

# G54MDP

# Mobile Device Programming

Lecture 9 – Services, IPC

# Remote Services

- For communicating across process boundaries
  - i.e. using a Service belonging to a different application / process
  - Likely to be used by multiple processes at once
  - Declare the service as *exported* in the Manifest
    - Explicit rather than implicit
    - More sophisticated permissions system later on
- Using a Messenger
  - Simplest implementation
  - C.f. using a Handler to talk between Threads
    - Queues Messages into a single Thread
  - Messages must be Parcelable

# Parcelable

- Locally (same process) bound Services share the same process memory space
  - Easy to call methods, transfer objects between classes
- How should different processes talk to each other?
  - `java.io.Serializable`
    - Short-term persistence
    - Write object ID, field via reflection
    - Change the class / variable name, what happens?
    - Slow
  - `Parcelable`
    - Define a simple wire-protocol for writing primitives
    - Immune to minor changes to class definitions
      - Same interface, different class
    - Supported by Android kernel driver
    - Fast!

# Remotely Bound Services

- Define remote interface in the Android Interface Definition Language (AIDL)
  - Providing OS wide services for all applications
    - i.e download management
  - Multithreading with complex client / server bi-directional communication
- Implement remote interface
  - Stub and application specific methods
- Implement Service methods
- Implement Client methods

# AIDL

- Similar to Java interface definition syntax
  - Can declare methods
  - Cannot declare static fields
- Label method parameters
  - in: transferred to the remote method
  - out: returned to the caller
  - inout: both in and out
- Types
  - Java **primitive** types
  - StringList
    - List elements must be valid AIDL data types
  - Map
    - Map elements must be valid AIDL data types
  - CharSequence
  - Other AIDL-generated interfaces
  - Classes implementing the Parcelable protocol

# AIDL

- Generate a Java interface with same name as .aidl file
  - Android Studio does this automatically
- Generated interface contains:
  - Abstract inner class called Stub
  - Interface & helper methods

Let's have a look...



root@android:/ # service list

Found 68 services:

```
0      phone: [com.android.internal.telephony.ITelephony]
1      iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
2      simphonebook: [com.android.internal.telephony.IIccPhoneBook]
3      isms: [com.android.internal.telephony.ISms]
4      dreams: [android.service.dreams.IDreamManager]
5      commontime_management: []
6      samplingprofiler: []
7      diskstats: []
8      appwidget: [com.android.internal.appwidget.IAppWidgetService]
9      backup: [android.app.backup.IBackupManager]
10     uimode: [android.app.IUiModeManager]
11     serial: [android.hardware.ISerialManager]
12     usb: [android.hardware.usb.IUsbManager]
13     audio: [android.media.IAudioService]
14     wallpaper: [android.app.IWallpaperManager]
15     dropbox: [com.android.internal.os.IDropBoxManagerService]
16     search: [android.app.ISearchManager]
17     country_detector: [android.location.ICountryDetector]
```

