

Introduction to Android Connectivity

CS 436 Software Development on Mobile

Dr.Paween Khoenkaw

Department of Computer Science
Maejo University



Connectivity Manager

Connectivity Manager

Connectivity Manager is class that answers queries about the state of network connectivity. It also notifies applications when network connectivity changes

- Monitor network connections (Wi-Fi, GPRS, UMTS, etc.)
- Send broadcast intents when network connectivity changes
- Attempt to "fail over" to another network when connectivity to a network is lost
- Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks

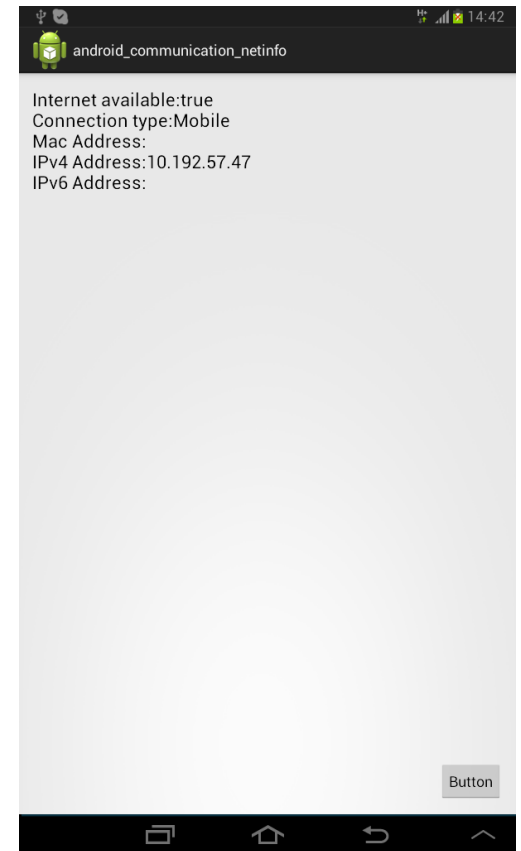
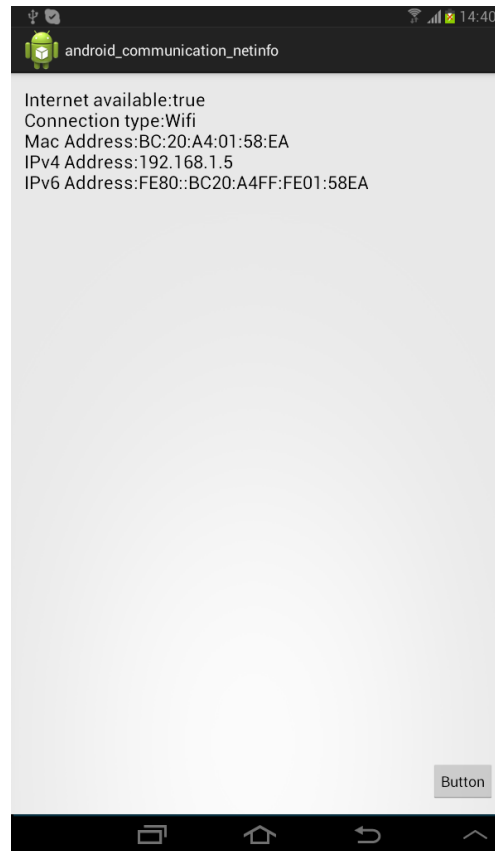
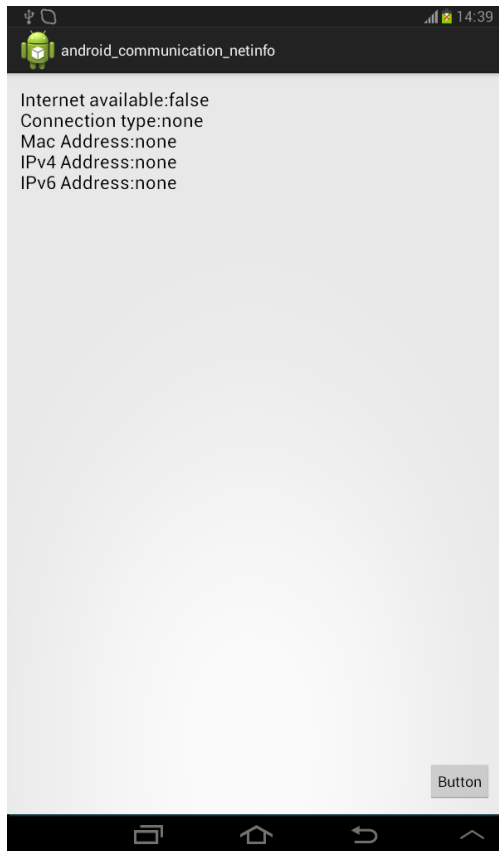
Connectivity Manager must used with Broadcast Receiver Class

