Non-parametric Passive Traffic Monitoring in 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, VA, USA
† Utah State University, Logan, Utah, USA

***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 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 our proposed framework outperforms the existing schemes significantly.**

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.

Along with the technical innovations to enable practical CR communications, the security of CR networks has recently aroused many interests [3], most of which focused on securing the enabling technologies such as spectrum sensing [4], incumbent identification, etc. On the other hand, malicious network activities, such as false data injection attacks, Denial of Service (DoS) attacks, have posed serious threats to wireless networks, which will result in significant performance degradation of CR communications. However, the detection of these activities in CRNs remains largely untouched in the literature. Network forensics is a viable method to detect anomalous network behaviors through traffic monitoring and analysis. As the quality of network forensics mainly depends on that of traffic monitoring, it is non-trivial to build a traffic monitoring framework with excellent monitoring performance. *Passive monitoring* has been used to measure Wi-Fi networks [5]–[7] 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. 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. It is therefore 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, the SUs have to vacate the channels immediately when the PUs start transmissions on the corresponding channels. This inevitable channel switching behavior of SUs potentially complicates the sniffers' strategies to capture SUs' traffic. Last but not the least, the network traffic in each channel comes from multiple SUs, who share the available spectrum by following a certain medium access control (MAC) mechanism. Thus, the traffic pattern observed by the sniffers is highly dynamic and unpredictable, further complicating the sniffers' monitoring strategies.

To meet these challenges, we utilize 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 little 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, the proposed monitoring framework 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 serve different network diagnostic problems. 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 [7]. 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 [8]. We cast the monitoring optimization problem as a sniffer channel assignment problem with objective of maximizing the corresponding QoMs.

To the best of our knowledge, this is the first work to investigate passive monitoring strategies for Wi-Fi like CR networks. In this paper, we make the following contributions:

The work of Y.T. Hou and W. Lou was supported in part by NSF grant 1247830 and ONR grant N000141310080. T. Jiang was supported by an NSF Graduate Research Fellowship.

(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 conduct extensive simulations and experiments to validate our statistical model and monitoring framework.

II. Related Work

Passive monitoring in wireless networks has been an active research area. Some recent work investigated the problem of optimal sniffer channel assignment to maximize the amount of monitored information. Shin *et al.* [9] considered to obtain optimal strategies by selecting a limited number of sniffers to monitor multiple channels in wireless mesh networks, in which they formulate the sniffer channel assignment problem as a maximal coverage problem and design approximation algorithms to solve this problem. In [10], 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.* [11] proposed to use multi-armed bandit to perform sequential learning for 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. Moreover, they only consider traditional wireless networks. Note that all the above work consider maximizing the number of active users covered by the sniffers, while we further consider to maximize the number of captured frames.

Chen *et al.* studied frame capturing problem for network forensics in CRNs [12], 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 their 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.

III. 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). As PUs' networks are regulated by service providers or specific WM users, they are out of the interests of our monitoring system. Instead, 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. In WhiteFi networks, multiple clients share their working channel decided by the AP using widely adopted CSMA/CA mechanism. We assume channel availability statistics are not available to both SUs and sniffers, so we require both entities to perform spectrum sensing periodically with assumed perfect accuracy.

Inside one *monitoring area*, we assign a certain number of sniffers to sense channels and capture packets. Each sniffer is equipped with single antenna, which allows him to sense/capture traffic over a single channel at one time. We assume different APs in the monitoring area pick different working channels to avoid interference, and the sniffers can overhear all the inbound and outbound traffic from APs inside their monitoring area. Similar to [12], 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 central server for centralized decision making. The monitoring system architecture is demonstrated in Fig. 2. 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 the following, a *slot* stands for the sensing slot unless otherwise noted.

Next, we will describe the sensing outcomes of inspection sniffers, which are closely related to the duration of a sensing slot. A sensing slot is composed of channel sensing time and channel switching time, whose length depends on the number of channels to be scanned. Typically, channel sensing time is approximately $1ms$ per channel using energy detection, while channel switching for a commodity $802.11b/g$ network card takes about $1ms - 5ms$ [13]. Therefore, we assume each inspection sniffer spends $2ms$ for sensing one channel and switching to another channel. If each inspection sniffer is assigned ten channels to scan, one sensing slot will be $20ms$ long.

