AirTrack: Locating Non-WiFi Interferers using Commodity WiFi Hardware

Ashish Patro[*], Shravan Rayanchu[*], Suman Banerjee
{patro, shravan, suman}@cs.wisc.edu
University of Wisconsin Madison

Introduction

Recent studies [1, 4, 5] have shown that interference due to non-WiFi RF devices has become a major problem in today's 802.11 networks. In our own experiments, we observed that high powered interferers like analog cordless phones, video cameras can cause an 802.11 link to experience complete loss of connectivity. Figure 1 shows an example of non-WiFi RF activity in a university cafe, where a FHSS cordless phone and a Microwave oven caused increased interference. Knowledge about the active non-WiFi RF devices and their physical locations can help the WLAN administrators take corrective actions (e.g., bringing down rogue non-WiFi transmitters, or by altering some of the operational wireless parameters). In this paper, we focus on the problem of locating the non-WiFi interferers in a WLAN without using any specialized hardware. Specifically, we try to answer the following question:

"How can a system using only WiFi nodes accurately locate non-WiFi RF devices? Further, how can it do this in real-time, in a non-intrusive manner, and without the help of any additional sensors or hardware?"

In our attempt to answer the above question, we present AirTrack — a system to locate non-WiFi devices using commodity WiFi hardware. AirTrack is non-intrusive, as it employs a *passive approach* (*i.e.*, it does not introduce any additional traffic into the wireless medium) and localizes non-WiFi devices in *real-time*.

Goals and challenges. Locating non-WiFi transmitters using only WiFi hardware is particularly challenging because WiFi nodes cannot decode the transmissions from these devices (e.g., Microwaves, video cameras, Xbox). While it is possible to equip each WiFi node with additional hardware (e.g., a Bluetooth interface to detect Bluetooth devices, a ZigBee interface to detect ZigBee devices), such a solution is clearly not scalable. In some cases, doing so might not even help detect the interferer because the received interference power might be simply due to unintended radiations from the device (e.g., Microwave ovens). With AirTrack, we first wish to *detect* and *uniquely identify* the presence of multiple, simultaneously operating non-WiFi devices using limited signal information (e.g., RSS per sub-carrier) provided by commodity WiFi cards. For example, if two Bluetooth devices and an analog phone are operating simultaneously, AirTrack should be able to *detect all the three* device instances. We note that detecting the presence

[*]Student authors are listed in alphabetical order.

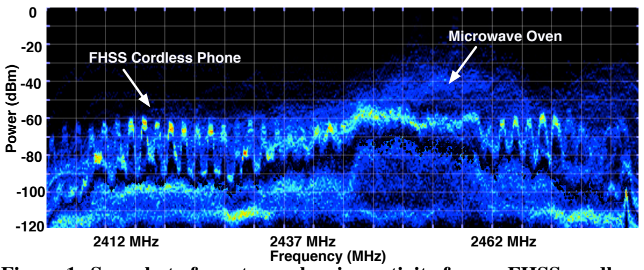

Figure 1: Snapshot of spectrum showing activity from a FHSS cordless phone and a Microwave oven. Measurements were taken at a university cafe using AirMaestro RF signal analyzer[1].

of multiple non-WiFi device instances of the same type is much harder as it requires *segregating* the transmissions belonging to each active device instance. Further, we wish to *physically locate* each non-WiFi device instance. An additional challenge in localization stems from the fact that the transmit power of the non-WiFi device is unknown.

Background. Emerging WiFi cards are capable of providing *spectral samples* — information about the signal power received in each of the sub-carriers of an 802.11 channel. Using these spectral samples, authors in [5] developed Airshark, a system that detects the presence of non-WiFi devices. Airshark extracts novel features using only received power samples, and employs decision tree classifiers to detect different device types. Post detection, it outputs a set of "pulses" (time-frequency blocks representing device transmissions), and tags these pulses with the appropriate device type (e.g., Bluetooth or a ZigBee pulse). Each pulse reported by an Airshark node consists of local timestamps indicating the start and end times of the pulse transmission derived from the "mactimestamp" of the WiFi radio employed by node, the center frequency and bandwidth of the pulse observed at this node, the received power of the pulse and finally a tag, indicating the non-WiFi device type.