Connectivity Manager

Technology supported

TYPE_BLUETOOTH	The Default Bluetooth data connection.
TYPE_DUMMY	Dummy data connection.
TYPE_ETHERNET	The Default Ethernet data connection.
TYPE_MOBILE	The Default Mobile data connection.
TYPE_MOBILE_DUN	A DUN-specific Mobile data connection.
TYPE_MOBILE_HIPRI	A High Priority Mobile data connection.
TYPE_MOBILE_MMS	An MMS-specific Mobile data connection.
TYPE_MOBILE_SUPL	A SUPL-specific Mobile data connection.
TYPE_WIFI	The Default WIFI data connection.
TYPE_WIMAX	The Default WiMAX data connection.

Connectivity Manager



Project: android_communication_netinfo

Connectivity Manager

Check for online status

- 1) Uses permission
"android.permission.ACCESS_NETWORK_STATE"
and "android.permission.INTERNET"
- 2) Create broadcast receiver for intent
"android.net.conn.CONNECTIVITY_CHANGE"
- 3) Use ConnectivityManager to query Network info
- 4) Query IP address and Mac address from Linux Kernel (using NetUtil class)

Connectivity Manager

Check for online status

1) Uses permission

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

Connectivity Manager

Check for online status

2) Create broadcast receiver for intent

“**android.net.conn.CONNECTIVITY_CHANGE**”

```
IntentFilter ifilter_charge = new  
IntentFilter("android.net.conn.CONNECTIVITY_CHANGE");  
BroadcastReceiver batteryLevelReceiver = new  
BroadcastReceiver(){  
    @Override  
    public void onReceive(Context arg0, Intent arg1) {  
        //UpdateNetStatus();  
    }  
};  
  
registerReceiver(batteryLevelReceiver, ifilter_charge);
```


Connectivity Manager

Check for online status

3) Use ConnectivityManager to query Network info

```
public boolean isOnline() {  
    ConnectivityManager cm =(ConnectivityManager)  
    getSystemService(Context.CONNECTIVITY_SERVICE);  
  
    return cm.getActiveNetworkInfo() != null &&  
           cm.getActiveNetworkInfo().isConnectedOrConnecting();  
}
```

Connectivity Manager

Check for online status

3) Use ConnectivityManager to query Network info

```
private void UpdateNetStatus()
{
    boolean isOnline_msg=isOnline();
    String connectiontype="none";
    if(isOnline_msg==true)
    {    ConnectivityManager cm =
(ConnectivityManager)this.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
boolean isConnected = activeNetwork.isConnectedOrConnecting();
```

Connectivity Manager

Check for online status

```
switch(activeNetwork.getType())
{
case ConnectivityManager.TYPE_BLUETOOTH:connectiontype="Bluetooth";break;
case ConnectivityManager.TYPE_ETHERNET:connectiontype="Ethernet";break;
case ConnectivityManager.TYPE_MOBILE:connectiontype="Mobile";break;
case ConnectivityManager.TYPE_MOBILE_DUN:connectiontype="Dial up";break;
case ConnectivityManager.TYPE_WIFI:connectiontype="Wifi";break;
case ConnectivityManager.TYPE_WIMAX:connectiontype="Wimax";
}
}
}
```

Connectivity Manager

Check for online status

4) Use NetUtil class to decode IP address and Mac address

Required:
NetUtil.java

```
String MacAddress="none";  
String IpV4Address="none";  
String IpV6Address="none";
```

```
MacAddress=NetUtil.getMACAddress("wlan0");  
IpV4Address=NetUtil.getIPAddress(true);  
IpV6Address=NetUtil.getIPAddress(false);
```

True= IPv4,False=IPv6

Connectivity Manager

```
public class CheckNet extends BroadcastReceiver {  
    public CheckNet() {  
    }  
    @Override  
    public void onReceive(Context arg0, Intent arg1) {  
Log.v("net state", "change");  
// do something  
    }  
}
```