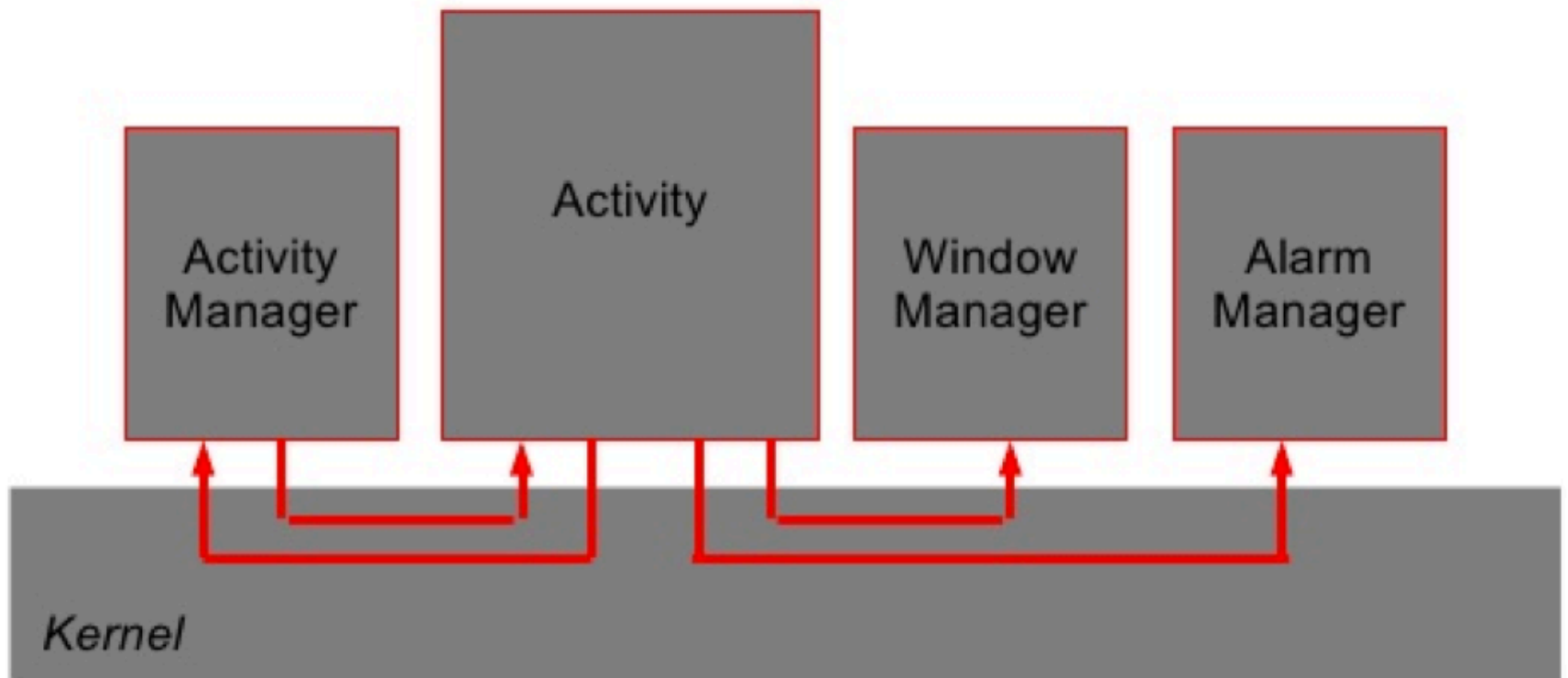


root	29	1	276	156	c0098770	0000e840	S	/sbin/ueventd
system	30	1	836	344	c0195c08	40036fc0	S	/system/bin/servicemanager
root	31	1	4008	820	ffffffff	4003e76c	S	/system/bin/vold
root	33	1	8632	1232	ffffffff	4006a76c	S	/system/bin/netd
root	34	1	880	388	c01a10a0	40037a70	S	/system/bin/debuggerd
radio	35	1	5468	836	ffffffff	4003776c	S	/system/bin/rild
system	36	1	25336	9348	ffffffff	4006bfc0	S	/system/bin/surfaceflinger
root	37	1	143452	33584	ffffffff	400370e4	S	zygote
drm	38	1	6564	2320	ffffffff	400befc0	S	/system/bin/drmservice
media	39	1	23012	6080	ffffffff	4008cfc0	S	/system/bin/mediaserver
install	40	1	848	456	c021db90	40036d50	S	/system/bin/installd
keystore	41	1	1796	888	c01a10a0	40037a70	S	/system/bin/keystore
root	42	1	828	372	c00b4eb0	40037ebc	S	/system/bin/qemud
shell	45	1	764	460	c0148178	40031d50	S	/system/bin/sh
root	46	1	5516	292	ffffffff	00015ef0	S	/sbin/adbd
root	279	46	752	428	c002a7a0	4003294c	S	/system/bin/sh
root	284	279	720	408	c0098770	400370e4	S	logcat
system	293	37	228248	44312	ffffffff	40036fc0	S	system_server
u0_a20	383	37	154684	20256	ffffffff	40037ebc	S	com.android.inputmethod.latin
radio	397	37	170880	23520	ffffffff	40037ebc	S	com.android.phone
u0_a21	415	37	167224	29712	ffffffff	40037ebc	S	com.android.launcher
u0_a0	445	37	171808	25212	ffffffff	40037ebc	S	android.process.acore
u0_a10	480	37	152876	16772	ffffffff	40037ebc	S	com.android.defcontainer
root	521	46	764	476	c002a7a0	4003294c	S	/system/bin/sh
u0_a37	529	37	160068	37056	ffffffff	40037ebc	S	com.android.systemui
u0_a17	557	37	153868	16452	ffffffff	40037ebc	S	com.android.location.fused
u0_a25	585	37	153388	17488	ffffffff	40037ebc	S	com.android.music
system	601	37	161068	18392	ffffffff	40037ebc	S	com.android.settings
u0_a14	610	37	157504	20524	ffffffff	40037ebc	S	android.process.media
u0_a0	632	37	159880	18888	ffffffff	40037ebc	S	com.android.contacts
u0_a6	650	37	159192	18932	ffffffff	40037ebc	S	com.android.providers.calendar

# Services

- Entropy Service
- Power Manager
- Activity Manager
- Telephony Registry
- Package Manager
- Account Manager
- Content Manger
- System Content Providers
- Battery Service
- Lights Service
- Vibrator Service
- Alarm Manager
- Init Watchdog
- Window Manager
- Bluetooth Service
- Device Policy
- Status Bar
- Clipboard Service
- Input Method Service
- NetStat Service
- NetworkManageme  
nt Service
- Connectivity Service
- Throttle Service
- Accessibility  
Manager
- Mount Service
- Notification  
Manager
- Device Storage  
Monitor
- Location Manager
- Search Service
- DropBox Service
- Wallpaper Service
- Audio Service
- Headset Observer
- Dock Observer
- USB Observer
- UI Mode Manager  
Service
- Backup Service
- AppWidget Service
- Recognition Service
- DiskStats Service

# IPC – Inter-Process Communication

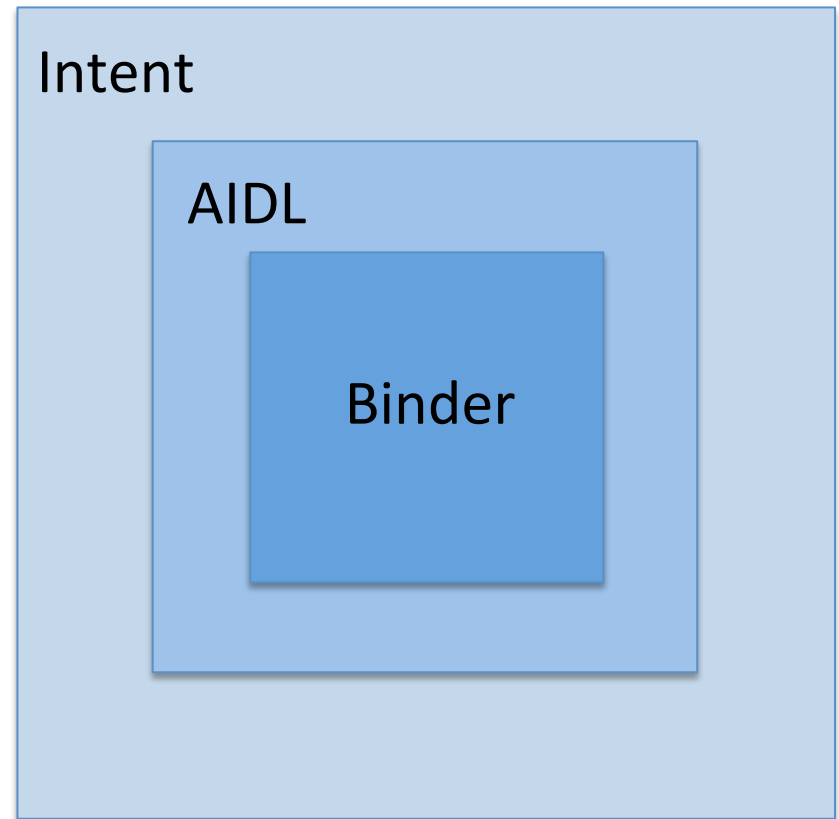


# IPC

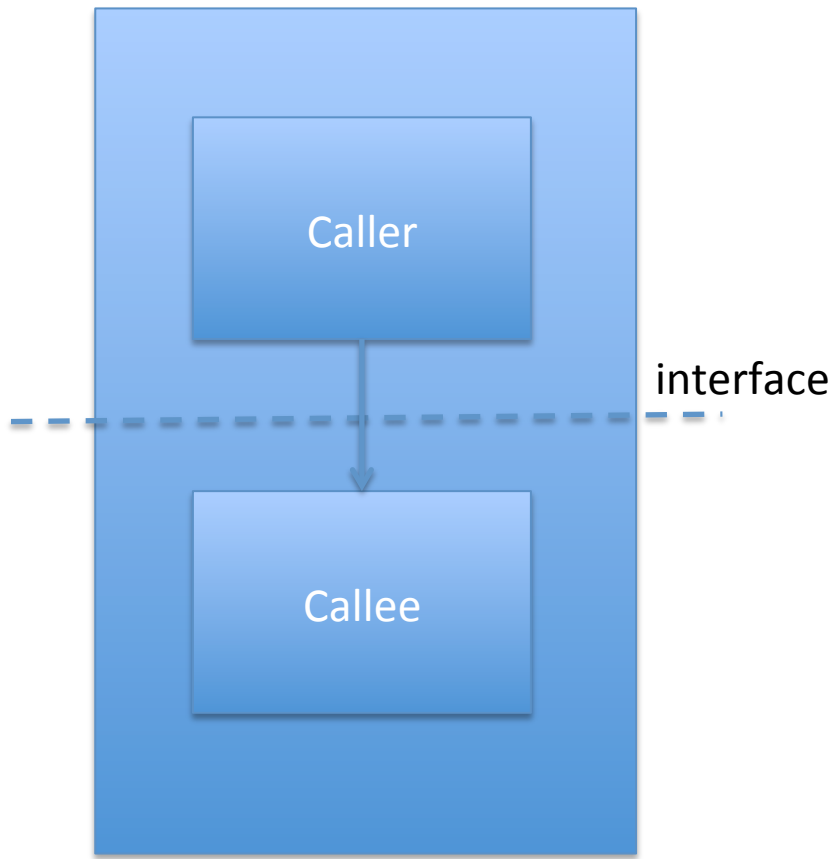
- Each process has its own address space
  - Provides data isolation
  - Prevents direct interaction between different processes
    - However, often required for modularisation
- What **actually** happens when we start a Service, or send an Intent?
- Binder
  - Underpins most Android communication
    - i.e. when we use the NotificationManager
  - Provides lightweight RPC (remote procedure communication)
    - C.f. Linux/Unix signals / pipes / sockets etc
  - Kernel driver
  - High performance via shared memory
  - Per-process thread pool for handling requests
  - Synchronous calls between processes