While Airshark can detect the presence of non-WiFi device transmissions, *it cannot differentiate amongst the multiple non-WiFi device instances of the same type that are active simultaneously and it cannot determine their physical locations*. For example, if two FHSS phones are active at the same time, Airshark will report that it detected an FHSS phone, but it cannot uniquely identify the actual number of phones or their physical locations. This is because Airshark cannot segregate the pulses belonging to each device instance. Traditional WiFi localization systems do not have to deal with this issue, as the source address field can be used to segregate frames belonging to a particular WiFi transmitter.

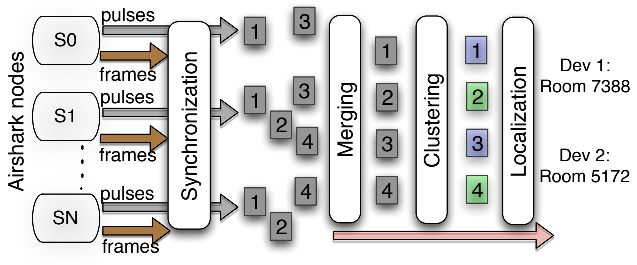

Figure 2: Overview of the localization process used in AirTrack.

Our proposed approach — AirTrack

AirTrack addresses the above issues by using multiple WiFi APs (equipped with Airshark module), and provides mechanisms to facilitate: (i) synchronizing timestamps of multiple APs, (ii) establishing a unique ID (or a mock-MAC address) for each active non-WiFi device instance and segregating pulse transmissions belonging to each device (iii) localizing each non-WiFi device instance (of an unknown transmit power). We first present the basic idea and intuition behind our approach. We then explain each of the components in detail.

Basic idea. We assume an enterprise WLAN set up where all the APs are connected to a central controller. Tagged pulses from different APs can be collected at the controller for further analysis. Consider an example where two Bluetooth devices d_1 and d_2 are switched on in the vicinity of 4 APs. Assume that APs s_1, s_2 and s_3 can hear transmissions from device d_1, whereas s_2, s_3 and s_4 can hear transmissions from device d_2. In this case, all 4 APs report pulses tagged with type 'Bluetooth'. How can we separate the pulses that belong to device d_1 from those that belong to d_2? If the timestamps of all APs are synchronized, then we can examine the pulses that are received by a subset of the APs at the same time instance. In this example, if we find three pulses received by the set of APs s_1, s_2 and s_3 that have the same device type, frequency and bandwidth and the same start and end times, then it is most likely that the three APs received the same pulse. This enables us to identify the distinct pulses, and the set of APs that received these pulses along with their received signal strength (RSS). The next task is to then *cluster* these distinct pulses based on some criteria in order to segregate them into different device instances. AirTrack uses criteria like timing properties exhibited by various non-WiFi devices, and the RSS from the set of APs that hear the pulse to perform clustering. Each cluster then represents transmissions from a single non-WiFi device instance, and the cluster center represents a mock-MAC — a unique ID for this non-WiFi device. Finally, the signal strengths of the pulses in each cluster can be used to localize the device. Figure 2 presents an overview of the whole process. We now explain each of the above components.

Opportunistic synchronization. Our first task is to synchronize the timestamps at multiple APs. A typical approach to synchronization is to find common reference points and then adjusting the clocks to establish a global timeline [2]. One can use "common" pulses among different APs in order to achieve this. However, this poses a chicken-and-egg problem: we had set out to use synchronized timestamps at APs to find the common pulses. AirTrack solves this issues by leveraging the WiFi hardware — the timestamps of the pulses are derived from the same clock which is used to timestamp the received 802.11 frames. Therefore, we first synchronize all the APs using "common" frames received as reference points, and then using the synchronized APs to find "common" pulses. Since it is highly unlikely that we find one reference frame that is received by all APs, AirTrack *opportunistically synchronizes* "pairs of APs" using common frames, and then transitively synchronize across all APs. This procedure is repeated every *sync interval*. We currently use a sync interval of 100 ms which results in tight synchronization between the APs (synchronization error of less than $2-4$ μs).

