

# G53MDP

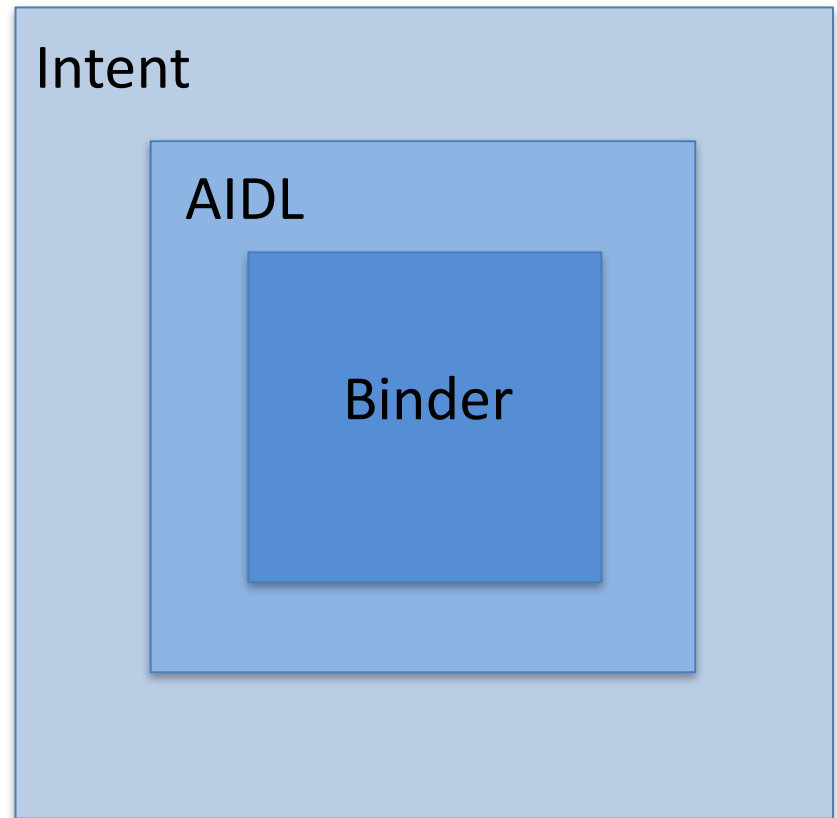
# Mobile Device Programming

Binder and IPC

Storage

# IPC Abstraction

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



```
root@android:/ # service list
```

```
Found 68 services:
```