# IPC Abstraction

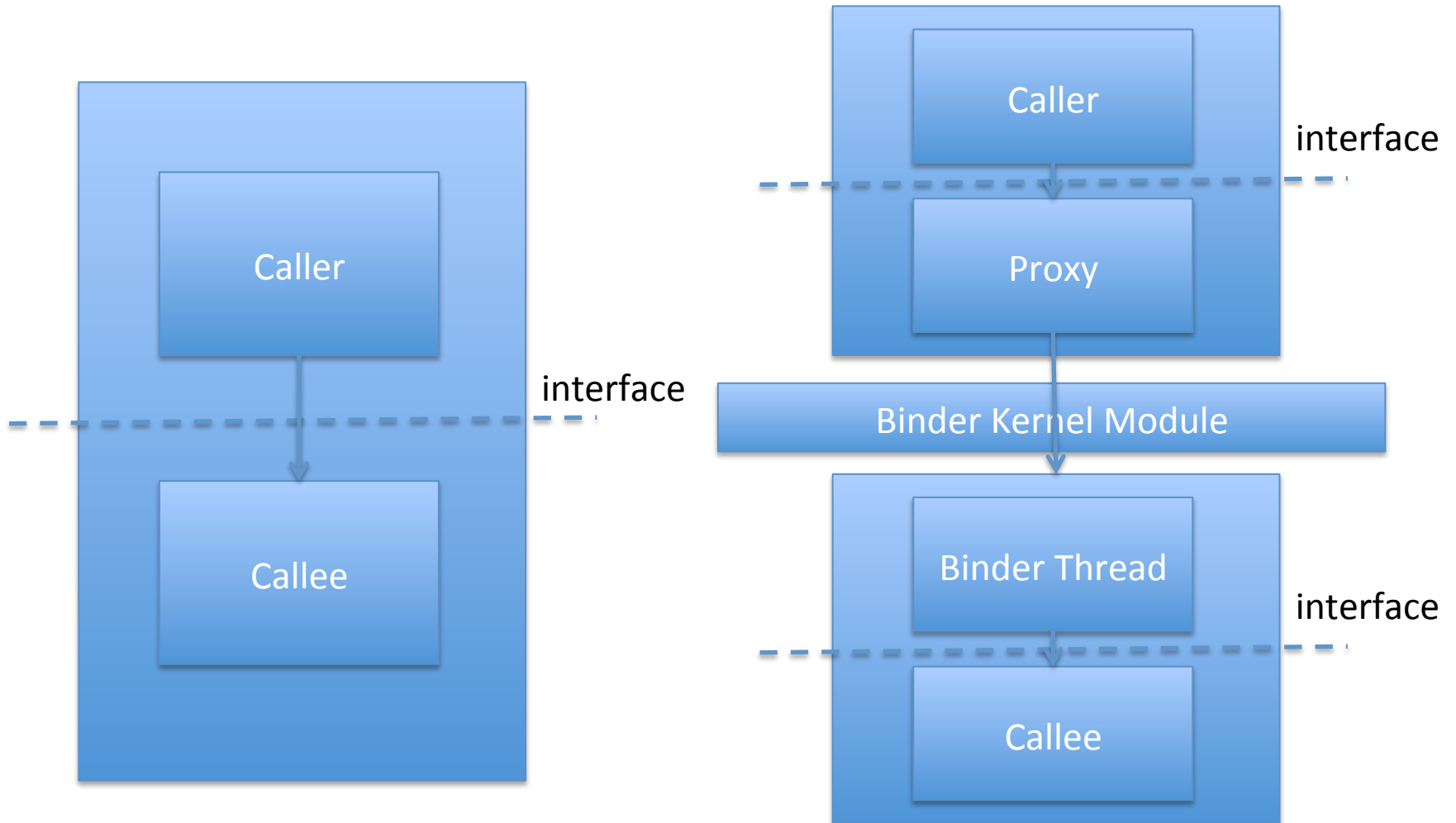
- Intent
  - Highest level abstraction
- Inter process method invocation
  - AIDL
- binder: kernel driver
- ashmem: shared memory



