TRAKr
CROWD ANALYTICS

Peter Ott

Abstract

The aim of this project is to build a scanner that gathers data from the wireless environment and is able to produce analytics. These analytics are intended to be used by businesses or organizations to help gain insights to the way patrons or visitors come and go from their location. The users of TRAKr will be able to see how many devices are present at a given time as well as other metrics, and with this, draw conclusions about how many people are present. TRAKr is intended to be run on a small, inexpensive computer, though it can be run on a compatible laptop as well.

Introduction

This project is aimed at creating a tool that can be used by businesses, offices, municipalities, and anyone wishing to gain metrics on the crowds that frequent their location. The tool will allow users to gain insights into how large of a population is present and how it changes over time, as well as noticing trends over days or weeks of use. Businesses can use this data to make informed choices on what times to have more or less staff, evaluate if a sale brought in new customers, and more. It can also be quickly set-up and used to gather participant data at events that only last a day or two such as a farmers' market or a music festival.

There are two main use-cases. The first is running the application on a laptop with a wireless card capable of network monitoring. This will be an all-in-one solution, allowing both the monitoring and the analysis to be done on one machine. This would be the ideal situation for someone wanting to test out TRAKr without investing much. The second scenario is the intended use case and is deployed on a small, inexpensive, and portable computer: a Raspberry Pi[1]. The Raspberry Pi would need to be paired with a wireless card capable of network monitoring. The Pi's limited power does not suffice for analysis, only scanning, so its capture data will have to be unloaded for processing at a later point. This can either be done by taking the files from the Pi via network file transfer or by setting up a storage media device and saving output to that. When the data is loaded unto a stronger machine, the analysis will be run and the output data will be produced all at once. This may take some time as hours or days of capture data must be processed. This is the intended use case because a Raspberry Pi can be inexpensively purchased and powered, plugged in somewhere inconspicuous, let run for an indefinite amount of time using very little electricity, and allow the user to

[1] https://www.raspberrypi.org/

take data off of it as they please. Additionally, it can be powered by a battery pack, allowing its use to be set up somewhere for more on-the-go uses without the need to devote an entire laptop to TRAKr.

TRAKr operates by listening on the wireless environment for any Wi-Fi traffic. The nearby devices do no have to be connected to a Wi-Fi access point, they only need to be on. This sets TRAKr apart from most solutions that require a device to be connected to a wireless network. This capability is made possible by the way Wi-Fi devices broadcast a searching beacon when they look for wireless access points or they simply are already connected to a network and give off traffic. Packets are sent at variables rates, depending on the devices internet activities, they could be as frequent as 100 per second or as sparse as 1 every 10 seconds. Whatever the packet rate is, TRAKr will pick each one up and extract the semi-unique MAC address from the sender and file it away for analysis later.

Due to the large amounts of data that are taken in from wireless environments, TARKr reduces its dataset regularly while running. Data reduction operates in a breathing-like style. TRAKr captures packets for a set time (the default is 5 minutes), building up its collection of capture data. Then, it reduces all of the held packets, exhausting it's collection. This happens on a loop as TRAKr is run.

With the data in this reduced state, TRAKr can be manually called to analyze the data and export a file. This operations is a more intensive process and can take several minutes or even a few hours depending on the size of the data. In this process, TRAKr scans through each entry and exit action chronologically, keeping track of what each device is doing at the incrementing time. The data produced is in the form of a Comma Separated Values (.csv) file and is then able to be imported to most spreadsheet programs such as Microsoft Excel or Apple Numbers for graphing and further use.

Background

TRAKr relies solely on WiFi traffic for gathering its data. There are many advantages to this, however it is also not without a few disadvantages. TRAKr identifies users based off of a semi-unique Media Access Control (MAC) address. This is a string of 6 pairs of hexadecimal values (commonly separated by colons for readability). For example: "1A:2B:3C:4D:5E:6F" is a possible MAC address. Each MAC address is commonly comprised of two parts: the first 6 hex values (the prefix) which correspond to a hardware manufacturer, and the last 6 hex values (the suffix), which are randomly chosen.

