SpecMonitor: Towards Efficient Passive Traffic Monitoring for Cognitive Radio Networks

Qiben Yan* Ming Li† Feng Chen‡ Tingting Jiang* Wenjing Lou* Y. Thomas Hou* Chang-Tien Lu*
* Virginia Polytechnic Institute and State University, Blacksburg, VA. Email:
{qbyan,virjtt03,wjlou,thou,ctlu}@vt.edu
† Utah State University, Logan, Utah. Email: ming.li@usu.edu
‡ University at Albany - SUNY, Albany, NY. Email: fchen5@albany.edu

***Abstract*—Passive monitoring by distributed wireless sniffers has been used to strategically capture the network traffic, as the basis of automatic network diagnosis. However, the traditional monitoring techniques fall short in cognitive radio networks (CRNs) due to the much larger number of channels to be monitored, and the secondary users' channel availability uncertainty imposed by primary user activities. To better serve CRNs, we propose a systematic passive monitoring framework, SpecMonitor, for traffic collection using a limited number of sniffers in Wi-Fi like CRNs. We jointly consider primary user activity and secondary user channel access pattern to optimize the traffic capturing strategy. In particular, we exploit a non-parametric density estimation method to learn and predict secondary users' access pattern in an online fashion, which rapidly adapts to the users' dynamic behaviors and supports accurate estimation of merged access patterns from multiple users. We also design near-optimal monitoring algorithms that maximize two levels of quality-of-monitoring goals respectively, based on the predicted channel access patterns. The simulations and experiments show that SpecMonitor outperforms the existing schemes significantly.**

Keywords*—Cognitive radio network, passive monitoring, non-parametric density estimation, optimization algorithm.

I. Introduction

Cognitive Radio (CR) has been envisioned as a new paradigm to better utilize the spectrum resources, by allowing unlicensed or *secondary users* (SUs) to opportunistically access the licensed bands, as long as they do not cause any interference to licensed or *primary users* (PUs). While most of the prior research in CRNs focused on the problem of establishing a single link between SUs [1], recent research has gone beyond a single link to identify the challenges of implementing a Wi-Fi like CR network [2] consisting of secondary Access Points (APs) associated with multiple secondary clients.

Passive monitoring has been used to measure Wi-Fi networks [3]–[5] using a dedicated set of hardware devices, called *sniffers*. It has been shown to complement the wire side monitoring by gathering detailed PHY/MAC information. Passive monitoring serves as the basis of numerous applications ranging from network forensics, fault diagnosis to resource management. As the quality of those applications mainly depends on that of traffic monitoring, it is non-trivial to build a traffic monitoring framework with excellent monitoring performance. Passive monitoring is particularly important to CRNs, because: (1) cognitive radios are programmable and difficult to manage; (2) the interference requirement in CRNs is mandatory and extremely high. In this paper, we consider the construction of a passive monitoring framework for Wi-Fi like CR networks, or "WhiteFi" networks for short.

However, passive monitoring becomes a challenging task in WhiteFi networks. First, WhiteFi networks have a much wider spectrum ($50MHz$-$698MHz$) than traditional wireless networks, which makes it infeasible to deploy one sniffer for each channel. As a result, the sniffers have to decide which subsets of channels they will operate on, referred to as *sniffer channel assignment problem*. Second, SUs have to vacate the channels immediately once PUs start transmissions on the corresponding channels. Such inevitable channel switching behavior potentially complicates the sniffers' traffic monitoring strategies. Last but not the least, network traffic on each channel typically comes from multiple SUs, who share the spectrum by following a certain medium access control (MAC) mechanism. Thus, traffic patterns observed by the sniffers are highly dynamic, further complicating the sniffers' monitoring strategies.

To meet these challenges, we propose a monitoring framework, SpecMonitor, which utilizes a non-parametric density estimation method to model SUs' channel usage pattern. This method makes no assumptions on the unknown distribution of channel access pattern, thus offers accurate and flexible models which can be updated in an online fashion with acceptable complexity. Moreover, we design a sliding window method to perform online learning of data dynamics, and an accumulative combination method to further improve modeling accuracy. Then, SpecMonitor takes inputs from SUs' channel usage model to construct monitoring strategies.

In this paper, we consider two levels of monitoring objectives: frame-level and user-level, to diagnose different network issues. The frame-level objective can be interpreted as maximizing the *frame-level quality-of-monitoring* (FL-QoM), defined as the amount of captured MAC frames of interest, due to their significance for the subsequent aggregated traffic analysis [5]. The user-level objective is to maximize the *user-level quality-of-monitoring* (UL-QoM), defined as the expected number of active users monitored, which can facilitate user behavior analysis [6]. We cast the monitoring optimization problem as a sniffer channel assignment problem with objective of maximizing the corresponding QoMs.

In this paper, we make the following contributions:

(1) We design a general framework to monitor the WhiteFi networks, which jointly considers the channel availability and secondary user access pattern. In particular, we design an *online non-parametric density estimation* mechanism to model the secondary user channel activity, which is able to support dynamic and complex access patterns.

(2) We formulate the sniffer channel assignment problems as *integer programming* (IP) problems by incorporating the channel switching costs with the QoM objective, for which we provide algorithms to optimize two different levels of QoMs respectively.

(3) We present approximation algorithms to provide near-optimal solutions. Numerical analysis shows the solutions can offer objective values that are very close to the optimal value, thus confirming their near-optimality. Furthermore, we conduct extensive simulations and experiments to validate the efficacy and efficiency of our statistical model and monitoring framework.

The remainder of this paper is organized as follows. We introduce the related work in section II. In section III, we describe the monitoring system model. The secondary user channel access model is depicted in section IV, followed by section V, which formulates the monitoring optimization problems and provides near-optimal solutions. Section VI presents the evaluation results using both synthetic data from simulations and real data from experiments. Finally, section VII concludes the paper.

II. Related Work

Passive Monitoring in Traditional Wireless Networks: Passive monitoring in wireless networks has been an active research area. Yeo *et al.* were the first to use dedicated sniffers to passively measure a Wi-Fi network, successfully identifying protocol anomalies and malicious WLAN usages [3]. Cheng *et al.* presented Jigsaw, which is a large-scale passive monitoring infrastructure to collect and dissect wireless traffic for cross-layer network diagnosis in a large enterprise Wi-Fi network [4], [5]. While the above works focused on developing the monitoring infrastructure, some recent works investigated the problem of optimal sniffer channel assignment to maximize the amount of monitored information. Shin *et al.* [7] formulated the sniffer channel assignment problem in the wireless mesh network as a maximal coverage problem, and designed approximation algorithms to solve this problem. In [8], Chhetri *et al.* further extended the preceding work by taking into account the users' access patterns. They proposed two monitoring models: user-centric model and sniffer-centric model. However, they assume the statistics for different users' activities are known. Recently, Arora *et al.* [9] proposed to use multi-armed bandit to perform sequential learning of the unknown channel statistics, which can be used to facilitate optimal channel assignments. However, multi-armed bandit is too complex to be used for online and efficient channel assignments. In this paper, we present an efficient online channel assignment mechanism without any prior knowledge of channel access statistics, which is the first mechanism in the literature to provide optimized channel assignments in real-time. All the above works only considered maximizing the number of active users covered by the sniffers, while we further address the problem of maximizing the number of captured frames.

Spectrum Monitoring in Cognitive Radio Networks: Chen *et al.* studied frame capturing problem for network forensics in CRNs [10], in which support vector regression (SVR) method is employed to predict the frame arrival time to guide channel assignments. They have similar objectives as ours, however, our method has the following advantages: 1) SVR method requires a time consuming training phase, while we utilize density estimation to produce new estimates in an online fashion avoiding of the expensive training and retraining phases; 2) SVR method falls short of dealing with interleaved traffic from multiple users, which corresponds to dynamic traffic statistics, while our scheme can adapt promptly to the traffic dynamics; 3) their monitoring framework has poor performance when the monitored channels carry high data rate traffic, because of frequent channel switching behavior induced by the heuristic channel assignments. In contrast, as we jointly consider channel switching costs and frame capturing gains to optimize channel assignments, our method can achieve better performance with fast traffic flows. Recently, Yi *et al.* formulated the secondary user data capturing problem as multi-armed bandit problem [11], which takes a long period of learning process before it is able to produce an accurate estimation of user access pattern. Thus, their method is not efficient enough to capture adaptive and interleaved traffic patterns either.