Therefore, during each slot, the inspection sniffer scans ten channels to reveal their channel states (*busy/idle*). Here, $busy$ indicates the channel is occupied by SUs, while $idle$ represents the opposite. Let X_k^i be the state of channel i at k-th sensing slot, which takes only binary values "1/0", corresponding to $busy/idle$ 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 $busy$ slots. Therefore, in our design, the inspection sniffers produce the active slot interarrival time as their outcomes, which will be used as the inputs of channel usage model as explained in Section IV.

Note that one straightforward way of meeting frame capturing objectives is to predict SUs' frame arrival time by

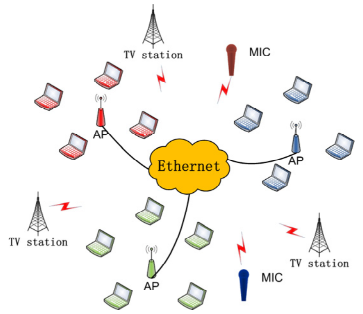


Fig. 1: Wi-Fi like Cognitive Radio Network Architecture

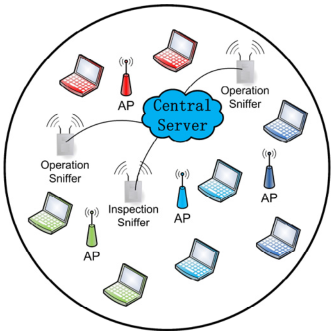


Fig. 2: Monitoring System Architecture Inside a Monitoring Area

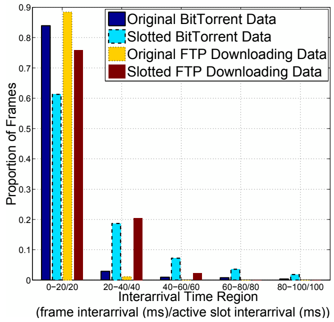


Fig. 3: The proportion of frames within certain interarrival time (assume 20ms sensing slot, 1ms sensing period)

modeling frame arrival pattern. However, it is infeasible to derive optimized channel assignments based on predicted frame arrival time [12]. Therefore, instead of directly modeling frame arrival pattern, we model the active slot interarrival pattern as the basis of our monitoring framework. The major difference between the above two patterns is that active slot interarrival pattern is slot-wise rather than frame-wise. In order to motivate/justify the adoption of active slot interarrival time, we performed a real-world experiment using BitTorrent and FTP downloading traffic in operational $802.11g$ WLAN, with results shown in Fig. 3. We can see that the proportion of frames accumulated within certain active slot interarrival time are comparable to that accumulated within the original frame interarrival time (most of the frames are concentrated within small interarrival time region), which indicates active slot interarrival time characterizes channel usage pattern as good as frame interarrival time.

Obviously, if the slot length becomes shorter, active slot interarrival pattern will approximate frame interarrival pattern more closely. However, channel scan with a shorter sensing slot requires more inspection sniffers or faster channel sensing and switching operation. Additionally, we infer from Fig. 3 that an operative $802.11g$ WLAN has a high traffic load, since most of the frame interarrival time is rather short.

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 our monitoring framework. In order to build the unified model, we first study primary user detection mechanism, 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

The sniffers are required to detect PUs' activity in order not to waste time and energy listening on the primary user occupied channels. Energy detection is the most straightforward way for the sniffers to detect PUs' appearance. We adopt the energy detection mechanism in our framework due to its simplicity. On the other hand, another alternative mechanism has a better performance but a higher complexity. It models PUs' activities observed by the inspection sniffers as an alternating renewal process, and then predicts PUs' idle probability using maximal likelihood estimation or Bayesian estimation [14]. Compared with energy detection, this mechanism provides PUs' activity information ahead of time, which can further improve the monitoring performance if the model retains a high accuracy.

B. Secondary User Channel Usage 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 usage 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. As mentioned previously, different from support vector machine and neural network based methods, density estimation method does not involve a time-consuming training phase, which makes it appropriate for online prediction. More importantly, this non-parametric approach provides a greater flexibility and accuracy in modeling a given data set, compared with other parametric approaches. 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 $f(x)$, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}$. Thus, the asymptotic optimal value of σ^* is obtained by minimizing AMISE:

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

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 σ_j^* w.r.t. $\|f^{(j)}(x)\|^2$ complies with 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^*. 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 provide 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. In other words, if the distribution of data is changing very frequently, a smaller W is more favorable for maintaining a fast reaction to the dynamic data; otherwise, a larger W can be selected to improve estimation accuracy. In the WhiteFi network scenario, we set a relatively small W, since the data distribution will change more dynamically than that in the traditional wireless networks.

