NIIP lab 6: Proximity and indoor location determination

In lots of countries the discussion about contact tracing apps is still ongoing. One of the technical challenges is to get an accurate estimation of the location of a person within a certain area (indoors). Obviously there are methods available that use computer vision or real-time tracking - using dedicated technologies/hardware. However, in this lab we want to try and use only the infrastructure that is already available.

A popular and currently used approach for indoor usage scenarios is to turn your smartphone into a “BLE beacon”. However, as not all students have such hardware at their disposal, we will substitute this by WiFi. You can find lots of references in literature on how to use this technology for location determination.

Intuitively, one could envisage a solution that uses straightforward maths (triangulation) to determine the location of a device based on received signal strength indication or RSSI. In this lab, we will take a similar approach to tackle the issue.

Even at home, your laptop will probably ‘see’ lots of different networks. For some of these, you can easily locate the AP, for others not. In this lab, you may assume that you know the location of at least two access points : the one providing your internet connection (e.g. your Teleneter router/Proximus BBOX) in your home and secondly your smartphone (which you can configure as an AP using settings or apps). Ideally, you would find a third access point that you can locate and ‘see’ from your laptop. However, if this is not available, location determination should still function to some degree.

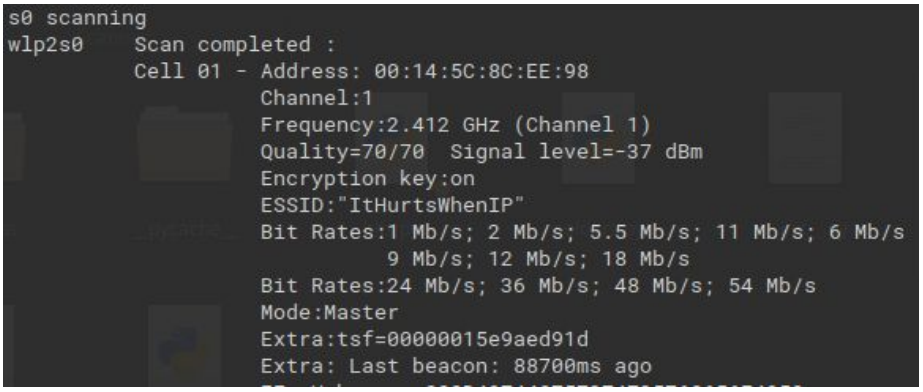
Each AP broadcasts beacons at a certain interval (depending on configuration). You need to implement an application on your laptop that captures these beacons and extracts relevant information (mainly identity of the AP and signal level). Normally, this information can be collected using wireshark/tcpdump with appropriate parameters, but there is a lot of variation between wifi chipsets. Some expose the information easily, others don’t. You may have to experiment with different operating systems to see what information you can obtain.

Once you have access to the relevant information, combine it to figure out where your laptop is located. This works best if the three APs are located in different areas (e.g. rooms or building) surrounding your location. You will probably notice quite quickly that the raw data will require various types of noise filtering (high/low pass, kalman, …). It is up to you to decide what is feasible and how you integrate this in your application. You may of course use libraries for processing the data. Tip : you can also have a look at existing literature (publications etc) for ideas on how to improve the system.

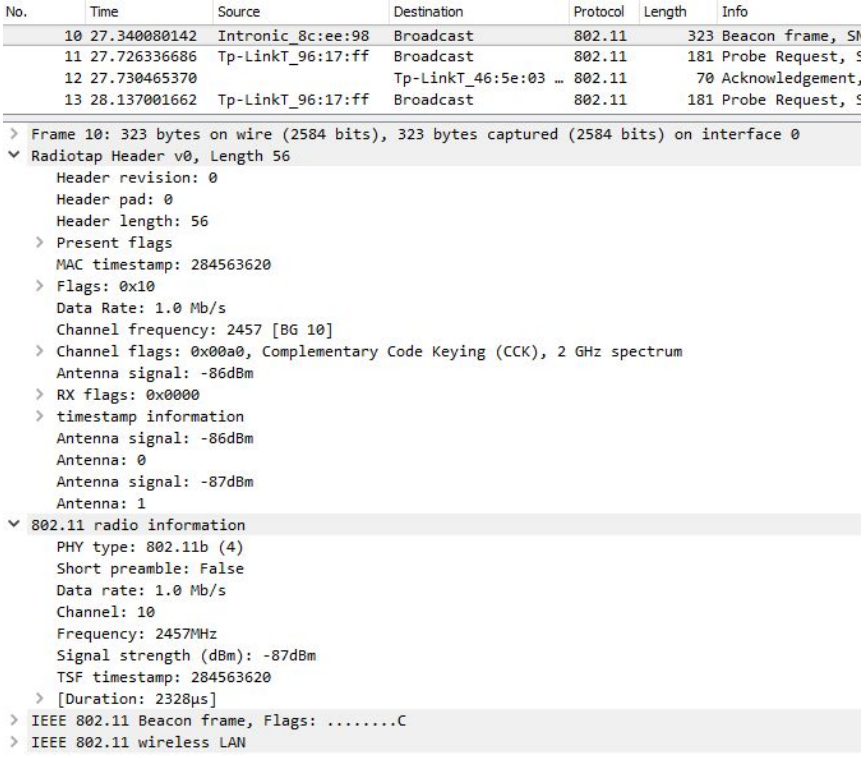
At the end of the assignment, we require a brief presentation of your results. This will be planned on Thursday, May 28. Handing in your assignment is also planned for that morning.

Mogelijkheden

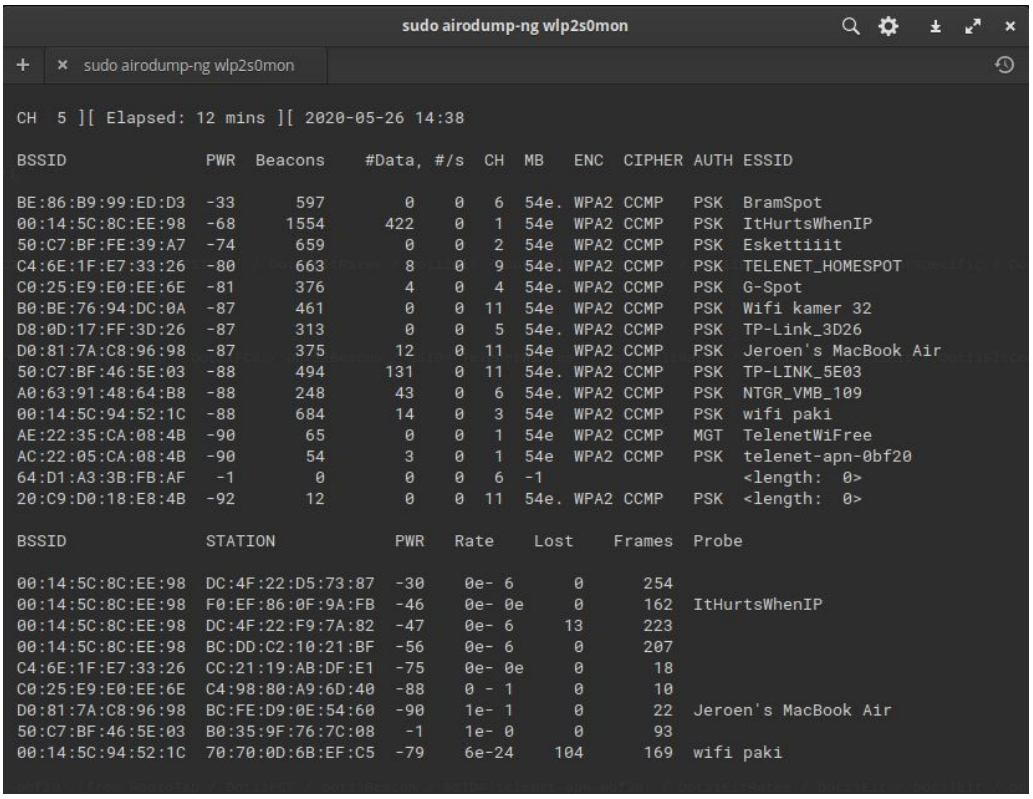
1. Met Python module `access_points` de signaalinfo uit netwerkkaart opvragen. Deze module is een wrapper rond:
   a. `netsh wlan show networks mode=bssid`
      (Windows, geeft signal strength percentage)
   b. `nmcli -t -f ssid,bssid,signal,security device wifi list`
      (Unix, geeft signal strength percentage)
   c. `iwlist <interface> scanning`
      (Unix, geeft dBm waarde; zelfde als in Beacon frames)



