DMA - Lab3 - Positionnement en intérieur – Le Wi-Fi RTT

auteurs: Yanis Ouadahi, Rachel Tranchida, Eva Ray

date: 08.04.2025

Observations

Influence de l'appareil utilisé sur les résultats

En testant l'application sur plusieurs mobiles, nous avons pu constater une différence de précision assez importante selon l'appareil utilisé. En effet, nous avons utilisé un Samsung Galaxy 21 Plus (janvier 2021) et un Pixel 3a (mai 2019) pour tester l'application et le Pixel 3a permettait un positionnement bien plus précis que le S21+ pour la deuxième manipulation. Le pixel 3a permettait d'obtenir une précision entre 0.6 à 1.5m, alors que le S21+, pour la même manipulation, donnait une précision entre 0.8 à 3.5m. En observant les logs, il semble que le pixel 3a détecte les APS de manière plus stable et précise que le S21+, il n'y avait jamais de scan qui ne détectait aucun AP. Ces différences peuvent avoir plusieurs causes:

- La qualité d'implémentation des pilotes Wi-Fi
- Les restrictions logicielles (accès développeur)
- Les différences matérielles (antenne, calibrage, etc.) Après avoir fait quelques recherches, il semble que les smartphones Google Pixel soient reconnus pour leur prise en charge complète et optimisée d Wi-Fi RTT.

Cette observation est importante car elle montre que la précision de la localisation en intérieur peut varier selon le smartphone utilisé et cela indépendamment de la manière dont l'application est implémentée.

Manipulation 1: Lister les AP à portée

Choix d'implémentation

Détection des points d'accès compatibles Wi-Fi RTT

La détection des points d'accès (AP) se fait directement dans la méthode `scanAndRange()` à l'aide du service système `WifiManager`. Ce service permet d'accéder à la liste des AP récemment détectés grâce à la propriété `scanResults`. Cette liste contient des objets de type `ScanResult`, chacun représentant un point d'accès visible à l'instant du scan.

```
val scanResults = wifiManager.scanResults
```

Pour utiliser le Wi-Fi RTT (Round-Trip-Time) basé sur la norme 802.11mc, il est nécessaire de filtrer cette liste afin de ne conserver que les AP compatibles avec cette technologie. Chaque objet `ScanResult` expose une propriété `is80211mcResponder` qui permet d'identifier si l'AP supporte le protocole de ranging.

```
val rttCapableAps = scanResults.filter { it.is80211mcResponder }
```

Une fois cette filtration effectuée, les AP compatibles sont ajoutés à une requête de ranging (`RangingRequest`) via un `RangingRequest.Builder`. Cette requête est ensuite utilisée pour lancer une opération de mesure de distances :

```
val req: RangingRequest = RangingRequest.Builder().apply {
    rttCapableAps.forEach { addAccessPoint(it) }
}.build()
```

Garder les APs non détectés pendant 15 secondes

La classe `WifiRttViewModel` garde une trace locale des points d'accès mesurés via RTT dans une liste mutable interne appelée `accessPointList`. Cette liste contient des objets de type `RangedAccessPoint`, chacun représentant un point d'accès ayant répondu à une requête de mesure RTT. C'est l'attribut `age` de la classe `RangedAccessPoint` qui permet de savoir si un AP a été détecté récemment.

La gestion de `accessPointList` est effectuée dans la méthode `onNewRangingResults()` qui est appelée chaque fois que de nouveaux résultats de ranging sont disponibles. Dans cette méthode, nous mettons à jour la liste des APs détectés en les ajoutant s'ils ne sont pas déjà présents ou en mettant à jour leur attribut `age` s'ils le sont. Si un AP n'a pas été perçu depuis 15 secondes, il est supprimé de la liste.

```
accessPointList.removeAll { currentTime - it.age > 15000 }
```

Une fois la liste mise à jour, nous la postons dans la LiveData `_rangedAccessPoints`, puis nous estimons la distance des APs détectés par rapport au smartphone en utilisant la méthode `estimateDistance()`.

C'est une manière simple et efficace de garder une trace des APs détectés, en s'assurant que seuls les APs récents sont pris en compte, sans créer un objet plus conséquent, comme une classe de caching.

Questions

> 1.1 Par rapport à un seul AP, que pouvez-vous dire sur la précision de la distance estimée ? Est-ce que la présence d'un obstacle (fenêtre, mur, personne) entre l'AP et le smartphone a une influence sur la précision ? Est-ce que faire une moyenne sur plusieurs mesures permet d'avoir une estimation plus fiable de la distance ?

