

**MINI PROJECT
(2020-21)**

File Sharing Android Application- "Easy Share"

MID-TERM REPORT



Institute of Engineering & Technology

Submitted by-

- 1) Prachi Bansal (181500459)
- 2) Neha Adnekar (181500421)
- 3) Mani Bansal (181500364)
- 4) Vivek Goyal (181500817)
- 5) Jitendra Singh (181500300)

Supervised By: -

Mr. Neeraj khanna
Technical Trainer

Department of Computer Engineering & Applications

Contents

Abstract	3
1. Introduction	4
1.1 General Introduction to the topic	4
1.2 Working and Application Area	5
1.3 Hardware and Software Requirements	
1.4 Development challenges	6
2. Problem definition	6
3. Objectives	6
4. Implementation Details	7
5. Progress till Date & The Remaining work	17
6. Some Screenshots	18
7. References	30

Abstract

In this we will build an android application that will transfer data from one device to another device using Wi-fi Direct. Our Team named this app "Easy Share". Easy Share is an open source Wi-fi Direct file sharing application that will enable sharing of data between Android devices running Android 4.0 or higher using a Wi-Fi direct connection without the use of a separate Wi-Fi access point. This will enable data transfer between devices without relying on any existing network infrastructure. This application is intended to provide a much higher speed alternative to Bluetooth file transfer.

There will be a client and a server i.e. the two devices that are connected through Wi-Fi direct. Once the connection is established, you can transfer data. Easy Share involves creating and registering a broadcast receiver for the application itself, discovering peers, connecting to a peer, and transferring data to a peer.

In this project comparative study and analysis of file sharing applications has been performed and presented. These file sharing applications enable the users for faster sharing rate among them through WLAN. They provide ease of access among multiple compatible devices. Several tests have been conducted including transferring several types of files and results have been gathered and these results are used in the analysis. A comparative analysis has also been done and the challenges or issues that are present in the concerned file sharing applications have been mentioned. The work primarily focuses on the commonly used file sharing applications in Smartphone's worldwide namely Xender and SHARE it.

Introduction

1.1 General Introduction to the topic

Today's computers are capable of sharing all types of files, including documents, songs, videos, and full applications. File sharing is a key function for many businesses and other use cases.

Easy Share is an open source Wi-fi Direct file sharing application that will enable sharing of data

between Android devices. File sharing is transferring or sharing the digital data such as image, audio, video, document etc. stored on an electronic device (mobile, laptop etc.) with similar compatible electronic devices. In this project we are using a Wi-Fi direct connection without the use of a separate Wi-Fi access point for sharing the files.

Wi-Fi Direct, initially called Wi-Fi P2P, is a Wi-Fi standard enabling devices to easily connect with each other without requiring a wireless access point. It is usable for everything from internet browsing to file transfer, and to communicate with more than one device simultaneously at typical Wi-Fi speeds. One advantage of Wi-Fi Direct is the ability to connect devices even if they are from different manufacturers. Only one of the Wi-Fi devices needs to be compliant with Wi-Fi Direct to establish a peer-to-peer connection that transfers data directly between them with greatly reduced setup.

Wi-fi Direct negotiate the link with a Wi-fi Protected Setup System that assigns each device a limited wireless access point. The "pairing" of Wi-fi direct devices can be set up to require the proximity of a near field communication, a Bluetooth, signal, or a button press one or all Devices. Wi-fi Direct may not only the need for Routers, but may also replace the need of Bluetooth for Application that do not rely on low battery.

File sharing is essential, and we are now having many ways to perform this but every way has its merits and demerits. For storing the use of dedicated server is used which has various demerits and also the cloud server is used which is complicated for deployment and also require more systems. The file sharing has become very popular now a day for its higher data secure and utilisation properties.

They should be able to send and receive the files from mobile, through Wi-fi Direct and which is designed as this project and the product of the project is named as "Easy Share".

1.2.1 Working:

The basic steps for a Wi-fi direct application are:

- Initial Setup
- Discovering Peers
- Connecting to Peers
- Transferring Data

We followed these steps in our application and registered a Broadcast receiver to listen to the intents. The way Wi-fi direct works is that we can call certain methods from the main activity to perform certain Wi-fi direct functions like search peers and connect. The methods will interact with the hardware. When hardware is ready, it will broadcast some intents to the OS and software. The broadcast receiver then can catch these intents and do the follow up works accordingly.

1.2.2 Application Area:

- **Cyber Security field-** File sharing apps are a common tool used in cyber-attacks, both by professional hackers and ordinary individuals.
- **Education field-** Study stuff & business documents instantly shared.
- **Government-** Government use file sharing to share the important information on websites with limited rights.
- **IT Industries-** With file sharing app, employee effort and time—become efficiently utilized.

1.3.1 Hardware Requirements:

- Processor: i3
- Hard Disk: 5GB
- Memory: 1GB RAM
- Android Phone with KitKat and higher

1.3.2 Software Requirements:

- Windows 7 (ultimate, enterprise, Windows 8, Windows 10)
- Android Studio

1.4 Development Challenges

- **Scalability:** The aim of scaling is to allow the system to expand in scale without disrupting the activities of the users. The enormous data should not lead to corrupt or any other issues.
- **Heterogeneity:** the system should be available and accessible from different platform without any difference in performance.
- **Real-Time Data Delivery:** the system should be able to deliver data in real-time whenever data is requested.
- **Limited Bandwidth:** Although Internet is available to all devices(within the targeted), but many of devices have very slow connection speed and for better costing Limited Bandwidth is to be taken care of.

Problem Statement

To design File storing and sharing system having following features

1. To able to implement Peer-to-Peer server :
Peer-to-peer (P2P) computing or networking is a [distributed application](#) architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.
2. To create Data base as back end to store data on the servers:
A **database server** is a server which uses a [database application](#) that provides [database](#) services to other computer programs or to [computers](#), as defined by the [client-server model](#).
3. Create a web-platform as user interface for downloading, uploading and sharing of files:
In its simplest form, a peer-to-peer (**P2P**) **network** is **created** when two or more PCs are connected and share resources without going through a separate server computer. A **P2P network** can be an ad hoc connection—a couple of computers connected via a Universal Serial Bus to transfer files
4. Create a desktop application which facilitate to synchronise a folder with server ensuring the files in that folder has its copy on server :
File synchronization ensures that two or more data storage devices, computers, or programs have the same information at a given time. You would agree to this fact that organizing your business data across various devices is not as simple as it sounds.
5. Designing an Android application, so that user can store or send any file from the mobile to server :
Once you have set up your app to share files using content URIs, you can respond to other apps' requests for those files. One way to respond to these requests is to provide a file selection interface from the server app that other applications can invoke. This approach allows a client application to let users select a file from the server app and then receive the selected file's content URI.
6. Ensuring the encryption of data on the server for batter security:

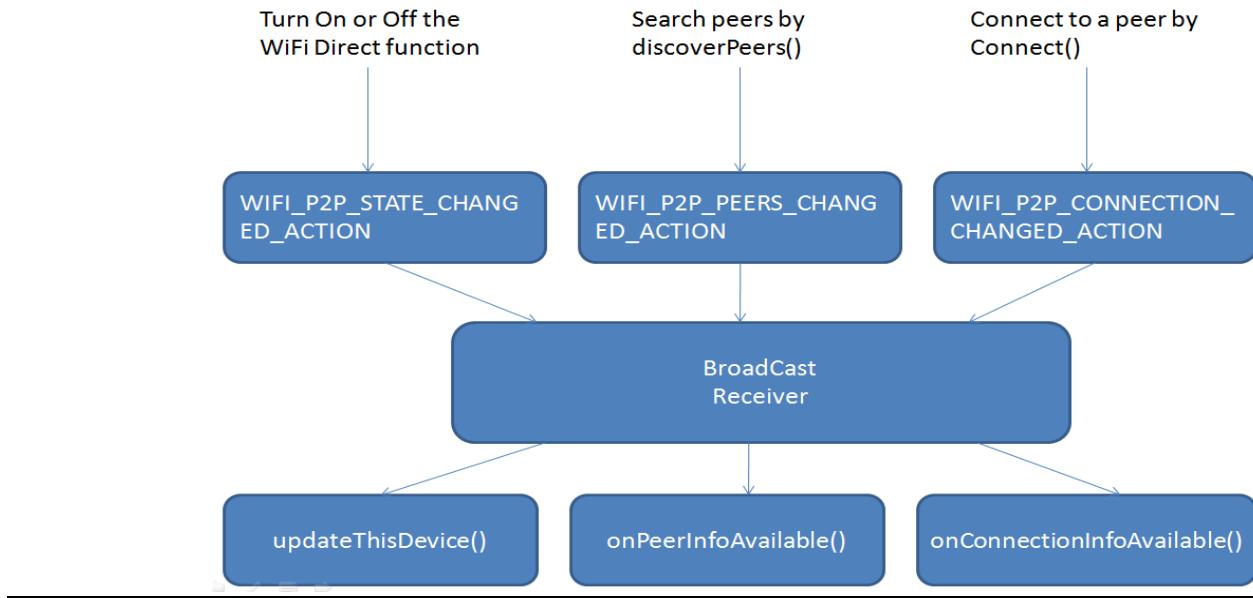
To transfer files to and from a server without danger of hackers compromising or stealing data, it is vital to use **File Transfer Protocol Secure (FTPS)**. It encrypts data files and your authentication information. FTPS uses both a command channel and a data channel, and the user can encrypt both. Bear in mind that it only protects files during transfer. As soon as they reach the server, the data is no longer encrypted. For this reason, encrypting the files before sending them adds another layer of security.

7. Applying several business concepts like advertisement and Entrepreneurship skills:
There are different processes in a single business - for example, operations, marketing, recruiting, customer support, finance, etc., and ensuring that the same data is available to all the team members working in these different processes is itself a challenge.

