Andriod的联通性---Wi-Fi Direct（二）

2012-11-24 20:57 7443人阅读 评论(6) 收藏 举报

分类： 学习笔记（426）

创建Wi-Fi Direct应用程序

创建Wi-Fi Direct应用程序涉及到给应用程序创建和注册广播接收器、发现对等设备、连接对等设备和把数据传送给对等设备。下面会介绍如何完成这些事情。

初始安装

在使用Wi-Fi Direct API之前，必须确保你的应用程序能够访问硬件，并且该设备要支持Wi-Fi Direct协议。如果支持Wi-Fi Direct，你就可以获得一个WifiP2pManager实例，然后创建和注册你的广播接收器，开始使用Wi-Fi Direct API。

1. 在**Android**清单中申请使用设备上Wi-Fi硬件的权限，并声明要使用的最小的SDK版本：

<uses-sdkandroid:minSdkVersion="14"/>

<uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>

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

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

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

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

2. 检查是否支持Wi-Fi Direct。做这项检查的一个好的位置是，接收WIFI_P2P_STATE_CHANGED_ACTION类型的Intent的广播接收器中，把Wi-Fi Direct的状态通知给你的Activity，并作出相应的反应：

```
@Override
publicvoid 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){
            // Wifi Direct is enabled
        }else{
            // Wi-Fi Direct is not enabled
        }
    }
    ...
}
```

3. 在你的Activity的onCreate()方法中，获得一个WifiP2pManager的实例，并且要调用initialize()方法把你的应用程序注册到Wi-Fi Direct框架中。这个方法会返回一个WifiP2pManager.Channel对象，它用于把应用程序连接到Wi-Fi Direct框架。你还应该创建一个带有WifiP2pManager和WifiP2pManager.Channel对象以及你的Activity的引用的广播接收器实例。这样就允许你的广播接收器根据变化把你感兴趣的事件通知给你的Activity。如果需要，你还可以维护设备的Wi-Fi状态：

```
WifiP2pManager mManager;
Channel mChannel;
BroadcastReceiver mReceiver;
...
@Override
protectedvoid onCreate(Bundle savedInstanceState){
    ...
    mManager =(WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
    mChannel = mManager.initialize(this, getMainLooper(),null);
    mReceiver =newWiFiDirectBroadcastReceiver(manager, channel,this);
    ...
```

```
}
```

4. 创建一个Intent过滤器，并给这个Intent添加你的广播接收器要检查操作：

```
IntentFilter mIntentFilter;

...

@Override

protectedvoid onCreate(Bundle savedInstanceState){

    ...

    mIntentFilter =newIntentFilter();

    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);

    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);

    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);

    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

    ...

}
```

5. 在你的Activity的onResume()方法中注册广播接收器，并且要在你的Activity的onPause()方法注销它：

```
/* register the broadcast receiver with the intent values to be matched */

@Override

protectedvoid onResume(){

    super.onResume();

    registerReceiver(mReceiver, mIntentFilter);

}

/* unregister the broadcast receiver */
```

```
@Override

protectedvoid onPause(){

    super.onPause();

    unregisterReceiver(mReceiver);

}
```

当你已经获得了一个WifiP2pManager.Channel对象并建立一个广播接收器时，你的应用程序就能够调用Wi-Fi Direct方法和接收Wi-Fi Direct的Intent对象了。

现在，通过调用WifiP2pManager对象中的方法，你能够使用Wi-Fi Direct功能了。接下来向你介绍如何使用发现和连接对等设备等通用操作。

发现对等设备

要发现有效的可连接的对等设备，就要调用discoverPeers()方法，在一定的范围内检查有效的对等设备。这个功能调用是异步的，如果你创建了WifiP2pManager.ActionListener监听器，那么你的应用程序就会使用onSuccess()和onFailure()方法来完成成功或失败的传递。onSuccess()方法只会通知你，发现处理成功了，它并不提供发现的相关实际对等设备的任何信息：

```
manager.discoverPeers(channel,newWifiP2pManager.ActionListener(){

    @Override

    publicvoid onSuccess(){

        ...

    }



    @Override

    publicvoid onFailure(int reasonCode){

        ...

    }

});
```

如果发现处理成功，并检测到对等设备，系统会广播WIFI_P2P_PEERS_CHANGED_ACTION类型

的Intent，你能够在一个广播接收器中监听这个Intent，以便获得对等设备的列表。当你的应用程序接收到这个Intent时，你能够使用requestPeers()方法来请求被发现的对等设备的列表。下列代码显示了如何做这件事：

```
PeerListListener myPeerListListener;

...

if(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)){


    // request available peers from the wifi p2p manager. This is an

    // asynchronous call and the calling activity is notified with a

    // callback on PeerListListener.onPeersAvailable()

    if(manager !=null){

        manager.requestPeers(channel, myPeerListListener);

    }

}
```

requestPeers()方法也是异步的，有效的对等设备列表是使用onPeersAvailable()回调来通知你的Activity的，这个回调方法是在WifiP2pManager.PeerListListener接口中定义的。onPeersAvailable()方法会给你提供一个WifiP2pDeviceList对象，通过迭代该对象就能够找到你想要连接的对等设备。

