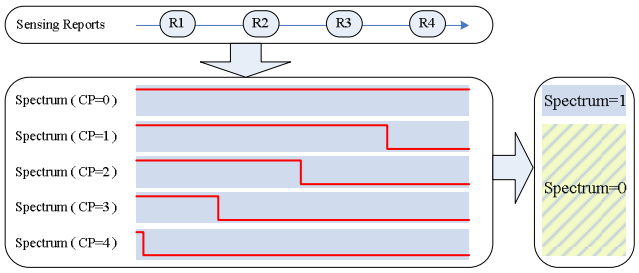

Fig. 3. Divide and Conquer method

removing until we are about to have no enough confidence. The method makes sure that we have enough confidence to make decision, meanwhile minimize the number of observed reports.

2) Divide and Conquer Method

By Sliding Window methods, we get a proper set of reports to process, now we discuss how to calculate the likelihood ratio basing on them. Take Fig. 3 as a sample. $R_1 \ldots R_4$ are current observation window, in which R_4 is the latest report and $R_{1\text{-}3}$ are outdated. The known spectrum status is 1, which means primary user is active. Now the problem is to determine the likelihood of current spectrum under $R_{1\text{-}4}$. Since only R_4 is the sensing result of current spectrum, how to make use of R_1 to R_3?

We analyze this problem in a novel way. We notice this fact: in practice, spectrum sensing must be much more frequent than spectrum status's change. Otherwise, e.g. if spectrum changes once an hour and secondary user also sense once or twice an hour, there is no way to avoid interference with primary users. Based on this fact, we can imply: spectrum changes at most 1 time in the period of an observation window. Consequently, there are limited possible hypotheses of spectrum status. As illustrated in Fig. 3, there are only 5 possibilities: spectrum didn't change, changed before report R_4, before R_3, before R_2, and before R_1. We use *CP* (Change Point) = *0 to 4* to indicate them. For each possibility, the spectrum status at different time is fixed exactly. So we can calculate the likelihood of each possibility under known $R_{1\text{-}4}$. After that, we can see that under situation of *CP=0*, current spectrum is still 1, and under situation of *CP=1 to 4*, current spectrum is changed to 0. Therefore we calculate the likelihood in a Divide and Conquer way: at first divide the problem into 5 sub problems, and then solve them individually, at last combine them into the result we want.

So far the Sliding-Window algorithm's framework has been introduced; in the next section we describe its details.

IV. Algorithm Design

In Section III, the framework of Sliding-Window algorithm is introduced, but not in detail. In this section we at first model the asynchronous spectrum sensing problem, then describe our algorithm in detail.

A. Problem Modeling

For a secondary user, as user *A* in Fig. 1, it receives sensing reports from all neighbors, define the received reports as:

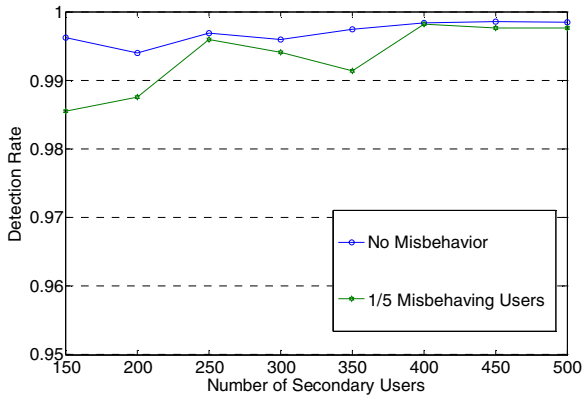

Fig. 4. Detection Rate (Requirement is 0.98)

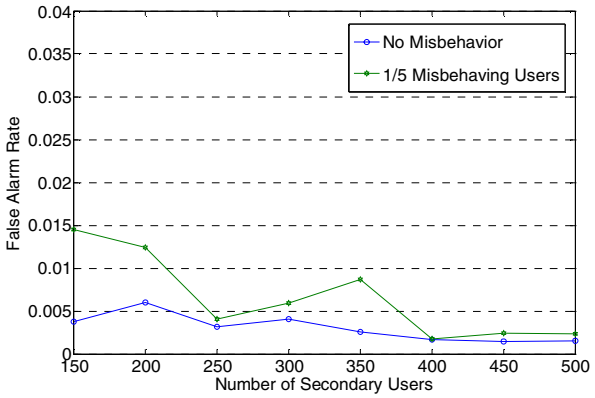

Fig. 5. False Alarm Rate (Requirement is 0.02)

$$Seq=\{r_i \mid i=1,2,3\ldots\}$$

Where r_i is the *i*th report, it comes from node $n(r_i)$. Because each report is sensed independently, we can assume that r_i are independent to each other. Though a node receives a lot of reports, it only observes the latest *w* ones, define the observation window as:

$$\Omega = \{r_{t-w+1}, r_{t-w+2}, r_{t-w+3}, \ldots r_t\}$$

Where r_t is the latest report, *w* is the current width of the node's window. A sample of Ω is r_{t-3} to r_t in Fig. 2. The received reports are raw sensing results; their accuracy is related its source node's sensing accuracy:

$$\partial_n = P(r_i=S \mid n(r_i)=n, S) \quad (2)$$

Where ∂_n is the accuracy of node *n*; *S* is the spectrum status, value 1 means primary user is active and 0 means inactive.

