Efficient Data Capturing for Network Forensics in Cognitive Radio Networks

Shaxun Chen[†], Kai Zeng[‡], Prasant Mohapatra[†]
[†]Department of Computer Science, University of California, Davis, CA 95616
[‡]Department of Computer and Information Science, University of Michigan - Dearborn, Dearborn, MI 48128
sxch@ucdavis.edu, kzeng@umich.edu, pmohapatra@ucdavis.edu

Abstract— **Network forensics is an emerging interdiscipline used to track down cyber crimes and detect network anomalies for a multitude of applications. Efficient capture of data is the basis of network forensics. Compared to traditional networks, data capture faces significant challenges in cognitive radio networks. In traditional wireless networks, usually one monitor is assigned to one channel for traffic capture. This approach will incur very high cost in cognitive radio networks because it typically has a large number of channels. Furthermore, due to the uncertainty of the primary user's behavior, cognitive radio devices change their operating channels dynamically, which makes data capturing more difficult. In this paper, we propose a systematic method to capture data in cognitive radio networks with a small number of monitors. We utilize incremental support vector regression to predict packet arrival time and intelligently switch monitors between channels. We also propose a protocol which schedules multiple monitors to perform channel scanning and packet capturing in an efficient manner. Monitors are reused in the time domain and geographic coverage is taken into account. The real-world experiments and simulations show that our method is able to achieve the packet capture rate above 70% using a small number of monitors, which outperforms the random scheme by 200%-300%.**

Keywords— **Network forensics, cognitive radio network, efficient data capture**

I. Introduction

Network forensics is a discipline that monitors and analyzes network traffic, aiming at detecting malicious network activities and preserving network data as evidences. It is widely used in preventing attacks, tracing down criminals, diagnosing the network, etc. Although it is a newly emerged research area, network forensics attracts a great attention from both network researchers and law enforcement practitioners.

Network forensics is composed of two steps: data capture and data analysis. Data capture is the basis of network forensics, on which the quality of data analysis largely depends. However, data capture is not as easy as it seems, especially in wireless networks. Channel fading, signal coverage and interference, mobility of transceivers and ever-increasing data rate make data capture a non-trivial task.

Existing work on network forensics as well as data capture is studied in the context of traditional networks. As an emerging type of network, cognitive radio network is a promising technology to mitigate the scarcity of wireless spectrum. However, in cognitive radio networks, data capture faces additional challenges.

First, according to the Federal Communications Commission (FCC)'s regulation, unlicensed users should evacuate immediately when an incumbent user appears, which means unlicensed users may frequently change their working channels. Second, cognitive radio networks have a much wider spectrum than other wireless networks. For example, the white space is from 50MHz to 700MHz approximately. Assuming a channel width of 6MHz (the same as the TV channel-width in U.S.), there can be about one hundred available channels. If we capture the traffic of cognitive radio networks in the traditional way, one monitor should be tuned to listen to one channel. That is, we will need one hundred monitors, making the cost prohibitively high.

In this paper, we propose a novel method, which intelligently switches monitors between different channels to capture the network data spread over a large number of channels with a small number of monitors. Our method is based on two observations. First, although a cognitive radio network may have a large number of channels, not all of them are busy at the same time. Second, for a single channel, there are always intervals between packets. Furthermore, a network forensics system usually does not require all the packets on the channel; instead, they capture a certain subset of the packets based on their specific needs.

Based on these observations, we propose to predict the arrival time of the next *packet of interest* (PoI) by using incremental support vector regression, and then switch monitors between different channels according to our prediction approach. To the best of our knowledge, this is the first work investigating data capture for network forensics in cognitive radio networks, and also the first effort to monitor multiple channels with fewer monitors by exploiting the prediction of packet arrival time.

We conduct extensive experiments and simulations. The results demonstrate that given a large number of channels, we can achieve a high packet capture rate with a small number of monitors. Our method outperforms the random scheme by 200%-300%.

The rest of this paper is organized as follows. Section II discusses related work. Section III states the problem, and Section IV introduces our method for packet arrival time prediction. In Section V, we present the protocol for efficient data capture in cognitive radio networks. Section VI evaluates our work. Section VII discuses several related technical issues and Section VIII concludes the paper.

II. Related Work

Data capture techniques for network forensics can be categorized into two types: *catch-it-as-you-can* and *stop-and-*

listen [1]. The former tries to capture everything on the channel and requires larger amounts of storage; the latter selectively captures packets and puts higher pressure on the CPU performance. Wireshark, WinPcap, TCPdump, etc. are the tools commonly used for data capture. These tool fall into the first category, but they are also capable of capturing certain packets based on predefined filters.

Efforts have been made to effectively capture packets in high speed networks [2] [3]. Siles studied the performance related issues and encryption-overcoming of data capturing in wireless networks [4]. Geiger et al. [5] pointed out that a monitor can capture the packets from adjacent channels in WLAN when the data rate is low.

In almost all the existing studies, including the work mentioned above, one monitor is used for capturing data on one channel (or link). Choong proposed to use a single software defined radio device to sample multiple channels in ZigBee networks [6]. This feasibility is based on the fact that the maximum channel width of the software defined radio can cover multiple ZigBee channels. However, their approach only works when the modulation rate of the channels being sampled is very low (250kbps). It is acceptable for ZigBee networks, but far from practical for general data capturing. When the modulation rate goes higher, the sampling and computation overhead quickly exceeds the hardware processing ability. Besides, the channel width of ZigBee is only 2MHz. For other types of wireless networks, even a software defined radio device cannot cover many channels at the same time. In contrast, our method reuses monitors in the time domain, therefore is not constrained by the modulation rate or the bandwidth of the channel being watched.

Chhetri et al. proposed to schedule sniffers among multiple channels [19], but the goal is to monitor the appearance of wireless users, which is much easier compared to traffic capturing. Moreover, in their work, a pre-known transmission probability is assumed for each user; the sniffers are scheduled without considering the real time user behavior.

Arora et al. employed multi-armed bandit to formalize the multi-channel monitoring problem [20]. Similar as [19], this method is only good for transient activities and cannot be used to capture packets. Specifically, a slot system is assumed in [20], but they do not care the length of the slot. It is possible that during a slot there come multiple packets or a packet lasts across multiple slots. Besides, the channel switching overhead is not considered in this work.

Time series prediction has been well studied for decades. Autoregressive moving average models and Kalman filter are most widely used, but they both require that the process being predicted is linear and stationary. Machine learning techniques, such as support vector machine and neural network, do not have such restrictions [7], but they are usually not fast enough for online prediction. Besides, these works are dedicated to predict continuous values; they may experience large errors when predicting a binary variable (in our case, appearance or disappearance of *packets of interest*).

Phit et al. proposed to predict packet arrival time using neural networks [8]. Historical data of packet inter-arrival time are used as the input. However, this method is only suitable for offline analysis, because the training phase takes considerably long time.

A preliminary version of this work is reported in [21]. In this paper, we further consider the geographic coverage of wireless signals, so that our method can be easily applied to practical scenarios. The channels being watched are not required to be co-located, and monitors are scheduled according to their workload as well as their locations and listening range.

III. Problem Definition

A. Background

In June 2009, the FCC released the analogue TV broadcasting channels (54MHz - 698MHz, often referred to as white space). Unlicensed (secondary) users are allowed to access the spectrum opportunistically, but they must evacuate immediately upon incumbent (primary) users' presence.

Most important primary users in white space are TV towers and wireless microphones. They typically transmit analogue signals. In practice, network forensics systems are interested in capturing data packets (of secondary users) instead of analogue signals from primary users.

IEEE 802.22 and 802.11af proposed by FCC are introduced as "super WiFi" which operates as secondary users in white space; individual users are allowed to access white space freely as long as their devices meet certain interference requirements. Monitoring and capturing data in such a network faces new challenges because of the large number of channels in white space and very dynamic accessing behaviors of white space users.

An alternative to capturing packets wirelessly is to physically connect to the base station or wired infrastructure of cognitive radio networks. However, in some wireless communication systems, such infrastructure may not exist; even if it exists, forensics systems may not have the access. In addition, information like channel quality and signal strength, which can be used to infer users' location and mobility pattern, will be lost if data are only captured in the wired side. Wireless forensics is different from wired forensics in both methods and applicable scenarios. In this paper, we focus on wireless data capture in cognitive radio networks.