III. System Model

A. Monitoring System Model

In this section, we describe the monitoring system model for CR networks. We consider CR networks with coexisting PUs and SUs. The most common PUs are TV towers and wireless microphones (WMs). Our monitoring system is interested in the network traffic from SUs including APs and clients who form a WhiteFi network, as illustrated in Fig. 1. Curious readers please refer to [2] for the design and implementation details of WhiteFi system.

For a multi-hop network, we segment the whole network region into small regions, called *monitoring areas*, and assign a certain number of sniffers to monitor the traffic for each monitoring area. Each sniffer may be equipped with multiple antennas, which allow him/her to sense/capture traffic over multiple channels at one time. We assume different AP-client pairs in the monitoring area pick different working channels to avoid interference, and each sniffer can overhear all the inbound and outbound traffic from any secondary device inside its monitoring area if they tune into the same channel. Similar to [10], some sniffers are used as dedicated *inspection sniffers* that periodically sense channels to gain channel usage statistics, while other sniffers called *operation sniffers* are responsible for capturing information. All the sniffers are connected to a *sniffer center* for centralized decision making, as shown in Fig. 1. Each inspection sniffer is assigned multiple channels to scan. A *sensing slot* is a period during which the inspection sniffer scans through all the assigned channels. In

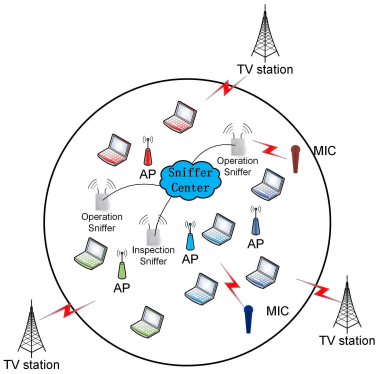


Fig. 1: Monitoring system architecture for WhiteFi network inside a monitoring area

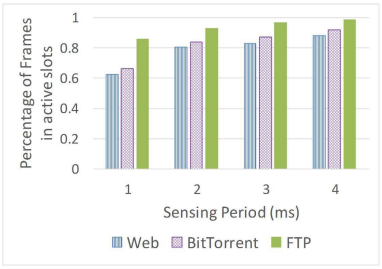


Fig. 2: The percentage of frames in active slots (20 ms slot length)

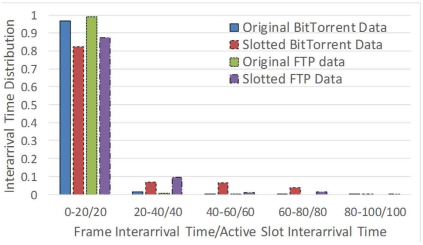


Fig. 3: Frame/Active slot interarrival time distribution (20 ms sensing slot, 2 ms sensing period)

the following, a *slot* stands for the sensing slot unless otherwise noted.

B. Channel Access Model

A sensing slot is composed of channel sensing and channel switching time, whose length depends on the number of channels to be scanned. Typically, a channel sensing period is approximately $1ms$ per channel using energy detection [12], while channel switching for a commodity $802.11b/g$ network card takes about $1-5ms$ [13].

During each slot, the inspection sniffer scans several channels to reveal their channel states (*active/idle*). Here, an *active slot* indicates a slot during which the sniffer spots SUs' traffic after channel sensing, while *idle slot* represents the opposite. As channel sensing period is less than a full slot length, the discovered active/idle state may not reflect the genuine state of a slot. Let X_k^i be the state of channel i at k-th sensing slot, which takes only binary values "$1/0$", corresponding to $active/idle$ state (in the following, we omit the subscript i for i-th channel). Then, the sequential data X_k $(k=1,2,\ldots)$ are used to calculate the *active slot interarrival time* for each channel, which is defined as the time interval between two consecutive $active$ slots. In our design, inspection sniffers produce active slot interarrival time as their sensing outcomes, which will be used as the inputs to build channel access model as explained in Section IV.

Note that one straightforward way of meeting frame capturing objectives is to predict SUs' frame arrival time by modeling frame arrival pattern. However, it is infeasible to derive optimized channel assignments with dynamic frame length and unslotted frame transmission [10]. Instead of directly modeling frame arrival pattern, we model the active slot interarrival pattern as the basis of our monitoring framework, in which monitoring an active slot implies capturing all the frames in the slot. To motivate/justify the adoption of active slot interarrival time, we performed a real-world experiment using the traffic from different types of applications (e.g. Web browsing, Bittorrent, FTP) in operational $802.11g$ WLAN. The experiment settings are illustrated in Section VI-B. Fig. 2 shows the percentage of frames in active slots corresponding to different sensing periods, from which we notice most of the frames reside in the identified active slots, especially when the sensing period is longer than $2ms$. In other words, by capturing frames in active slots, we are able to collect most of the frames. Fig. 3 plots the histograms of frame interarrival time versus active slot interarrival time, with each bar showing the percentage of frames or active slots whose interarrival time is indicated by the x-axis. These two distributions appear very similar to each other with most of the frames concentrated within small interarrival time region, indicating active slot interarrival time well characterizes channel usage pattern. For ease of reference, the commonly used notations are summarized in Table I.

K	Number of time slots
N	Number of channels
$\mathcal{N}$	Channel index set
M	Number of antennas of all operation sniffers
$\mathcal{S}_{op}$	Operation sniffer antenna index set
X_k	Sensing results of inspection sniffers at slot k
$\mathcal{Z}(k)$	Current data set at slot k
$\mathcal{Z}_{in}$	Input data set for density estimation
T_{int}	Active slot interarrival time
W	Sliding window size for online estimation
n_w	Number of previous windows considered for combination
t_w	Number of different samples in the previous window for combination
f	Probability density estimate
F	Cumulative density estimate
$SCAP_i$	Slotted channel access probability of channel i
$\triangle$	Sensing slot length
$IdleCount$	Number of idle slots counted
y_i	Vector of binary random variables for channel i
$z_{s,i}$	Vector of binary random variables indicating whether sniffer s is assigned to channel i
α	Switching cost weight
U_i	Number of distinct users in channel i

TABLE I: Summary of Symbols and Notations

IV. User Channel Access Prediction

In this section, we propose a unified model to estimate secondary user channel access pattern, as the front-end of SpecMonitor. In order to build the unified model, we first study the primary user detection issue, and then we design an online non-parametric density estimation mechanism to predict SUs' *slotted channel access probability* ($SCAP$) pertained to each sensing slot. As its name suggests, slotted channel access

probability is defined as the probability of SUs' channel access during each slot.

A. Primary User Detection

To enable CR communications, SUs need real time knowledge of PUs' activity to identify available spectrum. Similarly, the sniffers are also required to detect PUs' activity in order not to waste time and energy listening on the *primary-occupied channels*. Primary user detection can be achieved by using either spectrum sensing or by querying a geo-location white space database over the internet. Spectrum sensing is expensive in cost, energy consumption and complexity of hardware. On the other hand, the database approach is easier to implement, which allows devices to report their locations to a web server that returns a list of available channels at that location. However, database approach suffers from utilization inefficiency, since it uses propagation models to decide the available spectrum, and hence, is conservative in the channels it returns for a given location. Either of these two approaches can be applied to our monitoring framework.

Feature detection is one popular spectrum sensing method for the sniffers to detect PUs' appearance. The feature detection algorithms described in [14] can be used to sample the UHF spectrum to detect the presence of TV broadcasts and wireless microphone signals, which can effectively differentiate between the SUs' and PUs' signals. Then, the sniffers can directly perform feature detection in the beginning of every slot to sense the availability of monitored channels.