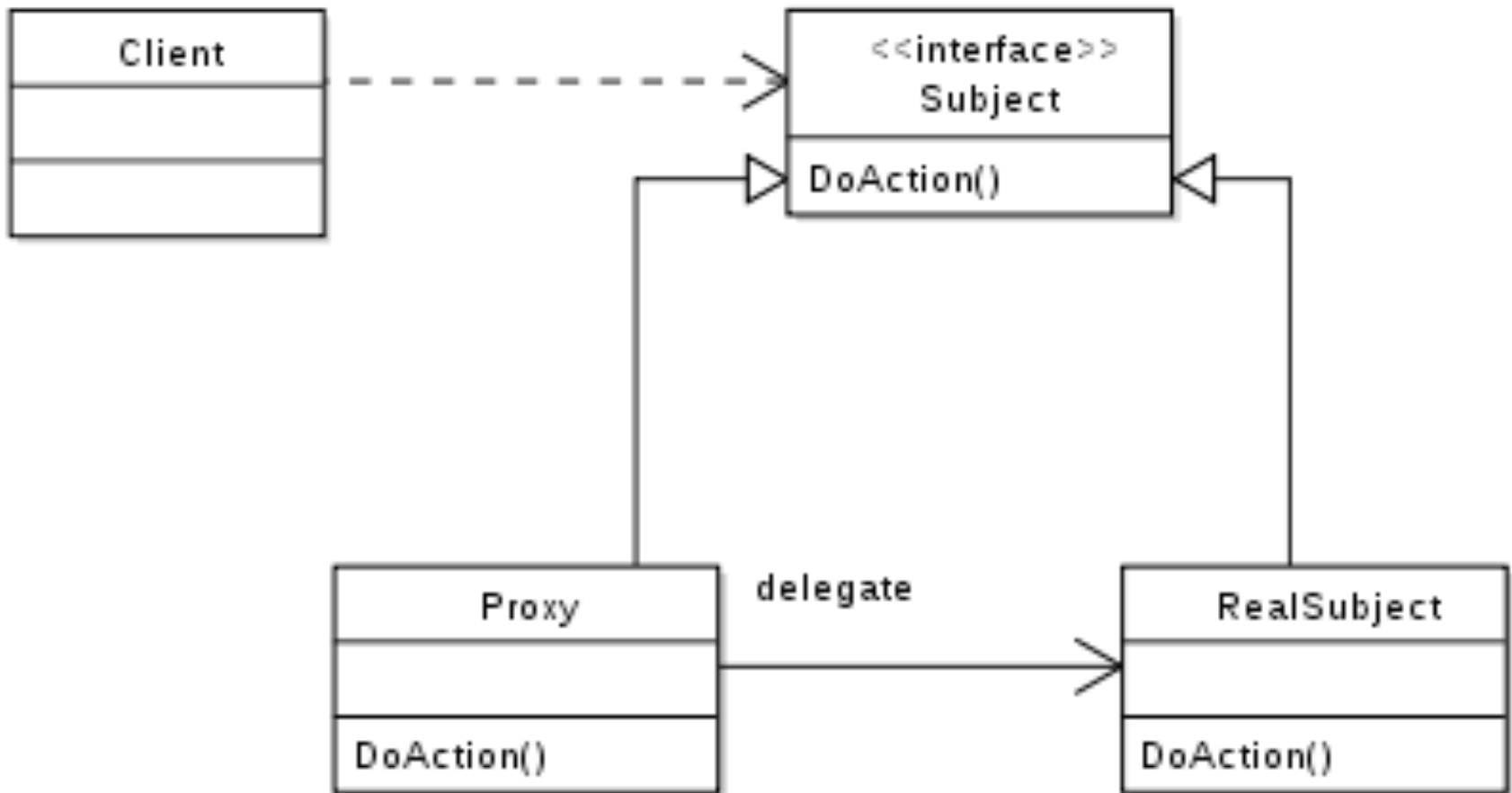
# Inter-process method invocation



# Inter-process method invocation



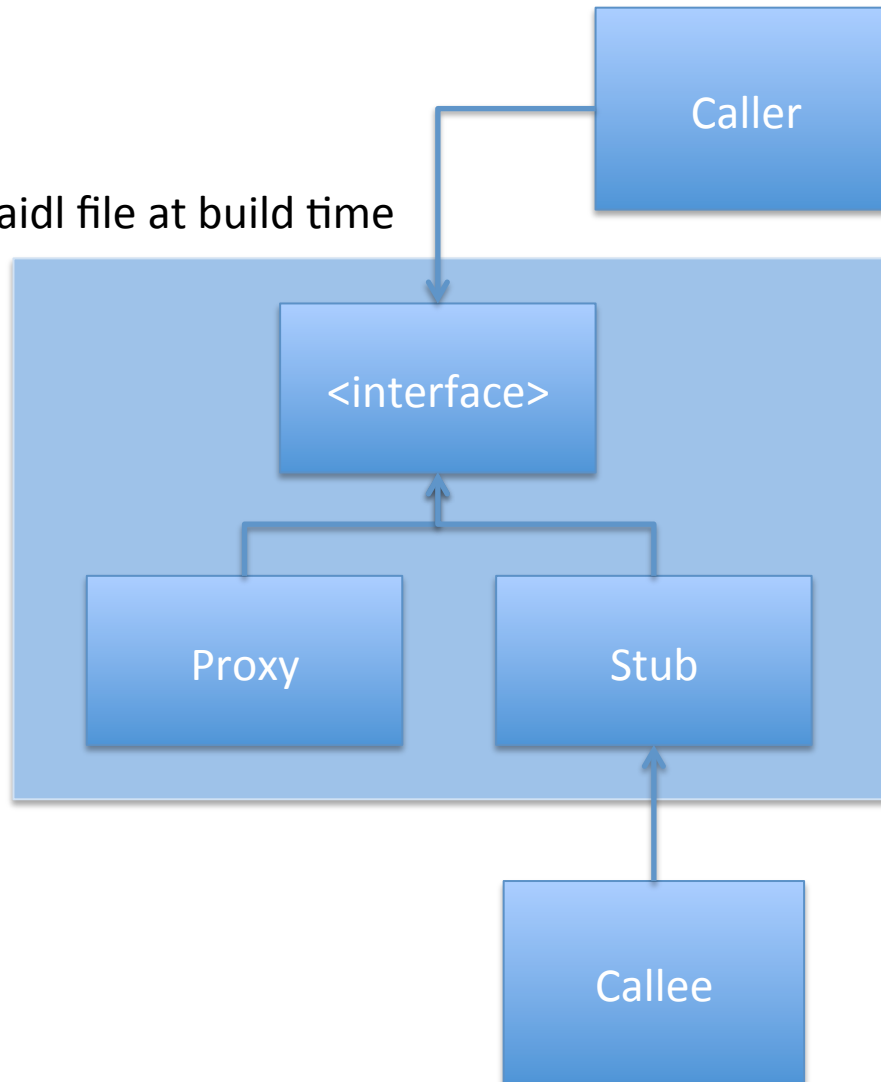
