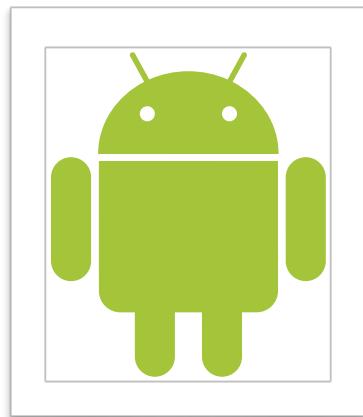


Porting Generic Android™ Drivers and 64-bit Binder ABI

Linux Plumbers, October 2014



Şerban Constantinescu
Systems & Software, ARM®

About Me

- Software Engineer @ ARM®
 - Versatile Express Android ports
 - Binder
 - Ashmem
 - Bionic
 - Dalvik
 - ART
 - Other Android bits

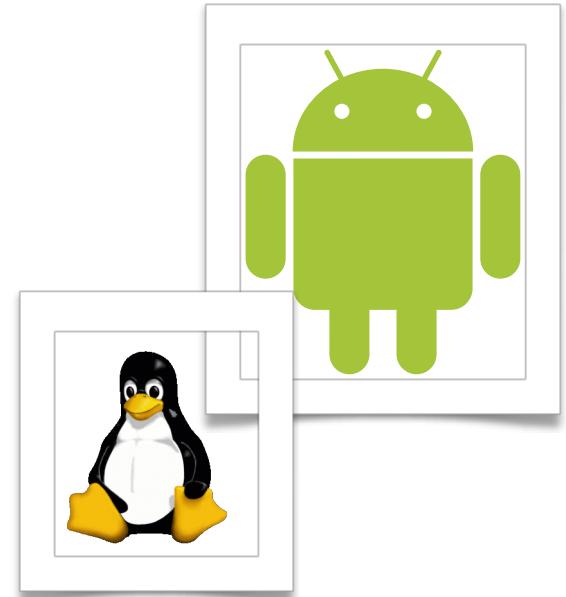
Overview

- Portable Android Drivers
 - Generic Issues
 - Guidelines
- Binder 64-bit Support
 - Support for 32-bit Processes
 - 64-bit Binder ABI(Application Binary Interface)
- Replacing Binder

Portable Android Drivers

Android Environment

- Android on 64-bit kernels
 - Support for 32-bit userspace
 - Compat layer
 - Support for 64-bit userspace
 - 64-bit kernel/userspace ABI
- Android on 32-bit kernels
 - Existing 32-bit kernel & userspace



loctl Interface

```
size_t size = 0x100;  
open("/dev/ashmem", O_RDWR);  
ioctl(fd, ASHMEM_SET_SIZE, size);
```

Userspace

Kernel



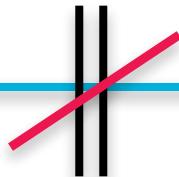
```
long ashmem_ioctl(file, cmd, arg)  
{  
    switch (cmd) {  
        case ASHMEM_SET_SIZE:  
            return 0;  
        default:  
            return -ENOTTY;  
    }  
}
```

loctl Interface

```
#define ASHMEM_SET_SIZE      _IOW('a', 3, size_t)
#define _IOW(type,nr,size)    _IOC(_IOC_W,(type),(nr),sizeof(size))
```

32-bit

64-bit



```
#define ASHMEM_SET_SIZE      _IOW('a', 3, size_t)
#define _IOW(type,nr,size)    _IOC(_IOC_W,(type),(nr),sizeof(size))
```

loctl Interface

```
#define ASHMEM_SET_SIZE      _IOW('a', 3, size_t)
```

32-bit

64-bit

```
#define ASHMEM_SET_SIZE      _IOW('a', 3, size_t)
#define COMPAT_ASHMEM_SET_SIZE _IOW('a', 3, compat_size_t)
```

File Operation Table

```
static const struct file_operations ashmem_fops = {
    .owner = THIS_MODULE,
    .open = ashmem_open,
    .release = ashmem_release,
    .read = ashmem_read,
    .llseek = ashmem_llseek,
    .mmap = ashmem_mmap,
    .unlocked_ioctl = ashmem_ioctl,           /* 64-bit entry */
#ifdef CONFIG_COMPAT
    .compat_ioctl = ashmem_ioctl,             /* 32-bit entry */
#endif
};
```

File Operation Table

```
static const struct file_operations ashmem_fops = {
    .owner = THIS_MODULE,
    .open = ashmem_open,
    .release = ashmem_release,
    .read = ashmem_read,
    .llseek = ashmem_llseek,
    .mmap = ashmem_mmap,
    .unlocked_ioctl = ashmem_ioctl,           /* 64-bit entry */
#ifdef CONFIG_COMPAT
    .compat_ioctl = compat_ashmem_ioctl,      /* 32-bit entry */
#endif
};
```