B. Problem Definition

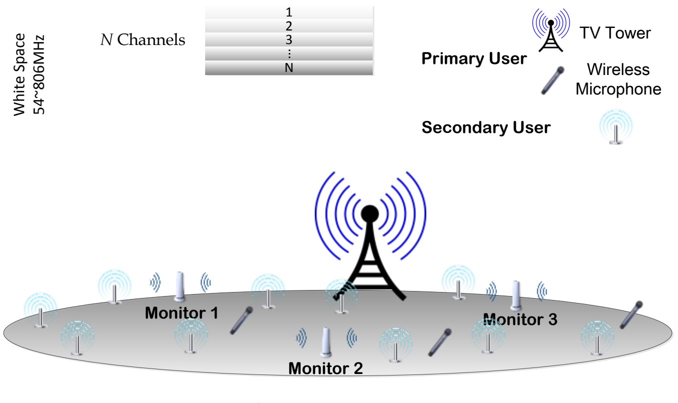

Figure 1. Overview of data capture in a cognitive radio network.

Assume that there are N channels in a cognitive radio network. We have only M monitors ($M<<N$). Among N channels, L are *busy* ($M<L<N$). Here *busy* channel only refers to the channel occupied by secondary users. Since secondary users in a cognitive radio network may not be located very closely, it is not required that every monitor can hear all the secondary users. Instead, we only assume that any secondary user who transmits *packet of interest* (which will be explained later) is within the listening range of at least one monitor. Figure 1 shows a cognitive radio network with primary users, secondary users, and three monitors.

During any period of time, new secondary users may join the network (idle channels get occupied); existing secondary users may quit (*busy* channels become idle); they can also switch to a new channel and continue communicating (due to appearance of primary signals, change of channel quality, or requirement of certain network protocol). These changes are, if not impossible, very difficult to predict (see discussion in Section VII).

The goal of our work is to capture as many *packets of interest* as possible from these *busy* channels.

Packets of interest (PoIs) are the packets that a network forensics system wants to capture for future analysis. Whether a packet is of interest or not depends largely on the purpose of the forensics system. Different systems (applications) may have very different interest. For example, a forensics system which monitors video streaming traffic may find I-frames more of interest than P-frames and B-frames, because I-frames can be decoded independently, and usually contain more fundamental information of the video. Another forensics system for network anomalies detection may want to capture ICMP packets instead of normal IP packets, since ICMP packets tend to relate to malicious or suspicious network activities [9]. To define and decide PoI is out of the scope of this paper. We assume networks forensics systems know what types of packets they need to capture.

In order to reduce the requirement of the number of monitors (or in other words, to capture more PoIs with a limited number of monitors), we propose to switch monitors between channels by predicting the arrival time of PoIs in each *busy* channel. We assume our monitors have the same ability (radio-wise) as the transceivers in the cognitive radio networks, and all the monitors are connected by dedicated channels.

The key idea of our method is to reuse monitors in the time domain. The main challenges are listed as follows.

1) Online prediction. Our method requires that the prediction of PoI should be performed on the fly, which calls for a very fast prediction algorithm. The sequence of packet inter-arrival time is not inherently a linear process, which makes traditional moving average models not appropriate. On the other hand, machine learning based methods usually take too much time for training, hence are not efficient enough for online prediction.

2) Overall optimization. The optimization problem of our method is not as straightforward as it seems. Conservative strategies tend to stay in a channel for longer time, while aggressive strategies tend to switch more often. The tradeoff is tricky because failing to capture a PoI not only means the loss of forensics data, but also hurts the accuracy of future packet predictions.

3) Data capture and channel scan. Our monitors have dual duties. In addition to packet capture, they are also responsible for scanning the channels in order to find *busy* ones. How to schedule the monitors to perform both tasks is also challenging.

We will provide solutions to these challenges in Section IV and V.

IV. Packet Arrival Time Prediction

In this section, we introduce our method for arrival time prediction of PoIs. We present the support vector regression in the first subsection and then, in the second subsection, we improve its performance for the online prediction.

A. Support Vector Regression for Packet Arrival Prediction

We propose to switch monitors between channels in order to capture more PoIs. Ideally, we want a monitor to stay in the channel when there is a PoI, and switch to other channels when there is not. Good switching strategy requires a good prediction algorithm to tell us when a PoI is likely to arrive.

Now we introduce our packet arrival time prediction method using support vector regression. As mentioned previously, traditional methods, such as autoregressive moving average and Kalman filter, are only applicable to linear processes. Among machine learning based methods, support vector machine / regression is often reported to have superior performance [7] [18].

The input of our algorithm is (a_0, a_1, a_2, …, a_n), which are the arrival time of n successive PoIs in a given channel. The output is a_{n+1}, the estimated arrival time of the next PoI in this channel.

In a nutshell, support vector machine is a classification tool. In the training phase, it tries to divide different groups of samples apart by a hyperplane (or a set of hyperplanes), which is carefully constructed and lies in the “middle” of the margin between groups. Support vector regression works similarly. The difference lies in that the hyperplane is built to approximate all the samples. An error ε is allowed in the approximation. That is, the distance from any sample to the hyperplane is less than ε.

Formally, the hyperplane (i.e. regression function) can be expressed as:

$$f(X) = W \bullet X + b \tag{1}$$

Where W and X are both n-dimensional vectors, b is a real number. X is the attributes of samples. In our case, X = (x_1, x_2, x_3, …, x_n), where $x_i = a_i - a_{i-1}$ (the time intervals between consecutive PoIs). The dot between W and X is inner product. W determines the slope of the hyperplane.

As mentioned, all the samples should be within a distance ε to the hyperplane $f(X)$. Apparently, there can be many hyperplanes satisfying this requirement. Support vector regression looks for the one lying in the “middle” of the region where the samples spread (referred to as *flatness*). This requirement is equivalent to minimizing $||W||$.

Formally, the problem can be described as minimizing $||W||^2/2$, subject to:

$$\begin{cases} y_j - W \bullet X_j - b \le \varepsilon \\ W \bullet X_j + b - y_j \le \varepsilon \end{cases}$$

where X_j is the attributes of the j^{th} training sample, and y_i is a_{n+1} of this sample. Minimizing $||W||^2/2$ equals minimizing $||W||$. We use the former for mathematical convenience.

Up to now, we assume such hyperplane $f(X)$ exists. However, sometimes it may not be true due to small ε and dispersed distribution of the training samples. To ensure the existence of $f(X)$, we allow some samples to have larger errors than ε, which is comparable to the *soft margin* in support vector machine. The problem can be formalized as:

$$\text{minimize} \quad \frac{1}{2}\|W\|^2 + C\sum_{j=1}^{l}(\xi_j + \xi_j')$$

$$\text{subject to} \quad \begin{cases} y_j - W \bullet X_j - b \le \varepsilon + \xi \\ W \bullet X_j + b - y_j \le \varepsilon + \xi' \end{cases}$$

where ξ and ξ' are nonnegative values accounting for extra errors (as shown in Figure 2, they introduce a penalty while ε does not), and C is a positive constant, which decides the tradeoff between the *flatness* of the hyperplane and the amount of extra errors. l is the number of training samples.

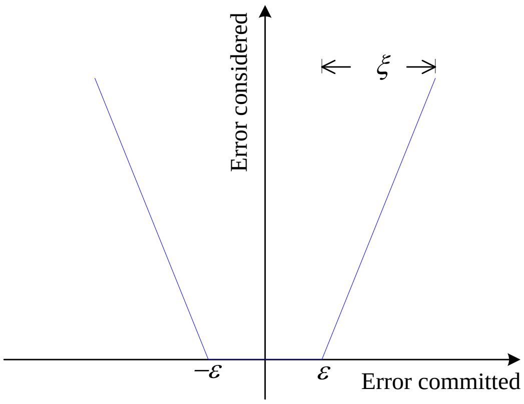

Figure 2. Error function.