2. AP info opvragen door beacon frames rechtstreeks te analyseren (Wireshark)
   a. Netwerkkaart in monitoring mode en 802.11 Beacon Frames filteren
   b. Voor unix:
      i. Starten: `airmon-ng start <interface>` (e.g. `wlp2s0`)
      ii. Stoppen: `airmon-ng stop <interface> && service network-manager start`
      iii. Nu kunnen we via Wireshark Beacon frames ontvangen, vb (met Kali distro van laptop Bram):



iv. Om te forceren dat er ook daadwerkelijk gescand werd naar APs moet het commando `airodump-ng wlp2s0mon` uitgevoerd worden, waaruit we volgende output krijgen:



c. Voor Windows?
   i. Npcap installatie (meegeleverd met Wireshark) en "raw 802.11 traffic" aanvinken (info)
   ii. Wlan mode opvragen geeft error:

```
C:\Windows\System32\Npcap>wlanhelper "Wi-Fi" mode
Error: makeOIDRequest::My_PacketOpenAdapter error (to use this function, you
need to check the "Support raw 802.11 traffic" option when installing Npcap)
Failure
```

Maar mode wijzigen met `wlanhelper "Wi-Fi" mode monitor` geeft `Success`?
In Wireshark, met monitor mode aangevinkt, komt er geen verkeer door op de interface. Waarschijnlijk omdat het onderliggend toch niet supported is. Hetzelfde resultaat verkrijgen we met de ingebouwde wlan adapter (Intel(R) Dual Band Wireless-N 7260) en een externe dongle (Ralink 802.11n USB Wireless LAN Card) (met win10 v1803 op laptop William).

3. AP info opvragen door beacon frames rechtstreeks te analyseren via Python?
   a. Unix: pakketten uit Beacon frames analyseren (na monitor mode activeren)
      i. Vb met Scapy

b. Windows?
    i. Wlan api?
    ii. https://github.com/changyuheng/winwifi.py
    iii. https://github.com/kedos/win32wifi

4. Output
    a. "Kaart" opstellen van AP locaties en normalisatie van dBm/% waarden van signalen naar afstanden om om kaart te plotten.
        i. Berekening voor trilateration doen door dBm/% om te zetten naar distance estimation.
    b. Fingerprinting 1: signaalsterktes gebruiken, zonder locatie van AP, en deze registreren als "coördinaten". Later kijken bij welk coördinaten een device het dichts bij ligt. Vb:
        i. Device staat in midden van kamer, `register location` commando uitvoeren en huidige signaalsterktes opslaan onder een naam (vb woonkamer).
        ii. Dit herhalen voor elke kamer.
        iii. Bij opvragen van `estimate location`, de locatie met kortste afstand tot huidige fingerprint teruggeven. (indien APs niet meer in bereik zijn, maar andere wel, ofwel onbereikbare uit het te vergelijken coördinaat halen en dan pas afstand berekenen, ofwel signaal sterkte heel laag zetten voor dat AP).
    c. Fingerprinting 2: signaalsterktes gebruiken, zonder locatie van AP, en deze registreren als "coördinaten" van een rechthoek. Later kijken in welke rechthoek een device nu ligt (of waar het kortste bij).
        i. Device `register location` laten uitvoeren in de 4 hoekpunten van de kamer.
        ii. Bij opvragen van `estimate location`, kijken binnen welke rechthoek de huidige set van signaalsterktes voor elk AP valt.
        OF
        3 punten per kamer opslaan en in die driehoek het midden nemen en dit gebruiken als punt om afstand tot die locatie te berekenen.
    d. Fingerprinting door een neural net locaties aan te leren en daaruit een likelihood te bepalen voor elke locatie en de hoogste terug te geven.
        i. Gelijkaardig aan: https://github.com/schollz/find
        (Python versie: https://github.com/kootenpv/whereami)

Uitwerking

Verkrijgen van access point info

In de basissituatie, gebruiken we, zoals eerder vermeld, de `access_points` module als basis om van de netwerkkaart op te vragen welke APs beschikbaar zijn en welke signaalsterktes ze hebben. Dit houden we bij in een dictionary en een aparte thread update dit elke seconde. We pasten de module aan zodat de signaalsterkte altijd een RSSI waarde voorstelt, en updaten deze waarde met een running average.

Hieronder het basisprogramma. Op de achtergrond zal er telkens `sample` opgeroepen worden, maar dit kan ook manueel gedaan worden. Met `print` zijn de gevonden APs weer te geven en met `position` worden de 3 kortste (signaalsterkte het hoogst) gebruikt om trilateration op uit te voeren. In het voorbeeld hieronder, worden APs met "5G" in de naam genegeerd. Deze signalen variëren sterker dan de 2.4GHz netwerken en laten we hier dan weg.

```
Enter command: sample
[Scanner] Updated info for 5 access points.
Enter command: print
Scanner Access Points:
Vibenet             (4c:ed:fb:a6:00:b8): -89.5dBm
Vibenet_5G          (4c:ed:fb:a6:00:bc): -74.5dBm
Vibenet             (60:45:cb:59:f0:51): -67.5dBm
Synchrotron         (96:5e:e3:af:4d:81): -58.5dBm
Vibenet_5G          (60:45:cb:59:f0:54): -56.0dBm
Enter command: position
Estimated AP distances:
Synchrotron         (96:5e:e3:af:4d:81) @ (+5.00, -2.00): 2.05m
Vibenet             (60:45:cb:59:f0:51) @ (+0.00, +0.00): 3.10m
Vibenet             (4c:ed:fb:a6:00:b8) @ (+1.00, -5.00): 8.53m
Updated device position: (+5.36, +4.79)
```

Merk verder op dat de dBm waarden hierboven verkregen zijn door het signaalsterktepercentage uit Windows om te rekenen met volgende formule:

```
dbm = (quality_percentage / 2) - 100
```

Dit zijn dus niet de werkelijke waarden. Op Linux, zou dat wel het geval zijn indien de output van `iwlist` gebruikt wordt. Om dit te doen kan men als argument `-m 1` of `--mode 1` meegeven om het gebruik van `iwlist` te forceren (enkel op Linux based systems).

In een meer geavanceerde modus, gaan we, tevens op Linux, de Beacon frames zelf onderscheppen om de RSSI waarde en de APs te weten te komen. Dit kan men starten door de scanner te starten met `-m 2`. Hierbij gaat en aparte thread met behulp van Scapy pakketten sniffen die een `Dot11` gerelateerde laag hebben. Dit zijn pakketten met Probe requests, Probe responses, Beacon frames of RadioTap lagen. Als het ontvangen pakket een probe of beacon frame bevat, zullen we de `RadioTap` laag gebruiken om het `dBm_AntSignal` attribuut, oftewel de signaalsterkte van de Wi-Fi antenne in dBm, uit te lezen. Dit blijkt enkel mogelijk met Scapy 2.4 of hoger. In vorige versies, werd het ontleden van deze pakketten nog niet ondersteund.