There are about 16 million unique suffix addresses for each hardware manufacturer prefix. Though commonly, hardware manufacturers have multiple prefix addresses. This is a small number in comparison to how many wireless devices are produced, meaning that a duplicate MAC address is highly likely, almost certain, to exist. However, hardware manufacturers are aware of this and attempt to ship devices with duplicate addresses to separate parts of the world to minimize overlap. Ultimately, because a MAC address is not completely unique, it is not considered as personally identifiable information. But it is close enough and TRAKr thrives because of that.

When a WiFi enabled device comes into the area where TRAKr would likely pick up its signal, it is in one of three states: Connected to a wireless network, searching for a wireless network, or WiFi has been disabled. If the device is connected to a network, this is the ideal situation for TRAKr, as the device will keep a static MAC address and be emitting wireless traffic regularly. In the second situation, while searching for a network, some devices send out a "poll request" to notify wireless access points that they are looking for a connection. this traffic is enough for TRAKr to find devices and log them, however MAC address randomization occurs here and interferers with data.

MAC address randomization is when the device sends out poll requests with a spoofed MAC address that is randomly generated on-the-fly in order to not allow surveillance of their actual MAC address. However, once connected to a network, the device will reveal it's true MAC address. The third situation in which WiFi has been disabled on the devices has no countermeasure, that device will not be detected.

There are existent versions of this software, however they are closed off and are only available as a paid service. One of the most notable companies offering this is Euclid Analytics with their "Insight"[2] product. Euclid Analytics was started in 2011 by a few engineers, some of which came from the Google Analytics team. They also offer a more powerful service "Connect" that utilizes a login through in-store Wi-Fi to gather even more data such as email, age, and gender.

Euclid Analytics' "Insight" product has many over laps with this project. Both are designed to product crowd analytics. However, their solution is enterprise oriented and is intended to be used only for long term. TRAKr is able to be used short-term or long-term and can be used on a private or enterprise scale.

A slightly different Wi-Fi tracking tool is produced by Solar Winds under the self-evident name of "User Device Tracker"[3]. This software is an IT management tool, though, intended on keeping track of a companies or customers devices and assets within a network. It also has more IT oriented capabilities such as supplying information about network use and individual user tracking. While its intention may be different, it uses much of the same technologies as "Insight" and TRAKr.

The technology being used to gather the MAC addresses has existed for years; it is intrinsic to wireless communication. When a wireless transmission is sent, it has a destination address. A wireless antenna that is listening for transmissions will pick up

---

[2] http://euclidanalytics.com/product/euclid-insight/

[3] http://www.solarwinds.com/user-device-tracker

all nearby transmissions, but only accept the ones that have its address as the destination. This is like talking to another person in a noisy room and only paying attention to that one other person. However, some wireless antennae can be set to a "promiscuous" mode which allows them to accept all transmissions, regardless of the intended destination. This is much like being able to stand in a crowded room and listen to everyone present. This is the technology used to gather wireless data. The wireless transmission data that can be scraped are the unique addresses of the sender and the receiver ensuring that regardless of direction, the unique MAC address will be found.

There also exist other ways to gather metrics on crowds. SRI International uses computer vision to detect humans captured on video. Their project to do this is called [4] and is intended as counter-terrorism. This method is more complex, obviously requiring cameras to be set up and more powerful computers to run the analysis.

[4] https://www.sri.com/work/projects/tracking-humans-crowds

Problem Statement

While closed source solutions exist for gathering crowd analytics, they are very costly. This project is aimed at lowering the barrier to entry of gathering Wi-Fi based crowd analytics, allowing any party interested in using such data to easily have access to it and be able to benefit from the insights produced. This will put innovative data into the hands of anyone who could want it. Allowing small businesses and other entities without the funds to use larger scale commercial systems to obtain similar data and be able to keep up with their competition.

All of the currently existing solutions are closed source. These solutions also may have privacy concerns over how the data is treated as the program running to gather it is a black box, giving no knowledge to the user what is happening with their data. On the contrary, this project will be open source, allowing transparency in what the software does, because the users are able to inspect every line of code. Additionally, it will allow the user to handle their own data that is produced.

