Systèmes mobiles
Laboratoire n°3 : Utilisation de données environnementales

24.11.2017

Cet exercice est constitué de manipulations de laboratoire qui vont permettre de vous familiariser avec l'utilisation de données environnementales.

1. Motivation

L'un des problèmes récurrents dans le cadre des applications de grande mobilité est de parvenir à se situer relativement à son environnement. Les applications tirant parti de la localisation sont bien connues, mais la notion d'environnement est plus vaste que le simple lieu où l'on se trouve, et de nombreux paramètres environnementaux ne peuvent pas simplement se réduire à une localisation géographique. Parmi ces paramètres citons, sans souci d'exhaustivité :

- Localisation sociale (sphère privée, sphère familiale, sphère professionnelle, sphère publique)
- Localisation indoor (salon, salle de bains, garage, chambre à coucher, ...)
- Météorologie locale
- Période de l'année, du jour, saison
- Type d'activités en cours (assis, marche, en voiture, en train, …)
- Agenda

Toutes ces informations, et bien d'autres encore, participent à l'environnement au sens large de l'utilisateur. Et il ne faudrait en aucun cas oublier le principal acteur dans cette notion d'environnement, qui est l'utilisateur lui-même ; c'est en effet lui qui définit, dans une large part, l'environnement dans lequel il est plongé à un instant donné.
Une part de la notion d'environnement est "artificielle" ; une géolocalisation par GPS ou triangulation d'antennes, par exemple, est une notion qui dépend de la position de l'utilisateur, et est largement indépendante de l'application ; en revanche, la lecture d'une balise RFID, même si elle est utilisée pour donner aussi une indication de position, est artificielle en ce sens que l'application a été spécifiquement prévue pour tirer parti de la présence de cette balise.
L'objectif de cette manipulation est d'examiner quelques-uns des outils utilisés pour "fabriquer" un environnement spécifique apte à permettre le fonctionnement d'applications particulières.

2. Balises NFC

2.1 Introduction à la technologie[1]

NFC (Near Field Communication) est une technologie sans fils autorisant l'échange d'informations entre deux terminaux compatibles à une distance maximale d'environ 4 centimètres[2]. La technologie fonctionne dans une bande de fréquences située vers 13.56 MHz, et permet des transferts de données à un débit compris entre 106 et 424 kbit/s. Contrairement à une technologie comme Bluetooth, il n'y a pas de phase de découverte, ni de pairage préalable entre partenaires de la communication, ce qui permet un temps de mise en fonction (setup time) très court (typiquement 100 ms).

NFC constitue une relation asymétrique, avec un initiateur qui va fournir le champ électrique nécessaire à alimenter une cible qui n'a quant à elle, pas besoin de source d'énergie indépendante. Il y a trois modes de communication différents définis dans le cadre de NFC (voir Fig. 1) :

- **Peer-to-peer** Dans ce mode, des échanges bidirectionnels peuvent avoir lieu, avec des types de données indifférents. Ce mode de communication peut être fonctionnellement comparé à une communication *Bluetooth* ou *WiFi*. Ce mode de communication est le mode le plus populaire dès lors que l'on intègre la technologie NFC sur des terminaux intelligents (comme des smartphones), car c'est le mode qui permet le mieux de tirer parti de la capacité de traitement du smartphone. La technologie *Android Beam* est basée sur ce mode de fonctionnement.
- **Read/Write** : En mode lecture, le périphérique NFC acquiert des données d'une balise externe, généralement passive. Ce mode remplit une fonctionnalité similaire à la lecture d'un code-barres. Certaines balises supportent aussi la fonction d'écriture, qui permet au terminal de stocker des données sur la balise. Les types et la quantité de données que l'on peut stocker sur les balises sont très divers, et les balises elle-même peuvent se présenter sous divers conditionnements (porte-clés, cartes de visite, étiquettes autocollantes, patches souples, ...)
- **Card / Tag emulation** : En mode émulation de balise[3], le terminal NFC se comporte comme une balise passive. Le lecteur ne peut faire la différence entre un terminal en mode émulation et une balise passive.

