**Chapter 3: Implementation**

**Section 1: Channel Detection Implementation**

To demonstrate the implementation details of the channel list detection, we takes the LTE List Fragment as an example. The implementation of Wi-Fi List Fragment is almost the same, the difference will be highlighted at the end of this section.

**Section 1.1: Data Model implementation**

**Section 1.2: XML file implementation**

A xml file is the specific file used in android to handle UI.

To use fragment, we first need to define a container in the xml file of the activity. A container is used to hold the fragment defined later. A container can hold any fragment, so LTE List Fragment and Wi-Fi List Fragment both can be hold by this container.

/\* activity\_fragment.xml \*/

<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/android"

**android:id="@+id/fragment\_container"**/>

We use a RecyclerView to display the LTE List we have obtain. RecyclerView is a sub-class of ViewGroup. It displays a list of child View objects.

/\*fragment\_list.xml\*/

<**android.support.v7.widget.RecyclerView**

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/recycler\_view" />

Also we need to define how each of the elements are going to be displayed inside the list.

/\*list\_item\_lte.xml\*/

<TextView

**android:id="@+id/list\_pci\_text\_view"**/>

<TextView

**android:id="@+id/list\_dbm\_text\_view"**

android:layout\_below="@+id/list\_pci\_text\_view"/>

<TextView

**android:id="@+id/list\_asulevel\_text\_view"**

android:layout\_below="@+id/list\_dbm\_text\_view" />

The each LTE channel is displayed as follow:

PCI:

DBM:

ASU:

**Section 1.3: Fragment implementation**

Then, we create the LTE List Fragment class which is a sub-class of the standard Fragment class from Android:

/\*LTEListFragment.java\*/

public class LTEListFragment extends Fragment {

**private RecyclerView mRecyclerView;**

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_list, container, false);

**mRecyclerView = (RecyclerView) view.findViewById(R.id.recycler\_view);**

return view;

}

}

Figure(2) RecyclerView Mechnism

As shown of figure two, when we use the RecyclerView, a LayoutManager and a Adapter are required.

The LayoutManager is responsible for placing the itew views on right position inside the RecyclerView. Also it determines when to reuse the item views which are created previously and no long visible to user. In our application, we use the LinearLayoutManager provided by RecyclerView API.

/\*LTEListFragment.java\*/

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_list, container, false);

mRecyclerView = (RecyclerView) view.findViewById(R.id.recycler\_view);

**mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));**

return view;

}

As for Adapter, it plays two roles: provide access to the data set and create the correct layout of each individual items the RecyclerView is going to display. We need to override two methods, onCreateViewHolder, which is used to inflate the view and its view holder, and also onBindViewHolder, which is used to bind the data to the view.

As for onCreateViewHolder, we need to define the holder we are going to use first. The holder is used to extract the information from the data model we used and display thoes informaiton on the itew view we have already created.

/\*LTEListFragment.java\*/

private class **Holder** extends RecyclerView.ViewHolder

{

private LTE mLTE;

private TextView mPci;

private TextView mDbm;

private TextView mAsuLevel;

public Holder(View itemView) {

super(itemView);

itemView.setOnClickListener(this);

mPci = (TextView) itemView.findViewById(R.id.list\_pci\_text\_view);

mDbm = (TextView) itemView.findViewById(R.id.list\_dbm\_text\_view);

mAsuLevel = (TextView) itemView.findViewById(R.id.list\_asulevel\_text\_view);

}

public void bindLTE(LTE lte) {

**mLTE = lte;**

**mPci.setText**(...**String.valueOf(lte.getmPci())**);

**mDbm.setText**(...**String.valueOf(lte.getmDbm())**);

**mAsuLevel.setText(String.valueOf(lte.getmAsuLevel())));**

}

}

/\*LTEListFragment.java\*/

public **Holder** onCreateViewHolder(ViewGroup parent, int viewType) {

LayoutInflater layoutInflater = LayoutInflater.from(getActivity());

View view = layoutInflater.inflate(R.layout.list\_item\_lte, parent, false);

return new Holder(view);

}

onBindViewHolder is

**Section 2: Localization detection implementation**