Draft:
Title: Defeating MAC Randomization to use Wi-Fi Sensors to Provide Location of Mobile Devices
By: Amarpreet Singh, William Bateman, Willis Chien, Simona Marinova

I. Introduction

Observing the movement of crowds and people has applications in many fields. The movement and density of a crowd can be used in the managing of events, such as street festivals or amusement parks. Similarly, observing traffic movement on roads and highways can provide useful data to improve traffic flow.

Many current solutions are based on live camera feeds. High-quality cameras that record and store long video footages can be expensive. Computer vision algorithms that use these videos to extract useful information can be computationally difficult. Moreover, the aforementioned techniques are limited by the line-of-sight of the camera.

We propose a crowd observer based on listening to Wi-Fi probe requests and using the MAC address unique to each device to estimate the number of people in an area. This solution is well suited for an Internet of Things (IoT) approach, as WiFi sensors are cheap, especially when compared to cameras, and do not require complex computer vision algorithms. With multiple sensors spread over an area, the people count and unique MACs can be used to analyze the flow of people.

In this paper, we cover background information, which include the mechanisms behind probe requests and MAC randomization, the controlled testing that we conducted to learn more about the behaviour of MAC randomization, and finally live testing for the purposes of gathering data to visualize results.

II. Background

Probe Requests

Periodically, devices will send out probe requests to attempt to connect to saved Wi-Fi network. These devices include laptops and tablets, but for this paper, we focus on smartphones. The assumption is that this solution will be used in a setting where tablets and laptops will be off, such as an outdoor festival, while smartphones will generally be kept powered on. Another key assumption made is that the number of devices tracked correlates to the number of people present in a given area. Some people may not have a device with them, or no device that is actively sending probe requests, while others may have many devices. For the purposes of counting the size of a crowd, we assume the device to person ratio is roughly one to one.

The probe request contains the MAC address of the source device, which can be used as a unique identifier. Each MAC is six bytes, where the first three bytes are the Organization Unique Identifier (OUI) and the last three bytes identify the device itself. The probe also contains additional information about the specific capabilities of the device, including the WLAN channels it can support. By passing this MAC through a hashing function, we can create anonymity for each device in our dataset while preserving the uniqueness of each address.

MAC Randomization

Some devices can use a fake MAC address under certain circumstances. When these devices are not connected to a Wi-Fi network, and are sending probe requests, they use a randomized MAC address. This randomization reduces the ability of third-parties to track individual devices and the activity of individual devices, and increases privacy.

MAC randomization was introduced to iPhones with iOS 8 [1], but only seems to work in the iPhone 5s and newer devices [2]. Android 6.0 uses MAC randomization for background Wi-Fi and Bluetooth scans [3], but only works if the driver and hardware can support randomization [4]. Windows 10 and Linux kernel version 3.18 also have their own version of MAC randomization [4].

However, as discussed in [4], there are ways to circumvent this MAC randomization. The MAC address itself is not required to uniquely identify devices, as this can be achieved using the information elements and sequence numbers sent in the probe request to create a fingerprint for each device.

III. Controlled Testing

In order to observe the MAC randomization problem firsthand, we collected data using a Wi-Fi sensor. An Onion Omega development board acted as the sensor, and we used the tcpdump command-line tool to capture and analyze the packets in the Wi-Fi network. To block out background traffic in the Wi-Fi network, we collect our data in an insulated room, used for radio testing, which acts as a Faraday cage, so neither device was connected to a Wi-Fi network for the duration of the test. Our tests were performed on an iPhone 6S running iOS 8.4.1 and a Moto X 2nd version (2014) running Android 6.0. Based on the tests, we observe that the iPhone used randomizes its MAC, whereas the Android phones used do not randomize MACs. Moreover, the iPhone generates new randomized MACs based on the way it is being used by the user.

Figure 1 shows the iPhone's MAC and probe requests while it is in sleep mode and has no background programs running. Figure 2 shows the iPhone with background programs running, and at the indicated times the phone is unlocked and locked. We notice that the iPhone's randomized MAC changes and probe requests occur simultaneously when the phone is being unlocked. Comparatively in the idle behaviour, both the MAC randomized and probe requests of the iPhone are more frequent. Figure 3 shows the iPhone as it is being actively used. In this case, while probe requests are much more frequent, the randomized MAC does not change. The Android phones we tested did not display any MAC randomization.

