MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

B01_e

认识进程与IPC架构(e)

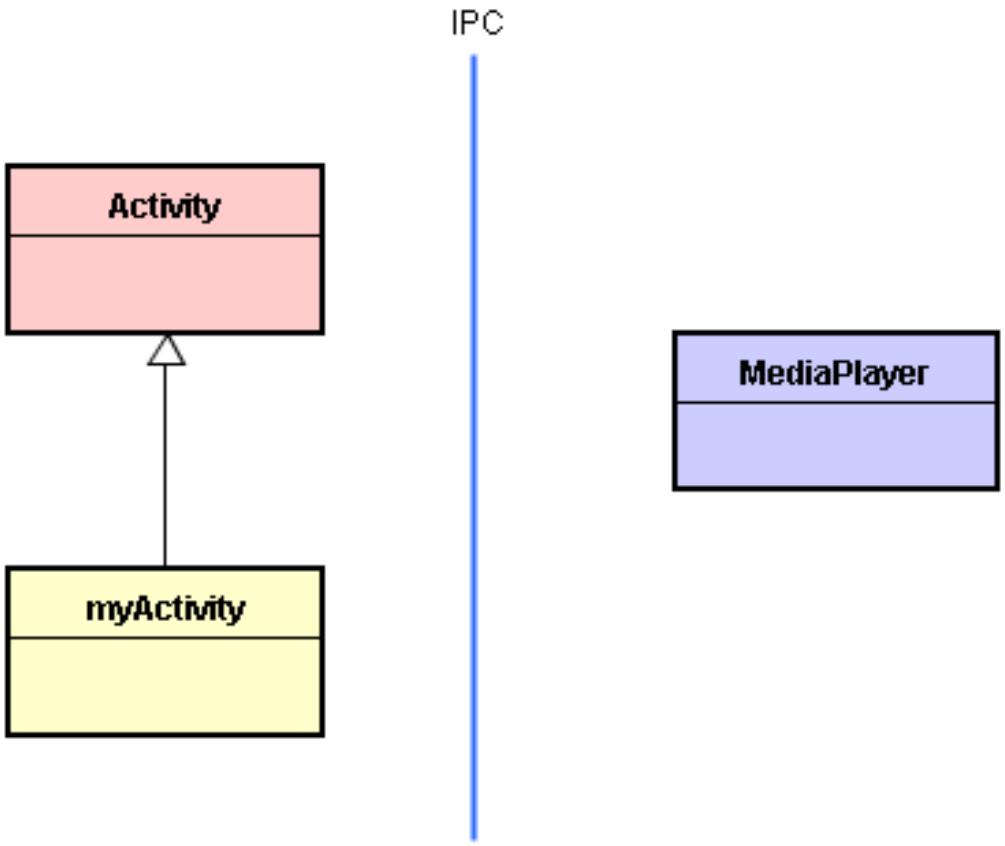
By 高煥堂

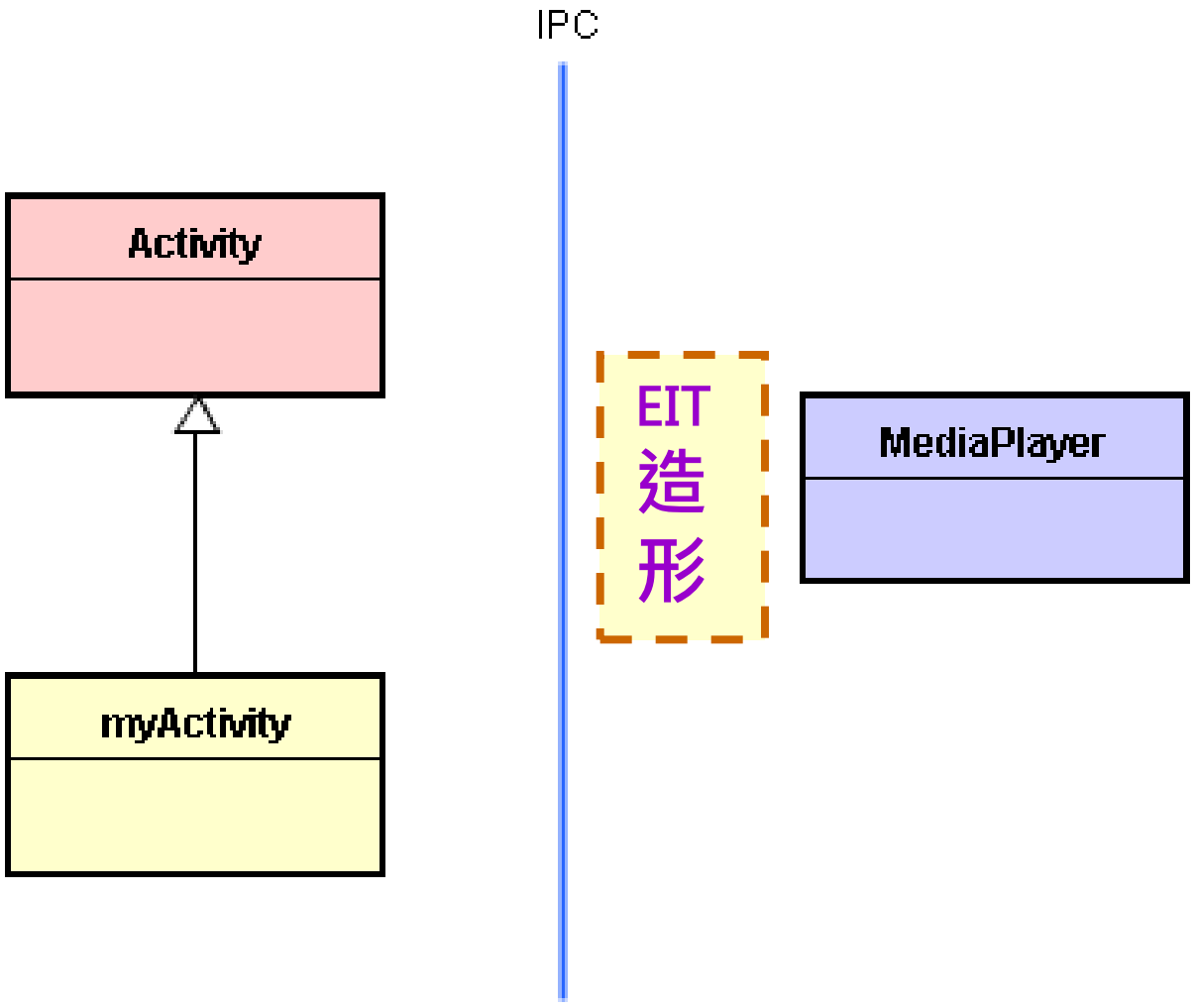
6、IPC通信的三步骤

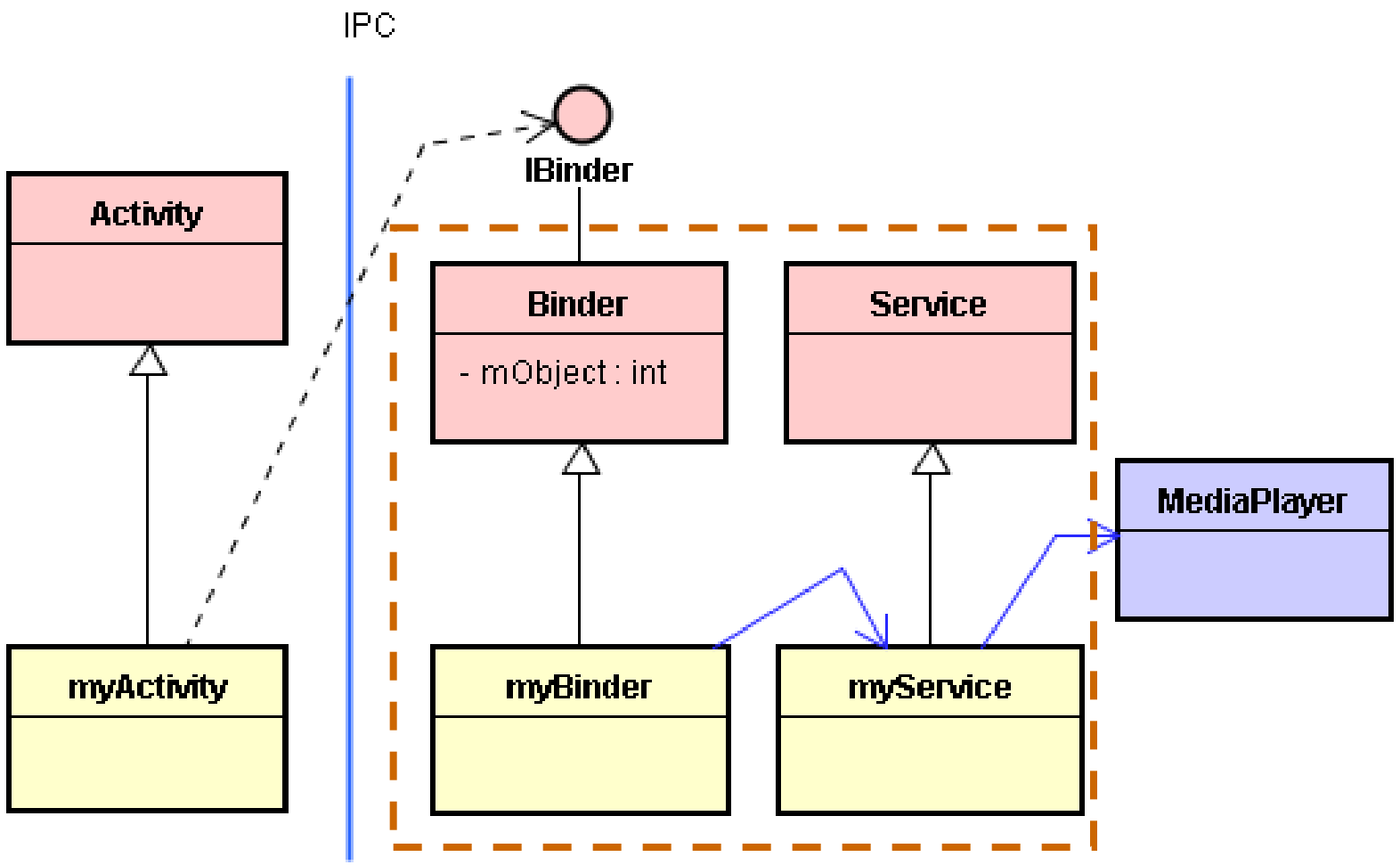
牛郎与织女的约会

<配对、建乔、相会>

- 兹拿刚才的MP3播放范例，来说明其详细步骤。Activity类想跨进程去调用MediaPlayer播放引擎，如下图所示：




IPC
Activity
myActivity
EIT
造
形
MediaPlayer


IPC
IBinder
Activity
Binder
- mObject : int
Service
MediaPlayer
myActivity
myBinder
myService

EIT造形的<組合>

牛郎与织女的约会三步骤：

Step-1. 配对。

Step-2. 建乔。

Step-3. 相会。

其IPC通信的三个步驟是：

Step-1. Activity使用startService()函數來啟動Service。

Step-2. Activity調用bindService()来绑定Service。亦即，Activity建立与Service之间的连结(Connection)。

Step-3. Activity調用IBinder接口的transact()函数，透过底层Binder Driver驱动而间接調用到Binder基类的execTransact()函数，转而調用 myBinder的onTransact()函数。