Objective

The main objectives of creating this android Application is to enable easy access and transfer of personal content between local storage devices using P2P networks and FTP(file transfer protocol) without turning on the data .

Implementation Details



1. Classes :-

WifiP2pManager: This is a class which is used to interact with the Wi-Fi hardware on our device which helps in connecting and discovering the peers.

2. Methods :-

`WifiP2pManager` class provides some methods for peer-to-peer connection.

- **initialize():** This method registers the application within a Wi-Fi framework, and it is called before calling any other Wi-Fi P2p methods.
- **connect():** This method is used to start a Wi-Fi pair connection with a device having a specific
- **discoverPeers():** This method is used to detect all the available peers that are in the range.
- **cancelConnect():** This method is used to cancel all the connected Wi-Fi pair connections.
- **removeGroup():** This method is used to remove the current peer-to-peer group.

3. Intents

When using the Wi-Fi pair connection, we need to listen for broadcast intents and handle the events occurred.

1WIFI_P2P_STATE_CHANGED_ACTION: It indicates a change in the Wi-Fi P2P status

WIFI_P2P_PEERS_CHANGED_ACTION: It indicates a change in the list of available peers

WIFI_P2P_CONNECTION_CHANGED_ACTION: It indicates the state of Wi-Fi P2P

connectivity is changed

WIFI_P2P_THIS_DEVICE_CHANGED_ACTION: It indicates the device details are changed

EasyShare with Source code

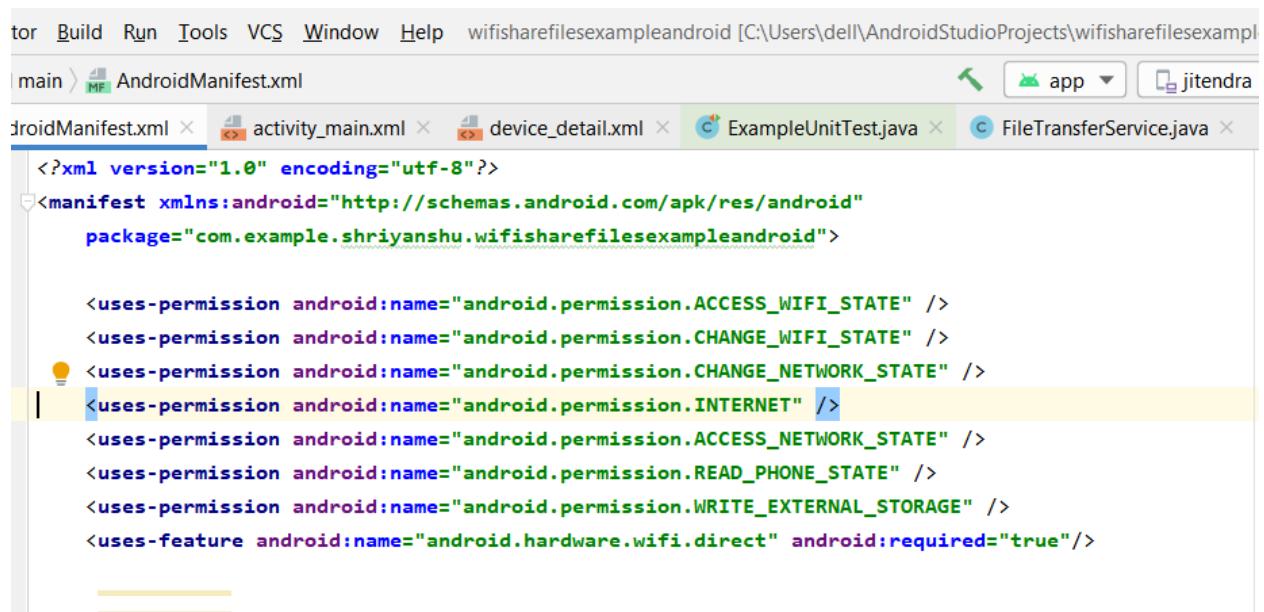
This Android Application which transfers files by creating a Wi-Fi pair connection in Android.

- Create an Application **WiFiShareFilesExampleAndroid** in Android Studio
- Add the permissions in the **AndroidManifest.xml** file.

AndroidManifest.xml

To use the Wi-Fi pair connection, we have to add some necessary permissions in this file.

Screenshot-



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shriyanshu.wifisharefilesexampleandroid">

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-feature android:name="android.hardware.wifi.direct" android:required="true"/>

```

Now do it programmatically by creating some Java files and XML files in our project.

Layout Files :-

1. activity_main.xml

This is our main layout file. Here, we add two fragments inside LinearLayout.

Screenshots

```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:weightSum="1">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_alignParentTop="true"
        android:layout_weight="0.9"
        android:orientation="vertical">

        <fragment
            android:id="@+id/fragment_list"
            class="com.example.shriyanshu.wifisharefilesexampleandroid.DeviceListFragment"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            tools:layout="@layout/device_list"></fragment>

        <fragment
            android:id="@+id/fragment_detail"
            class="com.example.shriyanshu.wifisharefilesexampleandroid.DeviceDetailFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:layout="@layout/device_detail"></fragment>
    </LinearLayout>
</LinearLayout>

```

2. device_detail.xml

Create this XML file to add the layout for device details.

Screenshots

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:visibility="gone">
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <LinearLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <Button
                android:id="@+id/btn_connect"
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Connect"
                android:background="@color/colorPrimary"
                android:layout_margin="5dp"
                android:textColor="#ffffff"/>
            <Button
                android:id="@+id/btn_disconnect"
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Disconnect"
                android:background="@color/colorPrimary"
                android:layout_margin="5dp"
                android:textColor="#ffffff"/>
            <Button
                android:id="@+id/btn_start_client"
                android:layout_width="130dp"
                android:layout_height="wrap_content"
                android:text="Launch Gallery"
                android:layout_width="150dp"
                android:layout_height="wrap_content"
                android:text="Launch Gallery"
                android:visibility="gone"
                android:background="@color/colorPrimary"
                android:layout_margin="5dp"
                android:textColor="#ffffff"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_gravity="center"
            android:layout_margin="5dp"
            android:id="@+id/ll_statusBar">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:layout_margin="5dp"
                android:id="@+id/tv_statusText">
        </LinearLayout>
    </FrameLayout>

```

3. device list.xml

Create this XML file to add the layout for device lists.

Screenshots

The image shows two side-by-side screenshots of the Android Studio interface. Both screenshots display the same XML files: `device_list.xml` and `row_devices.xml`.

device_list.xml Content:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:paddingTop="3dp">
8
9      <TextView
10         android:layout_width="fill_parent"
11         android:layout_height="wrap_content"
12         android:gravity="center_vertical"
13         android:text="ME" />
14
15     <View
16         android:layout_width="fill_parent"
17         android:layout_height="1dp"
18         android:gravity="center_vertical"
19         android:background="@android:color/holo_blue_light" />
20
21     <!-- Self information -->
22     <LinearLayout
23         xmlns:android="http://schemas.android.com/apk/res/android"
24         android:layout_width="fill_parent"
25         android:layout_height="?android:attr/listPreferredItemHeight"
26         android:background="?android:attr/activatedBackgroundIndicator"
27         android:padding="3dp">
28
29         <ImageView
30             android:id="@+id/iv_Icon"
31             android:layout_width="wrap_content"
32             android:layout_height="fill_parent"
33             android:layout_marginRight="2dp"
34             android:src="@mipmap/ic_launcher" />
35
36         <LinearLayout
37             android:orientation="vertical"
38             android:layout_width="0dp"
39             android:layout_weight="1"
40             android:layout_height="fill_parent">
41
42             <TextView
43                 android:id="@+id/tv_myName"
44                 android:layout_width="fill_parent"
45                 android:layout_height="0dp"
46                 android:layout_weight="1"
47                 android:gravity="center_vertical" />
48
49             <View
50                 android:layout_width="fill_parent"
51                 android:layout_height="1dp"
52                 android:gravity="center_vertical"
53                 android:background="@android:color/holo_blue_light" />
54
55         <!-- Available peers -->
56         <ListView
57             android:id="@+id/android:list"
58             android:layout_width="match_parent"
59             android:layout_height="match_parent"
60             android:layout_weight="1"
61             android:drawSelectorOnTop="false" />
62
63         <TextView
64             android:id="@+id/android:empty"
65             android:layout_width="match_parent"
66             android:layout_height="match_parent"
67             android:layout_gravity="center"
68             android:text="No devices found. Turn on P2P and perform dis/>
69
70     </LinearLayout>
71
72 </LinearLayout>
73
74 <LinearLayout
75     android:layout_width="match_parent"
76     android:layout_height="match_parent"
77     android:layout_gravity="center"
78     android:background="@android:color/holo_blue_light" />
79
80 </LinearLayout>

```

row_devices.xml Content:

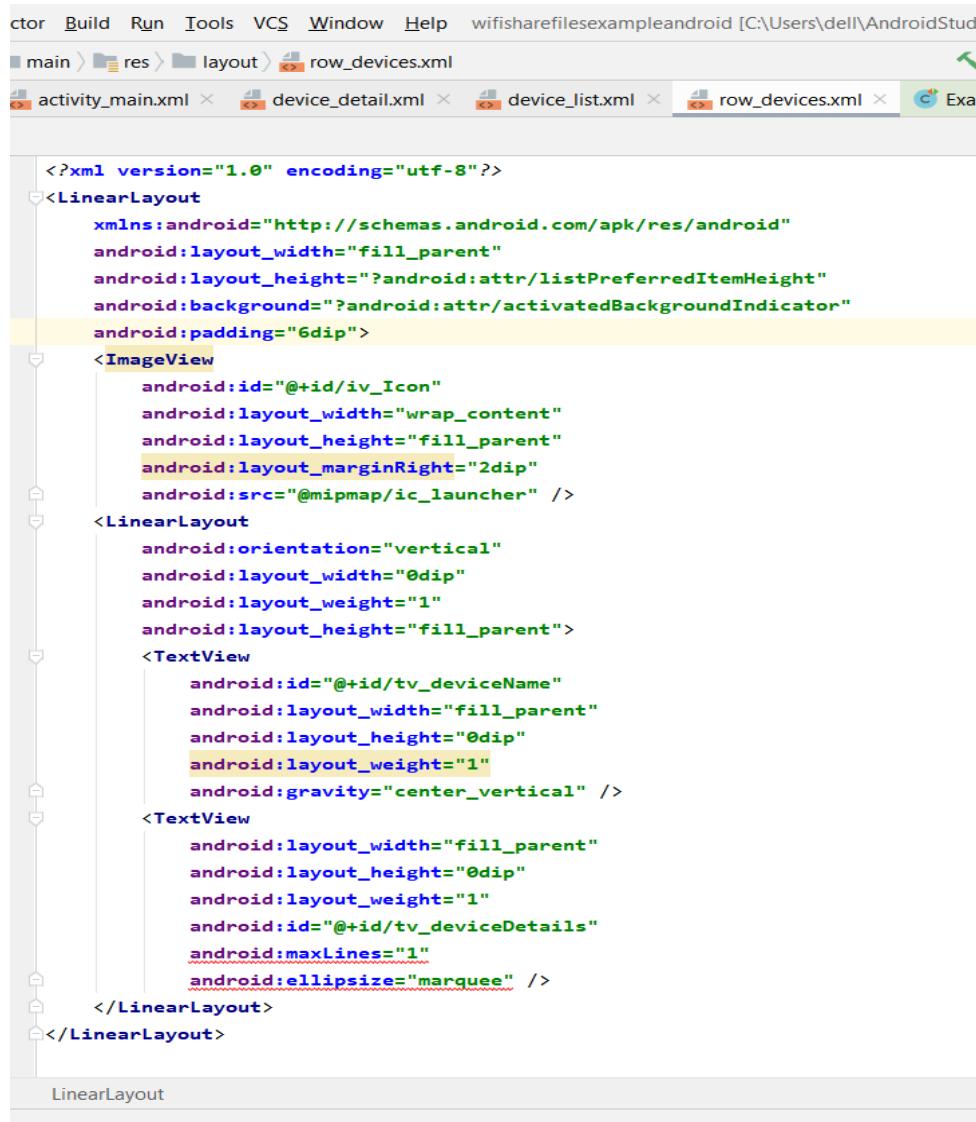
```