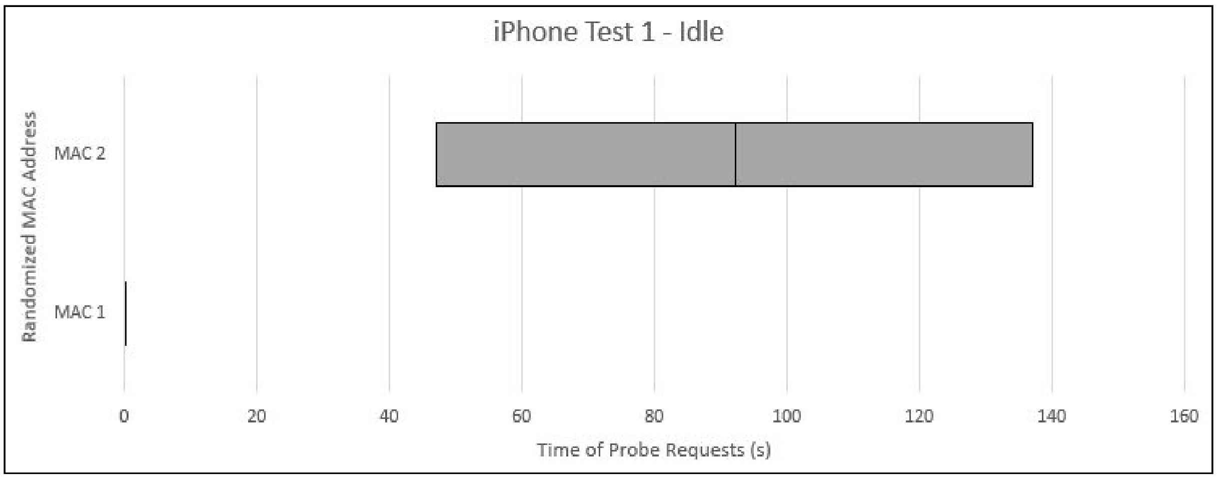


Figure 1. Probe requests and MAC addresses over time for an idle iPhone.

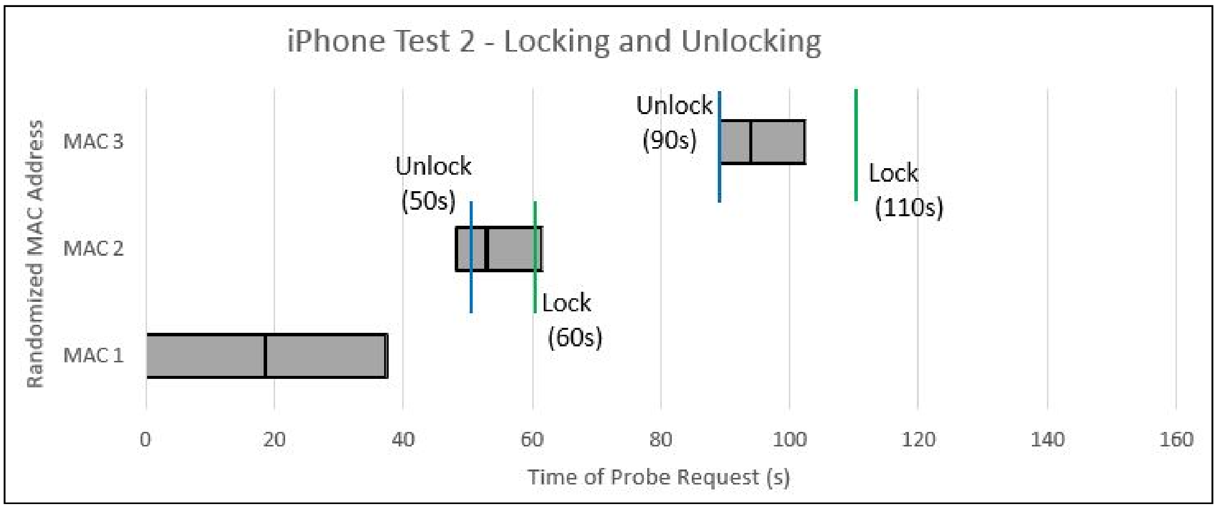


Figure 2. Probe requests and MAC addresses over time for an iPhone periodically being locked and unlocked.