Finally, for my own personal benefit, I would like to learn more about data analysis, much like that done in big data scenarios. Also for my own personal benefit, I would also like to be the author of an open source project that has the potential to be disruptive in a very exclusive market.

Solution Overview

TRAKr gathers data from the wireless environment around it through the use of a special wireless receiver that can use “promiscuous mode” which is also known as “monitor mode” to receive all traffic in the area, instead of just those transmissions intended for the device. Wireless data comes in the form of a packet. Each packet has information such as time, protocol information, the information the device is communicating with (usually and hopefully encrypted), and the MAC address of the sending and receiving devices. The MAC address part is all TRAKr cares about, so it strips out the addresses and files them away into a database with a timestamp of when it was seen. This completes the monitoring aspect.

The database full of MAC addresses and the times they were seen at is substantially large due to wireless devices sending easily a few hundred packets per second. A data reduction algorithm is run to reduce potentially tens of thousands of packets into just a few crucial data points, while completely preserving the data needed to produce accurate analytics. The resulting data is simply when a device entered the wireless environment, and when that device had seemingly left.

The final stage is analysis in which the reduced data points are used to keep track of how many devices are seen at a given time interval. The algorithm iterates over all the data points and produces data of the wireless environment for each 5 minute interval. The data consists of the number of devices that have entered, the number of devices that have exited, the maximum number of devices currently present within the interval, and the number of devices that have been seen at least once before. This data is outputted into Comma Separated Values (.csv) file which can then be imported to Excel or a similar spreadsheet application for easy visualization and further use.

Technical Overview

TRAKr is written in Python 2.7 and employs a few libraries to assist its functionality. TRAKr is used through the terminal and only requires a few commands to utilize the core functionality, allowing those who are unfamiliar with a terminal to use it and those who are more adept to gain more control. TRAKr's runtime can be thought of as three distinct sections which pass information from one to the next. Data Gathering, Reducing, and Exporting. Multiple instances of TRAKr can be run simultaneously. This allows the user to export data without interrupting a scan. There is a configuration file in TRAKr's directory that allows the user to control some functionality. This file is located in the TRAKr directory as "trakr.ini".

Data gathering will run infinitely and stops when the user interrupts it through the terminal. TRAKr will use the wireless interface specified in the configuration file to capture traffic. Wireshark's tool "tshark" is used for sniffing with the wireless interface. Super user functionality is required for running wireless sniffing.

There are two ways in which TRAKr can gather data. The first is a two step process: scanning then loading. This is ideally run on a small portable computer such as a Raspberry Pi. Gathering data in this way will output the scans as Packet Capture (.pcap) files. These will then need to be loaded later on on a more powerful computer. The second style of data gathering is an all-in-one style where TRAKr is run on a more powerful computer that can handle the intensive task of extracting and keeping up with the input from the scanner. This eliminates the need to extract the data later on and saves the user time. After scanning and extraction, the data is in a database in the form of a MAC address (after it has been hashed by the SHA-256 algorithm[5] for privacy

[5] https://en.wikipedia.org/wiki/Sha256

which is enabled by default) and the time that it was seen. This dataset is very large and will be reduced in the next stage.

Gather data and run .pcap extraction like so:

```
sudo python TRAKr.py -run
```

Gather data without running data reduction like so:

```
sudo python TRAKr.py -scan
```

Load a single .pcap file like so:

```
python TRAKr.py -load /path/to/file.pcap
```

Load a directory of .pcap files like so:

```
python TRAKr.py -loaddir /path/to/file.pcap
```

Data reduction starts with a large database of all instances of noticing any devices in the wireless environment and ends with a in a significantly reduced file size and entered in TRAKr's database. This must be run manually. TRAKr parses through the large database and uses the parameters in the configuration file to determine actions the users are doing. When a device is first noticed, it must be present for "entry_time" amount of seconds before TRAKr will decide that the device is truly there and not simply passing by. Then if TRAKr stops noticing the device, it will mark the device as having exited. The device must be away for "exit_time" seconds before that happens. When finished, a second database is filled with entry and exit times of each device. This is immensely helpful to both performance and file size. The file size reductions when going from one database to the next are usually around the magnitude of 10^-2 so a 50Mb database of all the observations would be reduced to a 0.5Mb database file of actions, approximately. And the number of data points are reduced in a similar scale. Usually, tens of thousands of data points from a wireless device being present for a long duration of time are reduced to simply an entry time and an exit time.