连接对等设备

在获得可能的对等设备列表之后，并从中找到了你想要连接的设备时，就要调用connect()方法来连接设备。调用这个方法需要一个WifiP2pConfig对象，该对象包含了要连接的设备的信息。通过WifiP2pManager.ActionListener监听器，你能够获得连接成功或失败的通知。下列代码显示了如何创建跟期望的设备的连接：

```
//obtain a peer from the WifiP2pDeviceList

WifiP2pDevice device;

WifiP2pConfig config =newWifiP2pConfig();

config.deviceAddress = device.deviceAddress;
```

```
manager.connect(channel, config,newActionListener(){


    @Override

    publicvoid onSuccess(){

        //success logic

    }


    @Override

    publicvoid onFailure(int reason){

        //failure logic

    }

});
```

传输数据

一旦建立了连接，就能够使用套接字在设备之间传输数据。基本的步骤如下：

1. 创建一个ServerSocket对象。这个套接字会在指定的端口上等待来自客户端的连接，并且要一直阻塞到连接发生，因此要在后台线程中做这件事。

2. 创建一个客户端的Socket对象。该客户端要使用服务套接字的IP地址和端口来连接服务端设备。

3. 把数据从客户端发送给服务端。当客户端套接字跟服务端套接字成功的建立了连接，你就能够以字节流的形式，把数据从客户端发送给服务端了。

4. 服务套接字等待客户端的连接（用accept()方法）。这个调用会一直阻塞到客户端的连接发生，因此这个调用要放到另外一个线程中。当连接发生时，服务端能够接收来自客户端的数据。并对这个数据执行一些操作，如保存到文件或展现给用户。

下例来自Wi-Fi Direct Demo示例，向你展示了如何创建这种客户-服务套接字的通信，并从客户端把JPEG图片传输给服务端。完整的示例请编译和运行Wi-Fi Direct Demo示例：

```
public static class FileServerAsyncTask extends AsyncTask {
```

```
    private Context context;
    private TextView statusText;
    public FileServerAsyncTask(Context context, View statusText) {
        this.context = context;
        this.statusText = (TextView) statusText;
    }
    @Override
    protected String doInBackground(Void... params) {
        try {
            /**
             * Create a server socket and wait for client connections. This
             * call blocks until a connection is accepted from a client
             */
            ServerSocket serverSocket = new ServerSocket(8888);
            Socket client = serverSocket.accept();
            /**
             * If this code is reached, a client has connected and transferred data
             * Save the input stream from the client as a JPEG file
             */
            final File f = new File(Environment.getExternalStorageDirectory() + "/"
                    + context.getPackageName() + "/wifip2pshared-" +
System.currentTimeMillis()
                    + ".jpg");
            File dirs = new File(f.getParent());
```

```
            if (!dirs.exists())
                dirs.mkdirs();
            f.createNewFile();
            InputStream inputstream = client.getInputStream();
            copyFile(inputstream, new FileOutputStream(f));
            serverSocket.close();
            return f.getAbsolutePath();
        } catch (IOException e) {
            Log.e(WiFiDirectActivity.TAG, e.getMessage());
            return null;
        }
    }
    /**
     * Start activity that can handle the JPEG image
     */
    @Override
    protected void onPostExecute(String result) {
        if (result != null) {
            statusText.setText("File copied - " + result);
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            intent.setDataAndType(Uri.parse("file://" + result), "image/*");
            context.startActivity(intent);
        }
```

```
    }
}
```

在客户端，客户套接字连接到服务套接字，并传输数据。这个例子是把客户端设备的文件系统上的一个JPEG文件传输给服务端。

```
Context context =this.getApplicationContext();
String host;
int port;
int len;
Socket socket =newSocket();
byte buf[]  =newbyte[1024];
...
try{
    /**
     * Create a client socket with the host,
     * port, and timeout information.
     */
    socket.bind(null);
    socket.connect((newInetSocketAddress(host, port)),500);

    /**
     * Create a byte stream from a JPEG file and pipe it to the output stream
     * of the socket. This data will be retrieved by the server device.
     */
    OutputStream outputStream = socket.getOutputStream();
```

```
    ContentResolver cr = context.getContentResolver();
    InputStream inputStream =null;
    inputStream = cr.openInputStream(Uri.parse("path/to/picture.jpg"));
    while((len = inputStream.read(buf))!=-1){
        outputStream.write(buf,0, len);
    }
    outputStream.close();
    inputStream.close();
}catch(FileNotFoundException e){
    //catch logic
}catch(IOException e){
    //catch logic
}


/**
 * Clean up any open sockets when done
 * transferring or if an exception occurred.
 */
finally{
    if(socket !=null){
        if(socket.isConnected()){
            try{
                socket.close();
            }catch(IOException e){
```

```
            //catch logic

        }

    }

}
```