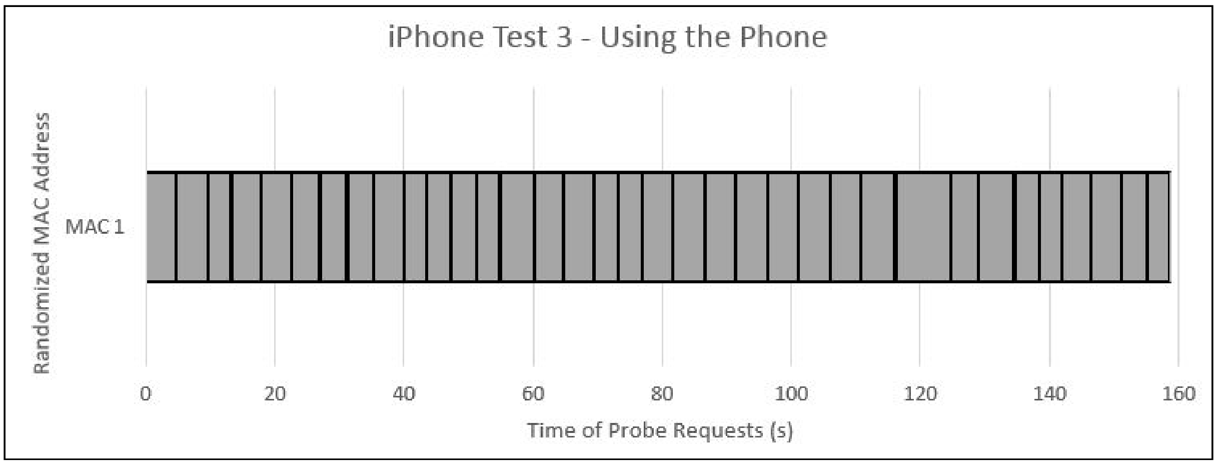


Figure 3. Probe requests and MAC addresses over time for an iPhone in active use.

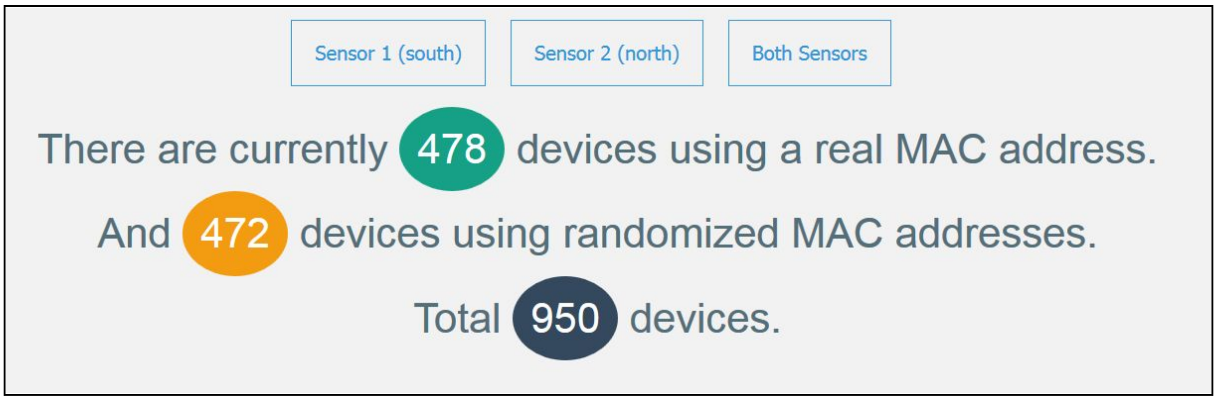
Using the behaviour of the iPhone during the tests, we create a method to distinguish randomized MACs from the real MACs. The randomized MACs are completely random, meaning that the first three bytes, which is the OUI, are also randomized. Therefore, any MAC address that does not correspond to a valid OUI is a randomized MAC. If a MAC is not truly randomized, it is possible for the MAC to posses a valid OUI. However, the probability of this occurring is insignificant enough that we can ignore it completely.

