Android : Communications par Wifi

Frédéric Lemoine

Conservatoire National des Arts et Métiers

Communications par Wifi

- Wifi (Wireless Fidelity) est un système de communication sans-fil normalisé prévu pour relier, sous la forme d'un réseau commun, un ensemble d'appareils informatiques (ordinateurs, tablettes ...).
- Rayon d'action: environ 300 mètres
- Autonomie: de l'ordre de quelques heures.

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

WifiManager wifiManager= (WifiManager) getSystemService(Context.WIFI_SERVICE);

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE">
</uses-permission>
Autorise l'application à obtenir des informations sur les réseaux Wifi.

<uses-permission android:name="android.permission.CHANGE_WIFI_STATE">
</uses-permission>
Autorise l'application a être prévenue lors d'un changement de l'état de connexion.

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE">
</uses-permission>
Autorise l'application à obtenir des informations sur les réseaux.

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

Méthodes de la classe WifiManager

Méthodes	Commentaires
int getWifiState()	Donne l'état de l'adaptateur Wifi (actif ou non).
boolean isWifiEnabled()	Retourne vrai si l'adaptateur Wifi est actif
boolean setWifiEnabled(boolean enabled)	Active (enabled=true) ou désactive (enabled=false) l'adaptateur Wifi
boolean startScan()	Débute une recherche de points d'accès
List<ScanResult> getScanResults()	Donne la liste des points d'accès récemment découverts.
static int calculateSignalLevel(int rssi, int numLevels)	Donne le niveau du signal reçu. Il s'échelonne de 0 à numLevels-1 et est calculé à partir du RSSI (Received Signal Strength Indication).
static int compareSignalLevel(int rssiA, int rssiB)	Compare 2 niveaux de signaux
boolean pingSupplicant()	Teste si le service responsable des connexions répond.

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

Int addNetwork(WifiConfiguration config)	Ajoute le paramétrage pour la connexion à un nouveau réseau Wifi. Retourne un identifiant sur cette configuration.
boolean removeNetwork(int netId)	Supprime une configuration réseau existante.
int updateNetwork(WifiConfiguration config)	Modifie une configuration réseau existante.
boolean disableNetwork(int netId)	Désactive une configuration donnée
boolean enableNetwork(int netId, boolean disableOthers)	Active un réseau. si disableOthers est à vrai, tous les autres réseaux sont désactivés et une nouvelle connexion est tentée sur le réseau sélectionné.
List<WifiConfiguration> getConfiguredNetworks()	Retourne la liste de toutes les configurations réseaux précédemment ajoutées.
boolean saveConfiguration()	Sauve toutes les configurations réseaux. Attention: les identifiants fournis précédemment pourront avoir été modifiés après cet appel.

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

WifiInfo getConnectionInfo()	Retourne une série d'informations sur la connexion actuelle si active.
boolean disconnect()	Se déconnecte du point d'accès courant.
boolean reconnect()	Se reconnecte sur le point d'accès.
boolean reassociate()	Se reconnecte sur le point d'accès courant même si l'on est déjà connecté.
DhcpInfo getDhcpInfo()	Retourne la dernière adresse DHCP (Dynamic Host Configuration Protocol) alloué par un serveur.
WifiManager.WifiLock createWifiLock(String tag)	Place un verrou sur le point d'accès et empêche le signal radio de se couper après une période d'inactivité.
WifiManager.WifiLock createWifiLock(int lockType, String tag)	Idem que précédemment mais en précisant un type de verrou.
WifiManager.MulticastLock createMulticastLock(String tag)	L'appareil demande à recevoir les paquets réseaux adressés sur des adresses multicast.

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

```
<uses-permission android:name="android.permission.WAKE_LOCK">
</uses-permission>
```

Si l'adaptateur Wifi n'est pas activé, on l'active:

```
if(!wifiManager.isWifiEnabled()) {
    wifiManager.setWifiEnabled(true);
}
```

Communications par Wifi
Accès au gestionnaire Wifi (WifiManager)

```
int wifiState=wifiManager.getWifiState();
```

Etat de l'adaptateur	Commentaires
WifiManager.WIFI_STATE_DISABLED	Désactivé
WifiManager.WIFI_STATE_DISABLING	En cours de désactivation
WifiManager.WIFI_STATE_ENABLED	Activé
WifiManager.WIFI_STATE_ENABLING	En cours d'activation
WifiManager.WIFI_STATE_UNKNOWN	Inconnu

Communications par Wifi
Détails de l'état de la connexion

WifiInfo getConnectionInfo()

Méthodes	Commentaires
String getSSID()	Le nom donné au réseau sur lequel on est connecté. SSID est l'acronyme de Service Set Identifier.
String getBSSID()	L'adresse matérielle (MAC) du point d'accès sur lequel on est connecté.
int getIpAddress()	Donne l'adresse Ip de l'appareil.
int getLinkSpeed()	Donne la vitesse de transmission en WifiInfo.LINK_SPEED_UNITS (Méga bits par seconde actuellement).
String getMacAddress()	Donne l'adresse matérielle (MAC) de l'appareil.
int getNetworkId()	L'identifiant de la configuration réseau utilisée.