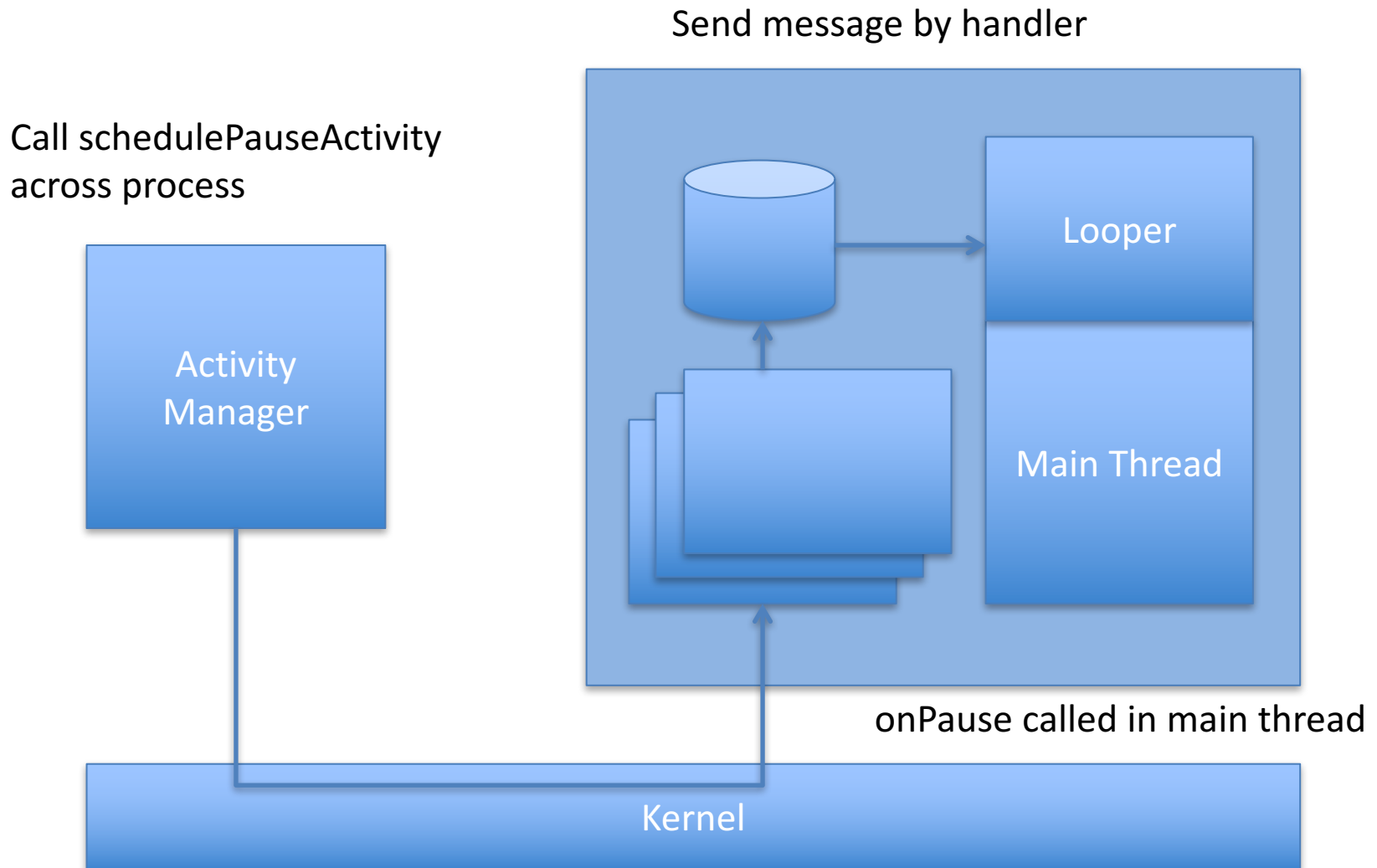
```
0      phone: [com.android.internal.telephony.ITelephony]
1      iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
2      simphonebook: [com.android.internal.telephony.IIccPhoneBook]
3      isms: [com.android.internal.telephony.ISms]
4      dreams: [android.service.dreams.IDreamManager]
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]
```