Then, we apply techniques from [4] to create a fingerprint for each device, based on the additional information sent in each probe request. Specifically, we used the data of the information elements in the frame body of the probe request. This method is not perfect, but "can correctly track as much as 50% of devices for at least 20 minutes." [4, p1]

IV. Real-World Testing

We proceeded to collect data from outside of the Faraday cage. Using a similar process, we used tcpdump to record the Wi-Fi traffic, and processed the packet data after collection. Using the data, we only analyzed probe requests, and disregarded all other packets in the Wi-Fi network. We used two sensors in predefined locations for approximately three hours. Using this data, we extracted the MAC addresses and fingerprinting information for each probe request, and stored the results in a database. We created a web portal using Google Charts to visualize the data. The portal shows the number of real and randomized MACs, the number of devices per vendor, the density of people on a map using the signal strength, and additional information about the the rate of probe requests and lifetime of MAC addresses.

Figure 4 shows the initial view of the portal, which allows the user to select which sensor's data to show. Figure 4 also shows the device count for both sensors. As shown, roughly half of all devices used a randomized MAC in probe requests.



There are currently 478 devices using a real MAC address.

And 472 devices using randomized MAC addresses.

Total 950 devices.

Figure 4. Device count from both sensors over the full testing period.

Figure 5 shows the number of devices detected by each sensor over the testing period of just under two hours. Figure 6 shows the movement of devices between the two sensors, using the MAC address, or the device fingerprint in the case of MAC randomization to track the devices. Note that figure 6 does not take into account devices that simply leave the area, or new devices, but simply notes whenever a device seen by one sensor had been recently seen by the other sensor.

As shown, there were initially more devices near the northern sensor (sensor 2) than near the southern sensor (sensor 1). At around t=30 minutes, the number of devices detected by both sensors rises, peaks at t=45 minutes, then steadily declines, leaving an approximately equal number of devices near each sensor. This correlates with what we see in Figure 6, as from t=30 minutes to t=50 minutes there is a large number of devices moving from sensor 2 to sensor 1. This explains the decrease in devices at sensor 2, and the increase in devices seen by either both sensors or just sensor 1. After t=60 minutes, the movement of devices becomes more equal, which corresponds to the more constant lines in figure 5.

