前言

本文是针对Android N开始的，还没有整理完全，只到StateMachine就结束了，待我搞清楚了WifiStateMachine的套路再进行更加详细的介绍

时序图

以下是我整理的时序图

代码分析

涉及的代码

1. com.android.settings.wifi.WifiSettings
2. android.net.wifi.WifiManager
3. com.android.internal.util.AsyncChannel
4. com.android.server.wifi.WifiServiceImpl
5. com.android.internal.util.StateMachine
6. com.android.server.wifi.WifiStateMachine

1. WifiSettings.java

```
protected void connect(final WifiConfiguration config, boolean isSavedNetwork) {
    // Log subtype if configuration is a saved network.
    MetricsLogger.action(getActivity(), MetricsEvent.ACTION_WIFI_CONNECT,
                         isSavedNetwork);
    //其他的没有进行分析是因为都是界面上的内容，没有啥好分析的
    mWifiManager.connect(config, mConnectListener);
}
```

2. WifiManager.java

```
//1.1
public void connect(WifiConfiguration config, ActionListener listener) {
    if (config == null) throw new IllegalArgumentException("config cannot be null");
    // Use INVALID_NETWORK_ID for arg1 when passing a config object
    // arg1 is used to pass network id when the network already exists
    //见1.2
    //sendMessage见3.7
    getChannel().sendMessage(CONNECT_NETWORK, WifiConfiguration.INVALID_NETWORK_ID,
                             putListener(listener), config);
}
//1.2
private synchronized AsyncChannel getChannel() {
    if (mAsyncChannel == null) {
      //获取目标的发送Messenger
      Messenger messenger = getWifiServiceMessenger();
      if (messenger == null) {
        throw new IllegalStateException(
          "getWifiServiceMessenger() returned null!  This is invalid.");
      }

      mAsyncChannel = new AsyncChannel();
      mConnected = new CountDownLatch(1);

      Handler handler = new ServiceHandler(mLooper);
      //见AsyncChannel的分析, 3.1
      mAsyncChannel.connect(mContext, handler, messenger);
      try {
        mConnected.await();
      } catch (InterruptedException e) {
        Log.e(TAG, "interrupted wait at init");
      }
    }
    return mAsyncChannel;
}

public Messenger getWifiServiceMessenger() {
    try {
      //IWifiManager mService;见WifiServiceImpl分析
      return mService.getWifiServiceMessenger();
    } catch (RemoteException e) {
      throw e.rethrowFromSystemServer();
    }
}

//WifiServiceImpl.java
public class WifiServiceImpl extends IWifiManager.Stub {
    private ClientHandler mClientHandler;
    public Messenger getWifiServiceMessenger() {
        enforceAccessPermission();
        enforceChangePermission();
        return new Messenger(mClientHandler);
    }
}
```

