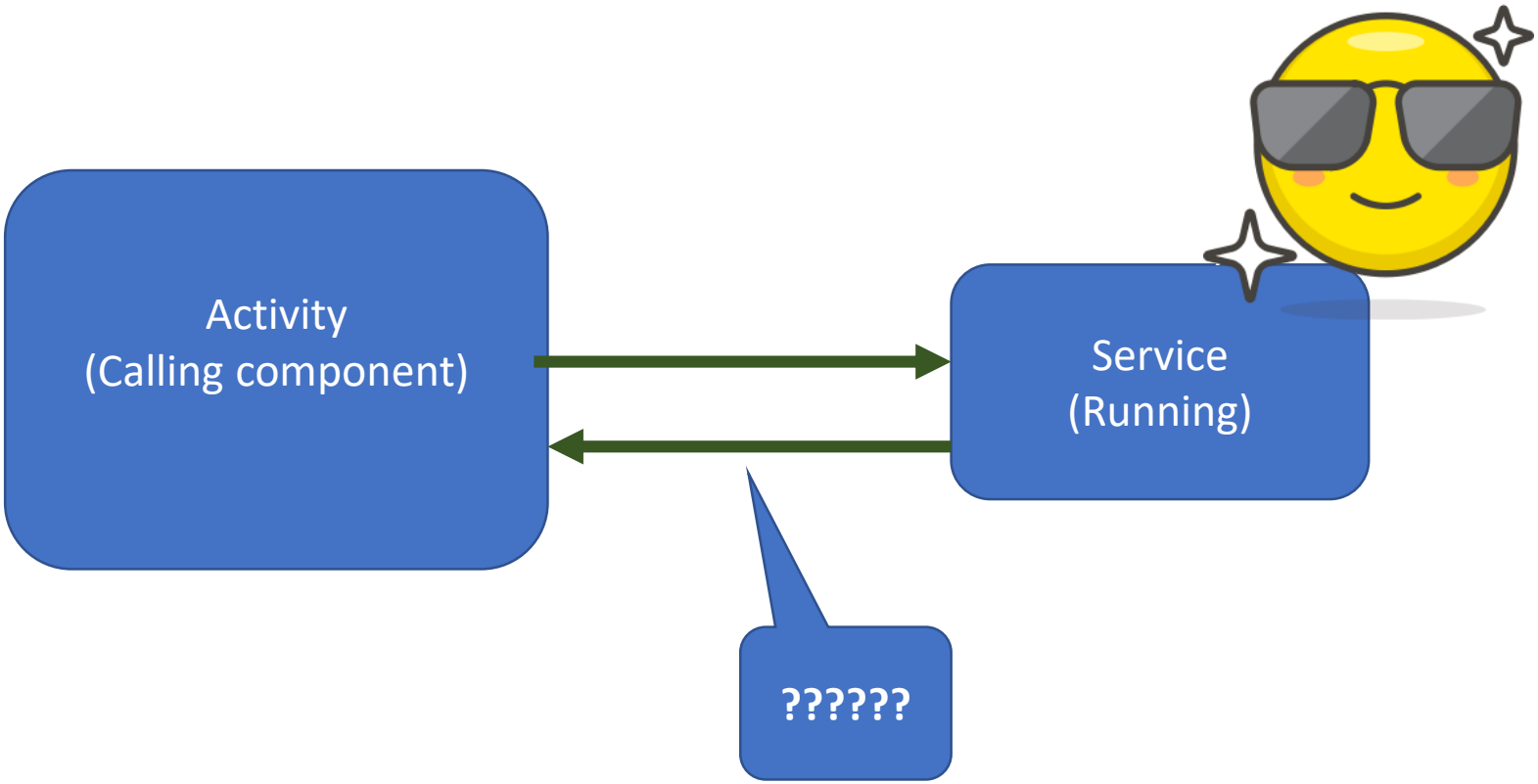
Summer 2, 2019 - CS 4520/CS5520 – Mobile Application Development

Pratheep Kumar Paranthaman, Ph.D.,

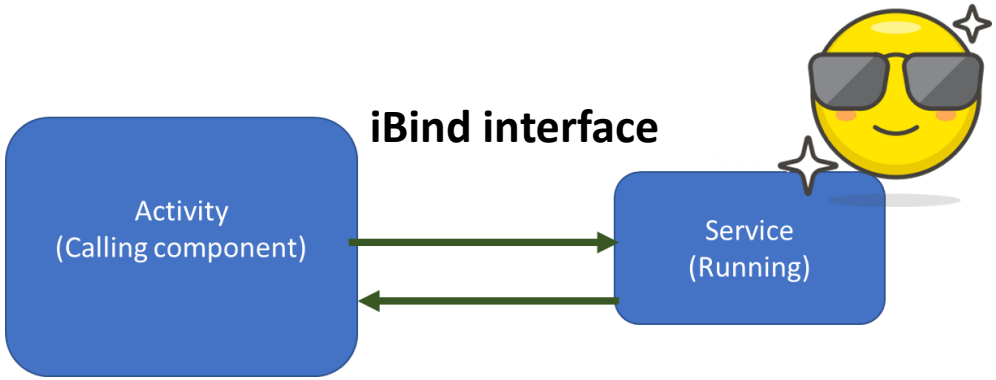
Communication between Service and Android Component

