
Яндекс

Яндекс

Как общаются приложения

Антон Дудаков
Лаборатория встраиваемых автомобильных решений

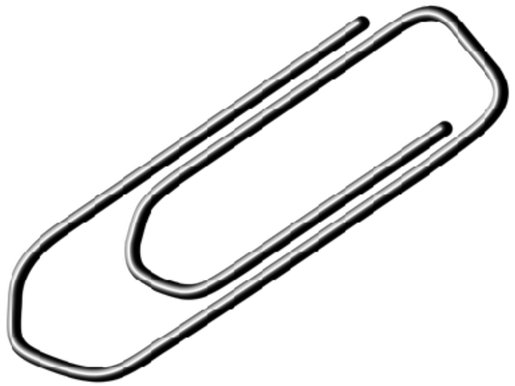
Задачи решаемые взаимодействием

- запуск приложениями друг друга (и получение результатов работы)
- обмен уведомлениями о событиях
- предоставление данных
- предоставление функций
- предоставление элементов пользовательского интерфейса

API предоставляемый Anroid

- startActivity, startActivityForResult
- sendBroadcast
- ContentProvider
- AIDL (+Messenger)
- Специализированные средства, обёрнутые в системные сервисы (widgethost, mediasession, remote controller, notification listener)

Под капотом



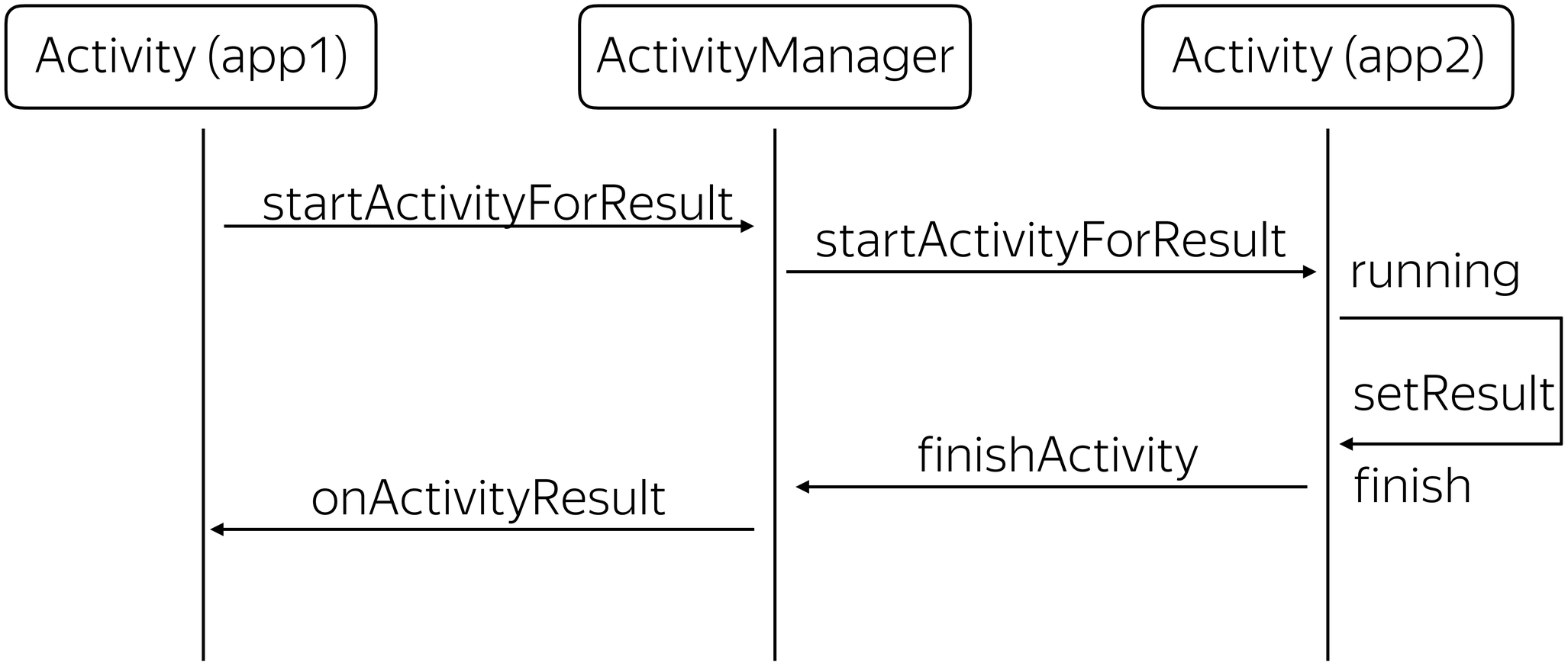
Binder

/dev/binder

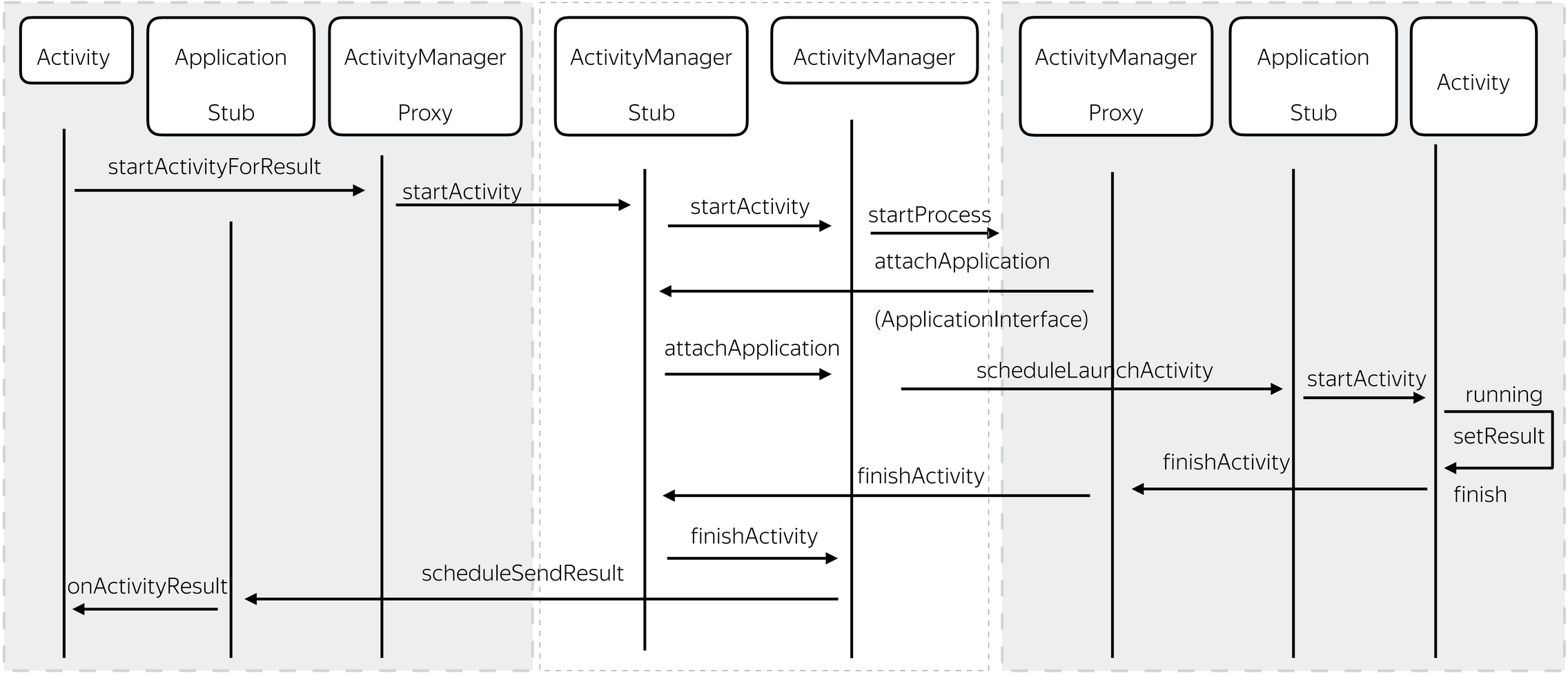
startActivityForResult

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);

startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
```

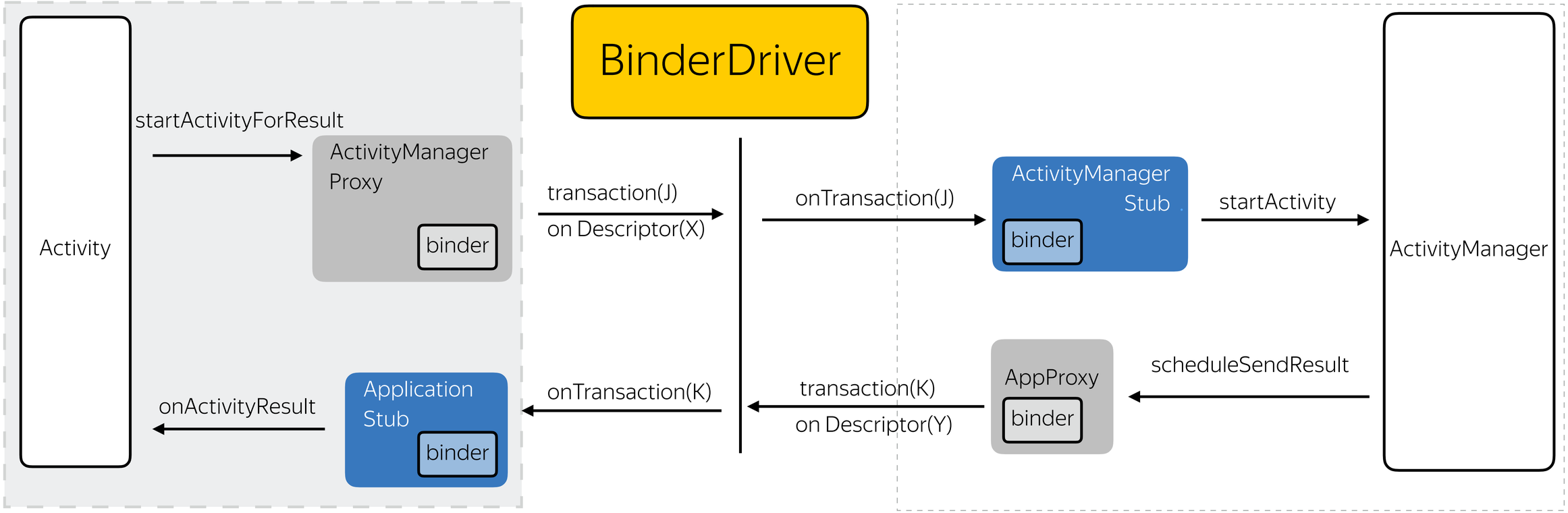
Binder на примере startActivityForResult



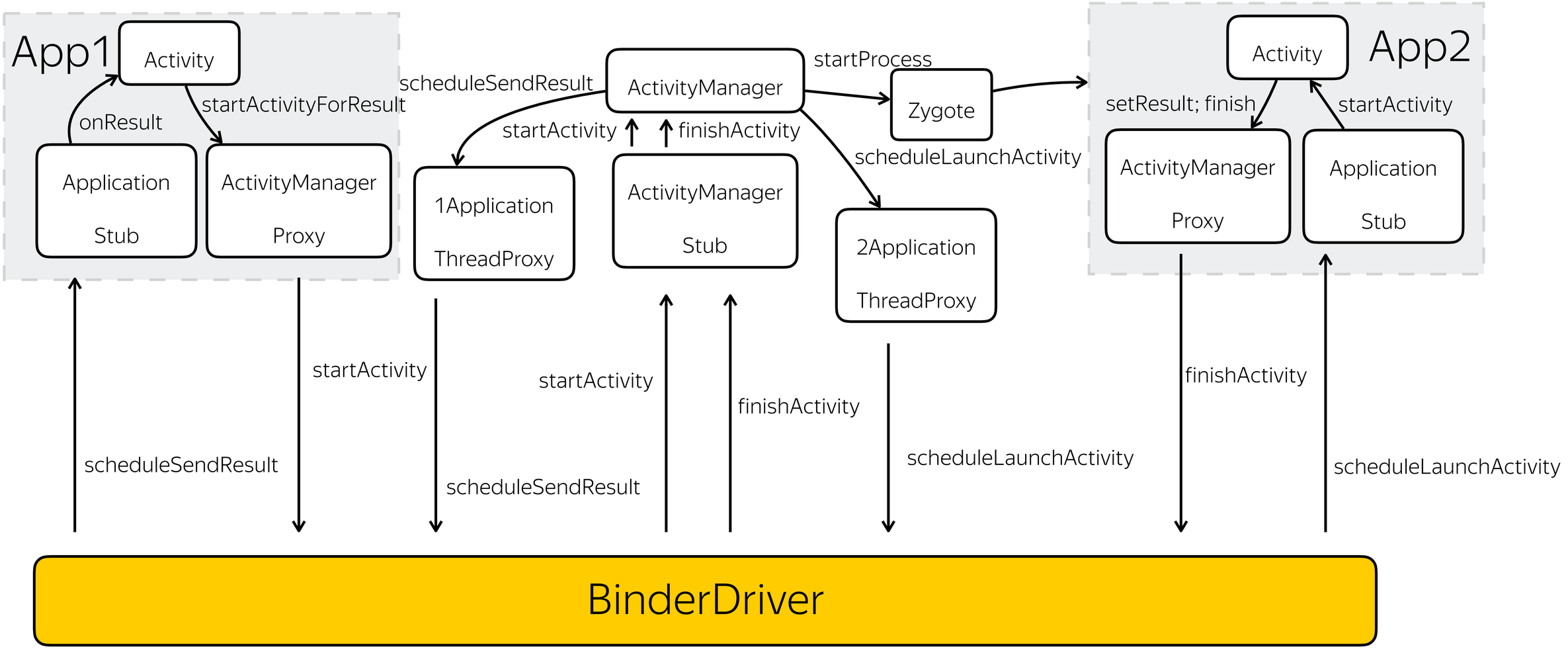
Binder на примере startActivityForResult 2



Приложения и ActivityManager находятся в разных процессах



Binder на примере startActivityForResult 3



android.os.TransactionTooLargeException

/frameworks/native/libs/binder/ProcessState.cpp