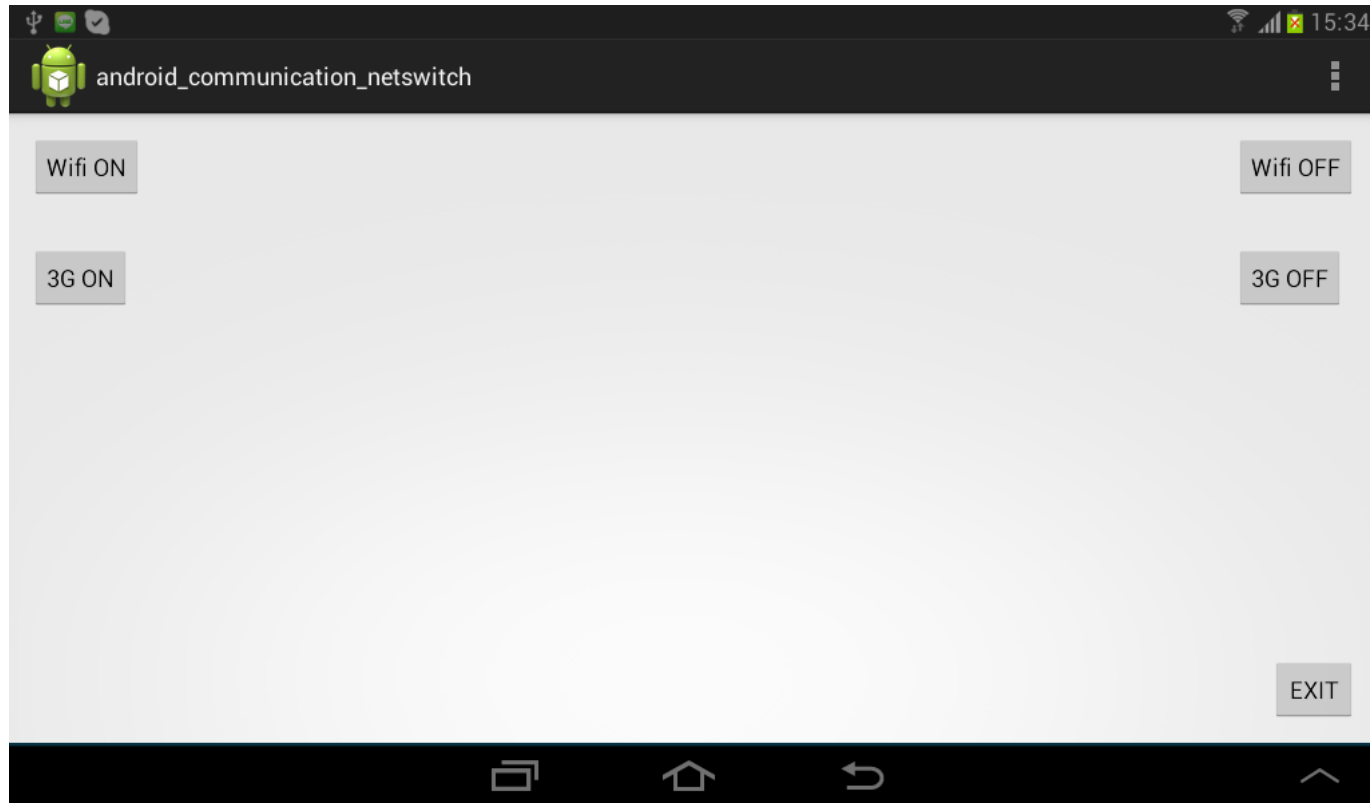
```
<receiver android:name="CheckNet" android:enabled="true"  
android:permission="android.net.conn.CONNECTIVITY_CHANGE">  
    <intent-filter>  
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>  
    </intent-filter>  
</receiver>
```

Connectivity Manager

Control network interfaces

Wifi Manager and Connectivity Manager

Connectivity Manager



Project: android_communication_netswitch

Connectivity Manager

Control Mobile data

- 1) Uses permission "`android.permission.CHANGE_WIFI_STATE`" and "`android.permission.CHANGE_NETWORK_STATE`"
- 2) Create `WifiManager` instance
- 3) Control WIFI using method `setWifiEnabled`

Connectivity Manager

Control Mobile data

- 1) Uses permission "**android.permission.CHANGE_WIFI_STATE**" and "**android.permission.CHANGE_NETWORK_STATE**"

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

Connectivity Manager

Control Mobile data

2) Create **WifiManager** instance

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

Connectivity Manager

Control Mobile data

3) Control WIFI using method `setWifiEnabled`

```
wifiManager.setWifiEnabled(true);  
wifiManager.setWifiEnabled(false);
```