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 different 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 statistics 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 statistics 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. 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. 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

Algorithm 1 Online non-parametric density estimation protocol

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

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. In theory, the predicted secondary user $SCAP$ at $(k+1)$-th slot should be represented as $SCAP(k+1) = Pr(X_{k+1} = 1|X_1, \dots, X_k)$, which takes all the historic channel states into consideration. However, in practice, the predicted $SCAP$ can be written as follows:

$$SCAP = \begin{cases} Pr(X_{k+1} = 1|X_k = 1), \textbf{if } X_k = 1, \\ Pr(X_{k+1} = 1|X_k = 0, ..., X_{j+1} = 0, X_j = 1), \\ \textbf{if } X_k = 0. \end{cases}$$

$$= \begin{cases} \int_0^{\Delta} \tilde{f}(k)dt, \textbf{if } X_k = 1 \\ \int_0^{(k+1-j)\cdot\Delta} \tilde{f}(k)dt, \textbf{if } X_k = 0, \end{cases} \qquad (6)$$

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 usage pattern, which takes into account the channel availability, SUs' current activity, and SUs' traffic pattern learnt from their recent 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 considers the problem of sniffer channel assignment to maximize two different levels of QoMs, which is carried out by the central server. In particular, at k-th slot, the central server collects all the channel usage information gathered by the inspection sniffers to produce a prediction of $SCAP(k+1)$ for all the channels simultaneously. This predicted $SCAP$ is then leveraged to provide optimized channel assignments for the next slot.

Although channel switching enables the sniffers to capture channel dynamics adaptively, its negative effects should not be neglected in computing QoMs. We claim that channel switching indeed produces non-negligible overhead in practice. For example, as mentioned previously, the $802.11b/g$ wireless cards take approximately $1-5ms$ for one channel switching operation. If one slot lasts $20ms$, at most $1/4$ frames in the slot will be missing during the channel switching operation which constitutes a non-negligible fraction of frames. To be more specific, the typical frame interarrival time in a $10Mbps$ wireless network is $0.1ms$ (assume 1000 bit frame). Then, during the $5ms$ channel switching, we may lose 50 frames, nearly $1/4$ of the total frames in this slot. In addition to that, frequent channel switching also raises energy costs. These are the reasons why we integrate channel switching costs into the optimization objectives. In the following, we show our formulation of sniffer channel assignment problem with two levels of QoMs, respectively.

Algorithm 2 The Computation of Slotted Channel Access Probability

```
Input: current density estimate $\tilde{f}(k)$, current sensing result $X_k$, the sensing slot length $\Delta$.
Initialization: $IdleCount = 1$
If $X_k ! = 0$
    Compute $SCAP(k+1) = \int_0^{\Delta} \tilde{f}(k)dt$;
    Reset $IdleCount = 1$;
else
    Update $IdleCount = IdleCount + 1$;
    Compute $SCAP(k+1) = \int_0^{(IdleCount \cdot \Delta)} \tilde{f}(k)dt$;
end
```

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 can represent 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 $i \in \mathcal{N} = \{1, \dots, N\}$, $s \in \mathcal{S}_{op} = \{1, \dots, M\}$ for indexing channels and operation sniffers respectively. The optimization problem can be formulated as the following integer programming (IP) problem:

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

$$y_i(k) = \sum_{s=1}^{M} z_{s,i}(k), \forall i \in \mathcal{N}, k \quad (10)$$

$$y_i(k), z_{s,i}(k) \in \{0, 1\}, \forall s \in \mathcal{S}_{op}, i \in \mathcal{N}, k. \quad (11)$$

Each operation sniffer 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. And $y_i(k)$ is the binary variable indicating whether or not the channel i is monitored by some sniffer. 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 $z_{s,i}(k+1)$ by solving the IP problem. Clearly, the sniffer channel assignment is updated once every slot.

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. In the previous example with $5ms$ channel switching and $20ms$ slot, we have $\alpha = 1/4$. However, the definition of α can be further extended to incorporate more sophisticated metrics for channel switching costs.

The constraints (8), (9) arise due to the facts that one sniffer can only monitor one channel and one channel is better to be covered by one sniffer. In particular, we put forward the second constraint, because if we allow multiple sniffers to listen over the same channel, their captured frames will provide duplicate information. This IP problem can be viewed as a NP-hard problem following the proof in [10], thus we need to find an approximation algorithm to solve the IP problem.