# Proxy Design Pattern



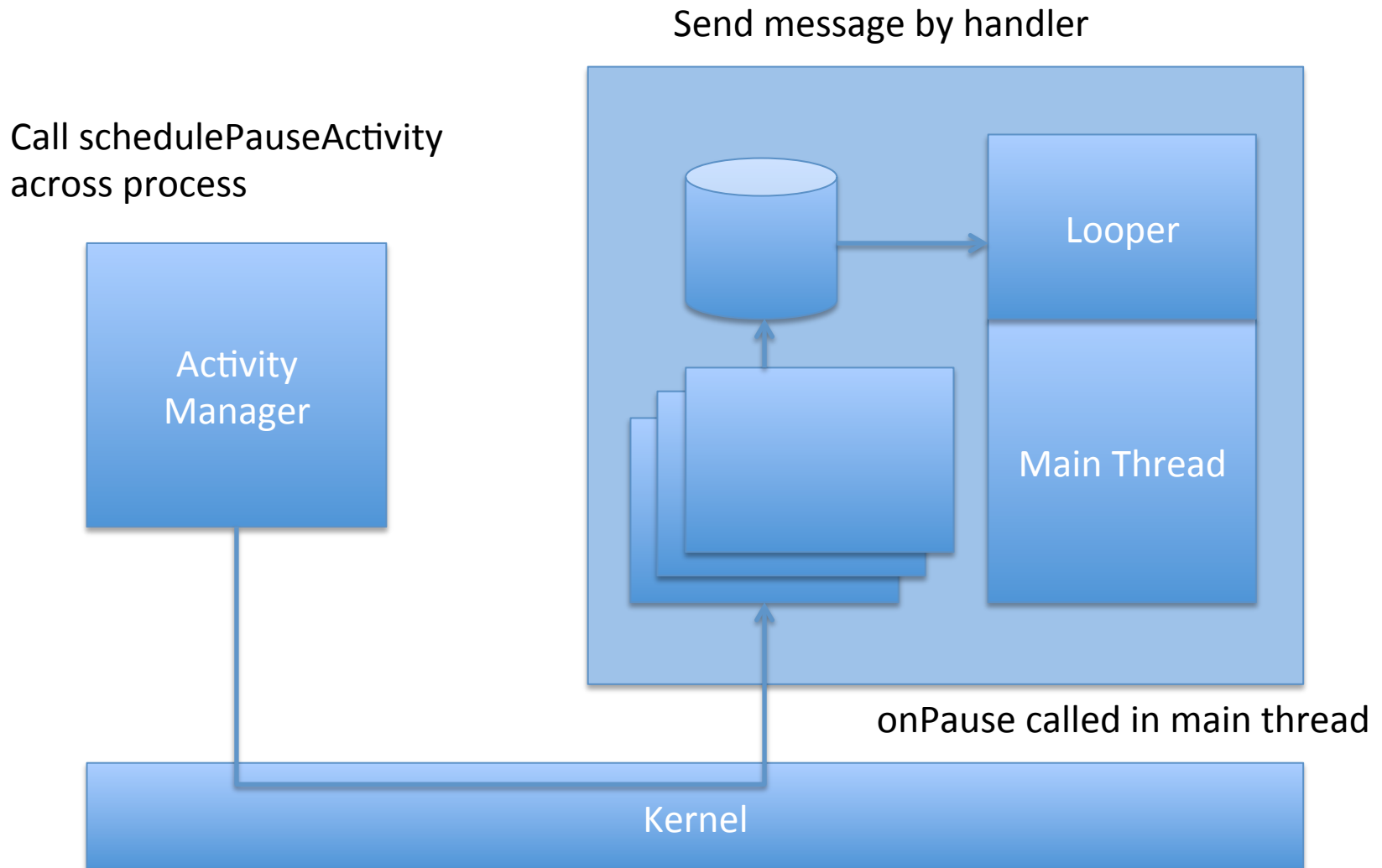


# AIDL

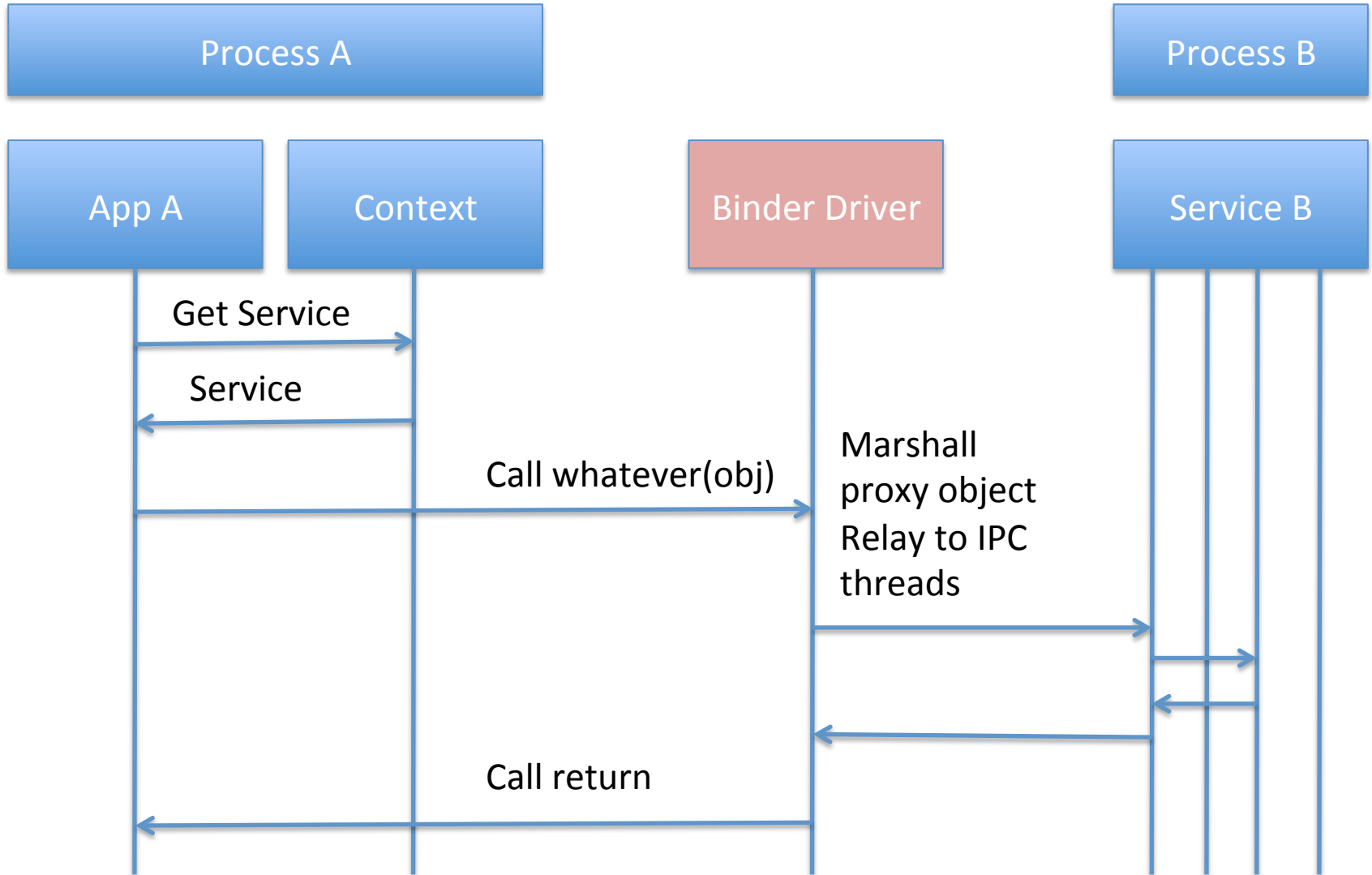
Auto generated from .aidl file at build time