Communications par Wifi
Détails de l'état de la connexion

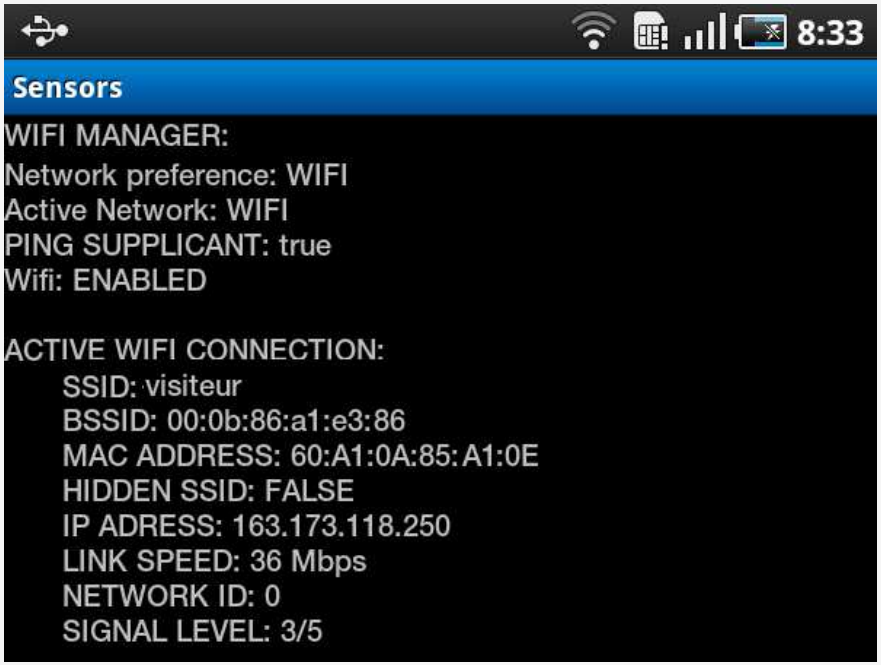
int getRssi()	Donne la force du signal reçu. RRSI est l'acronyme de Received Signal Strength Indication.
boolean getHiddenSSID()	Vrai si le réseau ne diffuse pas son SSID.
SupplicantState getSupplicantState()	Donne l'état des négociations entre le service Wifi et le point d'accès (en cours d'association, d'authentification ...)
static NetworkInfo.DetailedState getDetailedStateOf(SupplicantState suppState)	Donne un ensemble de renseignements supplémentaires sur un état obtenu par la fonction précédente.

Communications par Wifi
Détails de l'état de la connexion

```
WifiInfo wifiInfo = wifiManager.getConnectionInfo();

String ssid=wifiInfo.getSSID();
String bsid=wifiInfo.getBSSID();
int ipAdress=wifiInfo.getIpAddress();
String ipa=String.format("%d.%d.%d.%d",(ipAdress & 0xff),(ipAdress >> 8 &
0xff),(ipAdress >> 16 & 0xff),(ipAdress >> 24 & 0xff));
// Vitesse en Mbps
int linkSpeed=wifiInfo.getLinkSpeed();
String macAddress=wifiInfo.getMacAddress();
int networkId=wifiInfo.getNetworkId();
// Niveau de signal de 0 à 4
int signalLevel=WifiManager.calculateSignalLevel(wifiInfo.getRssi(),5);
boolean bHidden=wifiInfo.getHiddenSSID();
```

Communications par Wifi
Détails de l'état de la connexion



Communications par Wifi
Détection des points d'accès

```
wifiManager.startScan();

BroadcastReceiver connectionReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

if(intent.getAction().equals(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)) {
                // Recherche terminée
                }
        }
};
```

Communications par Wifi
Détection des points d'accès

```
IntentFilter filterScan = new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);

@Override
protected void onResume() {
	super.onResume();
	registerReceiver(connectionReceiver, filterScan);
}

@Override
protected void onPause() {
	super.onPause();
	unregisterReceiver(connectionReceiver);
}
```

Communications par Wifi
Détection des points d'accès

message SCAN_RESULTS_AVAILABLE_ACTION

List<ScanResult> scanResults = wifiManager.getScanResults();

Caractéristiques du point d'accès	Commentaire
String BSSID	L'adresse matérielle du point d'accès
String SSID	Le nom du point d'accès
String capabilities	Les capacités de cryptage et d'authentification offerte par le point d'accès.
int frequency	La fréquence du canal sur lequel émet le point d'accès en MHz.
int level	Le niveau du signal en dBm (décibels au-dessus d'un milliwatt. La puissance de référence est 1 mW).

Communications par Wifi
Détection des points d'accès

```
for(ScanResult scan:scanResults) {
        String ssid=scan.SSID;
        String bsid=scan.BSSID;
        String capabilities=scan.capabilities;
        int frequency=scan.frequency;
        int level=scan.level;
        }
```

Communications par Wifi
Détection des points d'accès