39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

```

4. row_devices.xml

Create this XML file and add the code here.



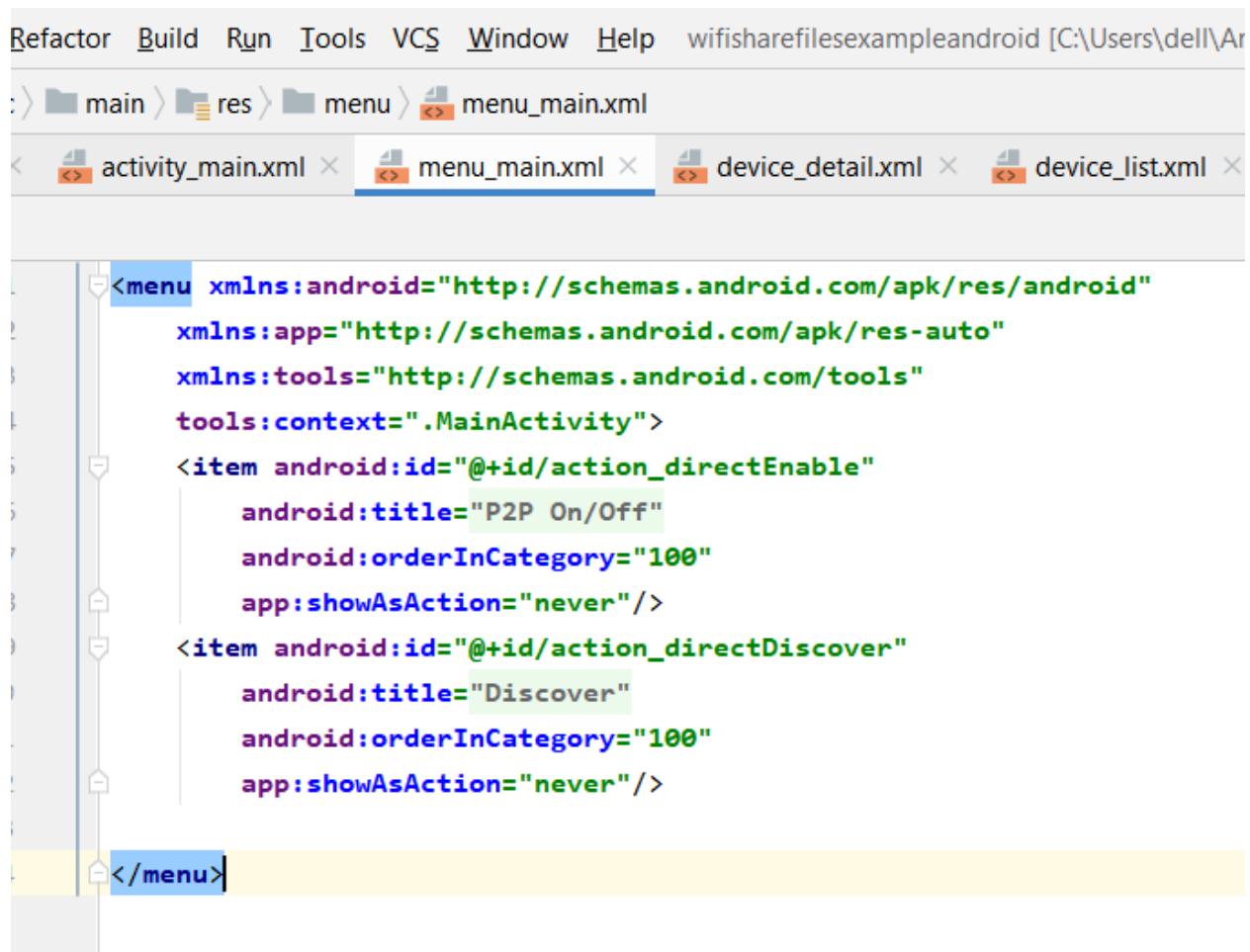
The screenshot shows the Android Studio interface with the XML code for `row_devices.xml`. The code defines a list item layout with a `LinearLayout` containing an `ImageView` and two `TextView`s. The `ImageView` has a margin right of 2dp and a source of `@mipmap/ic_launcher`. The first `TextView` displays the device name, and the second displays device details, both using a `marquee` effect.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:background="?android:attr/activatedBackgroundIndicator"
    android:padding="6dip">
    <ImageView
        android:id="@+id/iv_Icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_marginRight="2dip"
        android:src="@mipmap/ic_launcher" />
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="0dip"
        android:layout_weight="1"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/tv_deviceName"
            android:layout_width="fill_parent"
            android:layout_height="0dip"
            android:layout_weight="1"
            android:gravity="center_vertical" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="0dip"
            android:layout_weight="1"
            android:id="@+id/tv_deviceDetails"
            android:maxLines="1"
            android:ellipsize="marquee" />
    </LinearLayout>
</LinearLayout>
```

5. menu_main.xml

Create this XML file inside res/menu folder. Here we add the two action items.

Screenshot



```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">
    <item android:id="@+id/action_directEnable"
        android:title="P2P On/Off"
        android:orderInCategory="100"
        app:showAsAction="never"/>
    <item android:id="@+id/action_directDiscover"
        android:title="Discover"
        android:orderInCategory="100"
        app:showAsAction="never"/>
</menu>

```

Java Files

WifiDirectBroadcastReceiver.java

Create this class and extends with BroadcastReceiver. Here, this class notifies important Wi-Fi P2p events. The events are discussed above.

Initialize

```

private WifiP2pManager manager;
private Channel channel;
private MainActivity activity;

```

Add the Code in onReceive() Method:

```

    import android.net.wifi.p2p.WifiP2pManager;
    import android.net.wifi.p2p.WifiP2pManager.Channel;
    import android.net.wifi.p2p.WifiP2pManager.PeerListListener;

    public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {
        private WifiP2pManager manager;
        private Channel channel;
        private MainActivity activity;
        public WiFiDirectBroadcastReceiver(WifiP2pManager manager, Channel channel,
                                           MainActivity activity) {
            super();
            this.manager = manager;
            this.channel = channel;
            this.activity = activity;
        }
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
                int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
                if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
                    activity.setIsWifiP2pEnabled(true);
                } else {
                    activity.setIsWifiP2pEnabled(false);
                    activity.resetData();
                }
            } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
                if (manager != null) {
                    manager.requestPeers(channel, (PeerListListener) activity.getFragmentManager()
                        .findFragmentById(R.id.fragment_list));
                }
            } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
                if (manager == null) {
                    return;
                }
                NetworkInfo networkInfo = (NetworkInfo) intent
                    .getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);
                if (networkInfo.isConnected()) {
                    DeviceDetailFragment fragment = (DeviceDetailFragment) activity
                        .getFragmentManager().findFragmentById(R.id.fragment_detail);
                    manager.requestConnectionInfo(channel, fragment);
                } else {
                    activity.resetData();
                }
            } else if (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
                DeviceListFragment fragment = (DeviceListFragment) activity.getFragmentManager()
                    .findFragmentById(R.id.fragment_list);
                fragment.updateThisDevice((WifiP2pDevice) intent.getParcelableExtra(
                    WifiP2pManager.EXTRA_WIFI_P2P_DEVICE));
            }
        }
    }

```

MainActivity.java

This is our Main Activity. Here, it adds the necessary intent values by creating an intent-filter, registers the broadcast receiver, obtains the instance of WifiP2pManager class, registers the application by using initialize() method, also discover peers and connects with the available devices.

