Android Framework Traning: Binder

2014.2

Figo

Agenda

- **What is Binder**
- Binder driver
- Binder framework service
- How to use Binder?

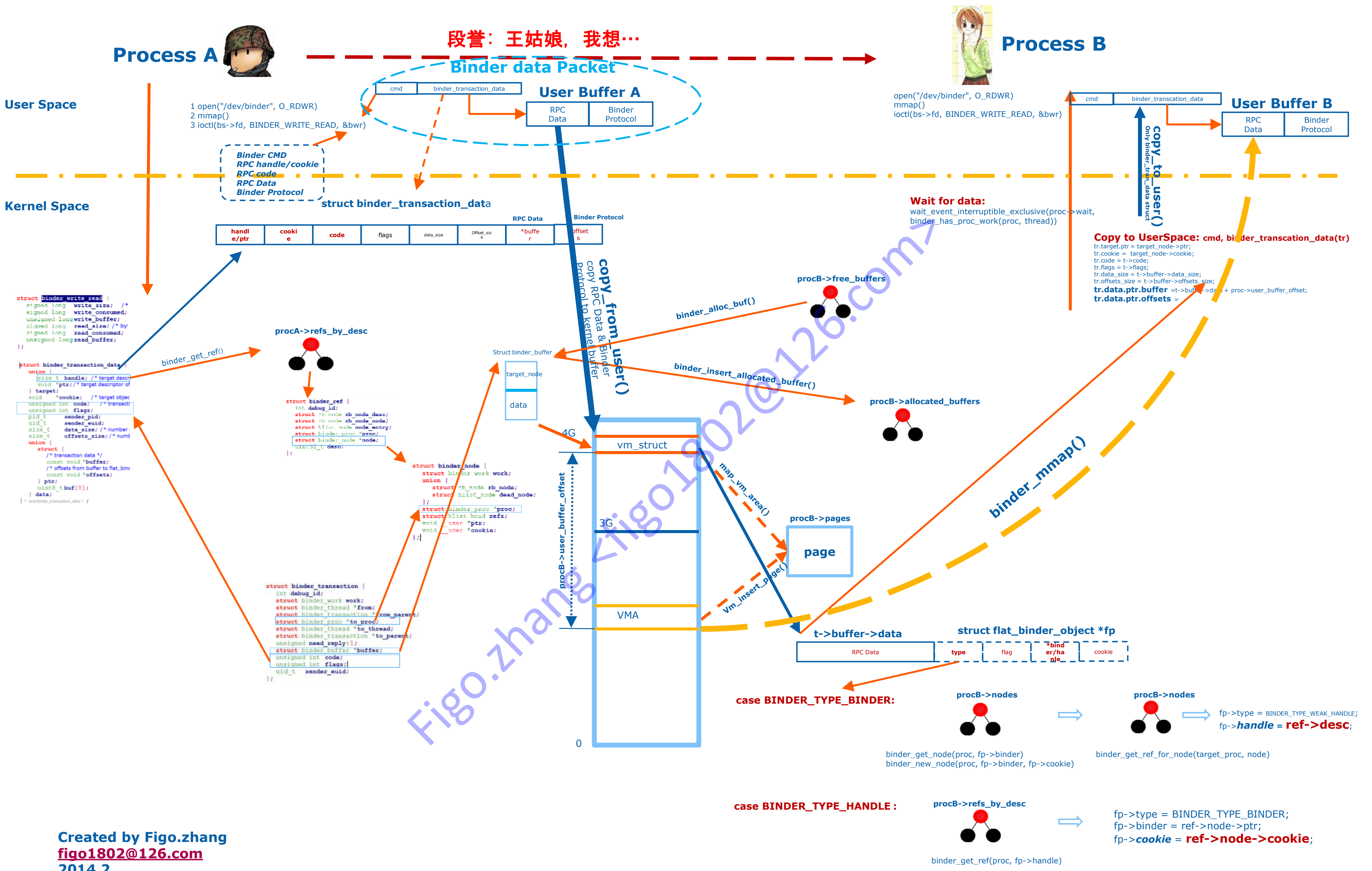
What is Binder

- Start with OpenBinder
- Why Android choose Binder?