Compat ioctl

```
#ifdef CONFIG_COMPAT
static long compat_ashmem_ioctl(file, cmd, arg)
{
    switch (cmd) {
        case COMPAT_ASHMEM_SET_SIZE:
            cmd = ASHMEM_SET_SIZE;
            break;
        case COMPAT_DUMMY_PTR:
            /* use compat_ptr() & ptr_to_compat() */
            cmd = DUMMY_PTR;
            break;
    }
    return ashmem_ioctl(file, cmd, arg);
}
#endif
```

ABI

```
struct flat_binder_object {  
    /* 8 bytes for large_flat_header.* /  
    unsigned long type;  
    unsigned long flags;  
    /* data. */  
    union {  
        void *binder;  
        signed long handle;  
    };  
    /* extra data. */  
    void *cookie;  
};
```

ABI

```
struct flat_binder_object {
    /* 8 bytes for large_flat_header.*/
    __u32 type;           /* unsigned long      type */
    __u32 flags;          /* unsigned long      flags */
    /* data. */

    union {
        void __user *binder; /* void *binder      */
        __u32 handle;       /* signed long handle */
    };
    /* extra data. */
    void __user *cookie;   /* void *cookie      */
};

};
```

Alignment

32-Bit

```
struct binder_handle_cookie {  
    __u32 handle;  
    __u64 cookie;  
};  
  
sizeof() : 12  
alignof() : 4
```

64-Bit

```
struct binder_handle_cookie {  
    __u32 handle;  
    __u64 cookie;  
};  
  
sizeof() : 16  
alignof() : 8
```

Alignment

32-Bit

```
struct binder_handle_cookie {  
    __u32 handle;  
    __u64 cookie;  
};  
  
sizeof() : 12  
alignof() : 4
```

64-Bit

```
struct binder_handle_cookie {  
    __u32 handle;  
    __u64 cookie;  
} __attribute__((packed));  
  
sizeof() : 12  
alignof() : 1
```

Portable Android Device Drivers

- Use explicit sized types (`linux/types.h`)
- Use native kernel types - e.g `pid_t`, `key_t`, `gid_t`, etc.
- Use compat types and expansion macros (`linux/compat.h`)
- Ensure alignment of individual fields of a structure
- Provide a thin [compat_ioctl\(\)](#)

- 32-bit Compat and 64-bit ABI Stats:

<code>drivers/staging/android/binder.c</code>	73	+++++-----
<code>drivers/staging/android/binder.h</code>	34	++++-
<code>drivers/staging/android/ashmem.c</code>	21	+++++-----
<code>drivers/staging/android/ashmem.h</code>	7	+++++

Portable Android Device Drivers

- **Upstream your changes!**

- Documentation
 - [Kernel-Tutorial](#)



Fix your driver ...

Build and test at least for ARM, ARM64, x86, x86_64

Run:

- *<kernel>/scripts/checkpatch.pl*
- *<kernel>/scripts/get_maintainer.pl*

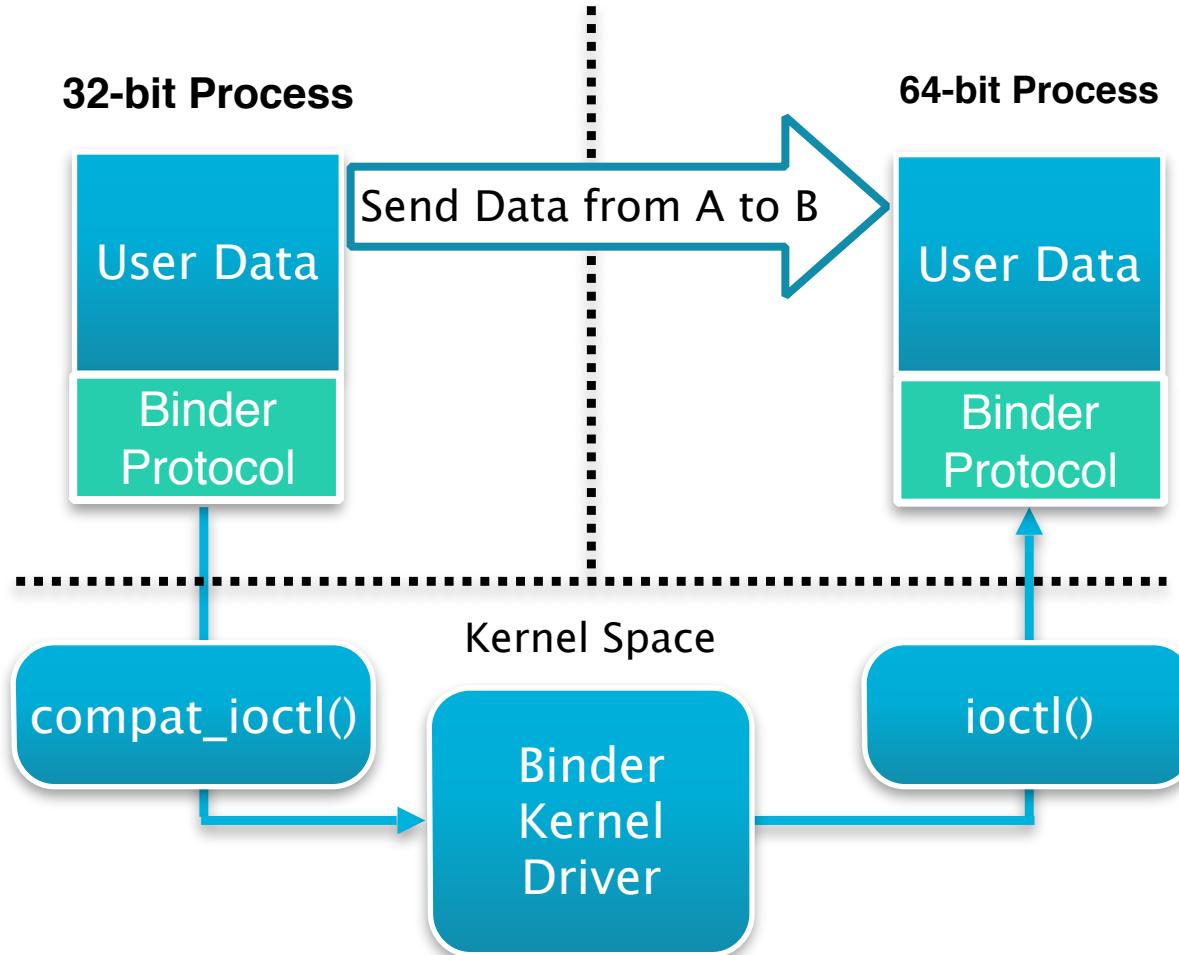
Submit your patch upstream

Binder Compat Layer

Binder Overview

- Android's Inter Process Communication Mechanism
- *“In the Android platform, the binder is used for nearly everything that happens across processes in the core platform.”**
- Designed for fast IPC on devices with “*almost no RAM and very low CPU resources.*”**
 - Kernel driver: <kernel>/drivers/staging/android/binder.c
 - Userspace HAL: <aosp>/frameworks/native/libs/libbinder

Binder IPC Diagram



Binder Compat

- Not a conventional kernel compat layer
 - **Userspace compat layer**
 - All the expansion is done in the userspace
 - Same ABI for 32-bit & 64-bit processes
- 64-bit Binder ABI used for:
 - 64-bit Userspace
 - 32-bit Userspace running on 64-bit kernels

Binder Compat

- Most of the changes hidden in the UAPI structures
 - Kernel headers exported to the userspace
 - Pristine headers: `<aosp>/external/kernel-headers/original/uapi`
 - **Userspace headers:** `<aosp>/bionic/libc/kernel/uapi`
 - Generated using: `<aosp>/libc/kernel/tools/update_all.py`
- Userspace ABI build switch: `TARGET_USES_64_BIT_BINDER`

Binder Userspace Compat

```
struct binder_transaction_data {
    union {
        __u32 handle;
        void *ptr;
    } target;
    void *cookie;
    __u32 code;
    __u32 flags;
    pid_t sender_pid;
    uid_t sender_euid;
    size_t data_size;
    size_t offsets_size;
    union {
        struct {
            void *buffer;
            void *offsets;
        } ptr;
        __u8 buf[8];
    } data;
};
```