Unbound Services



Bound Services

- Android Component(Activity) binds to the service.
- **Client-Server model** -> Activity(Client) --- Service(Server)
- Destroying all **the calling components** will destroy the **Service** as well(as they are tied to each other).
- **Two-way communication** is possible -> Calling component can extract data from service.



Ideal scenarios for using Bound Services

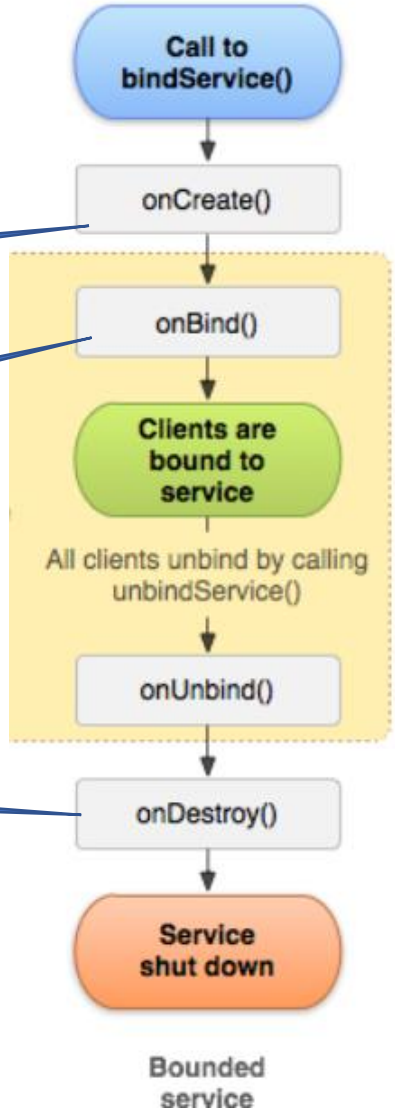
- Service is private to your application
- Client and server run on same process

Bound Service Lifecycle

Bound Service-Created

Binds to the client

Destroyed when the client is killed



Analogy????

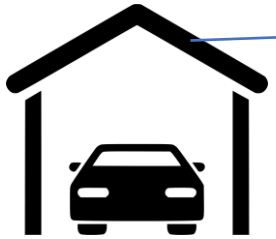
Creating a Bound Service

- In your service, create an instance of Binder that does one of the following:
- Contains public methods that the client can call.
- Returns the current Service instance, which has public methods the client can call.
- Returns an instance of another class hosted by the service with public methods the client can call.
- Return this instance of Binder from the onBind() callback method.
- In the client, receive the Binder from the onServiceConnected() callback method and make calls to the bound service using the methods provided.

[developer.android.com/guide/components/services]

Analogy????

Analogy



-→ **Garage** → Service Class

→ **Ladder** → iBinder Interface

→ **Key** → MyServiceBinderKey (instance of your Service)

→ **Car**→ Methods(Play and Pause) in Service

Bound Services – server side

- **Configure the MyBoundService Class**

  - Create the Play and Pause methods for starting Media Player.
  - Declare iBinder Interface and connect it to the instance of this service

```
public IBinder ibinder = new MyServiceBinder();
```

Creating iBinder interface

  - Generate the instance of this service within MyBoundService

```
public class MyServiceBinder extends Binder{
    MyBoundService getService(){
        return MyBoundService.this;
    }
}
```

Create an instance of this service and attach it to iBinder

  - Make sure to return iBinder in onBind()

Binding activity and Service – client side

- Create a Service Connection and connect with iBinder Interface from Service (use onServiceConnected)

```
public void onServiceConnected(ComponentName name, IBinder service) {
    //connect iBinder to activity
    MyBoundService.MyServiceBinder connectBinder = (MyBoundService.MyServiceBinder)service;
    //set the iBinder interface
   myBoundService = connectBinder.getService();

}
```