程序码：

```
// myActivity.java
// ……..
public class myActivity extends Activity
                                implements OnClickListener {
   public void onCreate(Bundle icicle) {
     // ……..
     startService(new Intent("com.misoo.pk01.REMOTE_SERVICE"));
     bindService(new Intent("com.misoo.pk01.REMOTE_SERVICE"),
                                mConnection,
     Context.BIND_AUTO_CREATE);
      //……..
   }
```

```
    private ServiceConnection mConnection =
                new ServiceConnection() {
                        public void onServiceConnected(
                          ComponentName className, IBinder ibinder) {
                             mb = ibinder;
                }
  };
  //……..
  public void onCreate(Bundle icicle) {
          //……..
          startService(new
                  Intent("com.misoo.pk01.REMOTE_SERVICE"));
          //……..
   }
}}
```

```
// myService.java
// ..........
public class myService extends Service {
      private IBinder mb = null;
        //……….
        @Override public void onStart()
              {  mb = new myBinder();     }
        @Override public IBinder onBind(Intent intent)
              {  return mb;  }
}
```

```
//  myBinder.java
// ……..
public class myBinder extends Binder{
    private Context ctx;

    public myBinder(Context cx)
                { ctx= cx; }
    @Override
     public boolean onTransact(int code,
                              Parcel data, Parcel reply, int flags)
        {
        // ……
      }
}
```