LP rounding algorithm has been adopted to solve the IP problem [9], [10]. 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:

$$\sum_{i=1}^{N} CAP_i(k+1) \cdot y_i(k+1) -$$

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

Note that $z_{s,i}(k)$ is already known before solving optimization problem. The PRA algorithm has been proven [18] to produce an optimized sniffer channel assignment in linear time. However, the execution of PRA disregards the constraint (9) completely. Hence, the resulted channel assignment obtained from PRA cannot prevent multiple sniffers from listening on the same channel. We define this problem as *channel conflict problem*, and the sniffers assigned to the same channel as *conflict sniffer set*.

In response, we propose a heuristic 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 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 guarantees to provide a valid channel assignment for all the sniffers within linear time, which turns out to be a near-optimal solution for the sniffer channel assignment problem. Another alternative strategy is to exhaustively fix every sniffer in the conflict sniffer set to the conflicted channel, and search for the best assignment among all the valid assignment possibilities. However, such strategy suffers from exponential time complexity in the worst case, thus is unfavorable.

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 central server 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 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:

$$\text{maximize} \quad \sum_{i=1}^{N} U_i(k) \cdot SCAP_i(k+1) \cdot y_i(k+1) \quad (12)$$

$$\text{subject to} \quad (8) - (11).$$

The above optimization problem can be solved using exactly the same approximation algorithm illustrated in the previous section. 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. Complexity Analysis

We analyze the complexity of the above approximation algorithms. Two most complex steps involved in the problem solving are: (1) solving LP relaxation, and (2) executing LP rounding 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 $O((N + MN)^3/log(N + MN))$ [9], which is determined by the complexity of LP solver. On the other hand, the PRA algorithm has the linear complexity, governed by the input vector size (MN) [18]. Our heuristic strategy to solve channel conflict problem has to call PRA algorithm at most N times in the worst case scenario. Thus, the overall LP rounding algorithm takes the complexity $O(MN^2)$. Hence, solving the LP relaxation dominates the overall complexity, resulting in an overall complexity of $O((N+MN)^3/log(N+MN))$ for solving the optimization problems.

VI. Evaluation

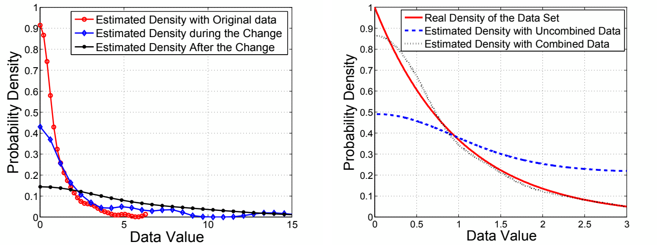
In this section, we conduct extensive simulations and experiments to evaluate the performance of our passive monitoring framework in CRNs. The simulations leverage synthetic traces, which allow us to vary the number of channels and sniffers, as well as the traffic patterns of different users. We also carry out experiments and test the performance of our monitoring framework on real traces collected from the experiments. Aside from our proposed passive monitoring framework, we also implemented two baseline algorithms and one previously proposed algorithm, listed as follows, for performance comparison purposes.

- **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 [12].

We assume the PUs' presence can be detected promptly by both inspection and operation sniffers, as illustrated in section IV-A. In the following sections, we first evaluate the performance of the proposed secondary user channel usage model, and then the frame capturing performance and user capturing performance are examined.

A. Performance of Secondary User Channel Usage Model

The proposed secondary user channel usage model utilizes the online non-parametric density estimation to achieve an accurate estimated distribution of active slot interarrival time. As mentioned in section IV-B, our channel model uses sliding window technique to track the dynamic traffic patterns of SUs. We generate a data set with changing data distribution (pdf of the data set changes from e^{-x} to $\frac{1}{5}e^{-\frac{x}{5}}$). The tracking performance is shown in Fig. 4(a), from which we can see that the estimated distribution is gradually changing from the original data distribution to the current data distribution. The transition happens when the current window takes data from both distributions and completes when the whole current window only contains data from the new distribution. The tracking speed is determined by the window size, i.e., larger window size will cause more delays during the transition.



