

IEMS5722

Mobile Network Programming and Distributed Server Architecture

Lecture 8

Peer-to-Peer Networking in Android

Agenda

- Wireless Communication with Peers
- Peer-to-Peer Wi-Fi Communication
- Bluetooth
- Near-Field Communication (NFC)

Wireless Communication with Peers

Wireless Communication with Peers

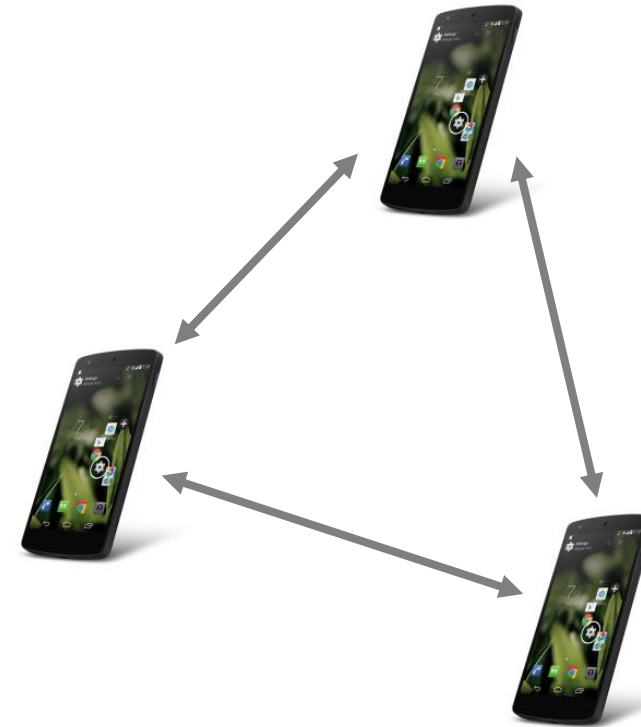
- So far we have mainly discussed the **client-server architecture**
- Clients request data or services from the server through a network. (e.g. an Intranet or the Internet)
- There are cases in which we would like to set up **peer-to-peer connections**
 - Sharing files directly with a device nearby
 - Playing multi-player games without an Wi-Fi access point (AP)
 - ...

Wireless Communication with Peers

Client-Server Architecture



Peer-to-Peer Architecture



Wireless Communication with Peers

- Methods for P2P wireless communication
 - Peer-to-peer Wi-Fi (Wi-Fi Direct)
 - Bluetooth
 - NFC



Wi-Fi

- **Wi-Fi**: “Wireless Fidelity”
 - A trademark of the Wi-Fi Alliance
 - A family of products using the **IEEE 802.11** standards
 - Usually used to set up a “wireless local area network” (**WLAN**)

| Protocol | Date of Release | Frequency (GHz) | Stream Data Rate (Mbps) (Max) | Range (m) (Indoor/Outdoor) |
|----------|-----------------|-----------------|----------------------------------|-------------------------------|
| 802.11a | Sep 1999 | 5 | 54 | 38/140 |
| 802.11b | Sep 1999 | 2.4 | 11 | 35/140 |
| 802.11g | Jun 2003 | 2.4 | 54 | 38/140 |
| 802.11n | Oct 2009 | 2.4/5 | 600 | 70/250 |
| 802.11ac | Dec 2013 | 5 | 3466.8 | 35/n.a. |
| 802.11ax | Sep 2019 | 2.4/5/6 | 10530 | 30/120 |

Reference: https://en.wikipedia.org/wiki/IEEE_802.11

Wi-Fi Peer-to-Peer/Wi-Fi Direct

- Connects devices directly using Wi-Fi protocols, **without requiring a wireless access point** (AP)
- Supported in **Android 4.0** or above
- Devices communicate with each other by establishing P2P Groups
 - A device will become “**Group Owner**” (act as an AP in the group)
 - Other device will become “**Clients**”

Bluetooth

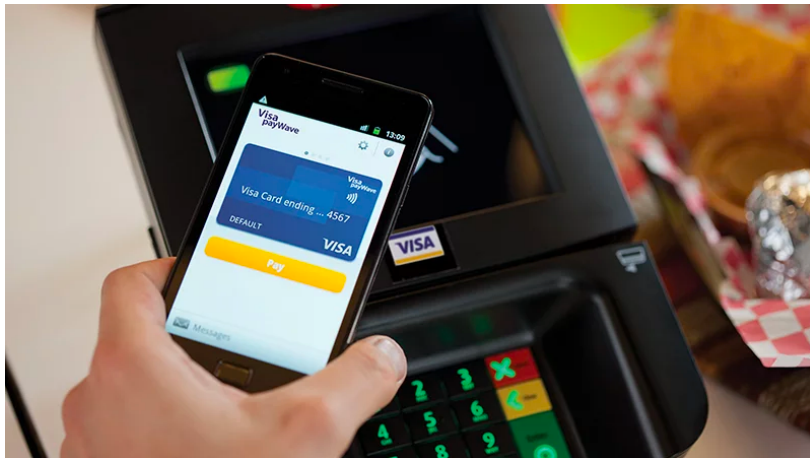
- A wireless solution for data and voice communication **over short distances**
- Intended for building **personal area networks** (PANs)
- Range from 0.5 m to 100 m, typically less than **10 m**

| Class | Maximum Permitted Power (mW) | Typical Range (m) |
|-------|------------------------------|-------------------|
| 1 | 100 | ~100 |
| 2 | 2.5 | ~10 |
| 3 | 1 | ~1 |
| 4 | 0.5 | ~0.5 |

| Features | Version | Data Rate (Mbps) (Max) |
|----------------------------|-----------|------------------------|
| Basic Rate (BR) | 1.2 | 0.721 |
| Enhanced Data Rate (EDR) | 2.0 + EDR | 2.1 |
| High Speed (HS) | 3.0 + HS | 24 |
| Bluetooth Low Energy (BLE) | 4.0 | 1 |

NFC

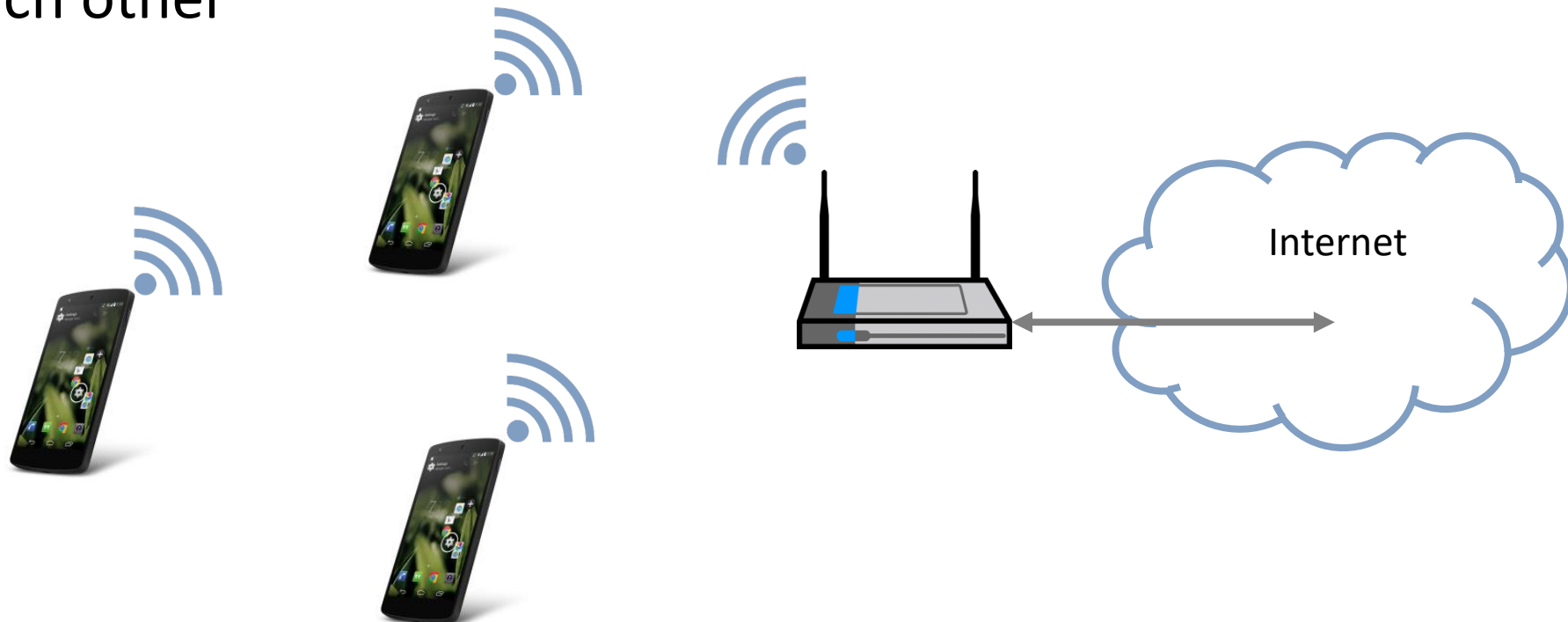
- **Near-Field Communication (NFC)**
 - For short-range (<10cm) data communication
 - One of the **RFID** (Radio Frequency Identification) standards
 - Data rate: 106 kbps to 424 kbps
 - Allows **two-way** communication between two devices