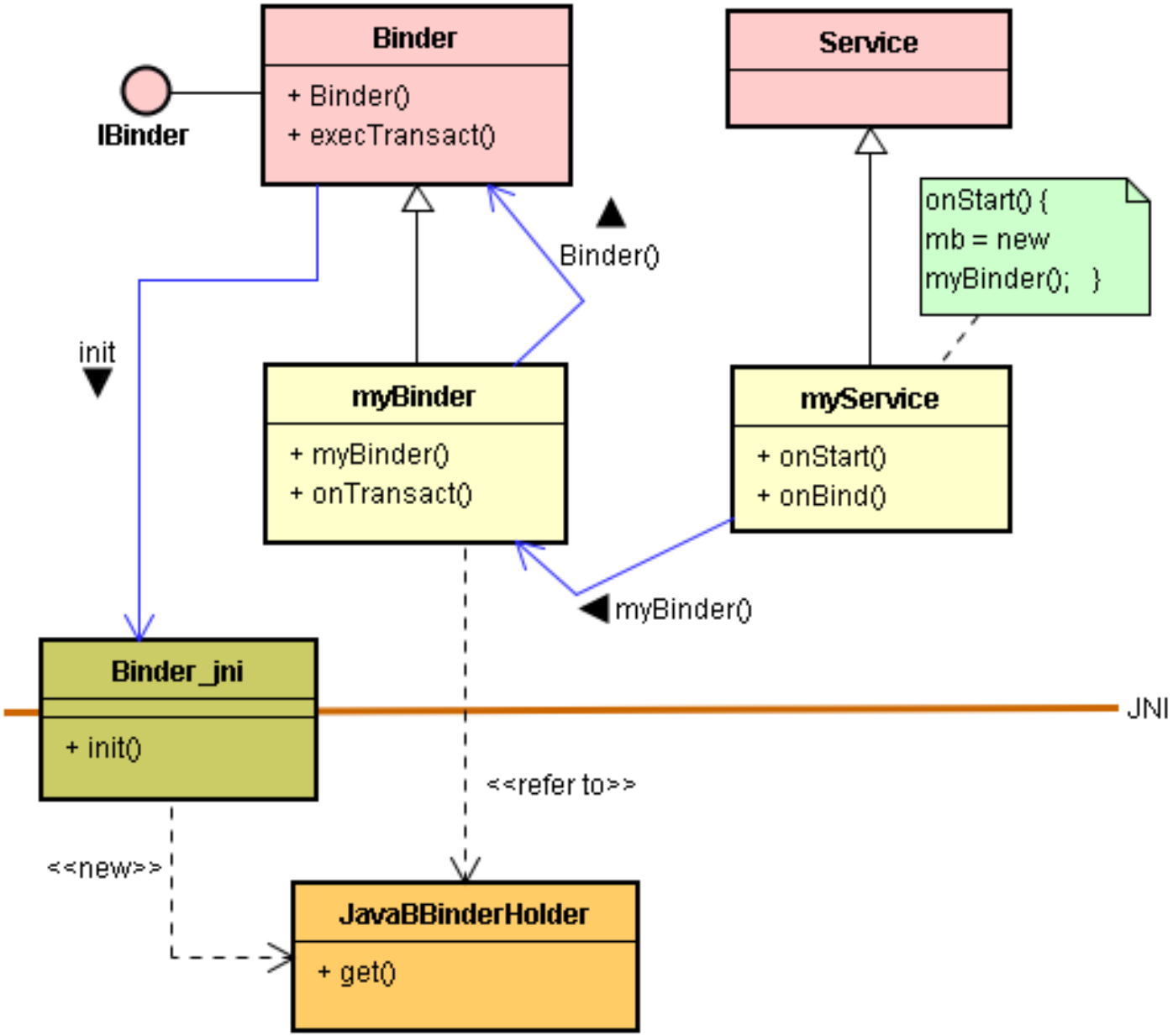
Step-1：調用startService()

- 当myActivity調用startService()时，就調用Service.onStart()函数，执行到指令：

  mb = new myBinder()