Connecting to iBinder

Get the instance of Service

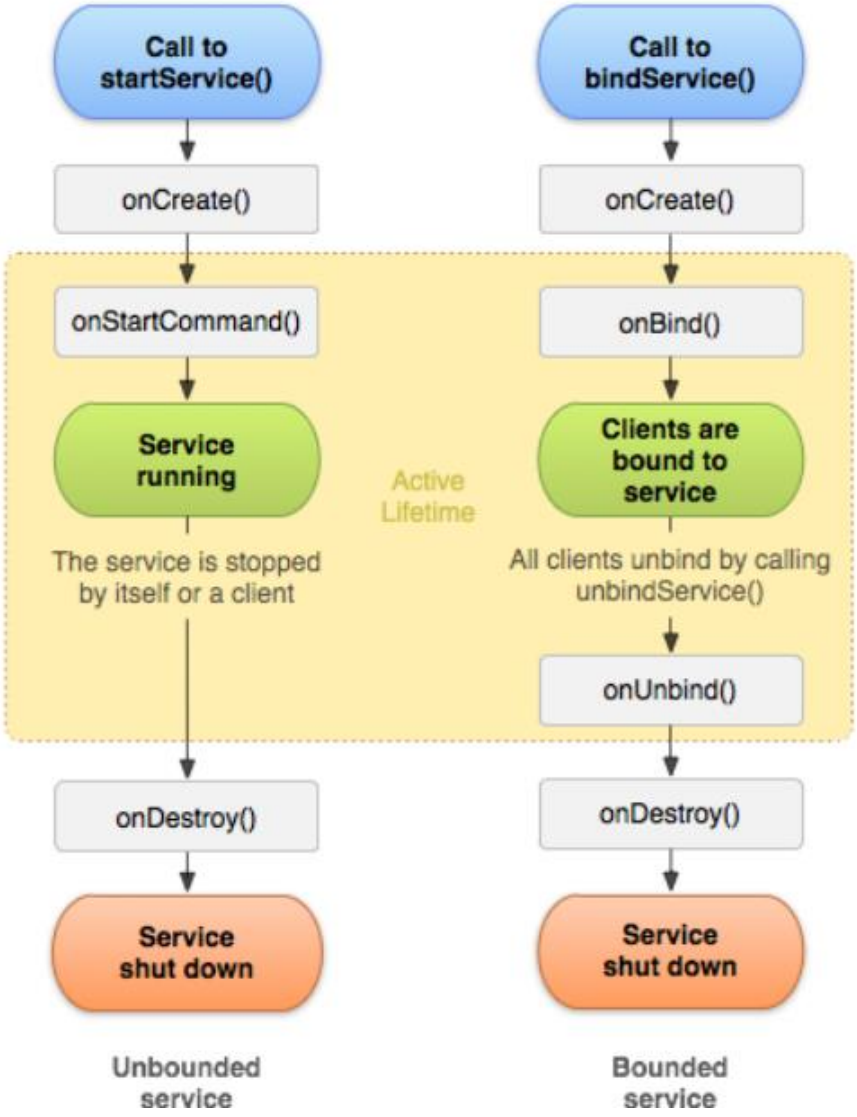
- Create an Intent and use BindService

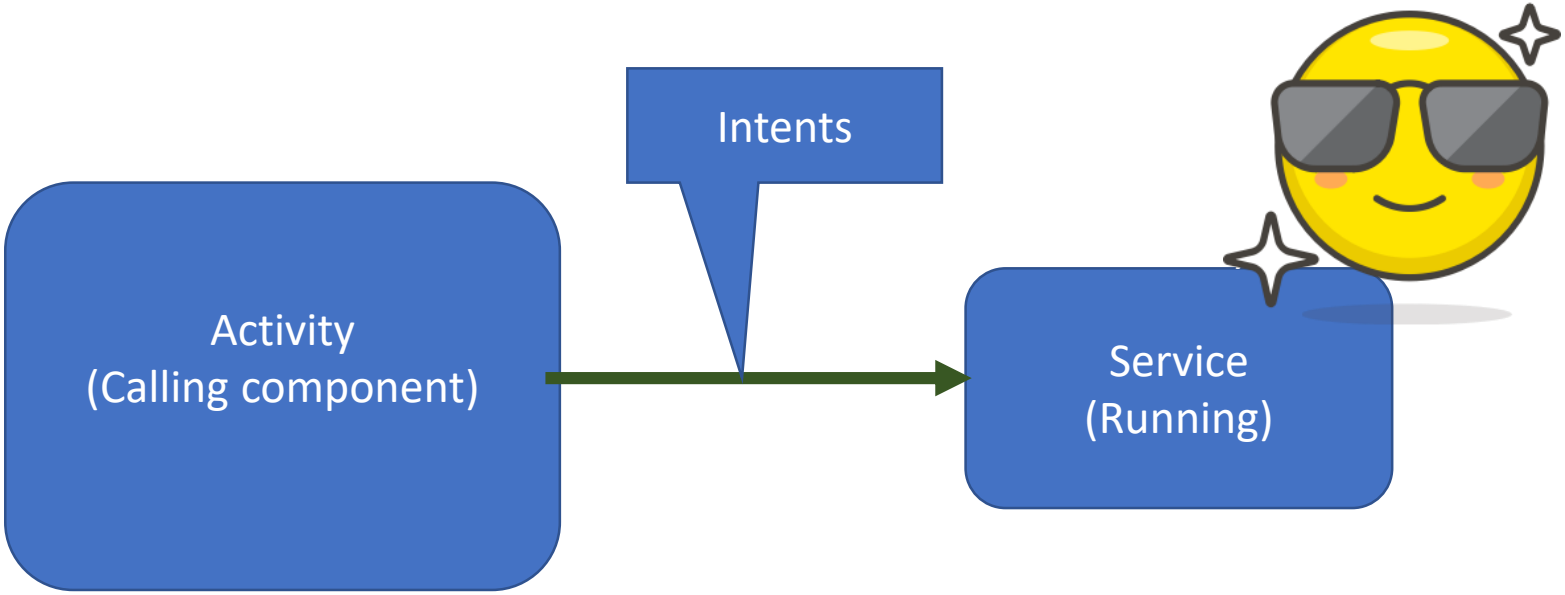
```
Intent intent = new Intent(this,MyBoundService.class);

bindService(intent,myServiceConnectionToGarage,Context.BIND_AUTO_CREATE)
```