De Scanner zal één van deze drie methoden gebruiken om APs te zoeken en hun RSSI waarde op te vragen. Dit gebeurt elke seconde op de achtergrond of manueel met het sample command zoals eerder vermeld.

Schatten van afstand tot AP

Non-monitoring mode

Nu we (een schatting van) de signaalsterktes weten met de commando's uit het OS, kunnen we deze gebruiken om een schatting te doen van de afstand tussen het device en het AP. Om dit te doen hebben we twee methodes geïmplementeerd:

```python
@staticmethod
def _calc_signal_distance_simple(signal_strength_dbm, signal_frequency=2.4):
    exp = (27.55 - (20 * math.log10(signal_frequency * 1000)) + abs(signal_strength_dbm)) / 20.0
    return round(10 ** exp, 4)

@staticmethod
def _calc_signal_distance(signal_strength_dbm, signal_attentuation=3, signal_strength_ref=-50, signal_dist_ref=4):
    beta_numerator   = float(signal_strength_ref - signal_strength_dbm)
    beta_denominator = float(10 * signal_attentuation)
    beta = beta_numerator / beta_denominator
    return round((10**beta) * signal_dist_ref, 4)
```

In de eerste methode wordt er geen rekening gehouden met de attentuation en wordt er een meer generieke berekening toegepast om de verzwakking over afstand in te schatten. In de tweede methode worden deze parameters wel in acht genomen, wat meestal tot een meer accuraat resultaat zal leiden. We gebruiken dan ook standaard de tweede methode om de afstand tot een AP te bepalen. Dit vereist echter dat de verzwakking over afstand gekend is, wat niet altijd het geval is. Daarom hebben we de eerste methode eveneens laten staan.

De tweede methode neemt ook twee kalibratieparameters mee, een signaalsterkte referentie en de afstand die een AP had op die sterkte. Dit kan men gebruiken om de schatting nog nauwkeuriger te maken. Voor nu hebben we deze waarde vast laten staan in de code, maar indien AP beter gemapt worden, zou men deze parameters samen met de positie van een AP kunnen opslaan. Zo kan het ook opgezocht worden samen met het coördinaat en kunnen we altijd een juiste waarde bekomen voor de afstand tot het AP.

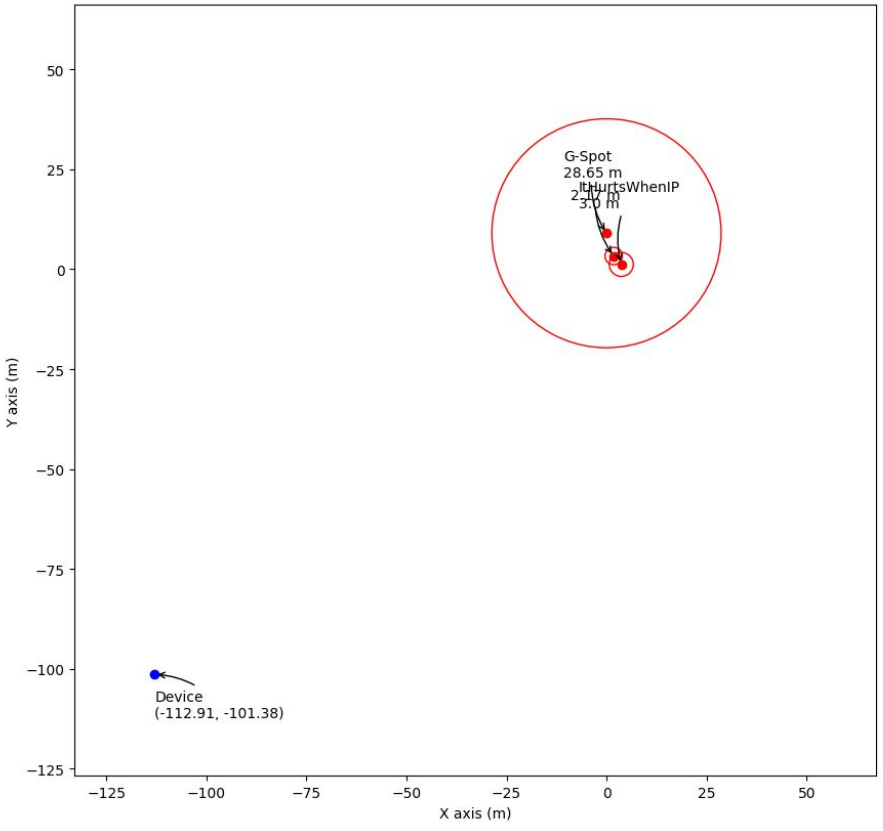
Monitoring mode

Zoals voorheen beschreven hebben we tevens geprobeerd om de afstand tot een AP te bepalen door onze netwerkadapter in monitoring mode te zetten. Dit laat ons toe om te scannen op beacon frames en probe requests (en responses). Hieruit kunnen we vervolgens de signaalsterkte afleiden die we kunnen gebruiken om de afstand te schatten zoals voorheen. We merkte echter dat deze signaalsterkte sterk fluctueert, wat er ons toe heeft aangezet een gemiddelde te nemen over de verstreken tijd met een running average en ook met en weighted average.