Binder Userspace Compat

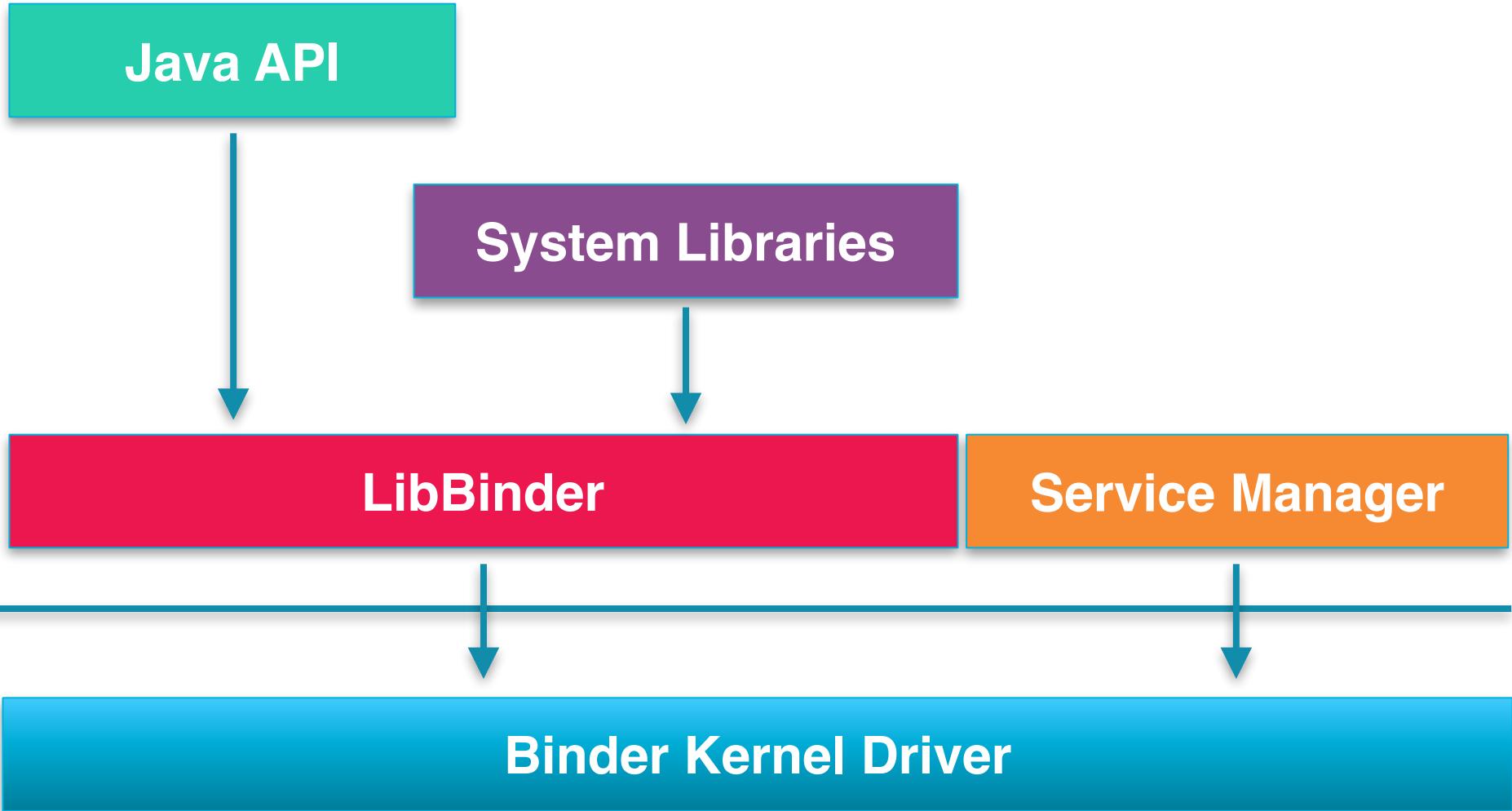
```
struct binder_transaction_data {
    union {
        __u32 handle;
        binder_uintptr_t ptr;
    } target;
    binder_uintptr_t cookie;
    __u32 code;
    __u32 flags;
    pid_t sender_pid;
    uid_t sender_euid;
    binder_size_t data_size;
    binder_size_t offsets_size;
    union {
        struct {
            binder_uintptr_t buffer;
            binder_uintptr_t offsets;
        } ptr;
        __u8 buf[8];
    } data;
};
```

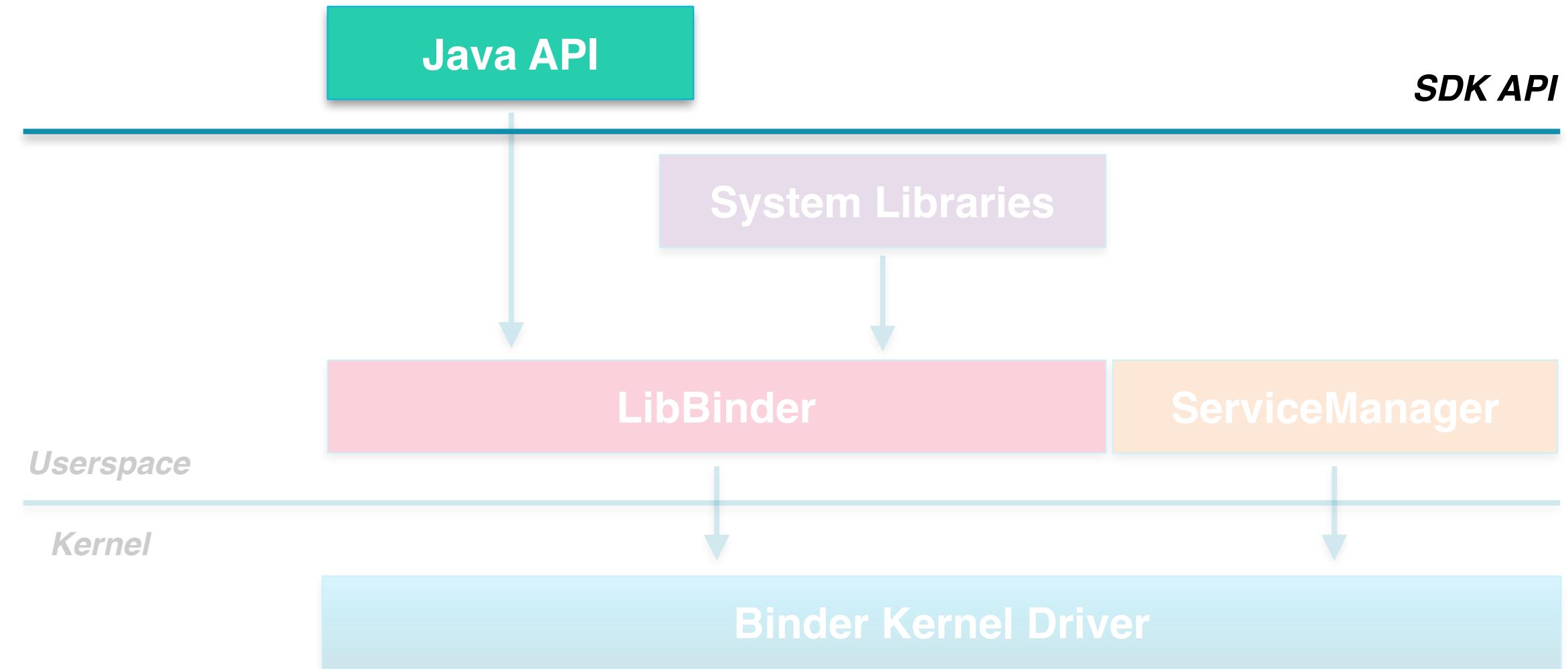
```
#ifdef BINDER_IPC_32BIT
typedef __u32 binder_size_t;
typedef __u32 binder_uintptr_t;
#else
typedef __u64 binder_size_t;
typedef __u64 binder_uintptr_t;
#endif
```