Peer-to-Peer Wi-Fi Communication

Common WLAN

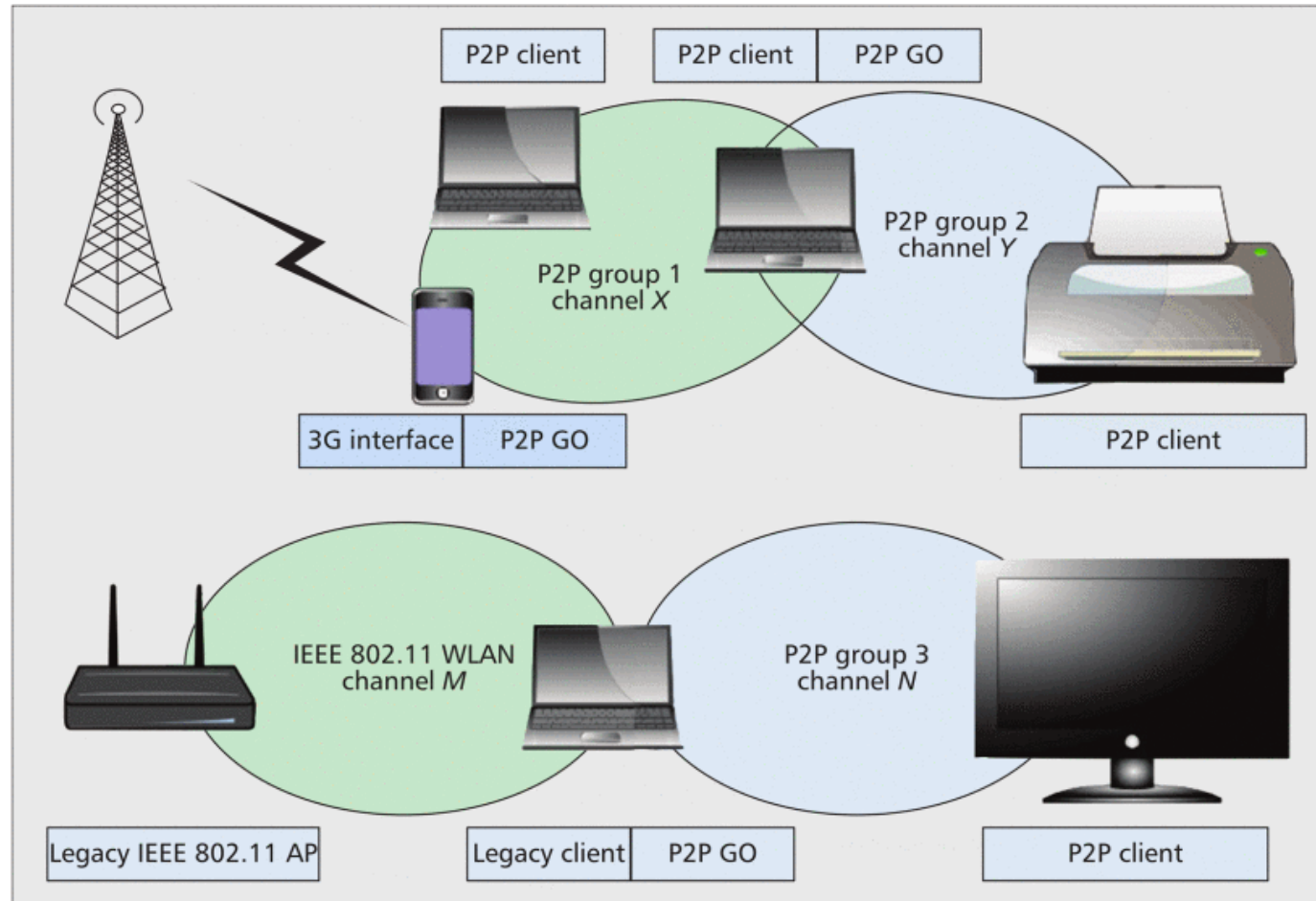
- In a common wireless network, an access point (AP) is present to allow devices to connect to a network
- Devices within the WLAN communicate through the AP, not directly to each other



Wi-Fi Direct

- Wi-Fi Direct allows devices to communicate with one another without the presence of an AP
- Any device may assume the role of an AP
- Communication is done by creating **dynamic groups**
 - Device playing the role of an AP is called the **Group Owner** (GO)
 - Other devices are then **clients** within this group
 - The role of GO cannot be transferred
 - If GO leaves, the group is automatically closed

Wi-Fi Direct



Reference:

D. Camps-Mur, A. Garcia-Saavedra and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," in *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96-104, June 2013, doi: 10.1109/MWC.2013.6549288.

Group Formation in Wi-Fi Direct

- When two devices have discovered each other and want to establish a connection, they will negotiate their roles in the group
- Three group formation technique:
 1. Standard
(Two devices negotiate their roles in the group)
 2. Autonomous
(One device automatically creates a group)
 3. Persistent
(Roles and credentials are stored for future group establishment)

Power Saving in Wi-Fi Direct

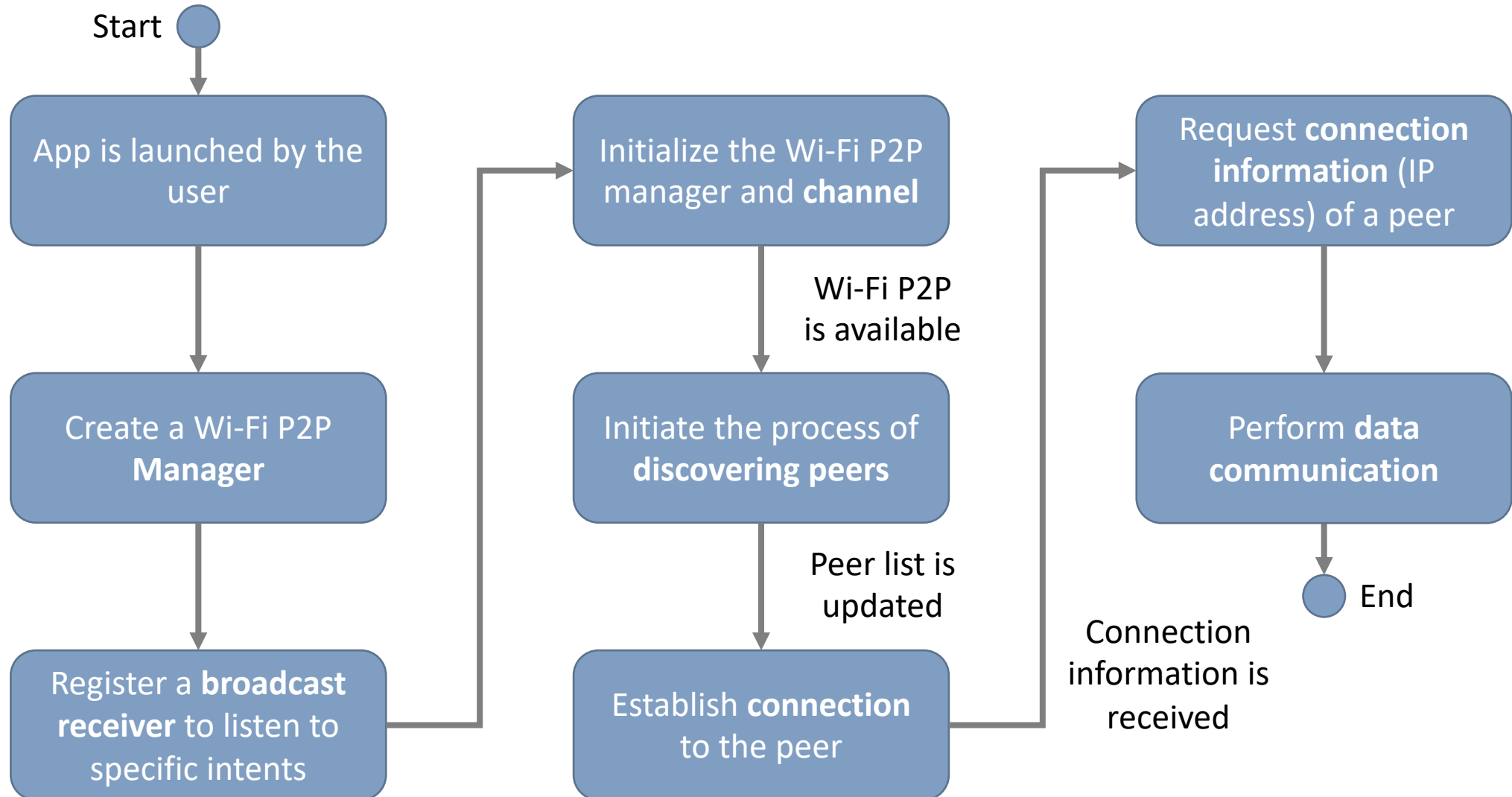
- Wi-Fi Direct is specifically designed for mobile devices
- As any devices can potentially become an AP, power saving is very important
- Two mechanisms for saving power in the AP:
 1. **Opportunistic power-saving protocol**
When all clients are sleeping, the AP will go to sleep
 2. **Notice-of-absence protocol**
The AP will announce certain period of time when it will go to sleep to save power, clients are not allowed to access the channel during these times