En mode peer-to-peer, l'initiant et la cible sont tous deux actifs (génèrent donc des champs RF). Dans les autres modes, seul l'initiant génère un champ qui sera utilisé par la cible. Il n'y a aucun chiffrement au niveau liaison dans le cadre de la technologie NFC. Si une application a un besoin spécifique de sécurité, le chiffrement du contenu est à sa charge. NFC est un cas particulier de RFID, avec une portée limitée à quelques centimètres. Dans le cas d'utilisation dans un contexte de balise, le profil NDEF (NFC Data Exchange Format) est particulièrement bien adapté.

[1] Le texte est largement inspiré d'un travail préparatoire à une thèse master de M. Carlo Criniti, TR

[2] En principe, la distance dépend essentiellement des antennes en présence, ainsi que de l'intensité du champ. Une balise NFC pourrait en théorie être lue à beaucoup plus grande distance dans la mesure où l'antenne de lecture a les dimensions suffisantes qui lui autorisent un gain adéquat. Bien sûr, une telle antenne est peu compatible avec un lecteur de poche (téléphone portable, par exemple), aussi bien du point de vue des dimensions que de la consommation d'énergie.

[3] https://developer.android.com/guide/topics/connectivity/nfc/hce.html

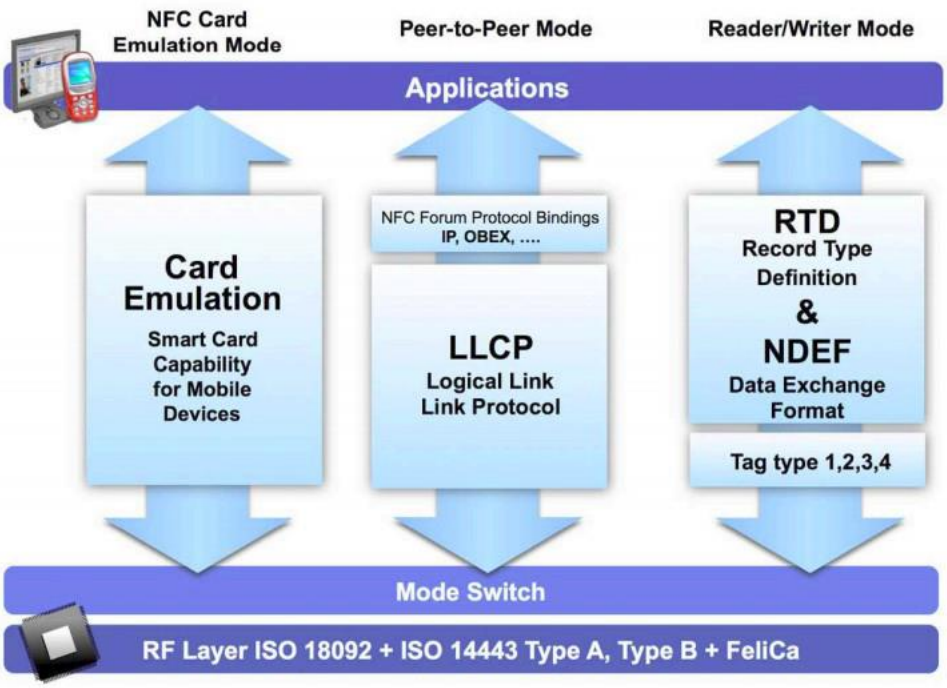

Figure 1 - Modes de fonctionnement NFC

2.2 Manipulation

Dans l'optique de la sécurisation d'une application, on aimerait utiliser un porte-clés muni d'une puce NFC en guise de "credential", en conjonction avec un dispositif mobile (smartphone, tablette) capable de lire des balises NFC. Pour autant, on ne désire pas toutefois renoncer au traditionnel login/mot de passe. Développer une application dont l'accès est sécurisé :