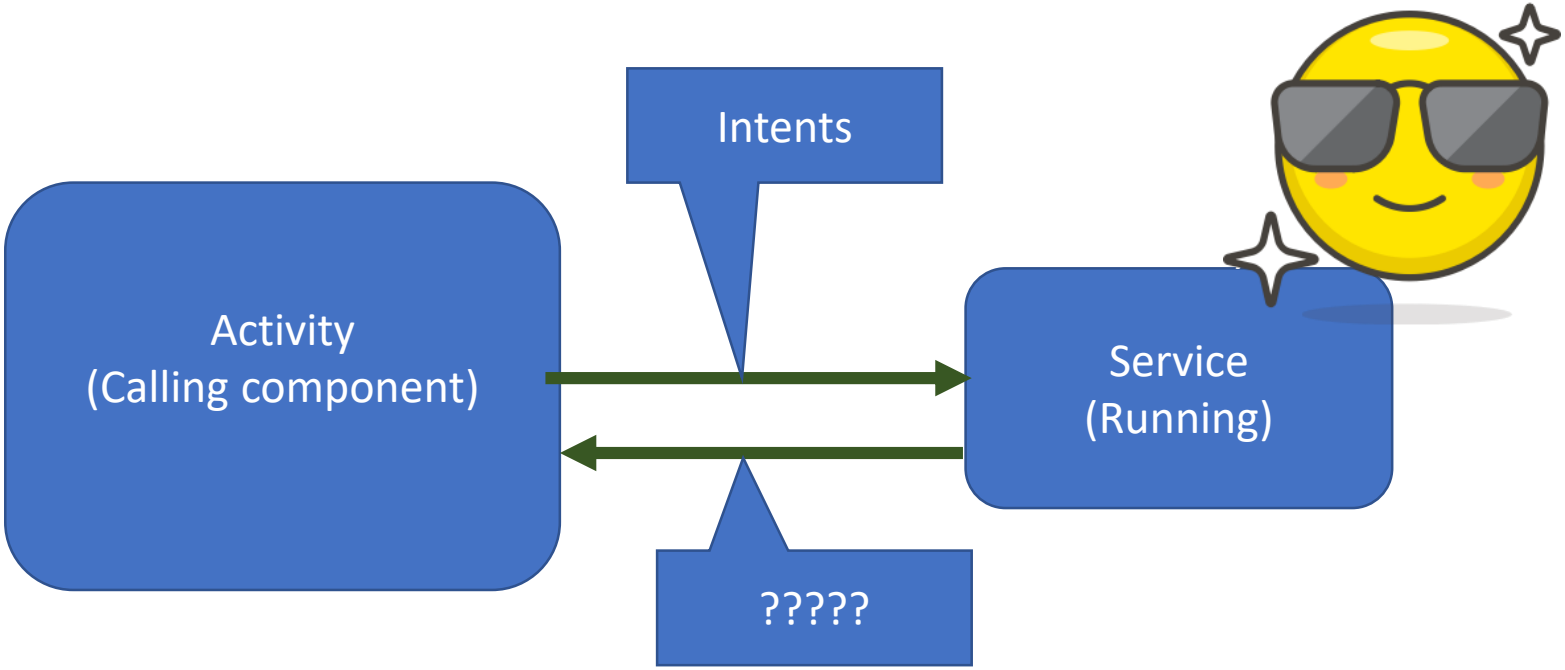
Intent setup to establish connection

Binds the service and Activity

Service Lifecycle



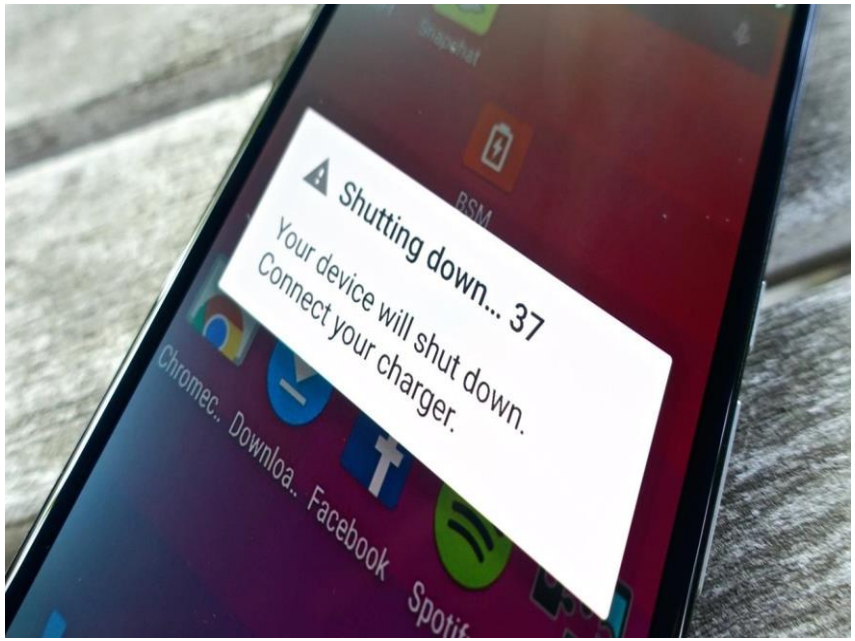

Intents
Activity
(Calling component)
Service
(Running)


Intents
Activity
(Calling component)
Service
(Running)
?????

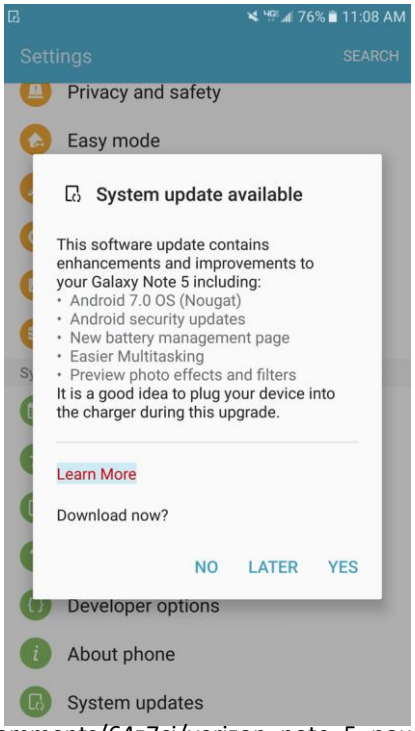
Broadcast Receivers

Android Components