The database approach allows both sniffers and SUs to query the database for spectrum availability at a certain location. After querying the database, the SUs begin operating on a set of available channels, while inspection sniffers tune onto these available channels to monitor SUs' traffic patterns, and operation sniffers are assigned to the SU-occupied channels correspondingly. In SpecMonitor, we adopt the database approach for simplicity.

B. Secondary User Channel Access Model

In this section, we propose a framework to estimate the secondary users' SCAP at each slot by modeling the active slot interarrival time distribution. The SUs' channel access pattern in WhiteFi networks is complicated, mainly due to the dynamics brought by time-evolving mixed traffic from multiple SUs with channel switching behavior.

1) Non-parametric Density Estimation Model: Instead of assuming a specific active slot interarrival time distribution for quantifying SUs' traffic pattern, we propose a SU channel usage model using the *non-parametric density estimation* method to better capture SUs' traffic dynamics. Currently, one of the most popular non-parametric density estimation approaches is *Kernel Density Estimator* (KDE) with a Gaussian kernel function [15]. Given n independent realizations X_i $(i = 1, 2, \ldots, n)$ drawn from an unknown *probability density function* (pdf) $f(x)$, the Gaussian KDE with bandwidth σ is defined as:

$$\hat{f}(x;\sigma) := \frac{1}{n}\sum_{i=1}^{n} K_G(x, X_i, \sigma), x \in \mathbb{R}, \tag{1}$$

where

$$K_G(x, X_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-X_i)^2/(2\sigma^2)}, \tag{2}$$

from which we can see that Gaussian KDE is essentially the overall sum of Gaussian kernels centered at location X_i with an equal bandwidth σ.

In fact, the setting of σ is of utmost importance for the density estimation performance. A classic measure to determine the optimal σ is *Mean Integrated Squared Error* (MISE):

$$MISE_{\{\hat{f}\}}(\sigma) := E[\hat{f}(x;\sigma) - f(x)]^2, \tag{3}$$

where $f(x)$ is the underlying genuine distribution. Assuming a large sample set, we can obtain an asymptotic approximation to MISE, denoted as *asymptotic MISE* (AMISE), written as [15]:

$$AMISE_{\{\hat{f}\}}(\sigma) = \frac{1}{4}\sigma^4 \|f''(x)\|^2 + \frac{1}{2n\sqrt{\pi}\sigma}, \tag{4}$$

where $f''(x)$ is the second derivative of $f(x)$, and $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}$. Thus, the asymptotic optimal value of σ^* is obtained by minimizing AMISE:

$$\sigma^* = \left(\frac{1}{2n\sqrt{\pi}\|f''(x)\|^2}\right)^{1/5}. \tag{5}$$

In order to compute σ^* from Eq. 5, we need to approximate $\|f''(x)\|^2$ by estimating the general form $\|f^{(j)}(x)\|^2$ for arbitrary j. The corresponding optimal solution $\sigma_j^* = \left(\frac{1}{2n\sqrt{\pi}\|f^{(j)}(x)\|^2}\right)^{1/5}$ with a generalized term of $\|f^{(j)}(x)\|^2$ can be solved in a recursive form, namely $\sigma_j^* = \gamma_j(\sigma_{j+1}^*)$, where γ_j is a complicated formula given in [15]. Then, a fixed point iteration method is employed to compute σ_2^*, which is equivalent to the target value σ^*. This KDE algorithm provides a viable means of automatically selecting optimal bandwidths with superior density estimation performance.

2) Modeling Active Slot Interarrival Time Distribution: The KDE collects the data set of active slot interarrival time measured by inspection sniffers to generate the density estimates. Since the distribution of collected data sets may vary over time, the modeling accuracy of the KDE will be affected by taking into account outdated historic data. Thereby, only the most recent data should be imported into the modeling process. On the other hand, the modeling accuracy also largely depends on the size of the input data sets. If we only consider the most recent observations by discarding all the historical ones, the modeling accuracy will be brought down significantly. Furthermore, the amount of inputs to KDE has great impacts on its computational efficiency. Generally speaking, KDE with a small data set runs more efficiently than that with a large data set. Therefore, the major issue of this model is to decide how much historical data should be incorporated for density estimation, in order to produce an accurate and efficient model.

Now we present our proposed online non-parametric density estimation protocol. The basic idea is to use sliding window method to perform online updating of the density estimates, and to incorporate additional historic data sets for improving the estimation accuracy. The whole protocol is presented in

Algorithm 1, which is repeated for each channel. Whenever a new observation arrives, the online estimation model only takes the data in a sliding window of size W, i.e., the data sets exporting to the KDE only hold the most recent W observations. The setting of window size W is pertained to the data dynamics, thus is empirical. A simple guideline would be: first, we set an initial value for the sliding window size and run KDE; second, we move the sliding window forward to see whether the estimated distribution changes over time; third, if the change is significant, we decrease the window size, otherwise, increase it, until we reach a satisfactory window size. Specifically in the WhiteFi network scenario, we set a relatively small W as 50 data samples, since the data distribution will change more dynamically than that in a traditional wireless network.

One of the most favorable features of sliding window method is attributed to its support for online learning of density estimates. As time advances, our density estimator will take newest sets of data falling inside the sliding window to compute the latest estimate. Therefore, our model enables the effective characterization of the time-evolving active slot interarrival distribution, and allows us to update density estimates with every newly arrival observation.

However, the major drawback of the sliding window method resides in the following fact: the sliding window to specify input data also deteriorates the accuracy of KDE, because the size of sliding window restricts the number of observations (only W). Hence, we need to improve the estimates by expanding the input data size.

As depicted in Algorithm 1, we propose to combine the data sets from multiple sliding windows according to some well-defined criteria, in order to enlarge the sample space. How to define such criteria for merging sample space is crucial to the ultimate estimation performance. At first glance, more recent windows of data sets should have higher relevance to current window. Therefore, one intuitive method to achieve more accurate estimation is to combine the most recent density estimates from latest windows to capture the data freshness [16]. However, because of the uncertain channel availability and underlying MAC protocol, multiple clients may generate interleaved traffic due to alternate channel accesses. Therefore, the most recent windows may not necessarily reflect the underlying density of current window best, while some earlier historical data originating from the same clients pertaining to the current window might do. Accordingly, we propose an accumulative combination method to make the decision of merging historical data based on *statistical correlation* among the samples. As shown in Algorithm 1, we simplify the computation of statistical correlations by employing *Kolmogorov-Smirnov test* $(KS\ test)$. KS test is characterized as a *non-parametric inferential statistical method*, since it makes no assumption about the distributions of samples, thus is completely data-driven. The Kolmogorov-Smirnov statistic is defined as follows:

Definition 1: Consider two sets of observations $\mathcal{Z}_1$ and $\mathcal{Z}_2$, with $n_1 = |\mathcal{Z}_1|$ and $n_2 = |\mathcal{Z}_2|$ samples. The Kolmogorov-Smirnov statistic is defined as:

$$D_{z_1,z_2} = sup_x |F_1(x) - F_2(x)|,$$

where F_1 and F_2 represent the empirical cumulative distribution functions (cdfs) of the samples in $\mathcal{Z}_1$ and $\mathcal{Z}_2$, respectively.

Then, given D_{z_1,z_2}, we can confirm two sample sets are from the same distribution with a certain significance level β, if $\sqrt{\frac{n_1 n_2}{n_1+n_2}} D_{z_1,z_2} \leq K_\beta$, where K_β can be set according to a well-defined table [17]. Note that *cdf* is a byproduct of the KDE, denoted as $\tilde{F}(k)$ in Algorithm 1. After KS test, we combine all the data sets passing the tests into one single data set, which is provided for the KDE to update density estimates $\tilde{f}(k)$ for the current slot. To tradeoff the performance improvement and computational overhead, we limit the number of KS tests by only preserving the previous n_w windows of data sets for each channel. Meanwhile, two consecutive windows only differ with one data point, thus it becomes more beneficial to test windows with interval of t_w samples. In this way, every previous window passing the KS test can export t_w more samples into the merged data set (see line 10 of Algorithm 1).