```
#define BINDER_VM_SIZE ((1*1024*1024) - (4096 *2))
…

// mmap the binder, providing a chunk of virtual address
// space to receive transactions.
mVMStart = mmap(0, BINDER_VM_SIZE, PROT_READ,
                MAP_PRIVATE | MAP_NORESERVE, mDriverFD, 0);
```

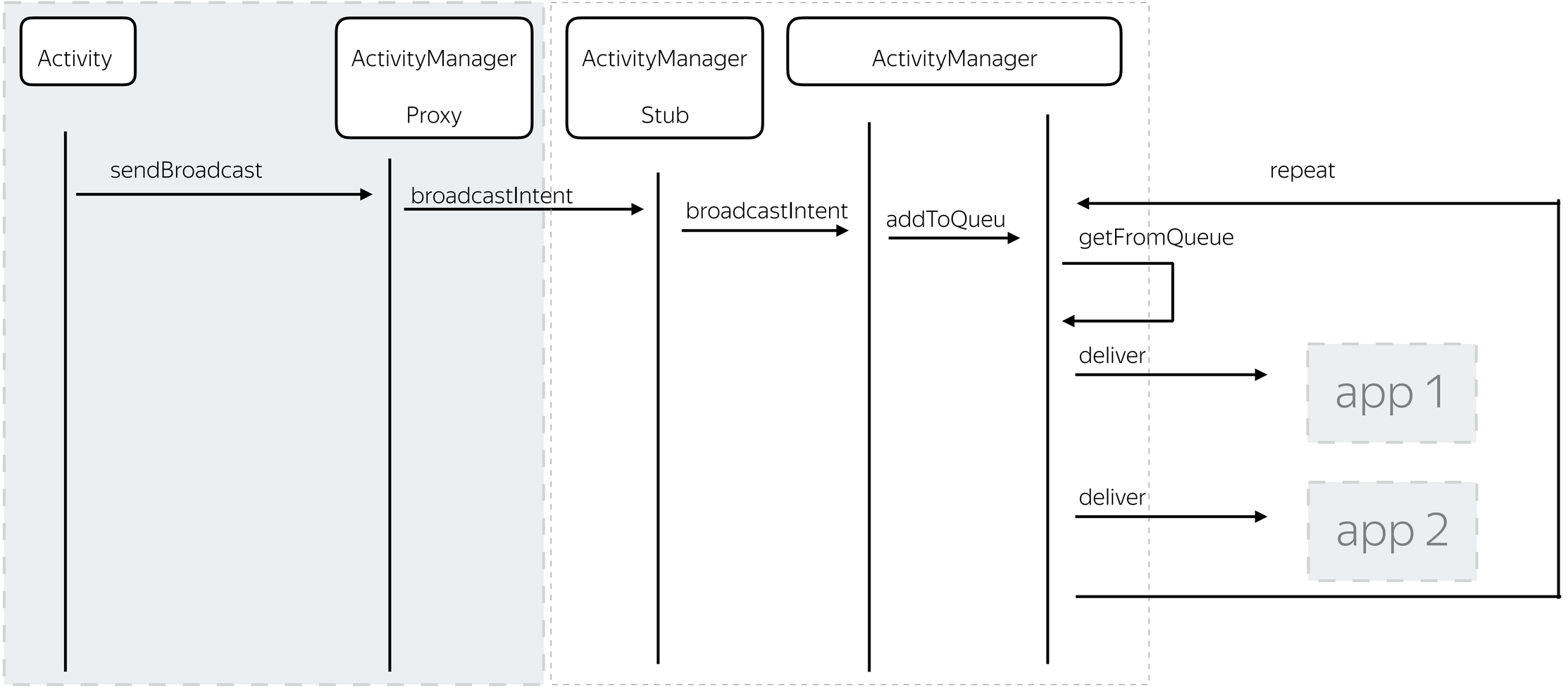
Особенности startActivityForResult

- 1Mb на транзакцию
- пользователь покидает ваше приложение
- как следствие система вас может убить или пользователь просто не вернётся в него
- передавать можно примитивы, String, Parcelable, Serializable и их списки
- **Основное назначение – запуск других activity, получение результата их работы по завершению**

sendBroadcast