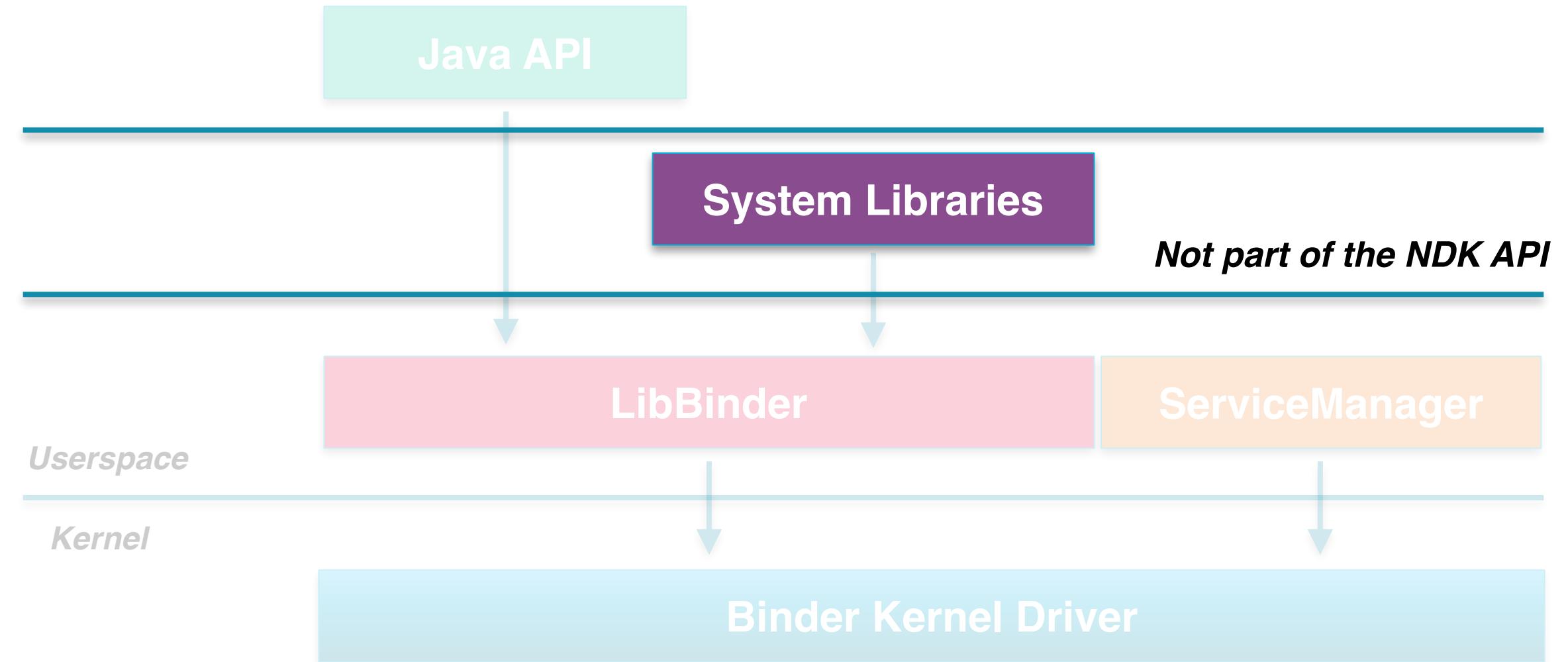
New Binder ABI for 64-bit!