The above objective function and constrains is equal to minimizing L, which is called Lagrange function:

$$L = \frac{1}{2}\|W\|^2 + C\sum_{j=1}^{l}(\xi_j + \xi_j') - \sum_{j=1}^{l}(\eta_j\xi_j + \eta_j'\xi_j')$$
$$-\sum_{j=1}^{l}\alpha_j(\varepsilon + \xi_j - y_j + W \bullet X_j + b)$$
$$-\sum_{j=1}^{l}\alpha_j'(\varepsilon + \xi_j' + y_j - W \bullet X_j - b)$$

where α, α', η and η' are Lagrange multipliers and they are all positive. Minimizing a Lagrangian can be converted to a solvable dual optimization problem. Due to the space limitation, we do not present the detailed derivation here. Finally, the hyperplane can be expressed as:

$$f(X) = \sum_{j=1}^{l}(\alpha_j - \alpha_j')X_j \bullet X + b \qquad (2)$$

Here X_j is the attributes of training sample j. In the above equation, b can be calculated by exploiting Karush-Kuhn-Tucker conditions. Details can be referred to [10]. Now we get the hyperplane $f(X)$.

In the above discussions, hyperplanes are used to approximate samples. Since packet arrival time is not a linear process, using hypersurface could increase the performance. Therefore, we introduce the kernel tricks. It can be proven that the property of support vector regression still holds if we substitute the inner product in Equation 2 with kernel functions. In practice, we employ Gaussian radial basis function, which is one of the most commonly used kernel functions. It is defined as:

$$k(\omega_i, \omega_j) = \exp(-\gamma \|\omega_i - \omega_j\|^2)$$

where γ is a positive parameter. We use $1/2\delta^2$ for γ. The updated regression function is:

$$f(X) = \sum_{j=1}^{l}(\alpha_j - \alpha_j')\exp(-\frac{1}{2\delta^2}\|X_j - X\|^2) + b \qquad (3)$$

In the training phase, assuming we have recorded the arrival time of m ($m > n$+1) PoIs: a_0, a_1, a_2, …, a_{m-1}, we first calculate the time interval between them, noted as x_1, x_2, x_3, …, x_{m-1}. In the training phase, these m-1 items are organized into m-n-1 samples, i.e. (x_1, x_2, …, x_{n+1}), (x_2, x_3, …, x_{n+2}), …, (x_{m-n-1}, x_{m-n}, …, x_{m-1}). For each sample, first n elements are attributes and the last element is the label (y_j). After training, we determine the parameters of the regression function f.

In the prediction phase, n most recent intervals between arrived PoIs are used as input to predict the next one. If we want to predict the arrival time of the k^{th} ($k > m$-1) PoI (a_k), then let X = (x_{k-n}, x_{k-n-1}, …, x_{k-1}), we have $x_k = f(X)$, and $a_k = a_{k-1} + x_k$.

In Section VI, we will evaluate the accuracy of this algorithm with different training dataset size (l) and different number of attributes (n). We will also show that in a single channel, if PoIs can be divided into multiple categories, it is better to predict them separately.

B. *Expediting Learning Process*

For support vector regression based algorithms, prediction is fairly fast while training phase usually takes more time.

In this subsection, we propose several approaches to reduce the training time, which is especially important to our method that performs online prediction.

First, we employ incremental learning for the training of support vector regression, which enables us to dynamically add or remove a sample from the training dataset without learning from scratch [11] [12]. The mathematica explanation of incremental learning is complex; the main idea is described as follows.

It can be derived that for most samples, α_j = α'_j in Equation 2. That is, the regression function only depends on a small number of samples, which lie in the “fringe” of the sample space. These samples are called *support vectors*. In the incremental learning, when a new sample comes, it checks if it is a *support vector*. If not, the training result remains unchanged. Otherwise, it is added into the support vector set and the parameters in the regression function are re-tweaked. It works similarly when a sample is removed.

Although the regression function can be used to predict repeatedly once it is trained, the prediction will become less and less accurate as time passes, because the training data gets obsolete and the traffic pattern changes over time. Traditionally, without incremental learning, training frequently is not affordable for online predictions due to its high computational overhead. However, with incremental learning, we are now able to update our regression function in a timely fashion.

Second, we use dual regression functions to reduce retraining. As introduced in Section IVA, in order to predict the k^{th} PoI (a_k), we need the arrival time of n+1 PoIs just before it (a_{k-n-1}, …, a_{k-1}). If we fail to capture a PoI (say, k-1^{th} packet), n following packets cannot be predicted (from k^{th} to k+n-1^{th}), because the input needed by the regression function is incomplete. In this case, we have to stay in this channel to capture these n packets, giving up the opportunities of capturing packets in other channels, which is a non-negligible performance loss. Moreover, we probably have to retrain the regression function, because without the prediction results, we cannot compare them with the ground truth, and tell whether the regression function is obsolete or not.

In order to alleviate such performance degradation, we introduce dual regression functions (f and f'). The former predicts the arrival time of the next packet and the latter predicts the one after next.

$$(x_1, x_2, ..., x_n) \xrightarrow{f} x_{n+1}$$

$$(x_1, x_2, ..., x_n) \xrightarrow{f'} x_{n+1} + x_{n+2}$$

f' is defined similarly as f, and uses the same model we presented in Section IVA. The only difference is that f' predicts two packets ahead. Of course, the training data of f' are in the form of (x_1, x_2, …, x_n, x_{n+1}+ x_{n+2}), in which the first n items are attributes and the last is the label.

We maintain f and f' simultaneously. If a PoI is missed (say, k-1^{th}, caused by mis-prediction of f or monitor shortage), we utilize f' to predict the k^{th} PoI. If it is a match, the process goes on as normal. No retraining is needed and a_{k-1} (predicted value) is used as the ground truth for the next few predictions. On the other hand, if the prediction of f' still does not match, the monitor will keep staying at this channel for at least n PoIs' duration and then perform an incremental retraining.

In this updated version of method, two consecutive mis-predictions (k-1^{th} and k^{th}) suggest the obsoleteness of the regression function f (as well as f'). In contrast, the old method has to stick on a channel for quite a while upon a single miss, which may occur frequently and does not necessarily imply the invalidation of the regression function. Therefore, the updated method reduces a large amount of retraining and the time stuck in a single channel. Of course, maintaining f' itself introduces overhead. However, this overhead is not high with incremental learning, and it is worthwhile because being stuck on a channel may cause loss of packets in other channels and thus a vicious cycle.

An alternative method for dual regression functions is to treat the sequence of PoI arrival time as discrete time series. We can still use the model presented in Section IVA to perform prediction. However, in this case, the training samples are in the form of (j, a_j), where j is the single attribute (sequence number) and a_j is the label (arrival time of the j^{th} PoI). The advantage of this method is that it is able to predict multiple future packets with a single regression function. However, compared with the regression function we use, it requires much more training samples to achieve decent accuracy. We will compare their performance in Section VI.

Besides two modifications discussed above, we also apply some tricks to further expedite our method. We store the values of the kernel (Gaussian radial basis function) in a matrix, thus avoid computing every time during the training. Besides, the regression function is traditionally trained using various ε and ξ, and then the value with the best accuracy is adopted. However, this process is very time-consuming. We fix the values of ε and ξ at $\tau/40$ and $\tau/20$ respectively (τ is the average inter-arrival time of the recent PoIs), which largely reduces the computation time without obvious decreasing of prediction accuracy. We will show the results in Section VI.

V. Monitor Mutiple Channels with A Small Number of Monitors

In the previous section, we present our method for packet arrival time prediction. Multiple efforts are made to accelerate the algorithm and make it qualified for online use. In this section, we first introduce the monitor scheduling method based on the prediction results, and then present the complete protocol for data capturing in cognitive radio networks.

A. Monitor Scheduling

Packet arrival prediction is independent for each channel. Based on these predictions, a limited number of monitors are scheduled to cover a large number of channels. In this subsection, we assume that we already have the prediction results.

Figure 3 shows an example of three channels. Each square is a PoI, and we assume there are two monitors, originally residing at channel A and C.

In order to capture all the PoIs, a valid scheduling is that the first monitor catches A1, B1, A2, A3, A4 and B4, while the second captures C1, C2, B2, B3, C3 and C4. Of course, there is more than one valid scheduling scheme. Among them, the one with minimum channel switches is preferred. The reason is as follows.

First, channel switching has overhead. Although switching under the monitor mode is faster than other modes, it still needs some time. Taking 802.11bg wireless cards for example, channel switching takes 3-20ms [13]. The more a monitor switches, the less time it can spend on data capturing.

Second, continuously staying in a channel for longer time helps verify the prediction algorithm. Prediction results are compared with the ground truth to decide whether retraining is necessary. Frequent channel switching impedes the gathering of the ground truth.