# onPause()



# Binder in action



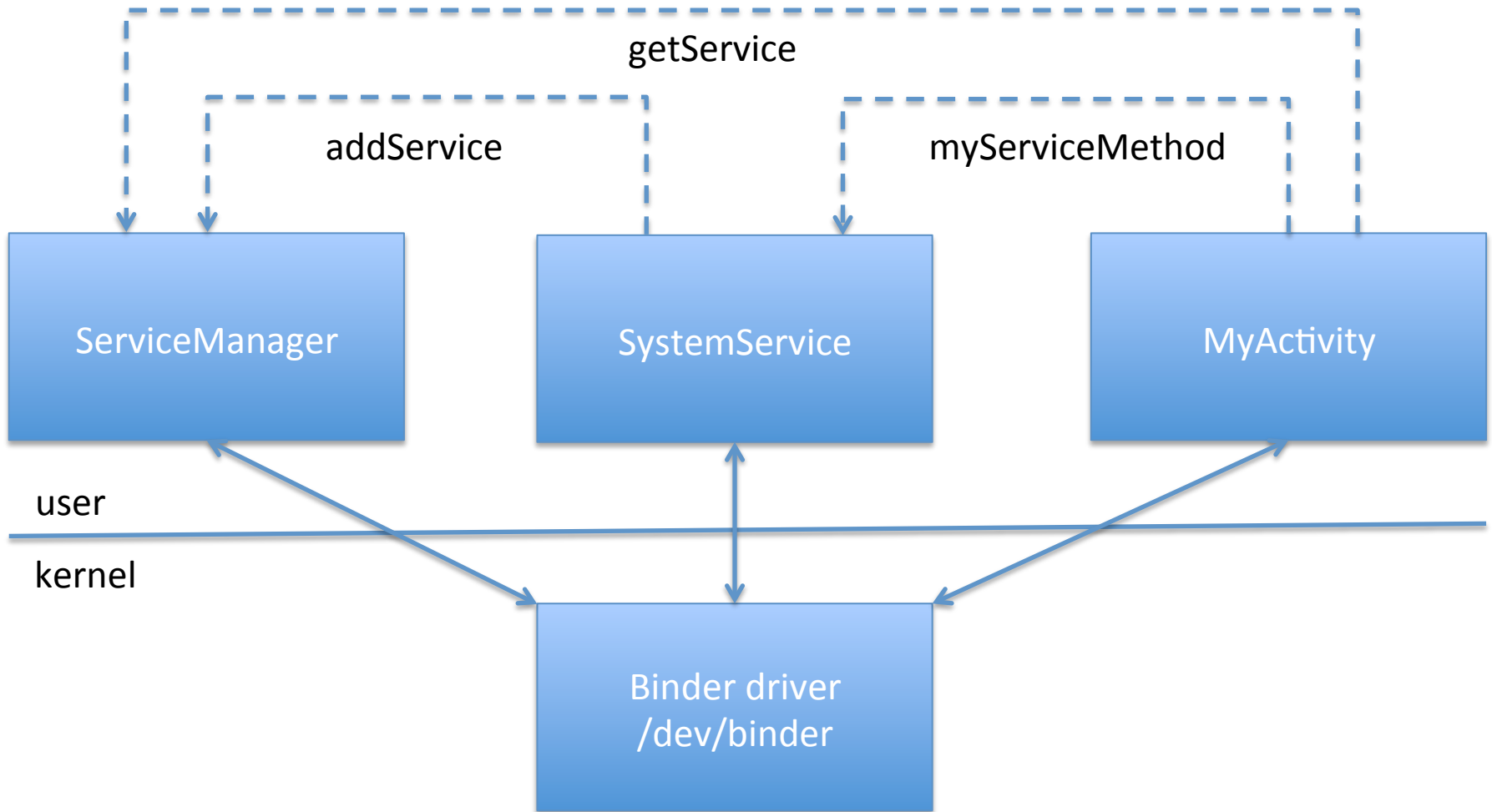
# Binder Functionality

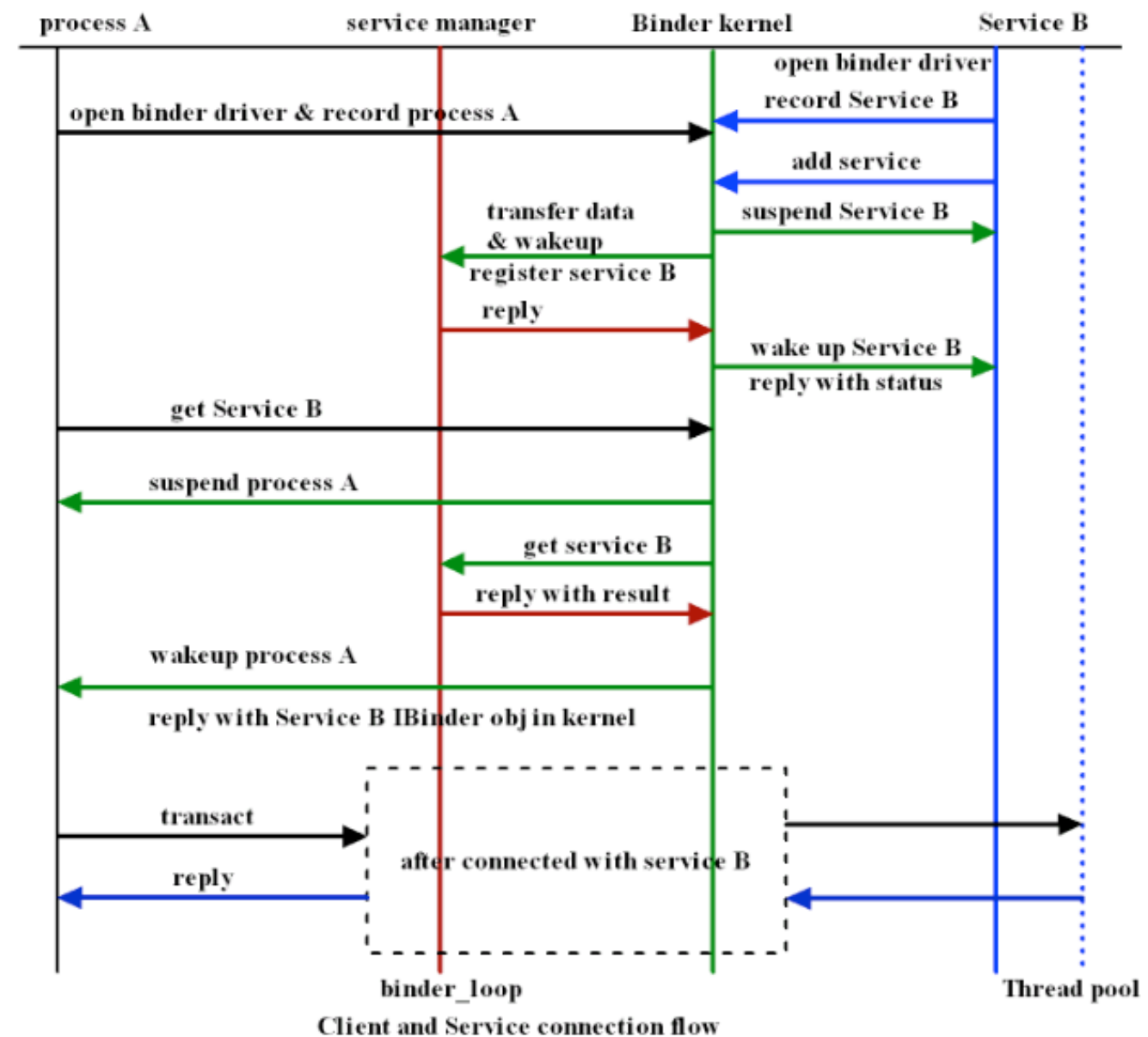
- Architecture
  - Binder kernel driver
  - Instance of Binder objects within user-space
    - Implements the IBinder interface
- Managing communication between processes
  - Simple inter-process messaging
    - Parcelable objects
  - Inter-process message calls
    - Call methods on remote objects as if they were local
  - Notifying processes of service events
- Identifying processes and services
  - Binder Token
    - Numerically uniquely identify a Binder instance
  - Basis of Android's **permissions** model
    - What are processes allowed to do?

