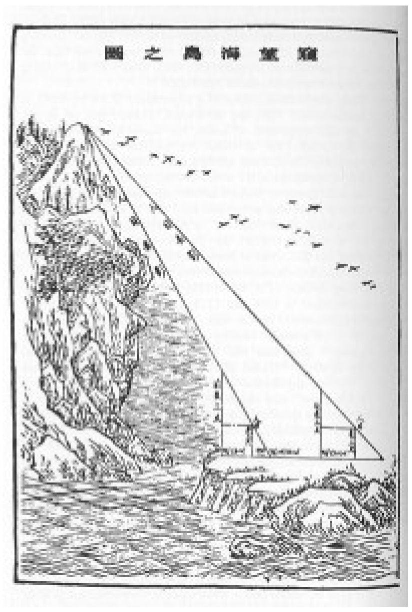
Ryan Miller's Blog

Software Developer in San Francisco Bay Area

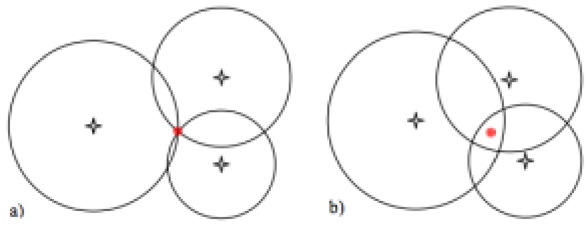
Wifi-based trilateration on Android



Triangulation offers a way to locate yourself in space. Cartographers in the 1600s originally used the technique to measure things like the height of the cliff, which would be too impractical to measure directly. Later, triangulation evolved into an early navigation system when Dutch mathematician Willebrord Snell discovered three points can be used to locate a point on a map.

While triangulation uses angles to locate points, trilateration uses lateral distances. If we know the positions of three points *P1*, *P2*, and *P3*, as well as our distance from each of the points, *r1*, *r2*, and *r3*; we can look at the overlapping circles formed to estimate where we are relative to the three points. We can even extend the technique to 3D, finding the intersecting region of spheres surrounding the points.

In this project, I'd like to show how we can use the Wifi signal strength, in dB, to approximate distance from a wireless access point (AP) or router. Once we have this distance, we can create a circle surrounding an AP to show possible locations we might occupy. In the next part of the project, I plan to show how we can use three APs to estimate our position in a plane using concepts of trilateration. (Note: I haven't had time to implement this, but you can use this Wiki article to implement it yourself).



Trilateration using 3 access points providing a very precise position (a) and a rougher estimate (b)

Determining distance from decibel level

There's a useful concept in physics that lets us mathematically relate the signal level in dB to a real-world distance. Free-space path loss (FSPL) characterizes how the wireless signal degrades over distance (following an inverse square law):

$$\mathrm{FSPL(dB)} = 20\log_{10}(d) + 20\log_{10}(f) + 92.45$$

The constant there, 92.45, varies depending on the units you're using for other measurements (right now it's using GHz for frequency and km for distance). For my application I used the recommended constant -27.55, which treats frequency in MHz and distance in meters (m). We can re-arrange the equation to solve for d, in Java:

```java
public double calculateDistance(double levelInDb, double freqInMHz)    {
   double exp = (27.55 - (20 * Math.log10(freqInMHz)) + Math.abs(levelInDb)) / 20.0;
   return Math.pow(10.0, exp);
}
```

Now, there are few drawbacks to this rough approximation:

1. FSPL explicitly requires "free space" for calculation, while most Wifi signals are obstructed by walls and other materials.
2. Ideally, we will want to sample the signal strength many times (10+) to account for varying interference.

Problem (1) will be resolved in the future by using the signal-to-noise ratio to more accurately estimate (that sounds like an oxymoron) obstructions to the wifi signal. Problem (2) can be implemented in code by sampling many times and computing the average signal level.