Wi-Fi Direct in Android

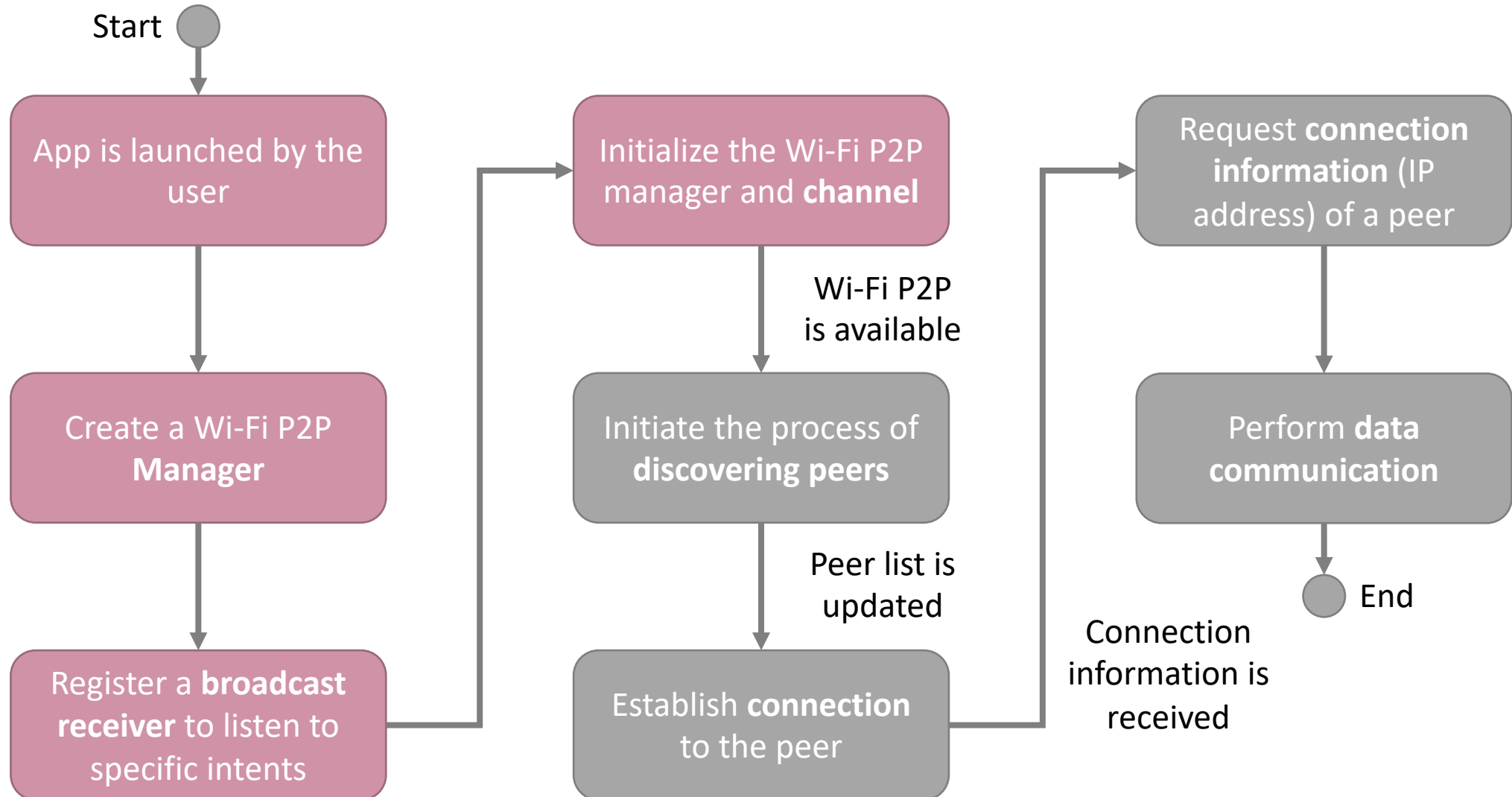
Wi-Fi Direct on Android

- The Wi-Fi Direct (P2P) APIs are available in Android 4.0 or later
- **Four** major operations in a Wi-Fi P2P app
 - Creating and registering a **broadcast receiver**
 - **Discovering** peers
 - **Connecting** to a peer
 - **Transferring data** to a peer

Wi-Fi Direct on Android – System Flow



Wi-Fi Direct on Android – Setup



Wi-Fi Direct on Android – Setup

- P2P Wi-Fi related events that will be broadcasted in the Android system
 - **WIFI_P2P_STATE_CHANGED_ACTION**
Indicates whether Wi-Fi P2P is enabled.
 - **WIFI_P2P_PEERS_CHANGED_ACTION**
Indicates that the available peer list has changed.
 - **WIFI_P2P_CONNECTION_CHANGED_ACTION**
Indicates the state of Wi-Fi P2P connectivity has changed.
 - **WIFI_P2P_THIS_DEVICE_CHANGED_ACTION**
Indicates this device's configuration details have changed.

Reference: <https://developer.android.com/training/connect-devices-wirelessly/wifi-direct>

Wi-Fi Direct on Android – Setup

- Wi-Fi Direct requires the following permissions to be added to the manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="..."
    ...
    <uses-permission
        android:required="true"
        android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <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"/>
    ...
```

Wi-Fi Direct on Android – Setup

- You need to instantiate an **IntentFilter** to listen for broadcast intents

```
private final IntentFilter intentFilter = new IntentFilter();
...
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // Indicates a change in the Wi-Fi P2P status.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
    // Indicates a change in the list of available peers.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
    // Indicates the state of Wi-Fi P2P connectivity has changed.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
    // Indicates this device's details have changed.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
    ...
}
```

Wi-Fi Direct on Android – Setup

- At the end of the onCreate() method, initialize the **WifiP2pManager** and get a channel object. You will use it later to connect your app to the Wi-Fi P2P framework.

```
Channel channel;  
WifiP2pManager manager;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    ...  
    manager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);  
    channel = manager.initialize(this, getMainLooper(), null);  
}
```


Wi-Fi Direct on Android – Setup

- You also need to implement a **BroadcastReceiver** to listen for change broadcasts

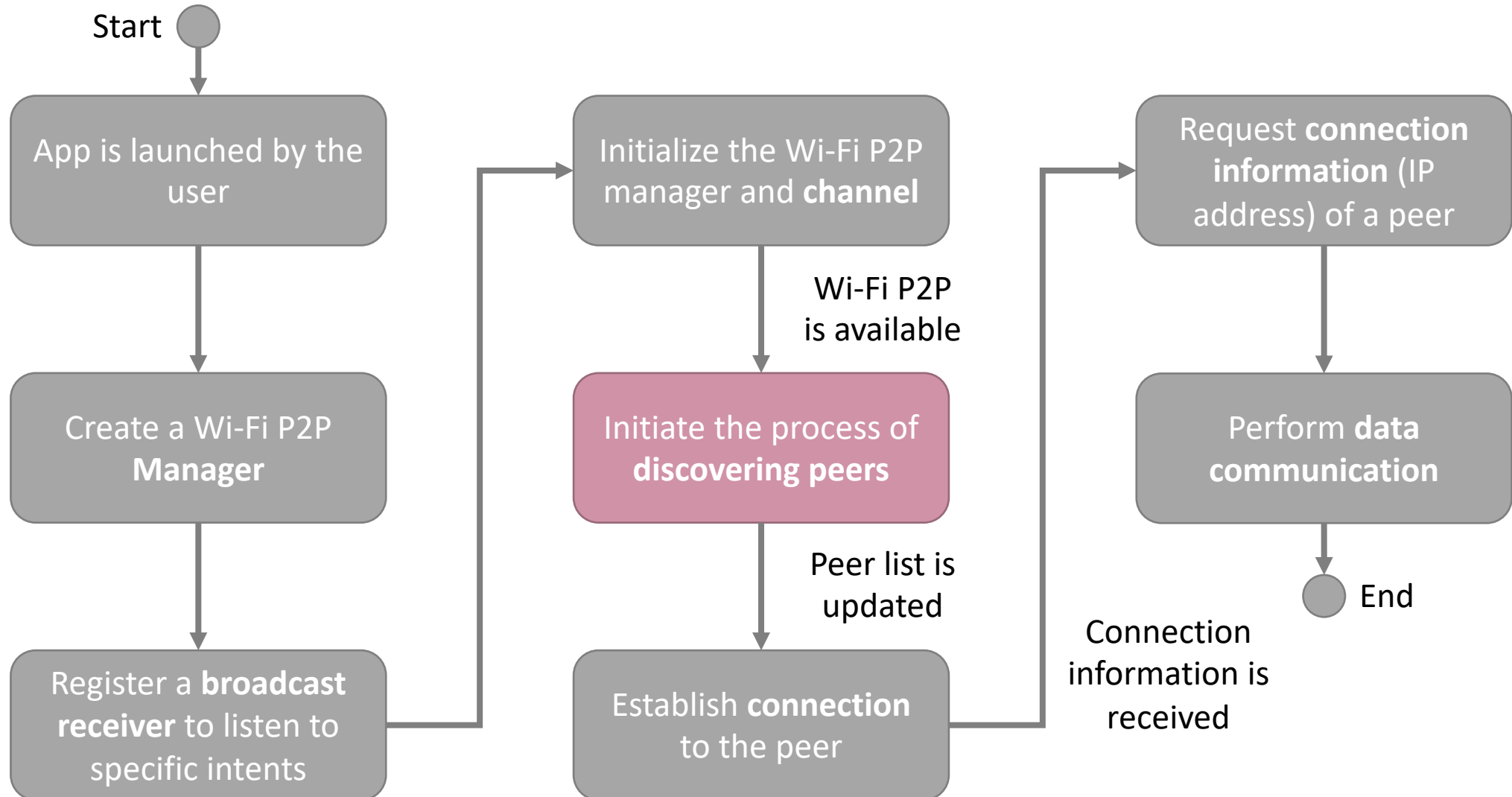
```
@Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
        // Determine if Wifi P2P mode is enabled or not, alert the Activity.
        int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
            // Wifi P2P mode is enabled
        } else {
            // Wifi P2P mode is disabled
        }
    } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
        // The peer list has changed!
    } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
        // Connection state changed!
    } else if (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
        // Configuration of this device has changed!
    }
}
```

Wi-Fi Direct on Android – Setup

- Finally, register the **IntentFilter** and **BroadcastReceiver** when main activity is active, and unregister them when the activity is paused.

```
/** register the BroadcastReceiver with the intent values to be matched */  
@Override  
public void onResume() {  
    super.onResume();  
    receiver = new WifiDirectBroadcastReceiver(manager, channel, this);  
    registerReceiver(receiver, intentFilter);  
}  
  
@Override  
public void onPause() {  
    super.onPause();  
    unregisterReceiver(receiver);  
}
```

Wi-Fi Direct on Android – Peer Discovery



Wi-Fi Direct on Android – Peer Discovery

- Search for nearby devices with Wi-Fi P2P.