Agenda

- What is Binder
- **Binder driver**
- Binder framework service
- How to use Binder?

Android Binder driver : Pa how to talk with Pb



```
struct binder_write_read {
    signed long   write_size;   /*
    signed long   write_consumed;
    unsigned long write_buffer;
    signed long   read_size; /* by
    signed long   read_consumed;
    unsigned long read_buffer;
};
```

```
struct binder_transaction_data {
    union {
        size_t  handle;  /* target descr
        void  *ptr; /* target descriptor of
    } target;
    void     *cookie;   /* target objec
    unsigned int  code;    /* transacti
    unsigned int  flags;
    pid_t       sender_pid;
    uid_t       sender_euid;
    size_t      data_size; /* number
    size_t      offsets_size; /* numb
    union {
        struct {
            /* transaction data */
            const void *buffer;
            /* offsets from buffer to flat_bind
            const void *offsets;
        } ptr;
        uint8_t buf[8];
    } data;
} /* end binder_transaction_data */ ;
```

```
struct binder_ref {
    int debug_id;
    struct rb_node rb_node_desc;
    struct rb_node rb_node_node;
    struct hlist_node node_entry;
    struct binder_proc *proc;
    struct binder_node *node;
    uint32_t desc;
};
```

```
struct binder_node {
    struct binder_work work;
    union {
        struct rb_node rb_node;
        struct hlist_node dead_node;
    };
    struct binder_proc *proc;
    struct hlist_head refs;
    void __user *ptr;
    void __user *cookie;
};
```

```
struct binder_transaction {
    int debug_id;
    struct binder_work work;
    struct binder_thread *from;
    struct binder_transaction *from_parent;
    struct binder_proc *to_proc;
    struct binder_thread *to_thread;
    struct binder_transaction *to_parent;
    unsigned need_reply:1;
    struct binder_buffer *buffer;
    unsigned int  code;
    unsigned int  flags;
    uid_t   sender_euid;
};
```

Copy to UserSpace: cmd, binder_transcation_data(tr)

```
tr.target.ptr = target_node->ptr;
tr.cookie =  target_node->cookie;
tr.code = t->code;
tr.flags = t->flags;
tr.data_size = t->buffer->data_size;
tr.offsets_size = t->buffer->offsets_size;
tr.data.ptr.buffer =t->buffer->data + proc->user_buffer_offset;
tr.data.ptr.offsets =
```

Created by Figo.zhang
figo1802@126.com
2014.2

Agenda

- What is Binder
- Binder driver
- **Binder framework service**
- How to use Binder?

Binder Framework - Step 1: Register service

AudioFlinger::instantiate()

defaultServiceManager()->**addService**("media.audio_flinger", **new AudioFlinger**);

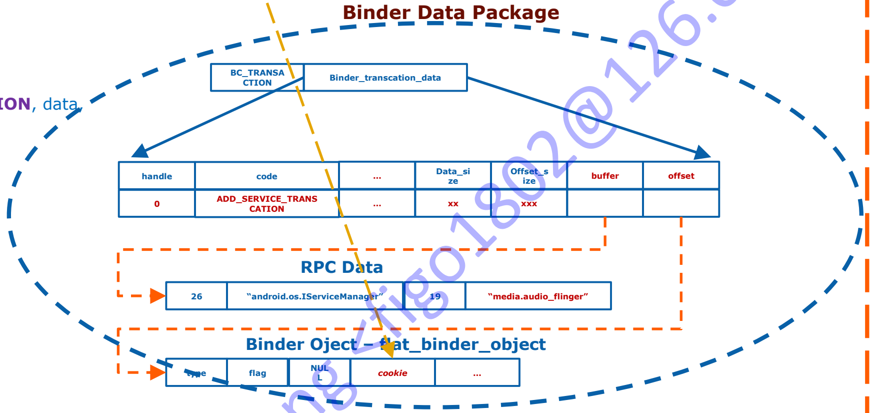
interface_cast<IServiceManager>(
ProcessState::self()->getContextObject(NULL));

BpServiceManager(new BpBinder(0))

