Estimating Human Presence through WiFi Packet Sniffing

Koppány Levente Cserna
NetID: kcserna
Student No.: 5301378
Email: K.L.Cserna-1@student.tudelft.nl

Nishad Mandlik
NetID: nmandlik
Student No.: 5242541
Email: N.G.Mandlik@student.tudelft.nl

Report contains 4 of maximum 4 pages.

I. Introduction

Crowd activity monitoring, which is useful in several applications, is a challenging task. In this paper, we attempt to estimate the number of people present at a site, by counting the number of WiFi-enabled devices in the vicinity. The use of handheld devices for accessing internet-based services has increased significantly in the recent years. With features like smart network switch, smartphone users no longer need to manually switch from WiFi to Cellular data when they go outdoors. Thus, most of users leave the WiFi radio enabled, even when not connected to any access point. We exploit the fact that WiFi stations transmit packets whether or not they are associated with an access point. By sniffing these packets, the devices present at the site can be identified.

II. Method

WiFi devices are capable of measuring RSSI values for every received packet. This value is indicative of the power of the received signal and can be used to understand the proximity between the transmitter and receiver. We filter out sniffed packets by setting a threshold on the RSSI value, in order to detect the WiFi devices in the surroundings. Devices are identified using the transmitter MAC addresses from the captured frames. Existing work in this field deals with analysing only probe request frames sent during a WiFi scan. For non-associated stations, this is a reasonable method. However, for associated stations, our preliminary investigation showed that the frequency of probe requests highly depends on the network manager software. Thus, the sniffer may not be able to discover such a device, by capturing only probe requests. We therefore focus on all frame types which are transmitted only by a WiFi station (e.g. Probe Requests, Association Requests, QoS Null Function Messages, etc.). [1] We discard frames like Beacons, Probe Responses and Association Responses, which are obviously emitted by Access Points. Based on this method, we formulate our hypothesis and limitations as follows:

A. Hypothesis

- The number of detected MAC addresses should reflect the human activity at the corresponding time and location.
- Probe requests alone are not sufficient to detect the presence of a WiFi station.

B. Limitations

- Devices with WiFi radio disabled, or those operating in mobile-hotspot mode will not be detected.
- If MAC address randomization is enabled, devices may use different MAC addresses for probe and association requests, causing false detections.
- Stations which are not associated to an access point will be detected only if they perform active scanning (i.e emit probe requests). Although passive scanning is uncommon, a small minority of applications on older operating systems may employ this technique.[4]

III. Test setup

A. Hardware Setup

In order to make the sniffer portable, we chose to implement it on a Raspberry Pi. To be able sniff all WiFi traffic without associating with an access point, the network card should operate in monitor mode. [6] Since the built-in wireless interface on the Pi does not support monitor mode, we used a USB WiFi adapter with monitor capability. The power consumption of the sniffer is measured using a USB meter. The equipment used is shown in Figure 1 and listed below:

- 1x Raspberry Pi 4 Model B (4GB variant).
- 1x Alpha AWUS036NHA WiFi Adapter.
- 1x Baseus X30 30000mAh Power Bank.
- 1x USB current and voltage detector.

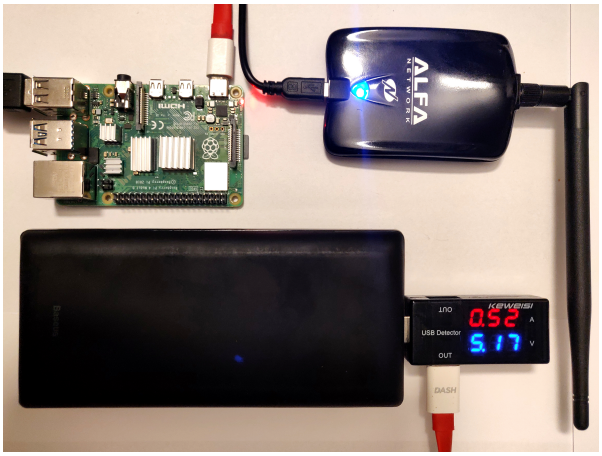


Fig. 1: Hardware Setup

Note: The USB WiFi adapter operates at 2.4GHz, and hence it does not allow to sniff packets in the 5GHz band.

B. Software Setup

The Raspberry Pi runs *Ubuntu 20.10 Desktop*. To minimize overheads during sniffing for long intervals, we use *dumpcap* in favour of *tshark*. From our list of filters, the filters which are supported by *pcap* are provided to *dumpcap* using the capture filters option. The rest of the filters are applied during analysis of the captured packets. We created a *Python* wrapper which identifies the monitor-capable interface on a system and sets it to monitor mode. It also passes the necessary options to the *dumpcap* program. The packet dumps are read using *PyShark* and analyzed using *NumPy* and *pandas*. Along with filters for frame types and RSSI values, an additional filter is added to discard packets originating from any wireless interface of the Raspberry Pi itself. Time synchronization does not need to be performed manually because *Ubuntu 20.10* has a built-in daemon for this purpose.