```
manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {  
    @Override  
    public void onSuccess() {  
        // When the discovery action is successful  
        // Usually left blank  
    }  
  
    @Override  
    public void onFailure(int reasonCode) {  
        // When the discovery action fails  
        // Alert the user that something went wrong.  
    }  
});
```

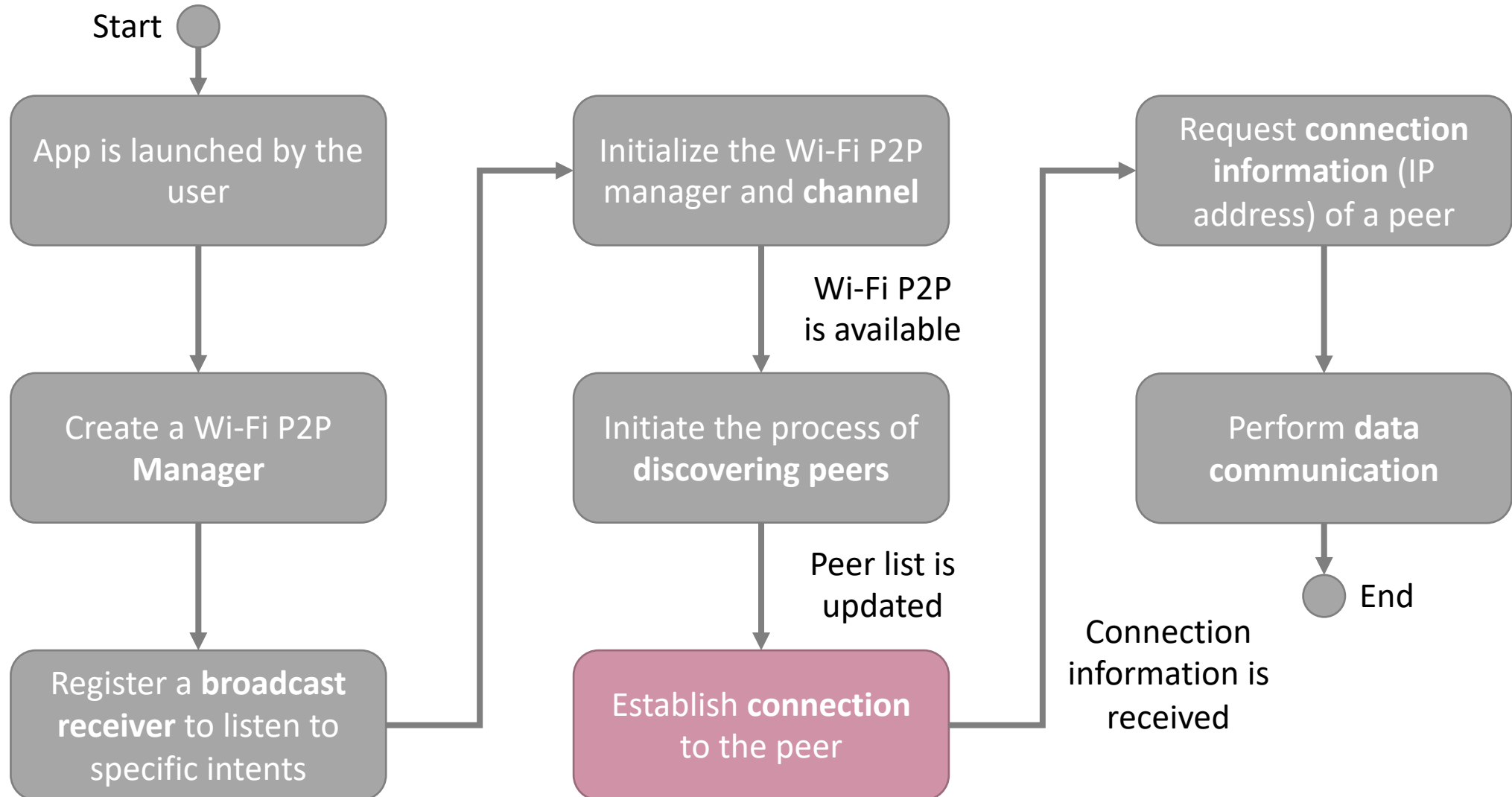
- When peers are discovered, a **WIFI_P2P_PEERS_CHANGED_ACTION** will be broadcasted

Wi-Fi Direct on Android – Peer Discovery

- When the BroadcastReceiver receives **WIFI_P2P_PEERS_CHANGED_ACTION**, call **requestPeers()** with an implemented **WifiP2pManager.PeerListListener** interface to get the list of peers.

```
else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {  
    // The peer list has changed!  
    if (manager != null) {  
        manager.requestPeers(channel, new PeerListListener() {  
            @Override  
            public void onPeersAvailable(WifiP2pDeviceList peerList) {  
                List<WifiP2pDevice> peers = peerList.getDeviceList();  
                ...  
            }  
        }  
    }  
}
```

Wi-Fi Direct on Android – Peer Connection



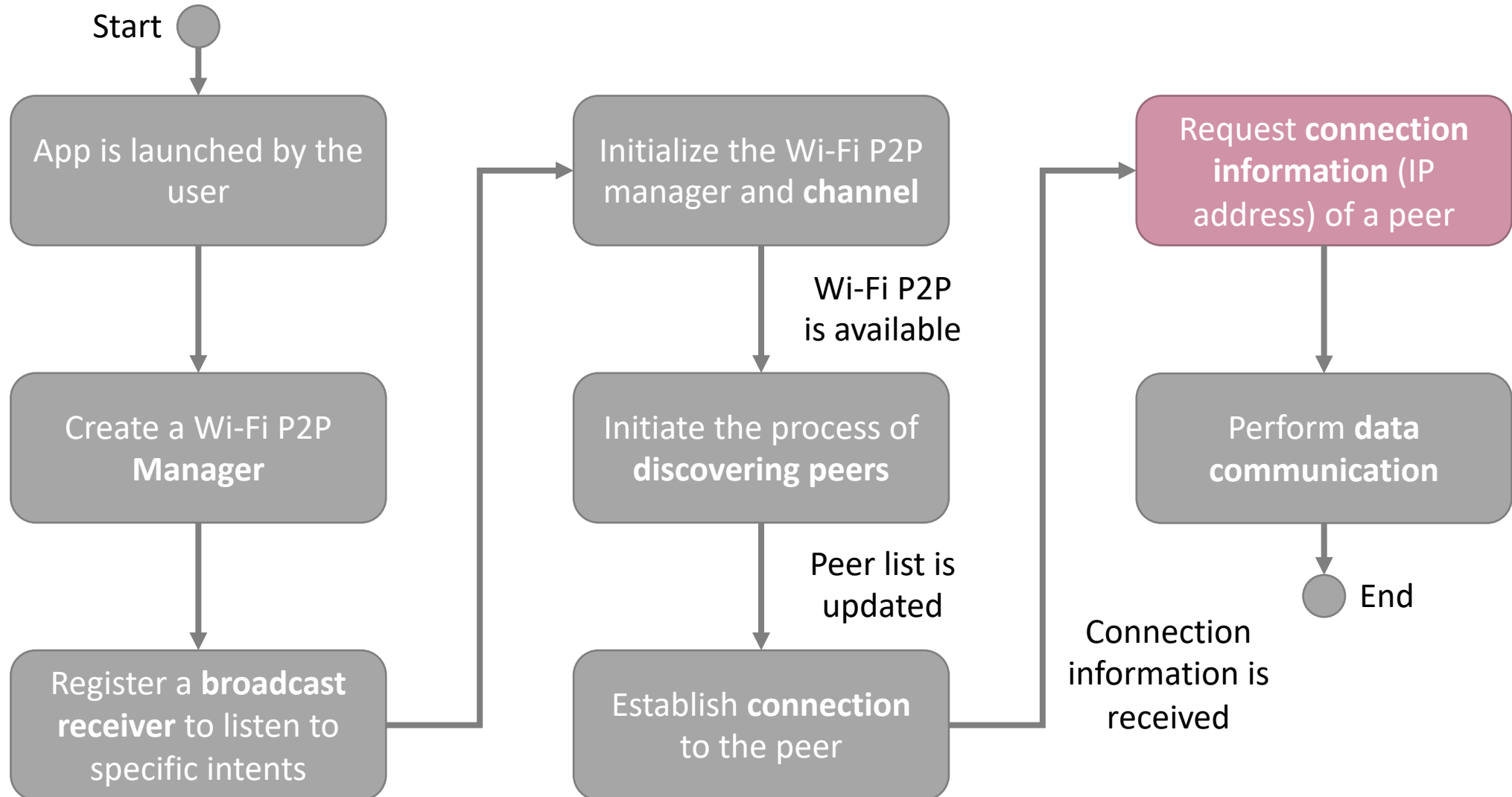
Wi-Fi Direct on Android – Peer Connection

- To connect to a peer, create a new **WifiP2pConfig** object and copy **WifiP2pDevice** data into it.

```
// Picking the first device found on the network.
WifiP2pDevice device = peers.get(0);
WifiP2pConfig config = new WifiP2pConfig();
config.deviceAddress = device.deviceAddress;
config.wps.setup = WpsInfo.PBC;

manager.connect(channel, config, new ActionListener() {
    @Override
    public void onSuccess() {
        // WiFiDirectBroadcastReceiver notifies us. No action is needed here.
    }
    @Override
    public void onFailure(int reason) {
        // Connection failed. Alert the user.
    }
});
```

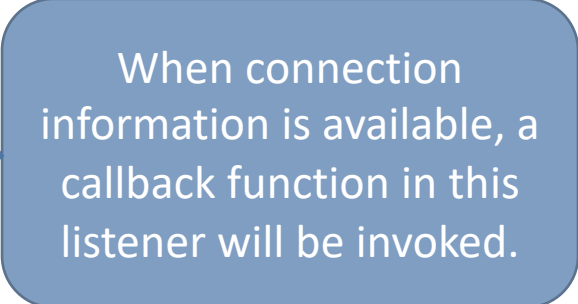
Wi-Fi Direct on Android – Information Request



Wi-Fi Direct on Android – Information Request

- After successful connection indicated by **WIFI_P2P_CONNECTION_CHANGED_ACTION** intent, information can be requested for this connection.