For the example in Figure 3, the solution mentioned above is the optimum in this sense, which only has 5 switches (shown as arrows in Figure 3). However, in a general case, finding a scheduling scheme that minimizes the number of monitor switches is not easy (when the number of monitors is less than the number of channels). The arrival time of the future PoIs is not deterministic. The prediction algorithm cannot forecast very far ahead, and prediction errors are inevitable. Therefore, it is not feasible to establish an algorithm that always gives the optimal solution.

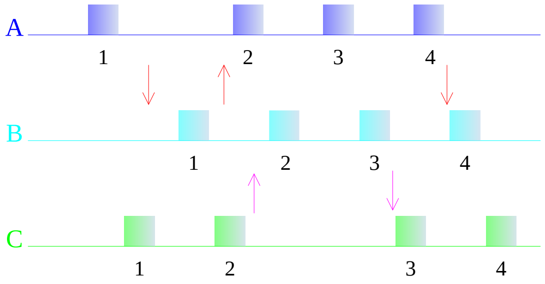

Figure 3. Monitor Scheduling.

Instead, we propose a greedy method to schedule the monitors with relatively few channel switches. We choose a greedy algorithm because the packet arrival prediction can only forecast the near future, which has a myopic nature. The method we use is explained as follows. If a PoI will arrive within *v* ms by prediction and no monitor is now in this channel, a scheduling activity is triggered. Among all the *available* monitors, the one that currently has the longest "free interval" is selected and switched to capture this packet. The monitor will stay in this channel until being scheduled and switched again. The algorithm is shown as follows.

```
An upcoming packet in channel $i$ triggers scheduling
    latestNext = 0; monitorSel = -1;
    for any monitor $j \in AM_i$
        if ($a_N^{ch(j)}$ > latestNext)
            lastestNext = $a_N^{ch(j)}$;
            monitorSel = $j$;
    if (monitorSel != -1)
        switch monitor monitorSel to channel $i$
        delete monitorSel from $AM_i$
    else return false
```

Algorithm 1. Monitor scheduling

Here, AM_i is the set of *available* monitors, ch(*j*) is the current channel that monitor *j* residents, and $a_N^{ch(j)}$ is the predicted arrival time of the next PoI on channel ch(*j*). *Available* monitors are defined as follows.

Three types of monitors are *busy*. First, if a PoI will arrive within *w* ms by prediction, the monitor currently on this channel is set to *busy* until this packet is captured or timeout. The second type is the monitors being occupied in a retrain process triggered by two successive mis-predictions (please refer to Section IVB). Third, a few monitors are used for channel scanning (will be discussed in Section VB). All other monitors are *available*.

The above description is correct if we do not consider signal's geographic coverage. In reality, the signal sources being watched are not necessarily co-located. That is, at a given time, one monitor may only hear a subset of channels. Therefore, strictly speaking, AM_i is the set of monitors which are able to hear channel *i* at that moment, excluding the three types of *busy* monitors mentioned above. It is easy to see that the larger the *AM* is, the more flexibility the algorithm gets, and the better performance we can expect. This suggests, if possible, we should make the monitors cover the channels (secondary users) evenly. Let $U_i = |AM_i|$ (the number of monitors in AM_i) and u = min U_i for all *i* where channel *i* are *busy* (*busy* channels are defined in Section IIIB); we can use *u* to indicate the coverage evenness. Since AM_i changes over time (different secondary users access channel *i*) and so does *u*, an optimum monitor placement may not achievable (if monitors are not mobile). However, a decent placement is good enough for our method.

Algorithm 1 is linear and fast enough for online scheduling. The greedy strategy it uses is a good approximation of minimizing switches in practice. For the example shown in Figure 3, the scheduling performed by this algorithm is the same as the optimum. *w* and *v* mentioned above will be defined in the next subsection.

B. Protocol for Data Capture in Cognitve Radio Networks

We have discussed the packet arrival prediction and monitor scheduling algorithm in the above sections. In this subsection, we first present our method for channel scan, which detects channels for secondary signals, and then present the complete version of the data capturing method in cognitive radio networks.

In Section VA, we introduced our algorithm that switches monitors between channels. We assumed these channels are all *busy*. However, in a cognitive radio network, only some of the channels are occupied by secondary users (referred to as *active* channels; we do not capture primary users' traffic, because they transmit analogue signal; see Section III). The rest of them are used by primary users, experiencing low channel quality or simply idle (referred to as *inactive* channels). Leaving monitors staying in *inactive* channels is a big waste. We should find out *active* channels before applying packet prediction and monitor scheduling algorithms.

Before going into the details, we define and recall some notations. The cognitive radio network has *N* channels and we have *M* monitors. The number of monitors that can hear channel *i* is U_i. Algorithm 1 is executed *v* ms before a packet arrives, and a monitor is set as *busy w* ms ahead of packet arrival (see Section VA). t_r is the time relax of packet arrival prediction. That is, for any predicted arrival time *a*, we schedule the time slot $[a\text{-}t_r, a+t_r]$ for packet capture. If a PoI is captured in this time slot, it is called a match. Otherwise, a mis-prediction is assumed. t_s stands for the time overhead for channel switching. *l* is the number of samples needed for the first-time training in a new channel.

At any given time, *S* monitors are used for channel scanning (we choose $S = \lceil M/6 \rceil$ in our method, which is determined by experiments; see Section VIC). It is a tradeoff between the number of monitors wasted (in terms of data capturing) and the detection delay of secondary users. The scanning monitors are chosen carefully considering their listening coverage. In Figure 1, for example, assuming there are 8 monitors in total, we will employ $\lceil 8/6 \rceil = 2$ monitors dedicatedly for scanning; choosing monitor 1 and 3 is better than using 1 and 2 because the latter is less likely to cover the range of all secondary users (If $\lceil M/6 \rceil$ monitors cannot cover the range by any means, we will increase the number of scanning monitors by 1 until all covered; it is reasonable that a sparse network needs more monitors). Each scanning monitor iterates all the *inactive* channels sequentially and repeatedly. They report the emergence of secondary users.

In addition, for any other monitor, if it successfully captures a PoI in the first half of the scheduled slot ([a-t_r, a]), it quickly switches to an *inactive* channel to search for secondary signals. The channel that has not been scanned for the longest time will be chosen. This operation is transparent to the monitor scheduling algorithm. The reason for doing this is that we want to make full use of the scheduled slot, and help those dedicated monitors to accelerate the discovery of new secondary users.

In case of the disappearance of secondary signals, detection is easier. After two mis-predictions, a monitor will be scheduled to stay in this channel and perform retraining. Absence of the secondary signal will then be found. No extra efforts are needed.

Now we briefly describe the protocol of our method for data capturing in cognitive radio networks.

1) Monitors scan the *inactive* channels in the manner as above. Once a new secondary signal is detected, this channel is marked as *active*. At the same time, an *available* monitor is switched to this channel to perform training, and removed from *AM*.

2) After collecting l PoIs, the initial training is completed. The monitor is set back to *available* state unless the next PoI will arrive within w ms.

3) After training, future PoIs are predicted by f and f'in each *active* channel. v ms before the next packet arrival, Algorithm 1 is executed to pick a monitor from *AM* to capture it if no monitor is currently in the channel. Otherwise, the monitor in this channel is set to *busy* w ms before the arrival unitl the packet is captured.

4) If two consecutive mis-predictions occurs in an *active* channel, an *available* monitor is assigned to this channel and perform incremental retraining. This monitor is removed from *AM* until retraining is done.

5) Once Algorithm 1 returns *false* (no more available monitors), our method enters *saturated* mode and stops marking a channel as *active* even if a secondary signal is found. *Saturated* mode ends when an existing secondary user quits from an *active* channel.

6) Under the *saturated* mode, if Algorithm 1 returns *false* with the ratio higher than a threshold, an *active* channel is marked as *inactive*, which means we give up data capturing in this channel temporarily. By default, remarking process starts from the *active* channel with minimum number of PoIs per unit time.

In our method, as mentioned in Section III, all the monitors are connected by dedicated channels and their clocks are synchronized. This assumption is reasonable, since monitor array products are widely available in the market (but the number of monitors in the array is limited). Communications between monitors (and the controller) are fast and knowledge is shared.