Merging pulse traces to identify distinct pulses. After the pulses captured by multiple APs are synchronized in time, AirTrack attempts to 'merge' the common pulses captured by the different APs using a *merging module*. When multiple APs record a particular pulse emitted by a non-WiFi device, AirTrack combines the pulse information (e.g., RSS) from these APs to create a single pulse instance or a "merged pulse".

The merging process is handled separately for devices that operate by emitting short pulses (e.g., frequency hoppers or devices like ZigBee) and high duty devices that emit continuously emit energy when they are active (e.g., analog phone). For the former class of devices, we identify if the same pulse (emitted by a particular non-WiFi device instance) was observed at different APs by matching the start/end times of the pulses, center frequency and bandwidth of the pulses along with the pulse type. If a pulse if transmitted by the same non-WiFi device instance (e.g. a particular Bluetooth device), then APs in the reception range of this device hear this pulse at the same time and frequency. For the latter class of devices (e.g. analog phone) which emit energy continuously while operating, we use a more coarse grained approach: we merge instances from those APs that report this device and have the same start time and center frequency. For example, if multiple APs observe instances of an analog phone at the same center frequency, but have different start times for these instances, we don't merge them as they correspond to different devices present in the environment.

Creating a mock-MAC to identify device instances. When multiple devices of the same type operate simultaneously, it becomes necessary to separate their merged pulses in order to carry out localization for each device. For high duty devices, we simply create a mock-MAC as $\{t_s, f_c, (s_i, RSS_i)\}$ where t_s is the coarse-grained start time at which the device instance was first detected, f_c is the center frequency of this device, and the set (s_i, RSS_i) represents all the APs which reported this device at the same instant, along with their mean RSS. However, creating a mock-MAC for pulse transmitters (e.g., Bluetooth) is not as simple. We need clustering mechanisms to segregate the merged pulses belonging to each device. Each resulting cluster represents the set of merged pulses

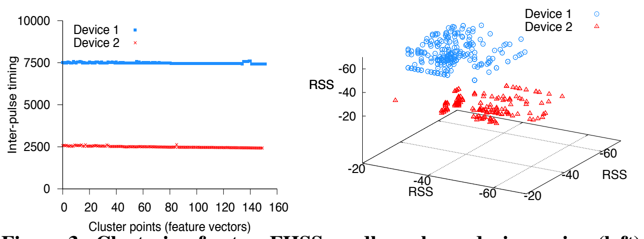

Figure 3: Clustering for two FHSS cordless phone devices using (left) timing property and (right) signal strength (This experiment was done using 3 APs, so each axis represents the RSS observed at each AP).

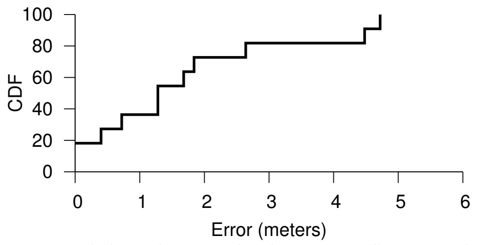

Figure 4: Non-WiFi device localization error (in meters) for our deployment of 5 APs.

obtained from an individual non-WiFi device and we use the cluster center as the unique mock-MAC. We currently cluster the merged pulses along two dimensions: (i) Inter-pulse timing signatures (for devices like Microwave ovens and FHSS Phones) (ii) RSS vector created during the merging stage that contains the received signal strengths at the APs constituting each merged pulse.