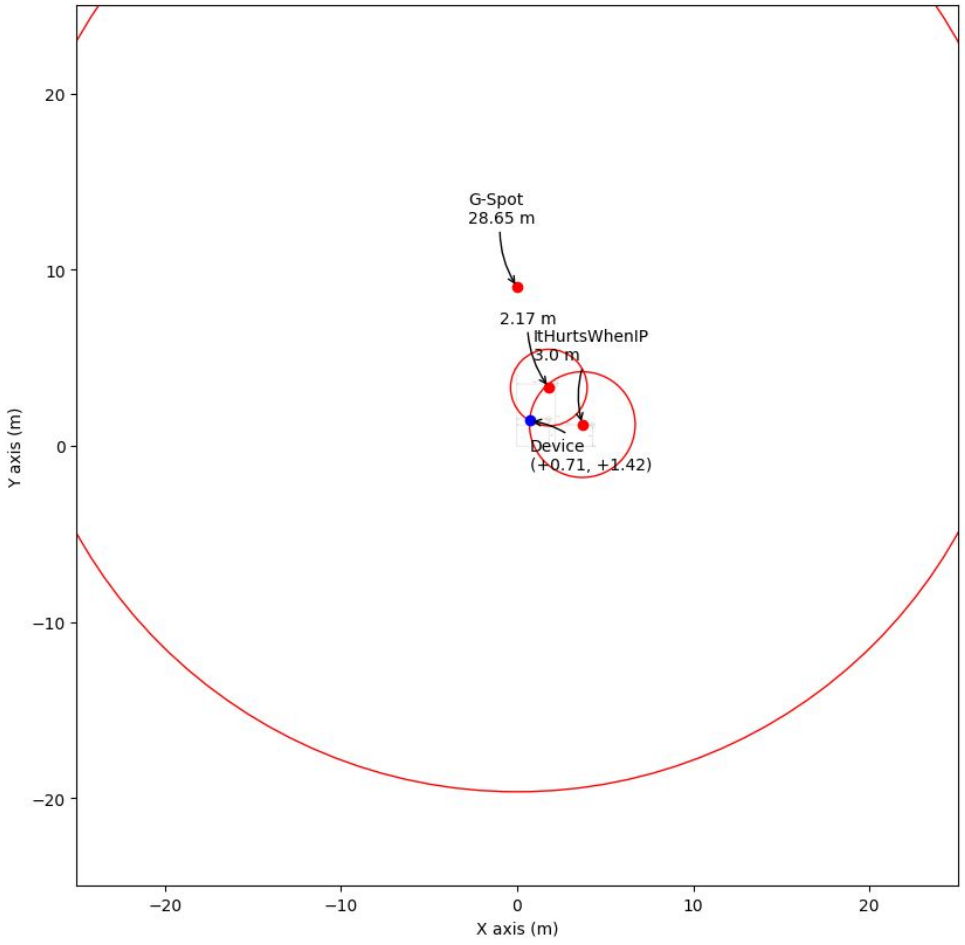
Om een schatting te krijgen van de verschillende APs en hun signaalsterkte voeren we het commando `airdump-ng wlp2s0` uit en krijgen onderstaand resultaat. Dit commando forceert de interface om nieuwe frames te ontvangen.

```
sudo airodump-ng wlp2s0mon
 sudo airodump-ng wlp2s0mon

CH  5 ][ Elapsed: 12 mins ][ 2020-05-26 14:38

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

BE:86:B9:99:ED:D3  -33      597        0    0   6  54e. WPA2 CCMP   PSK  BramSpot
00:14:5C:8C:EE:98  -68     1554      422    0   1  54e  WPA2 CCMP   PSK  ItHurtsWhenIP
50:C7:BF:FE:39:A7  -74      659        0    0   2  54e  WPA2 CCMP   PSK  Eskettiiit
C4:6E:1F:E7:33:26  -80      663        8    0   9  54e. WPA2 CCMP   PSK  TELENET_HOMESPOT
C0:25:E9:E0:EE:6E  -81      376        4    0   4  54e. WPA2 CCMP   PSK  G-Spot
B0:BE:76:94:DC:0A  -87      461        0    0  11  54e  WPA2 CCMP   PSK  Wifi kamer 32
D8:0D:17:FF:3D:26  -87      313        0    0   5  54e. WPA2 CCMP   PSK  TP-Link_3D26
D0:81:7A:C8:96:98  -87      375       12    0  11  54e  WPA2 CCMP   PSK  Jeroen's MacBook Air
50:C7:BF:46:5E:03  -88      494      131    0  11  54e. WPA2 CCMP   PSK  TP-LINK_5E03
A0:63:91:48:64:B8  -88      248       43    0   6  54e. WPA2 CCMP   PSK  NTGR_VMB_109
00:14:5C:94:52:1C  -88      684       14    0   3  54e  WPA2 CCMP   PSK  wifi paki
AE:22:35:CA:08:4B  -90       65        0    0   1  54e  WPA2 CCMP   MGT  TelenetWiFree
AC:22:05:CA:08:4B  -90       54        3    0   1  54e  WPA2 CCMP   PSK  telenet-apn-0bf20
64:D1:A3:3B:FB:AF   -1        0        0    0   6  -1                    <length:  0>
20:C9:D0:18:E8:4B  -92       12        0    0  11  54e. WPA2 CCMP   PSK  <length:  0>

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:14:5C:8C:EE:98  DC:4F:22:D5:73:87  -30    0e- 6      0       254
00:14:5C:8C:EE:98  F0:EF:86:0F:9A:FB  -46    0e- 0e     0       162  ItHurtsWhenIP
00:14:5C:8C:EE:98  DC:4F:22:F9:7A:82  -47    0e- 6     13       223
00:14:5C:8C:EE:98  BC:DD:C2:10:21:BF  -56    0e- 6      0       207
C4:6E:1F:E7:33:26  CC:21:19:AB:DF:E1  -75    0e- 0e     0        18
C0:25:E9:E0:EE:6E  C4:98:80:A9:6D:40  -88    0 - 1      0        10
D0:81:7A:C8:96:98  BC:FE:D9:0E:54:60  -90    1e- 1      0        22  Jeroen's MacBook Air
50:C7:BF:46:5E:03  B0:35:9F:76:7C:08   -1    1e- 0      0        93
00:14:5C:94:52:1C  70:70:0D:6B:EF:C5  -79    6e-24    104       169  wifi paki
```

Wanneer we bovenstaande waarden gebruikten om de positie te bepalen, door de RSSI waarde uit de ontvangen pakketten uit te lezen, kregen we echter een resultaat dat totaal niet representatief was met de realiteit:



Wanneer we ons geoptimaliseerd algoritme `TRILAT_ESTIM` (zie verder) toepassen, krijgen we een beter resultaat:



Dit is echter een zeer logisch gevolg, daar dit algoritme zich baseert op de snijpunten van de gevonden cirkels.

Het ontvangen van die frames ziet er uit zoals op onderstaande afbeeldingen. Ontvangen beacon frames of probe requests/responses komen binnen en de antenna signal strength zal uit het pakket gehaald worden.Hieronder zien we nieuwe APs opgeslagen worden met BSSID (MAC-adres), SSID en uit welk pakket het kwam. De Scanner kan later de laatste RSSI waarde voor elk AP opvragen.

```
bram@bram-HP-EliteBook-850-G3:~/Documents/School/2 MA/NIIP/NIIP-Labo/Lab6/Scanner$ sudo python3 scanner.py -m 2
[Sniffer] Start sniffing...
[Scanner] Start sampling thread (1s delay)...
[Sniffer] New Access Point @ 00:14:5C:8C:EE:98 'ItHurtsWhenIP' (from RadioTap / Dot11FCS / Dot11Beacon / SSID='It
fic / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ C4:6E:1F:E7:33:26 'TELENET_HOMESPOT' (from RadioTap / Dot11FCS / Dot11Beacon / SSID=
rSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ C0:25:E9:E0:EE:6E 'G-Spot' (from RadioTap / Dot11FCS / Dot11Beacon / SSID='G-Spot' /
 / Dot11EltVendorSpecific / Dot11EltVendorSpecific)
Command list: print, sample, position, positionest, positionall, plot, clear, exit
Enter command: [Sniffer] New Access Point @ D8:0D:17:FF:3D:26 'TP-Link_3D26' (from RadioTap / Dot11FCS / Dot11Bea
lt / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ B0:BE:76:94:DC:0A 'Wifi kamer 32' (from RadioTap / Dot11FCS / Dot11Beacon / SSID='Wi
Elt / Dot11Elt / Dot11EltVendorSpecific / Dot11Elt / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ 50:C7:BF:46:5E:03 'TP-LINK_5E03' (from RadioTap / Dot11FCS / Dot11Beacon / SSID='TP-
rSpecific / Dot11EltVendorSpecific / Dot11EltRSN / Dot11EltMicrosoftWPA / Dot11Elt / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ D0:81:7A:C8:96:98 'Jeroen's MacBook Air' (from RadioTap / Dot11FCS / Dot11Beacon / S
Elt / Dot11EltVendorSpecific / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ BC:FE:D9:0E:54:60 'Jeroen's MacBook Air' (from RadioTap / Dot11FCS / Dot11ProbeReq /
11EltVendorSpecific / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ A0:63:91:48:64:B8 '!' (from RadioTap / Dot11FCS / Dot11Beacon / Dot11Elt / SSID='NT
/ Dot11Elt / Dot11Elt / Dot11Elt / Dot11Elt / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpe
sample[Sniffer] New Access Point @ 50:C7:BF:FE:39:A7 'Eskettiiit ' (from RadioTap / Dot11FCS / Dot11Beacon / SSI
11Elt / Dot11Elt / Dot11EltVendorSpecific / Dot11Elt / Dot11EltVendorSpecific)
[Sniffer] New Access Point @ 00:14:5C:94:52:1C 'wifi paki' (from RadioTap / Dot11FCS / Dot11Beacon / SSID='wifi p
1EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpecific / Dot11EltVendorSpe
```

Bepalen van posities

De implementatie biedt 3 mogelijkheden aan als het neerkomt op het bepalen van de locatie van een device. Deze gebruiken de gevonden APs, zoeken afhankelijk van het MAC-adres hun locatie op in een look-up table (hardcoded in een `Locations` klasse), en bereken de radius van de cirkel met die locatie als oorsprong, gegeven de RSSI waarde uit de vorige stap. Hierna wordt deze lijst gesorteerd, zodat de APs die zogenaamd het kortste bij liggen eerst kunnen worden beschouwd.