Connectivity Manager

Control Mobile data

- 1) Uses permission "`android.permission.CHANGE_WIFI_STATE`" and "`android.permission.CHANGE_NETWORK_STATE`"
- 2) Create `ConnectivityManager` instance
- 3) Use Java reflection to access method `MobileDataEnabled`
- 4) Use method `MobileDataEnabled` to turn 3G `on` and `off`

Connectivity Manager

Control Mobile data

- 1) Uses permission "**android.permission.CHANGE_WIFI_STATE**" and "**android.permission.CHANGE_NETWORK_STATE**"

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

Connectivity Manager

Control WIFI

- 2) Create **ConnectivityManager** instance

```
ConnectivityManager conman;  
conman = (ConnectivityManager)  
this.getSystemService(Context.CONNECTIVITY_SERVICE);
```

Connectivity Manager

Control WIFI

3) Use Java reflection to access method **MobileDataEnabled**

Reflection is the ability of a computer program to **examine** (see type introspection) and **modify** the structure and behavior (specifically the values, meta-data, properties and functions) of an object **at runtime**.

In this case, we use reflection for late binding method

Connectivity Manager

3) Use Java reflection to access method **MobileDataEnabled**

API

```
conman = (ConnectivityManager) this.getSystemService(Context.CONNECTIVITY_SERVICE);
conman.s
Class co
try {
    con
    iCon
    iCon
    iCon
    iCon
    setM
    setM
} catch
// T
e.pr
} catch
// T
e.pr
} catch
// T
```

setNetworkPreference(int preference) : void - ConnectivityManager
startUsingNetworkFeature(int networkType, String feature) : int
stopUsingNetworkFeature(int networkType, String feature) : int

public void setNetworkPreference (int preference)
Added in [API level 1](#)

Press 'Ctrl+Space' to show Template Proposals
Press 'Tab' from proposal table or click for focus

Android source code

```
isTetheringSupported() : boolean
requestRouteToHost(int, int) : boolean
setBackgroundDataSetting(boolean) : void
setMobileDataEnabled(boolean) : void
setNetworkPreference(int) : void
setRadio(int, boolean) : boolean
setRadios(boolean) : boolean
startUsingNetworkFeature(int, String) : int
stopUsingNetworkFeature(int, String) : int
tether(String) : int
untether(String) : int
```

Returns:
Whether mobile data is enabled.

Hide:

```
361
362
363
364
365
366
367
368
```

```
public boolean getMobileDataEnabled() {
    try {
        return mService.getMobileDataEnabled();
    } catch (RemoteException e) {
        return true;
    }
}
```

Sets the persisted value for enabling/disabling Mobile data.

Parameters:
enabled Whether the mobile data connection should be used or not.