Initialize

```

public class MainActivity extends AppCompatActivity implements ChannelListener, DeviceActionListener {

    public static final String TAG = "WiFiShareFilesExampleAndroid";
    private WifiP2pManager manager;
    private boolean isWifiP2pEnabled = false;
    private boolean retryChannel = false;
    private final IntentFilter intentFilter = new IntentFilter();
    private Channel channel;
    private BroadcastReceiver receiver = null;
}

```

Add the code in onCreate(...){..}:

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // add necessary intent values to be matched.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

    manager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
    channel = manager.initialize(srcContext: this, getMainLooper(), listener: null);
}

```

Now, register the receiver in onResume() method

```

@Override
public void onResume() {
    super.onResume();
    receiver = new WiFiDirectBroadcastReceiver(manager, channel, activity: this);
    registerReceiver(receiver, intentFilter);
}

@Override
public void onPause() {
    super.onPause();
    unregisterReceiver(receiver);
}

```

Add the code for discover peers

```

final DeviceListFragment fragment = (DeviceListFragment) getFragmentManager()
    .findFragmentById(R.id.fragment_list);
fragment.onInitiateDiscovery();
manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {

    @Override
    public void onSuccess() {
        Toast.makeText(context: MainActivity.this, text: "Discovery Initiated",
            Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onFailure(int reasonCode) {
        Toast.makeText(context: MainActivity.this, text: "Discovery Failed : " + reasonCode,
            Toast.LENGTH_SHORT).show();
    }
});
return true;
default:
    return super.onOptionsItemSelected(item);
}
}

```

Now call connect() method for connecting with an available peer.



```

@Override
public void connect(WifiP2pConfig config) {
    manager.connect(channel, config, new ActionListener() {

        @Override
        public void onSuccess() {
            // WiFiDirectBroadcastReceiver will notify us. Ignore for now.
        }

        @Override
        public void onFailure(int reason) {
            Toast.makeText(context: MainActivity.this, text: "Connect failed. Retry.",
                Toast.LENGTH_SHORT).show();
        }
    });
}

```

DeviceDetailFragment.java

Create this fragment class. Here, it manages with a particular peer and “allows the interaction with the device” means when a connection is established successfully, we can transfer the data to the device by setting up a network connection.

Add the code here by creating a server socket that accepts the connection and writes the data.

```

1 package com.example.shriyanshu.wifisharefilesexampleandroid;
2 import android.app.ListFragment;
3 import android.app.ProgressDialog;
4 import android.content.Context;
5 import android.content.DialogInterface;
6 import android.net.wifi.p2p.WifiP2pDevice;
7 import android.net.wifi.p2p.WifiP2pDeviceList;
8 import android.net.wifi.p2p.WifiP2pManager.PeerListListener;
9 import android.os.Bundle;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.ViewGroup;
13 import android.widget.ArrayAdapter;
14 import android.widget.ListView;
15 import android.widget.TextView;
16 import java.util.ArrayList;
17 import java.util.List;
18
19 public class DeviceListFragment extends ListFragment implements PeerListListener {
20     private ArrayList<WifiP2pDevice> peers = new ArrayList<WifiP2pDevice>();
21     ProgressDialog progressDialog = null;
22     View mContentView = null;
23     private WifiP2pDevice device;
24     @Override
25     public void onActivityCreated(Bundle savedInstanceState) {
26         super.onActivityCreated(savedInstanceState);
27         this.setAdapter(new WiFiPeerListAdapter(getActivity()), R.layout.row_devices, peers);
28     }
29     @Override
30     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
31         mContentView = inflater.inflate(R.layout.device_list, null);
32         return mContentView;
33     }
34     public WiFiP2pDevice getDevice() {
35         return device;
36     }
37     private static String getDeviceStatus(int deviceStatus) {
38         switch (deviceStatus) {
39             case WiFiP2pDevice.AVAILABLE:
40                 return "Available";
41             case WiFiP2pDevice.INVITED:
42                 return "Invited";
43             case WiFiP2pDevice.CONNECTED:
44                 return "Connected";
45             case WiFiP2pDevice.FAILED:
46                 return "Failed";
47             case WiFiP2pDevice.UNAVAILABLE:
48                 return "Unavailable";
49             default:
50                 return "Unknown";
51         }
52     }
53     @Override
54     public void onListItemClick(ListView l, View v, int position, long id) {
55         WiFiP2pDevice device = (WiFiP2pDevice) getListAdapter().getItem(position);
56         ((DeviceClickListener) getActivity()).showDetails(device);
57     }
58     private class WiFiPeerListAdapter extends ArrayAdapter<WiFiP2pDevice> {
59         ...
60     }
61 }

```

Progress [70%]

Completed Parts of Project

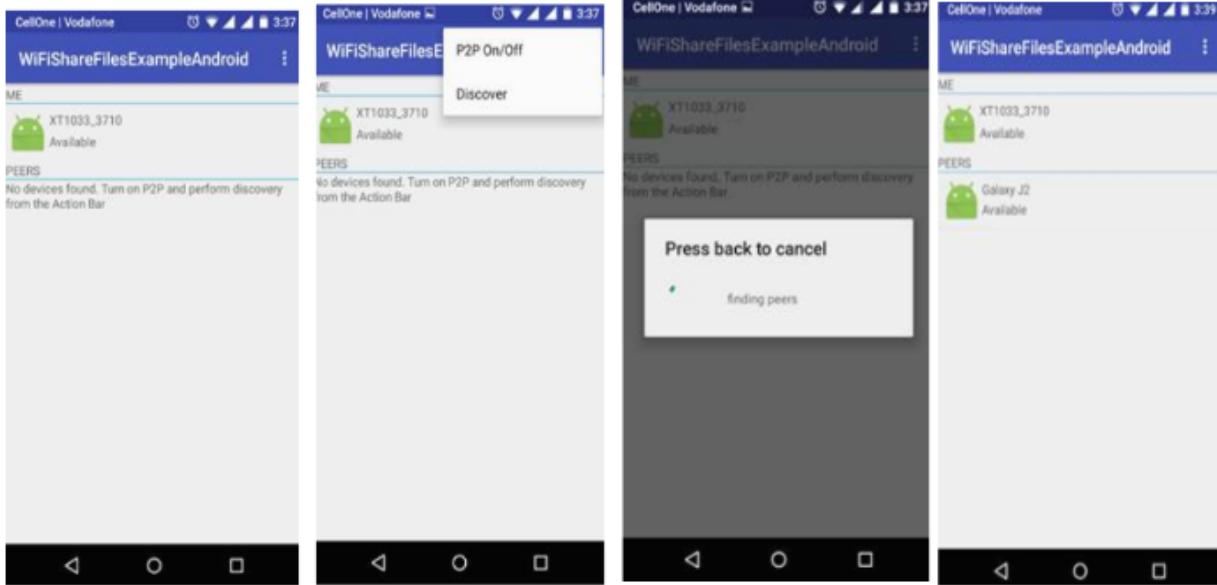
- 1- User Interface Of Easy Share is created.
- 2- Layout files are created.
- 3- Mostly Java files created.

Remaining parts of Project

- 1- Some Java files are left to create.
- 2- Sending & Receiving Threads for larger files.
- 3-Manipulation of Received files.

6.1 Screenshots

6.1.1 Screenshot Of User interface of Android Application.

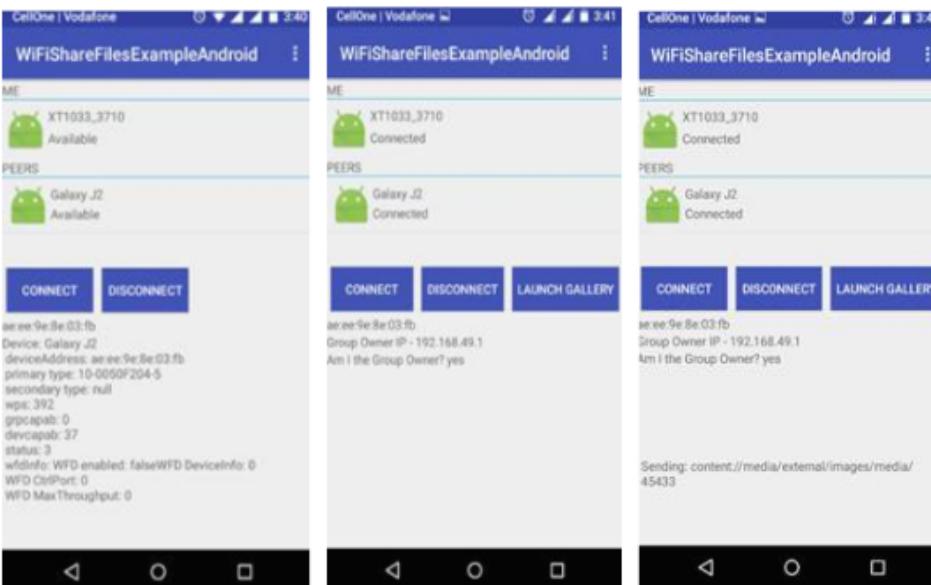


Before Discover

Perform Discover

Finding Peers

Peer Available



Connecting Device

Device Connected

Status after Sending the data successfully

6.1.2 Screenshots of Source code

```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:weightSum="1">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_alignParentTop="true"
        android:layout_weight="0.9"
        android:orientation="vertical">