1. Par la combinaison d'un mot de passe **ET** d'une balise NFC adéquate
2. Par l'introduction d'un mot de passe **OU** la lecture d'une balise NFC adéquate

Condition supplémentaire, on désire que cette identification ait une durée de vie restreinte, et qu'elle implémente un effet de vieillissement. Ainsi, immédiatement après l'authentification, on souhaite que le niveau d'authentification soit maximal, et corresponde par exemple à un score arbitraire de AUTHENTICATE_MAX (disons égal à 10, par exemple). Après un temps à définir, ce niveau va décroître, jusqu'à tomber finalement à zéro. Ceci permettrait dans une hypothétique application d'avoir des objets très sensibles qui sont consultables uniquement immédiatement après une authentification (Required : AUTHENTICATE_MAX) pour prévenir une utilisation après vol du mobile en mode authentifié, alors que certains autres, moins sensibles, peuvent être utilisés pendant une période plus longue après l'authentification (Required : AUTHENTICATE_MEDIUM ou AUTHENTICATE_LOW). Bien évidemment, une nouvelle lecture de la balise NFC remet le niveau d'authentification à son maximum.

On trouvera des exemples d'implémentation de lecture de tags NFC sur le web (par exemple http://mobile.tutsplus.com/tutorials/android/reading-nfc-tags-with-android/) en grand nombre, aussi bien pour lire que pour écrire sur un tag NFC. En l'occurrence, on se contentera de lire le tag (au format NDEF !) pour vérifier son authenticité, puis le combiner avec d'éventuelles autres informations pour décider de la validité de l'authentification.

2.3 Détail d'implémentation

Il existe plusieurs familles de tags NFC : NFC-A (ISO 14443-3A), NFC-B (ISO 14443-3B), NFC-F (JIS 6319-4) et NFC-V (ISO 15693) qui sont des normes ouvertes et par exemple *MifareClassic* ou *MifareUltralight* qui sont des formats propriétaires, pas disponibles sur tous les devices. Le SDK Android permet de travailler à plusieurs niveaux d'abstraction avec les tags NFC, nous allons rester au plus haut niveau possible, nous manipulerons des messages formatés en NDEF uniquement.

Le SDK Android dédié à NFC travail avec des *Intents*, il est possible d'ajouter des *<intent-filter>* dans le `manifest` afin que notre activité ou notre service soit lancé automatiquement par le système lors de la lecture d'un tag ou alors d'enregistrer notre activité à l'exécution pour qu'elle soit notifiée uniquement lorsqu'elle est visible à l'écran, nous utiliserons la seconde méthode dans ce laboratoire. La Fig. 2 donne un exemple de code pour que notre activité reçoive les événements NFC, vous devrez en plus surcharger la méthode *onNewIntent()* qui sera appelée pour les tags à proximités.

```
// called in onResume()
private void setupForegroundDispatch() {

    if(mNfcAdapter == null)
        return;

    final Intent intent = new Intent(this.getApplicationContext(),
                this.getClass());
    intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

    final PendingIntent pendingIntent =
         PendingIntent.getActivity(this.getApplicationContext(), 0, intent, 0);

    IntentFilter[] filters = new IntentFilter[1];
    String[][] techList = new String[][]{};

    // Notice that this is the same filter as in our manifest.
    filters[0] = new IntentFilter();
    filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
    filters[0].addCategory(Intent.CATEGORY_DEFAULT);
    try {
        filters[0].addDataType("text/plain");
    } catch (IntentFilter.MalformedMimeTypeException e) {
        Log.e(TAG, "MalformedMimeTypeException", e);
    }

    mNfcAdapter.enableForegroundDispatch(this, pendingIntent, filters, techList);
}

// called in onPause()
private void stopForegroundDispatch() {
    if(mNfcAdapter != null)
        mNfcAdapter.disableForegroundDispatch(this);
}
```