# ServiceManager

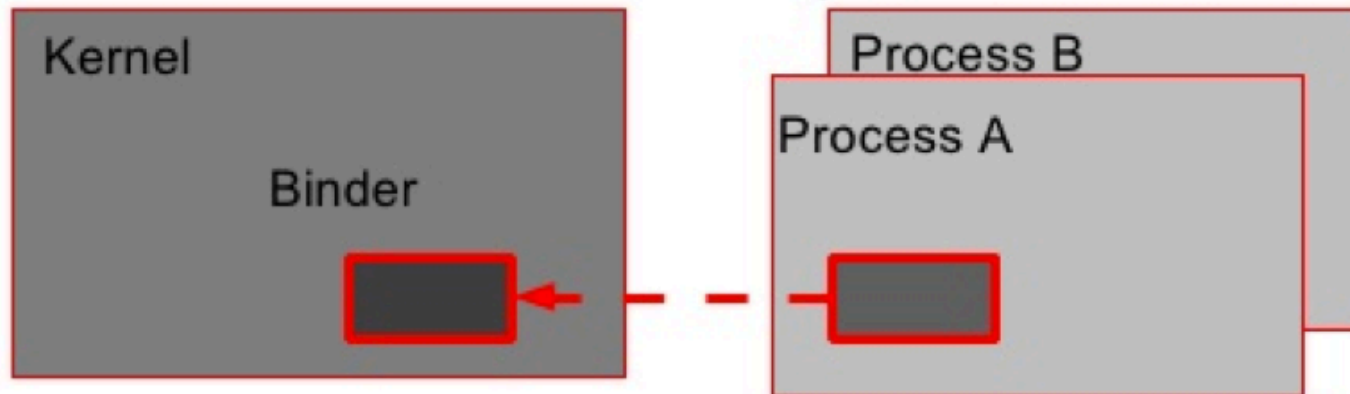
- A special Binder instance with a known Binder address
  - Hosts many system services within its process
  - Knows about other remote services
- Client does not know the token of remote Binder
  - Only the Binder interface knows its own address
- Binder submits a service name and its Binder token to the ServiceManager via IPC
  - Client retrieves remote service Binder address with service name
  - Client communicates with remote service