- TRILATERATION
  Neemt 3 APs en berekent een snijpunt van de verzameling volgens de formule die te vinden is op 101computing.net
  - Demo
  - In scanner beschikbaar met het `position` commando.

- TRILAT_ESTIM
  Neemt het resultaat uit het bovenstaande algoritme maar gaat bijkomend de snijpunten van de 'cirkels' rond elk van de 3 APs berekenen. Hieruit ontstaat een verzameling snijpunten, waarbij het snijpunt dat het dichtst bij het bovenstaande resultaat ligt (snijpunt van alle 3 cirkels, met `position`), als locatie wordt beschouwd.
  - In scanner beschikbaar met het `positionest` commando.

- TRILAT_ESTIM_ALL
  Geeft een verzameling terug van alle resultaten die in bovenstaande algoritmes bekomen werden.
  - In scanner beschikbaar met het `positionall` commando.

De bovenstaande algoritmes steunen allemaal op hetzelfde algoritme om de afstand tot een AP te bepalen (zie eerder bij Schatten van afstand voor AP). Om een accurate positie te kunnen bepalen, hebben we een vaste lijst gecreëerd met daarin het MAC-adres (en SSID in commentaar voor referentie) van de verschillende APs die we wensen in acht te nemen, met een bijhorende locatie die gebruikt kan worden voor deze berekeningen en later in een plot. Dit hebben we specifiek gedaan, omdat we van deze APs de locatie kennen, wat ons de mogelijkheid geeft een meer accurate positiebepaling te doen van onze devices. Bijgevolg kunnen we dus APs waarvan we de positie niet kennen, ook niet beschouwen in deze berekeningen.

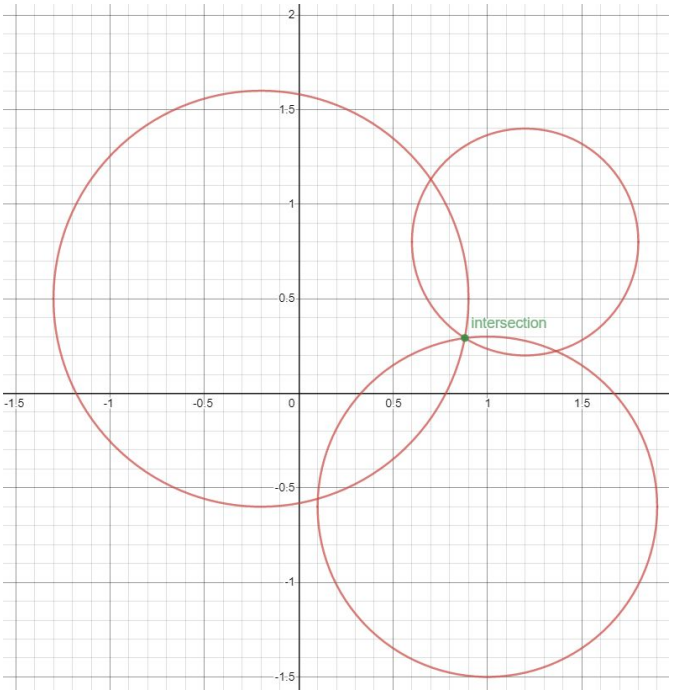
Deze algoritmes gaat er van uit dat de positie en afstand geweten is van 3 APs. In het geval van 2 APs wordt er een fallback procedure gebruikt om tevens een schatting te krijgen van de locatie door de snijpunten van de 2 cirkels te bepalen, als die cirkels elkaar snijden, hoewel deze niet zo accuraat zal kunnen zijn. Als ze niet snijden, zouden we het midpoint tussen de 2 centers kunnen nemen, maar het zou accurater zijn om een lijn loodrecht op de lijn tussen de centers, op de locatie van het midpoint te nemen. We lieten dit in commentaar staan.

De geschatte afstanden tot elk AP en de mogelijke positie(s) van het huidige toestel worden geprint in de console. Deze kunnen met het `plot` commando ook worden weergegeven.

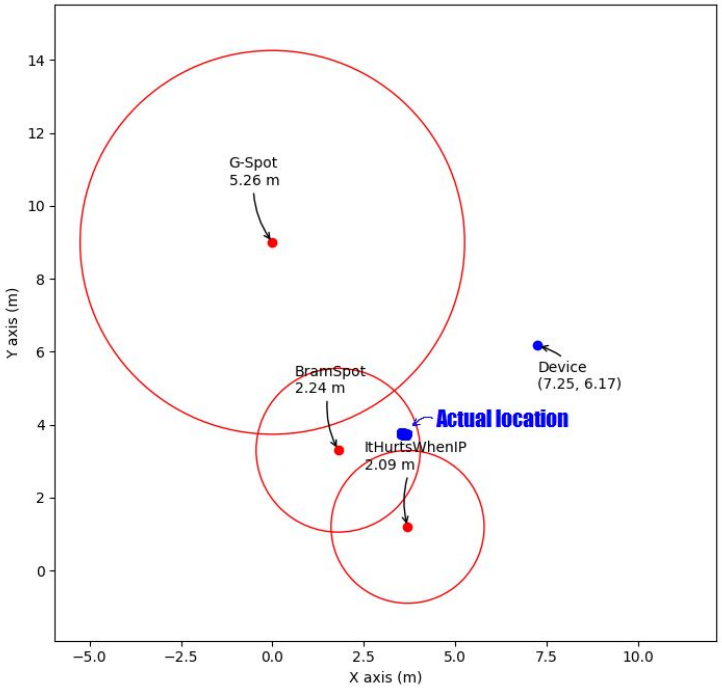
Hieronder gaan we in meer detail wat deze methoden nu eigenlijk geven als output.

TRILATERATION

In het geval van 3 APs gebruiken we de formule die we gevonden hebben op de webpagina van 101computing.net. Deze formule neemt de functies van de cirkels en bepaalt aan de hand daarvan een punt waarin de drie cirkels snijden. In de deze demo die we voor onszelf maakten, kunnen we experimenteren met de locatie en radius van de cirkels en zien we waar het "snijpunt" ligt. Zoals in onderstaande afbeelding te zien is, geeft dit een mooi resultaat als er ook daadwerkelijk een snijpunt voorkomt voor alle drie de cirkels:



Tijdens het experimenteren zijn we er echter achter gekomen dat deze situatie zich zelden voordoet. APs die te ver van elkaar liggen, in elkaar liggen, elkaar niet snijden… zorgden er telkens voor dat er geen duidelijk gemeenschappelijk punt was voor de drie cirkels. De formule echter, gaf wel een resultaat terug, wat vaak neerkwam op een ruwe schatting (zie onderstaande afbeelding). Dit heeft ons ertoe aangezet om de `TRILAT_ESTIM` functie te implementeren (zie verder).