```
Intent intent = new Intent();
intent.setAction("com.example.broadcast.BROADCAST_ME");
intent.putExtra("data","Hello world");
sendBroadcast(intent);
```

sendBroadcast на схеме



sendBroadcast очереди

- В системе всего три (4.2+) очереди бродкастов:

1. ~~mStickyBroadcasts~~ (Deprecated)
2. mFgBroadcastQueue
3. mBgBroadcastQueue

Особенности sendBroadcast

- 1Mb на транзакцию
- время доставки Intent’а не гарантировано
- полностью асинхронно
- вы не знаете когда получатель(и) получит ваш Intent и как он на него ответит. И ответит ли (кроме orderedBroadcast)
- передавать можно примитивы, String, Parcelable, Serializable и их списки

Для чего подходит Broadcast

- сообщать о событиях, если время доставки вас не сильно беспокоит
- или время доставки вас беспокоит чуть больше чем «не сильно» и приложение, которому вы отправляете находится в Foreground
- registerReceiver
  принимать сообщения от системы (CONNECTIVITY_ACTION, ACTION_HEADSET_PLUG, ACTION_MEDIA_MOUNTED, ...)
- **Основное назначение – рассылка сообщений/ уведомлений**

AIDL – термины

- **AIDL** – Android Interface Definition Language
- **IPC** – InterProcess Communications
- **RPC** – Remote Procedure Call

AIDL данные 1

```
// CLIENT

Intent service = new Intent();
service.setComponent(component);
bindService(service, mServiceConnection, BIND_AUTO_CREATE);

//…

ServiceConnection mServiceConnection = new ServiceConnection() {

  void onServiceConnected(ComponentName name, IBinder service) {
    mService = ITestService.Stub.asInterface(service);
    SomeData someData = mService.getData(SIZE);
    mService.setData(getProcessedData(someData));
    unbindService(this);
  }

  void onServiceDisconnected(ComponentName name) {
    mService = null;
  }
};
```

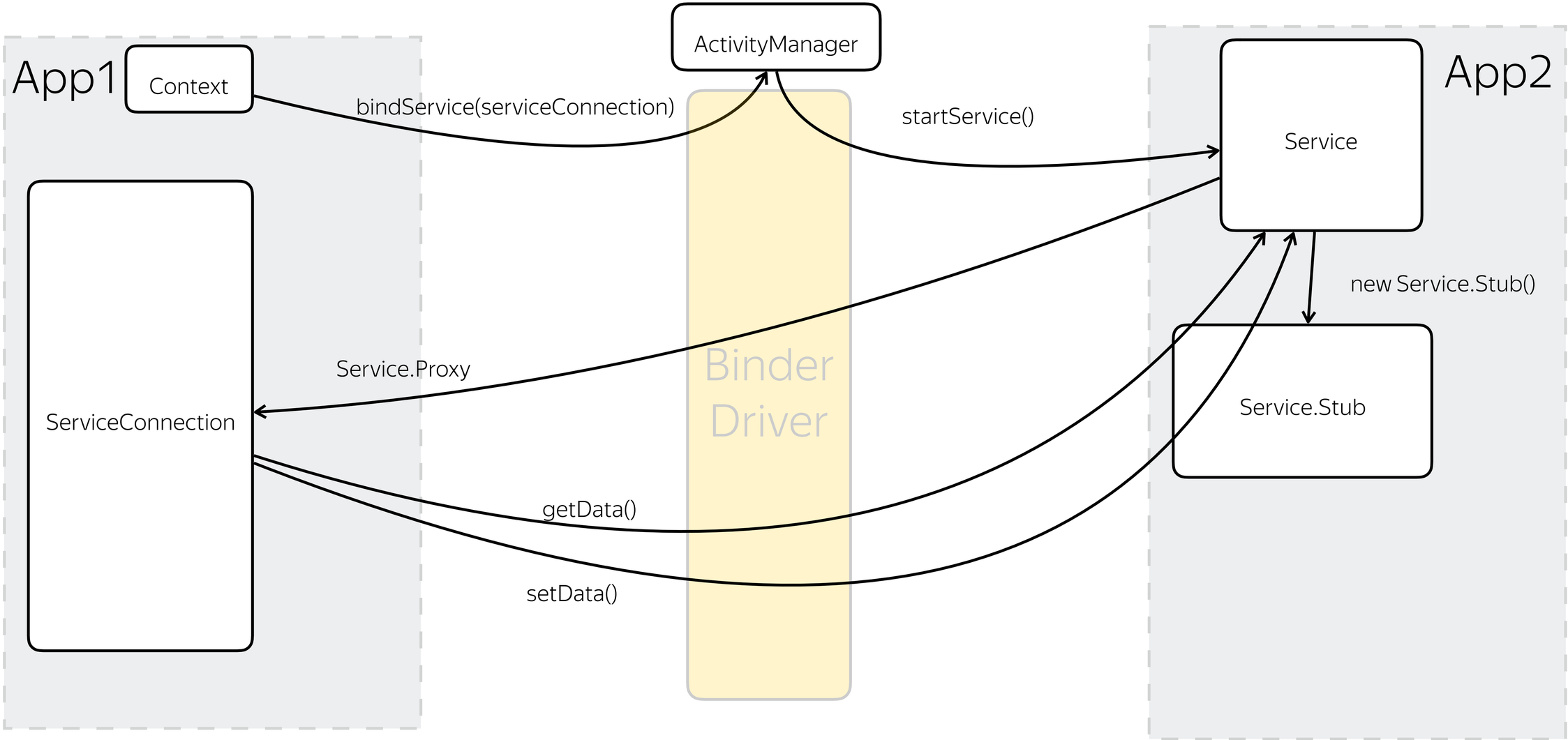
```
//SERVER