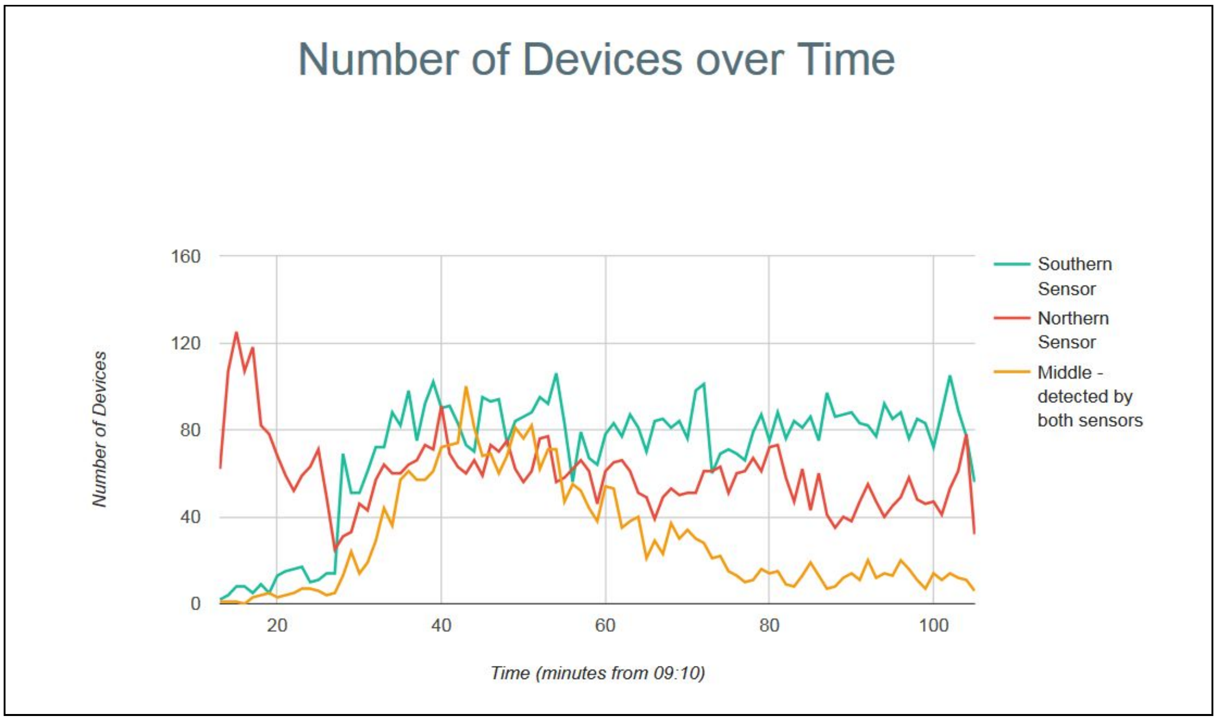


Figure 5. Number of devices over a span of just under two hours. The orange line denotes devices that were detected by the both sensors within a frame of 15 minutes.

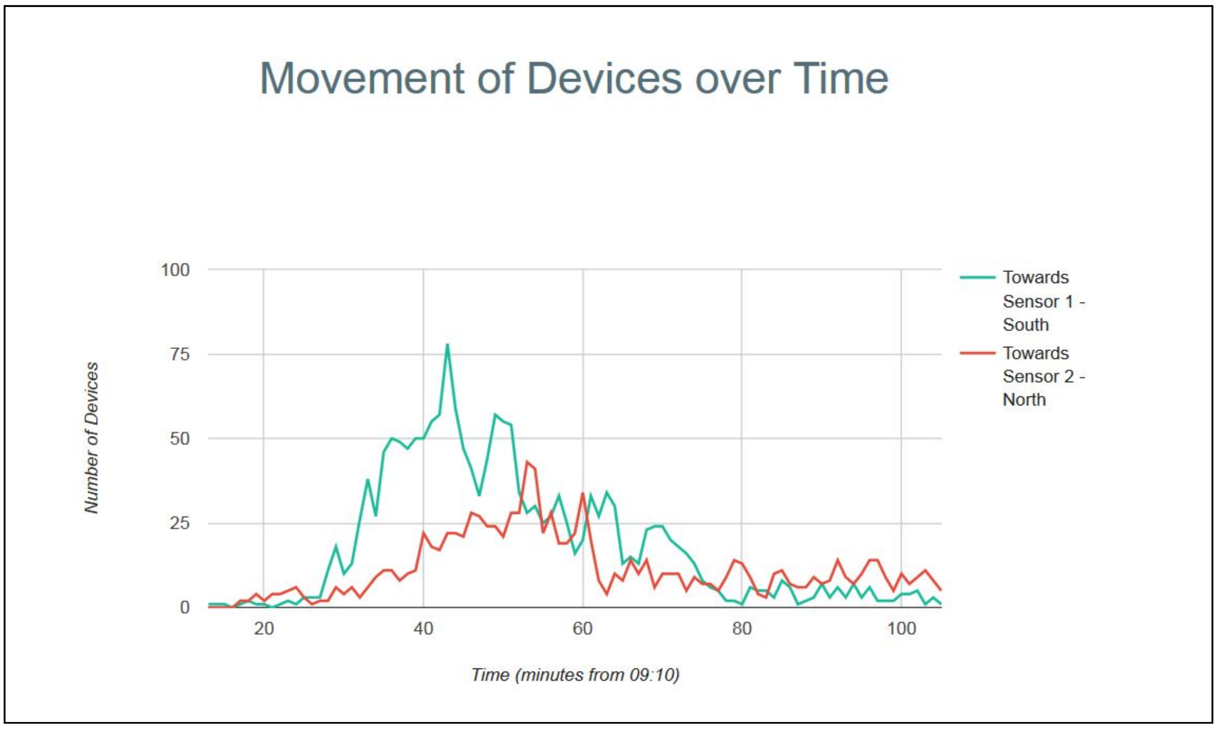


Figure 6. Movement of devices over a span of just under two hours. The red line shows devices that were seen at sensor 1, then moved north to sensor 2. The green line shows devices that were seen at sensor 2, then moved south to sensor 1.