```
} else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {  
    // Connection state changed!  
    if (manager == null)  
        return;  
  
    NetworkInfo networkInfo = (NetworkInfo) intent.  
        getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);  
  
    if (networkInfo.isConnected()) {  
        manager.requestConnectionInfo(channel, connectionListener);  
    }  
    ...  
}
```



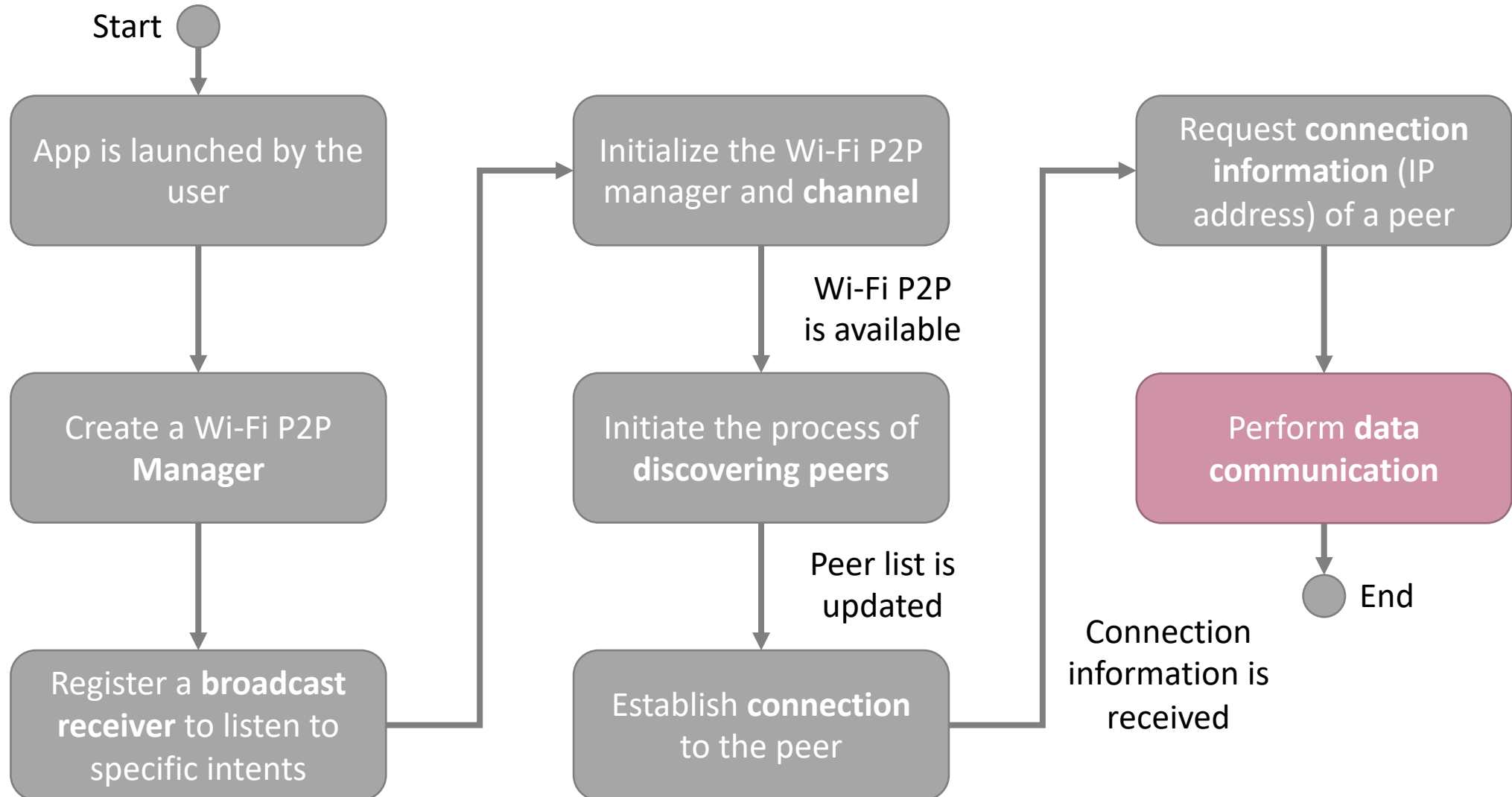
When connection information is available, a callback function in this listener will be invoked.

Wi-Fi Direct on Android – Information Request

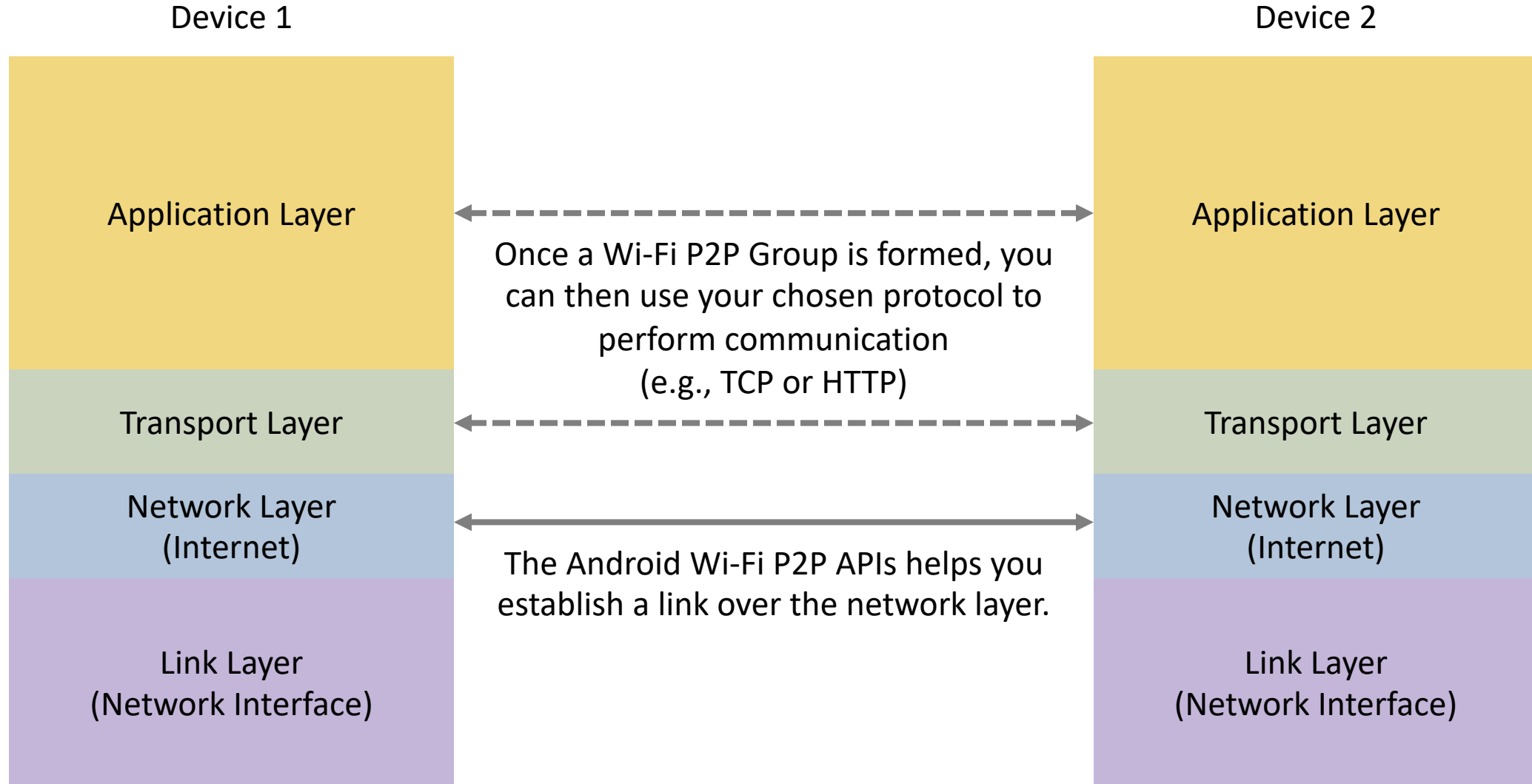
- Implement the **WifiP2pManager.ConnectionInfoListener** interface to get the connection information when available.