3. AsyncChannel.java

```
//3.1
public void connect(Context srcContext, Handler srcHandler, Messenger dstMessenger) {
    if (DBG) log("connect srcHandler to the dstMessenger  E");

    // We are connected
    connected(srcContext, srcHandler, dstMessenger);

    // Tell source we are half connected
    //回答已经连接成功,见3.3
    replyHalfConnected(STATUS_SUCCESSFUL);

    if (DBG) log("connect srcHandler to the dstMessenger X");
}

//3.2
public void connected(Context srcContext, Handler srcHandler, Messenger dstMessenger) {
    if (DBG) log("connected srcHandler to the dstMessenger  E");

    // Initialize source fields
    mSrcContext = srcContext;
    mSrcHandler = srcHandler;
    mSrcMessenger = new Messenger(mSrcHandler);

    // Initialize destination fields
    mDstMessenger = dstMessenger;
    linkToDeathMonitor();
    if (DBG) log("connected srcHandler to the dstMessenger X");
}
//3.3
private void replyHalfConnected(int status) {
    Message msg = mSrcHandler.obtainMessage(CMD_CHANNEL_HALF_CONNECTED);
    msg.arg1 = status;
    msg.obj = this;
    msg.replyTo = mDstMessenger;
    if (!linkToDeathMonitor()) {
      // Override status to indicate failure
      msg.arg1 = STATUS_BINDING_UNSUCCESSFUL;
    }
    mSrcHandler.sendMessage(msg);
}

//WifiManager.java ServiceHandler
private class ServiceHandler extends Handler {
    @Override
    public void handleMessage(Message message) {
      synchronized (sServiceHandlerDispatchLock) {
        dispatchMessageToListeners(message);
      }
    }

  private void dispatchMessageToListeners(Message message) {
    Object listener = removeListener(message.arg2);

    switch (message.what) {
```

```
        case AsyncChannel.CMD_CHANNEL_HALF_CONNECTED:
          if (message.arg1 == AsyncChannel.STATUS_SUCCESSFUL) {
            //又会新建一个连接，并且会把源和目标Messenger反过来
            //见3.4
            mAsyncChannel.sendMessage(AsyncChannel.CMD_CHANNEL_FULL_CONNECTION);
          } else {
            Log.e(TAG, "Failed to set up channel connection");
            // This will cause all further async API calls on the WifiManager
            // to fail and throw an exception
            mAsyncChannel = null;
          }
          mConnected.countDown();
          break;
        case WifiManager.CONNECT_NETWORK:
        case WifiManager.SAVE_NETWORK: {
          WifiConfiguration config = (WifiConfiguration) msg.obj;
          int networkId = msg.arg1;
          if (msg.what == WifiManager.SAVE_NETWORK) {
            Slog.d("WiFiServiceImpl ", "SAVE"
                   + " nid=" + Integer.toString(networkId)
                   + " uid=" + msg.sendingUid
                   + " name="
                   + mContext.getPackageManager().getNameForUid(msg.sendingUid));
          }
          if (msg.what == WifiManager.CONNECT_NETWORK) {
            Slog.d("WiFiServiceImpl ", "CONNECT "
                   + " nid=" + Integer.toString(networkId)
                   + " uid=" + msg.sendingUid
                   + " name="
                   + mContext.getPackageManager().getNameForUid(msg.sendingUid));
          }

          if (config != null && isValid(config)) {
            if (DBG) Slog.d(TAG, "Connect with config" + config);
            //config不为空会走着
            mWifiStateMachine.sendMessage(Message.obtain(msg));
          } else if (config == null
                     && networkId != WifiConfiguration.INVALID_NETWORK_ID) {
            if (DBG) Slog.d(TAG, "Connect with networkId" + networkId);
            //只有networkId会走这
            mWifiStateMachine.sendMessage(Message.obtain(msg));
          } else {
            Slog.e(TAG, "ClientHandler.handleMessage ignoring invalid msg=" + msg);
            if (msg.what == WifiManager.CONNECT_NETWORK) {
              replyFailed(msg, WifiManager.CONNECT_NETWORK_FAILED,
                          WifiManager.INVALID_ARGS);
            } else {
              replyFailed(msg, WifiManager.SAVE_NETWORK_FAILED,
                          WifiManager.INVALID_ARGS);
            }
          }
          break;

      }
```

```
        ...
    }
}
//3.4,回到了AsyncChannel.java中
public void sendMessage(int what) {
    Message msg = Message.obtain();
    msg.what = what;
    sendMessage(msg);
}
//3.5
public void sendMessage(Message msg) {
    //包含了WifiManager.ServiceHandler
    msg.replyTo = mSrcMessenger;
    try {
        //mDst包含了WifiServiceImpl.ClientHandler
        mDstMessenger.send(msg);
    } catch (RemoteException e) {
        replyDisconnected(STATUS_SEND_UNSUCCESSFUL);
    }
}
//3.6 WifiServiceImpl.ClientHandler
private class ClientHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        case AsyncChannel.CMD_CHANNEL_FULL_CONNECTION: {
            AsyncChannel ac = new AsyncChannel();
            //把Client和Service反过来了，但是好像形成了一个死循环。。然后在不断的发送连接成功的消息，直到连
接失败
            ac.connect(mContext, this, msg.replyTo);
            break;
    }
}

//3.7 连接网络的开始
    public void sendMessage(int what, int arg1, int arg2, Object obj) {
        Message msg = Message.obtain();
        msg.what = what;
        msg.arg1 = arg1;
        msg.arg2 = arg2;
        msg.obj = obj;
        //见3.5
        sendMessage(msg);
    }

}
```

4. StateMachine.java