- 接着，調用myBinder()建构式(Constructor)；进而調用父类别Binder()建构式，转而調用JNI本地的init()函数。

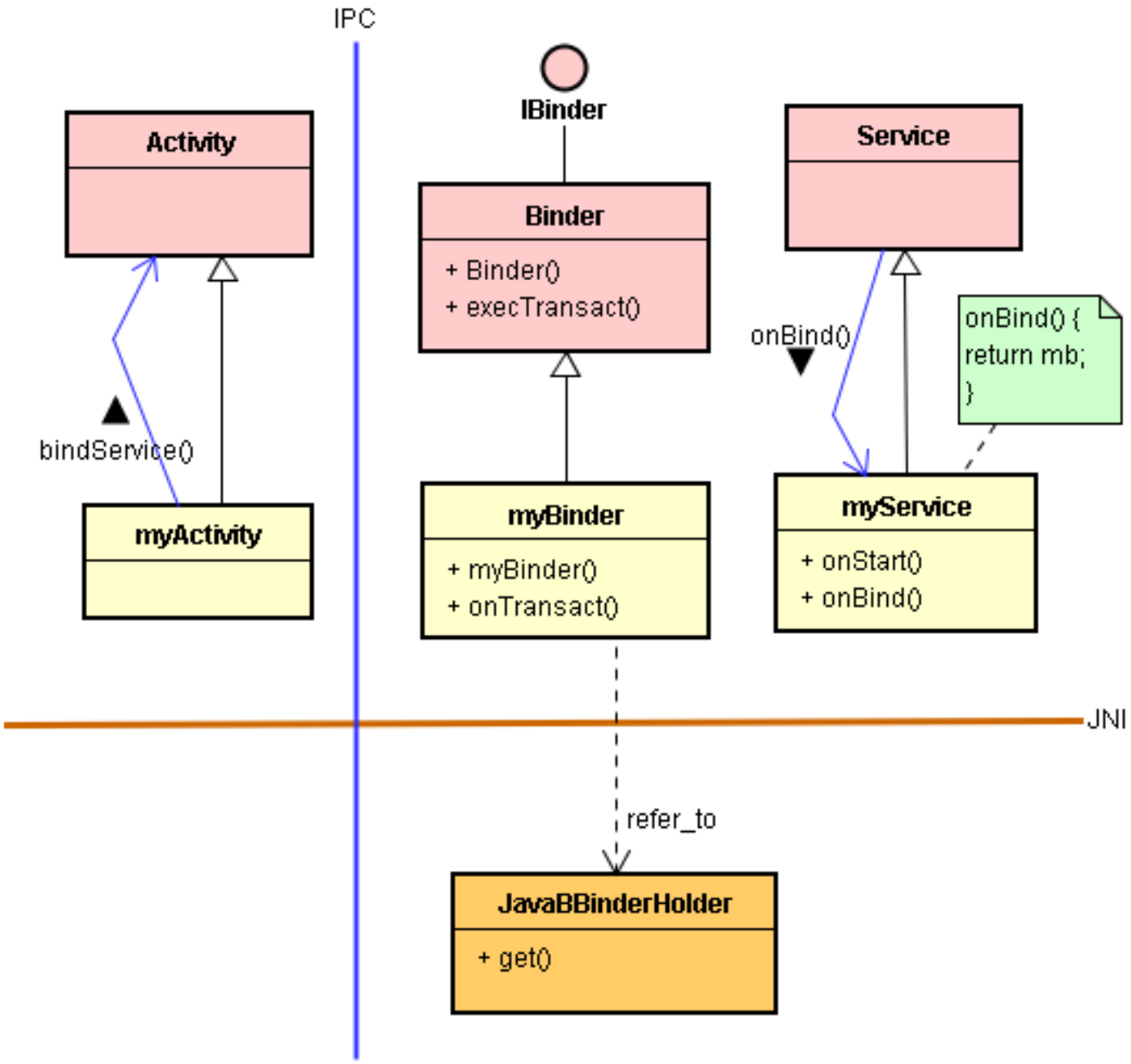
- 此刻执行init()函数时，会在C/C++层里诞生一个JavaBBinderHolder类别的对象，并且将这JavaBBinderHolder对象的指针存入到myBinder对象里，让myBinder对象指向JavaBBinderHolder对象。


Binder
+ Binder()
+ execTransact()
IBinder
Service
onStart() {
mb = new
myBinder(); }
Binder()
init
myBinder
+ myBinder()
+ onTransact()
myService
+ onStart()
+ onBind()
myBinder()
Binder_jni
+ init()
JNI
<<refer to>>
<<new>>
JavaBBinderHolder
+ get()

Step-2：調用bindService()