Consequently, we derive an accurate density estimate for active slot interarrival time distribution at each channel, by online learning of data dynamics and cumulative combination of historic data.

3) Computing Slotted Channel Access Probability: The problem we are going to address in this section is how to estimate the $SCAP$ based on the predicted distribution of active slot interarrival time. As mentioned before, $SCAP(k+1)$ represents the probability that $(k+1)$-th slot is active. In theory, the predicted secondary user $SCAP$ at $(k+1)$-th slot should be represented as $SCAP(k+1) = Pr(X_{k+1} = 1 | X_1, \ldots, X_k)$, whose computation appears intractable since it takes all the historic channel states into consideration. However, we can take advantage of the updated active slot interarrival time distribution to simplify the computation. We note that if the current slot is active, the current active slot interarrival time will be the time period between the current slot and the most recent active slot. Consequently, the probability that current slot is active $SCAP(k+1)$ can be interpreted as the probability that the active slot interarrival time is equal to the time period between the current slot and the preceding

Algorithm 1 Online non-parametric density estimation protocol

1: **Input**: W, n_w, t_w, current sensing result X_k at k-th sensing slot.
2: If $X_k! = 0$
3: Calculate the new observed active slot interarrival time $T_{int}(k)$;
4: Update the current data set $\mathcal{Z}(k) = \{T_{int}(k), \ldots, T_{int}(k-W+1)\}$;
5: Update the input data set $\mathcal{Z}_{in} = \mathcal{Z}(k)$;
Update the current density estimate $[\tilde{F}(k), \tilde{f}(k)] = \mathsf{KDE}(\mathcal{Z}_{in})$;
6: for i$\leftarrow$ 1 to n_w
7: Perform KS test: $\mathsf{KStest}(\mathcal{Z}(k), \mathcal{Z}(k - i \cdot t_w))$;
8: If pass KS test
9: Update the input data set $\mathcal{Z}_{in} = \{\mathcal{Z}(k) \cup \mathcal{Z}(k - i \cdot t_w)\}$;
10: end
11: Update the current density estimate $[\tilde{F}(k), \tilde{f}(k)] = \mathsf{KDE}(\mathcal{Z}_{in})$;
12: else return.

Algorithm 2 The Computation of Slotted Channel Access Probability

1: **Input**: current density estimate $\tilde{f}(k)$, current sensing result X_k, the sensing slot length Δ.
2: **Initialization**: $IdleCount = 1$
3: If $X_k! = 0$
4: Compute $SCAP(k+1) = \int_0^{\Delta} \tilde{f}(k)dt$;
5: Reset $IdleCount = 1$;
6: else
7: Update $IdleCount = IdleCount + 1$;
8: Compute $SCAP(k+1) = \int_{(IdleCount-1)\cdot\Delta}^{(IdleCount\cdot\Delta)} \tilde{f}(k)dt$;
9: end

active slot. If we assume the preceding active slot is k, $SCAP(k+1) = Pr(X_{k+1} = 1|X_k = 1)$ with active slot interarrival time becoming Δ. If we assume the preceding active slot is j, $SCAP(k+1) = Pr(X_{k+1} = 1|X_k = 0, \ldots, X_{j+1} = 0, X_j = 1)$ with active alot interarrival time becoming $(k+1-j)\cdot\Delta$. Therefore, the predicted $SCAP(k+1)$ can be written as follows:

$$SCAP(k+1) = \begin{cases} Pr(X_{k+1} = 1|X_k = 1), \textbf{if } X_k = 1, \\ Pr(X_{k+1} = 1|X_k = 0, \ldots, X_{j+1} = 0, X_j = 1), \\ \textbf{if } X_k = 0. \end{cases}$$
$$= \begin{cases} \int_0^{\Delta} \tilde{f}(k)dt, \textbf{if } X_k = 1 \\ \int_{(k-j)\cdot\Delta}^{(k+1-j)\cdot\Delta} \tilde{f}(k)dt, \textbf{if } X_k = 0, \end{cases} \quad (6)$$

where Δ is defined as the sensing slot length. The algorithm to compute the $SCAP$ for each slot is given in Algorithm 2. $SCAP$ provides an appropriate measure for quantifying the secondary user channel access pattern, which takes into account the channel availability, SUs' current activity, and SUs' traffic pattern learnt from their past activities. The major goal of the inspection sniffers is to predict $SCAP(k+1)$ that guides the operation sniffers' channel assignment strategies, which is the main focus of the following section.

V. Near-Optimal Monitoring Mechanism

The monitoring mechanism of SpecMonitor addresses the problem of sniffer channel assignment to maximize two different levels of QoMs, which is carried out by the sniffer center. In particular, at k-th slot, the sniffer center collects all the channel usage information gathered by the inspection sniffers to produce a prediction set of $SCAP(k+1)$ for all the channels simultaneously. This set of predicted $SCAP$ is then leveraged to provide optimized channel assignments for the forthcoming slot.

Although channel switching enables the sniffers to capture channel dynamics adaptively, its negative effects should not be neglected in computing QoMs, especially in the CRNs with channel availability issue. We claim that channel switching indeed produces non-negligible overhead in terms of frame losses in practice. In the following, we show our formulation of sniffer channel assignment problem with two levels of QoMs, respectively.

A. Frame-level Quality-of-Monitoring Optimization

The goal of FL-QoM optimization is to maximize the number of captured frames, given a set of channels and operation sniffers inside one monitoring area. In section III, we show that active slot interarrival pattern is closely associated with frame arrival pattern, so that the number of captured frames during K slots from a certain channel can be written as: $N_f = \sum_{k=0}^{K}(\mathbb{I}_k \cdot n_f^{(k)})$, where $\mathbb{I}_k$ is an indicator indicating whether the k-th slot is active, $n_f^{(k)}$ denotes the number of frames inside the k-th slot. Therefore, instead of directly maximizing the number of captured frames, we transform FL-QoM into an objective of maximizing the number of active slots captured. For notation convenience, let us define index sets $i \in \mathcal{N} = \{1, \ldots, N\}$, $s \in \mathcal{S}_{op} = \{1, \ldots, M\}$ for indexing channels and operation sniffer antennas respectively. The optimization problem can be formulated as the following integer programming (IP) problem:

$$\text{maximize} \quad \sum_{i=1}^{N} SCAP_i(k+1) \cdot y_i(k+1) \quad (7)$$
$$- \alpha \sum_{s=1}^{M}\sum_{i=1}^{N} \frac{1}{2}[z_{s,i}(k+1) - z_{s,i}(k)]^2$$
$$\text{subject to} \quad \sum_{i=1}^{N} z_{s,i}(k) \leq 1, \forall s \in \mathcal{S}_{op}, \forall k \quad (8)$$
$$\sum_{s=1}^{M} z_{s,i}(k) \leq 1, \forall i \in \mathcal{N}, \forall k \quad (9)$$
$$y_i(k) = \sum_{s=1}^{M} z_{s,i}(k), \forall i \in \mathcal{N}, \forall k \quad (10)$$
$$y_i(k), z_{s,i}(k) \in \{0,1\}, \forall s \in \mathcal{S}_{op}, i \in \mathcal{N}, \forall k. \quad (11)$$

Each operation sniffer antenna in the set $\mathcal{S}_{op}$ is associated with a binary decision vector $z_{s,i}(k) \in \{0,1\}$, $i \in \mathcal{N}$, which is called sniffer channel assignment indicator, with $z_{s,i}(k) = 1$ if the sniffer is assigned to channel i at slot k; 0 otherwise. $y_i(k+1)$ is the binary variable indicating whether or not the channel i is monitored by some sniffer in (k+1)-th slot. The IP formulation is supposed to run iteratively: at k-th slot, after obtaining $z_{s,i}(k)$ and predicted $SCAP_i(k+1)$, we can acquire $y_i(k+1)$ and $z_{s,i}(k+1)$ by solving the IP problem. Clearly, the sniffer channel assignment is updated once every slot, which allows our mechanism to quickly adapt to the traffic dynamics.