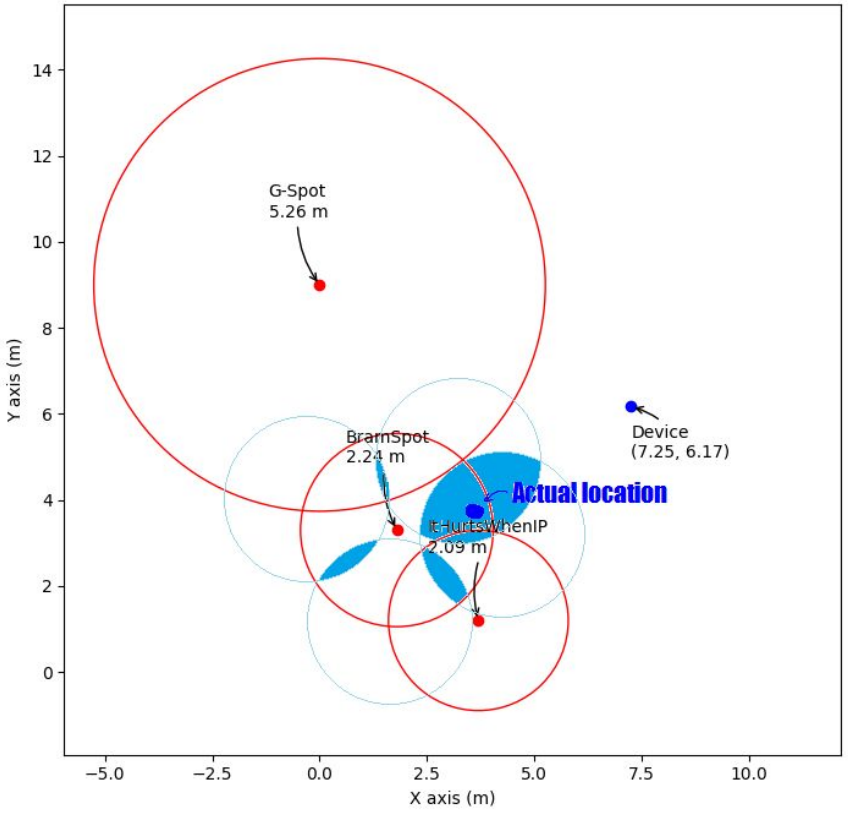


In de "echte" situatie hierboven zien we dat de geschatte positie van het Device tamelijk ver uit de buurt van de drie cirkels wordt geduwd. De formule kan er niet zo goed mee overweg dat de drie cirkels elkaar niet allemaal snijden.

Wanneer er slechts 2 APs gevonden zijn (doordat ze niet allemaal hardcoded geregistreerd staan), kunnen we de bovenstaande formule natuurlijk niet toepassen. Daardoor hebben we er voor gekozen om simpelweg de snijpunten van de twee cirkels te bepalen en deze terug te geven als eventuele locatie.

TRILAT_ESTIM

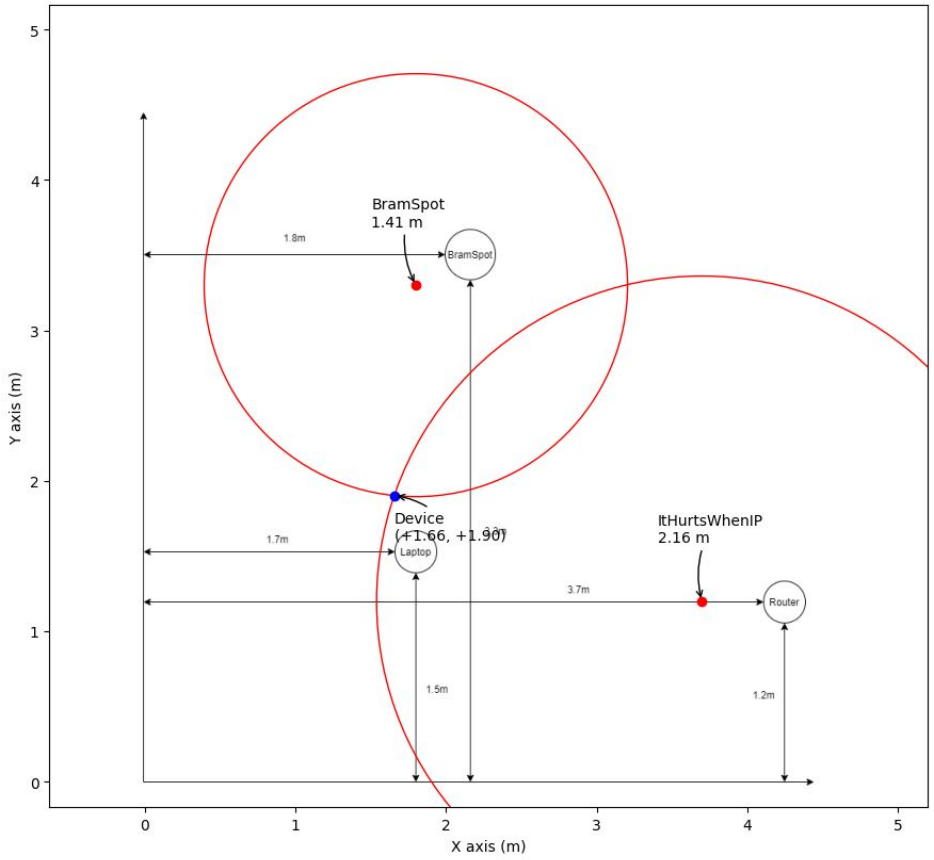
Bij het uitvoeren van vorig algoritme waren we niet tevreden over het bekomen resultaat. We zijn dan dieper in de plot gaan kijken en zagen we dat de eigenlijke locatie zich in de buurt van één van de snijpunten van de cirkels ligt. We hebben dit ter verduidelijking aangeduid op onderstaande afbeelding:



We tekenden snel even de lichtblauwe cirkels op de snijpunten van de rode en markeerden de gemeenschappelijke oppervlakken. Hieruit maakten we op dat het misschien een beter resultaat zou opleveren als we de verschillende snijpunten van de cirkels vergelijken met het resultaat van voorgaand algoritme. Dit brengt ons bij het `TRILAT_ESTIM` algoritme.

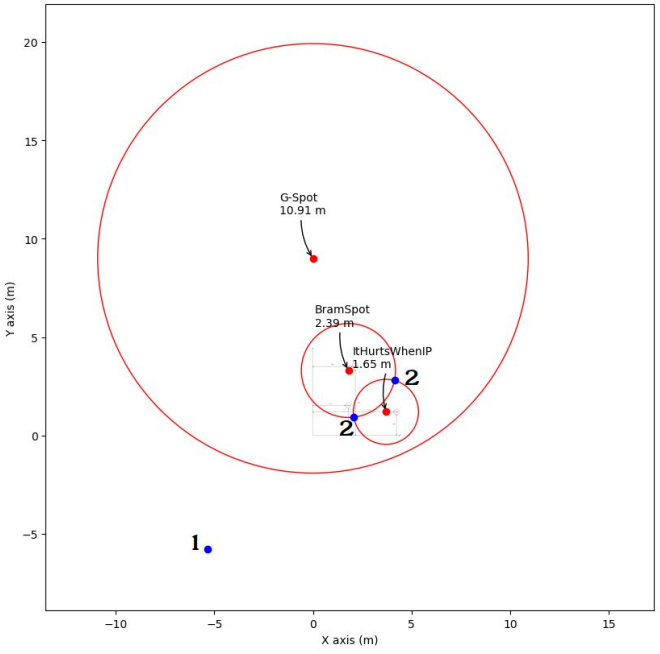
Dit algoritme begint met het resultaat te nemen dat bekomen werd uit de `TRILATERATION` berekening dat hierboven beschreven staat. Vervolgens zullen de drie cirkels rond de APs 1 per 1 met elkaar vergeleken worden en de verschillende snijpunten tussen deze cirkels bepaald worden. Deze verzameling snijpunten vergelijken we met het resultaat uit voorgaand algoritme en we nemen het snijpunt dat het dichtst bij dit resultaat ligt als locatie van het device, ervan uitgaande dat het device zich toch ergens op een rode cirkel zou bevinden. Op onderstaande afbeelding staat een plattegrond afgebeeld van de eigenlijke

situatie (bij Bram), met daarbovenop de geschatte locatie van het device. We zien dat onze schatting heel dicht in de buurt van de daadwerkelijke situatie komt:



TRILAT_ESTIM_ALL

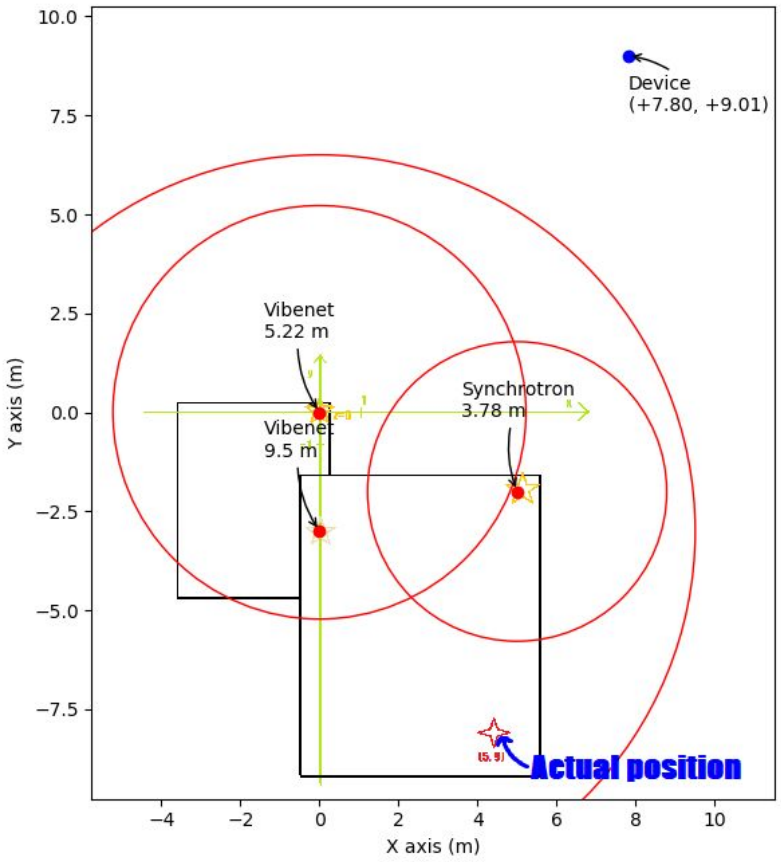
Dit algoritme bundelt alle resultaten die we bekomen uit voorgaande algoritmes en geeft deze weer op het uiteindelijke resultaat. Op onderstaande afbeelding staat aangeduid welk punt we uit het `TRILATERATION` algoritme bekomen hebben (1) en welke punten we uit het `TRILAT_ESTIM` algoritme zijn bekomen (2):



Plot van locatie en radius

Nu we elk aspect van de implementatie besproken hebben, overlopen we nog enkele resultaten. Merk op dat we in de code APs die "5G" in de naam hebben, en dus op 5GHz zitten, wegfilteren uit de resultaten. De verzwakking van een 2.4GHz en een 5GHz signaal is namelijk anders, dus zouden er andere parameters moeten gebruikt worden om de afstand uit te rekenen, die we nu niet opslaan per AP. Ook APs die te ver naar boven (verdieping) of onder liggen, halen we eruit.

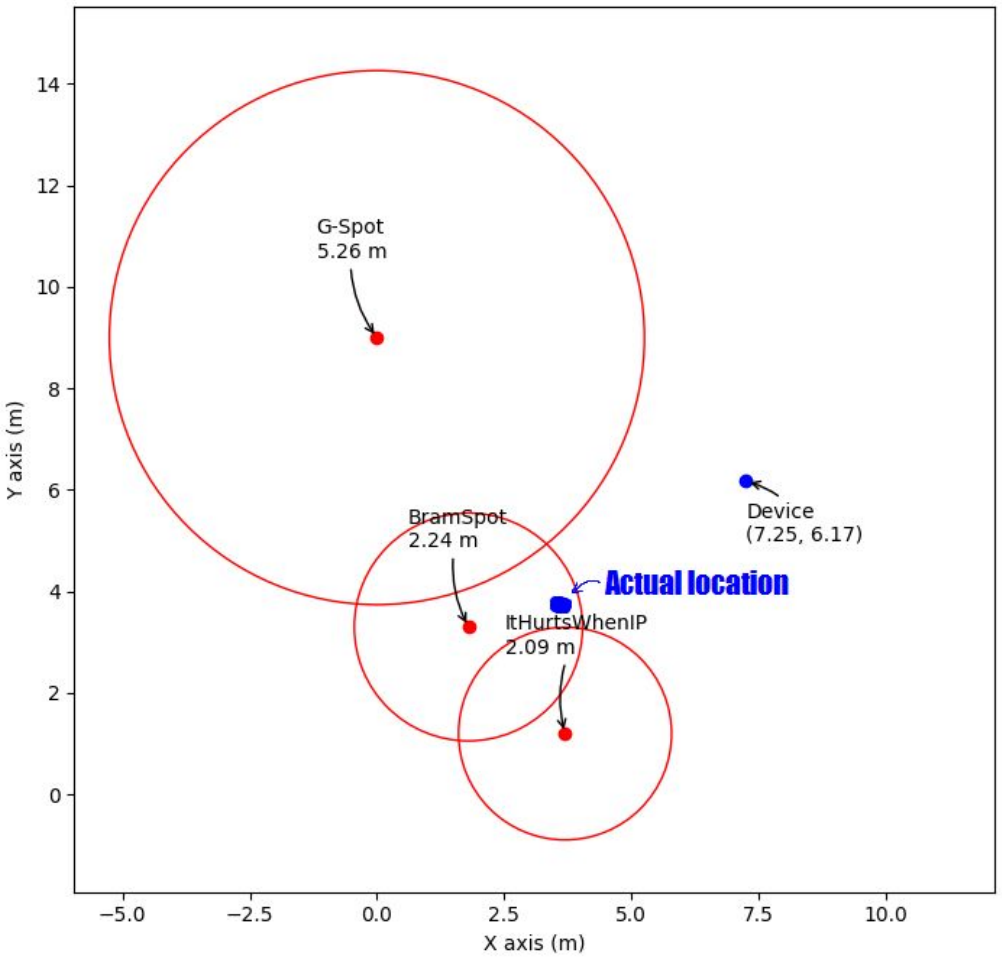
Hieronder een test met signal strength percentages uit Windows (bij William):



In deze situatie is het Vibenet AP op (0, 0) een router op gelijkvloers, terwijl de 2de daaronder op de eerste verdieping staat. Synchrotron is het AP door een gsn uitgezonden, ook op gelijkvloers. Het device, een Windows laptop, bevindt zich werkelijk op (5, -9).

Met de normale `TRILATERATION` zien we dat het geschatte punt terug uit de weg wordt geduwd naar (7.8, 9.0), terwijl de eigenlijke locatie op ongeveer (5, -9) zat. Indien we hier `TRILAT_ESTIM` zouden toepassen, zouden de 2 snijpunten van de binnenste cirkels gebruikt worden, wat toch een beter resultaat is. Ongeacht het feit dat het Vibenet AP korter bij staat, lijkt dit verder weg te zijn (op 9.5 meter). Dit ligt wellicht aan diens locatie. Doordat het signaal door het plafond heen moet, wordt het afgezwakt en daarom is de sterkte ook minder en de afstand dus groter.

Hier dan een test met RSSI waarden uit Linux.