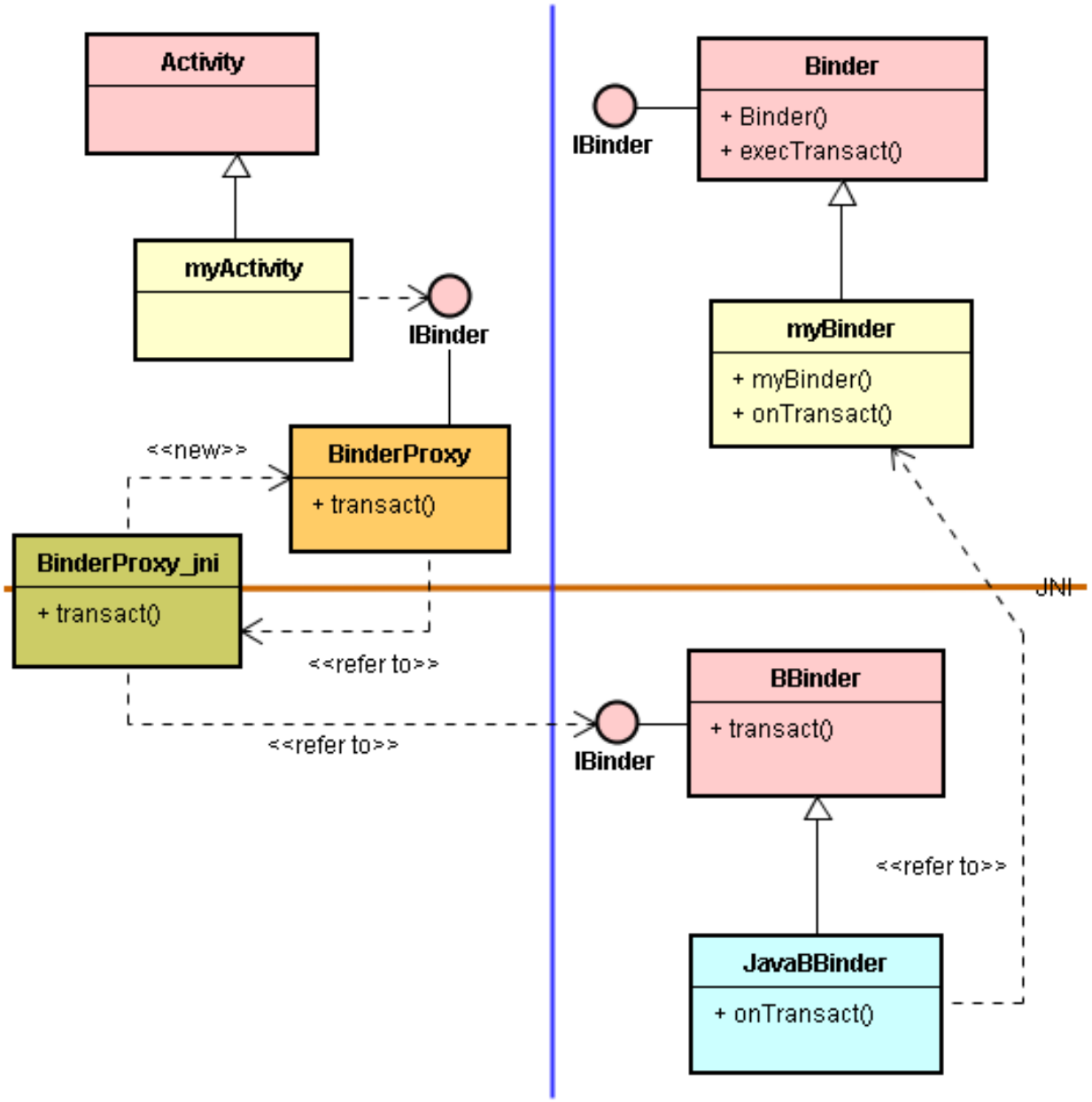
- 目前，已经执行完startService()函数了。接着，myActivity继续調用bindService()函数，想去绑定Service服务。如果找到该服务，且它尚未被任何Client所绑定的话，就会調用myService的onBind()函数。此时，执行到指令：return mb;
- 如下述的程序码：

- 这onBind()函数将mb(即myBinder对象的IBinder接口)回传Android框架(其实是框架里的AMS(ActivityManagerService)。


IPC
IBinder
Activity
Binder
+ Binder()
+ execTransact()
Service
onBind()
onBind() {
return mb;
}
bindService()
myActivity
myBinder
+ myBinder()
+ onTransact()
myService
+ onStart()
+ onBind()
JNI
refer_to
JavaBBinderHolder
+ get()

