SCIENCES SORBONNE UNIVERSITÉ

Peer2Peer

Renault@lrde.epita.fr

Main Goals

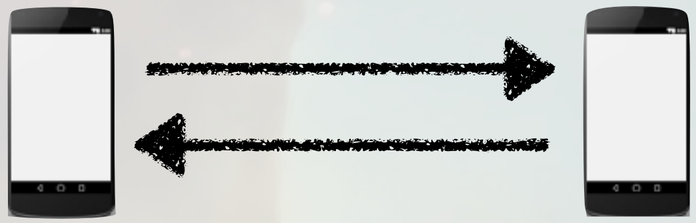
Use local network to exchange information between multiple devices

- Chats
- Content Sharing Applications
- LAN Games

How does it work?

- A device declares a service on the local network
- Look for devices connected to the network
- Peer-2-Peer connection

In this kind of connection devices are services and clients



Connection to local network

Multiple local networks

- NFC (Near Field Communication)
  - **Small distance 10 cm**
  - **Contactless payment**
- Bluetooth
  - **Medium distance 10 m**
  - **Connection with wearable**
  - **Low speed**
- Wifi
  - **Large distance**
  - **High speed**
  - **Internet**

Here we focus on P2P over Wifi

WifiP2Manager

Methods to interact with the Wifi device and to detect and connect with other peer

initialize()	Register a service on a Wifi network
connect()	Start a P2P connection with another device
discoverPeers()	Lookup for other peers
createGroup()	Construct a group with the current device as the group owner
removeGroup()	Remove the current group

How to declare a network service?

Modify AndroidManifest.xml

- ACCESS_WIFI_STATE
  - access to wifi informations
- CHANGE_WIFI_STATE
  - modify Wifi connectivity for the device
- INTERNET
  - allows the application to open sockets

```
<uses-permission
    android:required="true"
    android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission
    android:required="true"
    android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission
    android:required="true"
    android:name="android.permission.INTERNET"/>
```

Registering a Service


```
int SERVER_PORT = 9000;
WifiP2pManager mManager;
WifiP2pManager.Channel channel;
private void startRegistration() {
    Map record = new HashMap();
    record.put("listenport", String.valueOf(SERVER_PORT));
    record.put("buddyname", "John doe (on "
              + android.os.Build.MODEL + ")" );
    record.put("available", "visible");
    WifiP2pDnsSdServiceInfo serviceInfo = WifiP2pDnsSdServiceInfo
            .newInstance("_test", "_presence._tcp", record);
    mManager.addLocalService(channel, serviceInfo,
                             new WifiP2pManager.ActionListener() {
        @Override
        public void onSuccess() {
        }
        @Override
        public void onFailure(int arg0) {

        }
    });
}
```

Detecting existing service (1/3)


```
final HashMap<String, String> buddies =
                new HashMap<String, String>();
private void discoverService() {
    WifiP2pManager.DnsSdTxtRecordListener txtListener =
        new WifiP2pManager.DnsSdTxtRecordListener() {
            @Override
             /* Callback includes:
              * fullDomain: full domain name:
              *                 e.g "tt._ipp._tcp.local."
              * record: TXT record dta as a map of key/value pairs.
              * device: The device running the advertised service.
              */
            public void onDnsSdTxtRecordAvailable(
                    String fullDomain, Map record,
                WifiP2pDevice device) {
                 buddies.put(device.deviceAddress,
                         (String) record.get("buddyname") +
                  "--" + (String) record.get("available"));
    }
};
```

Detecting existing service (2/3)


```
  WifiP2pManager.DnsSdServiceResponseListener servListener =
                new WifiP2pManager.DnsSdServiceResponseListener() {
    @Override
    public void onDnsSdServiceAvailable(String instanceName,
                                        String registrationType,
                                        WifiP2pDevice
resourceType){

    // Update the device name with the human-friendly version
from
    // the DnsTxtRecord, assuming one arrived.
    resourceType.deviceName = buddies
            .containsKey(resourceType.deviceAddress) ?
               buddies.get(resourceType.deviceAddress) :
               resourceType.deviceName;
    }
};

 mManager.setDnsSdResponseListeners(channel,
                                    servListener, txtListener);
```

Detecting existing service (3/3)


```
WifiP2pDnsSdServiceRequest serviceRequest =
                    WifiP2pDnsSdServiceRequest.newInstance();
mManager.addServiceRequest(channel,
        serviceRequest,
        new WifiP2pManager.ActionListener() {
            @Override
            public void onSuccess() {
                // Success!
            }
            @Override
            public void onFailure(int code) {
                // Command failed.
                // Check for P2P_UNSUPPORTED, ERROR, or BUSY
            }
        });
    mManager.discoverServices(channel, new WifiP2pManager.ActionListener() {
        @Override
        public void onSuccess() {
        }
        @Override
        public void onFailure(int reason) {

        }
    });
}
```

Summary

- **We can now discover, create and connect to services**

- **To exchange with a device**
  - just open a socket and exchange data

- **Over other network P2P communication works the same way**
  - Just read the documentation

- **Think to remove your service when you leave the Wifi or the application**