Avec un seul AP, nous disposons uniquement d'une zone, un cercle, mais il est impossible de déterminer "où" nous nous trouvons sur ce cercle. Au lieu d'une localisation précise, un point, nous avons donc plutôt un périmètre dans laquelle nous pouvons nous trouver.

Oui la présence d'un obstacle a une influence sur la précision. En effet, si un obstacle est présent, la force du signal peut être diminuée et il peut également y avoir des effets multipath où le signal atteint le smartphone par plusieurs chemins indirects en plus du chemin direct

Faire la moyenne permet effectivement d'augmenter la précision. Le RSSI fluctue constamment à cause du bruit électronique, des interférences mineures, des effets multipath changeants et des légers mouvements ou changements d'orientation de l'appareil. Prendre la moyenne permet de lisser ces fluctuations.

Manipulation 2: Déterminer la position du smartphone

Choix d'implémentation

Pour calculer la position du smartphone, nous avons utilisé la méthode de trilatération avec la librairie proposée : https://github.com/lemmingapex/trilateration.

On filtre les APs dont la position est disponible dans notre configuration, pour s'assurer que le nombre de distances et le nombre de positions seront les mêmes, on ne peut utiliser que les APs dont on a la localisation. De plus, on s'assure d'avoir au moins 3 positions/APs pour calculer la position, ce qui est le minimum requis pour effectuer le calcul

```
if (positionsArray.size > 2 ) {
            val solver = NonLinearLeastSquaresSolver(
                TrilaterationFunction(positionsArray, distancesArray),
                LevenbergMarquardtOptimizer()
            )
            val optimum = solver.solve()

            val centroid = optimum.point.toArray()
            Log.e(TAG, "Centroid: ${centroid.joinToString()}")
            _estimatedPosition.postValue(centroid)
        } else {
            Log.e(TAG, "Insufficient or mismatched data for trilateration")

        }
```

Dans le cas de l'étage B30, bien que l'on ait que 3 APs, dans notre implémentation on prend les APs compatibles détectés, on les classe par odre croissant de distances et on prend les 4 plus proches. Dans le cas où l'on rajouterait des positions dans notre map, l'implémentation n'aurait pas besoin d'être modifiée et utiliserait donc seulement les 3 APs les plus proches. On en garde 4 car nous avons rajouté la hauteur et il est nécessaire d'avoir au moins 4 APs pour calculer la position en 3D, ce qui a notamment été utilisé pour l'étage B.

Questions

> 2.1 Nous avons également placé des AP à différents endroits de l'étage B. La carte et la position de ces huit AP sont fournies dans le code. Pour activer une localisation sur l'étage B, il suffit de modifier la configuration placée dans la LiveData _mapConfig dans le WifiRttViewModel. Que pouvons-nous dire de la position obtenue en se promenant dans les couloirs de l'étage ? Doit-on tenir compte de tous les AP pour calculer la position ?

Elle est assez fiable, mais avec le lissage et en se déplaçant vite il y a des fluctuations. Surtout si un AP n'est plus détecté car trop loin mais est persisté avec la distance de la dernière détection. Il n'y a pas nécessairement besoin de tous les APs. En effet, on peut se contenter de 3 APs pour une localisation en 2D et

de 4 APs pour une localisation en 3D. Néanmoins, en limitant, le nombre d'AP, le plus judicieux est d'utiliser les APs les plus proches pour le calcul.

> 2.2 Pouvons-nous déterminer la hauteur du mobile par trilatération ? Si oui qu’est-ce que cela implique ? La configuration pour l’étage B contient la hauteur des AP et vous permet donc de faire des tests.

Oui, il est possible de déterminer la hauteur du mobile par trilatération. Pour cela, il faut obligatoirement utiliser au minimum 4 points de référence. Pour avoir une intuition de pourquoi cela, on peut pense en terme d'intersection de sphères dans l'espace.

- 1 sphère : Le point est quelque part à la surface de cette sphère.
- 2 sphères : Leur intersection est un cercle.
- 3 sphères : Leur intersection est 2 points (là où les 3 sphères se croisent).
- 4 sphères : Le quatrième point permet de éliminer l’ambiguïté (l’un des deux points n’est pas cohérent avec la distance à ce quatrième point). C'est pour avoir un seul point d'intersection qu'il faut au minimum 4 points de référence si on veut avoir aussi la hauteur du mobile par trilatération.