        <fragment
            android:id="@+id/fragment_list"
            class="com.example.shriyanshu.wifisharefilesexampleandroid.DeviceListFragment"
            android:layout_width="match_parent"
            android:layout_height="@dimen/phone_list_height"
            tools:layout="@layout/device_list"></fragment>

        <fragment
            android:id="@+id/fragment_detail"
            class="com.example.shriyanshu.wifisharefilesexampleandroid.DeviceDetailFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:layout="@layout/device_detail"></fragment>
    </LinearLayout>
</LinearLayout>

```

The screenshot shows the Android Studio interface with the code editor open to the `activity_main.xml` file. The code defines a main layout as a vertical `LinearLayout` with a weight sum of 1. It contains another `LinearLayout` with a height of 0dp and an alignment of `alignParentTop`. This inner layout contains two `fragment` elements. The first fragment is identified by the ID `fragment_list` and is an instance of the `DeviceListFragment` class. The second fragment is identified by the ID `fragment_detail` and is an instance of the `DeviceDetailFragment` class. Both fragments have a layout width of "match_parent". The code editor shows syntax highlighting for XML attributes and Java code.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:visibility="gone">
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
<Button
    android:id="@+id/btn_connect"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="@string/connect_peer_button"
    android:background="@color/colorPrimary"
    android:layout_margin="5dp"
    android:textColor="#ffffffff"/>
<Button
    android:id="@+id/btn_disconnect"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="@string/disconnect_peer_button"
    android:background="@color/colorPrimary"
    android:layout_margin="5dp"
    android:textColor="#ffffffff"/>
<Button
    android:id="@+id/btn_close"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="@string/close_button"
    android:background="@color/colorPrimary"
    android:layout_margin="5dp"
    android:textColor="#ffffffff"/>

```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="3dp">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:text="NET" />

    <View
        android:layout_width="fill_parent"
        android:layout_height="1dp"
        android:gravity="center_vertical"
        android:background="#android:color/holo_blue_light" />

    <!-- Self informative -->
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="android:attr/listPreferredItemHeight"
        android:background="android:attr/activatedBackgroundIndicator"
        android:padding="3dp">
        <ImageView
            android:id="@+id/iv_icon"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_marginRight="2dp"
            android:src="@+mipmap/icon_launcher" />
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="2"
            android:layout_weight="1">
            <Textview
                android:id="@+id/tv_myname"
                android:layout_width="fill_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
                android:layout_gravity="center_vertical" />
        
    

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:background="?android:attr/activatedBackgroundIndicator"
    android:padding="6dp">
    <ImageView
        android:id="@+id/sv_Icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_marginRight="2dp"
        android:src="@mipmap/ic_launcher" />
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/tv_deviceName"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:gravity="center_vertical" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:id="@+id/tv_deviceDetails"
            android:maxLines="3"
            android:ellipsize="marquee" />
    </LinearLayout>
</LinearLayout>

```

```

package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Toast;

public class CommonMethods {

    public static String Tag = "WiFiShareFilesExampleAndroid";

    public static String getPath(Uri uri, Context context) {
        if (uri == null) {
            CommonMethods.e("", "uri is null");
            return null;
        }
        // this will only work for images selected from gallery
        String[] projection = { MediaStore.Images.Media.DATA };
        Cursor cursor = context.getContentResolver().query(uri, projection, null, null, null);
        CommonMethods.e("", "get path method->> after cursor init");
        if (cursor != null) {
            int column_index = cursor
                    .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
            cursor.moveToFirst();
            return cursor.getString(column_index);
        }
        CommonMethods.e("", "get path method->> after cursor");
        CommonMethods.e("", "get path method->> " + uri.getPath());
        return uri.getPath();
    }

    public static void downloadFileFromUri(Context context, String url) {
    }
}

```

The screenshot shows the Android Studio interface with the project 'exampleandroid2' open. The code editor displays the 'DeviceDetailFragment.java' file under the 'com.example.shriyanshu.wifishare' package. The code implements a Fragment that interacts with a WiFiManager to handle P2P connections. It includes imports for various Android classes and interfaces, as well as static fields for WiFi-related parameters.

```
1 package com.example.shriyanshu.wifisharefilesexampleandroid;
2
3 import android.app.Fragment;
4 import android.app.ProgressDialog;
5 import android.content.Context;
6 import android.content.Intent;
7 import android.net.Uri;
8 import android.net.wifi.WpsInfo;
9 import android.net.wifi.p2p.WifiP2pConfig;
10 import android.net.wifi.p2p.WifiP2pDevice;
11 import android.net.wifi.p2p.WifiP2pInfo;
12 import android.net.wifi.p2p.WifiP2pManager.ConnectionInfoListener;
13 import android.os.AsyncTask;
14 import android.os.Build;
15 import android.os.Bundle;
16 import android.os.Environment;
17 import android.os.Handler;
18 import android.util.Log;
19 import android.view.LayoutInflater;
20 import android.view.View;
21 import android.view.ViewGroup;
22 import android.widget.TextView;
23 import com.example.shriyanshu.wifisharefilesexampleandroid.DeviceListFragment.DeviceActionListener;
24 import java.io.File;
25 import java.io.FileOutputStream;
26 import java.io.IOException;
27 import java.io.InputStream;
28 import java.io.ObjectInputStream;
29 import java.io.OutputStream;
30 import java.net.ServerSocket;
31 import java.net.Socket;
32
33 import static com.example.shriyanshu.wifisharefilesexampleandroid.R.id.tv_deviceInfo;
34 public class DeviceDetailFragment extends Fragment implements ConnectionInfoListener {
35     protected static final int CHOOSE_FILE_RESULT_CODE = 20;
36     private View mContentView = null;
37     private WifiP2pDevice device;
38     private WifiP2pInfo info;
39     ProgressDialog progressDialog = null;
40     private static ProgressDialog mProgressDialog;
41     public static String WIFIserverip = "";
42     public static String WIFIclientip = "";
43
44     @Override
45     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
46         mContentView = inflater.inflate(R.layout.device_detail, container, false);
47         return mContentView;
48     }
49
50     @Override
51     public void onConnectionInfoAvailable(WifiP2pInfo info) {
52         Log.d("DeviceDetailFragment", "onConnectionInfoAvailable");
53         this.info = info;
54         TextView tvDeviceInfo = (TextView) mContentView.findViewById(tv_deviceInfo);
55         tvDeviceInfo.setText("Device Info: " + info.toString());
56     }
57
58     @Override
59     public void onDisconnect() {
60         Log.d("DeviceDetailFragment", "onDisconnect");
61     }
62
63     @Override
64     public void onError(int error) {
65         Log.d("DeviceDetailFragment", "onError");
66     }
67
68     @Override
69     public void onJoinFailure(WifiP2pDevice device, int reason) {
70         Log.d("DeviceDetailFragment", "onJoinFailure");
71     }
72
73     @Override
74     public void onLost() {
75         Log.d("DeviceDetailFragment", "onLost");
76     }
77
78     @Override
79     public void onStartSuccess(WifiP2pDevice device) {
80         Log.d("DeviceDetailFragment", "onStartSuccess");
81     }
82
83     @Override
84     public void onStopSuccess(WifiP2pDevice device) {
85         Log.d("DeviceDetailFragment", "onStopSuccess");
86     }
87 }
```

The screenshot shows the Android Studio interface with the DeviceDetailFragment.java file open in the main editor. The code implements a fragment for displaying device details, including a progress dialog for connecting to a WiFi peer. The code uses various Java classes like GlobalApplication, SharedPreferencesHandler, and WiFiClientP2PService. The editor includes syntax highlighting, code completion, and navigation tools.

```
private void wifip2p_connect() {
    ProgressDialog progressDialog = null;
    private static ProgressDialog mProgressDialog;
    public static String WiFiServerIp = "";
    public static String WiFiClientIp = "";
    static Boolean ClientCheck = false;
    public static String GroupOwnerAddress = "";
    static long ActualFileLength = 0;
    static int Percentage = 0;
    public static String FolderName = "WiFiShareFilesExampleAndroid";

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        setContentView(inflater.inflate(R.layout.device_detail, root, false));
        mContentView.findViewById(R.id.btn_connect).setOnClickListener(v) -> {
            WifiP2pConfig config = new WifiP2pConfig();
            config.deviceAddress = device.deviceAddress;
            config.wps.setup = WpsInfo.PBC;
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
            progressDialog = ProgressDialog.show(getActivity(), null, "Press back to cancel", true, false, "Connecting to :" + device.deviceAddress, true, true);
        };
        ((DeviceActionListener) getActivity()).connect(config);
    }

    mContentView.findViewById(R.id.btn_disconnect).setOnClickListener(v) -> {
        ((DeviceActionListener) getActivity()).disconnect();
    };

    mContentView.findViewById(R.id.btn_start_client).setOnClickListener(v) -> {
        // Start client logic here
    };
}
```

The screenshot shows the Android Studio interface with the DeviceListFragment.java file open in the main editor. The code implements a ListFragment that lists WiFiP2pDevice objects. It includes methods for creating the fragment, setting up the adapter, and handling item clicks. The code uses various Android libraries like Context, LayoutInflater, and ArrayAdapter.