Figure 2 - Exemple de code pour inscrire l'activité aux événements NFC

2.4 Questions

Sachant que les collaborateurs de l'entreprise UBIQOMP SA se déplacent en véhiculant des informations précieuses dans leurs dispositifs informatiques mobiles (munis de dispositifs de lecture NFC), et qu'ils sont amenés à se rendre dans des zones à risque, un expert a fait les estimations suivantes :

- La probabilité de vol d'un mobile par une personne malintentionnée et capable d'utiliser les données à des fins préjudiciables pour la société est de 1% ;
- La probabilité que le mot de passe puisse être découvert, soit par analyse des traces de doigts sur l'écran, soit par observation en cours d'utilisation est de 4% ;

- La probabilité de vol du porte-clés est de 0.1% ;
- Environ 10% des criminels susceptibles d'accéder aux données du mobile sait que le porte-clés permet l'accès au mobile.

Quelle est la probabilité moyenne globale que des données soient perdues, dans le cas où il faut la balise ET le mot de passe, ainsi que dans le cas où il faut la balise OU le mot de passe (on négligera dans le calcul la probabilité de l'intersection des deux ensembles), ou encore le cas où seule la balise est nécessaire ? En d'autres termes, si l'on envoie cent collaborateurs en déplacement, quel est le risque encouru de vol de données sensibles ? Mettre vos conclusions en rapport avec l'inconfort subjectif de chaque solution.
Peut-on améliorer la situation en introduisant un contrôle des informations d'authentification par un serveur éloigné (transmission d'un hash SHA256 du mot de passe et de la balise NFC) ? Si oui, à quelles conditions ? Quels inconvénients ?
Proposer une stratégie permettant à la société UBIQOMP SA d'améliorer grandement son bilan sécuritaire, en détailler les inconvénients pour les utilisateurs et pour la société.

3. Codes-barres

La définition proposée par Wikipedia : « *Un code-barres, ou code à barres, est la représentation d'une donnée numérique ou alphanumérique sous forme d'un symbole constitué de barres et d'espaces dont l'épaisseur varie en fonction de la symbologie utilisée et des données ainsi codées. Il existe des milliers de codes-barres différents ; ceux-ci sont destinés à une lecture automatisée par un capteur électronique, le lecteur de code-barres. Pour l'impression des codes-barres, les technologies les plus utilisées sont l'impression laser et le transfert thermique.* »

Il existe de nombreux types de codes-barres différents, mais on peut les classer en 2 catégories principales, les codes unidimensionnels (1D) et les codes bidimensionnels (2D). Les premiers permettent généralement de stocker une courte chaine de caractères alors que les seconds permettent de stocker plus de données. Dans le cas des codes-barres 2D, plus on ajoute d'information, plus le code sera grand et donc moins il sera aisé de le lire.

Figure 3 - Un exemple de code-barres 1D à gauche et 2D à droite

Les codes-barres 2D de type **code QR** (Quick Response Code, exemple à droite sur la Fig. 3) sont composés de pixels noirs sur un fond blanc carré, l'information est contenue dans l'agencement de ces points noirs, les données sont protégées avec un code correcteur d'erreur de type Reed-Solomon. Bien qu'il soit possible de stocker du texte brut, certains types prédéfinis existent et permettent de déclencher une action lors de la lecture, par exemple :

- Ouvrir une URL (Page web, vidéo, position sur une carte, etc.) ;
- Se connecter à une borne Wi-Fi ;
- Déclencher un appel vers un numéro de téléphone ou envoyer un SMS ;
- Ajouter un contact à son annuaire ;
- Envoyer un e-mail, etc.

Ce type de code est destiné à une lecture sans contact optique. Le coût de fabrication d'un tel code est pratiquement nul (inférieur à la technologie NFC) ; mais requiert une focalisation optique pour une lecture correcte. Les lecteurs laser à absorption de lumière réfléchie, tels que les lecteurs utilisés dans les caisses enregistreuses de grands magasins, sont actuellement très performants.