In the above protocol, one or more channels are temporarily relinquished when monitor shortage occurs. We use this conservative strategy because recklessly covering more channels will cause more retraining, less available monitors, and thus a vicious circle. In fact, if some channels are given up in step *6)*, it suggests that, even using our method, the total number of monitors is still too small to effectively capture the traffic in the current network.

When the network is very sparse, our method could experience low efficiency. First, the *AM* set will be relatively small. Second, if secondary users are very unevenly distributed, the entering and quitting of the *saturated* mode may not reflect the global situation of the network (step *5)*). However, a sparse network is not the typical application scenario of our method. An extreme case is that within one monitor's listening range, there is only one link, where the time domain reuse is not applicable and there is no better way to save monitors in such cases.

Some parameters in the protocol have certain constrains. v should be larger than (t_s + t_r), as well as w. The reason for the former is straightforward. For the latter, if the monitor in the current channel is marked as *available* and switched to another channel, the remaining time should be long enough for other monitors to switch to this channel and catch the next packet. Besides, t_r is larger than t_s, which helps maintain the transparency when ordinary monitors are opportunistically used for channel scan. Concrete value of the parameters will be assigned in the next section.

VI. Evaluations

We conduct comprehensive experiments and simulations to evaluate our method for data capturing in cognitive radio networks. We first test the accuracy of the packet arrival prediction method under various traffics, and then evaluate the performance of monitor scheduling algorithm. After that, the effectiveness and overall performance of the complete method are evaluated.

A. *Performance of Packet Arrival Time Prediction*

In this subsection, we evaluate the performance of our method for packet arrival time prediction. As discussed in Section IV, a support vector regression based model is built upon training. The arrival time of n+1 latest packets are used to predict the arrival time of the next packet.

We first test the influence of different types of traffics on our prediction method. Three types of trace data (FTP, VoIP, and web browsing traffic) are collected from real-world scenarios. In the FTP and VoIP traces, we assume all packets are of interest. For web browsing, we test two cases: all packets are of interest and only ICMP packets in the trace are of interest.

The results are shown in Figure 4. The y-axis is the relative estimation error of predicted arrival time, which is defined as |real – estimated| / τ, where τ is average inter-arrival time of PoIs. The errors of 120 predictions are averaged for each point. If there are averagely 50 PoIs per second, relative estimation error is 10% means that the prediction error is 2ms in average. The x-axis shows the number of attributes (n) used as input of the regression function. In this experiment, all regression functions are trained by 100 recent PoIs.

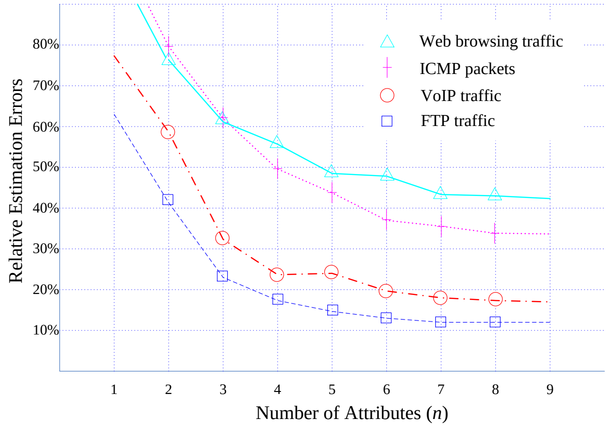

Figure 4. Accuracy of packet arrival time prediction.

From the result we can see that our prediction method has higher accuracy on FTP and VoIP traffic than web browsing and ICMP. This is reasonable because FTP and VoIP traffic tend to be more regular and have less randomness. Even the case of ICMP performs better than the web browsing traffic from which the former is extracted. This result suggests that it is better to categorize packets before prediction for hybrid traffics. We will further investigate it soon.

When n gets larger, the prediction becomes more accurate. But large n also has drawbacks, in that a monitor has to stay in the channel waiting for (n+1) PoIs if two consecutive mispredictions occur. The larger n, the longer it waits. For our method, we choose n = 6, since the performance gain quickly shrinks when n > 5. All the following experiments use this value unless otherwise specified.

In the following experiment, we compare our prediction method with two other strategies. Strategy A tests various ε and ξ, and then chooses the best for the regression function. The rest of its settings are the same as our method. Strategy B treats the packet sequence as discrete time series, which can predict far ahead with current knowledge (please refer to Section IVB). We use web browsing traffic for this test, and only ICMP packets are of interest. The results are plotted in Figure 5.

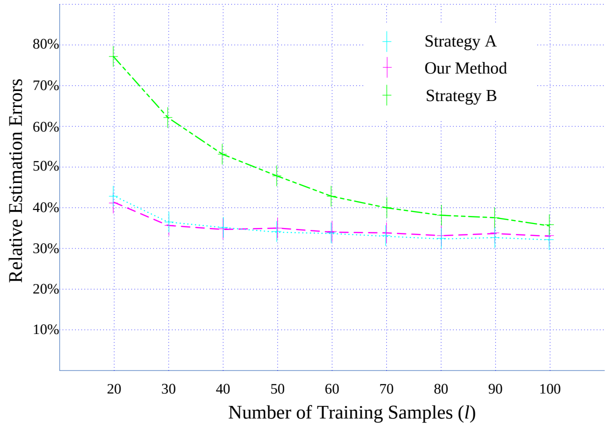

Figure 5. Comparison of three arrival time prediction strategies.

The y-axis is the relative estimation errors, while the x-axis stands for the number of samples used for training. Theoretically, strategy A should perform better than ours, yet the result shows that their accuracies are close, which may stem from the over-fitting effect of the former. Since strategy A is far more time-consuming than ours, we do not choose it. For strategy B, it is more sensitive to the size of the training dataset. It cannot achieve comparable performance as ours with less training data. Considering the result of this test, we use l = 35 for our method to balance between the training time and performance. All the following experiments use this value unless otherwise specified.

In the next experiment, we test the scenario of interleaved PoIs, where VoIP traffic and web browsing traffic are transmitted in the same channel. That is, a user is making an IP phone call and browsing web pages at the same time. Similarly, we assume VoIP packets and ICMP packets are of interest.

We run our prediction algorithm twice. In the first round, VoIP traffic and ICMP packets are treated as a single sequence. In the second round, we separate them, and train two different regression functions to predict the next VoIP packet and the next ICMP packet separately. VoIP packets are also IP packets. We distinguish them from the IP packets in web browsing traffic by identifying source and destination IP addresses. The result is shown in Figure 6.

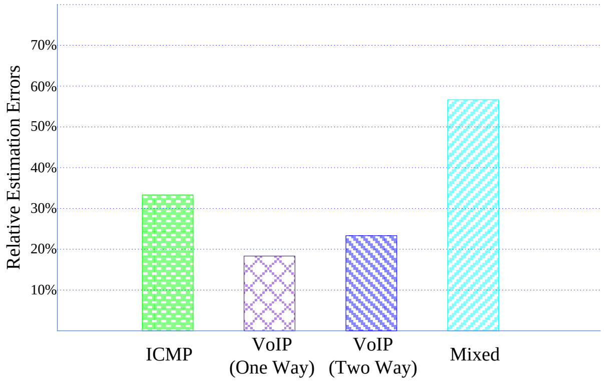

Figure 6. Categorized PoIs vs. interleaved PoIs.

From Figure 6, we can see that separate prediction has much better performance than mixing them together. Therefore, if a network forensics system wants to capture multiple types of packets in one channel, we should categorize the traffic first, and then apply our prediction method to each category separately.

B. Data Capture Performance of Small Number of Channels

We have presented the evaluation results of our prediction algorithm above. Now we further incorporate the monitor scheduling algorithm (Algorithm 1) to test the overall performance of our data capturing method. In this subsection, real-world tests are performed in a simplified scenario, where we do not consider dynamic join and leave of secondary users, and monitors dedicated for scanning are not used. We also assume the monitor can hear all the secondary users (i.e. do not consider the geographic coverage issue) in this experiment.

We use HP nc6000 and Dell E5400 laptops equipped with

802.11bg wireless cards (Atheros or Intel chipset) for our test. Three pairs of laptops (or AP-laptop pair) are working at channel 1, 6, and 11, respectively. It simulates a cognitive radio network with three channels and one monitor.

