CS601 Capstone Proposal

Michael Tripp

10/13/2022

Objective

Build an application that uses a technique called Wi-Fi trilateration to build a three-dimensional model of a given network provided a scan of the network and at least three known locations of network devices as input.

Background

This app will attempt to use a type of Wi-Fi positioning to determine the locations of transmitting wireless devices. There are two primary methods to implement Wi-Fi positioning: Wi-Fi trilateration and fingerprinting.

Wi-Fi trilateration is a technique that uses a measurement called RSSI (Received Signal Strength Indicator), a measurement inversely proportional to distance, to determine the location of the target device relative to given access points. This method requires known locations of at least three fixed access points, such as Wi-Fi routers or fixed hotspots. Then, using these locations along with the RSSI data, the distance between the device and each access point can be estimated and used to approximate the devices relative position. It can otherwise be called multilateration when more than three access points are used.

It is worth noting that there is a variation of this method called Wi-Fi triangulation. While similar to trilateration, a major difference is that Wi-Fi triangulation determines position based on "angles of arrival (AoA) of signals received by the antennae of the access points" whereas Wi-Fi trilateration determines position based on known distances (see Figure 1 below) [1].

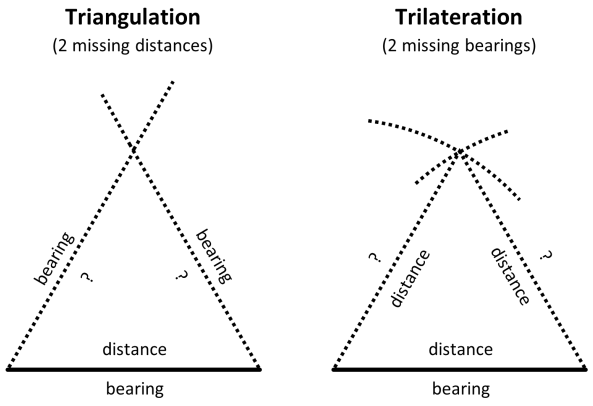

Figure 1. Triangulation vs Trilateration. Note that bearings are angles. Courtesy of stack exchange user [12].

Wi-Fi triangulation, however, requires MIMO antenna systems, hardware which may not be available for the purposes of this capstone. The algorithm for this application will be using the FSPL (Free Space Path Loss) formula to approximate the necessary distances, so trilateration can be used instead.

FSPL attempts to account for “the attenuation of radio energy between the feedpoints of two antennas” due to obstacles and other obstructions [11].

Building upon these ideas, fingerprinting can also be used. Fingerprinting is said to significantly improve the accuracy of the previous two methods by utilizing historical RSS data in addition to knowledge of known device locations to approximate position. The problem with this method, however, is that it is highly environment dependent. It relies on the environment surrounding the network devices to stay the same, as the historical data record assumes the same conditions. Even just moving a chair or table could disrupt the signal strength values—as moved furniture could now be dampening the signal or could even free up line of sight from a previously blocked signal—providing less accurate historical data unless the fingerprint (environment) was updated accordingly [2]. This may be something to investigate down the line but implementing Wi-Fi trilateration will be the first and foremost goal.

Previous Work

There are two major ways I see this technology being utilized today. One is with indoor positioning and the other is positioning with the use of external sources such as Wi-Fi towers and GPS data.

One leading company that makes use of this first method is Navigine. Navigine is a company that provides indoor positioning and navigation through Wi-Fi. This technology can be applied in places such as shopping malls, large universities, and museums to help visitors navigate throughout the building as their position can be updated on a map in real time. Navigine implements this using a multilateration approach using RSSI and MAC-address data. A MAC-address is a unique number assigned to a network device that is used to identify it over the network.

Another leading company implementing a solution to indoor navigation is Mapsted. Mapsted’s technique is particularly unique as they are able to determine a mobile device’s location accurate to 1-3 meters without beacons, Wi-Fi, or any additional external hardware. Rather, their algorithm “uses magnetic or wireless disturbances in the environment to learn and determine location, essentially converting that ‘noise’ into useful information” [7]. As such, they claim this technology works with any “off-the-shelf smartphone.”

One primary case of the second method is with Google’s Geolocation API (application programming interface). Rather than only using positions of local network devices, Google determines a user’s position based on the absolute positions of “cell towers and Wi-Fi nodes the mobile client can detect” [3]. Thus, Google can query a device’s location without needing to know the positions of local network devices since it knows the locations of cell towers/Wi-Fi towers. I am not sure whether triangulation or trilateration/multilateration is used here.

On an even broader scale, it is worth noting that GPS also works in a very similar way. Although it does not make use of Wi-Fi positioning specifically, it does use trilateration to locate a specific position on Earth. Rather than using locations of network devices or cell towers, however, satellites are used instead with “a GPS receiver [that] measures the distances to satellites using radio signals” [6].

Implementation

While it is clear this technology is being used in many different facets, many of these implementations are more focused on corporate use and user navigation. They provide a specific way the data is meant to be used, only providing enough information to the user to be able to navigate through a building. I plan to take a more general approach by using Wi-Fi positioning to build a visual model that can be used for whatever informational purposes are desired by the user. Unlike Navigine or Mapsted, which only show the user's device, my application will attempt to display all network devices detected, building as complete a model of the network as possible. While Google's Geolocation API also allows the user to query it for general use, it does not tell you the locations of other local wireless devices, nor does it provide a visual component.

