**RSSI based Trilateration code**

-Srinivas

-Technical Solutions Engineer

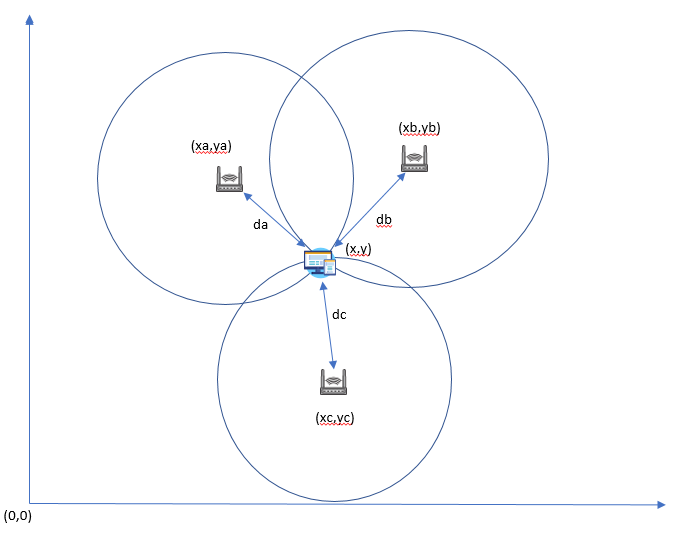
-Hughes Network Systems

1. Introduction:

This document gives a brief overview of the trilateration code written in python 3.x. The code was written to give xy coordinates of a client based on rssi values from different APs. The code works on the simple concept of triangulating the position from 3 distance value i.e the 3 sides of the triangle (hence triangulation). The rest of the sections explain the concept used, the code written by the author and the parts taken from online, and finally ends with some citations and plagiarism checker values.

1. Concept and code:

The basic concept of triangulation can be seen in the figure below:



In an ideal case (which it always isn’t) the client would lie at the perfect intersection point of atleast 3 AP regions. Using the known co-ordinates of each AP and the distance of the client from them, we can form 3 quadratic equations with the client coordinates (x,y) as the unknown variables. All that is left to do then is to solve them to get (x,y).

Now to get the distance of the client from each AP, we make use of the rssi values that are provided in the ale feed-reader. To convert the rssi values to distance we need to apply a pathloss formula. The pathloss or free space pathloss (FSPL) formula is a direct result from the Friss transmission formula, and is used to calculate the total attenuation a wireless signal undergoes from the source to destination. There are many different variations of the pathloss formula which take different parameters into account.

The formula used in this code is:

The parts of code showing the values taken by the author:

ptx=-30 #Average wifi ap received signal at a distance of <1m

n=2.5 #Pathloss coefficient for indoor scenario

The average reference power for an wireless AP is usually <-40dbm (good results was seen in –30 dbm) and the indoor pathloss coefficient is usually 2-4.

However, the author recommends using the pathloss formula which takes the signal wavelength (and inversely the frequency) into account. This would enable a finer calculation of the distance in case of 2.4Ghz and 5Ghz antennas as well.

The following lines in the code show this:

#Distance from AP1

x1=(ptx-rssi[0])/(10\*n)

da=pow(10,x1)

#Distance from AP2

x1=(ptx-rssi[1])/(10\*n)

db=pow(10,x1)

#Distance from AP3

x1=(ptx-rssi[2])/(10\*n)

dc=pow(10,x1)

Once the distances are found, we end up with 3 equations with 2 unknowns. This can be solved by the code shown below.

va = ((db\*db-dc\*dc) - (xb\*xb-xc\*xc) - (yb\*yb-yc\*yc)) / 2

vb = ((db\*db-da\*da) - (xb\*xb-xa\*xa) - (yb\*yb-ya\*ya)) / 2

temp1 = vb\*(xc-xb) - va\*(xa-xb)

temp2 = (ya-yb)\*(xc-xb) - (yc-yb)\*(xa-xb)

#Estimated user position:

ycorr = temp1 / temp2

xcorr = (va - ycorr\*(yc-yb)) / (xc-xb)

print("X co-ordinate: ",xcorr)

print("Y co-ordinate: ",ycorr)

The author doesn’t claim to have come up with this code. It was found in an example on Matlab which in turn was copied from R, which in turn was also adopted from the Wiki entry on trilateration. The author has merely written it in python (read ‘Spheres, equations and terminology’ by Paul Bourke[d] for more information on this). As such no one person can possibly claim ownership as this is a general method to solve such quadratic equations. While the author could come up with an alternative, it is something that could be changed if this code holds any water for the heatmaps.

The final piece in this code is the RMSE or root mean square error. The RMSE mathematically denotes the difference in the estimated client positions and the actual position that it should be. This piece has been added to serve as an indicator of how good the calculated location is.

#RMSE part

aps=3

mse=0

for i in range(aps):

dxcorr = pow((pow((xcorr-xp[i]),2)+pow((ycorr-yp[i]),2)),0.5)

mse+=pow((dxcorr-dmse[i]),2)

#print(mse/(i+1))

rmse = pow((mse/aps),0.5)

print("RMSE: ",rmse)

As such the smaller the RMSE is, the more accurate is the client position. This is because, while in an ideal case, the client would lie on the perfect intersection of the 3 circles, that is not always the case. Hence the values calculated here, on their own cannot be taken as accurate. The RMSE can then be used to minimize this, by plugging in the obtained values to calculate the distance from each AP to client. This is then compared to distance obtained from rssi in the previous step. However, it is like an average value in all directions and not in any fixed direction. The way it works is, that for different clients, the RMSE would be different. Hence using a few fixed clients, the user could change code parameters to see which set of values results in the lowest RMSE for all clients. This can then be used in a machine learning way to optimize the locations. The parameters varied here has been the reference power at 1m and the pathloss coefficient.

The only parts left in the code is the dictionary created for sample client values (since the author accepts to not being able to work on real ale feeds currently)

clients = {

'1':[-43, -44, -30],

'2':[-31, -29, -30],

'3':[-50, -55, -52],

'4':[-61, -63, -59],

'5':[-41, -32, -31],

'6':[-31, -33, -32],

'7':[0, -32, 0],

'8':[-56, -56, 0],

'9':[-53, -56, -60]

}

The only other parts left are the plot commands, and the AP coordinates + random noise (not required but precautionary)

1. Conclusion:
2. What this code is: It is an alternative to get client locations from ale rssi feeds. This code was born out of a necessity since aruba’s ale doesn’t seems to be able to work in setups where the distance between APs are not large. It is a method to not only control the locations but to also improve them per our understanding of the sites.
3. What it is not: It is not a solution to work in all cases. We have seen the ale give rssi feeds from less than 3 APs (even when the client is in LOS of all APs) wherein the code needs atleast 3 feeds. The author recommends using dummy values to compensate for that. It is not a solution when more than 3 APs are to be used. In which case, theoretically the accuracy would be better. However, the code cannot scale to more than 3 APs.
4. The author has written the entire code in python himself. The script was tested on multiple plagiarism tools online (including Grammarly, plagiarisma) and has been shows as 100% unique.
5. References:
   1. WiFi Trilateration With Three or More Points: <https://appelsiini.net/2017/trilateration-with-n-points/>
   2. Trilateration: <https://en.wikipedia.org/wiki/Trilateration>
   3. FSPL: <https://en.wikipedia.org/wiki/Free-space_path_loss>
   4. Solving intersection point of circles: <http://paulbourke.net/geometry/circlesphere/>
   5. RMSE: <https://en.wikipedia.org/wiki/Root-mean-square_deviation>