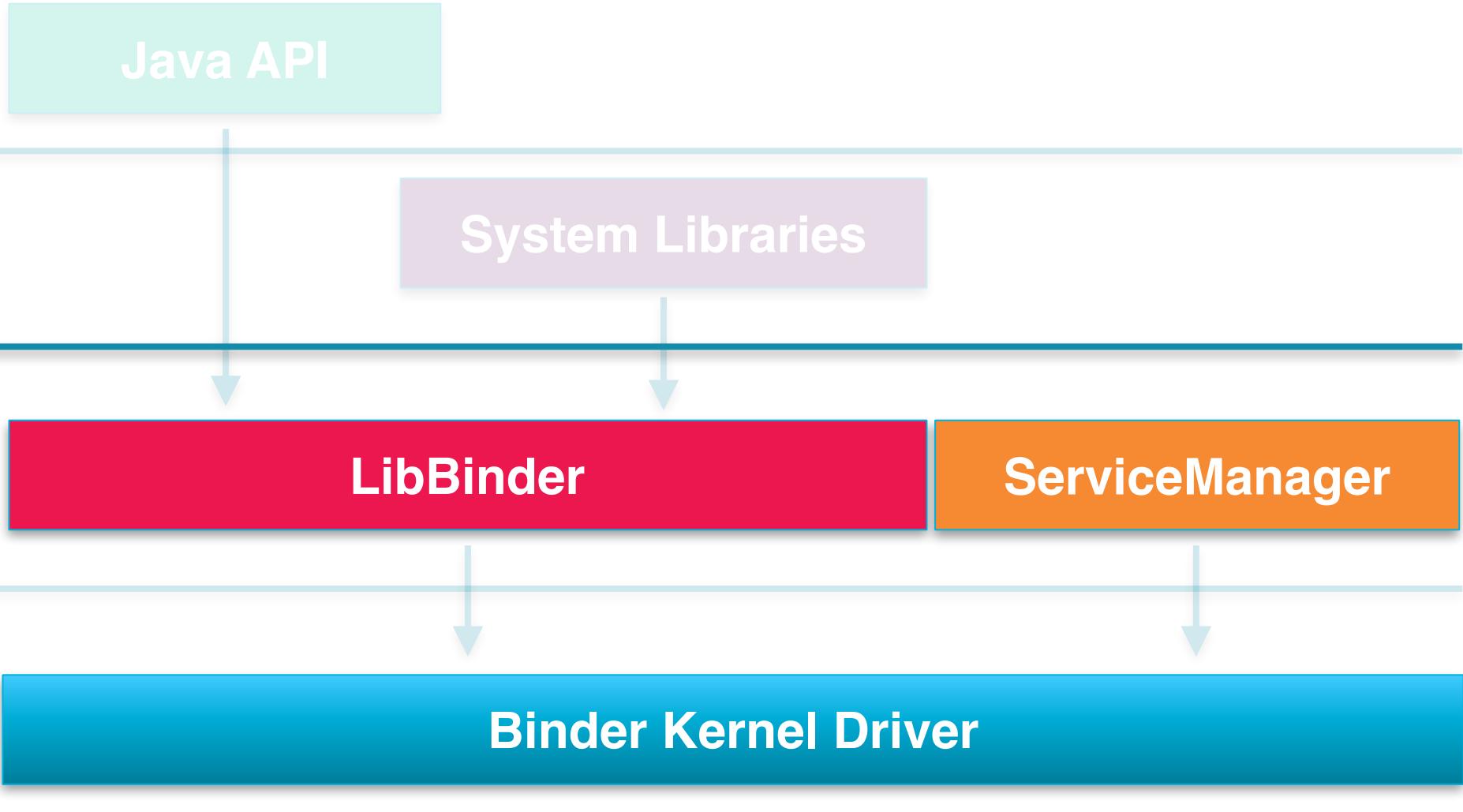
New Binder ABI

- Binder ABI:
 - Legacy 32-bit ABI (existing Android releases)
 - New 64-bit ABI (future Android releases)
- Android depends on the Binder functionality
 - But not on the low level Binder ABI
- **Binder can be replaced with a new IPC with similar functionality**
- No application compatibility failures caused by Binder ABI change









How can we replace Binder?

- **Replace the userspace LibBinder**
 - Provide the same public API
 - IPC with similar functionality