Note that the objective function Eq. (7) is comprised of two parts: the positive part represents the average number of captured active slots, while the negative part indicates the channel switching costs. For simplicity, we use the number of channel switches between every two subsequent slots to approximate the channel switching costs. In addition, we set a *switching cost weight* α to represent the relative significance of channel switching costs w.r.t. the gains obtained from captured slots, which is a constant value residing within $[0,\ 1]$. Here, we define α as the ratio of channel switching duration to the slot duration. If channel switching takes $5ms$ and one slot

has $20ms$, we have $\alpha = 1/4$. However, the definition of α can be further extended to incorporate more sophisticated metrics for channel switching costs. For instance, we can further incorporate the probability that the current channel will be idle in the next slot, because sniffer's channel switching does not incur frame loss overhead if the sniffer listens on an expected idle channel.

The constraints (8), (10) arise due to the facts that one sniffer antenna can only monitor one channel, and one channel is better to be covered by one sniffer antenna *inside the monitoring area*. In particular, we put forward the second constraint, because if we allow multiple antennas to listen over the same channel in the same area, their captured frames will provide duplicate information. This IP problem can be viewed as a NP-hard problem following the proof in [8], thus we need to find an approximation algorithm to solve the IP problem.

LP rounding algorithm has been adopted to solve the IP problem [7], [8]. This algorithm solves the LP-relaxation of the IP formulation, and then rounds the fractional results into integral solutions using for example the *probabilistic rounding algorithm* (PRA) [18]. However, this algorithm is only applicable to *linear program* problem, while in our formulation, the objective function contains some quadratic terms. We then reformulate the objective function to remove the nonlinear terms. As $z_{s,i}(k)^2 = z_{s,i}(k)$ when $z_{s,i}(k) \in \{0, 1\}$, the objective function Eq. (7) can be rewritten into a linear form as follows:

$$\sum_{i=1}^{N} SCAP_i(k+1) \cdot y_i(k+1) - \alpha \sum_{s=1}^{M} \sum_{i=1}^{N} \frac{1}{2}[z_{s,i}(k+1) + z_{s,i}(k) - 2z_{s,i}(k) \cdot z_{s,i}(k+1)]$$

Note that $z_{s,i}(k)$ is already known before solving optimization problem. The PRA algorithm has been proven [18] to produce $(1 - 1/e)$-optimal sniffer channel assignment in linear time. However, the execution of PRA disregards the constraint (10) completely. Hence, the resulted channel assignment obtained from PRA cannot prevent multiple antennas from listening on the same channel. We define this problem as *channel conflict problem*, and the sniffer antennas assigned to the same channel as *conflict sniffer set*.

In response, we propose a heuristic *sniffer fixing* strategy to address the channel conflict problem, which takes the following steps:

(1) Find all the conflict sniffer sets in the solution obtained from the PRA algorithm;

(2) Pick one sniffer antenna in each conflict sniffer set randomly, and fix it to the conflicted channel;

(3) Run LP rounding algorithm again to get a new solution;

(4) Test whether the new solution contains any conflict sniffer set: if yes, go to step (1); otherwise return the solution.

The above heuristic channel assignment strategy fixes one sniffer antenna to one channel every round by adding constraints, thus it guarantees to provide a feasible solution of channel assignments for all the sniffers within linear time, which turns out to be a near-optimal solution for the sniffer channel assignment problem, as shown in Section V-C. In the end, all the confliction will be addressed after running through a sequence of LP rounding, which guarantees the convergence of the algorithm.

We call the channels to be assigned as *potential channels*. The resulted channel assignment strategy can provide the sniffers with the assignments of potential channels for the next slot. Then, the sniffer center checks every potential channel to determine whether it has already been monitored: if yes, it skips assigning this channel; if no, it selects a sniffer antenna which is not listening on any other potential channels to monitor this channel. In this way, the channel switching costs are further alleviated.

B. User-level Quality-of-Monitoring Optimization

The objective of UL-QoM optimization is to maximize the expected number of active users monitored. In order to capture the user-level information, it is indispensable to identify the source of each frame, even encrypted frames. Let $U_i(k)$ for $i \in \mathcal{N}$ denote the number of active users operating in channel i at the k-th slot. We assume once a sniffer is tuned into a channel, it covers all the active users operating in this channel. We do not consider the channel switching costs in this case, because there are typically multiple frames from a single user so that a small number of frame loss due to channel switching does not have a big impact on the number of users measured. The UL-QoM optimization problem can be casted as the following IP problem:

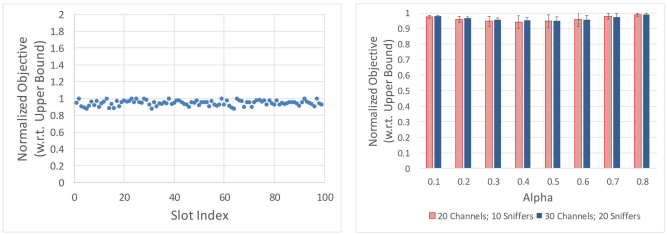
$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} U_i(k) \cdot SCAP_i(k+1) \cdot y_i(k+1) \qquad (12) \\ \text{subject to} \quad & (8) - (11). \end{aligned}$$

The above optimization problem can be solved using exactly the same approximation algorithm illustrated in the previous section, thus is omitted here. Note that $U_i(k)$ can be measured by counting the number of different MAC addresses from frames passing through the AP running in channel i within time slot k. In practice, $U_i(k)$ may not be available at the beginning of the k-th slot, so it can be approximated by the measurement of $U_i(k-1)$, assuming users remain operating in the same channel for the next time slot. Small errors in estimating $U_i(k)$ would not affect the performance much. In the extreme case when false MAC addresses are inserted by the attackers, more sophisticated approach is required. For instance, machine learning methods to perform Internet traffic classification [19] can be used to differentiate different users based on their identified traffic types. This is out of the scope of this paper.

C. Numerical Analysis

In this section, we present numerical results for our approximation algorithm and compare them to the upper bound of the problem. Without loss of generality, we focus on the FL-QoM optimization problem.

Deriving an Upper Bound: The complexity of the optimization problem formulated in Section V-A stems from



(a) Normalized objective (with respect to the computed upper bound) for 20 channels and 10 sniffers with α=0.3

(b) Normalized objectives for different α

Fig. 4: Numerical analysis

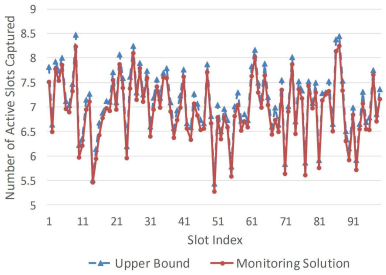


Fig. 5: Number of captured active slots using our algorithm vs. Upper bound for 20 channels and 10 sniffers with α=0.3

the binary $y_i(k)$ and $z_{s,i}(k)$ variables, for $\forall$ k. To derive an upper bound for the problem, we relax the integer (binary) requirement on $y_i(k)$ and $z_{s,i}(k)$ with $0 \leq y_i(k) \leq 1$ and $0 \leq z_{s,i}(k) \leq 1$. The relaxed problem is a standard LP problem, the solution of which can be obtained in polynomial time. Since the relaxation enlarges the optimization space, the solution to the relaxed LP problem yields an upper bound for the original optimization problem.

