Android : Statut du réseau

On a souvent besoin de savoir

- si le réseau est disponible.
- quel type de réseau est disponible

Usage : par exemple, pour les applications gourmandes en échanges de données, on veut limiter l'usage de certaines fonctions aux connexions de type wifi, car les données mobiles peuvent revenir cher, selon le type d'abonnement de l'utilisateur.

Dominique Archambault
Master MIASHS « Handi »
Université Paris 8-Vincennes-Saint-Denis

Gestionnaire de réseaux

- Classe **android.net.ConnectivityManager**
  C'est la classe qui permet d'avoir des informations sur les réseaux

```
ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
```

Liste des réseaux disponibles

- **public Network[] getAllNetworks()**
  renvoie un tableau de tous les réseaux présents
- Lister tous les réseaux :

```
for (Network network : connMgr.getAllNetworks()) {
        …
}
```

Ce réseau est-il connecté à Internet ?

- **public NetworkCapabilities getNetworkCapabilities(Network)**
  => informations sur les possibilités d'un réseau

```
NetworkCapabilities capabilities =
        connMgr.getNetworkCapabilities(network);
Boolean IsConnected = false ;
if (capabilities != null) {
        isConnected = capabilities.hasCapability(
                NetworkCapabilities.NET_CAPABILITY_INTERNET) ;
}
```

Type du réseau (WIFI, data, *etc.*)

- **public NetworkInfo getNetworkInfo(Network)** informations sur le type réseau

```
NetworkInfo networkInfo = connMgr.getNetworkInfo(network);

if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI)
        isWifiConn |= networkInfo.isConnected();

else if (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE)
                isMobileConn |= networkInfo.isConnected();
```

Type du réseau (WIFI, data, *etc.*)

- Rmq : la méthode getType() est deprecated depuis l’API 28
- La nouvelle méthode consiste à utiliser NetworkCapabilities et de considérer les méthodes de transport avec NetworkCapabilities.hasTransport(int)
- *Cette méthode permet aussi de regarder la bande passante*
- *Néanmoins même sur dev.android il est indiqué que la plupart des apps n’ont pas besoin de se préoccuper des transports*

Détecter un changement de réseau

- On va implémenter un **BroadcastReceiver**

```
public class NetworkReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            // ce qu'on veut faire dans quand il y a un changement
        }
    }
```

Détecter un changement de réseau

- Puis on va le déclarer avec :
  Intent registerReceiver(
  BroadcastReceiver receiver,
  IntentFilter filter)
  (par exemple dans le onCreate() de l'activité)
- Attention à libérer dans le onDestroy()*
- On utilise un objet IntentFilter pour limiter la réception aux changements de connectivité

Détecter un changement de réseau


```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	...
	IntentFilter filter =
		new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
	receiver = new NetworkReceiver();
	this.registerReceiver(receiver, filter);
	...
}

@Override
public void onDestroy() {
	super.onDestroy();
	if (receiver != null) { this.unregisterReceiver(receiver); }
}
```

Documentation

- https://developer.android.com/training/basics/network-ops/managing
- https://developer.android.com/reference/android/net/ConnectivityManager
- https://developer.android.com/reference/android/net/NetworkCapabilities
- https://developer.android.com/reference/android/net/NetworkInfo