- **Broadcast receivers**
  - Apps can send or receive broadcast messages from Android OS or other apps



[developer.android.com/guide/components/broadcasts]



[reddit.com/r/galaxynote5/comments/64z7cj/verizon_note_5_nougat_update_available/]

Publisher – Subscriber Pattern(Pub-Sub)

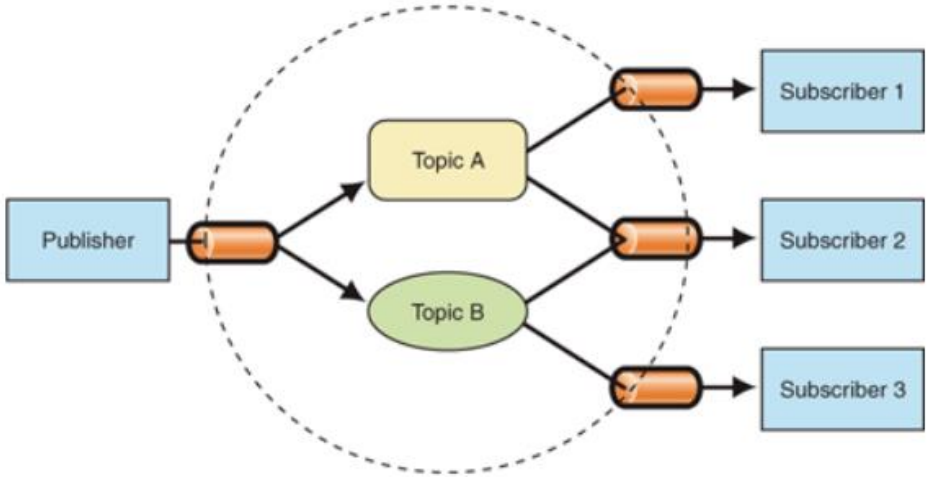
Publisher:

- Typically the one who broadcasts messages.