Numerical Results: We consider $N = 20$ or 30 channels, $M = 10$ or 20 sniffer antennas. $SCAP$ values are randomly generated for every channel over 1000 slots. We first present the simulation results for 20 channels and 10 sniffer antennas. We used the PRA approximation and sniffer fixing algorithms to determine a feasible solution which serves as a low bound, and compared the corresponding objective value with the upper bound. Fig. 4(a) shows the normalized objective values with respect to the computed upper bound (i.e. feasible solution/upper bound) for 20 channels and 10 sniffer antennas. The average normalized objective value obtained among 1000 slots is 0.95 and the standard deviation is 0.03. We further adjust switching cost weight α to examine the variations of the normalized objective values. Fig. 4(b) shows the tiny gap between the achieved solution and the upper bound for different α.

Since the actual optimal value lies between the feasible solution value and the upper bound, the solution value of our approximation algorithm must be even closer to the optimal value than the foregoing normalized ratio (normalized objective value). Thus, the derived solution value of our approximation algorithm is close to optimality, confirming its near-optimality. Finally, we run an experiment to compare the average number of captured active slot using our algorithm with the upper bound in Fig. 5. We notice that the difference between the monitoring solution and upper bound is kept small over the time, which proves the near-optimality of our approximation algorithm from the experimental perspective.

D. Complexity Analysis

In this section, we analyze the complexity of the above approximation algorithms. We first analyze the complexity of LP rounding algorithm. The LP rounding algorithm involves two major steps: (1) solving LP relaxation, and (2) executing PRA algorithm. We notice that the above two IP problems contain $(N + MN)$ unknown variables. Therefore, the complexity of solving the LP relaxation of IP formulation is given as $O((N + MN)^3/log(N + MN))$ [7], which is determined by the complexity of LP solver. On the other hand, the PRA algorithm has a linear complexity $O(M * N)$, governed by the input vector size $(M * N)$ [18]. Thus, the LP rounding algorithm can be solved with polynomial time complexity $O((N + MN)^3/log(N + MN))$.

Second, the heuristic sniffer fixing strategy will solve channel conflict problem by running through a series of LP rounding algorithm. In the worst case scenario, it will invoke LP rounding M times. Hence, the sniffer channel assignment problem can be solved with an overall worst case complexity of $O(M*(N+MN)^3/log(N+MN))$. The efficiency evaluation of the algorithm implementation is presented in Section VI-A.

VI. Evaluation

In this section, we conduct extensive simulations and experiments to evaluate the performance of SpecMonitor for CRNs. The simulations leverage synthetic traces, which allow us to vary the number of channels and sniffers, as well as the traffic patterns of different users. We also carry out experiments and test the performance of SpecMonitor on real traces collected from the experiments. Aside from implementing the proposed SpecMonitor framework, we also implemented the following algorithms for comparison.

- **Random channel assignment:** the sniffer channels are randomly assigned.
- **Greedy channel assignment:** the sniffers are always assigned to the predicted busiest channels based on $SCAP$ at every sensing slot, i.e. the channels with the largest $SCAP$.
- **Support Vector Regression (SVR) channel assignment:** the sniffers are assigned to the channels in which the next frame is predicted to arrive within a short period based on the frame interarrival time predication using SVR method [10].

We assume the PUs' presence can be detected promptly by both inspection and operation sniffers, as illustrated in section IV-A. The default systematic parameters used in the evaluation are shown in the Table II.

A. Real-Time Monitoring Performance

As illustrated in Section IV,V, our monitoring framework has a very stringent real-time requirement. Basically, we are required to complete the channel assignments before a slot ends, i.e., within 20 ms according to our setting. In this section, we evaluate the running time of SpecMonitor using

Parameters	Values
W	50
n_w	10
t_w	25
Sensing slot length Δ	20ms
Sensing period	2ms
Multi-slot updating	5 slots
Gaussian mean values of synthetic traces	[3, 42]
Gaussian standard deviation of synthetic traces	2

TABLE II: Parameters

experiments, and propose to relax the stringent requirement without compromising the monitoring performance. We implement SpecMonitor framework using MATLAB R2011b on a Windows machine with 3.2 GHz Intel Xeon W3565 CPU and 18 GB memory, including the channel access model and near-optimal channel assignment algorithm. In our original design, whenever there is a new observation of active slot interarrival time, SCAP values are updated and channels are reassigned.

We carry out experiments to count the running time of the monitoring framework and breakdown the running time into different sections to identify the bottleneck as shown in Table. III. Note that the recorded running time is the average value after running 1000 slots. The overall running time is 91.2 ms, with 97.5% of time spent on KDE operation and optimization algorithm. By delving into the KDE function and optimization algorithm, we find the bottleneck of KDE operation is on the fixed point iteration algorithm [15] consuming 95% of overall time for each KDE invocation, while the bottleneck of optimization algorithm is on the linprog MATLAB function (52%) and LP rounding algorithm with sniffer fixing strategy (40%). In our experiment, the sniffer fixing strategy runs through LP rounding algorithm *two times* on average until a valid channel assignment is generated. Consequently, the overall running time far exceeds a slot duration. To address this issue, we can convert the code using more computationally efficient programming language such as C.

Another alternative and more viable approach is to relax the stringent real-time requirement. Instead of updating SCAP and assigning channels every slot, we relax the per-slot updating requirement into T-slot updating requirement, which allows SCAP to be renewed every T slots. To satisfy the relaxed requirement, the sniffer center only needs to check for new observations and incorporate them in the channel usage model every T slots. In other words, SCAP gets updated and channels are reassigned, only if there is at least one new observation during T slots. In our implementation, we can set $T = 5$, so that the implemented model update and channel assignment complete before the next channel assignment can be carried out, i.e., within $100ms$ or 5 slots (as $91.2ms < 100ms$). We show in the following sections that per 5-slot updating does not sacrifice the monitoring performance much compared with per-slot updating, thus it satisfies the real-time processing while retaining an excellent performance. Note that we neglect the wire side communication costs between inspection sniffers and sniffer center, which are in the order of microsecond, regarded as negligible.

KDE Operation	**KStest Operation**	**SCAP Computation**	**Optimization Algorithm**	**Total Time**
58.1	0.6	0.6	30.8	91.2

TABLE III: Average running time with 20 channels and 10 sniffer antennas (in ms)

B. Frame Capturing Performance

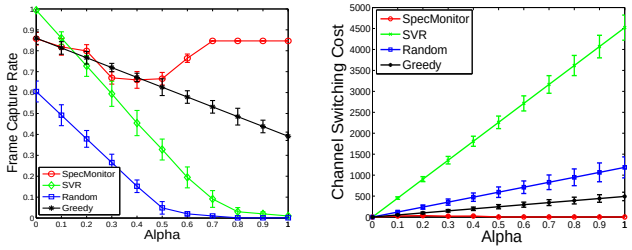
In this section, frame capturing performances of different channel assignment algorithms are evaluated. First, we generate synthetic traces to evaluate the *frame capture rate* and channel switching cost. *Frame capture rate* is defined as the ratio of the number of captured frames versus the overall number of frames passing through all the channels up to the current time, while channel switching cost is represented by the negative part of Eq. 7. Then, we collect real-world traces using AirPcap Nx [20] with Wireshark. The traffic traces are captured from multiple channels of operative WLANs (802.11g mode) to emulate the scenarios in WhiteFi networks. We evaluate SpecMonitor by comparing its frame capturing performance with other algorithms. For all the following evaluations, we measure the performance of algorithms running through 1000 slots for 100 rounds.

1) Synthetic Traces: First, we generate synthetic time series traces to represent frame interarrival time, using Gaussian distribution with an exponential correlation function. Each trace corresponds to the traffic generated in one channel with different mean values to simulate different traffic loads. We evaluate the capturing performance w.r.t. different switching cost weights α. Generally speaking, one channel switching causes a penalty of losing α slot, $\alpha \in [0, 1]$.