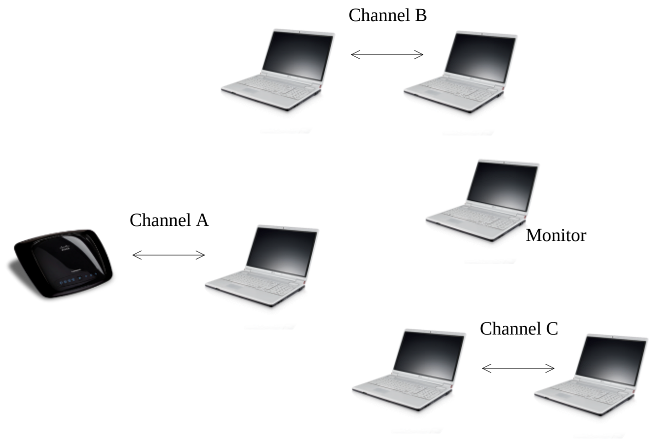

Figure 7. Experiment settings.

In the first experiment, channel A is web browsing in which ICMP packets are of interest. Channel B is occupied by VoIP streaming, which has a data rate of approximately 6Kbps and all the packets are of interest. Channel C is not used. We only have one monitor (a laptop with 802.11bg wireless card, in monitor mode) to capture the traffic on channel A and B using our method. t_s (channel switch time) of the monitor is about 5ms. w and v are both set to $(t_r + t_s)$, where t_r is the time relax for packet arrival prediction (see Section VB).

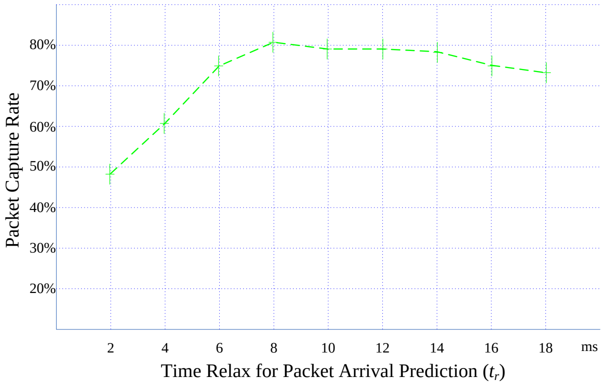

Figure 8. Influence of t_r on packet capture rate.

We vary t_r from 2 to 18ms and the packet capture rate (i.e. the number of captured PoIs divided by the number of total PoIs) is shown in Figure 8. From the experiment result we can see that when t_r is around 8ms, we are able to achieve the packet capture rate as high as 82%. When t_r is small the capture rate decreases because even small prediction errors cannot be tolerated. When t_r gets larger, w also gets larger, thus the monitor may not have enough time to switch to other channels. We choose t_r = 8ms for our method. The rest of experiments use this value unless otherwise specified.

In the next experiment, we test the influence of traffic loads on our method, and compare its performance to the baseline (random capture). Channel A and B are the same as above, while channel C is used for FTP downloading (assume all download packets are of interest). We vary the download speed of channel C and use one and two monitors respectively to capture the traffic of all three channels. The result is shown in Figure 9.

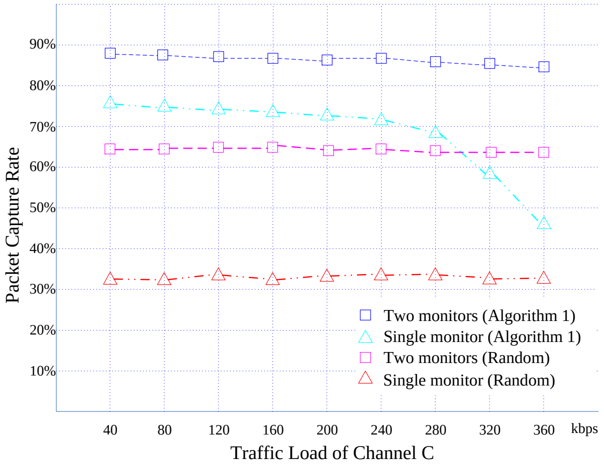

Figure 9. Influence of traffic load on packet capture rate.

In the one monitor case, we can see the packet capture rate of our method decreases quickly when the data rate of channel C goes higher than 280Kbps. This is because with higher data rate, the time intervals between packets in channel C become shorter, which is not enough for a single monitor to switch to the other two channels. In the case of two monitors, such performance deterioration is not found. Even if one monitor is stuck in the busiest channel, the other still can switch between the other two channels freely.

For the random scheme, monitor(s) switch between channels randomly. The experiment results demonstrate that for both one or two monitor cases, our method significantly outperform the random method. This experiment also implies that our data capturing method is able to achieve high channel-over-monitor rate if the distribution of PoI is sparse.

C. *Data Capture Performance of Large Number of Channels*

Now we move on to the performance of the complete version of our method. In this subsection, we increase the number of channels; consider dynamic join and leave of secondary users; in addition, secondary users will change their current communication channel upon the appearance of primary users. 802.11bg networks only have three non-overlapping channels; this part of evaluation is conducted by simulation.

In our settings, there are 60 channels (N = 60) in total, and about 20 (18-22, due to dynamic join and leave) of them are occupied by secondary users at any given time. When newly coming or changing a channel, a secondary user randomly chooses one of the currently available channels. Real traces are used to simulate the packet communication. Among active channels, 10 are web browsing traffic where in half of them, ICMP packets are of interest; and for another half, all packets are of interest. 5 channels are occupied by VoIP traffic, and another 5 are streaming videos. For the former, all packets are of interest while for the latter, only I-frames are of interest. Primary users are simulated by placing special packets in the channel.

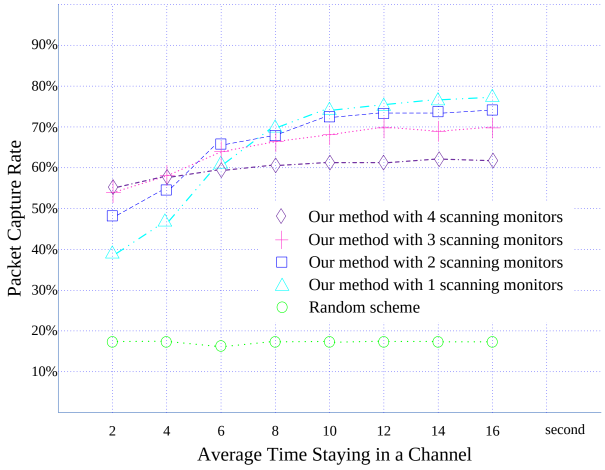

Figure 10. Overall performance of data capturing in CRN.

We use 11 monitors (M = 11), and 1 to 4 of them are dedicated for secondary signal scanning. The rest of parameters are the same as the experiments above. Secondary users are relatively close to each other; every monitor can hear all the communications. The result is shown in Figure 10. The x-axis is the average time a secondary user stay in a channel before it quits the network or jumps to another channel.

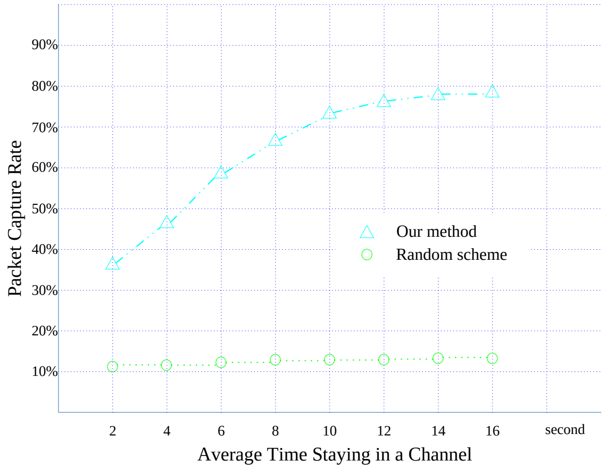

Figure 11. Overall performance considering geographic coverage.

Our method significantly outperforms the random scheme. Without channel scan and packet arrival prediction, 11 monitors had a difficult time dealing with 60 channels. The packet capture rate of the random scheme is less than 20%. On the contrary, with a proper number of scanning monitors, our method is able to achieve a packet capture rate of 70%-75% most of the time (when secondary users stay in a channel averagely longer than 8 seconds). Compared to the one-scanning-monitor case, the capture rate using more scanning monitors is higher if secondary users have high dynamics (stay at one channel for only a short time). It is because a single scanning monitor has larger delay to find newly coming signals. When there is less dynamics, the configuration of single scanning monitor has better performance, for it in turn leaves more monitors for packet capture. We can see that using two scanning monitors has relatively the best overall performance in this experiment. For our protocol, we assign $\lceil M/6 \rceil$ monitors dedicatedly for channel scanning to achieve a balance between the number of monitors used for data capture and the detection delay of secondary signals.