class TestService extends Service {
  ITestService.Stub mBinder =
      new ITestService.Stub() {

    SomeData getData(int listSize) {
      return new SomeData(listSize);
    }

    void setData(SomeData someData){
      someData = someData;
    }
  };

}
```

AIDL данные 2

```
// ITestService.aidl
package ru.example.testservice;

import ru.example.testservice.SomeData;

interface ITestService {
    SomeData getData(int listSize);
    void setData(in SomeData route);
}
```

```
// SomeData.aidl
package ru.example.testservice;

parcelable SomeData;
```

AIDL данные 3. Схема



AIDL что можно передать

- Примитивы
- String
- Parcelable-объекты
- IInterface
- Списки поддерживаемых типов

AIDL Callback 1

```
// ITestService.aidl
package ru.example.testservice;

import ru.example.testservice.SomeData;
import ru.example.testservice.ISomeCallback;
interface ITestService {
interface ITestService { listSize);
  SomeData getData(int listSize);e);
  void setData(in SomeData route);

  void setCallBack(
       in ISomeCallback someCallback);
}
```

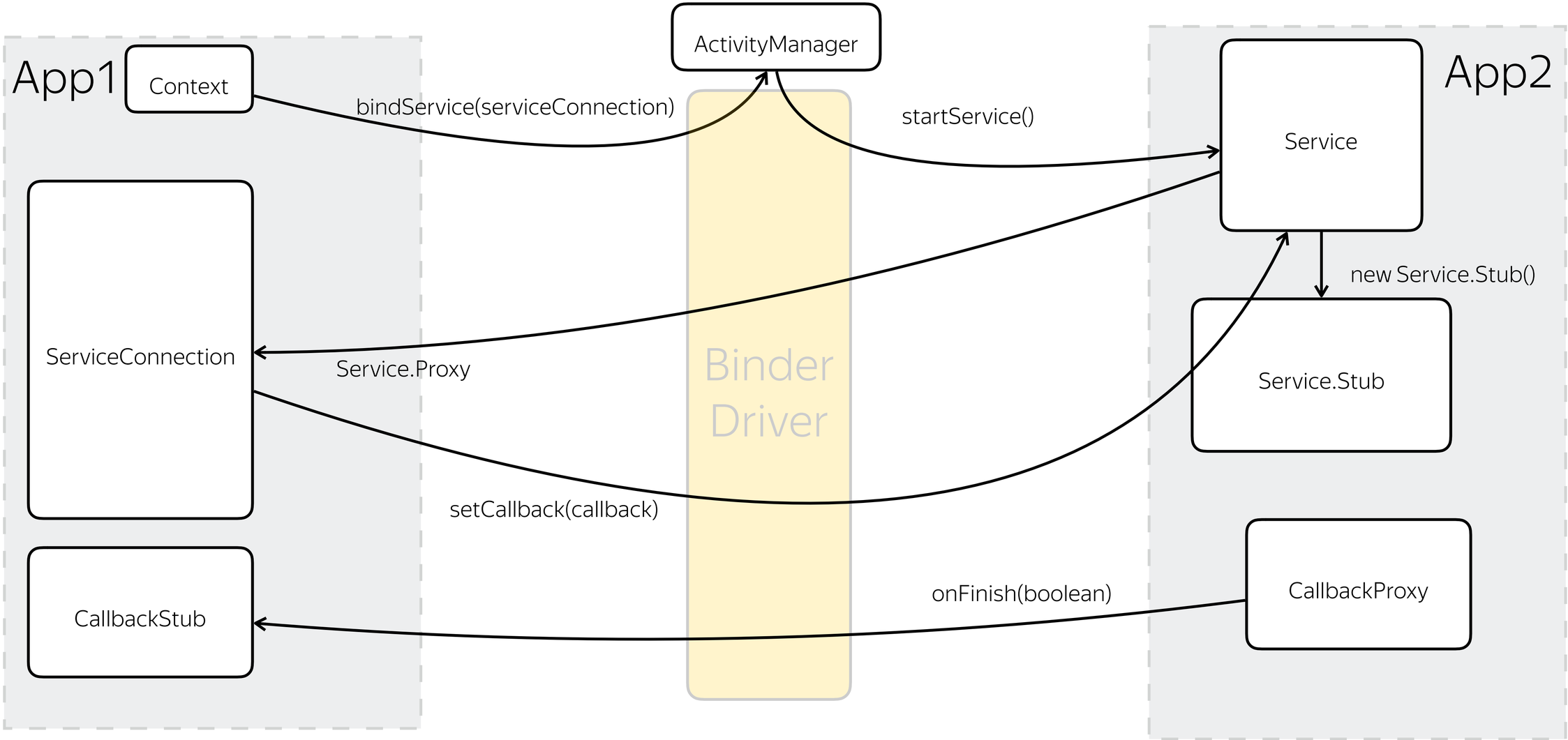
```
// ISomeCallback.aidl
package ru.example.testservice;