Using the above code along with Android's WifiManager and ScanResult classes, I can print out our final measurements:

```java
WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

registerReceiver(new BroadcastReceiver()
{
    @Override
    public void onReceive(Context c, Intent intent)
    {
        results = wifi.getScanResults();
        for (ScanResult s : results)    {
            DecimalFormat df = new DecimalFormat("#.##");
            Log.d(TAG, s.BSSID + ": " + s.level + ", d: " +
                    df.format(calculateDistance((double)s.level, s.frequency)) + "m");
        }
    }
}, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));

wifi.startScan();
```

And we can get back data that appears to be correct when moving further away from my test router (MAC address: 84:1b:5e:2c:76:f2):

[Image lost during host transition, but basically just showed how the distance increased]

Share this:

Print Email Facebook Google Digg Reddit Twitter

This entry was posted in project and tagged 802.11, Android, GPS, Java, Triangulation, Trilateration, WIfi, Wireless on May 30, 2013 [http://rvmiller.com/2013/05/part-1-wifi-based-trilateration-on-android/] .

23 thoughts on "Wifi-based trilateration on Android"

Guilherme
September 4, 2013 at 7:49 am

Hey Ryan,

That's a very interesting approach there. I'm working on a similar project myself. Would it be possible to contact you via e-mail so we can perhaps share and discuss findings and information?

Thanks,
Guilherme

dung hoang
September 23, 2013 at 11:57 pm

hi miller,
im interested in your project but there are some thing that i have not understood
+ the signal level (levelInDb) that we achieved by android app is FSPL ???
+ so if signal level is FSPL, i suppose that the formula should be like this :
double exp = (27.55 – (20 * Math.log10(freqInMHz)) + levelInDb) / 20.0
+ if you was successful when applying this method, can i know the position accuracy ?
hope to see your reply soon 😀

rvmiller Post author
September 24, 2013 at 8:22 am

It looks like you're correct, the formula should read " + levelInDb". I've corrected it above. Thanks!

jqjq
October 28, 2013 at 5:16 am

hi miller,
This article don't have part 2?

Antonis
November 12, 2013 at 6:09 am

Hi,
i'm basically new to android. My project at uni is about finding the exact coordinates of a mob device through trilateration. I have a question about the code snippet you gave us. Does it calculates all the distances needed(i mean from all the 3 hotspots) or is it just for one? Hope i get an answer as soon as you have time. Thx.
PS Please excuse any errors in my english.

rvmiller Post author
November 12, 2013 at 9:19 am

Hi Antonis,

The code samples only calculate the distance from one access point (AP). I haven't had a chance to implement the actual trilateration using multiple APs, but it should be straightforward to extend the code to do so.

Thanks.

tiger
November 21, 2013 at 11:18 am

hi, i have a question :
you use s.level in calculateDistance((double)s.level, s.frequency)

but s.level will returm to dBM (in developer.android.com)
and you use it for FSPL(dB) ?

---

tiger
November 21, 2013 at 7:29 pm

hi ,i see the link say different dBm vs dB:
http://www.isa.org/Content/ContentGroups/InTech2/Features/20023/November7/dB_vs__dBm.htm

or:
http://stackoverflow.com/questions/15797920/how-to-convert-wifi-signal-strength-from-quality-percent-to-rssi-dbm

---

Hao
November 24, 2013 at 10:15 pm

Dear Ryan,

It's very interesting and impressive!
When will you be able to give us some advice for calculating the position?
If three ACs lie in different Z ordinate, the theory still works.
But I had a hard time figuring out how to locate the point.
Appreciate that if you can give me any advice.

Thanks.
Hao

Khawar Raza
December 3, 2013 at 11:34 am

Hi, I was testing the code snippet you mentioned above. I am getting distances like 2.1883797959435243E-8 m which is highly inaccurate. Do you know where I am going wrong?

---

rvmiller Post author

December 3, 2013 at 5:30 pm

I think this is related to Tiger's point above — Android's ScanResult level is in dBm (mW), while the Wikipedia article uses just dB (decibels). Perhaps someone with more of a background in RF can comment on the conversion.