- Do not send messages directly to the receivers.
- Pass in a message to a channel

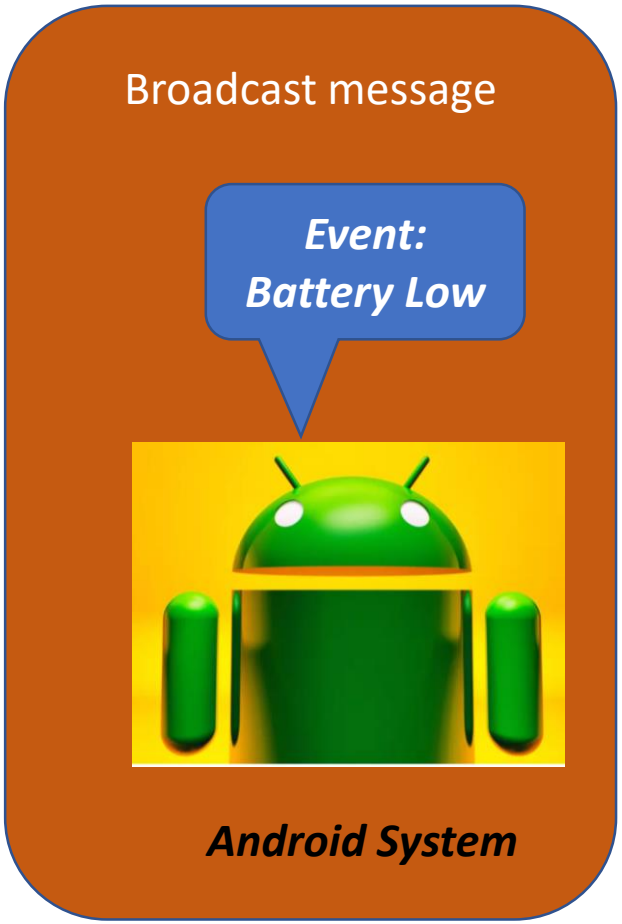
Subscriber (receivers):

- Receives messages only if they had subscribed to a particular channel of interest



[medium.com/easyread/difference-between-pub-sub-pattern-and-observable-pattern-d5ae3d81e6ce]

Broadcast receivers in Android



Broadcast receivers

- Communication tool to share information between apps and also within the components in an App. (States, events)
- Apps or the app components register for receiving broadcasts

Local Broadcast

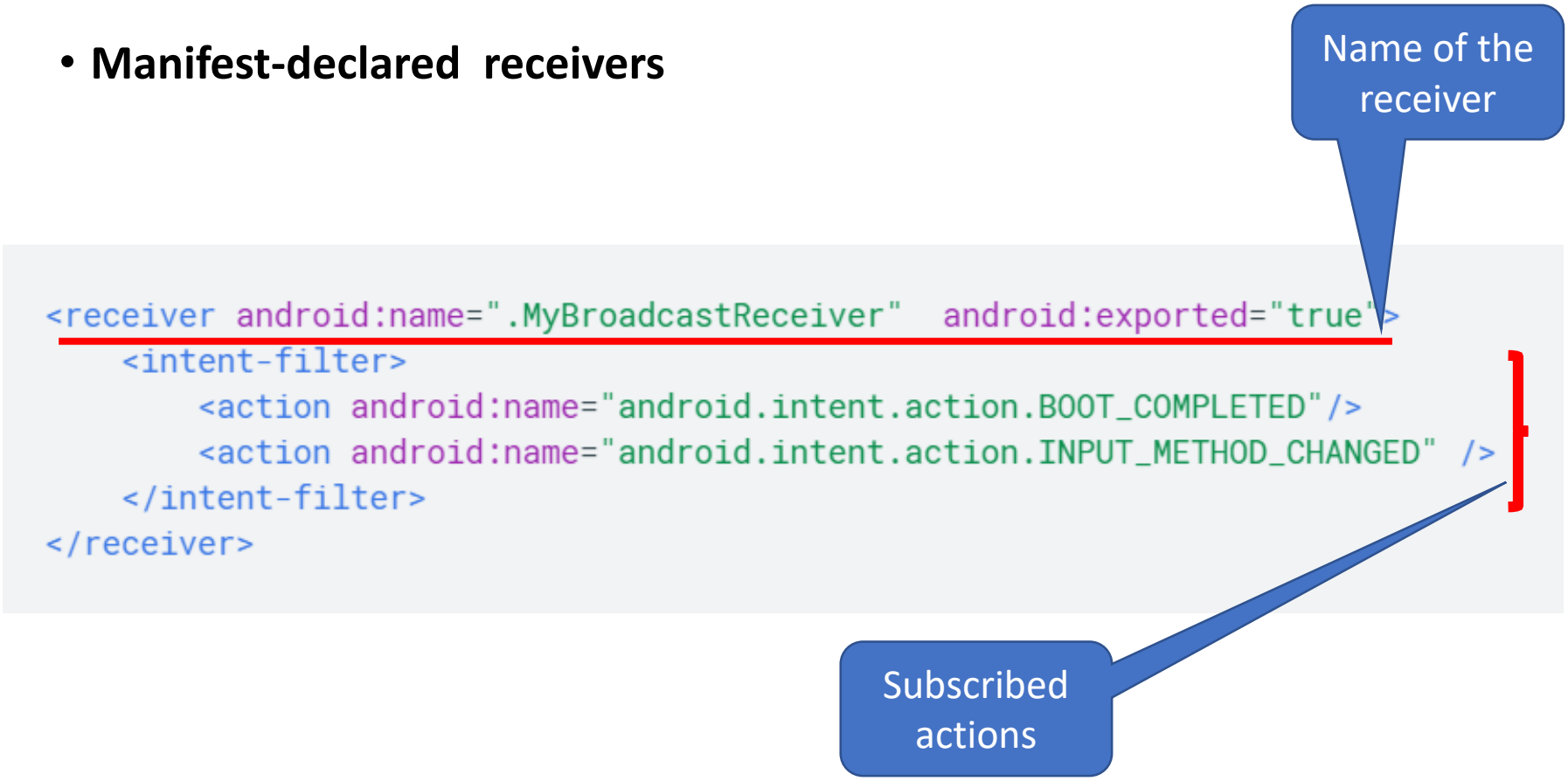
- Broadcast and receive messages between the android components within an app
- Uses ***LocalBroadcastManager during broadcasting process***