We assume the training process of SVR scheme has already been done, which takes about 35 training samples [10]. Fig. 6(a) shows the frame capture rates of different methods. With the increase of α, frame capture rates of all three compared schemes fall down steadily because of the increasing penalty for channel switching. However, with excessive α, SpecMonitor will force the sniffers to switch channel only when the reward from channel switching is higher than the penalty; otherwise, it keeps the sniffers staying in the current channels. In this way, SpecMonitor retains an excellent frame capture performance. Regarding the SVR scheme, it performs best when the channel switching costs are neglected ($\alpha = 0$ or 0.1), which means SVR achieves accurate estimations of frame interarrival time. However, when α grows larger than 0.2, frame capture rate of SVR scheme drops steadily because of the aggravating switching penalty caused by frame loss. Note that the capturing performance of SpecMonitor also drops a bit due to higher switching costs, but then reverts back to surpass the performance curves of any other schemes.

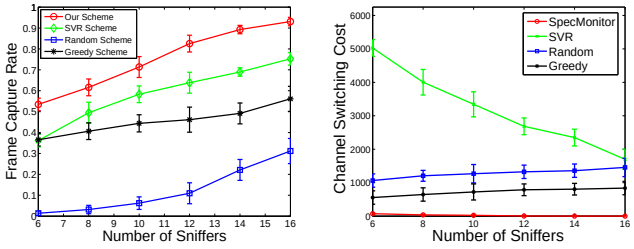
Fig. 6(b) shows channel switching costs w.r.t. α, from which we can see SVR methods induce highest switching costs among all methods, because of its unslotted and heuristic switching strategy. In contrast, the switching cost of SpecMonitor remains the lowest.

Finally, Fig. 7(a) shows the different capturing capabilities w.r.t. the number of sniffer antennas, the frame capturing performance of all the methods keeps growing with the increasing number of antennas. SpecMonitor achieves the highest frame



(a) Frame Capture Rates (b) Channel Switching Costs

Fig. 6: Performance with different methods using synthetic time series data (5 channels, 3 sniffer antennas)

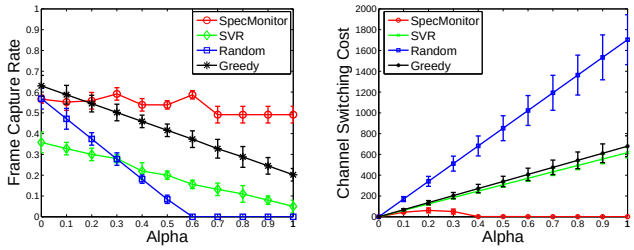


(a) Frame Capture Rates (b) Channel Switching Costs

Fig. 7: Performance with varied number of sniffer antennas using different methods (α=0.25, 20 channels)

capture rate. We also compare channel switching costs in Fig. 7(b). With more sniffer antennas, the channel switching costs of SVR method decreases significantly, because the increased traffic capturing capability refrains SVR method from aggressive channel switching behavior. Meanwhile, the other methods have much less, yet more stable channel switching costs.

2) Real Traces: We collect the real traces from 802.11g WLAN network, captured by a sniffer listening on the channel established by one AP and client pair running various applications. The captured traces include both the uplink traffic to AP and the downlink traffic from AP. We consider five different types of trace data (FTP, BT, Web Browsing, Skype Voice and Skype Video) with one trace per channel. FTP and BT traces are obtained by running an automated script on the client to download/upload several files from/to a server continuously, and we write another automated script to browse several websites to collect Web Browsing trace. Skype voice trace and video trace are collected by connecting to a client using Skype voice call or Skype video call. We evaluate the performance with seven channels of real-world traffic, while the additional two channels contain mixed traffic pattern. Namely, one is the traffic combined from two clients using Skype Voice and BT, and the other one is generated from two clients using Skype Voice and Web Browsing. The performance is shown in Fig. 8(a) and Fig. 8(b), from which we can see SVR method performs even worse than random scheme. The reason is that the SVR method takes a long time for retraining, when the predicted value has a large deviation from the genuine one because of the real-world traffic dynamics. Frequent retraining and channel switching operations significantly deteriorate the



(a) Frame Capture Rates (b) Channel Switching Costs

Fig. 8: Performance using real-world traffic (7 channels, 4 sniffer antennas)

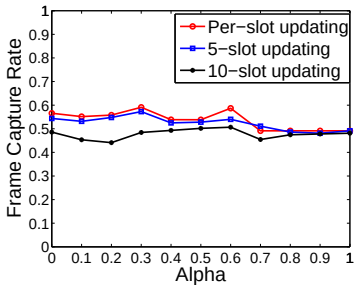
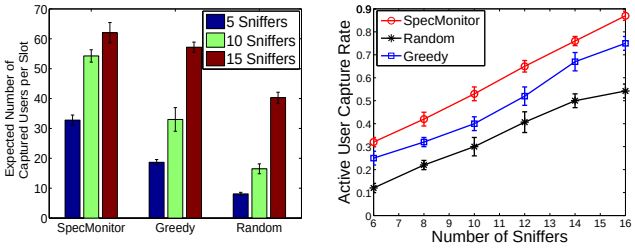


Fig. 9: Average frame capture rate comparison of multi-slot updating and per-slot updating with 20 channels and 10 sniffer antennas

capturing capability of SVR method. However, SpecMonitor retains the best performance, except in the case of small α, the greedy method performs better when channel switching only incurs a small penalty. This comparison result also indicates that our model can accurately capture the traffic statistics regardless of whether the traffic is interleaved or not.

3) Comparison of multi-slot updating and per-slot updating: As mentioned in Section VI-A, multi-slot updating relaxes the real-time requirement. In this section, we compare the performance of multi-slot updating with per-slot updating. Fig. 9 presents the frame capturing performance comparison for multi-slot updating and per-slot updating, which shows a slight performance degradation using multi-slot updating method. Interestingly, 5-slot updating achieves a better frame capture performance when $\alpha = 0.7$, because it incurs less switching costs by switching at least every 5 slots. However, in most cases, per-slot updating captures more frames, due to its more rapid adaptation to the traffic dynamics. From this performance comparison, we conclude that 5-slot updating retains an excellent frame capturing performance while fulfilling the real-time requirements as presented in Section VI-A.

The intuition behind the fact that T-slot updating achieves similar results is that the data distribution presented in our experiments does not change rapidly over the course of T slots. However, this fact does not apply to all the traffic scenarios. For example, for traffic scenarios when the traffic statistics are rapidly changing, T should be assigned a small value. The exact value of T can be picked using the above performance comparison method, via evaluating the performance of various T values and selecting a larger one with acceptable perfor-



(a) User-Level QoM Objective (b) Active User Capture Rate

Fig. 10: User capture performance with 20 channels

mance degradation.

C. User Capturing Performance

Finally, we evaluate the performance for maximizing UL-QoM using synthetic data. We assume different channels contain different numbers of SUs, and the numbers are dynamically changing within range $[0, 10]$ (assume uniform distribution); also the frame interarrival time is exponentially distributed with mean values residing in $[1, 40]$, specifying the traffic pattern. First, we compare the expected number of captured users per slot using three different monitoring schemes in Fig. 10(a). The result indicates that SpecMonitor is able to capture more users per slot, because the optimized monitoring strategy keeps the sniffers watching the channels with more users.

Then, we define *Active User Capture Rate* as the ratio of number of active users captured versus the overall number of the active users appeared in all the channels. The performance of active user capture rate w.r.t. different number of sniffers is shown in Fig. 10(b), from which we notice that SpecMonitor can select best sets of channels to maximize the number of active users captured during the monitoring period. The result implies SpecMonitor significantly outperforms two baseline schemes, in terms of user capturing performance.

VII. Conclusion

In this paper, we have introduced a systematic passive monitoring framework, SpecMonitor, for Wi-Fi like CRNs to maximize two levels of QoMs incorporating switching costs. Both the primary user and secondary user channel usage patterns are considered to optimize the monitoring strategy. Specifically, we proposed an online non-parametric density estimation scheme to learn and predict the time-evolving mixed traffic pattern from SUs. Based on the predicted traffic pattern, the optimization problems of sniffer channel assignment are formulated, for which we designed near-optimal monitoring algorithms. One major limitation of the SpecMonitor system is that SpecMonitor requires a substantial amount of traffic of interest on the channel in order to produce a reasonable channel access model. If the traffic amount over a channel is small, the produced model may be unreliable. In future research, we will consider the impact of traffic amount to the channel access model. We plan to evaluate the modeling accuracy in real time. The model built with a small traffic amount will be deemed as unreliable, which will not be used for future predictions and channel assignments.