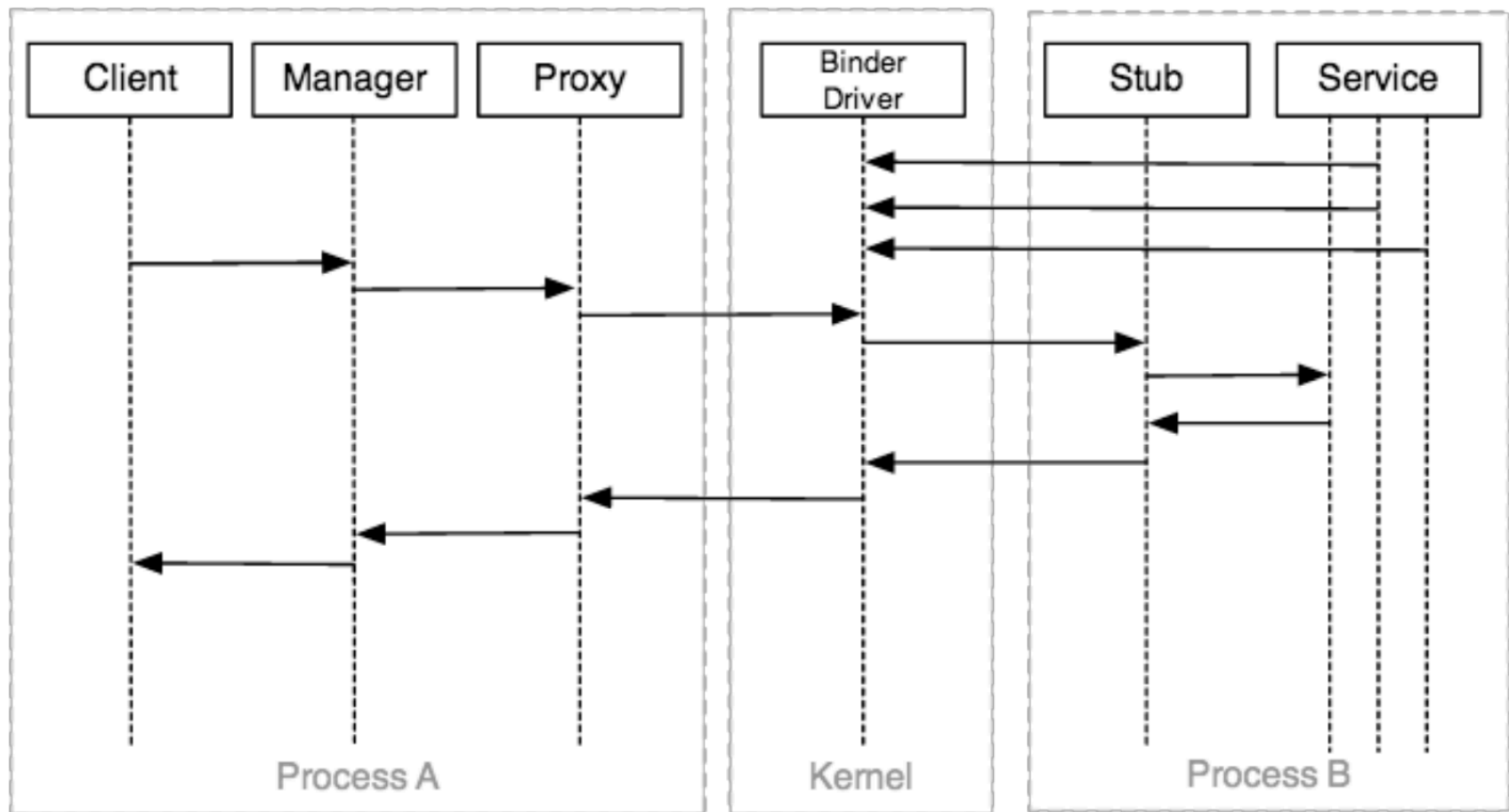
root	29	1	276	156	c0098770	0000e840	S	/sbin/ueventd
system	30	1	836	344	c0195c08	40036fc0	S	/system/bin/servicemanager
root	31	1	4008	820	ffffffff	4003e76c	S	/system/bin/vold
root	33	1	8632	1232	ffffffff	4006a76c	S	/system/bin/netd
root	34	1	880	388	c01a10a0	40037a70	S	/system/bin/debuggerd
radio	35	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

