IPC in Apps - Android

Lohith
Founder and CTO, Fairket

Fairket

Basics

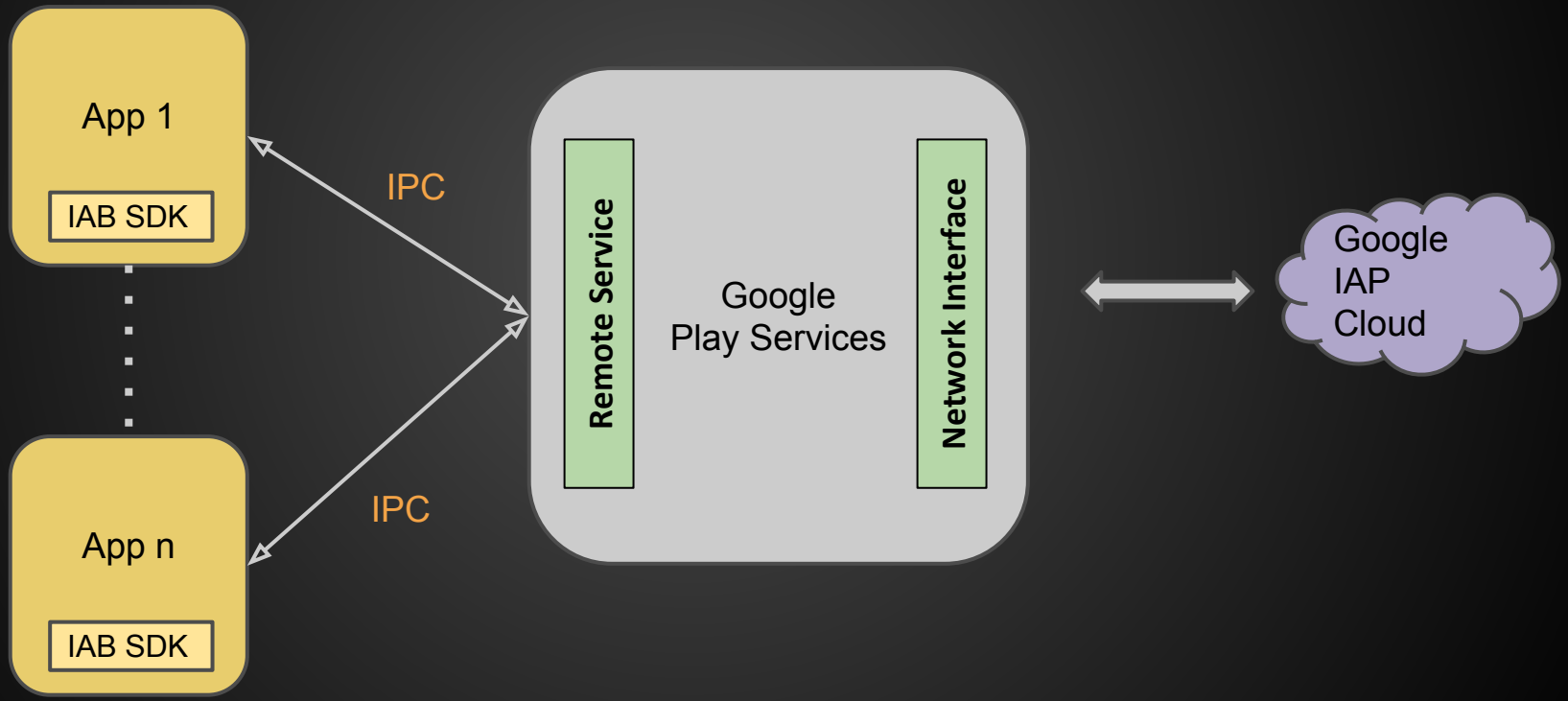
- Default - Each App runs on its own process
  - Stability, Security, Memory
- App can run in two or more process and vice versa
- Default - <activity>, <service>, <receiver> and <provider> run in same process
- All communication happen on the Looper (Main) thread by default

Fairket

Examples of IPC

- Access to Android services like:
  - Camera, ActivityManager, DownloadManager, Media etc
- External Examples
  - Google IAP
  - License Verification Library