In the next experiment, we take geographic coverage into account. All the secondary users are distributed within the area whose radius is twice of a node's transmission range (as well as a monitor's listening range). We have 60 channels and 18 monitors. Monitors are randomly placed (but same for our method and the random scheme). $\lceil 18/6 \rceil = 3$ of them are used dedicatedly for scanning. They are selected based on the criterion whether they can cover all the secondary users. If all the three-monitor-combination cannot meet this criterion, the number of scanning monitors is increased by one (see Section VB) until satisfying. The rest of settings are the same as above. We change the locations of secondary users and monitors for each run; the results of ten runs are averaged and plotted in Figure 11, which shows our method still largely outperforms the random scheme.

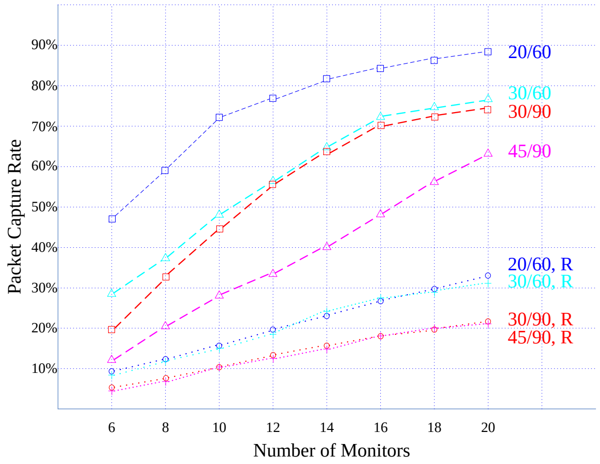

Figure 12. Scalability of our method.

Now, we vary the numbers of monitors, the number of total channels, and the number of *busy* channels, testing the scalability of our method. The average time a secondary user staying in a channel is set to 10. The traffic types are the same as above. The number of traces for each type is adjusted proportionally. Figure 12 shows the results without considering geographic coverage, while in Figure 13, we take geographic coverage into account and further vary the radius of secondary users' distribution area.

In Figure 12, "20/60" means in total 60 channels 20 are *busy*, and so forth. "R" stands for the random scheme, while without "R" refers to our method. We can observe from the results that our method has good scalability. When the number of monitors is relatively small, the capture rate almost increases linearly. When the number of *busy* channels is significantly larger than the number of monitors, our method still provides best-effort service without sharp performance deterioration. Under the various conditions, it always performs much better than the random scheme.

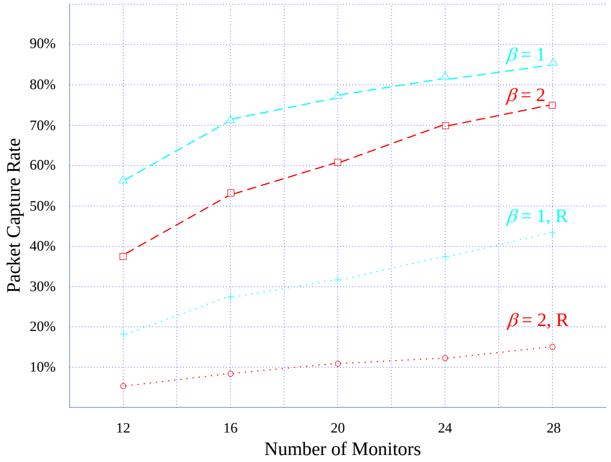

Figure 13. Influence of secondary user's distribution.

In Figure 13, β is the ratio of the secondary users' spanning radius to the monitors' listening range. For example, $\beta = 2$ means the radius of secondary users' distribution region is twice of a monitor's listening distance (so the area is four times as large). The number of channels and *busy* channels are fixed at 60 and 30, respectively. From the plot, we can see that as the network become sparser, the performance degradation of our method is much smaller than the random scheme (curves marked with "R"). However, it is reasonable to employ more monitors to achieve similar performance when the secondary users are spanning over a larger area.

VII. Discussion

A. Dynamics of Secondary Users

As mentioned, in cognitive radio networks, secondary users opportunistically access the spectrum. They dynamically join and leave the network, and also change their channels to avoid primary users. An optional operation mode in IEEE 802.22 even requires channel hopping on a regular basis [14].

In order to catch up with such dynamics, we employ scanning monitors in our method to scan *inactive* channels repeatedly, as well as opportunistically utilize the free time intervals of other monitors (see Section VB).

An alternative might be probabilistically predicting the appearance of secondary signals. Some research work has been done to predict channel availability in cognitive radio networks [15] [16]. It seems that we can utilize them to predict unavailable channels (secondary signals) and save scanning monitors. However, these studies assume that secondary users' behavior is consistent over time or follow certain probabilistic distributions. These assumptions may not be true in practice.

First, secondary users' behavior usually has a lot of randomness. Secondary users of a network in a period of time may largely different from the users of the same network in another period of time. It is difficult to assume they have similar behavior. Second, the channel choice of secondary users is a function of the scanning algorithm and the channel measurement. Different users may use different scanning algorithms, such as sequential scan, optimal stopping, random access, etc [17]. On the other hand, a user's location, environment and hardware accuracy can greatly affect the measurements of channel state and channel quality. Therefore, the channel choice of a secondary user is very difficult to predict. Affected by these factors, probabilistically predicting the appearance channel of a secondary user can hardly achieve high accuracy in practice.

B. Geographical Coverage

Compared with the preliminary version [21], in this paper, we do not require that one monitor can hear all the secondary users. Each monitor may listen to part of the network and they cooperate together to capture all the traffic of interest. This improvement brings more flexibility and makes our method widely applicable to real world scenarios.

On the other hand, with the same number of channels and same amount of traffic of interest, it is obvious that more monitors are needed to achieve the similar performance if the secondary users are sparsely distributed. In our experiments (Section VI), we assume the monitors are randomly located for the fairness of the performance comparison (with the random scheme). However, if we place the monitors carefully, we can achieve better performance (capture rate) with the same number of monitors, or the same performance with fewer monitors. A general rule is to put more monitors at the place where the secondary users (traffic of interest) are denser. As a cognitive radio network standard, IEEE 802.22 introduces a star topology. In this case, more monitors should be located near the base station.

In this paper, we assume that the capture range of a monitor is the same as the transmission range of the secondary users (they are physically identical devices). In practice, if secondary users are sparse, we can also use monitors with more powerful antenna (thus larger listening range) to reduce the number of monitors required.

C. Application Dependent Packet Prediction

As discussed, we predict packet arrival time using support vector regression. This method is not traffic type specific. In some cases, if the type of application is known, the prediction accuracy can be further improved. For example, FTP download traffic has a regular pattern that the intervals between the packets are almost identical. VoIP and video streaming also has very predictable behavior.

Of course, identifying application type by packets or traffic characteristics is a challenging problem, especially the payload of wireless packets are typically encrypted. Research on this area is orthogonal to our work [22] [23]. However, for some easy cases, information in the header, such as well-known port numbers and source / destination IP address can be used to identify the application and helps improve the accuracy of prediction.

If we do not have the information of the application type, but know that in a given channel only one single application is transmitting (and only one type of packets are of interest), the performance of our method can also be improved. Instead of a fixed t_r (see Section V), we introduce a dynamic t_r that is proportional to the recent τ (average packet inter-arrival time of PoIs). With this modification, we observe an average capture rate increase of 2%-5% in our experiment for the single application (per channel) case.

D. Encrypted Traffic

In wireless networks, some traffic flows are encrypted to ensure security and privacy. On the other hand, for the purpose of wireless forensics, a number of techniques have been developed to deal with encrypted traffics, which can be divided into three categories.

Clear part analysis. Even if the traffic is encrypted, packet headers are still in plain text. Besides, management packets and control packets also travel in the clear (e.g. in 802.11 networks). Such information can be utilized by forensic systems.