# System 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

# onPause()





# Binder Objects and Tokens

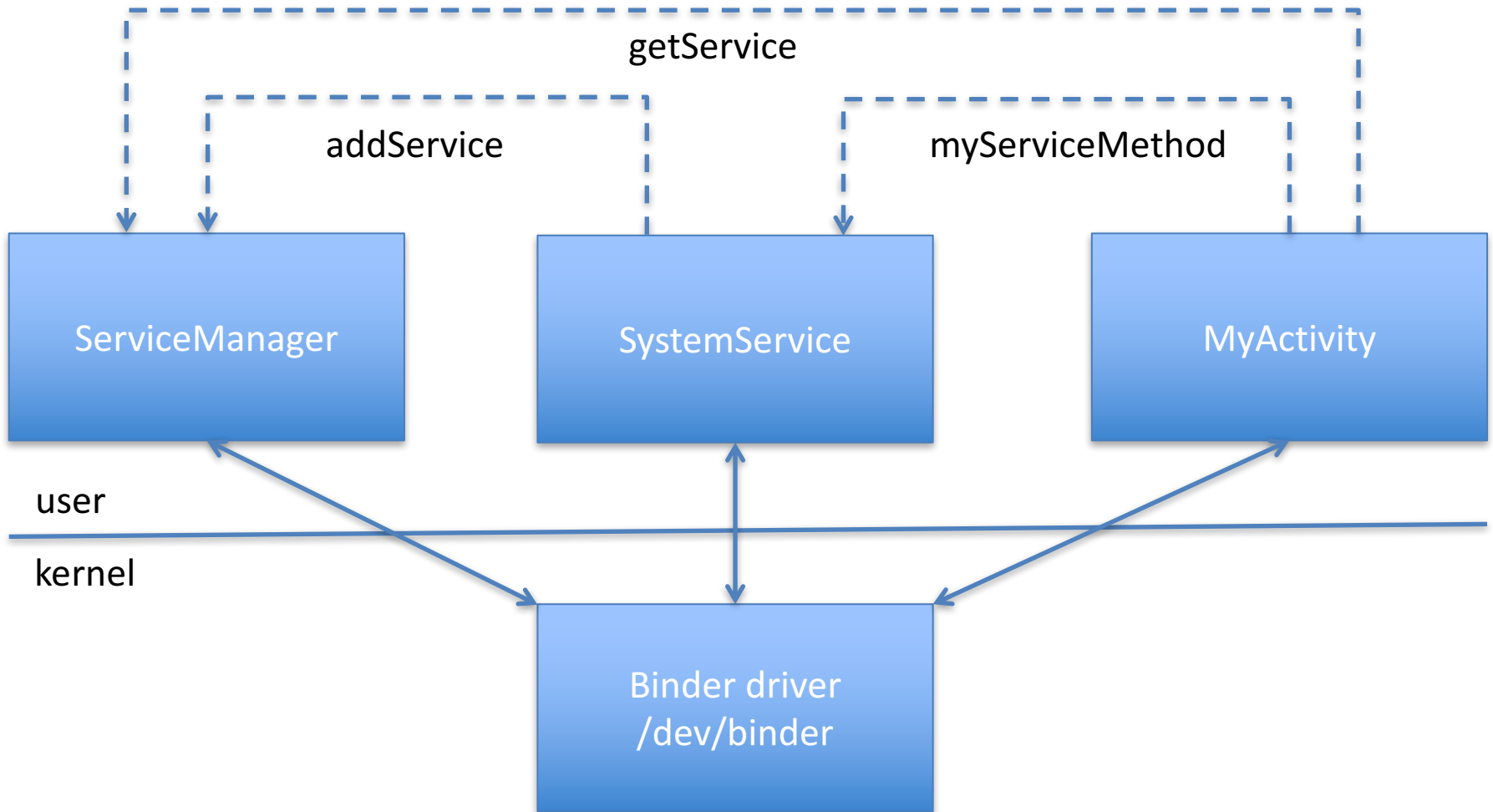
- Binder Object
  - An object that can be accessed through the Binder framework
  - Implements the *IBinder* interface
  - A unique identity maintained across processes
    - Allocated by the Binder driver
      - Cannot be duplicated
    - A 32 bit handle maintained by the kernel
- Process A creates a binder object <- references memory directly
  - Passes it to process B <- referenced by handle
    - Passes it to process C <- referenced by handle
- Capability-based security model
  - Processes are granted access to a particular resource by giving them a *capability* in the form of the binder object
    - Binder object as **token**
  - The possession of a token grants the owning process full access to the Binder object enabling it to perform Binder transactions on the target object
    - The only way to communicate with a Binder object is to be given a reference to it