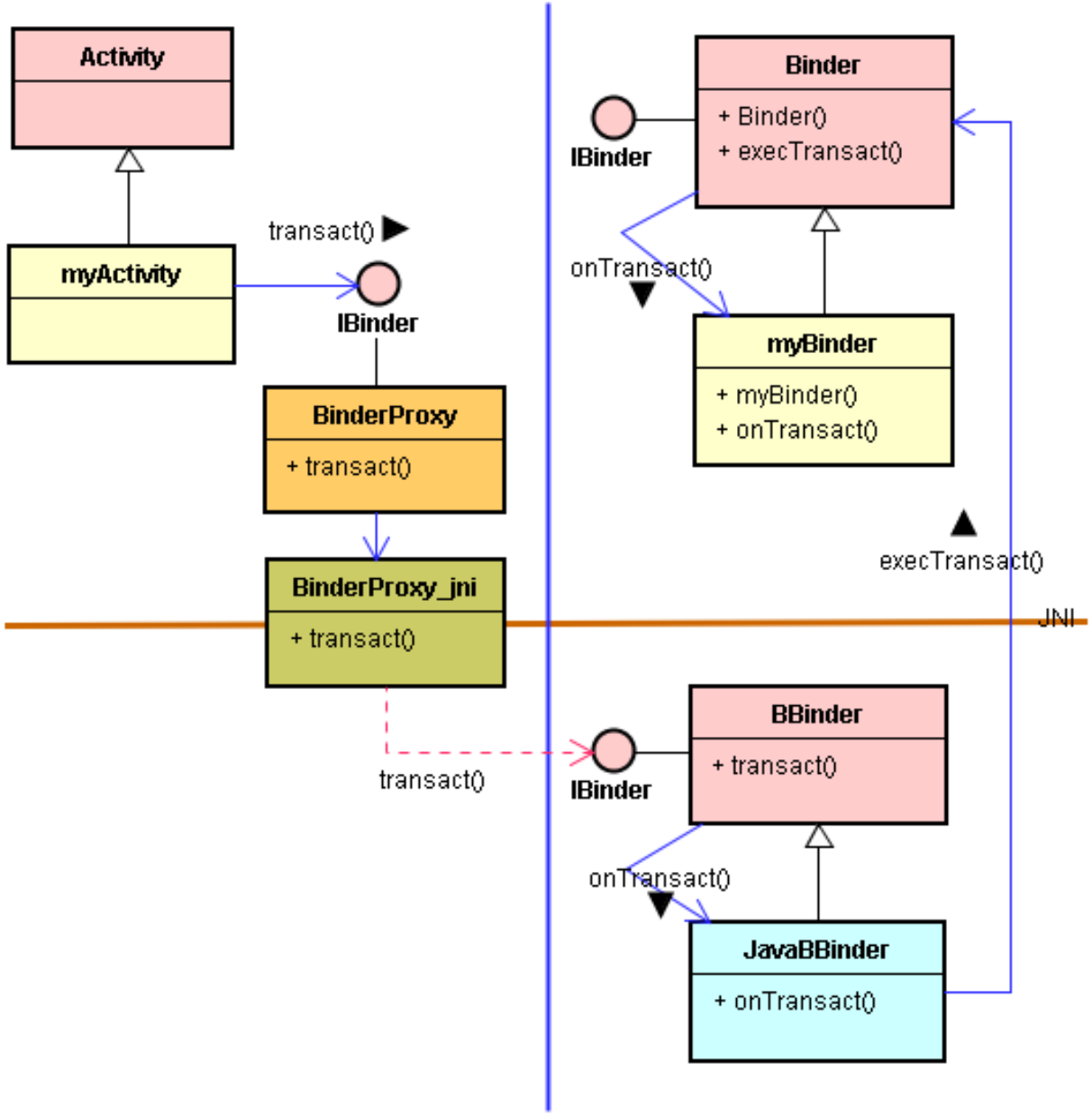
- 当 AMS接到回传来的myBinder对象指针(即其IBinder接口)时，就可以找到其在C/C++层所对映的JavaBBinderHolder对象。接着，右調用JavaBBinderHolder的get()函数去诞生一个JavaBBinder对象。

- 接着，AMS在Client端进程的java层里诞生一个BinderProxy对象来代表JavaBBinder的分身，也就是代表了myBinder的分身。最后将BinderProxy的IBinder接口回传给myActivity。
- 此时完成了跨进程的服务绑定(Bind)，如下图：