IPC Example - Google IAP



Fairket

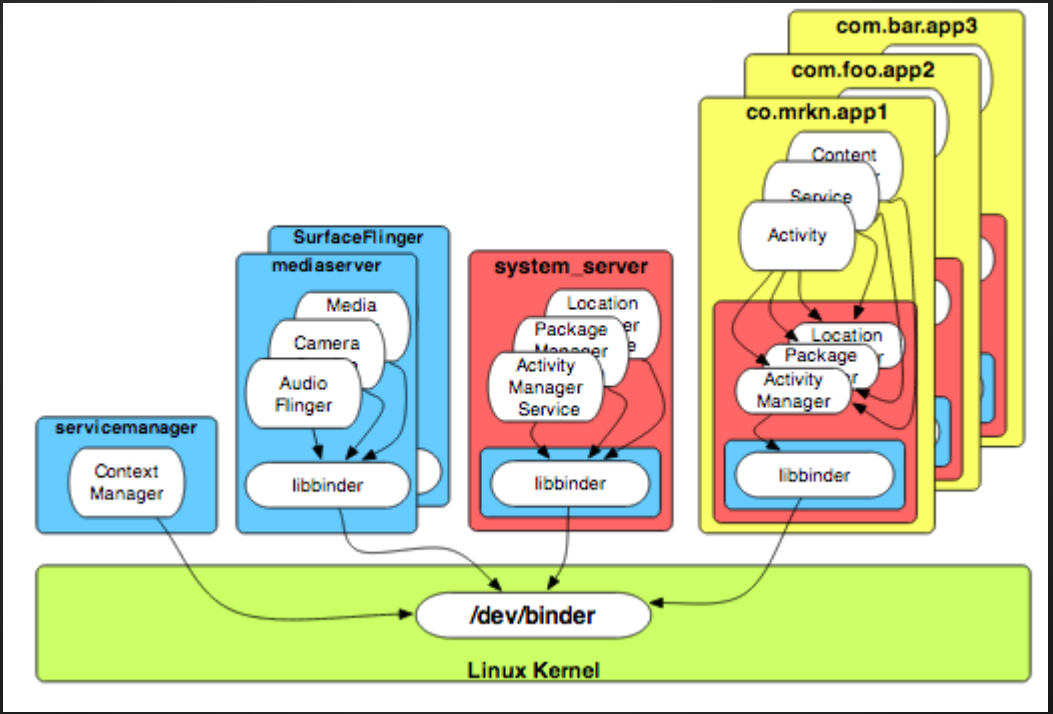
Android IPC Mechanisms

- Using intents
- Using services
- Using messenger and binder interfaces
- Using broadcast receivers

"Underneath all IPC mechanisms use Android IPC Binder"

What is Binder?

Android specific IPC mechanism based on shared memory



Fairket

Why Binder?

- ➔ Enables RPC style IPC calls
- ➔ Synchronous and Asynchronous (oneway) model
- ➔ Security - Receiver can recognize the sender (uid/pid)
- ➔ Simple Interface Definition Language (AIDL)
- ➔ Suitable for Android’s hostile environment
- ➔ Marshalling and Unmarshalling
- ➔ Local Execution Model

Fairket

Messenger IPC

- Message based communication across process
- Client - Implements Handler to receive messages
- Service - Uses Messenger to send notification to client
- Asynchronous like Intent but lower latency and overhead
- Ideal for receiving call-backs from service to client
- All underlying communication still uses Binder!