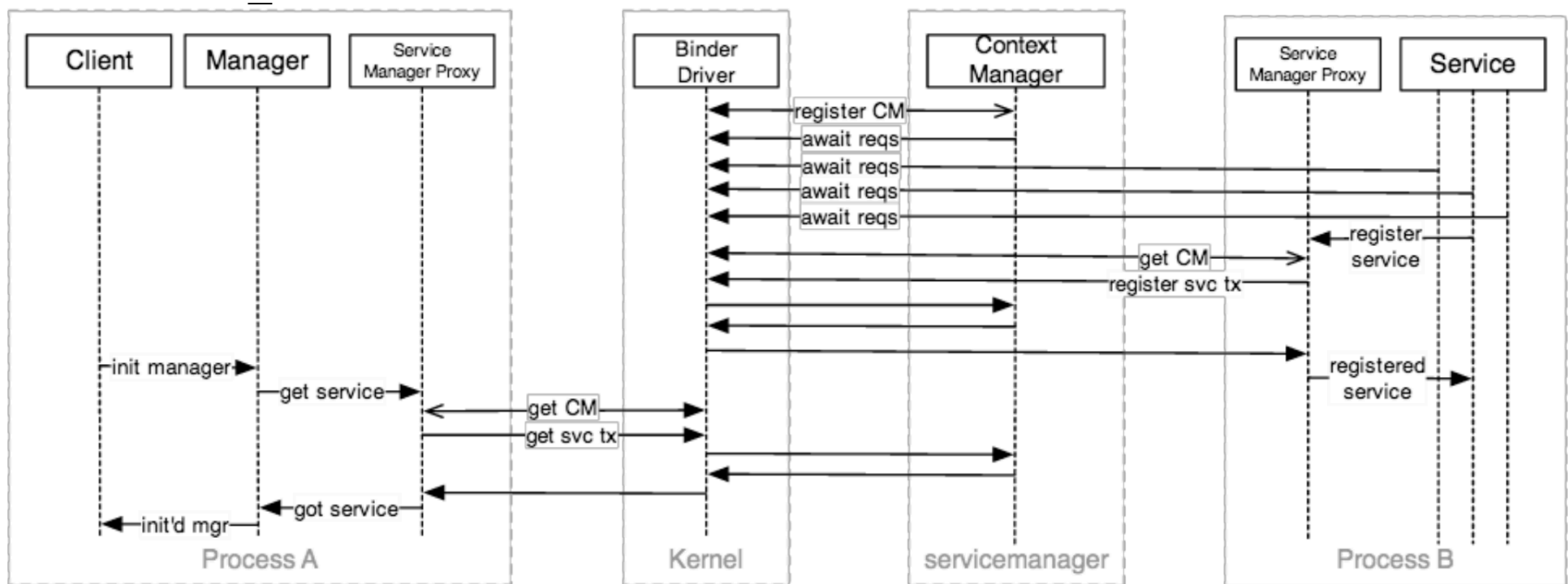


# ServiceManager

- So how do we get the token?
- A single *context manager* that maintains references to Binder objects
  - Implemented as ServiceManager
    - Hosts many system services within its process
  - A Binder instance with a known Binder handle (0)
  - Knows about other remote services
    - The first to be registered with Binder
    - Only “trusted” system services allowed to register
      - System, radio, media
- 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 handle with service name
  - Client communicates with remote service

# ServiceManager





```

public class MainActivity extends Activity {

    private PowerManager.WakeLock wakeLock;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        PowerManager pm =
            (PowerManager) getSystemService(Context.POWER_SERVICE);
        wakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "My Tag");
        wakeLock.acquire();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        wakeLock.release();
    }
}

```

```

public final class PowerManager {

    private final IPowerManager mService = null;

    public WakeLock newWakeLock(int levelAndFlags, String tag) {
        return new WakeLock(levelAndFlags, tag);
    }
}

```

```

public final class WakeLock {
    private final IBinder mToken;
    private final int mFlags;
    private final String mTag;

    WakeLock(int flags, String tag) {
        mToken = new Binder();
        mFlags = flags;
        mTag = tag;
    }

    public void acquire() {
        mService.acquireWakeLock(mToken, mFlags, mTag);
    }

    public void release() {
        mService.releaseWakeLock(mToken);
    }
}

```

[GrepCode](#) / [com.google.android / android](#)

```

updateWakeLockoids(IBinder, int[]) : void
updateWakeLockWorkSource(IBinder, Work
userActivity(long, int, int) : void
wakeUp(long) : void

```

Proxy in IPowerManager.Stub

```

Proxy(IBinder) : void
acquireWakeLock(IBinder, int, String, String, W
acquireWakeLockWithUid(IBinder, int, String, S
asBinder() : IBinder
boostScreenBrightness(long) : void
crash(String) : void

```

# Binder Security

- Binder doesn't deal with security
  - Enables a **trusted** execution environment
  - Transactions via the kernel
    - Client identity managed by the kernel
      - `Binder.getCallingUid()`, `Binder.getCallingPid()`
      - UID / PID included in each transaction
- Access controlled in two ways
  - Limit who can obtain a reference to a Binder object
    - Interface reference security
    - Client cannot guess “address” of a service without going via the Service Manager
  - Check caller identity before performing an action on the Binder objects
    - Service asks package manager about UID permissions
    - Check whether it holds a permission we want to enforce via `PackageManager.getPackageInfo(...)`
      - Another system service!

# Binder Performance

- Reference counting and Death notifications
  - Binder objects automatically freed when no longer referenced
  - Can be notified when a remote binder host process dies
  - Implemented in the kernel driver
- Explicit limitations
  - Transactional buffer size 1Mb per process for all concurrent transactions
    - Many moderately sized transactions could also exhaust its limit
      - Arguments and return values are too large
    - Keep transaction data small
- Implicit limitations
  - Data is copied
    - Duplication of resources
  - Native binary marshalling
    - Better than reflection based serialization
    - Still has overhead of parcel marshalling
      - Read byte, read byte, read byte
  - Not ideal for large data-streams
    - Good enough for window / activity / surface management
    - Pass file descriptors to shared memory regions (**ashmem** - anonymous shared memory)

# References

- <http://developer.android.com/guide/components/services.html>
- [http://elinux.org/Android\\_Binder](http://elinux.org/Android_Binder)
- [http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/5.1.1\\_r1/android/os/IPowerManager.java#IPowerManager](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/5.1.1_r1/android/os/IPowerManager.java#IPowerManager)
- [http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/5.1.1\\_r1/android/os/PowerManager.java](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/5.1.1_r1/android/os/PowerManager.java)