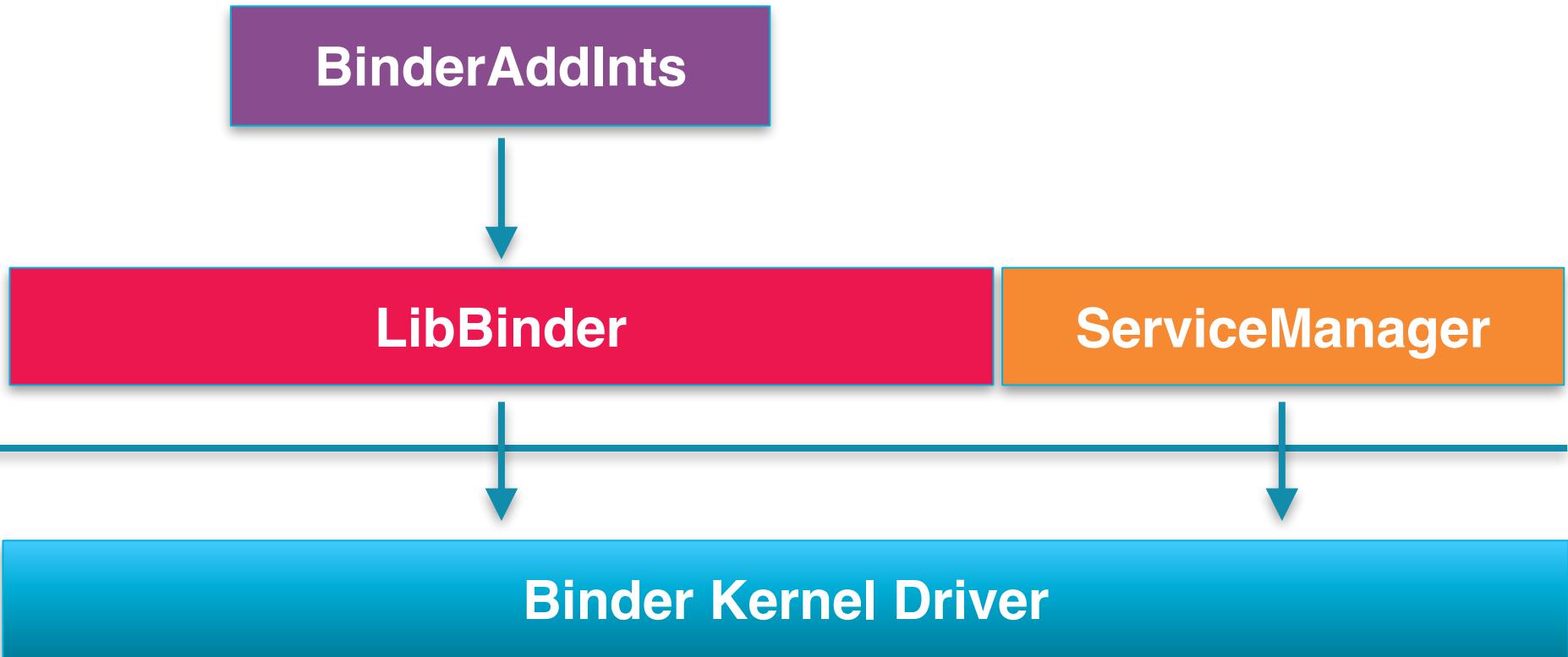
Replace LibBinder

- **1st step - New ServiceManager**
 - Implementation based on the API provided by LibBinder
 - No direct interaction with the kernel driver
 - [Available for review!](#)
- **2nd step - Proof of concept!**
 - Implement LibBinder functionality needed by BinderAddInts
 - Android Remote Procedure Call benchmark
- ...
 - Remove unused LibBinder public APIs
 - Implement remaining LibBinder API