Run analysis on all loaded data like so:

```
python TRAKr.py -analyze
```

Exporting the data is the fastest step of runtime. Simply put, TRAKr loops through it's database of actions, takes note of how many devices are present at any given time, and exported to a file. The exported data is in the form of increments per hour, which is able to be set by the user in the configuration file. For example, if the user specifies 12 increments per hour, then the data will be formatted as how many users are present in a 5 minute interval of time, dividing the hour into 12 such data pieces. The outputted data is in the form of a Comma Separated Values (.csv) file. The exported file will be located in the TRAKr directory as "trakr_export.csv". This can be loaded into Microsoft Excel or a similar spreadsheet data processing application for extended use and graph creation.

Export data like so:

```
python TRAKr.py -export
```

References and Research

I am in contact with Prof. Kim who specializes in wireless traffic, and a graduate student, Kevin Andrea, who is working in wireless location tracking. The commercial tools that are able to accomplish this type of tracking do not supply any information as to how they go about processing their data or even that they deal with MAC addresses at all. There are a few papers that give good insights into what types of tracking are possible over Wi-Fi. Two I've examined are "WiFi Told Me Everything About You" and "I know your MAC Address: Targeted tracking of individual using Wi-Fi" both by Mathieu Cunche. They investigate what types of metadata can be gathered, what conclusions can be drawn, and what civilians can do to mitigate tracking.

Research Timeline

March 3rd - Data capture and extraction of important aspects of packet traffic.

Use a library to parse packets produced from a network packet capture session and extract the relevant address and time information.

March 17th - Basic time block tracking of a single device

Using the address and time data from above and a SQL database to load all information from a capture session.

March 31st - Dataset reduction strategies to eliminate repetitive data points

Wireless devices are very sporadic and chatty and thus produce a vast amount of information that is not needed. The useful information is the entry and exit time, which are just two time points compared to what could easily be over 10,000 data points from a single device entering and exiting during a capture session.

April 14th - Basic heat map of user count

Using the reduced data above (entry and exit times) create a heat map based on a single devices history of actions. This heat map will be in the form of number of devices that have been seen in an arbitrary time interval. This will be in a form similar to 3:00 - 35 devices, 3:05, 37 devices, 3:10 - 40 devices, etc…

April 21st - In-depth crowd analytics and output

Create a Comma Separated Values (.csv) file report file based on all of the data gathered. This will contain the heat map discussed above along with number of new devices seen per interval. The .csv file will be importable to Microsoft Excel (or other similar spreadsheet program) to allow for graphing and further data analysis.

Expansions and Future Plans

For extended functionality and more useful analytics, I would like to expand TRAKr's functionality to include returning device count. This would be useful for gauging customer return rates. Built in network file transfer would also make using a Raspberry Pi much easier, eliminating the need to offload the .pcap files each time analysis is needed. Adding in bluetooth monitoring capabilities would be easy and it would work identically because they too have MAC addresses. Finally, and perhaps the most important future idea, is adding in a graphical user interface. Terminal use is just fine for development, but the overwhelming majority of intended users are not familiar with the terminal and a gui would help increase the user base greatly.

Evaluation Plan

Field testing is key to assuring this project works properly. Initially, I will create samples of data by monitoring my own devices in a known fashion. Validating that I am able to correctly identify entry and exit times is the foremost ability that needs to be verified. Next, I will capture in a location that I know has many devices enter and then subsequently exit. This will verify that it is able to correctly apply entry and exit times to a large population. With the operational basics working, testing can move on to producing and verifying the analytics. Running TRAKr for a prolonged period, such as in my dorm and filtering the results to my own unique MAC addresses, I would be able to verify if the heat maps created are accurate. Finally comes the full scale analytics testing. I will do this at a place where I generally can assume what the traffic is like, such as at a church on Sunday. I will run it for multiple weeks and verify that the output data correlates approximately with the number of people present.