The requirement of users includes P_d and P_f. P_d is the required minimum correct detection rate, and P_f is the required maximum false alarm rate.

Basing these formulations, the detail of Sliding-Window algorithm can be described accurately.

B. Sliding-Window Algorithm Details

In this algorithm, Sliding Window Method is used to get a minimal set of reports, and then apply LRT on it to determine current spectrum status, as shown in (1). In Section III the detail of Sliding-Window method is introduced, and now we focus on how to apply LRT.

TABLE II
RESPONSE TIME

P_d	P_f	Average Response Time of Sliding-Window (sec)		Average Response Time of WSPRT (sec)	
		no mis.	1/5 mis.	no mis.	1/5 mis.
90%	10%	17.34	26.41	54.59	61.32
99%	1%	50.00	56.36	82.97	92.98
99.9%	0.1%	71.45	94.09	102.67	109.18
99.99%	0.01%	97.21	117.54	142.14	150.01

P_d denotes requirement of detection rate and P_f denotes requirement of false alarm rate; "*no mis*" denotes no misbehavior while "*1/5 mis*" means 1/5 nodes are misbehaving

The two thresholds in (1) can be determined by P_d and P_f:

$$\eta_0 = \frac{1-P_d}{1-P_f} \quad \eta_1 = \frac{P_d}{P_f}$$

Proof can be found in Appendix A.

Known the thresholds of LRT, the crucial step is to calculate likelihood ratio $\Lambda(\Omega)$. As introduced in Section III and described in Fig. 3, we do it in a Divide and Conquer way.

Based on the assumption that spectrum status S changes at most 1 time in the period of current window, we can divide current problem into $w+1$ sub problems: change point $CP=0$ *to* w, where $CP=i$ stands for that the change point exists between report r_{t-i} and r_{t-i+1}.

Define the latest known S as S_{old}; and the current S as S_{new}, then:

$S_{new}=1$ is equal to $(S_{old}=1, CP=0)$ *or* $(S_{old}=0, CP=1$ *to* $w)$

$S_{new}=0$ is equal to $(S_{old}=0, CP=0)$ *or* $(S_{old}=1, CP=1$ *to* $w)$

With this knowledge, we can get likelihood of S under Ω:

$$L(S=1|\Omega)=P(\Omega|S=1)=\begin{cases} if(S_{old}=1), & P(\Omega|CP=0) \\ if(S_{old}=0), & P(\Omega|CP=1\cdots w) \end{cases}$$
$$L(S=0|\Omega)=P(\Omega|S=0)=\begin{cases} if(S_{old}=0), & P(\Omega|CP=0) \\ if(S_{old}=1), & P(\Omega|CP=1\cdots w) \end{cases} \quad (3)$$

In which

$$P(\Omega|CP=1\cdots w)=\frac{\sum_{c=1}^{w} P(\Omega|CP=c)P(CP=c)}{\sum_{c=1}^{w} P(CP=c)}$$

Generally assume CP follows a uniform distribution, then:

$$P(\Omega|CP=1\cdots w)=\frac{1}{w}\sum_{c=1}^{w} P(\Omega|CP=c) \quad (4)$$

If $CP=i$, then $S = S_{old}$ when r_{t-w+1} to r_{t-i} are sensed, and $S = S_{new}$ when r_{t-i+1} to r_t are sensed. So:

$$P(\Omega|CP=c)=\prod_{i=t-w+1}^{t-c} P(r_i|S=S_{old})\times\prod_{i=t-c+1}^{t} P(r_i|S=S_{new})$$

In which $P(r_i|S)$ is related to accuracy. By definition of accuracy, we get:

$$P(\Omega|CP=c)=$$
$$\prod_{i=t-w+1}^{t-c}\begin{Bmatrix}\partial_{n(r_i)} & ,if\ \ r_i=S_{old}\\ 1-\partial_{n(r_i)} & ,if\ \ r_i\neq S_{old}\end{Bmatrix}\prod_{i=t-c+1}^{t}\begin{Bmatrix}\partial_{n(r_i)} & ,if\ \ r_i=S_{new}\\ 1-\partial_{n(r_i)} & ,if\ \ r_i\neq S_{new}\end{Bmatrix} \quad (5)$$

Where $n(r_i)$ denotes the source node of report r_i.