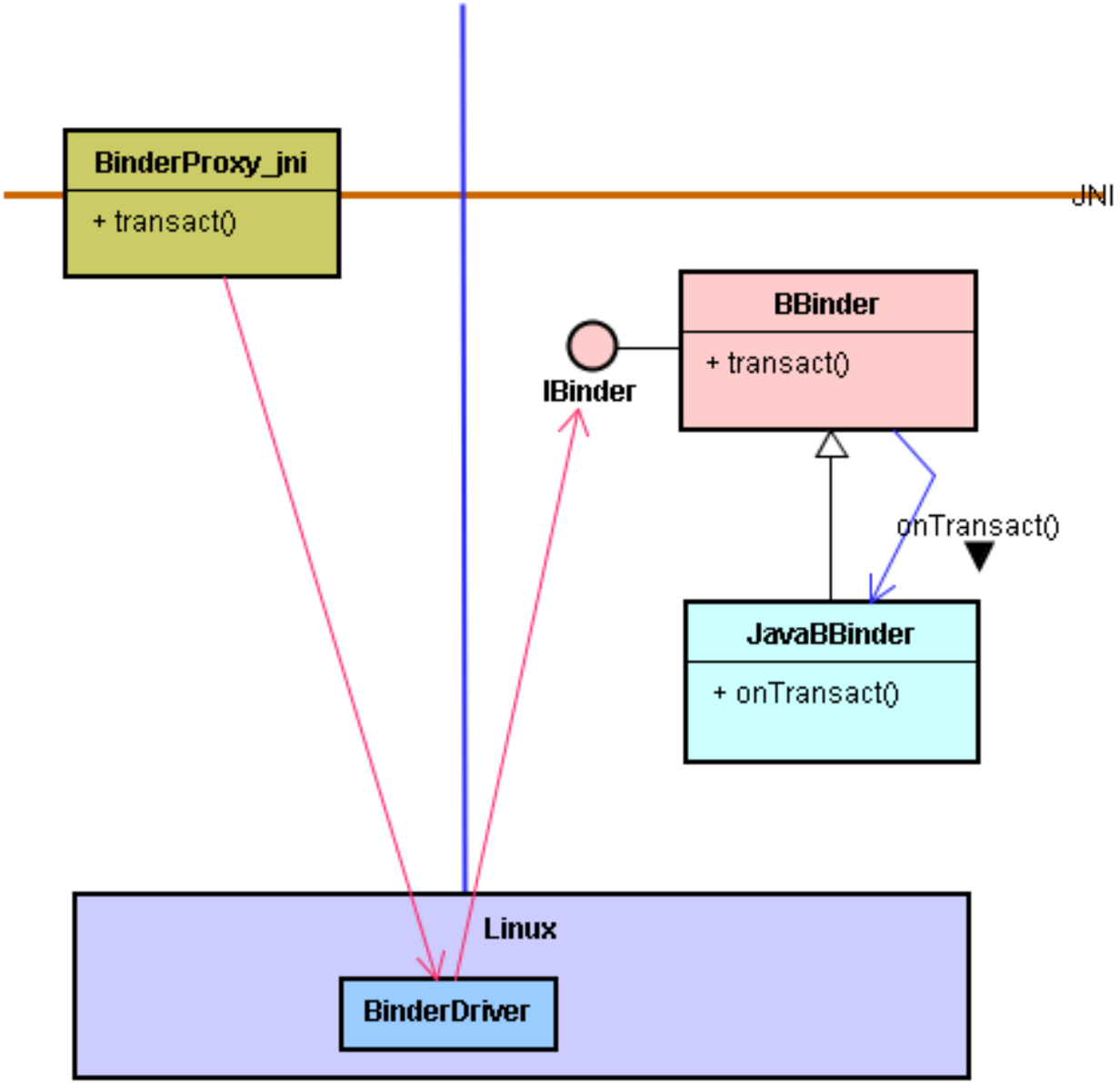

Activity
myActivity
IBinder
<<new>>
BinderProxy
+ transact()
BinderProxy_jni
+ transact()
<<refer to>>
<<refer to>>
IBinder
Binder
+ Binder()
+ execTransact()
myBinder
+ myBinder()
+ onTransact()
JNI
BBinder
+ transact()
<<refer to>>
JavaBBinder
+ onTransact()

Step-3：調用IBinder接口的transact()

- 所谓建好了服务绑定(Bind)之后，就如同建好了跨进程的桥梁。之后，就能随时透过这桥梁而进行从myActivity調用到myService的跨进程IPC通信。绑定了服务之后，就能从myActivity調用BinderProxy(透过IBinder接口)的IBinder接口，执行了transact()函数。如下图：


Activity
myActivity
transact()
IBinder
BinderProxy
+ transact()
BinderProxy_jni
+ transact()
transact()
Binder
+ Binder()
+ execTransact()
IBinder
onTransact()
myBinder
+ myBinder()
+ onTransact()
execTransact()
JNI
BBinder
+ transact()
IBinder
onTransact()
JavaBBinder
+ onTransact()

- 在上图里，从JNI本地模块拉了一条红色虚线，表示这并非直接的通信途径。也就是，实际上是透过底层Binder Driver驱动才調用到BBinder的IBinder接口。如下图：


BinderProxy_jni
+ transact()
JNI
BBinder
+ transact()
IBinder
onTransact()
JavaBBinder
+ onTransact()
Linux
BinderDriver

总结上述Activity与Service之间IPC通信的三个步骤：

- Step-1. 調用startService()。
- Step-2. 調用bindService()。
- Step-3. 調用IBinder接口的transact()。

~ Continued ~