Il est possible de déchiffrer un code-barres à l'aide d'un simple capteur photo ; mais cela pose le problème de la focalisation de l'objectif : même si les objectifs "fix-focus" des smartphones d'entrée de gamme sont très tolérants en termes de mise au point, il n'est pas toujours évident de leur faire lire correctement un code-barres, même simple (1D). On se propose dans cette manipulation de mettre en évidence les avantages et inconvénients de l'utilisation de balises optiques basées sur des codes-barres lorsque l'on utilise un smartphone pour déchiffrer la balise proprement dite.

Pour contourner l'obstacle assez considérable du décodage de l'image, nous allons utiliser un logiciel/librairie open source, accessible sur le Google Play Store, appelé Barcode Scanner et mis au point par l'éditeur Zxing. Ce n'est certainement pas le logiciel le plus performant en termes de vitesse et de précision de détection en conditions médiocres, mais il est bien documenté, et il peut être aisément invoqué depuis une application externe grâce à une API basée sur des Intent mise à disposition par l'éditeur :

- https://github.com/zxing/zxing/wiki/Scanning-Via-Intent[4]

Cette librairie peut aussi être intégrée directement dans votre application, cela permet d'éviter à vos utilisateurs de devoir installer une seconde application. Un portage de cette libraire permet une intégration plus facile car bénéficiant d'une meilleure documentation :

- https://github.com/journeyapps/zxing-android-embedded

3.1 Manipulation

L'objectif de cette manipulation est simplement d'être en mesure de lire un code-barres multidimensionnel (de type code QR), et d'afficher la valeur du code dans une activité que vous aurez définie vous-même. On peut par exemple imaginer que ce code barre remplace le code NFC de la manipulation précédente.

Il existe de nombreux sites internet permettant de générer des codes QR sur Internet, comme par exemple http://generator.code-qr.net/#text.

3.2 Questions

Comparer la technologie à codes-barres et la technologie NFC, du point de vue d'une utilisation dans des applications pour smartphones, dans une optique :

- Professionnelle (Authentification, droits d'accès, clés de chiffrage)
- Grand public (Billetterie, contrôle d'accès, e-paiement)
- Ludique (Preuves d'achat, publicité, etc.)
- Financier (Coûts pour le déploiement de la technologie, possibilités de recyclage, etc.)

[4] Ce n'est pas précisé sur cette page, mais il est possible d'ajouter cette librairie à votre projet en ajoutant au fichier build.graddle (module: app) dans les dépendances, la ligne suivante :

```
compile group: 'com.google.zxing', name: 'android-integration', version: '3.3.0'
```

4. Balises iBeacon

Présenté en 2013 par Apple, les iBeacons sont un système de positionnement en intérieur basé sur la technologie Bluetooth low energy. Le fonctionnement est très simple, le beacon diffuse régulièrement un identifiant qui peut être capté par les appareils à proximité, qui pourront alors déclencher une action spécifique (par exemple afficher une notification).
Les appareils, Android ou iOS, disposant de Bluetooth 4.0 sont en général compatibles avec les iBeacons. Contrairement à iOS, sur Android les iBeacon ne disposent pas d'un API dédiée dans le SDK, il est donc recommandé d'utiliser des librairies facilitant leur intégration. On peut conseiller l'utilisation de Android Beacon Library (https://github.com/AltBeacon/android-beacon-library). Attention pour que celle-ci fonctionne correctement il faut obligatoirement demander les permissions de géo-localiser l'utilisateur[5] et d'accès à Internet, attention à bien vérifier aussi que le Bluetooth est activé.