```
package com.example.shriyanshu.wifisharefilesexampleandroid;
import android.app.ListFragment;
import android.content.Context;
import android.content.DialogInterface;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import android.net.wifi.p2p.WifiP2pManager.PeerListListener;
import android.net.wifi.p2p.WifiP2pManager.ActionListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.util.ArrayList;
import java.util.List;
```

```
public class DeviceListFragment extends ListFragment implements PeerListListener {
    private List<WifiP2pDevice> peers = new ArrayList<WifiP2pDevice>();
    private WifiP2pManager.ActionListener peerListListener = null;
    View mContentview = null;
    private WifiP2pDevice device;
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        this.setListAdapter(new WiFiPeerListAdapter(getActivity(), R.layout.row_devices, peers));
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        mContentview = inflater.inflate(R.layout.device_list, container, false);
        return mContentview;
    }
    public WiFiPeerDevice getDevice() {
        return device;
    }
    private static String getDeviceStatus(int deviceStatus) {
        switch (deviceStatus) {
            case WiFiPeerDevice.AVAILABLE:
                return "Available";
            case WiFiPeerDevice.CONNECTED:
                return "Connected";
            case WiFiPeerDevice.FAILED:
                return "Failed";
            case WiFiPeerDevice.UNKNOWN:
                return "Unknown";
            default:
                return "Unknown";
        }
    }
    @Override
    public void onListitemClick(ListView l, View v, int position, long id) {
        WiFiPeerDevice device = (WiFiPeerDevice) getListAdapter().getItem(position);
        (DeviceActionListener) getActivity().shudhutlist(device);
    }
}
```

```
private class WiFiPeerListAdapter extends ArrayAdapter<WiFiPeerDevice> {
    public WiFiPeerListAdapter(Context context, int resource, List<WiFiPeerDevice> objects) {
        super(context, resource, objects);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if (convertView == null) {
            convertView = LayoutInflater.from(parent.getContext()).inflate(R.layout.listitem, parent, false);
            holder = new ViewHolder(convertView);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        WiFiPeerDevice device = getItem(position);
        holder.name.setText(device.getDeviceName());
        holder.ipAddress.setText(device.getIpAddress());
        return convertView;
    }
    static class ViewHolder {
        TextView name;
        TextView ipAddress;
        public ViewHolder(View view) {
            name = (TextView) view.findViewById(R.id.device_name);
            ipAddress = (TextView) view.findViewById(R.id.device_ip_address);
        }
    }
}
```

The screenshot shows the Android Studio interface with the project 'wifisharefilesexampleandroid' open. The left sidebar displays the project structure under 'app'. The 'src/main/java/com/example/shriyanshu/wifisharefilesexampleandroid' package contains several Java files: DeviceListFragment, FileTransferService, GlobalApplication, MainActivity, SharedPreferencesHandler, WiFiClientPTService, WiFiDirectBroadcastReceiver, and WiFiTransferModal. Below these are generated files and a BuildConfig file. The 'res' folder contains drawable, layout, and menu resources. The 'gradle Scripts' section shows build.gradle and gradle-wrapper.properties files. At the bottom, there are tabs for 'Event Log', 'Layout Inspector', 'Build', 'Logcat', and 'Terminal'. The status bar at the bottom right indicates '16:24 CRLF UTT-B 4 spaces'.

```
1 package com.example.shriyanshu.wifisharefilesexampleandroid;
2
3 import android.app.IntentService;
4 import android.content.ContentResolver;
5 import android.content.Context;
6 import android.content.Intent;
7 import android.net.Uri;
8 import android.os.Handler;
9 import android.widget.Toast;
10 import java.io.FileNotFoundException;
11 import java.io.IOException;
12 import java.io.InputStream;
13 import java.io.ObjectOutputStream;
14 import java.io.OutputStream;
15 import java.net.InetSocketAddress;
16 import java.net.Socket;
17
18 public class FileTransferService extends IntentService {
19
20     Handler mHandler;
21
22     public static final int SOCKET_TIMEOUT = 5000;
23     public static final String ACTION_SEND_FILE = "com.example.shriyanshu.wifisharefilesexampleandroid.SEND_FILE";
24     public static final String EXTRAS_FILE_PATH = "file_url";
25     public static final String EXTRAS_GROUP_OWNER_ADDRESS = "go_host";
26     public static final String EXTRAS_GROUP_OWNER_PORT = "go_port";
27
28     public static int PORT = 8888;
29     public static final String inetaddress = "inetaddress";
30     public static final int ByteSize = 512;
31     public static final String Extension = "extension";
32     public static final String Filelength = "filelength";
33
34     public FileTransferService(String name) {
35         super(name);
36     }
37
38     public FileTransferService() {
39         super("NAME", "FileTransferService");
40     }
41
42     @Override
43     public void onCreate() {
44         // TODO Auto-generated method stub
45         super.onCreate();
46         mHandler = new Handler();
47     }
48 }
```

The screenshot shows the Android Studio interface with the project 'filesexampleandroid2' open. The GlobalApplication.java file is the active editor. The code defines a static context variable and overrides the onCreate() method to set it to the application's context. It also provides a static method to get this global context.

```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.app.Application;
import android.content.Context;

public class GlobalApplication extends Application {
    private static Context GlobalContext;

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        if(GlobalApplication.GlobalContext == null){
            GlobalApplication.GlobalContext = getApplicationContext();
        }
    }

    public static Context getGlobalAppContext() {
        return GlobalApplication.GlobalContext;
    }
}
```

The project structure on the left shows the following files and folders:

- app:
 - sampledata
 - manifests
 - java:
 - com.example.shriyanshu.wifisharefilesexampleandroid:
 - CommonMethods
 - DeviceDetailFragment
 - DeviceListFragment
 - FileTransferService
 - GlobalApplication
 - MainActivity
 - SharedPreferencesHandler
 - WiFiClientPTransferService
 - WiFiDirectBroadcastReceiver
 - WiFiTransferModal
 - com.example.shriyanshu.wifisharefilesexampleandroid:
 - ExampleInstrumentedTest
 - generated
 - com.example.shriyanshu.wifisharefilesexampleandroid
 - BuildConfig
 - res:
 - drawable
 - layout
 - activity_main.xml
 - device_detail.xml
 - device_list.xml
 - row_devices.xml
 - menu
 - menu_main.xml
 - mipmap
 - values
 - Gradle Scripts
 - build.gradle (Project: wifisharefilesexampleandroid)
 - build.gradle (Module: app)
 - gradle-wrapper.properties
 - gradle-wrapper.properties (Gradle Version)
 - proguard-rules.pro

At the bottom, there are tabs for Todo, Build, Logcat, and Terminal, along with a note about selected configuration with coverage enabled. The status bar indicates 212 CRLF, UTF-8, 4 spaces, and a few other icons.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "app".
- Code Editor:** The main window displays the `SharedPreferencesHandler.java` file. The code implements a static method to get shared preferences and another to set string values.
- Toolbars and Status Bar:** The top bar includes standard Android Studio icons like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help, and a sync icon. The status bar at the bottom shows "Gradle sync finished in 16 s 966 ms (from cached state) (vesterday 16:43)".
- Bottom Navigation:** Icons for Todo, Issues, Build, Logcat, and Terminal are visible.

```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.preference.PreferenceManager;

public class SharedPreferencesHandler {

    public static SharedPreferences getSharedPreferences(Context ctx) {
        return PreferenceManager.getDefaultSharedPreferences(ctx);
    }

    public static void setStringValues(Context ctx, String key,
                                      String DataToSave) {
        Editor editor = getSharedPreferences(ctx).edit();
        editor.putString(key, DataToSave);
        editor.commit();
    }

    public static String getStringValues(Context ctx, String key) {
        return getSharedPreferences(ctx).getString(key, null);
    }
}
```

```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.app.IntentService;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;

public class WiFiClientIPTransferService extends IntentService {
    public WiFiClientIPTransferService(String name) {
        super(name);
        // TODO Auto-generated constructor stub
    }
    public WiFiClientIPTransferService() {
        super("WiFiClientIPTransferService");
    }
    Handler mHandler;
    @Override
    protected void onHandleIntent(Intent intent) {
        Context context = GlobalApplication.getGlobalLAppContext();
        if (intent.getAction().equals(FileTransferService.ACTION_SEND_FILE)) {
            String host = intent.getExtras().getString(FileTransferService.EXTRAS_GROUP_OWNER_ADDRESS);
            String InetAddress = intent.getExtras().getString(FileTransferService.inetAddress);
            CommonMethods.e.tag("Localip Received while first connect", msg: "host address"+host);

            Socket socket = new Socket();
            int port = intent.getIntExtras(FileTransferService.EXTRAS_GROUP_OWNER_PORT);

            try {
                
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `WiFiDirectBroadcastReceiver.java` file, which extends `BroadcastReceiver`. It handles `WIFI_P2P_STATE_CHANGED_ACTION` and `WIFI_P2P_PEERS_CHANGED_ACTION` intents to manage WiFi P2P state and peers respectively. The code uses `WifiP2pManager` and `WifiP2pDevice` from the `android.net.wifi.p2p` package.

```
package com.example.shriyanshu.wifisharefilesexampleandroid;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.NetworkInfo;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import android.net.wifi.p2p.WifiP2pManager.Channel;
import android.net.wifi.p2p.WifiP2pManager.PeerListListener;

public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {
    private WifiP2pManager manager;
    private Channel channel;
    private MainActivity activity;
    public WiFiDirectBroadcastReceiver(WifiP2pManager manager, Channel channel,
                                       MainActivity activity) {
        super();
        this.manager = manager;
        this.channel = channel;
        this.activity = activity;
    }
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
            int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
            if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
                activity.setsWifiP2pEnabled(true);
            } else {
                activity.setsWifiP2pEnabled(false);
                activity.resetData();
            }
        } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
            if (manager != null) {
                manager.getPeers(new PeerListListener() {
                    @Override
                    public void onPeersFound(List<WifiP2pDevice> devices, int error) {
                        // Handle peer list
                    }
                });
            }
        }
    }
}
```