(a) The tracking performance when the traffic pattern is changing

(b) Performance improvement with data combination

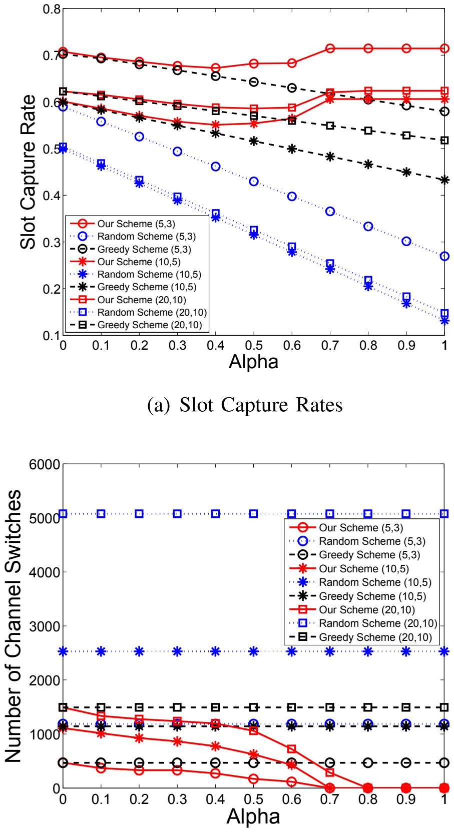
Fig. 4: Performance of Secondary User Channel Usage Model (W=50,n_w=10,t_w=25)

However, too small window size will affect the modeling accuracy, which motivates the combination of historical samples. Recall that only the data sets passing the KS tests can be combined to improve the estimation accuracy. Fig. 4(b) shows the performance comparison between the density estimates with data combining and that without data combining. We can see the significant improvement brought by the accumulative data combination.

B. Performance of Frame Capturing

In this section, the frame capturing performances of different channel assignment algorithms are evaluated. First, we generate synthetic traces of different distributions with either i.i.d. samples or correlated samples, to evaluate the *slot capture rate*, *frame capture rate* and number of channel switches under different settings. *Slot capture rate* is defined as the ratio of the number of captured active slots versus the overall number of active slots up to the current time. *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. Then, we collect real-world traces from operative WLANs to simulate the scenarios in WhiteFi networks, and evaluate our monitoring framework 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 multiple rounds.

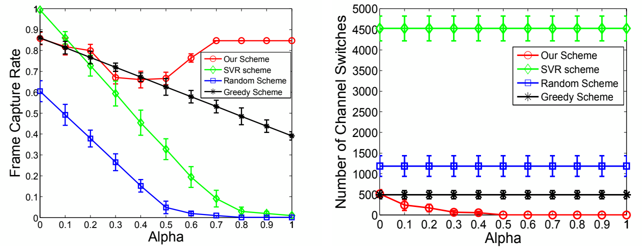
1) Synthetic Traces: First, we generate synthetic traces with i.i.d. frame interarrival time exponentially distributed. Each trace corresponds to the traffic generated in one channel with different mean values to simulate different traffic loads. We evaluate the capturing performance w.r.t. different switching



(a) Slot Capture Rates

(b) Number of Channel Switches

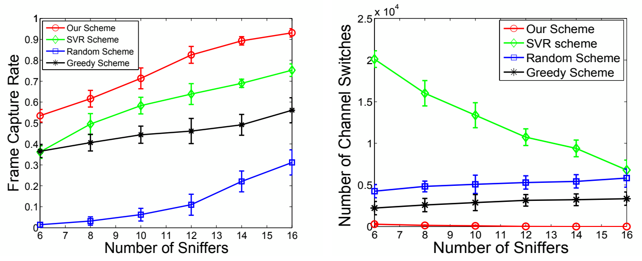
Fig. 5: Performance with Different Methods using Exponentially Distributed Frame Interarrival Time ((N, M) represents the case with N channels and M sniffers)



(a) Frame Capture Rates

(b) Number of Channel Switches

Fig. 6: Performance with Different Methods using Time Series Data (M=3,N=5)



(a) Frame Capture Rates

(b) Number of Channel Switches

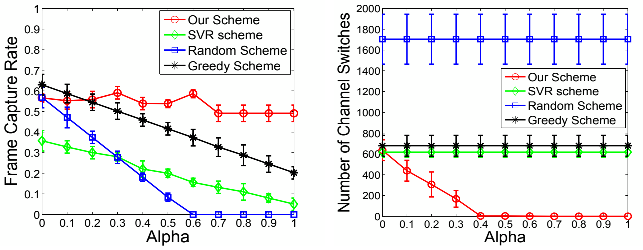
Fig. 7: Performance with Varied Number of Sniffers with Different Methods (α=0.25, N=20)

cost weights α. Generally speaking, one channel switching causes a penalty of losing α slot, $\alpha \in [0, 1]$. Fig. 5(a) shows the slot capturing performance with different number of sniffers and channels. Our channel assignment algorithm always outperforms the other two algorithms. With the increase of α, the slot capture rates of both baseline schemes drop down because of the increasing penalty for channel switching. However, our scheme with high α 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. Therefore, the slot capture rate of our scheme stays high. We also compare the number of channel switches in Fig. 5(b). With our scheme, the number of channel switches drop significantly with the increasing of α, while channel switch numbers of other schemes are irrelevant to α, thus keep constant. From the above two figures, we observe that our scheme is able to achieve superior performance with low channel switching costs. Note that the frame interarrival data points are i.i.d. samples from exponential distribution. These data points are time independent, causing SVR scheme to perform poorly, so we do not show its performance here.