Figure 7 shows the lifetime of randomized MAC addresses in the data. Lifetime is considered to be the length of time from when we first recorded the MAC to when we detect a change in the MAC address for the device. When compared to the data from controlled testing, most of the lifetimes shown are significantly longer than the lifetimes in the idle case or the locking/unlocking examples. Additionally, only ten devices were detected to have changed randomized MAC. Even assuming we have an accuracy of 50% as in [A], 20 devices is only a small fraction of the total 472 devices using non-randomized MAC addresses. This would suggest that the majority of devices are either in use or potentially have background programs running, which can affect the MAC randomization [1], or simply do not have the capability to perform MAC randomization.

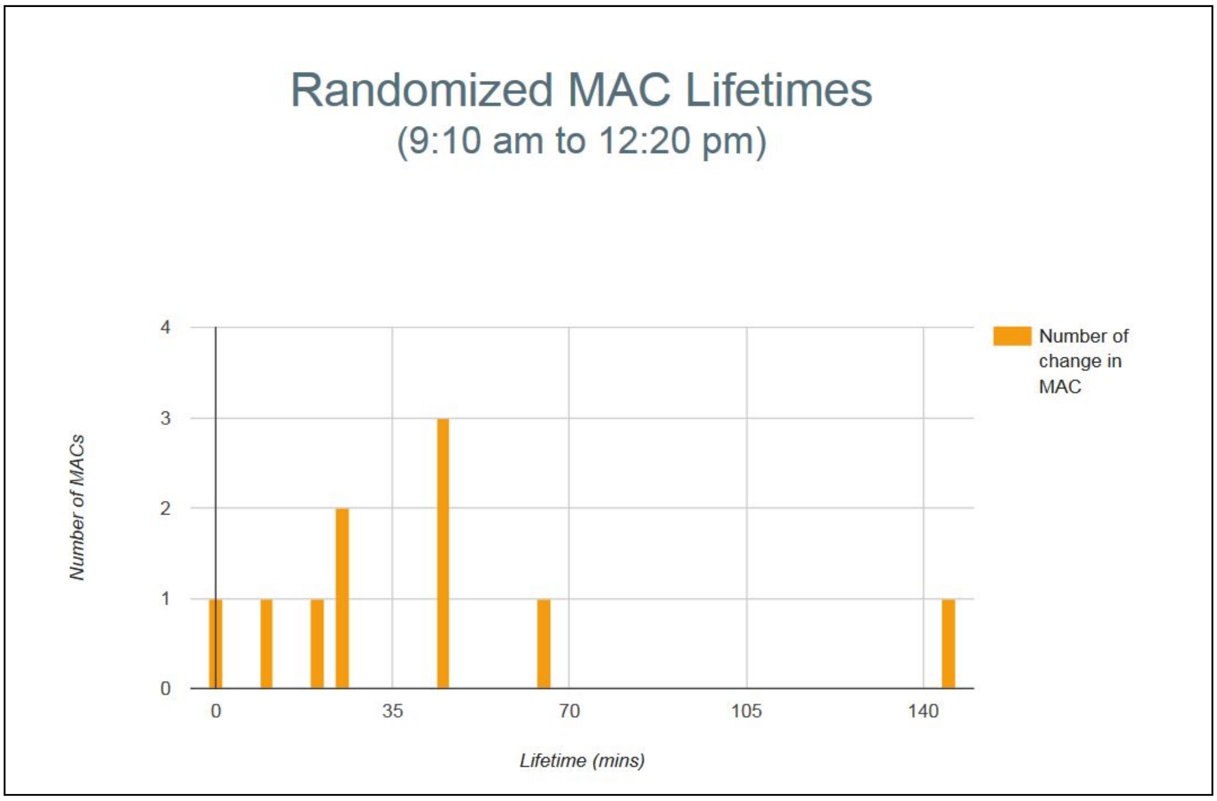


Figure 7. The lifetime of randomized MACs from the entire recording period.