```
ConnectionInfoListener connectionListener = new ConnectionInfoListener() {  
    @Override  
    public void onConnectionInfoAvailable(final WifiP2pInfo info) {  
  
        // InetAddress from WifiP2pInfo struct.  
        InetAddress groupOwnerAddress = info.groupOwnerAddress.getHostAddress();  
  
        // After the group negotiation, we can determine the group owner (server).  
        if (info.groupFormed && info.isGroupOwner) {  
            // The current device is the group owner. Start a server thread.  
        } else if (info.groupFormed) {  
            // The current device is a client. Connect to the group owner in a new thread.  
        }  
    }  
};
```

Wi-Fi Direct on Android – Data Communication



Wi-Fi Direct on Android – Data Communication



Wi-Fi Direct on Android

- For more information, refer to the documentation on official Android guide:
 - <https://developer.android.com/guide/topics/connectivity/wifip2p.html>
 - <https://developer.android.com/training/connect-devices-wirelessly/wifi-direct>
- Check the following project for a demo app that use P2P Wi-Fi
 - <https://android.googlesource.com/platform/development/+master/samples/WiFiDirectDemo/>

Bluetooth

Bluetooth

- A wireless solution for data and voice communication **over short distances**
- Intended for building **personal area networks** (PANs)
- Range from 0.5 m to 100 m, typically less than **10 m**

| Class | Maximum Permitted Power (mW) | Typical Range (m) |
|-------|------------------------------|-------------------|
| 1 | 100 | ~100 |
| 2 | 2.5 | ~10 |
| 3 | 1 | ~1 |
| 4 | 0.5 | ~0.5 |

| Features | Version | Data Rate (Mbps) (Max) |
|----------------------------|-----------|------------------------|
| Basic Rate (BR) | 1.2 | 0.721 |
| Enhanced Data Rate (EDR) | 2.0 + EDR | 2.1 |
| High Speed (HS) | 3.0 + HS | 24 |
| Bluetooth Low Energy (BLE) | 4.0 | 1 |

Bluetooth

- The Bluetooth specification is an open specification that is governed by the **Bluetooth Special Interest Group (SIG)**
- The Bluetooth SIG is led by a group of companies which forms the “**Promoter Group**”: Intel, Ericsson, Microsoft, Toshiba, Lenovo and Nokia
- Initiated by Ericsson in 1994, when it investigated the feasibility of a low power, low cost radio interface between mobile phones and their accessories
- The Bluetooth SIG was officially formed in **1998**

Bluetooth Applications

- Common applications
 - Wireless mouse and keyboard
 - Wireless earphones, headphones and microphones
 - File transmission between mobile devices
 - Wearable devices (e.g. smart watches)
 - Sensing (e.g. iBeacon, Eddystone)



Bluetooth Profiles

- **Bluetooth profiles** are additional protocols that build upon the basic Bluetooth standard to more clearly define what kind of data a Bluetooth module is transmitting.
- The profile(s) a Bluetooth device supports determine(s) what application it's geared towards, e.g.,
 - Headset profile (HSP)
 - Human interface device profile (HID)
 - Advanced Audio Distribution Profile (A2DP)
 - ...
- **Note:** For two Bluetooth devices to be compatible, they must support the same profiles.

Android Bluetooth APIs

References

- Android Bluetooth Guide
<https://developer.android.com/guide/topics/connectivity/bluetooth.html>
- Android Bluetooth Adapter API Reference
<https://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html>
- Bluetooth Chat Sample App
<https://github.com/googlearchive/android-BluetoothChat>

Near-Field Communication (NFC)

NFC

- A set of short-range wireless technologies, typically requiring a distance of 4 cm or less to initiate a connection.
- Share data between
 1. An NFC tag and an Android-powered device, or
 2. Two Android-powered devices
- Data is exchanged using an NFC Forum standard called **NFC Data Exchange Format** (NDEF)

NFC

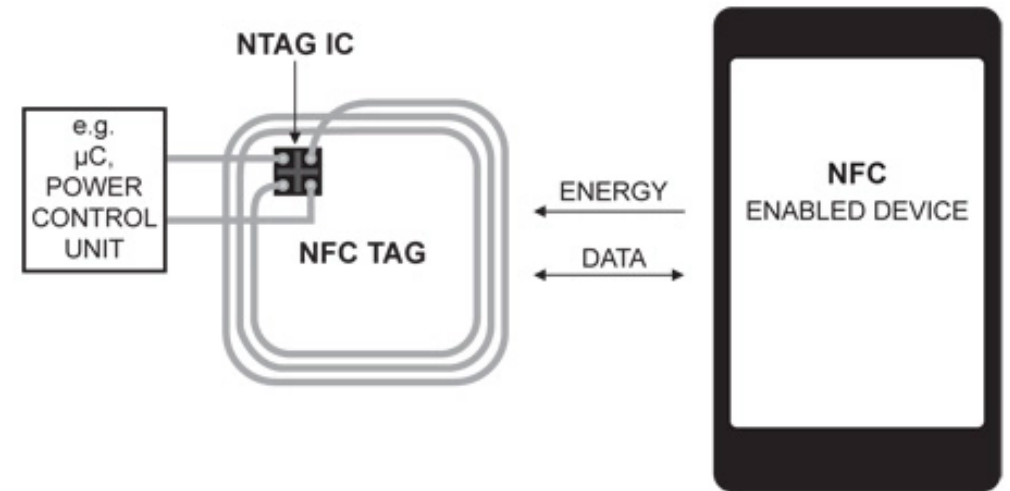
- Some history of NFC:
 - 2002: Philips and Sony attempted to establish a new RFID technology specification
 - 2004: Nokia, Philips and Sony established the NFC Forum
 - 2006: Initial specification of NFC Tags released
 - 2010: The first Android NFC phone – Samsung Nexus S
 - 2011: RIM devices (Blackberry) became first device in the world to be certified for MasterCard PayPass service
 - 2013: Samsung and VISA announce their partnership
 - 2015: Swatch established partnership with UnionPay and VISA to enable NFC financial transactions

NFC

- Each active NFC device can work in one or more of the three modes:
 - **NFC card emulation**
 - Enables NFC-enabled devices such as smartphones to act like smart cards, allowing users to perform transactions such as payment or ticketing.
 - **NFC reader/writer**
 - Enables NFC-enabled devices to read information stored on inexpensive NFC tags embedded in labels or smart posters.
 - **NFC peer-to-peer**
 - Enables two NFC-enabled devices to communicate with each other to exchange information in an ad hoc fashion.

NFC Tags

- NFC Tags are **passive devices**
 - An active device (e.g. a mobile phone) would generate a magnetic field which induces electric current in the antenna of the tag.
 - The tag then modulates the magnetic field that can be read back by the device to enable data transfer.



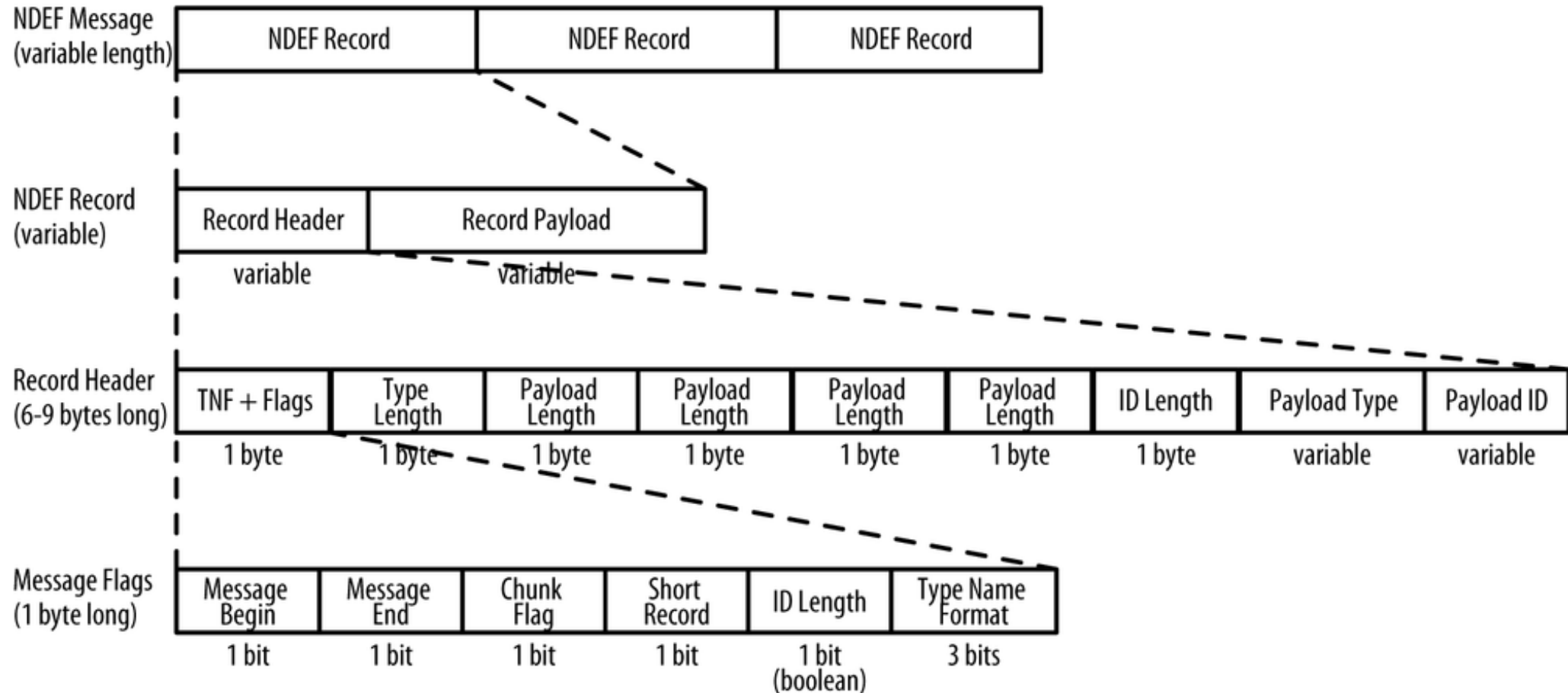
NFC Applications

- Common applications of NFC include:
 - Contactless payment (e.g., credit cards, electronic tickets).
 - Bootstrapping other connection technologies such Bluetooth and Wi-Fi.
 - Sharing contacts, photos, videos.
 - Acting as identity documents and keycards.
 - Smartphone automation and NFC tags.
 - Gaming
 - Sports
 - ...

NFC Forum Data Exchange Format

- **NFC Forum Data Exchange Format (NDEF):**
 - A lightweight binary message format designed to encapsulate one or more application-defined payloads into a single message construct.
 - A NDEF message contains one or more NDEF records, each carrying a payload of arbitrary type and up to $2^{32}-1$ bytes in size.
 - Records can be chained together to support larger payloads
 - An NDEF record carries three parameters for describing its payload: the payload length, the payload type, and an optional payload identifier.

NDEF Message Structure



Reference:

T. Igoe, D. Coleman, and B. Jepson, *Beginning NFC: Near Field Communication with Arduino, Android, and PhoneGap*, 1st Ed. Sebastopol, CA, USA: O'Reilly Media, 2014.

Android NFC APIs

NFC on Android

- On Android, there are three major use cases of NFC technology
 1. **Scanning NFC tags**, perform read or write operations
 2. Sending data from one device to another using **Android Beam**
 3. Making the device **emulate an NFC tag** and interact with an NFC reader

NFC Application Scenarios

- When you are developing for an app that uses NFC, mostly like it will be one of the following scenarios

1

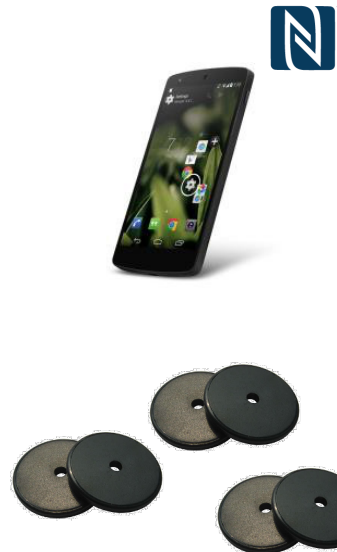
Read and write
your NFC Tags



You design the NFC tags
and how data is stored

2

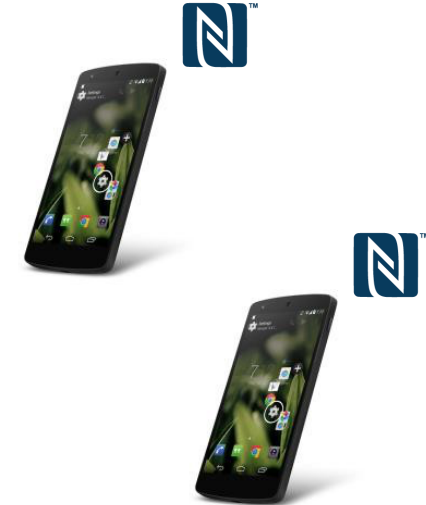
Read and write
others' NFC Tags



Others design the NFC tags
and tell you the data format

3

Exchange data
between two devices



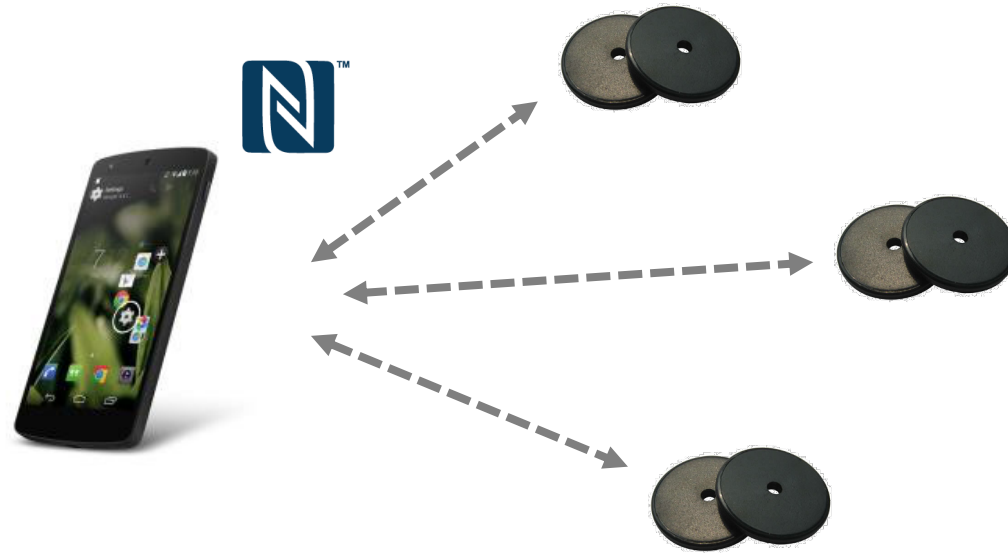
Both devices have
your app installed

NFC Application Scenarios

- Android has the most support for NFC tags with data stored according to NDEF
 - If you have control over the tags, use NDEF if possible
 - We will talk about how the following:
 1. How to read/write NDEF tags or to carry out actions
 2. How to exchange data using Android Beam
- For more advanced topics (e.g. handling non-NDEF data), refer to:
<https://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc>

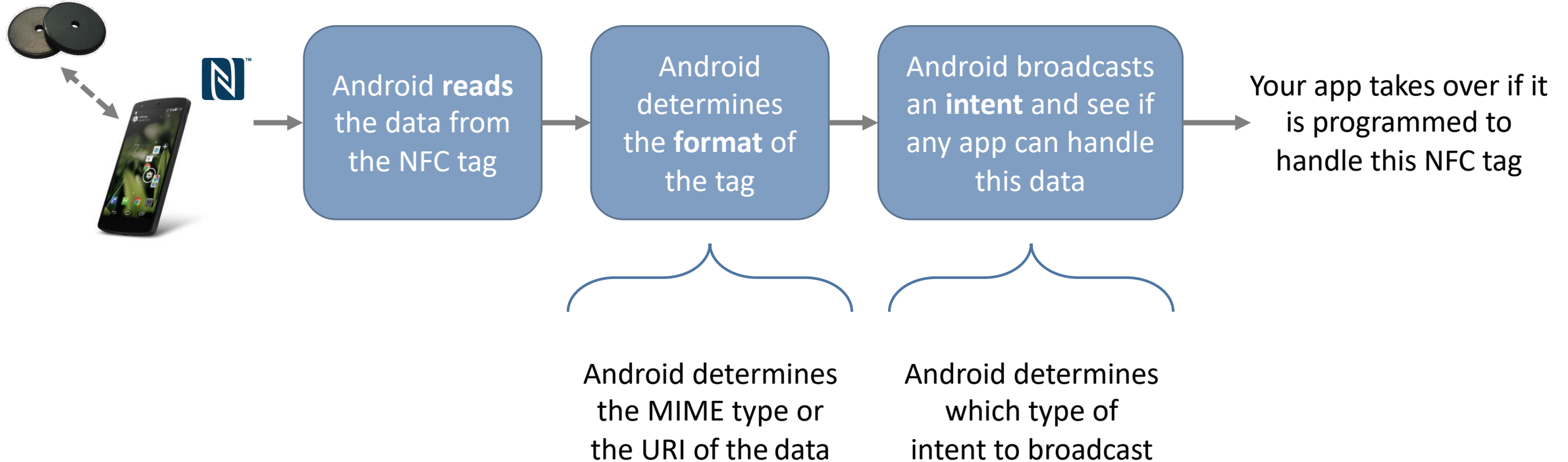
Read From an NFC Tag

- Android devices usually **scan** for NFC tags when
 - NFC is **enabled**, and
 - the screen is **unlocked**



Android NFC Tag Dispatch System

- When Android discovers an NFC tag around, it will read the data from the tag, and then dispatch the data to the apps installed in the device.



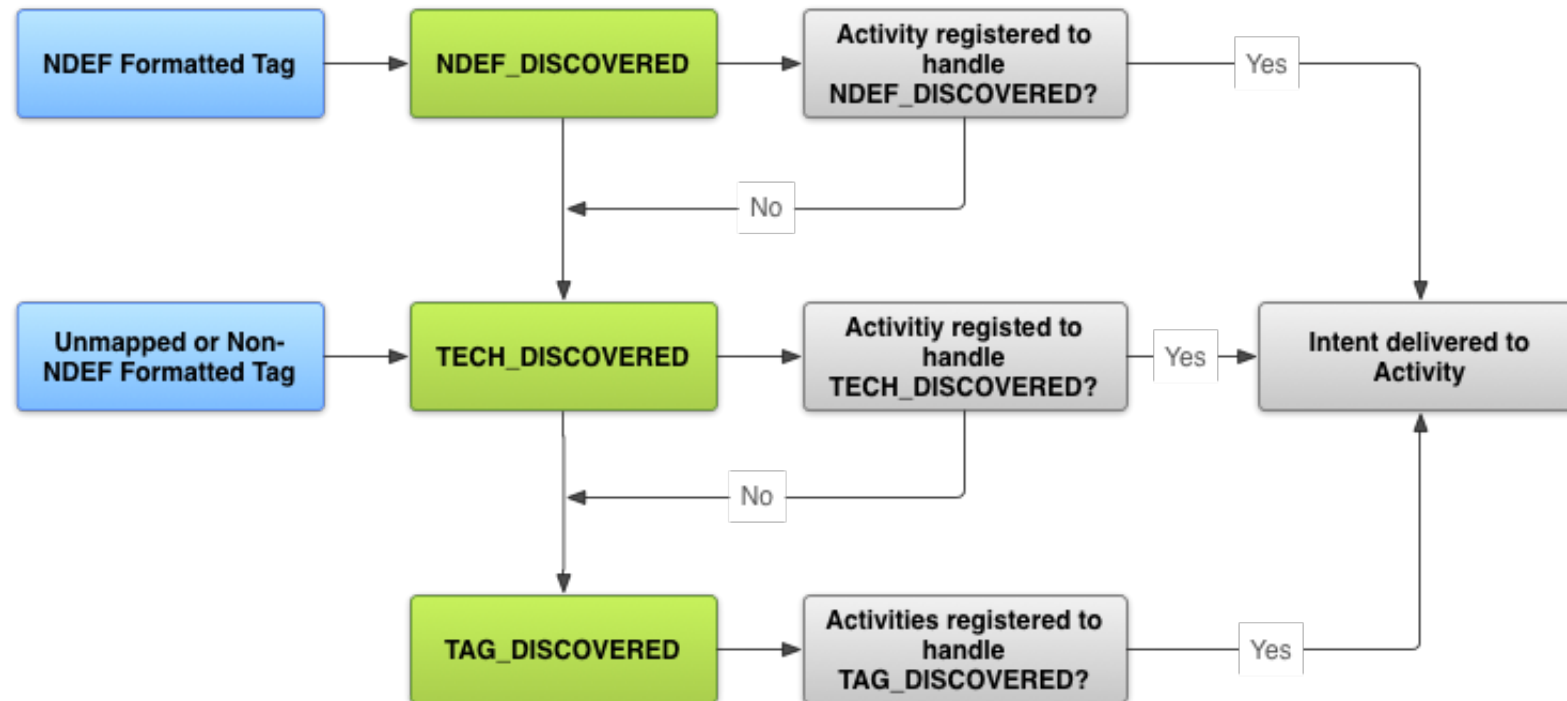
Android NFC Tag Dispatch System

- When Android discovers an NFC tag, it will try to parse it, and determine the format and data type
- The process mainly involves parsing the header of the first record in the NDEF formatted message (see reference)
- Once the data is parsed, Android will create an intent and broadcast it in the system
- Three different intents (in order of priority):
 1. **ACTION_NDEF_DISCOVERED**
 2. **ACTION_TECH_DISCOVERED**
 3. **ACTION_TAG_DISCOVERED**

Reference: <https://developer.android.com/guide/topics/connectivity/nfc/nfc#ndef>

Android NFC Tag Dispatch System

- After parsing the NFC tag data, Android will try to start an Activity with the intent created by the tag dispatch system.
- If no activity filter for the intent, Android will try to start an Activity with lower-priority intent.



NFC on Android

- To use the NFC features, the following permission is required