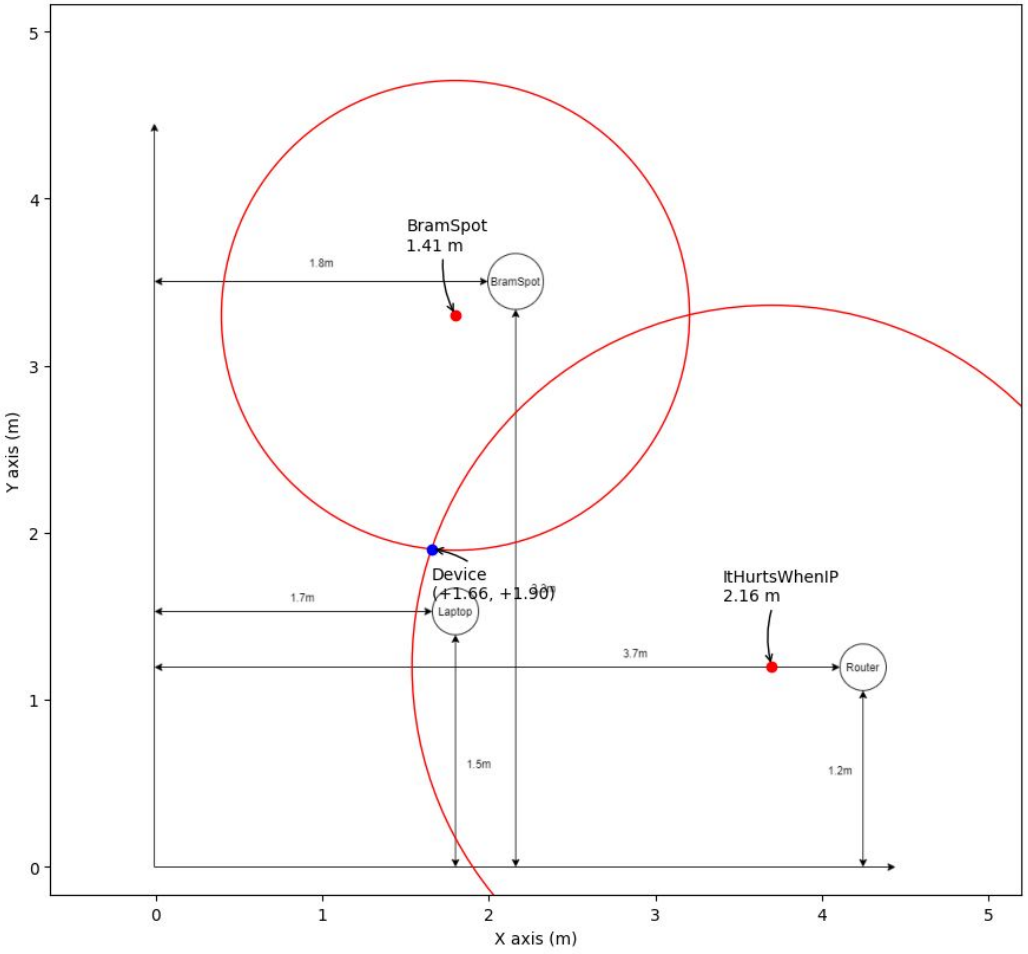


We zien bij het basisgeval (de signaalsterkte of RSSI waarde uit de netwerkmanager van het OS opvragen) dat er grote afwijking is, net zoals bij Windows. Dit is te wijten aan het feit dat de omgerekende afstand tot het AP niet overeenkomt met de werkelijkheid. We zouden de parameters van deze formule nog verder kunnen tweaken of voor elk AP een kalibratiewaarde toevoegen.

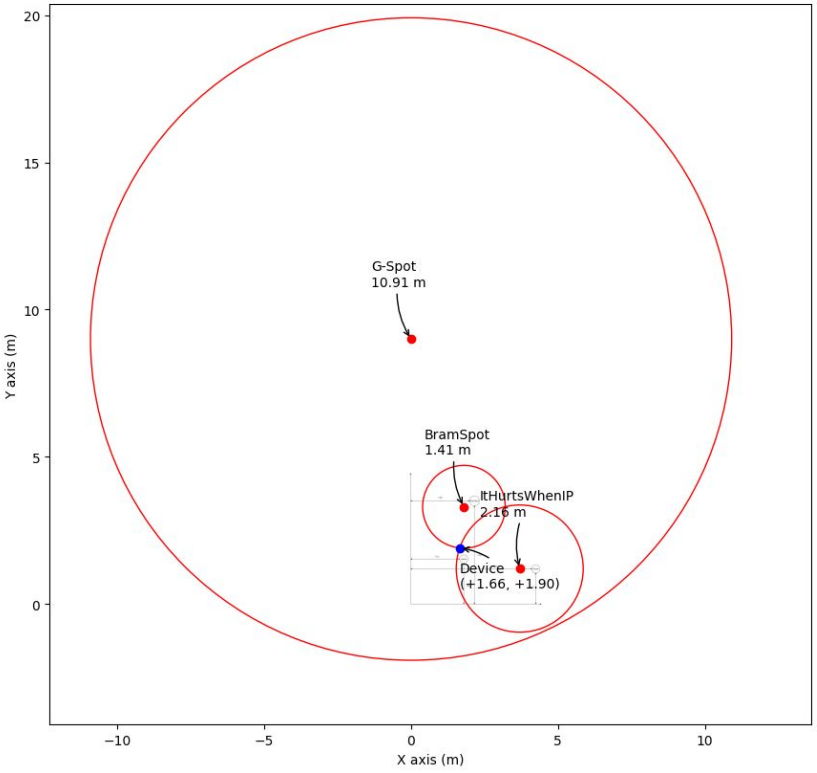
In een praktijksituatie waar het wel duidelijk zou moeten zijn waar iedereen zich bevindt, zou het aan te raden zijn om elk AP in de buurt een exacte locatie op te slaan, samen met een gemeten RSSI en afstand tot AP waar die waarde gemeten werd. Zo zou elk device wel een goede afstandsschatting kunnen maken.

Om toch een meer accurate positie te bepalen van ons device in bovenstaande geval, gebruiken we de `TRILAT_ESTIM` methode. Hierbij nemen we de output van hierboven en berekenen we de andere snijpunten van de cirkels onderling. We kiezen dan het nieuwe snijpunt dat het kortst bij het eerste punt ligt. In bovenstaande geval, zou dat dus effectief vlak langs de echte locatie liggen.

Dit geeft de volgende output, ter verduidelijking hebben we een achtergrond toegevoegd die een realistische weergave geeft van de posities van de verschillende devices.



Het bovenstaande resultaat is bekomen door 3 APs in acht te nemen, om duidelijk te maken dat de schatting vrij goed verloopt moest er echter sterk ingezoomd worden. Een overzicht van de gebruikte hotspots is te zien op onderstaande afbeelding:



Verder werk

Zoals aangegeven in het begin van dit verslag, dachten we er ook aan om op een locatie een fingerprint te nemen door de dBm-waarden van de beschikbare APs op te slaan. En één van de technieken toe te passen zoals beschreven op pagina 4. We hebben geen van deze uiteindelijk geïmplementeerd.

Maar met deze methoden zou het wellicht wel accurater kunnen zijn om de positie te bepalen van het device. Het device moet dan wel telkens zelf elke locatie waar hij zich zal bevinden, gaan kalibreren. Aan de andere kant, is er geen globale info over APs meer nodig op die manier. Een goede usecase voor deze technieken zou binnen een vaste omgeving of gebouw zijn, waar het device vb een karretje, heftruck of dergelijke of misschien wel gewoon een persoon, die de locatie niet verlaat. De specifieke posities die op voorhand bepaald werden, kunnen dan telkens gebruikt worden om te schatten bij welke het device zich momenteel bevindt. Indien het device zich uit deze regio verplaatst, werkt locatiebepaling natuurlijk niet meer.

In de praktijk zou men hiervoor, zoals aangegeven in de opgave, ook beter Bluetooth beacons voor kunnen gebruiken. Deze kunnen in een Beacon modus gezet worden en andere devices kunnen vervolgens altijd de frames van die punten ontvangen, zonder hun eigen connectiviteit op te moeten geven zoals het bij Wi-Fi het geval blijkt. Wel moeten ze dan op voorhand op een specifieke positie geregistreerd worden.