C. Duration and Locations

We conducted packet sniffing for a total of 5 different scenes, at 4 locations, which are mentioned below:

- Scene 1: Delft Station Bicycle Parking 1 (Weekday)
- Scene 2: Delft Station Bicycle Parking 1 (Weekend)
- Scene 3: Residential Building Bicycle Parking
- Scene 4: Residential Building Corridor
- Scene 5: Residential Building Mailbox Room

For each scene the data was collected for 24 hours at a stretch. (*For Scene 4, data was lost for an interval of* 30 *minutes*)

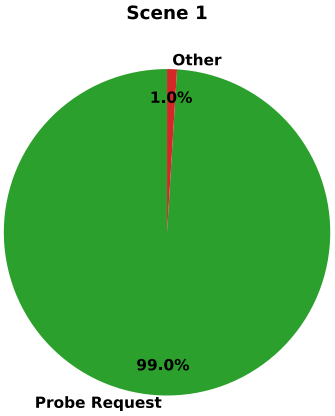
IV. Results and Evaluation

A. Temporal analysis of WiFi activity

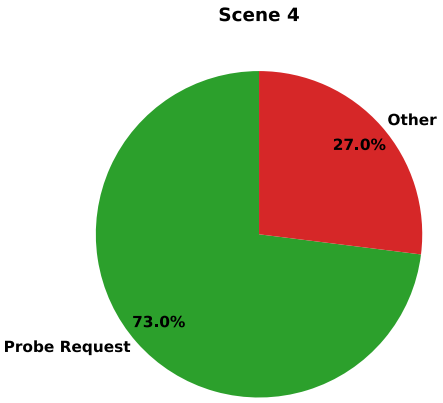
Figure 3 shows the number of WiFi devices at various locations in a period of 24 hours. The data collected at Delft Station (plots a and b) show that this number drops to zero at night, when no train services are plying. During weekdays (plot a), greater number of devices are detected during the peak commute times in the morning and evening. On weekends (plot b), a spike is observed before the start of night-time curfew (2100 hrs). A similar spike is also present in plot c (bicycle parking in a residential building). Plot d (residential building corridor) shows as compared to day-time, greater number of devices during the night, when most people are back in their apartments. It can also be observed that public places like train stations witness much WiFi activity compared to specific locations within a residential building. The mailbox room is not visited frequently, and the data gathered over there shows the least number of detected devices (plot c). Thus, according to our expectations the trends in the number of detected WiFi stations resemble crowd activity patterns at various locations.

B. Means of device detection

The means of device detection can be seen in Figure 2. The green sector shows the percentage of devices for which at least one probe request was captured. The red sector shows the percentage of devices which are detected only due to frame types other than probe requests. It can be observed that for Scene 1 (*Delft Station Bicycle Parking 1 (Weekday)*), our initial hypothesis fails. A negligible number of devices are detected without any probe request. The outcomes for Scenes 2, 3 and 5 are similar to those of Scene 1, and hence, are excluded. Scene 4 (*Residential Building Corridor*), however, supports our hypothesis. Here, probe requests were not captured for a significant number of detected devices.



(a) Delft Station Bicycle Parking 1 (Weekday)