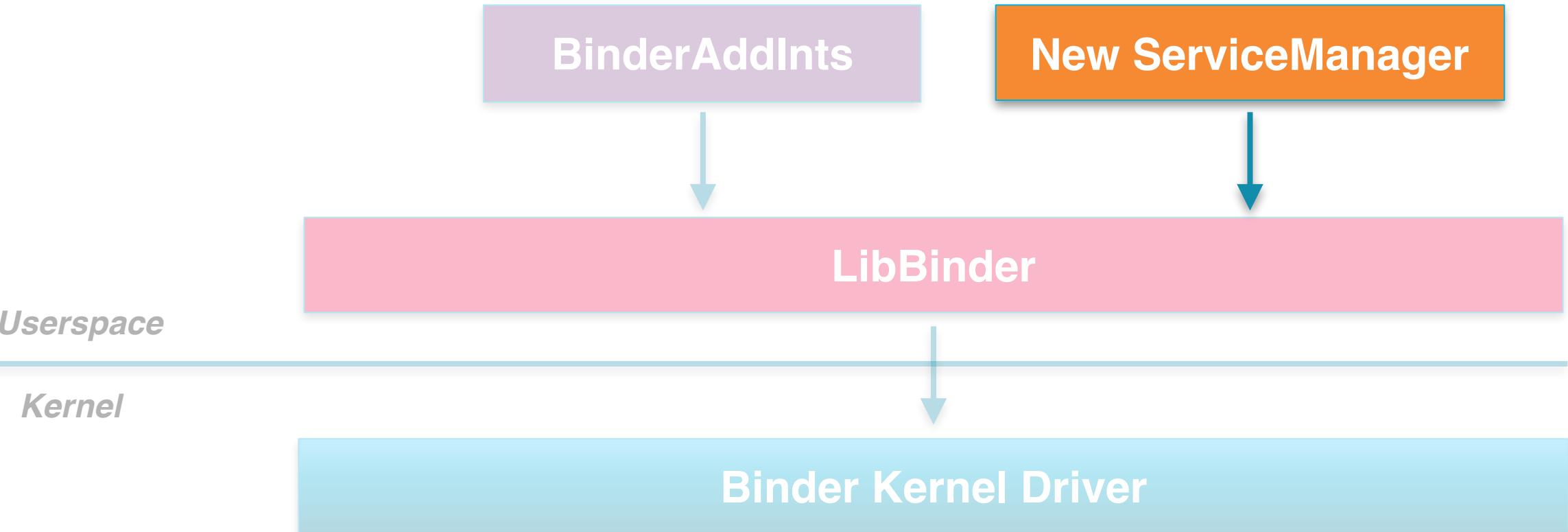
BinderAddInts

- Android RPC benchmark
 - Simple example of the Binder programming paradigm
 - Implementation in `<aosp>/system/extras/tests/binder/benchmarks`
- API Used
 - Server:
 - `ServiceManager->addService()`
 - `Binder->onTransact()`
 - Client:
 - `ServiceManager->getService()`
 - `Binder->transact()`

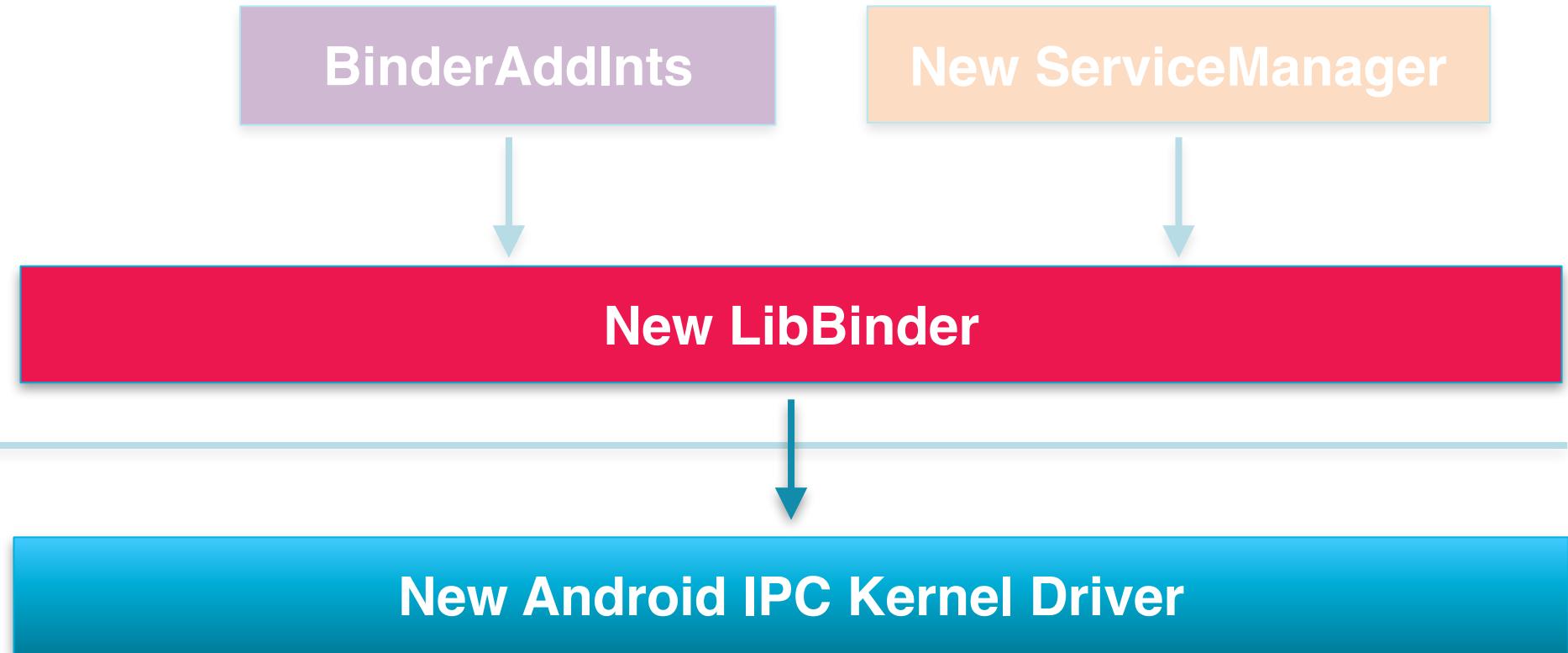
Binder Replacement - Proof of Concept



Binder Replacement - First Step



Binder Replacement - Second Step



Other ideas for replacing Binder?

Q & A

Thanks!

The trademarks featured in this presentation are registered and/or unregistered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

References

- **Writing Portable Device Drivers**
 - <http://www.linuxjournal.com/article/5783>
- **Kernel Tutorial**
 - <https://github.com/gregkh/kernel-tutorial/>
- **Kdbus Details**
 - <http://kroah.com/log/blog/2014/01/15/kdbus-details/>
- **Android Binder**
 - http://elinux.org/Android_Binder
- **Android Logo**
 - *The Android robot is reproduced or modified from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.*
- **Tux Logo**
 - *The Tux penguin is reproduced or modified from work created and shared by Larry Ewing and used according to terms described in the Creative Commons 3.0 Attribution License.*