Fairket

Messenger IPC - Limitations

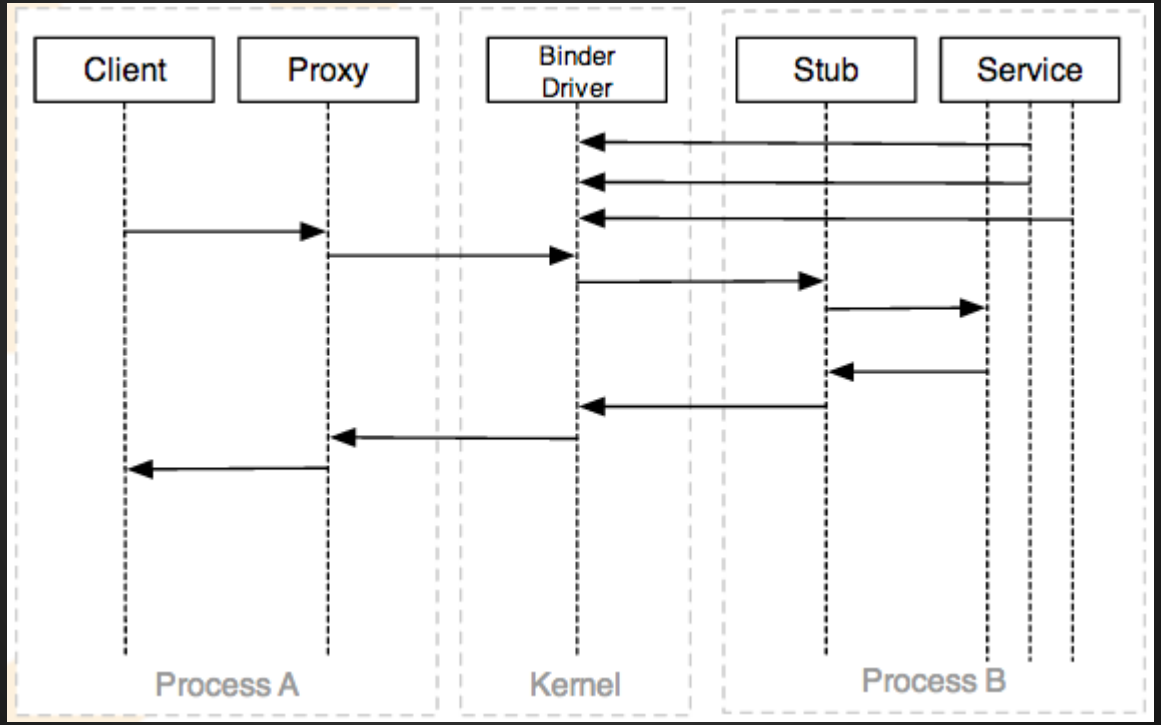
- ➔ Only supports Asynchronous
- ➔ No support for multithreaded IPC
- ➔ No Interface Definition Language

Fairket

Binder Interface IPC - 1

- ➔ RPC kind of API call
- ➔ Supports Synchronous and Asynchronous calls
- ➔ Supports multithreaded IPC calls
- ➔ Facilitated by Remote Service
- ➔ Interface defined through AIDL

Binder Interface IPC - 2



Fairket

AIDL

- AIDL follows Java like syntax
- Used both by client and service - Generates Proxy and Stub
- AIDL supports Java basic types, array, list and custom data types
- Custom data type needs to have an AIDL file of its own
- Specify the direction tag for custom data types - in, out, inout

Fairket

AIDL Example

```
// IRemoteService.aidl
package com.example.android;

// Declare any non-default types here with import statements

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();

    /** Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    void basicTypes(int anInt, long aLong, boolean aBoolean, float aFloat,
            double aDouble, String aString);
}
```

Limitations of Binder

- ➔ Maximum of 15 IPC binder threads
- ➔ No blocking operation - Same restriction as Looper Main thread
- ➔ No support to remote calls - Local to device
- ➔ Limits the transactional buffer to 1 Mb per process

Do’s and Don'ts - 1

- Don't use traditional Linux IPC technique like network sockets, pipes, shared files etc
- Security - Use relevant permission declaration on <service>
- Mark android:exported as “false” if <service> is not to be exposed
- Protect each IPC call by calling checkCallingPermission

Fairket

Do’s and Don'ts - 2

- Use bound services - Bind/Unbind as required
- Don't start a service unless really required
- onStart/onStop is good place to connect/disconnect to a service
- Use the AIDL direction tag judiciously
- Mark AIDL callback interface as one way
- Don’t send too much data over IPC

Additional References

- Android Binder on elinux.org
- Android Binder by Thorsten Schrieber from Ruhr-Universitat Bochum
- Deep Dive into Android Binder by Aleksandar Gargenta

Questions?

Thank you for your patience!

@v_lohith
lohith@fairket.com
http://www.fairket.com/