Acknowledgement

We thank Shaxun Chen, Kai Zeng, Fan Zhang for their useful inputs and discussions. We thank Huy Nguyen from University of Houston for providing the source code for probabilistic rounding algorithm. We appreciate anonymous reviewers for their helpful comments. This work was supported in part by NSF Grants 1405747, 1343222, 1247830, 1217889 and ONR Grant N000141310080. The work of T. Jiang was supported by an NSF Graduate Research Fellowship.

References

[1] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," *IEEE Journal on Selected Areas in Communications*, vol. 25, April 2007.

[2] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh, "White space networking with wi-fi like connectivity," in *SIGCOMM '09*, August 2009, pp. 27–38.

[3] J. Yeo, M. Youssef, and A. Agrawala, "A framework for wireless LAN monitoring and its applications," in *Wise 2004*, October 2004, pp. 70–79.

[4] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw: Solving the puzzle of enterprise 802.11 analysis," in *SIGCOMM '06*, Sep. 2006, pp. 39–50.

[5] Y.-C. Cheng, M. Afanasyev, P. Verkail, P. Benko, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker, "Automating cross-layer diagnosis of enterprise wireless networks," in *SIGCOMM '07*, Aug. 2007, pp. 25–36.

[6] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan, "Characterizing user behavior and network performance in a public wireless LAN," in *SIGMETRICS 2002, ACM*, June 2002, pp. 195–205.

[7] D.-H. Shin and S. Bagchi, "Optimal monitoring in multi-channel multi-radio wireless mesh networks," in *MobiHoc'09*, May 2010.

[8] A. Chhetri, H. Nguyen, G. Scalosub, and R. Zheng, "On quality of monitoring for multi-channel wireless infrastructure networks," in *MobiHoc'10*, Sep. 2010.

[9] P. Arora, C. Szepesvari, and R. Zheng, "Sequantial learning for optimal monitoring of multi-channel wireless networks," in *INFOCOM 2011, IEEE*, 2011.

[10] S. Chen, Z. Kai, and P. Mohapatra, "Efficient data capturing for network forensics in cognitive radio networks," in *Network Protocols, 2011. (ICNP '2011) 19th International Conference on*, 2011, pp. 176–185.

[11] S. Yi, K. Zeng, and J. Xu, "Secondary user monitoring in unslotted cognitive radio networks with unknown models," in *Wireless Algorithms, Systems, and Applications*, ser. Lecture Notes in Computer Science, vol. 7405, 2012, pp. 648–659.

[12] D. Cabric, A. Tkachenko, and R. W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*, ser. TAPAS '06, 2006.

[13] D. Murray, M. Dixon, and T. Koziniec, "Scanning delays in 802.11 networks," in *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies*, Sept. 2007, pp. 255–260.

[14] S. Narlanka, R. Chandra, P. Bahl, and J. I. Ferrell, "A hardware platform for utilizing tv bands with a wi-fi radio," in *IEEE LANMAN*, June 2007.

[15] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *Annals of Statistics*, vol. 38, no. 5, pp. 2916–2957, Nov. 2010.

[16] C. Heinz and B. Seeger, "Towards kernel density estimation over streaming data," in *International Conference on Management of Data (COMAD)*, Dec. 2006.

[17] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New York, USA: Wiley, 2009.

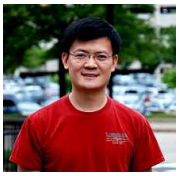
[18] A. Srinivasan, "Distributions on level-sets with applications to approximation algorithms," in *FOCS*, 2001.

[19] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *WiSec 2011*, June 2011, pp. 59–69.

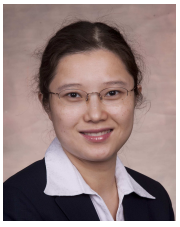
[20] "Airpcap Adapter," http://www.riverbed.com/products-solutions/products/network-performance-management/wireshark-enhancement-products/.

Qiben Yan (S'11) received his B.E. and M.E. in Electrical Engineering at Fudan University, China, in 2007 and 2010, respectively. He is currently a Ph.D student in Computer Science department at Virginia Tech. His current research interests include wireless network security and privacy, network monitoring and forensics.

Ming Li (S'08 - M'11) received his Ph.D. in Electrical and Computer Engineering from Worcester Polytechnic Institute. He joined the Computer Science Department at Utah State University as an assistant professor in 2011. His research interests are in the general areas of wireless network and cyber security, with current emphases on network coexistence, wireless physical layer security, data security and privacy, and cyber-physical system security. He is a recipient of the NSF CAREER Award in 2014. He is a member of both IEEE and ACM.



Feng Chen is an assistant professor at University at Albany, SUNY. He received his B.S. from Hunan University, China, in 2001, M.S. degree from Beihang University, China, in 2004, and Ph.D. degree from Virginia Tech in 2012, all in Computer Science. He has published 30 referred articles in major data mining venues. His research focuses on the detection of emerging events and other relevant patterns (e.g., disease outbreaks, crime events, road traffic congestion) in the mobile context and/or data mining of spatial temporal, textual, or social media data.



Tingting Jiang (S'11) received her B.S. degree (Summa Cum Laude) in Computer Science from Virginia Tech, Blacksburg, VA, in 2007. During 2007-2009, she was a Software Engineer at Intrexon Corp., Blacksburg, VA. Currently, she is a Ph.D. student in Computer Science at Virginia Tech, Blacksburg, VA. She is a recipient of an NSF Graduate Research Fellowship (2011 C 2014) and a Microsoft Research Graduate Women's Scholarship (2011). Her research area is in Wireless Security.

Wenjing Lou (S'01-M'03- SM'08) is a professor in the Computer Science department at Virginia Polytechnic Institute and State University. She received her Ph.D. in Electrical and Computer Engineering from University of Florida. Her research interests are in the broad area of wireless networks, with special emphases on wireless security and cross-layer network optimization.

Y. Thomas Hou (F'14) is a Professor in the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA. He received his Ph.D. degree from NYU Polytechnic School of Engineering (formerly Polytechnic Univ.). His current research focuses on developing innovative solutions to complex cross-layer optimization problems in wireless and mobile networks. He has published two graduate textbooks: Cognitive Radio Communications and Networks: Principles and Practices (Academic Press/Elsevier, 2009) and Applied Optimization Methods for Wireless Networks (Cambridge University Press, 2014). The first book has been selected as one of the Best Readings on Cognitive Radio by IEEE Communications Society. For his research accomplishments, Prof. Hou was awarded Virginia Tech College of Engineering Deans Award for Excellence in Research (2013) and Dean's Faculty Fellow Award (2008). He is currently an Area Editor of IEEE Transactions on Wireless Communications (overseeing a team of 10 editors in wireless networks area), and Editor for IEEE Transactions on Mobile Computing, IEEE Journal on Selected Areas in Communications (Cognitive Radio Series), and IEEE Wireless Communications. He is the Steering Committee Chair of IEEE INFOCOM conference.

Chang-Tien Lu received the MS degree in computer science from the Georgia Institute of Technology in 1996 and the PhD degree in computer science from the University of Minnesota in 2001. He is an associate professor in the Department of Computer Science, Virginia Polytechnic Institute and State University and is founding director of the Spatial Lab. He served as Program Co-Chair of the 18th IEEE International Conference on Tools with Artificial Intelligence in 2006, and General Co-Chair of the 20th IEEE International Conference on Tools with Artificial Intelligence in 2008 and 17th ACM International Conference on Advances in Geographic Information Systems in 2009. He also served as Vice Chair of the ACM Special Interest Group on Spatial Information (ACM SIGSPATIAL) from 2011 to 2014. His research interests include spatial databases, data mining, geographic information systems, and intelligent transportation systems.