Pulses from devices such as FHSS phone and Microwaves exhibit specific inter-pulse timing signatures that can be used to distinguish between multiple device instances. For example, in WDCT cordless phones, the handset and the base transmit alternatively, and the gap between consecutive pulse start times is 5 ms [5]. We found RSS based clustering to be more challenging because of missing RSS values for some of the APs — even if a non-WiFi device is in the vicinity of n APs, many pulses are captured only at a subset of these n APs due to factors such as missing spectral samples at the AP and RSS variations. We experimented with two clustering algorithms that can handle such missing attributes: Expectation-Maximization (EM) and DBSCAN. While the EM algorithm runs slower than DBSCAN, it results in better clustering. This clustering process is invoked once every 10 seconds, so we found the overhead of EM acceptable.

Localizing a non-WiFi device instance. We localize a non-WiFi device as follows: we divide the entire region into grids of size 1×1 meters. We denote the co-ordinate of a grid i as (x_i, y_i). Let d_{ij} denote the distance between grids i and j. We assume that the grid locations of the APs are known apriori. Further, for each AP j, let P_j denote the mean received power from the non-WiFi device.

Now, if the non-WiFi device is at a grid i, then the received power at an AP located at grid j can be computed as $P_{exp}(d_{ij}) = P^o - 10\gamma log_{10} d_{ij}$, where γ is the pathloss exponent and P^o is the power received from the non-WiFi device when placed at a distance of 1 meter from an AP (henceforth referred to as *transmit power*). Localizing the non-WiFi device would require us to find $\phi = (x_i, y_i, \gamma, P^o)$ that minimizes the error between the expected received power $P_{exp}(d_{ij})$ and actual received power P_j for all APs, i.e, we wish to find $\phi = (x_i, y_i, \gamma, P^o)$ that minimizes

$$\mathcal{E}(\phi) = \sum_j |P_j - P^o + 10\gamma log_{10} d_{ij}| \qquad (1)$$

In our current prototype of AirTrack, we perform an exhaustive search to find ϕ that results in minimum error for localization. To reduce the complexity of the algorithm, we collect measurements to estimate the path loss exponent, γ: each AP sniffs the frames (e.g., beacons) from other APs in the vicinity and records the average RSS. Since the co-ordinates of all APs are known, γ can be estimated periodically. We are currently exploring mechanisms similar to [3] to further reduce the complexity of our algorithm.

Results

Implementation and deployment. Our current prototype of AirTrack consists of 5 APs spanning one floor of our department building. Each AP is equipped with Atheros AR9280 AGN wireless card and runs an Airshark module [5]. All the APs are connected to a central controller over the Ethernet. The spectral samples from the card are processed by the Airshark module and it outputs pulses tagged with device type. Tagged pulses, along with captured wireless frames are used by AirTrack to perform synchronization and device localization.

Preliminary results. We experimented with different non-WiFi devices, and benchmarked the clustering algorithms to understand their performance. Figure 3 shows the performance of EM clustering in the presence of two FHSS cordless phone devices that were placed 15 meters apart — both RSS based clustering and timing based clustering correctly output two distinct device instances (clusters) and segregate their pulses. However, we observed that the performance of RSS based clustering degrades when the devices are close to each other (e.g., $<$ 3 meters apart) as it outputs only one cluster. We are currently developing an improved algorithm combining the above two metrics. We also ran our localization algorithm while placing the cordless phones at different distances and measured its performance. Figure 4 shows that the error in localizing non-WiFi devices is less than 6 meters in our current deployment. We are currently experimenting with different non-WiFi device combinations and AP topologies to benchmark AirTrack's performance in diverse scenarios.

1. REFERENCES

[1] Bandspeed AirMaestro. Understanding the Effects of Radio Frequency (RF) Interference on WLAN performance and Security, 2010. http://www.bandspeed.com/.

[2] Cheng Y. et al. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM'06*.

[3] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. MobiCom '10.

[4] S. Gollakota, F. Adib, D. Katabi, and S. Seshan. Clearing the RF Smog: Making 802.11 Robust to Cross-Technology Interference. In *ACM SIGCOMM 2011*.

[5] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: Detecting Non-WiFi RF Devices using Commodity WiFi Hardware. In *ACM IMC 2011*. `http://cs.wisc.edu/~shravan/IMC2011_airshark.pdf`.