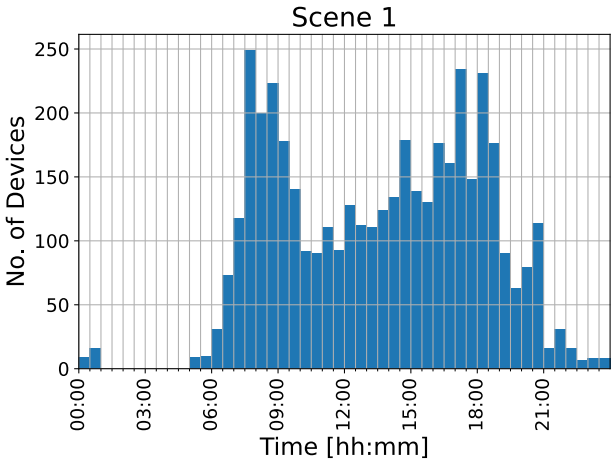
(b) Residential Building Corridor

Fig. 2: Statistics of the means of device detection

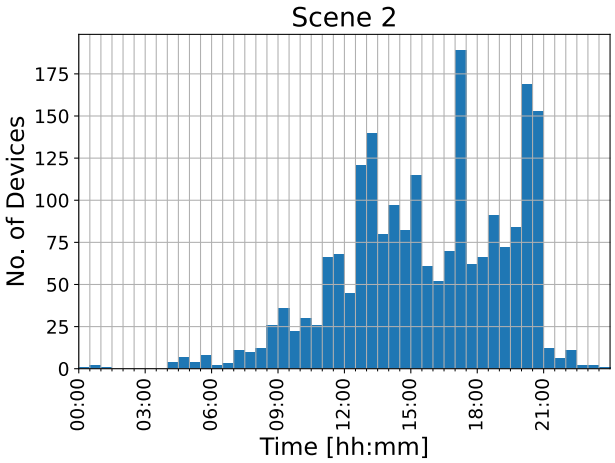
This behaviour can attributed to the association status of devices. In Scene 4, it is likely that several devices were connected to the home networks of users, and hence, transmitted lesser number of probe requests. Other scenes included areas having no open WiFi networks. In such a setting, devices perform scanning at more frequent intervals, thus increasing the odds of probe request based detection.

C. MAC address randomization

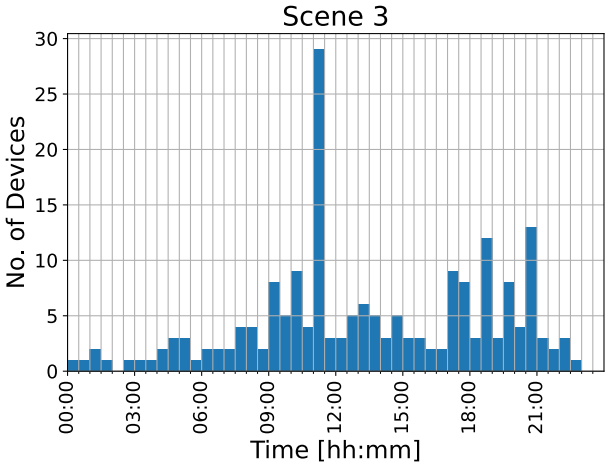
For randomized MAC addresses, the second most significant nibble is set to 2, 6, A or E.[5] Figure 4 (top) shows the statistics of MAC addresses randomization



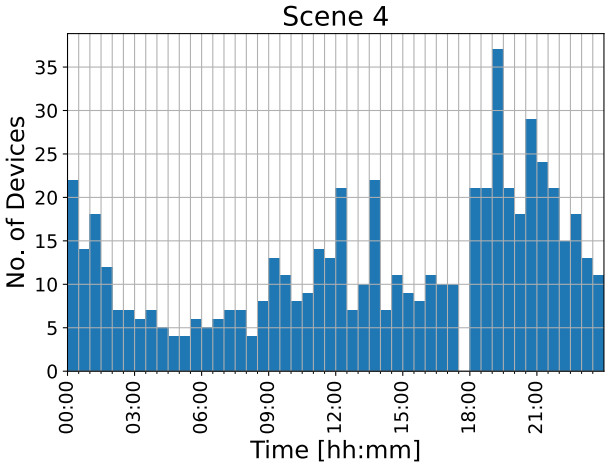
(a) Delft Station Bicycle Parking 1 (Weekday)



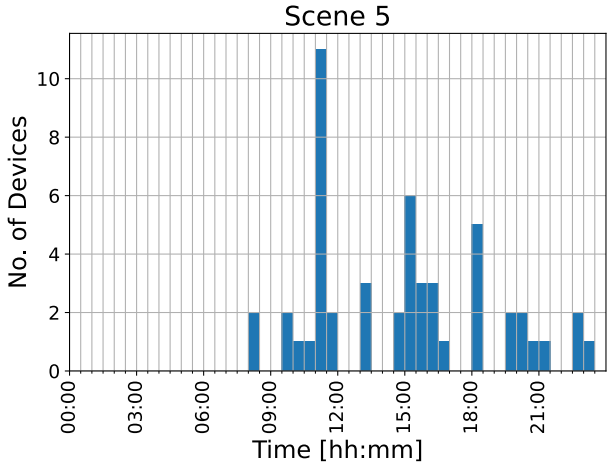
(b) Delft Station Bicycle Parking 1 (Weekend)



(c) Residential Building Bicycle Parking



(d) Residential Building Corridor



(e) Residential Building Mailbox Room

Fig. 3: Number of detected devices in a period of 24 hours.
(*For Scene 4, the data for the interval 17:30 to 18:00 was lost.)*

across devices detected from all the scenes. It can be seen that a vast majority of devices use randomized MAC addresses. Vendor lookup succeeded for only a small fraction of the MAC addresses, the results of which are shown in Figure 4(bottom). The *Python* package *mac-vendor-lookup* was used for this purpose. It should be noted that these percentages do not necessarily portray the market share of vendors. A greater number of devices of

a particular vendor, in the captured data, may also be due to address randomization being disabled on those devices.