I plan to implement this approach using a packet sniffer and the Unity game engine. A packet sniffer, also known as a packet analyzer, is a program used to collect packet information of data transmitted over the network. The implementation can be broken down into the following steps:

1. Record locations of at least three fixed network devices.
2. Collect RSSI and frequency data by analyzing network traffic with a simple packet sniffer.
3. Read in data to Wi-Fi trilateration program and output device locations.
4. Create visual component by building a model of these locations in a Unity project.
5. Expand model to be 3-dimensional by recording elevation of access points as well.

1. Record locations of at least three fixed network devices.

This step should be relatively straightforward as long as three fixed network devices can be identified. Once doing so, GPS via Google maps can be used to record latitude and longitude positions. A potential feature could be to program in a "record location" button that will log the position of the device running the application. This way, the user will just need press this button at each of the network device locations.

2. Collect RSSI and frequency data by analyzing network traffic with a simple packet sniffer.

The original plan was to use Wireshark packet analyzer to take care of this step, but I don't see a way to integrate this with my application in a C# program. One option would be to write a pack sniffer from the ground up in C#. While this may work, I don't think it would be an efficient use of my time. Rather, I am going to use the Scapy library for Python, which already has features built in to modify, send, and capture network packets. Most importantly, it contains functionality to capture RSSI and frequency data along with a plethora of other information such as hostnames and IPs. While C# is used for Unity programs, it also provides integration with Python via third-party software, so I will simply take advantage of this when implementing my script.

3. Read in data to Wi-Fi trilateration program and output device locations.

This is the most important step of the project and will also likely be the most difficult. In my research, I have noticed this algorithm is not widely known and most people implementing Wi-Fi position simply have Google's Geolocation API do the work. While this API will be my fallback, I plan

to write my own code using research and other relevant sources available to do so. Based on what I have read so far, my plan for the algorithm is as follows:

1. Maintain an array of at least three network device objects containing latitude and longitude values.
2. Use the following FSPL (Free Space Path Loss) formula using signal strength and frequency of signal to determine approximate distance the targeted device is from each network device:

$$distance = 10^{\left(\frac{(27.55 - (20 * log_{10}(frequency)) + signalLevel}{20}\right)}$$

   This formula was modified from the FSPL formula given on Wikipedia by a stack overflow user [9].

3. Calculate latitude and longitude of targeted device by comparing distances gathered in step 2 with the latitude and longitude values from step 1.

4. Create visual component by building a model of these locations in a Unity project.

I will start by making a flat, 2-dimensional model of the network. Unity provides a way to make GUIs (graphical user interfaces) to represent a flat image for something like this. The only difficult part of this step will be determining how to make the model dynamically. To do so, I plan to resize the GUI based on the differences between the latitude and longitude values of each network device, with some extra room around the edges. Based on the size, I can establish a coordinate plane system. Then I will simply position each network device on the screen by taking the mod of each device's latitude and longitude values by the limits of the x and y axis.

5. Expand model to be 3-dimensional by recording elevation of access points as well.

Expanding this to be a 3-dimensional model will be the final goal of this project but will need to be an optional one due to time constraints (or technical limitations). As provided by a user on stack overflow, one way this could be done is to build a depth map, which is a list of the access points and their floor numbers []. Then, the "elevation" of a device can be determined by measuring which access points give the strongest signal. The more variety of access points across floors, the better. This method, of course, does not tell exact elevation, however, and may not be accurate depending on access point placement, but does give some representation of elevation in the form of floor numbers.

Given a solution to this problem, I can then build the 3d model in Unity by the same method in step 4, except now including elevation and the z axis. As the Unity game engine provides an intuitive way to create 3d environments, porting this to such a model will be simple.

Timeline

October 30, 2022	Detect and list available network devices on network Obtain RSSI and frequency data from these devices (step 2)
December 2, 2022	Determine a working method for Wi-Fi positioning (step 3): ● FSPL Wi-Fi trilateration method ● Google's Geolocation API ● Other
January 18, 2023	Build a 2d model of network in Unity Meet with Andy Changoway about using application on Westminster network
February 28, 2023	Determine feasibility of building a 3d model Build 3d model if able
April 14, 2023	Test, test, test Additional features ● Make model look pretty ● Look into fingerprinting ● Other features thought of along the way

Sources

[1] Wi-Fi Trilateration Using Node.js | by Matt Croak Code | Better Programming

[2] Do WiFi positioning systems still make sense in 2021? (mapsted.com)

[3] Overview | Geolocation API | Google Developers

[4] Exploring GoogleGears Wi-Fi Geo Locator Secrets - CodeProject

[5] HTML5 Geolocation API - How it works, demos & tutorials (geotargetly.com)

[6] Does Google Maps use triangulation? - All Famous Faqs (allfamousbirthday.com)

[7] Indoor Navigation Technology - Indoor Navigation Solutions for Businesses | Mapsted

[8] android - Wi-Fi position triangulation - Stack Overflow

[9] android - Wi-Fi position triangulation - Stack Overflow

[10] geolocation - How to calculate distance from Wifi router using Signal Strength? - Stack Overflow

[11] Free-space path loss - Wikipedia

[12] gps - Differences between triangulation and trilateration - Geographic Information Systems Stack Exchange