Global broadcast

- Broadcast and receive messages from system or other applications.

Registering receivers

- **Manifest-declared receivers**



[developer.android.com/guide/components/broadcasts]

Broadcast receivers

- **Context-registered receivers**

```
BroadcastReceiver br = new MyBroadcastReceiver();
```

Create an instance of the receiver

```
IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
    filter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);
    this.registerReceiver(br, filter);
```

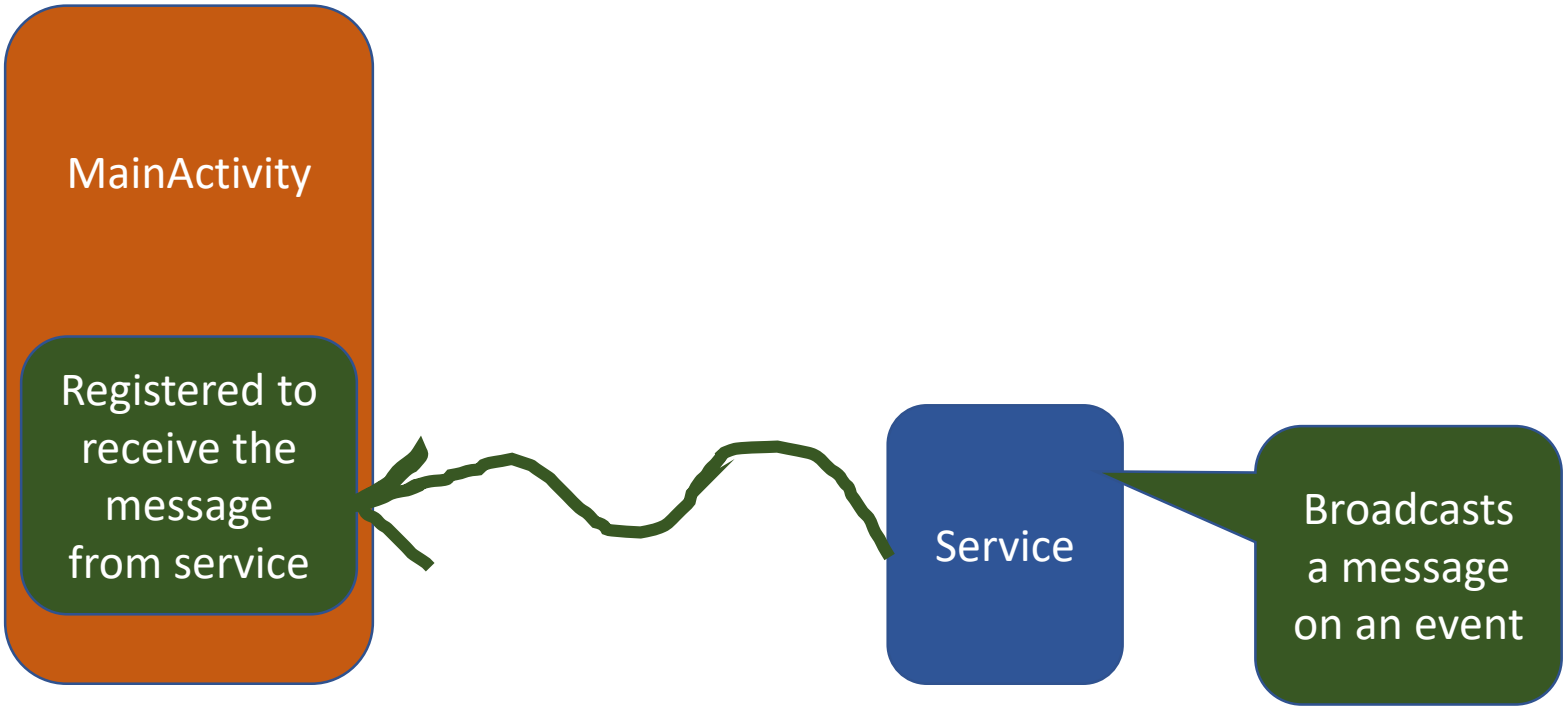
Subscribed action in the intent filter

[developer.android.com/guide/components/broadcasts]

Steps for Creating broadcast receivers

- Publisher (Component/Other app)
  - Broadcasts a message based on the event
- Subscriber (Component/Other app)
  - Creates an instance of receiver
  - Registers to the broadcast event
  - Receives the message when then event is triggered within the publisher

Broadcast Receivers in Android



Steps for implementing broadcast receivers

- **Broadcast**
- Event
  - Create an action (intent)

```
Intent localBroadcast = new Intent("ACTION_NAME");
```

  - Pass in the data from the component → message to be broadcasted

```
localBroadcast.putExtra("Key",Values);
```

  - Start broadcasting → in our case it's going to be a local broadcast