Testing and Results

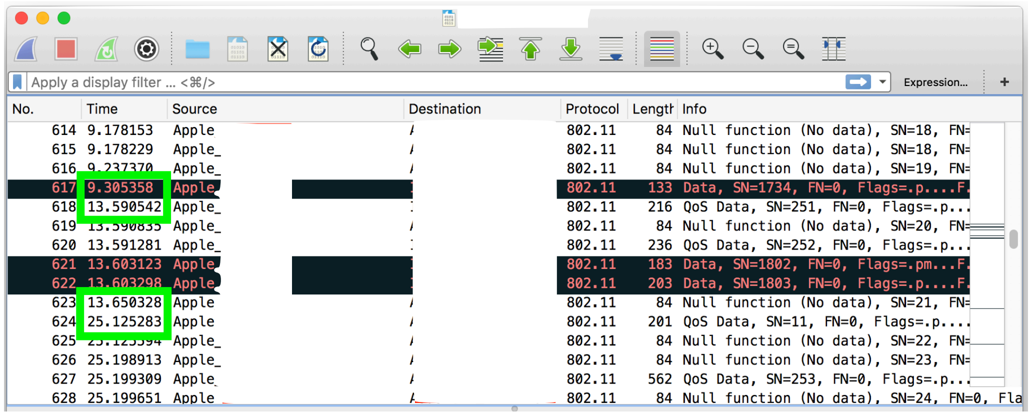
Testing was performed on a few scales, each as a prerequisite for the next. First, the system needed to be tested to correctly identify wireless devices by their MAC addresses and the times that they were seen. Secondly, the system was tested to correctly identify entry and exit times. With wireless traffic being noisy and nonuniform, this functionality would require tolerance to interference but still result in accurate identification of actions. And finally, a large scale test of many users entering and exiting a location would be performed to test the main purpose of this project: to estimate crowd populations and graph their change over time.

First Test

The basic functionality need for TRAKr is retrieving the correct MAC addresses from devices. Generally, this test is to determine that from a single packet, the MAC address of the transmitter can be retrieved. The tool tshark is used for this functionality as it is a well known and established tool. It is the terminal counterpart to Wireshark[6], a well-established and perhaps the definitive tool for inspecting packets. In developing this functionality, multiple types of packets needed to be accounted for. For example, some packets are beacons which are meant to be sent to all devices, and there

[6] https://www.wireshark.org/

Second Test



```
No.  Time       Source      Destination   Protocol Length Info
 614 9.178153   Apple       /             802.11      84 Null function (No data), SN=18, FN=
 615 9.178229   Apple_      /             802.11      84 Null function (No data), SN=18, FN=
 616 9.237370   Apple_      /             802.11      84 Null function (No data), SN=19, FN=
 617 9.305358   Apple                     802.11     133 Data, SN=1734, FN=0, Flags=.p....F.
 618 13.590542  Apple_                    802.11     216 QoS Data, SN=251, FN=0, Flags=.p...
 619 13.590835  Apple_      /             802.11      84 Null function (No data), SN=20, FN=
 620 13.591281  Apple_                    802.11     236 QoS Data, SN=252, FN=0, Flags=.p...
 621 13.603123  Apple                     802.11     183 Data, SN=1802, FN=0, Flags=.pm...F.
 622 13.603298  Apple                     802.11     203 Data, SN=1803, FN=0, Flags=.p....F.
 623 13.650328  Apple       /             802.11      84 Null function (No data), SN=21, FN=
 624 25.125283  Apple       /             802.11     201 QoS Data, SN=11, FN=0, Flags=.p....
 625 25.125394  Apple_      /             802.11      84 Null function (No data), SN=22, FN=
 626 25.198913  Apple_      /             802.11      84 Null function (No data), SN=23, FN=
 627 25.199309  Apple_      /             802.11     562 QoS Data, SN=253, FN=0, Flags=.p...
 628 25.199651  Apple       /             802.11      84 Null function (No data), SN=24, FN=0, Fla
```