Fingerprint analysis. The sequences of client-server exchanges, payload sizes, transmission intervals, involved protocols, etc. are exploited to extract signatures of certain software or certain user behavior. *fl0p* [24] is an example of this category.

Traffic statistics analysis. It is similar to fingerprint analysis, but it focuses more on the statistical data (e.g. average packet size, average data rate, etc.) on a larger traffic dataset and usually cares less about the sequence of events. Forensic systems make use of it to detect session types and network anomalies.

We can see that for encrypted traffic, packets (or headers) still need to be captured before further analyses in wireless forensics. Therefore, our data capture method is orthogonal to the above techniques and can still be applied in encrypted network scenarios.

VIII. Conclusion

In this paper, we introduced a systematic method for data capturing in cognitive radio networks. Given a large number of channels, our method is able to achieve high packet capture rate with a small number of monitors.

In order to reuse the monitors in the time domain, we proposed a packet arrival prediction method based on incremental support vector regression. A monitor scheduling algorithm and a comprehensive protocol are provided to coordinate monitors among channels. We also take geographic coverage of wireless signals into account. Monitors are scheduled not only according to their workload but also their locations and listening ranges. We conducted both real-world experiments and simulations to evaluate our method. The results demonstrate that our method significantly outperforms the random scheme under various conditions. We also evaluate the scalability of our method in sense of network scale, traffic load and secondary user density, which shows promising results.

References

[1] S. Garfinkel, "Network forensics: tapping the Internet," http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html.

[2] G. Iannaccone, C. Diot, I. Graham, N. McKeown, "Monitoring very high speed links," ACM Sigcomm Internet Measurement Workshop, San Francisco, USA, Nov. 2001.

[3] L. Deri, "Improving passive packet capture: beyond device polling," Proc. System Administration and Network Engineering (SANE), 2004.

[4] R. Siles, "Wireless forensics: tapping the air," http://www.symantec.com/connect/articles/wireless-forensics-tapping-air-part-one.

[5] D.J. Geiger, G. Scheets, K.A. Teague, J. Pitts, "Multi-channel packet capture in 802.11b/g wireless networks," 42nd Asilomar Conference on Signal, System and Computers, California, USA, 2008.

[6] L. Choong, "Multi-channel IEEE 802.15.4 packet capture using software defined radio," M.S. Thesis, UCLA, 2009.

[7] N.I. Sapankevych, R. Sankar, "Time series prediction using support vector machines: a survey," IEEE Computational Intelligence Magazine, pp. 24 - 38, May 2009.

[8] T. Phit, K. Abe, "Packet inter-arrival time estimation using neural network models," Internet Conference, Tokyo, Japan, 2006.

[9] S. Northcutt, J. Novak, "Network intrusion detection," 3rd Edition, New Riders Publishing, 2003.

[10] A.J. Smola, B. Scholkopf, "A tutorial on support vector regression," Statistics and Computing, Springer, 2004.

[11] G. Cauwenberghs, T. Poggio, "Incremental and decremental support vector machine learning," in T.K. Leen, T.G. Dietterich, V. Tresp, editors, Advances in Neural Information Processing Systems, volume 13, pp. 409 - 415. MIT Press, 2001.

[12] J. Ma, T. James, P. Simon, "Accurate online support vector regression," Neural Computation, 15(11): 2683 - 2703, 2003.

[13] D. Murray, M. Dixon, T. Koziniec, "Scanning delays in 802.11 networks," 2007 International Conference on Next Generation Mobile Applications, Services and Technologies, Wales, UK, 2007.

[14] W. Hu, D. Willkomm, G. Vlantis, M. Gerla, A. Wolisz, "Dynamic frequency hopping communities for efficient IEEE 802.22 operation," IEEE Communications Magazine, pp. 80 - 87, May 2007.

[15] A. Anandkumar, N. Michael, A.K. Tang, "Opportunistic spectrum access with multiple players: learning under competition," 29th IEEE Conference on Computer Communications (Infocom), San Diego, USA, 2010.

[16] K. Liu, Q. Zhao, B. Krishnamachari, "Distributed learning under imperfect sensing in cognitive radio networks," 44th Asilomar Conference on Signal, System, and Computers, California, USA, 2010.

[17] H. Jiang, L. Lai, R. Fan, H.V. Poor, "Optimal selection of channel sensing order in cognitive radio," IEEE Transactions on Wireless Communication, 8(1): 297 - 307, 2009.

[18] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, 2(2): 121 - 167, 1998.

[19] A. Chhetri, H. Nguyen, G. Scalosub, R. Zheng, "On quality of monitoring for multi-channel wireless infrastructure networks," 11th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 111 - 120, Chicago, USA, 2010.

[20] P. Arora, C. Szepesvari, R. Zheng, "Sequential learning for optimal monitoring of multichannel wireless networks," 30th IEEE Conference on Computer Communications (Infocom), Shanghai, China, 2011.

[21] S. Chen, K. Zeng, P. Mohapatra, "Efficient data capturing for network forensics in cognitive radio networks," 19th IEEE International Conference on Network Protocols (ICNP), Vancouver, Canada, 2011.

[22] S. Zander, T. Nguyen, G. Armitage, "Automated traffic classification and application identification using machine learning," 30th IEEE Conference on Local Computer Networks, Sydney, Australia, 2005.

[23] T. Nguyen, G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," IEEE Communications Surveys and Tutorials, 10(4): 56 - 76, 2008.

[24] http://permalink.gmane.org/gmane.comp.security.honeypots/3639

Shaxun Chen is currently a Ph.D. student in the Department of Computer Science at University of California, Davis. He received his B.Sc and M.Sc degrees in Computer Science from Nanjing University, China, in 2005 and 2008, respectively. He was a visiting researcher in Institute for Infocomm Research (I^{2}R), Singapore, in 2007. He has published over 20 papers in international conferences and journals, such as ICNP, INFOCOM, IEEE Transactions on Mobile Computing, and Data & Knowledge Engineering. His current research interests include wireless security, network forensics, and video forensics.

Kai Zeng received his Ph.D. degree in Electrical and Computer Engineering at Worcester Polytechnic Institute (WPI) in 2008. He obtained MS in Communication and Information Systems and BS in Communication Engineering both from Huazhong University of Science and Technology, China in 2004 and 2001, respectively. He was a postdoctoral scholar in the Department of Computer Science at University of California, Davis (UCD) from 2008 to 2011. He joined the Department of Computer and Information Science at University of Michigan - Dearborn as an assistant professor in 2011. He is a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) award in 2012. He won Excellence in Postdoctoral Research Award at UCD in 2011 and Sigma Xi Outstanding Ph.D. Dissertation Award at WPI in 2008. His current research interests are in wireless network security, physical layer security, cognitive radio networks, energy efficiency, and cyber-physical systems.

Prasant Mohapatra is currently a professor in the Department of Computer Science at University of California, Davis. He is the former Tim Bucher Family Endowed Chair Professor and the former chairman of the department. In the past, he has been on the faculty at Iowa State University and Michigan State University. He has also held Visiting Scientist positions at Intel Corporation, Panasonic Technologies, Institute of Infocomm Research (I^2R), Singapore, and National ICT Australia (NICTA). He has been a Visiting Professor at the University of Padova, Italy, Korea Advanced Institute of Science and Technology (KAIST), and Yonsei University, South Korea. He has served on the editorial boards of the IEEE Transactions on Computers, IEEE Transactions on Mobile Computing, IEEE Transaction on Parallel and Distributed Systems, ACM WINET, and Ad Hoc Networks. He has been on the program/organizational committees of several international conferences. He is the Editor-in-Chief of the IEEE Transactions on Mobile Computing.

Dr. Mohapatra received his doctoral degree from Penn State University in 1993, and received an Outstanding Engineering Alumni Award in 2008. He also received an Outstanding Research Faculty Award from the College of Engineering at the University of California, Davis. He is a recipient of the HP Labs Innovation Research Award. He is a Fellow of the IEEE and AAAS.

Dr. Mohapatra's research interests are in the areas of wireless networks, mobile communications, sensor networks, Internet protocols, and QoS. He has published more than 250 papers in reputed conferences and journals on these topics. Dr. Mohapatra's research has been funded through grants from the National Science Foundation, Department of Defense, Intel Corporation, Siemens, Panasonic Technologies, Hewlett Packard, Raytheon, Huawei Technologies, and EMC Corporation.