Contents

1 IPC

Inter-process communication. Each process has its own address space. This provides data isolation and prevents direct interaction between different processes. How can we communicate with a Service, or send an Intent? Use **Binder**.

- Underpins most Android communication, i.e. when we use various system capabilities
- Kernel driver: provides lightweight RPC (remote procedure calls), data passing. C.f. Linux/Unix signals / pipes / sockets etc. Reading and writing Parcels between processes. Process, user ID authority / trust
- Per-process thread pool for handling requests
- Synchronous calls between processes

2 Binder

2.1 Facilities

- Calls: Simple inter-process messaging system. One-way, two-way
- Identifying: PID, UID
- Notification: Link to death, Leaked Service connections
- Managing: Reference counting, object mapping across processes, sleeping and waking worker threads
- Indirect functionality: As a token, sharing fd (file descriptor) to shared memory area

2.2 Implementation

API for apps

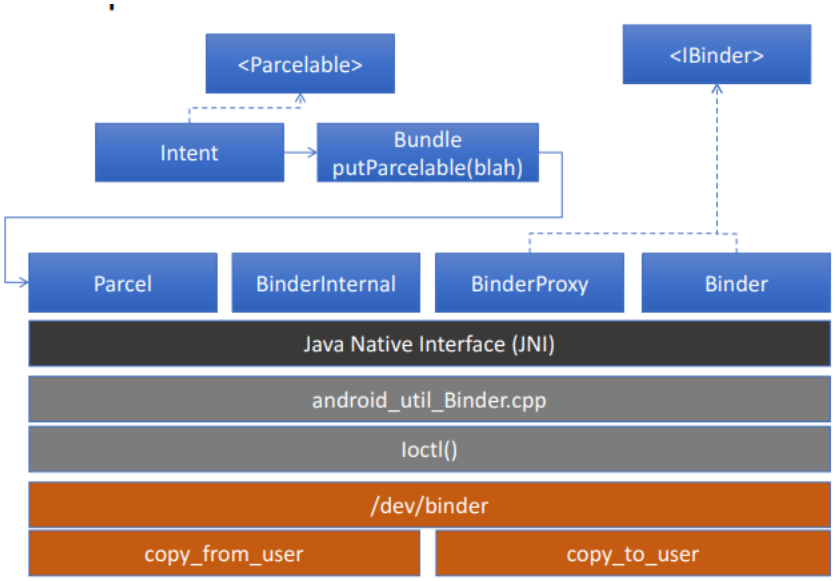
- AIDL
- Java API wrapper
  - Exposes the IBinder interface
  - Wraps the middleware layer
  - Parcelable object marshalling interface

Native Middleware

- Implements the user space (i.e. within a process) facilities of the Binder framework
- Marshalling and unmarshalling of specific data to primitives
- Provides interaction with the Binder kernel driver

Kernel driver

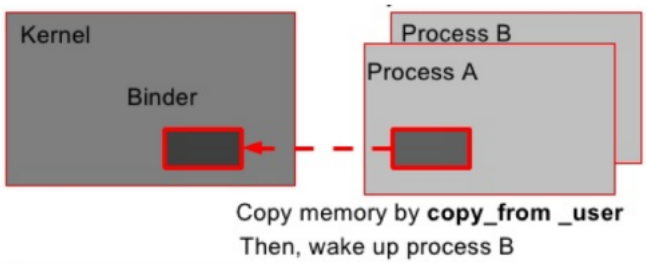
- Supports ioctl system calls from the middleware
- Supports cross-process file operations, memory mapping
- Thread pool for each service application for IPC
- Mapping of objects between processes via `copy_from_user`, `copy_to_user`

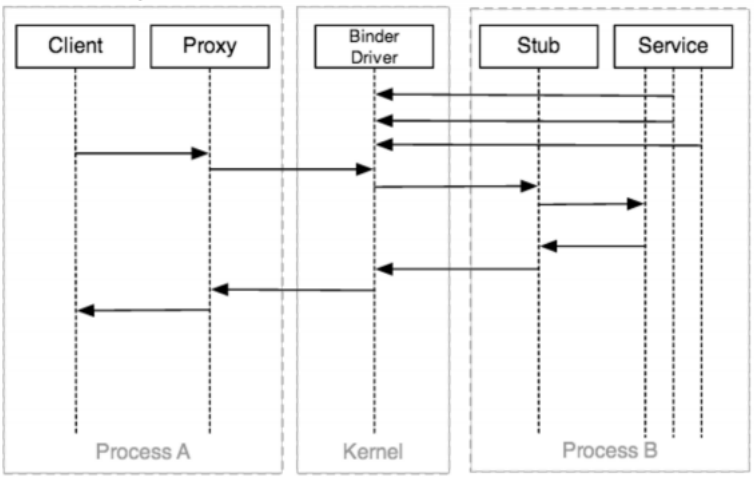


2.3 Transactions

A transaction between processes: IBinder.transact -> Binder.onTransact(). Binder maintains a pool of transaction threads in each process

- To dispatch all IPCs coming from other processes
- If process A calls process B
  - Calling thread in A blocks in transact()
  - Sends the transaction to process B
  - Next available thread in B receives the incoming transaction, calls onTransact() on the target object, replies with the resultant Parcel
  - Thread in process A returns, resumes execution
- Reliant on Service (B) responding in a timely manner
  - Hence catching remote exceptions, transaction failures
  - Developer defined worker threads
  - Handling multiple calls from multiple transaction threads





3 Defining Remotely Bound Services

Using the Android Interface Definition Language (AIDL)

- Specify an interface for the service functionality
- Generates a proxy object
- To be used locally as if the remote service was not remote
- Generates a stub implementation
- The remote side of the transaction
- Generates a communication protocol
- Parcelling and unparcelling steps

Similar to Java interface definitions. Has label method parameters for efficiency

- in: transferred to the remote method
- out: returned to the caller
- inout: both in and out
- oneway: asynchronous

Permitted types

- Java primitive types, Lists, Maps
- Other AIDL-generated interfaces
- Classes implementing the Parcelable protocol

4 IPC Abstraction

Intent

- Highest level abstraction
- Asynchronous message passing

Inter-process method invocation by using AIDL (Android Interface Definition Language). **Binder**: kernel driver. **Ashmem**: Shared memory.

- Passed as file descriptor objects by Binder
- The USB service gives a specific USB device to an app without giving it unrestricted access

placeholder