interface ISomeCallback {
  void onFinish(boolean success);
}// SomeData.aidl
package ru.example.testservice;

parcelable SomeData;
```

AIDL Callback 2

```
// CLIENT
ISomeCallback.Stub someCallback =
 new ISomeCallback.Stub() {
  void onFinish(boolean success) {/*.*/}
};
ServiceConnection mServiceConnection =
 new ServiceConnection() {
  void onServiceConnected(ComponentName name,
                          IBinder service) {
    mService = ITestService.Stub.asInterface(service);
    mService.setCallBack(someCallback);
  }

  void onServiceDisconnected(ComponentName name) {
    mService = null;
  }
};
//...
bindTestService();
```

```
//SERVER

class TestService extends Service {

 ITestService.Stub mBinder = new ITestService.Stub() {
  SomeData getData(int listSize) {
    return new SomeData(listSize);
  }
  void setData(SomeData someData) {
    CurrentSomeData = someData;
  }

  void setCallBack(ISomeCallback callback) {
    new Thread() {
      void run() {
        callback.onFinish(isEverythingOk());
      }
    }.start();
  }
 };
//...
}
```

AIDL Callback 3



AIDL Callback 4.1 Messenger

В качестве Callback можно использовать объект Messenger

```
// ITestService.aidl
package ru.example.testservice;

import ru.example.testservice.SomeData;

interface ITestService {
  SomeData getData(int listSize);
  void setData(in SomeData route);

  void setMessenger(in Messenger messenger);
}
```

```
// Messenger.aidl
package android.os;

import android.os.Message;

/** @hide */
oneway interface IMessenger {
    void send(in Message msg);
}
```

AIDL Callback 4.2 Messenger

```
// CLIENT

Messenger messenger = new Messenger(new Handler() {
    void handleMessage(Message msg) {
        showToast(msg.getData().getString("0"));
    }
});

ServiceConnection mServiceConnection =
        new ServiceConnection() {

  void onServiceConnected(ComponentName name,
                          IBinder service) {
    mService = ITestService.Stub.asInterface(service);
    mService.setMessenger(messenger);
  }
//...
};
//...
bindTestService();
```