# ServiceManager

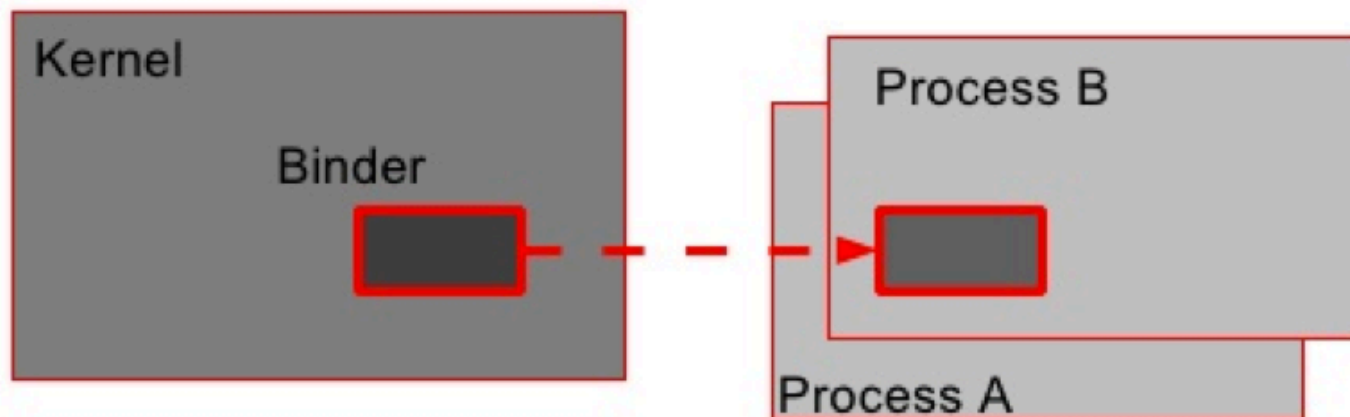




# Binder Transactions

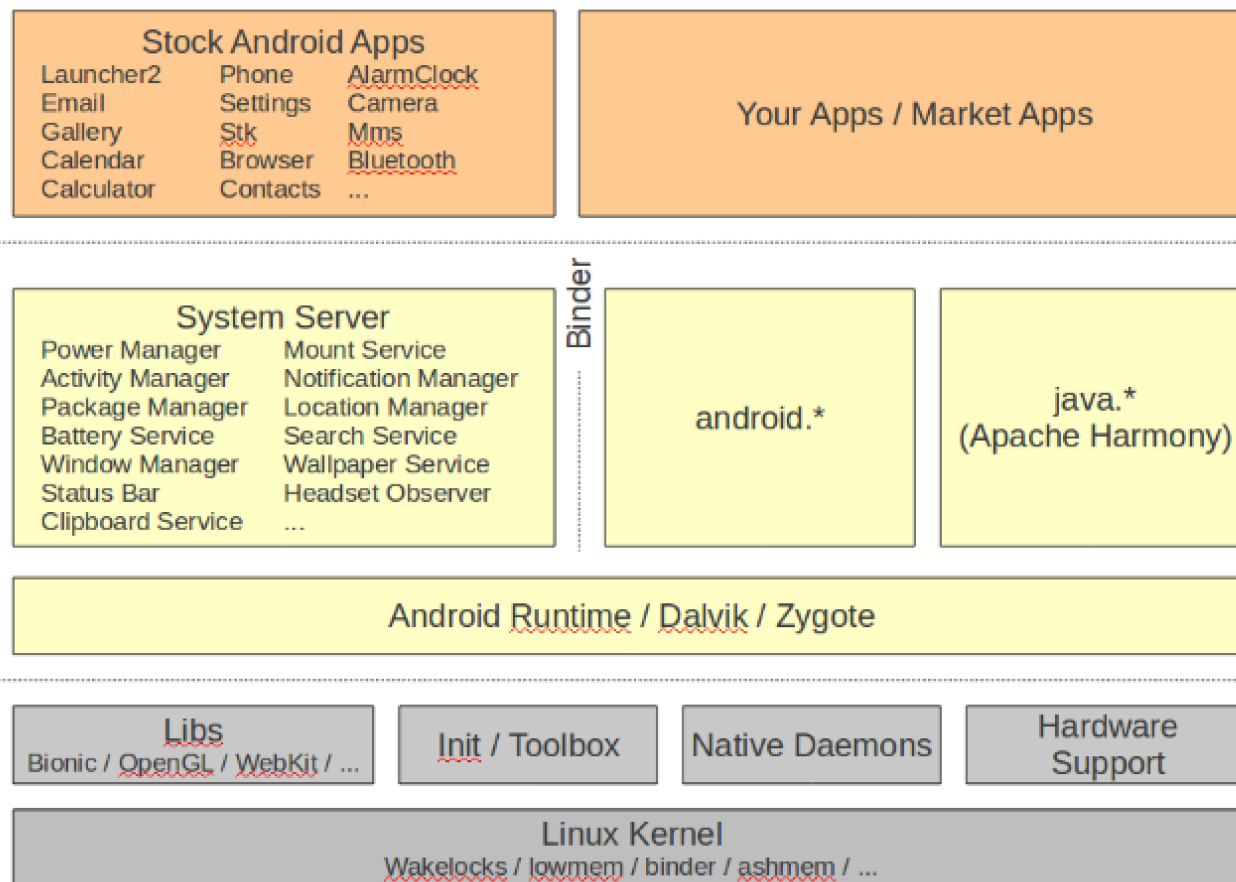


Copy memory by **copy\_from\_user**  
Then, wake up process B



Copy memory by **copy\_to\_user**





SDK, Eclipse, .apk

Manifest:  
Perms / SDK ver.

.dex, ddms

NDK, rootfs, initrc, adb

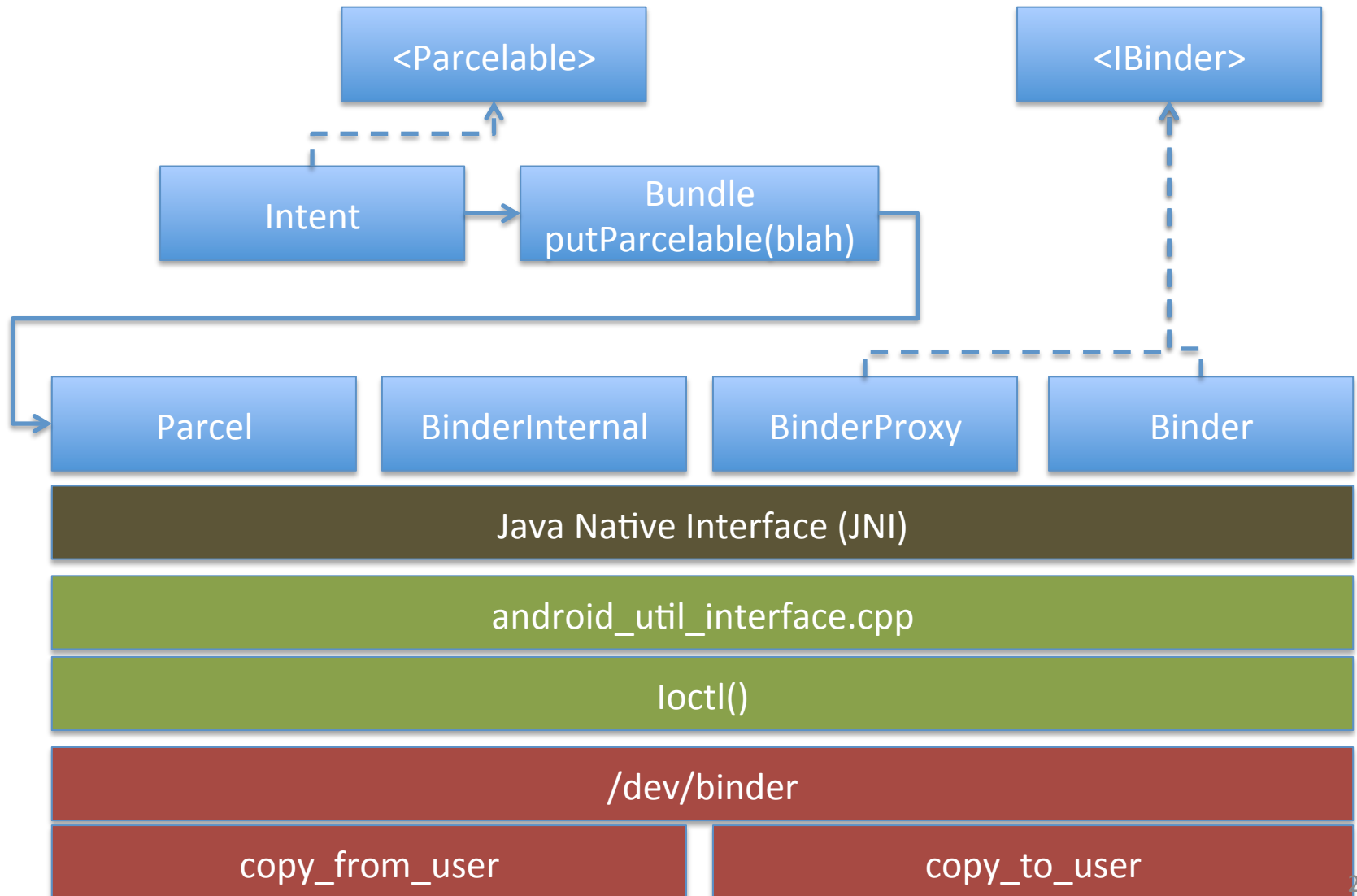
GNU toolchain

(fastboot)

# Binder Implementation

- API for apps
  - Written in Java
  - AIDL
  - Java API wrapper
    - Exposes the IBinder interface
    - Wraps the middleware layer
    - Parcelable object marshalling interface
- Middleware
  - Written in C++
  - 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 drivers
  - Written in C
  - 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`

# Binder Implementation



# Binder Performance / Limitations

- Binders communicate over process boundaries
  - Processes do not share a common virtual machine context
    - No direct access to objects
  - Not ideal of large data-streams
    - i.e. audio/video
    - Parcelable overhead
  - Good enough for window / activity / surface management
- Advantages
  - Native binary marshalling
    - Not java serialisation
  - Support of ashmem shared memory
- Disadvantages
  - Overhead of Dalvik Parcel marshalling
  - ioctl() not optimal
  - Passes file descriptors for faster binary data transfer

# Binder Security

- Binder Security Features
  - Client identity managed by the kernel
    - `Binder.getCallingUid()`, `Binder.getCallingPid()`
  - Interface reference security
    - Client cannot guess “address” of a service without going via the Service Manager
- Service Manager
  - A directory service for system services
    - Mediate access
  - Revoke access based on token
- Service could check client permissions at run-time
  - `Context.checkPermission(permission, pid, uid)`

# Services recap

- A second kind of Android component
  - An abstraction of Binder / IPC
    - Used throughout the Android OS
- Tightly or loosely coupled to Activities
  - Start / destroy
    - Either by the Application
      - If we start it, it will run until we stop it
    - Or by the OS
      - If the OS starts it because it was bound, the OS destroys it when it is unbound
  - Communicate tightly via a Binder instance
    - Locally or remotely across processes
  - Communicate loosely via Notifications / Intents / Messages

# References

- <http://developer.android.com/guide/components/processes-and-threads.html>
- <http://developer.android.com/guide/components/services.html>
- [http://elinux.org/Android Binder](http://elinux.org/Android_Binder)