Then by (3) to (5) we can finally get likelihood ratio:

$$\Lambda(\Omega)=\frac{L(S=0|\Omega)}{L(S=0|\Omega)}$$

Then with $\Lambda(\Omega)$ LRT can be applied with (1).

Now we know how to use Sliding Window method to control the observation window and how to apply LRT on it to estimate current spectrum. But note that the accuracy of a node is also an unknown parameter; we need to adjust it adaptively.

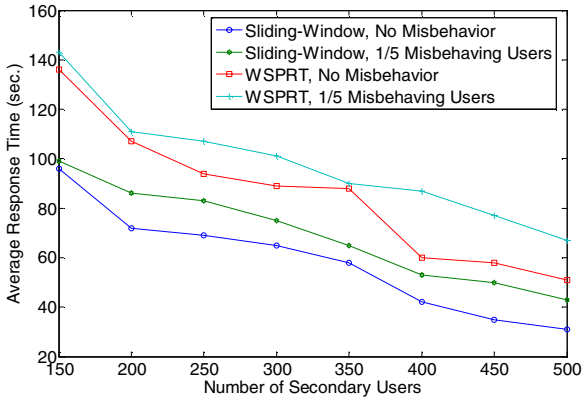

Fig. 6. Average Response Time

C. Accuracy Adjustment

As shown in (5), users' accuracy is needed to apply LRT. Accuracy is defined in (2); it's the probability for a node to report correct sensing result. A node's accuracy is analyzed basing on the history of its reports. When a decision is made, the current spectrum status is known, and the latest sensing report is not outdated, so we can use it to analyze the source node's accuracy. Here we can use Bayesian Estimation:

$$P(\partial_n=x)=\frac{P(r=S|\partial_n=x)P(\partial_n=x)}{\int_x P(r|\partial_n=x)P(\partial_n=x)dx}$$

Where r is the latest sensing result reported by user n. A user's accuracy is set to a default value at the beginning, and then adjusted when the algorithm goes on.

D. Algorithm Description

Now we have solved all the problems needed to work out the Sliding Window algorithm, in this sub section we combine them together and show the whole algorithm progress:

When a report r_t comes:

1. If source node of r_t is a new coming node, set its accuracy ∂_n to be the default value.
2. Add r_t to observation window Ω, width of Ω increases by 1.
3. Calculate the likelihood of $L(\Omega|S=1)$, $L(\Omega|S=0)$ and get their ratio $\Lambda(\Omega)$.
4. If $\Lambda(\Omega)>\eta_1$, decide $S=1$, if $\Lambda(\Omega)<\eta_0$, decide $S=0$, go to step 5, else step 7.
5. Use the latest report r_t and S decided in step 4 to adjust r_t's source node's accuracy $\partial_{n(rt)}$.
6. Try to remove redundant old reports in observation window Ω, if the oldest report is removed and we can still make a decision, remove it.
7. Report the result S to user, end this round and wait for next report.

The computation complexity can be analyzed: for each sensing decision, a LRT is applied. To apply a LRT, (3) is calculated for once; for each (3), (4) is calculated for once; for each (4), (5) is calculated w times; for each (5), ∂ or $(1-\partial)$ is multiplied for w times. As a result, for each decision, the complexity of Sliding-Window algorithm is $O(w^2)$, where w is the width of observation window. This workload is very low and can be accepted by most hardware.

V. EVALUATION

In order to evaluate the performance of our sensing scheme, we present the results from two major aspects. First, we conduct simulations to show the accuracy of our algorithm. Then both theoretical and numerical results are shown to elaborate the aspect of response time.

For the simulation, secondary users are randomly located in a 2000m ×2000m square, each with a transmission range of 250m. Secondary users sense spectrum at a period of 60s. Secondary users are either normal nodes or misbehaving nodes. In our setup, the sensing accuracy of normal nodes is 70% while misbehaving nodes have only 40% accuracy. A primary user operates nearby the secondary network and switches its status between active and inactive every 0.5 to 1 hours. The simulation lasts for a period of 100 hours.

A. Accuracy

At first, we set user's requirement of detection rate to be from 90% to 99.99%, and false alarm rate to be 10% to 0.01%. And for each requirement, we also consider the situation if 1/5 nodes are misbehaving. The results are shown in table 1.

From Table I we can see that Sliding-Window algorithm can always satisfy user's requirement, with P_d ranging from 90% to 99.99%, the detection rate of simulation results is always higher than it; and with P_f ranging from 10% to 0.01%, the false alarm rate is always lower than it. With 1/5 misbehaving users, the sensing is disturbed a little, but even though, the Sliding-Window algorithm can still satisfy user's requirement very well.