```
<uses-permission android:name="android.permission.NFC" />
```

- Also, API level 10 includes comprehensive NFC API support, and API level 14 is needed if Android Beam API is used.

```
<uses-sdk android:minSdkVersion="10" />
```

- If your app requires NFC features to work properly, you should include:

```
<uses-feature  
    android:name="android.hardware.nfc"  
    android:required="true" />
```

Intent Filters

- Next, you will have to prepare your application to filter for NFC related intents
- You can filter for the ACTION_NDEF_DISCOVERED, and specify the MIME type or URI your app can handle
- For example: the following declaration will allow the activity to filter for text data embedded in an NFC tag

```
<intent-filter>  
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <data android:mimeType="text/plain" />  
</intent-filter>
```

Intent Filters

- Another example: if you would like to handle URLs with a particular format

```
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />
  <category android:name="android.intent.category.DEFAULT" />
  <data android:scheme="http"
        android:host="example.com"
        android:pathPrefix="/index.html" />
</intent-filter>
```

- When the URL “<http://example.com/index.html>” is embedded in the NFC tag, your app will be able to pick up this intent

Creating NDEF Messages in Android

Creating NDEF Messages

- Before sending data from one device to another device, you need to put your data into a well-formatted NDEF message
 - An NDEF message is made up of one or more NDEF records
 - The type of the message should be embedded in the FIRST record
 - Starting from Android 4.1 (API Level 16), the following methods are available for you to create NDEF records more easily
 - `NdefRecord.createUri()`
 - `NdefRecord.createMime()`

Creating NDEF Messages

- Creating a NDEF record that contains an URI

```
String uri = "http://www.example.com/appdata";  
NdefRecord record = NdefRecord.createUri(uri);  
NdefMessage message = new NdefMessage(new NdefRecord[] { record });
```

Create an NDEF record
containing an URI

Create an NDEF message
containing the record

Creating NDEF Messages

- The corresponding intent filter for the above NDEF message

```
<intent-filter>  
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <data android:scheme="http"  
        android:host="www.example.com"  
        android:pathPrefix="/appdata" />  
</intent-filter>
```

Creating NDEF Messages

- Creating a NDEF record that contains data of a certain MIME type

```
NdefRecord record = NdefRecord.createMime("text/plain",  
    "Beam me up, Android".getBytes(Charset.forName("UTF-8")));  
  
NdefMessage message = new NdefMessage(new NdefRecord[] { record });
```

Create an NDEF record
containing plain text data

Create an NDEF message
containing the record

Creating NDEF Messages

- The corresponding intent filter for the above NDEF message

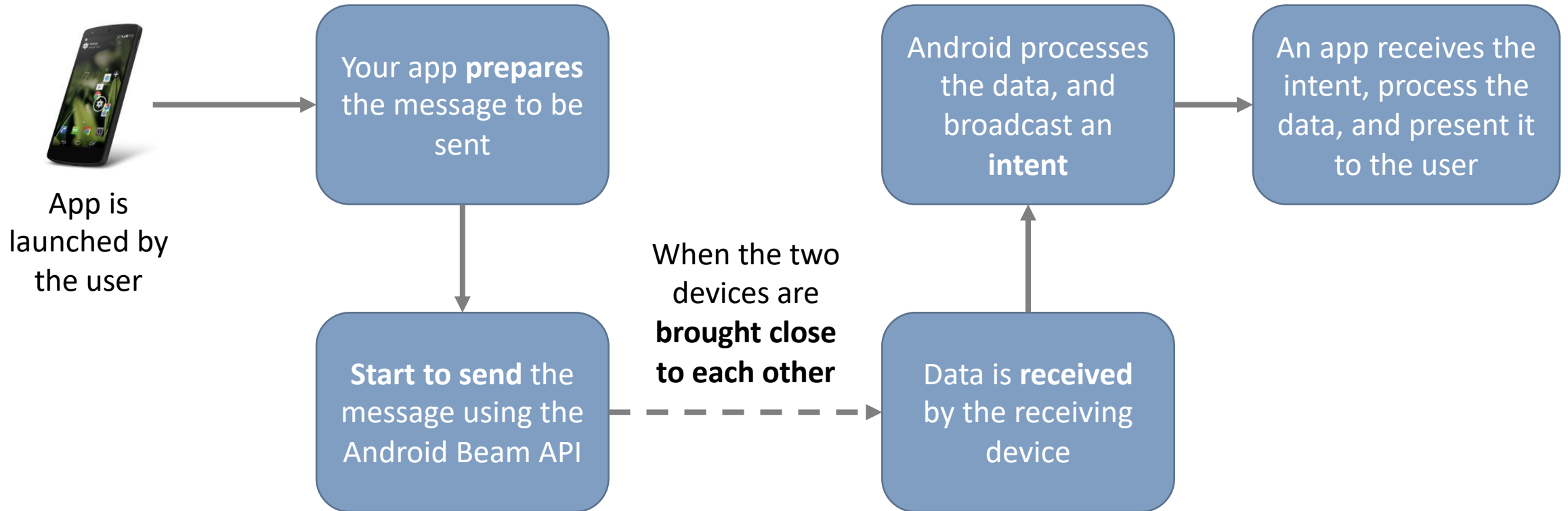
```
<intent-filter>  
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <data android:mimeType="text/plain" />  
</intent-filter>
```

Android Beam API

Android Beam

- Android Beam is a feature on Android devices that allow simple peer-to-peer data exchange using NFC technology
- Conditions for Android Beam to be executed:
 - The app that wants to send out data must be in the **foreground** of the device
 - The device accepting the data **must not be locked**

Android Beam



Android Beam

- An example of beam an NDEF message


```
public class Beam extends Activity implements CreateNdefMessageCallback {
    NfcAdapter nfcAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        nfcAdapter = NfcAdapter.getDefaultAdapter(this);
        if (nfcAdapter == null) {
            // NFC not available, skip
            finish();
            return;
        }
        // Register callback
        nfcAdapter.setNdefPushMessageCallback(this, this);
    }
    ...
}
```

Android Beam

- An example of beam an NDEF message (cont'd)

```
...
@Override
public NdefMessage createNdefMessage(NfcEvent event) {
    String text = "Some text to be beamed to another device";
    NdefMessage message = new NdefMessage(
        new NdefRecord[] {
            createMime("application/vnd.hk.edu.cuhk.ie.iems5722",
                text.getBytes())
        });
    return message;
}
...
```



Replace this with "vnd." and
the package name of your
app

Android Beam

- Receiving the data in the receiving device
- First you should set the intent filter in one of the activities in your app

```
<intent-filter>  
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />  
    <category android:name="android.intent.category.DEFAULT" />  
    <data android:mimeType="application/vnd.hk.edu.cuhk.ie.iems5722" />  
</intent-filter>
```

Android Beam

- In the activity, override the following methods to receive the intent

```
...
@Override
public void onResume() {
    super.onResume();
    // Check to see that the Activity started due to an Android Beam
    if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
        processIntent(getIntent());
    }
}

@Override
public void onNewIntent(Intent intent) {
    // onResume gets called after this to handle the intent
    setIntent(intent);
}
...
```

Android Beam

- Process the Intent and extract the data

```
...
void processIntent(Intent intent) {
    Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
        NfcAdapter.EXTRA_NDEF_MESSAGES);

    // only one message sent during the beam
    NdefMessage msg = (NdefMessage) rawMsgs[0];

    // record 0 contains the MIME type
    String data = new String(msg.getRecords()[0].getPayload());
}
...
```

References

- The NFC Forum
<https://nfc-forum.org/>
- Android NFC APIs
<https://developer.android.com/guide/topics/connectivity/nfc>
- NXP – NFC Tags and Developer Resources
https://www.nxp.com/products/rfid-nfc/nfc-hf:MC_71110#resources

Next Lecture: Asynchronous Tasks & Message Queues

End of Lecture 8