```
LocalBroadcastManager.getInstance(this).sendBroadcast(localBroadcast)
```

Steps for implementing broadcast receivers

- **Receive**

- Declare the Broadcast receiver in the component

```
private BroadcastReceiver mylocalreceiver = new BroadcastReceiver()
```

- Register the receiver
  - Use IntentFilter to register for the Broadcast

```
IntentFilter localIntentFilter = new IntentFilter("ACTION_NAME");

LocalBroadcastManager.getInstance(this).registerBroadcast(mylocalreceiver,localIntentFilter)
```

Unregister the receiver → when the activity/component is stopped

```
LocalBroadcastManager.getInstance(this).unRegisterBroadcast(mylocalreceiver ,localIntentFilter)
```

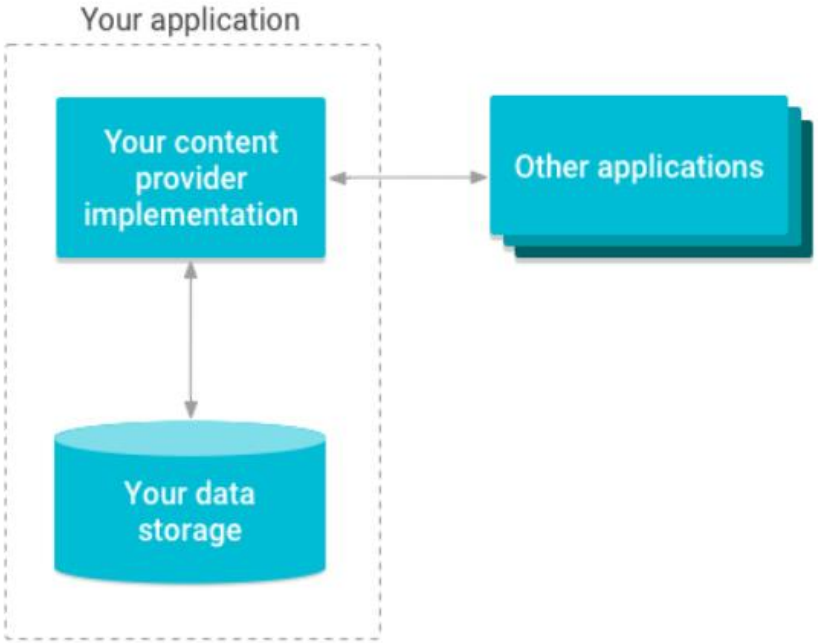
Steps for implementing broadcast receivers

- Extract the data in onReceive() of your broadcast
  - Populate the results on your Receiver side component

Content Providers

Android Components

- **Content Providers**
  - Provides data from one application to another(the one which requests)
  - Interface that connects data from one app with code running in another app



[developer.android.com/guide/topics/providers/content-providers]

References

- Content Providers - Chapter 8: https://learning.oreilly.com/library/view/beginning-android-programming/9781118705599/c08.xhtml#c8