```
//SERVER

ITestService.Stub mBinder = new ITestService.Stub() {
  // ..
  void setMessenger(Messenger messenger) {
    Message message = Message.obtain();
    Bundle bundle = new Bundle();
    bundle.putString("0", "Hello world");
    message.setData(bundle);
    messenger.send(message);
  }
};
```

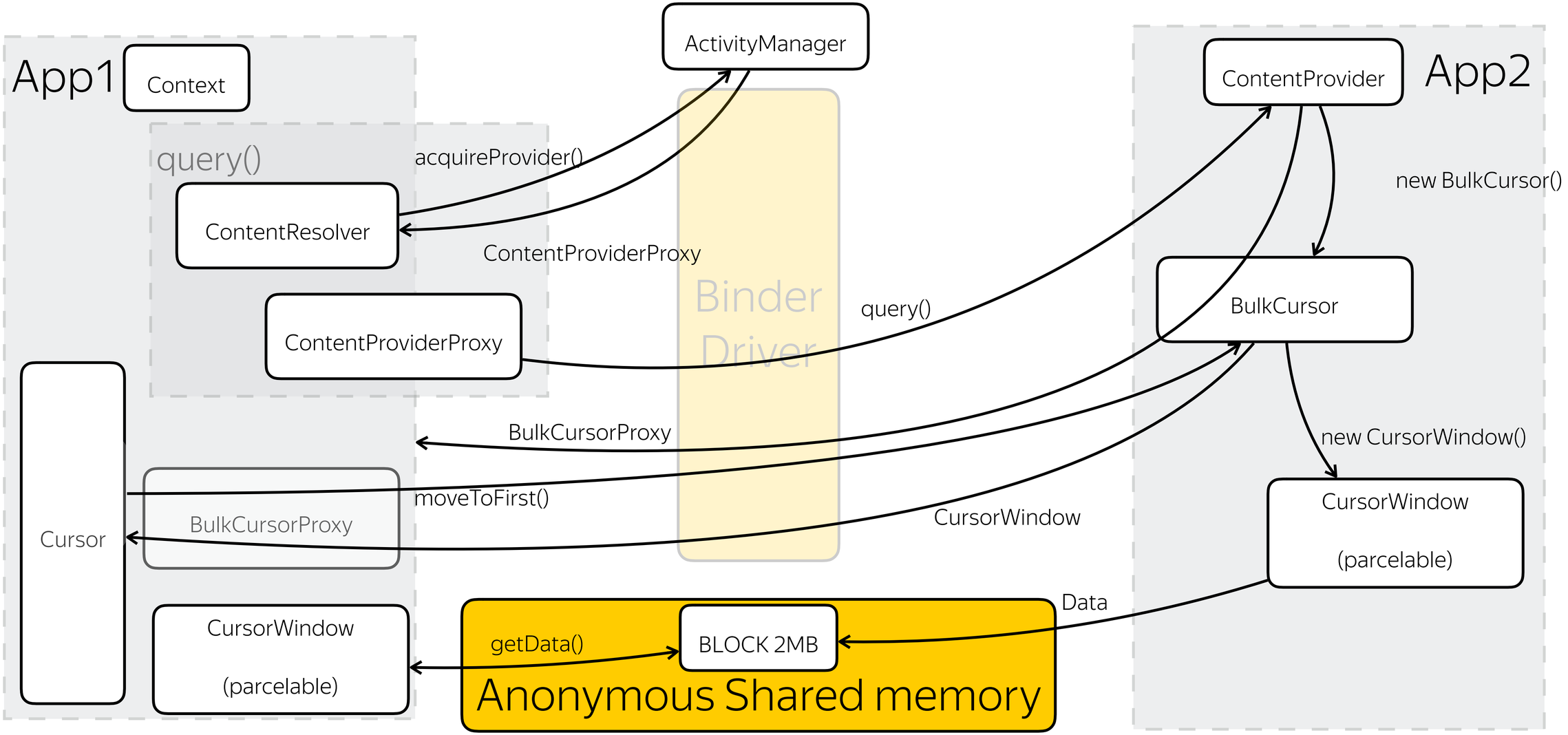
Для чего подходит AIDL

- для всего ;)
- вызов методов другого приложение как синхронно, так и асинхронно
- подписка на получение уведомлений без задержек (собственным Callback-объектом или используя Messenger или ResultReceiver)
- передача и получение небольших (до 1Мб) объектов
- **Основное назначение – удалённый вызов процедур**

ContentProvider

```
// Client
void onClick() {
    try (Cursor cursor = getContentResolver().query(uri, null, null, null, null)) {
        if (cursor != null && cursor.moveToFirst()) {
            byte[] blob = cursor.getBlob(cursor.getColumnCount() - 1);
        }
    } catch (Exception e) {
    }
}



.............



// Server

Cursor query(@NonNull Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
    MatrixCursor cursor = new MatrixCursor(new String[]{"_ID", "blob"}, 0);
    cursor.newRow().add("0").add(Byte.MAX_VALUE);
    return cursor;
}
```

ContentProvider2



Особенности ContentProvider

- 2Мб на каждую строку в курсоре
- строк может быть сколько хочешь
- запросы происходят синхронно
- передавать можно примитивы, byte[], String. Для остальных данных выполняется toString()
- а ещё есть метод call(), который возвращает Bundle
- **Основное назначение – передача данных**

Спецсредства для взаимодействия

- AccountManager – место для хранения аккаунтов с возможностью поделиться данными аккаунтов
- App Widgets – передача приложению реализующему AppWidgetHost (Launcher) своих View
- MyService extends NotificationListenerService – в этом случае ваш сервис будет получать все публикуемые уведомления (разрешение включается отдельной галочкой)
- remotecontroller (14 - 21) и mediasession (21+) – чтобы получить управление аудиосессиями нужно реализовать NotificationListenerService

Выводы

- **startActivity** – временно передать UI другому
- **sendBroadcast** – сказать всем желающим что что-то случилось или узнать что что-то случилось
- **ContentProvider** – поделиться или получить списки данных
- **AIDL** – включить другое приложение в фоне, чтобы оно работало, мы бы им управляли, а оно бы нам ещё и присылало что-нибудь
- **Спецсредства** – управление аккаунтами, рисование виджетов, чтение уведомлений, управление музыкой и много ещё чего другого не менее интересного...

Спасибо. Вопросы?

Антон Дудаков

Android разработчик,
постоянный участник AndroidDev Podcast

bwdude@yandex-team.ru

bwdude

bwdude