Wireshark is displaying some packet capture data of my iPhone. I had turned WiFi on and off in a timed fashion. Notice how there is a gap between 9 and 13 seconds as well as 13 and 25; this was when my WiFi was **turned off. For this test, the time requi**red for the device to have exited was set to 10 seconds, so the device should not have left during that first gap, as it's time of absence was only 4 seconds, but it should during the second, due to being absent for 11 seconds. The results can be seen via these database entries:

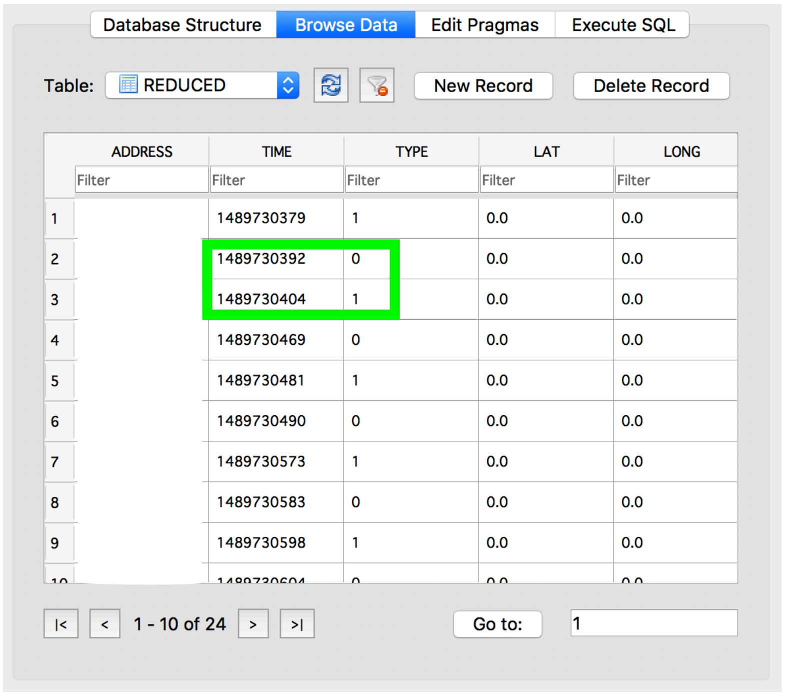


Table: REDUCED

	ADDRESS	TIME	TYPE	LAT	LONG
1		1489730379	1	0.0	0.0
2		1489730392	0	0.0	0.0
3		1489730404	1	0.0	0.0
4		1489730469	0	0.0	0.0
5		1489730481	1	0.0	0.0
6		1489730490	0	0.0	0.0
7		1489730573	1	0.0	0.0
8		1489730583	0	0.0	0.0
9		1489730598	1	0.0	0.0
10		1489730604	0	0.0	0.0

1 - 10 of 24

The two database entries highlighted in the green box are in the format of [unix epoch time in seconds, entry (1) or exit (0)] and the interpretation of that data is:

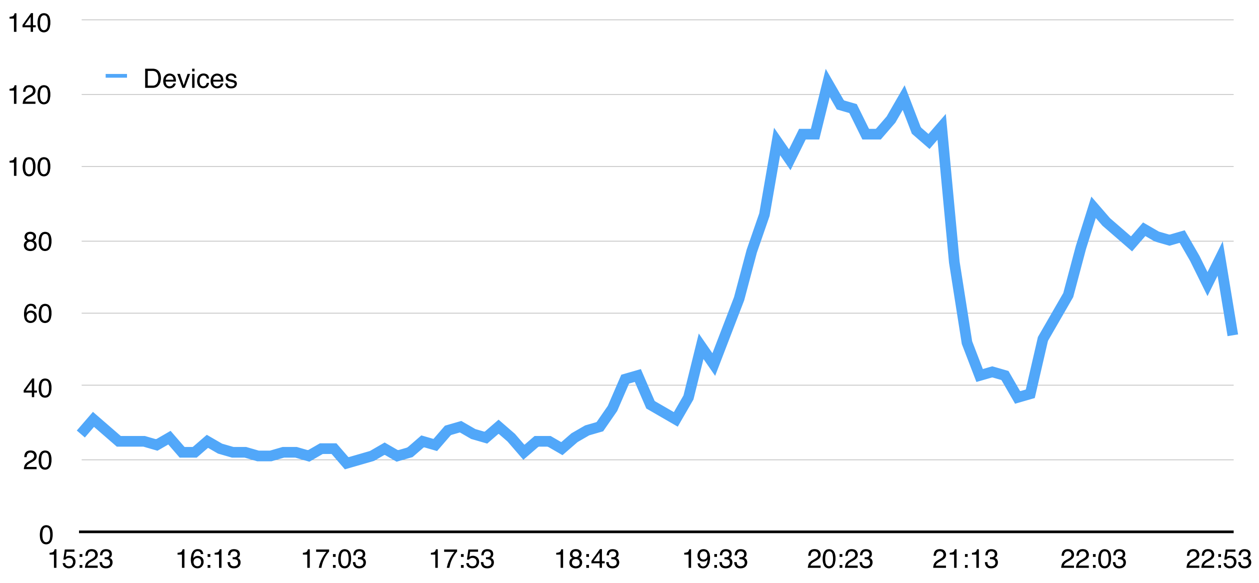
[1489730392, 0] means [13.65 seconds, exit action]

[1489730404, 1] means [25.12 seconds, entry action]

The entry action above the green highlighted box was when the device was first seen, which was at the beginning of the test.

Third Test

Monitored during a period of time in which two church services occurred. The services were at 8PM and 10PM. It is commonly known that typically the 8PM gets more attendants than the 10PM. Below is a graph of the data of time vs number of devices. It can be see that the church-goers start to head in to the 8PM mass around 30 minutes before, a large population stays for the approximately hour long service, then they proceed to trickle off with some staying. There is a basement in the church where community members spend time, this likely explains the 20 to 50 devices present when there is no service. After the 8PM comes the 10PM where a smaller, but still substantial group enters the range of the scanner, stays for about an hour, then leaves. The monitoring was ended shortly after because the church was locking up for the night and I needed to take my device. The data outputted approximately matches up with the estimated amount of people who attended each service. It was about 10% to 15% lower than the usual attendance, however this is likely due to people disabling WiFi on their devices when they leave their house or turning their phones off for the church service.



Conclusion

TRAKr performed accurately through all the tests when the variables “entry_time” and “exit_time” were tweaked toward the population that was being monitored. I ran analysis on the church data multiple times under various settings and found that having a higher entry_time variable reduced the total population because it would not account for the randomized MAC addresses. For example if a device randomizes its address every minute and it takes two minutes to be present, that device will not be counted. However, if the entry_time variable were lowered to say 45 seconds, then the data would include more noise as passers by would also be included. This slightly hinders results, however from the testing, it seems as if very few devices actually take advantage of MAC randomization, so overall this not a significant problem. It is also somewhat resistant to this, because as a device changes it’s MAC address, if TRAKr does notice it, it will be an exit as the address leaves followed by and entry as the new MAC address shows up, totaling to zero change in population.

The one aspect I am not satisfied with is that TRAKr must be run through a terminal. This is such a foreign method of using a computer to the vast majority of people. If I were to have worked with a partner for this project, making a graphical user interface would have been one of the things accomplished, allowing for a much larger user base due to a much friendlier application.

In the end, TRAKr does produce the results I was looking for. I originally intended for it to be able to be run completely on the Raspberry Pi, however the Pi could not keep up with large influxes of wireless traffic and began to get drown on data, queuing up multiple .pcap files for extraction. Even with the performance enhancements and concurrent aspects of TRAKr, the extraction and analysis must be run on a stronger computer.