V. Live Data Design

We develop a prototype system to process live data that is continuously sent from Wi-Fi sensors in a Raspberry Pi. We use the tcpdump command-line tool to capture and analyze the packets in the Wi-Fi network. Figure 8 illustrates a schematic of the data pipeline from an IoT perspective, where we use the design below to simulate the interactions of a distributed network of sensors to count people in a large event.

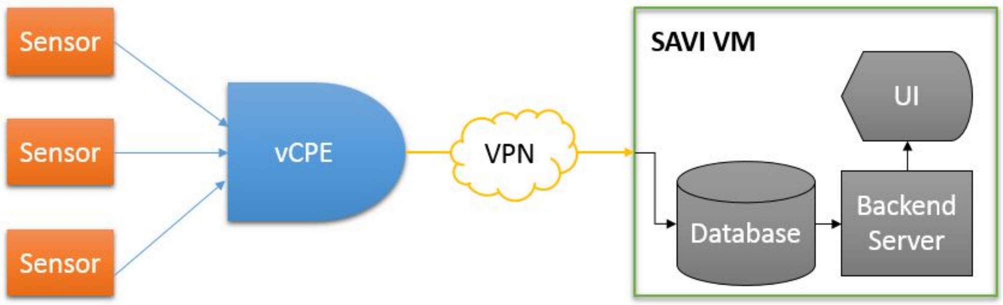


Figure 8. System overview.

A. Sensors

To capture the Wi-Fi packets, we use small development boards with Wi-Fi capabilities, like the Onion Omega, or the Raspberry Pi with a Wi-Fi USB adapter. The packets are captured and parsed using a python library called Scapy. The captured data is sent to the vCPE wirelessly. We assume the position of the boards is static, and that we know each position.

B. vCPE

In the data pipeline, the vCPE serves as an intermediary device to ensure data is reliably transferred from the Wi-Fi sensors to a Virtual Machine (VM) in the SAVI cloud. The vCPE is configured to be an access point, to which the sensors directly connect. We use ZeroMQ, a lightweight distributed messaging tool, to establish a request-reply TCP socket connection. The Wi-Fi sensors act as the clients, and send the data to the vCPE, which acts as a central server and aggregates the data from the sensors.

C. *VM*

To set-up secure communication between the vCPE and VM, we integrate an OpenVPN connection. The sensor data is sent using a RESTful API, with the server running on a VM. The data is then stored in a MongoDB database. From there, a backend service performs an analysis of the data, and serves a front-end to the user, shown in Figure 2. We use the same front end visualization tools as described in the Real World Testing section.

VI. Future Work

The main aspect of the future work is to develop this into an IoT application. By using many sensors deployed in the field with a robust communication system, real-time analytics on crowd movement can be performed. For both events and traffic, this would allow immediate reaction to issues such as bottlenecks. Furthermore, live video, either from stationary cameras or from a drone, could provide the number of people or number of cars to use as a reference.

This could also be developed for security purposes. For example, an event organizer could have a web portal where users can link their MAC address with their name and contact information.

This would allow the organizer to find the location of a specific person in the event using an array of Wi-Fi sensors.

VII. References

[1] “Analysis of iOS 8 MAC Randomization on Locationing,” Zebra Technologies, 2015. Available: http://mpact.zebra.com/documents/iOS8-White-Paper.pdf

[2] B. Misra. (2014). *iOS8 MAC Randomization - Analyzed!* [Online]. Available: http://blog.mojonetworks.com/ios8-mac-randomization-analyzed/

[3] *Android 6.0 Changes* [Online]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

[4] M. Vanhoef, C. Matte, M. Cunche, L. Cardoso, and F. Piessens. “Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS ‘16)*, May 2016, pp. 413-424.