Rahul
March 11, 2014 at 11:13 am

Hi, Thanks for the information. I tried your code. But getting distance as 1.9527402174346173E-6. Can you please help to solve it. Thanks a lot.

---

rvmiller Post author
March 15, 2014 at 11:04 am

Hi Rahul,

Android's ScanResult level is actually giving dBm (mW), while the Wikipedia article uses just dB (decibels). For now, you can change the code to calculate FSPL to use Math.abs(levelInDb). I have changed the code sample above. I'm not sure if this is correct, however, but gives more reasonable distance measures 🙂

Throdne
April 16, 2014 at 8:08 pm

Hello, I love the post on this subject. I'm currently trying to figure it out myself. But I'm having a some problems.

So, I took your code and quickly converted it into python to see what results I would get before fully adding it to my c# program. But for some reason I'm getting a distant reading of 1.787094… m. I know for a fact that is not right. I know that my router is on the other side of the house. at least 60ish feet. I I would have bet that It would be a little bit more accurate then what I'm getting.

Am also getting my dBm and Freq from two different apps with the same signal dbm and freq results. but i'm currently using http://www.nirsoft.net/utils/wifi_information_view.html here is my python code.

```
import math


freqInMHz = 2457
levelInDb = -76


results = (27.55 - (20 * math.log10(freqInMHz)) + math.fabs(levelInDb)) / 20.0


print results
```

Throdne

April 16, 2014 at 8:52 pm

So, I figured out my issues. I didn't see that when you returned that function you also Math.pow(10, x) So, I corrected my code and it works like I think it should. But I'm really interested in finding a solution for problem one. (FSPL explicitly requires "free space" for calculation, while most Wifi signals are obstructed by walls and other materials.) I can see were I get my SNR and my RSSI. But I'm not to sure where to start on the formula on how to calculate for the difference.

lets say I have a db of -72 with a SNR of -87 and a freq of 2457. and I know that my router is about 50 to 60 feet away (15.25-18.25m). I want to see if I can't get accuracy down to at least a meter or 2. Any ideas on how to calculate for SNR?

Cristian

May 4, 2014 at 4:13 am

Hi,
Could you please share with us how do you calculate signal-to-noise ratio in Android?
Thanks.

Michael Henke

October 23, 2014 at 3:01 pm

Excellent blog entry, but where is part 2?

Raviteja
October 27, 2014 at 9:48 pm

Hi,

I'm just a beginner in android. when I tried this project in my own Wifi, it gives the approximate distance in high signal strengthl. when the signal is low, same distance is changed unexpectedly… may i get the proper solution for this issue soon.

Lg Opt
December 11, 2014 at 11:16 am

Im very disappointed in this post… cuz hes taught us something wrong.. and then disappeared!

dB… is not the same as dBm
to get distance we need FSPL whose answer will only be in dB

and what we get from android is dBm… so they arent the same and neither can be converted…

looking for some direction Mr.Miller?

Graham Worsfold
January 5, 2015 at 6:41 pm

Hi Ryan

did you complete this project and achieve accurate results ?

i understand the RSSI and SNR technique to resolve distance, Did you research the use of TDOA to improve accuracy ?

i am working on a project that needs trilateration solution over wifi so any help you can give would be appreciated

---

Raviteja
January 9, 2015 at 11:12 pm

Hi,

I’m obtaining the distances from all the wifi devices thye connected to my device. I don’t know how to implement Triangulation/Trilateration. plz suggest me how to implement it.....

vjay
June 25, 2015 at 1:38 pm

@Raviteja.

Did you get Triangulation/trilateration using 3 Wi-Fi access points. If yes please let me know.
I am tired of working on it. help me if you can.

Thanks in advance,
viju_1008@yahoo.com

vjay
June 25, 2015 at 1:42 pm

Hello there,

How to get the triangulation with Wi-Fi triangulation/trilateration?

links, info, source code, anything will be helpful
thanks in advance.
vijay