Tous les constructeurs de beacons n'utilisent pas exactement le même format, il faudra spécifier à la librairie le format utilisé, pour nos beacons (*SmartBeacon*) :

```
new BeaconParser().setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24")
```

4.1 Manipulation

L'objectif est de lister sur une activité les différents iBeacons à proximité. Vous afficherez pour chacuns le rssi (force du signal), le numéro majeur et le numéro mineur dans une activité. Il faudra que la liste affichée se mette régulièrement à jour.

4.2 Questions

Les iBeacons sont très souvent présentés comme une alternative à NFC. Pouvez-vous commenter cette affirmation en vous basant sur 2-3 exemples de cas d'utilisations (use-cases) concrets (par exemple e-paiement, second facteur d'identification, accéder aux horaires à un arrêt de bus, etc.).

5. Capteurs

Nous avons vu pendant le cours que les appareils mobiles disposent d'un certain nombre de capteurs. Ceux-ci permettent d'obtenir diverses informations sur l'environnement physique entourant le smartphone, on pensera, par exemple à l'accélération, au champ magnétique, à la pression atmosphérique ou encore à la température ambiante.

5.1 Manipulation

L'objectif de cette manipulation et de créer une boussole en 3D permettant d'afficher la direction du nord magnétique tout en restant à l'horizontale (voir Fig. 4). Nous nous baserons pour cela les capteurs d'accélération (`SENSOR_TYPE_ACCELEROMETER`) et magnétique (`SENSOR_TYPE_MAGNETOMETER`) et nous utiliserons ensuite les données obtenues pour générer la matrice de rotation à l'aide de la méthode helper suivante :
`SensorManager.getRotationMatrix(float[] R, null, float[] gravity, float[] geomagnetic)`[6]

Pour vous aider à réaliser ceci nous vous fournissons le template d'une application qui implémente déjà la partie de visualisation en 3D, nous utilisons pour réaliser cela la librairie graphique *OpenGL* intégrée au SDK *Android*[7], vous n'avez pas besoin de comprendre le code de cette partie mais n'hésitez pas à le consulter et à nous poser des questions en cas d'intérêt. Vous utiliserez uniquement la méthode `float[] swapRotMatrix(float[] rotMatrix)` mise à disposition par la classe

[5] http://altbeacon.github.io/android-beacon-library/samples.html
[6] https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[],%20float[],%20float[],%20float[])
[7] https://developer.android.com/guide/topics/graphics/opengl.html

OpenGLRenderer fournie dans le projet template. Une instance de cette classe `opglr` existe dans l'activité `CompassActivity`. Cette méthode permet de passer au moteur de rendu 3D la matrice de rotation obtenue lors de cette manipulation afin que la flèche 3D soit dessinée à l'écran dans la bonne direction. Pour des raisons de performances (dans les applications 3D nous travaillons en général à une fréquence de 60 images par seconde, ce qui laisse environs 16 ms pour préparer la prochaine image) cette méthode va retourner l'ancienne matrice de rotation, l'idée est de travailler avec uniquement 2 tableaux que l'on échange sans avoir à recréer un nouveau tableau (qui devra être « garbage collecté ») plusieurs fois par secondes.

Figure 4 - Capture d'écran de la boussole 3D

5.2 Questions

Une fois la manipulation effectuée, vous constaterez que les animations de la flèche ne sont pas fluides, il va y avoir un tremblement plus ou moins important même si le téléphone ne bouge pas. Veuillez expliquer quelle est la cause la plus probable de ce tremblement et donner une manière (sans forcément l'implémenter) d'y remédier.

6. Rendu/Evaluation

Pour rendre votre code, nous vous demandons de bien vouloir zipper votre projet Android Studio, vous veillerez à bien supprimer les dossiers build (à la racine et dans app/) pour limiter la taille du rendu. En plus, vous remettrez un document **pdf** comportant au minimum les réponses aux questions posées.

Merci de rendre votre travail sur *CyberLearn* dans un zip unique. N'oubliez pas d'indiquer vos noms dans le code, sur vos réponses et de commenter vos solutions.

Bonne chance !