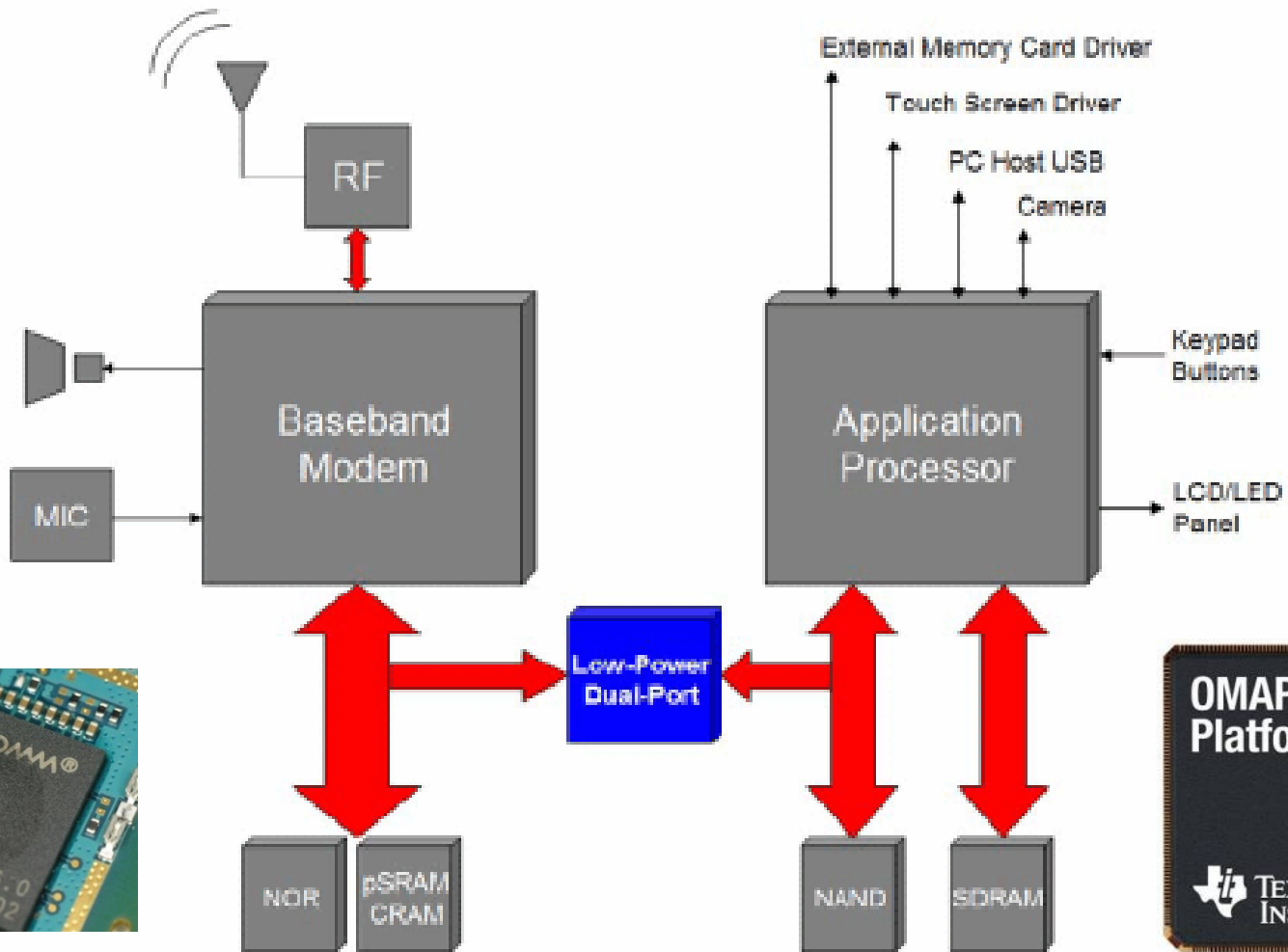
Hide:

```
376
377
378
379
380
381
382
```

```
public void setMobileDataEnabled(boolean enabled) {
    try {
        mService.setMobileDataEnabled(enabled);
    } catch (RemoteException e) {
    }
}
```

```
EXTRA_AVAILABLE_TETHER : String
EXTRA_ERRORED_TETHER : String
EXTRA_EXTRA_INFO : String
EXTRA_IS_FAILOVER : String
EXTRA_NETWORK_INFO : String
EXTRA_NO_CONNECTIVITY : String
EXTRA_OTHER_NETWORK_INFO : String
EXTRA_REASON : String
MAX_NETWORK_TYPE : int
MAX_RADIO_TYPE : int
mService : IConnectivityManager
```


THE SMART PHONE



Connectivity Manager

Control Mobile data

3) Use Java reflection to access method **MobileDataEnabled**

```
Field iConnectivityManagerField;  
Object iConnectivityManager;  
Class iConnectivityManagerClass;  
Method setMobileDataEnabledMethod;  
.....
```

Connectivity Manager

3) Use Java reflection to access method **MobileDataEnabled**

```
Class conmanClass;
try {
    conmanClass = Class.forName(conman.getClass().getName());
    iConnectivityManagerField = conmanClass.getDeclaredField("mService");
    iConnectivityManagerField.setAccessible(true);
    iConnectivityManager = iConnectivityManagerField.get(conman);
    iConnectivityManagerClass =
Class.forName(iConnectivityManager.getClass().getName());
    setMobileDataEnabledMethod =
iConnectivityManagerClass.getDeclaredMethod("setMobileDataEnabled",
Boolean.TYPE);
    setMobileDataEnabledMethod.setAccessible(true);
} catch (ClassNotFoundException e) {
} catch (NoSuchFieldException e) {
} catch (IllegalArgumentException e) {
} catch (IllegalAccessException e) {
} catch (NoSuchMethodException e) {
}
```

Connectivity Manager

4) Use method `MobileDataEnabled` to turn 3G **on** and **off**

```
try {
setMobileDataEnabledMethod.invoke(iConnectivityManager, true);
    } catch (IllegalArgumentException e) {
    } catch (IllegalAccessException e) {
    } catch (InvocationTargetException e) {
    }
}
```

```
try {
setMobileDataEnabledMethod.invoke(iConnectivityManager, false);
    } catch (IllegalArgumentException e) {
    } catch (IllegalAccessException e) {
    } catch (InvocationTargetException e) {
    }
}
```