The screenshot shows the Android Studio interface with the project 'wifisharefilesexampleandroid' open. The left sidebar displays the project structure, including the app module with its Java files and resources. The main editor window shows the code for WiFiTransferModal.java, which implements Serializable and contains methods for setting and getting file name, file length, and network address.

```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import java.io.Serializable;

public class WiFiTransferModal implements Serializable {

    private String FileName;
    private Long Filelength;
    private String InetAddress;

    public WiFiTransferModal(){

    }

    public WiFiTransferModal(String inetaddress){
        this.InetAddress = inetaddress;
    }

    public WiFiTransferModal(String name, Long filelength){
        this.FileName = name;
        this.Filelength = filelength;
    }

    public String getInetAddress() {
        return InetAddress;
    }

    public void setInetAddress(String inetAddress) {
        InetAddress = inetAddress;
    }

    public Long getFileLength() {
        return Filelength;
    }
}
```

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "app".
- Code Editor:** The main area displays the content of `ExampleInstrumentedTest.java`. The code is as follows:

```
1 package com.example.shriyanshu.wifisharefilesexampleandroid;
2
3 import android.content.Context;
4
5 import androidx.test.platform.app.InstrumentationRegistry;
6 import androidx.test.ext.junit.runners.AndroidJUnit4;
7
8 import org.junit.Test;
9 import org.junit.runner.RunWith;
10
11 import static org.junit.Assert.*;
12
13 /**
14 * Instrumented test, which will execute on an Android device.
15 *
16 * Testing documentation
17 */
18 @RunWith(AndroidJUnit4.class)
19 public class ExampleInstrumentedTest {
20     @Test
21     public void useApplicationContext() {
22         // Context of the app under test.
23         Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
24         assertEquals("com.example.shriyanshu.wifisharefilesexampleandroid", appContext.getPackageName());
25     }
26 }
```

- Toolbars and Status Bar:** The top bar includes standard file operations like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help. The status bar at the bottom right shows "19:14 CRLF UTF-8 4 spaces".
- Bottom Navigation:** Icons for Todo, Issues, Logcat, Layout Inspector, Event Log, and Terminal.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "sharefilesexampleandroid2". The `MainActivity.java` file is open in the editor.
- Editor:** The code for `MainActivity.java` is displayed. It imports various Android components like `BroadcastReceiver`, `Context`, `Intent`, `IntentFilter`, and `WifiP2pManager`. It also imports `AppCompatActivity` from `appcompat`. The class `MainActivity` extends `AppCompatActivity` and implements `ChannelListener` and `ActionListener`. It contains methods for setting WiFi P2P enabled status and handling device lists.
- Toolbars:** Standard Android Studio toolbars for File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help, and various tabs for the current file.
- Bottom Bar:** Shows tabs for Build, Logcat, and Terminal, along with build variants and favorite files.
- Status Bar:** Displays the message "Gradle sync finished in 16 s 966 ms (from cached state) (yesterday 16:43)" and the time "22:103 CRLF UTR-B 4 spaces".

```

    public void resetData() {
        DeviceListFragment fragmentList = (DeviceListFragment) getSupportFragmentManager()
            .findFragmentById(R.id.fragment_list);
        DeviceDetailFragment fragmentDetails = (DeviceDetailFragment) getSupportFragmentManager()
            .findFragmentById(R.id.fragment_detail);
        if (fragmentList != null) {
            fragmentList.clearPeers();
        }
        if (fragmentDetails != null) {
            fragmentDetails.resetViews();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_directEnable:
                if (manager != null & channel != null) {

                    // Since this is the system wireless settings activity, it's
                    // not going to send us a result. We will be notified by
                    // WiFiDeviceBroadcastReceiver instead.

                    startActivityForResult(new Intent(Settings.ACTION_WIRELESS_SETTINGS));
                }
                return true;
        }
    }
}

```

```

    public void resetData() {
        DeviceListFragment fragmentList = (DeviceListFragment) getSupportFragmentManager()
            .findFragmentById(R.id.fragment_list);
        DeviceDetailFragment fragmentDetails = (DeviceDetailFragment) getSupportFragmentManager()
            .findFragmentById(R.id.fragment_detail);
        if (fragmentList != null) {
            fragmentList.clearPeers();
        }
        if (fragmentDetails != null) {
            fragmentDetails.resetViews();
        }
    }

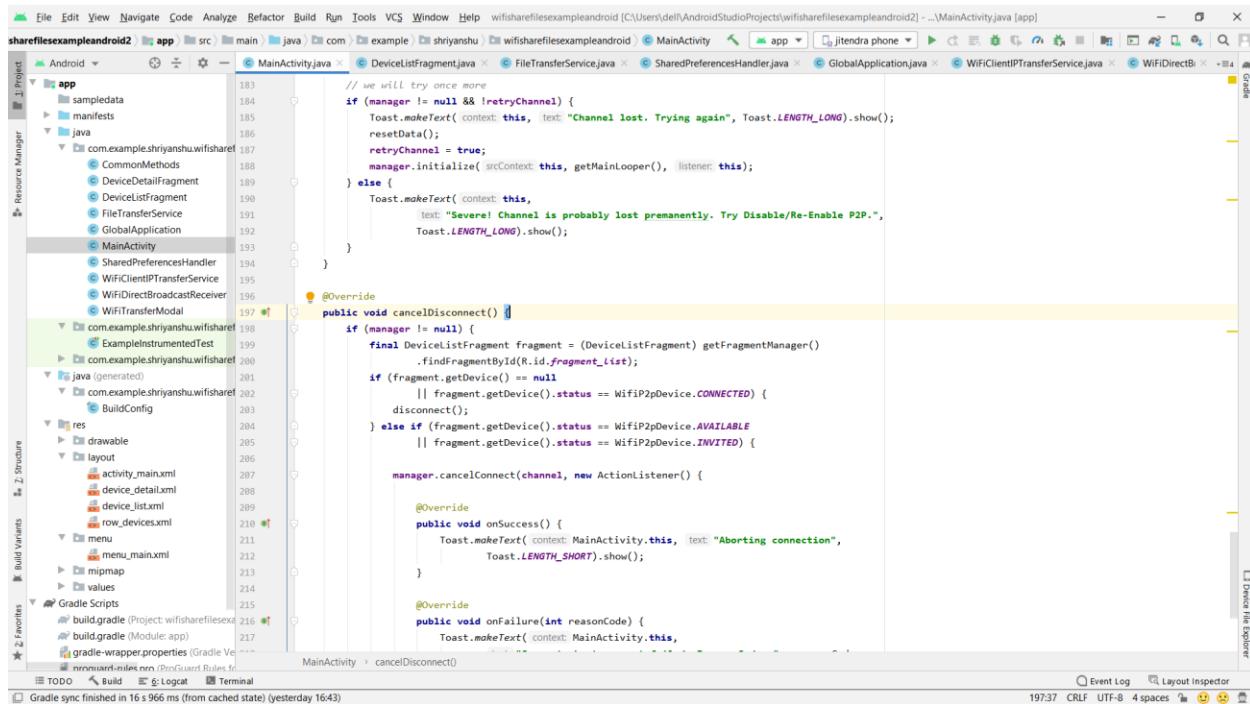
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_directDiscover:
                if (!isWifiP2pEnabled) {
                    Toast.makeText(context, "Enable P2P from action bar button above or system settings",
                        Toast.LENGTH_SHORT).show();
                    return true;
                }
                final DeviceListFragment fragment = (DeviceListFragment) getSupportFragmentManager()
                    .findFragmentById(R.id.fragment_list);
                fragment.onInitiateDiscovery();
                manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {

                    @Override
                    public void onSuccess() {
                        Toast.makeText(context, "Discovery Initiated",
                            Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onFailure(int reasonCode) {
                        Toast.makeText(context, "Discovery Failed : " + reasonCode,
                            Toast.LENGTH_SHORT).show();
                    }
                });
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}