Bibliography

Literature:

"I know your MAC Address: Targeted tracking of individual using Wi-Fi"
by Mathieu Cunche

"WiFi Told Me Everything About You"
by Mathieu Cunche

Analytics Products:

Euclid Analytics	http://euclidanalytics.com/product/euclid-insight/
SolarWinds	http://www.solarwinds.com/user-device-tracker
SRI	https://www.sri.com/work/projects/tracking-humans-crowds

Software Tools:

Python 2.7	
tshark	https://www.wireshark.org/
pyCLI	https://pythonhosted.org/pyCLI/
pcapy	https://github.com/CoreSecurity/pcapy
libdnet	https://github.com/dugsong/libdnet

Hardware Used:

Raspberry Pi:	https://www.raspberrypi.org/

Appendices

User Manual

Requirements:

- UNIX or MacOS computer
- Python 2.7 (or above, but not Python 3)
  - Python libraries: libdnet, libpcap, pyCLI,
- tshark (The terminal version of Wireshark)
- Wireless card with Monitor Mode (Macs do this natively, for others do research)

Installation

- Place TRAKr folder somewhere on your computer
- Navigate to it on the terminal
- Run as:

```
python TRAKr.py <your options here>
```

- Keep reading to the Parameters Guide to operate

Parameters Guide

Command	Description
-run	runs TRAKr all-in-one, does monitoring and analysis
-load /path/to/file.pcap	loads file.pcap into the database
-lat <num>	Sets the latitude of the loaded file to <num>
-long <num>	sets the longitude of the loaded file to <num>
-analyze	analyzes what is in the database, see config file parameters: back_track_hours and do_all_analysis_forever
-scan	scans and saves to a file
-deleteDB	deletes the current database
-reset	resets the config file to default values based on TRAKr's location in the file system
-load <pcap>	load a single packet capture file into the database
-loaddir /path/to/dir	load an entire directory of packet capture files into the database
-export	export the analyzed data into trakr_export.csv which will be placed in the TRAKr directory

Config File

Parameter	Prupose
[trakr]	
full_path	location of the TRAKr directory in the file system
autostart	unused
key	unused
[db]	
rolling_path	path to the database of all captured traffic
behavior_path	path to the database of the behavior data
graph_path	Unused
hash_values	hash device MAC addresses for anonymity
[scanner]	
interface	interface on which to run the capture
duration	seconds to capture network traffic and analyze
capture_dir	location of outputted capture files
keep_all_pcap	if True, all pcap will be kept if False, pcap files will be deleted after loaded into database
[analysis]	
exit_time	seconds for a device to not be seen to be considered exited
entry_time	seconds for a device to be seen to be considered present
seg_per_hour	number of segments to analyze per hour
back_track_hours	how many hours backwards to run analysis
do_all_analysis_forever	if True, will analyze everything in the database, this could take a very long time if False, will use back_track_hours to do analysis

Tweak Guide

Most of how TRAKr runs and gathers behaviors is based off of the two parameters *entry_time* and *exit_time*. To more easily filter out the quick visitors, increase *entry_time*. This will only allow devices that have been present for a longer duration to be considered present. This may be helpful in a very crowded area where many people pass by, such as a coffee shop on a busy street corner. The *exit_time* is the other parameter that strongly influences the data, though not as much as *entry_time*. *exit_time* can be decreased to a very low number if multiple instances of TRAKr are being used in somewhat-close proximity to each other. This could be helpful in a convention center, allowing a device to move from place to place more frequently but also correctly identifying entries and exits.

MAC address randomization will slightly affect your data. If you are suspicious of a large portion of the devices using this technology, a lower entry_time will still count an accurate approximation of the devices. This is due to the device only being able to emit one MAC address at a time. So, there may be more entry and exit data points created as the device drops one address for a new one, however the overall population remains unchanged as each exit action of the previous address is followed immediately by an entry action of the new address.

Both entry_time or exit_time are not dependent on the monitoring interval or any other variable for that matter, the only runtime parameter they affect, though indirectly, is each other.