Secondly, we study accuracy with different user densities. P_d is fixed at 98% and P_f at 2%, and the number of secondary user spans from 150 to 500, then the actual accuracy is tested. And we also consider the situation when 1/5 users are misbehaving. The results are shown in Fig. 4 and Fig. 5.

From Fig. 4 and Fig. 5 the following facts can be observed: at first, under all densities, actual detection rate is always higher than requirement and false alarm rate is always lower than requirement; the second, with the increase of number of nodes, accuracy is improved; this is because more neighbors around a node give it more reports for making decision; at last, with misbehaving nodes, though the accuracy is a little lower, but it's still good enough for user's requirement.

The simulation results show good performance of the accuracy of Sliding-Window algorithm. Next we analyze the response time.

B. Response Time

Response time means the time difference between the time spectrum status changes and the time secondary nodes detect it. This is very important for reducing interference to primary users.

At first response time is tested under different accuracy requirements. We spans P_d from 90% to 99.99% and P_f from 10% to 0.01%, and the corresponding results are shown in Table II. It's shown that the asynchronous scheme can improve response speed significantly. Especially when requirement is low, response time of Sliding-Window is only about 1/3 of WSPRT. When requirement is very high, Sliding-Window can also response about 30-40 seconds earlier than WSPRT.

Besides the affect of requirement, we also consider the density of secondary users. We fix requirement at P_d=*98%* and P_f=*2%*, and range number of secondary users from 150 to 500. The results are shown in Fig. 6. From the results it's shown that i) Sliding-Window can reduce response time significantly; whenever the users are sparse or dense, response time of Sliding-Window algorithm is always shorter than WSPRT; ii) with the increase of density, average response time is reduced, e.g. in the result of Sliding-Window algorithm without no misbehavior, when number of secondary users increases from 150 to 500, response time reduces from 96s to 31s; and iii) misbehavior slows down response a little, but both two cooperation schemes have the ability to survive in such chaos situations.

VI. CONCLUSION

Accurate spectrum sensing is the key requirement for CRs, and cooperation among nearby users is a good way to improve the sensing accuracy. In this paper we investigate existing works, and proposed a very efficient scheme: Sliding-Window algorithm. It works without synchronization but performs as well as synchronous schemes, and even better, because it can also improve response time to change of spectrums. In this scheme, each node uses Sliding-Window algorithm to analyze the spectrum status basing on the sensing results from its neighbors. This algorithm maintains a minimal and sufficient number of sensing results using Sliding-Window method, and applies Likelihood Ratio Test (LRT) on them to make a decision. This algorithm enables cooperative sensing in more various CR networks because its ease to implement, and also provides references for applying sequential analysis in other similar problems.

REFERENCES

[1] K. Challapali, S. Mangold and Z. Zhong, "Spectrum agile radio: Detecting spectrum opportunities", *Proc. 6th Annual Int'l Symposium on Advanced Radio Technologies*, March 2004.

[2] M. P. Olivieri, G. Barnett, A. Lackpour, A. Davis, and P. Ngo, "A scalable dynamic spectrum allocation system with interference mitigation for teams of spectrally agile software defined radios," *Proc. DySPAN*, Nov. 2005.

[3] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio networks," *Proc. DySPAN*, Nov. 2005.

[4] S. M. Mishra, A. Sahai, and R. Brodersen, "Cooperative sensing among cognitive radios," *Proc. 2006 IEEE International Conference on Communica-tions (ICC)*, June 2006.

[5] Yuan Yuan, Paramvir Bahl, Ranveer Chandra, et al. "KNOWS: Kognitiv Networking Over White Spaces," *Proc. DySPAN*, 2007

[6] R. Chen, J. M. Park, K. Bian, "Robust Distributed Spectrum Sensing in Cognitive Radio Networks," Technical Report TR-ECE-06-07, Dept. of Elec-trical and Computer Engineering, Virginia Polytechnic Institute and State University, July 2006.

[7] G. Lorden, "Procedures for reacting to a change in distribution," The *Annals of Mathematical Statics*, Vol. 42, No. 6, page 1897-1908, 1971.

[8] Daniel Barry. JA Hartigan, "A Bayesian Analysis for Change Point Problems," *Journal of the American Statistical Association*, Vol. 88, No. 421, page 309-319, Mar., 1993.

[9] Y. Mei, "Sequential change-point detection when unknown parameters are present in the pre-change distribution," *The Annals of Statistics*, vol. 34, no. 1, page 92-122, 2006.

[10] Robert W. Broderson, Adam Wolisz, Danijela Cabric, Shridhar Mubaraq Mishra, and Daniel Willkomm. White paper: CORVUS: A Cognitive Radio Approach for Usage of Virtual Unlicensed Spectrum. Technical report, 2004.