```



```

183 // we will try once more
184 if (manager != null && !retryChannel) {
185     Toast.makeText(context: this, text: "Channel lost. Trying again", Toast.LENGTH_LONG).show();
186     reloadData();
187     retryChannel = true;
188     manager.initialize(getApplicationContext(), getMainLooper(), listener: this);
189 } else {
190     Toast.makeText(context: this,
191                 text: "Several Channel is probably lost permanently. Try Disable/Re-Enable P2P.",
192                 Toast.LENGTH_LONG).show();
193 }
194
195 @Override
196 public void cancelDisconnect() {
197     if (manager != null) {
198         final DeviceListFragment fragment = (DeviceListFragment) getSupportFragmentManager()
199             .findFragmentById(R.id.fragment_list);
200         if (fragment.getDevice() == null
201             || fragment.getDevice().status == WiFiP2pDevice.CONNECTED) {
202             disconnect();
203         } else if (fragment.getDevice().status == WiFiP2pDevice.AVAILABLE
204             || fragment.getDevice().status == WiFiP2pDevice.INVITED) {
205             manager.cancelConnect(channel, new ActionListener() {
206
207                 @Override
208                 public void onSuccess() {
209                     Toast.makeText(context: MainActivity.this, text: "Aborting connection",
210                         Toast.LENGTH_SHORT).show();
211                 }
212
213                 @Override
214                 public void onFailure(int reasonCode) {
215                     Toast.makeText(context: MainActivity.this,
216                         text: "Connection failed with error code: $reasonCode", Toast.LENGTH_SHORT).show();
217                 }
218             });
219         }
220     }
221 }
222
223 @Override
224 public void onFileTransferSuccess(FileTransferResult result) {
225     if (result != null) {
226         String filePath = result.filePath;
227         String fileName = result.fileName;
228         String fileExt = result.fileExt;
229         String fileSize = result.size;
230         String fileHash = result.hash;
231         String fileMD5 = result.md5;
232         String fileSHA1 = result.sha1;
233         String fileSHA256 = result.sha256;
234         String fileSHA512 = result.sha512;
235         String fileSHA384 = result.sha384;
236         String fileSHA224 = result.sha224;
237         String fileSHA160 = result.sha160;
238         String fileSHA128 = result.sha128;
239         String fileSHA1024 = result.sha1024;
240         String fileSHA512_224 = result.sha512_224;
241         String fileSHA512_256 = result.sha512_256;
242         String fileSHA512_384 = result.sha512_384;
243         String fileSHA512_512 = result.sha512_512;
244         String fileSHA512_1024 = result.sha512_1024;
245         String fileSHA512_224_512 = result.sha512_224_512;
246         String fileSHA512_256_512 = result.sha512_256_512;
247         String fileSHA512_384_512 = result.sha512_384_512;
248         String fileSHA512_512_512 = result.sha512_512_512;
249         String fileSHA512_1024_512 = result.sha512_1024_512;
250         String fileSHA512_224_1024 = result.sha512_224_1024;
251         String fileSHA512_256_1024 = result.sha512_256_1024;
252         String fileSHA512_384_1024 = result.sha512_384_1024;
253         String fileSHA512_512_1024 = result.sha512_512_1024;
254         String fileSHA512_1024_1024 = result.sha512_1024_1024;
255         String fileSHA512_224_512_512 = result.sha512_224_512_512;
256         String fileSHA512_256_512_512 = result.sha512_256_512_512;
257         String fileSHA512_384_512_512 = result.sha512_384_512_512;
258         String fileSHA512_512_512_512 = result.sha512_512_512_512;
259         String fileSHA512_1024_512_512 = result.sha512_1024_512_512;
260         String fileSHA512_224_1024_512 = result.sha512_224_1024_512;
261         String fileSHA512_256_1024_512 = result.sha512_256_1024_512;
262         String fileSHA512_384_1024_512 = result.sha512_384_1024_512;
263         String fileSHA512_512_1024_512 = result.sha512_512_1024_512;
264         String fileSHA512_1024_1024_512 = result.sha512_1024_1024_512;
265         String fileSHA512_224_512_512_512 = result.sha512_224_512_512_512;
266         String fileSHA512_256_512_512_512 = result.sha512_256_512_512_512;
267         String fileSHA512_384_512_512_512 = result.sha512_384_512_512_512;
268         String fileSHA512_512_512_512_512 = result.sha512_512_512_512_512;
269         String fileSHA512_1024_512_512_512 = result.sha512_1024_512_512_512;
270         String fileSHA512_224_1024_512_512 = result.sha512_224_1024_512_512;
271         String fileSHA512_256_1024_512_512 = result.sha512_256_1024_512_512;
272         String fileSHA512_384_1024_512_512 = result.sha512_384_1024_512_512;
273         String fileSHA512_512_1024_512_512 = result.sha512_512_1024_512_512;
274         String fileSHA512_1024_1024_512_512 = result.sha512_1024_1024_512_512;
275         String fileSHA512_224_512_512_512_512 = result.sha512_224_512_512_512_512;
276         String fileSHA512_256_512_512_512_512 = result.sha512_256_512_512_512_512;
277         String fileSHA512_384_512_512_512_512 = result.sha512_384_512_512_512_512;
278         String fileSHA512_512_512_512_512_512 = result.sha512_512_512_512_512_512;
279         String fileSHA512_1024_512_512_512_512 = result.sha512_1024_512_512_512_512;
280         String fileSHA512_224_1024_512_512_512 = result.sha512_224_1024_512_512_512;
281         String fileSHA512_256_1024_512_512_512 = result.sha512_256_1024_512_512_512;
282         String fileSHA512_384_1024_512_512_512 = result.sha512_384_1024_512_512_512;
283         String fileSHA512_512_1024_512_512_512 = result.sha512_512_1024_512_512_512;
284         String fileSHA512_1024_1024_512_512_512 = result.sha512_1024_1024_512_512_512;
285         String fileSHA512_224_512_512_512_512_512 = result.sha512_224_512_512_512_512_512;
286         String fileSHA512_256_512_512_512_512_512 = result.sha512_256_512_512_512_512_512;
287         String fileSHA512_384_512_512_512_512_512 = result.sha512_384_512_512_512_512_512;
288         String fileSHA512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512;
289         String fileSHA512_1024_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512;
290         String fileSHA512_224_1024_512_512_512_512 = result.sha512_224_1024_512_512_512_512;
291         String fileSHA512_256_1024_512_512_512_512 = result.sha512_256_1024_512_512_512_512;
292         String fileSHA512_384_1024_512_512_512_512 = result.sha512_384_1024_512_512_512_512;
293         String fileSHA512_512_1024_512_512_512_512 = result.sha512_512_1024_512_512_512_512;
294         String fileSHA512_1024_1024_512_512_512_512 = result.sha512_1024_1024_512_512_512_512;
295         String fileSHA512_224_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512;
296         String fileSHA512_256_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512;
297         String fileSHA512_384_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512;
298         String fileSHA512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512;
299         String fileSHA512_1024_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512;
300         String fileSHA512_224_1024_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512;
301         String fileSHA512_256_1024_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512;
302         String fileSHA512_384_1024_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512;
303         String fileSHA512_512_1024_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512;
304         String fileSHA512_1024_1024_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512;
305         String fileSHA512_224_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512;
306         String fileSHA512_256_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512;
307         String fileSHA512_384_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512;
308         String fileSHA512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512;
309         String fileSHA512_1024_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512;
310         String fileSHA512_224_1024_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512;
311         String fileSHA512_256_1024_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512;
312         String fileSHA512_384_1024_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512;
313         String fileSHA512_512_1024_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512;
314         String fileSHA512_1024_1024_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512;
315         String fileSHA512_224_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512;
316         String fileSHA512_256_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512;
317         String fileSHA512_384_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512;
318         String fileSHA512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512;
319         String fileSHA512_1024_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512;
320         String fileSHA512_224_1024_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512;
321         String fileSHA512_256_1024_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512;
322         String fileSHA512_384_1024_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512;
323         String fileSHA512_512_1024_512_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512_512;
324         String fileSHA512_1024_1024_512_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512_512;
325         String fileSHA512_224_512_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512_512;
326         String fileSHA512_256_512_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512_512;
327         String fileSHA512_384_512_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512_512;
328         String fileSHA512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512_512;
329         String fileSHA512_1024_512_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512_512;
330         String fileSHA512_224_1024_512_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512_512;
331         String fileSHA512_256_1024_512_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512_512;
332         String fileSHA512_384_1024_512_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512_512;
333         String fileSHA512_512_1024_512_512_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512_512_512;
334         String fileSHA512_1024_1024_512_512_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512_512_512;
335         String fileSHA512_224_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512_512_512;
336         String fileSHA512_256_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512_512_512;
337         String fileSHA512_384_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512_512_512;
338         String fileSHA512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512_512_512;
339         String fileSHA512_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512_512_512;
340         String fileSHA512_224_1024_512_512_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512_512_512;
341         String fileSHA512_256_1024_512_512_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512_512_512;
342         String fileSHA512_384_1024_512_512_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512_512_512;
343         String fileSHA512_512_1024_512_512_512_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512_512_512_512;
344         String fileSHA512_1024_1024_512_512_512_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512_512_512_512;
345         String fileSHA512_224_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512_512_512_512;
346         String fileSHA512_256_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512_512_512_512;
347         String fileSHA512_384_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512_512_512_512;
348         String fileSHA512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512_512_512_512;
349         String fileSHA512_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512_512_512_512;
350         String fileSHA512_224_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512_512_512_512;
351         String fileSHA512_256_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512_512_512_512;
352         String fileSHA512_384_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512_512_512_512;
353         String fileSHA512_512_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512_512_512_512_512;
354         String fileSHA512_1024_1024_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512_512_512_512_512;
355         String fileSHA512_224_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512_512_512_512_512;
356         String fileSHA512_256_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512_512_512_512_512;
357         String fileSHA512_384_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512_512_512_512_512;
358         String fileSHA512_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512_512_512_512_512;
359         String fileSHA512_1024_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512_512_512_512_512;
360         String fileSHA512_224_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512_512_512_512_512;
361         String fileSHA512_256_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512_512_512_512_512;
362         String fileSHA512_384_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512_512_512_512_512;
363         String fileSHA512_512_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_1024_512_512_512_512_512_512_512_512_512_512_512;
364         String fileSHA512_1024_1024_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_1024_512_512_512_512_512_512_512_512_512_512_512;
365         String fileSHA512_224_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_512_512_512_512_512_512_512_512_512_512_512_512_512;
366         String fileSHA512_256_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_512_512_512_512_512_512_512_512_512_512_512_512_512;
367         String fileSHA512_384_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_512_512_512_512_512_512_512_512_512_512_512_512_512;
368         String fileSHA512_512_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_512_512_512_512_512_512_512_512_512_512_512_512_512_512;
369         String fileSHA512_1024_512_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_1024_512_512_512_512_512_512_512_512_512_512_512_512_512;
370         String fileSHA512_224_1024_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_224_1024_512_512_512_512_512_512_512_512_512_512_512_512;
371         String fileSHA512_256_1024_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_256_1024_512_512_512_512_512_512_512_512_512_512_512_512;
372         String fileSHA512_384_1024_512_512_512_512_512_512_512_512_512_512_512_512 = result.sha512_384_1024_512_512_512_512_512_512_512_512_512_512_512_512;
373         String fileSHA512_512_1024_512_512_512_512_512_512_
```

c

References

- 1) <https://nevonprojects.com/secure-file-sharing-using-access-control/>
- 2) <https://freeprojectsforall.com/an046-secure-file-sharing-using-access-control/>
- 3) https://ijarcce.com/wp-content/uploads/2012/03/IJARCCE5G_a_shailes_secured.pdf