Second, we use time series data to compare the performance with SVR scheme, since SVR scheme is supposed to have superior performance when modeling time series data. We generate time series data to represent frame interarrival time, using Gaussian distribution with an exponential correlation function. Different mean values of data distributions are used to represent different traffic loads. We assume the training process of SVR scheme has already been done, which takes 35 training samples [12]. Fig. 6(a) shows the frame capture rates of different methods. The SVR scheme performs best when the channel switching costs are neglected ($\alpha = 0 or 0.1$), which means the SVR has accurate estimation of frame interarrival time, when the traffic statistics are stable. However, when α grows larger than 0.2, the frame capture rate of SVR scheme drops steadily. Note that the capturing performance of our scheme also drops a bit due to increasing switching penalty, and then reverts back to surpass the performance curves of any other schemes. Fig. 6(b) shows the number of channel switches, from which we can see SVR methods frequently switch its channels, because its switching strategy is frame-wise rather than slot-wise. Still, the switching cost of our scheme remains the lowest.

Finally, Fig. 7(a) shows the different capturing capabilities w.r.t. the number of sniffers, the frame capturing performance of all the methods keeps growing with the increasing number of sniffers. Our scheme has the highest frame capture rate. We also compare the channel switching numbers in Fig. 7(b). With more sniffers, the number of channel switches with SVR method decreases, due to the increased traffic capturing capability, while the other methods have less and stabler number of channel switches.

2) Real Traces: We collect the real traces from $802.11g$ WLAN network. The traces are 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). We assign each trace for one channel, with five channels in total. We evaluate



(a) Frame Capture Rates (b) Number of Channel Switches

Fig. 8: Performance using Real-world Traffic (with 7 channels)

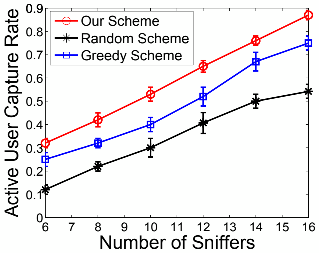


Fig. 9: User Capture Rates

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 and BT, and the other one is generated from two clients using Skype 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, induced by the highly dynamic real-world traffic. Frequent retraining and channel switching operations significantly deteriorate the capturing capability of SVR method. However, our scheme retains the best performance, except in the case of small α, the greedy method performs better since 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.

C. Performance of User Capturing

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 certain ranges; also the frame interarrival time is exponentially distributed, specifying the traffic pattern. 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. 9, from which we notice that our channel assignment scheme can select best sets of channels to maximize the number of active users captured for each slot. The result implies our channel assignment scheme significantly outperforms two baseline schemes, in terms of user capturing performance. We plan to examine user capturing performance using real-world traffic in our future work.

VII. Conclusion

In this paper, we have introduced a systematic passive monitoring framework 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 propose 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 design near-optimal monitoring algorithms. Our simulation and experimental results both show that our passive monitoring framework has superior capturing capability with low channel switching overhead.

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] T. C. Clancy and N. Goergen, “Security in cognitive radio networks: Threats and mitigation,” in *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on*, May 2008, pp. 1–8.

[4] Q. Yan, M. Li, T. Jiang, W. Lou, and T. Hou, “Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks,” in *INFOCOM 2012, IEEE*, March 2012, pp. 900–908.

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

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

[7] 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.

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

[9] D.-H. Shin and S. Bagchi, “Optimal monitoring in multi-channel multi-radio wireless mesh networks,” in *MobiHoc’09*, May 2010.

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

[11] P. Arora, C. Szepesvari, and R. Zheng, “Sequential learning for optimal monitoring of multi-channel wireless networks,” in *INFOCOM 2011, IEEE*, 2011.

[12] 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.

[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] H. Kim and K. G. Shin, “Fast discovery of spectrum opportunities in cognitive radio networks,” in *DySPAN 2008*, Oct. 2008, pp. 1–12.

[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.