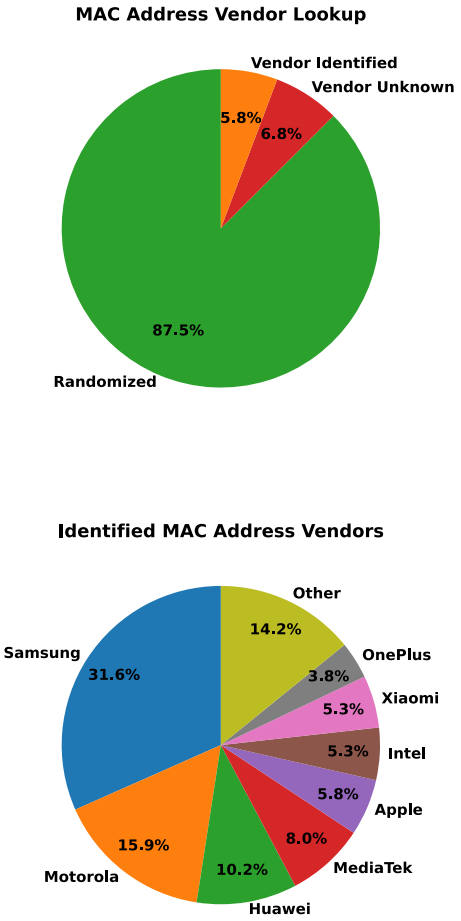


Fig. 4: Lookup results for MAC addresses from all scenes.

D. Power Consumption

When the USB power meter was attached in series between the WiFi adapter and the USB port of the Raspberry Pi, it did not display any readings for current consumption. However, it was able to measure the consumption for the entire system, when attached between the power source and the Raspberry Pi. With the WiFi adapter detached, the Pi was found to consume a current of $0.38A$ at a voltage of $5.18V$. When the adapter was attached the current consumption increased to $0.52A$. The least count of the meter used is $0.01A$, and probably differences due to mode setting (managed/monitor) or turning sniffing on/off were too small to be detected by the measuring device.

V. Conclusion and Future Work

It is evident that number of WiFi devices detected at a location is indicative of the crowd strength. However, computing the exact number of people is a difficult task due to the limitations mentioned earlier. False detections due to randomized MAC addresses are a major challenge to overcome. Since newer operating systems use randomized MAC addresses for probes as well as associations, the access point spoofing approach mentioned in Brower and Hokke [2] is no longer accurate. The packet-linking technique proposed in Freudiger [3] may work and needs to be investigated. But it may still not be possible to identify the original MAC address of a device.

Another interesting observation is that, at sites where majority of devices are expected to be connected to a WiFi access point, probe requests alone are not sufficient to measure the number of devices. Given the large amount of disk-space and processing time required for the packets, it is preferable to reduce the number of captured packets as much as possible. Thus in public places where no open WiFi networks are available, counting devices on the basis of only probe-requests, is a reasonable technique.

References

[1] *802.11 frames : A starter guide to learn wireless sniffer traces*. en. Oct. 2010. URL: https://community.cisco.com/t5/wireless-mobility-documents/802-11-frames-a-starter-guide-to-learn-wireless-sniffer-traces/ta-p/3110019 (visited on 03/02/2021).

[2] Jetse Brower and Niels Hokke. *MAC-based activity tracking using passive sniffing*. en. Tech. rep. Delft University of Technology. URL: https://github.com/NielsHokke/MAC_Tracker (visited on 03/02/2021).

[3] Julien Freudiger. "How talkative is your mobile device? an experimental study of Wi-Fi probe requests". In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '15. New York, New York: Association for Computing Machinery, June 2015, pp. 1–6. ISBN: 9781450336239. DOI: 10.1145/2766498.2766517. URL: https://doi.org/10.1145/2766498.2766517 (visited on 03/02/2021).

[4] Mike Lockwood. *a5ec95c - platform/frameworks/base - Git at Google*. URL: https://android.googlesource.com/platform/frameworks/base/+/a5ec95cdb1a7d2024249277dff1f99d0046c9b56 (visited on 03/02/2021).

[5] Wes Purvis and Slava Dementyev. *Get to know MAC Address Randomization in 2020*. en-US. Sept. 2020. URL: https://www.mist.com/get-to-know-mac-address-randomization-in-2020/ (visited on 03/02/2021).

[6] Younis Said. *Using Monitor Mode in Kali Linux 2020 – Linux Hint*. en-US. URL: https://linuxhint.com/monitor_mode_kali_linux_2020/ (visited on 03/02/2021).