BpServiceManager->addService:
data.writeInterfaceToken();
data.**writeString16**(name);
data.**writeStrongBinder**(service);
data.writeInt32();
remote()->transact(**ADD_SERVICE_TRANSACTION**, data, &reply);

BpBinder::transact:
IPCThreadState::self()->transact(
mHandle, code, data, reply, flags);

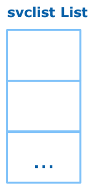
IPCThreadState::writeTransactionData
IPCThreadState::talkWithDriver



Send service name, cookie

ServiceManager (handle = 0)

svcmgr_handler: SVC_MGR_ADD_SERVICE



1. Get **service name from RPC data** :
2. Get binder object pointer (***handle***) from **flat_binder_object**
3. New service element, add to svclist

Send service name, handle to ServiceManager

Binder Driver

case BINDER_TYPE_BINDER:

proc->nodes

- Find binder_node by handle : binder_get_node(proc, fp->binder)
- New binder_node : binder_new_node(proc, fp->binder, fp->cookie)

node->cookie point to AudioFlinger

proc->nodes

binder_get_ref_for_node(target_proc, node)

fp->type = BINDER_TYPE_WEAK_HANDLE;
fp->***handle*** = **ref->desc**;

Binder Framework Step 2: get service

Binder Framework Step 3: RPC Call

Binder Framework – summary (For example: AudioFlinger)

Step 1: Register Service (AudioFlinger::instantiate())

- **Service_Process** send "service name" and cookie (point to **new BnAudioFlinger**) to binder driver
- Binder driver new a binder_node and binder_ref,
  binder_node->cookie
  binder_ref->desc

- Binder driver send struct flat_binder_object to user space
  fp->handle = ref->desc;
- ServiceManager add svcinfo to list
  svcinfo->name = "service name"
  svcinfo->ptr = fp->handle

Step 2: Get Service (AudioSystem::get_audio_flinger())

- Client_Process send "service name" to binder driver, binder driver bypass to ServiceManager
- ServiceManager find service element by "service name", get the **service handle**
- Client_Process use **service handle** to new a **Bpbinder**, and type conversion to **BpAudioFlinger.**

Step 3: RPC Call (AudioSystem::setMode(int mode))

- Client_process get **BpAudioFlinger**, and send binder package data to binder driver: handle, RPC func code, parameter
- Binder driver find binder_node by handle, and sent binder_node->cookie to Service Process
- Service Process get cookie and type conversion to **AudioFlinger** (BnAudioFlinger)
  BnAudioFlinger::onTranscat() => call AudioFlinger::setMode() => send reply to binder driver
  binder driver wait up client process
- Client process get result.

Binder App example: HelloWorldService

Source code : https://github.com/mcr/Android-HelloWorldService

Client

1 GetService

```
sm = android::defaultServiceManager()
sp<android::IBinder> binder;

binder = sm->getService("hellowold_name"));
```

2 Get BpHelloWorldClient

```
android::sp<IHelloWorldClient> shw;
shw = android::interface_cast<IHelloWorldClient>(binder);
```

3 RPC Call

```
shw->hellothere("fun");
```

BpHelloWorldClient talk to binder driver

```
void hellothere(const char *str)
    {
        android::Parcel data, reply;
        data.writeInterfaceToken();
        data.writeCString(str);
        remote()->transact(HW_HELLOTHERE, data,
&reply, android::IBinder::FLAG_ONEWAY);
    }
```

Service

1 Register Service and start service:

```
void HelloWorldService::instantiate() {
  android::defaultServiceManager()->addService("", new HelloWorldService());
}

android::ProcessState::self()->startThreadPool();
```

2 BnHelloWorldService

```
HelloWorldService::onTransact(code,)

switch(code) {
    case HW_HELLOTHERE: {
        CHECK_INTERFACE(IHelloWorldService, data, reply);
        const char *str;
        str = data.readCString();
        /* hellothere(str); */
        printf("hello: %s\n", str);
        return android::NO_ERROR;
    } break;
```

Discussion after class

1. For mmap system call, the function mmap for user space :

void* mmap(void *addr, size_t size, int prot, int flags, int fd, long offset)

but in kernel space like binder driver, the mmap function:

int binder_mmap(struct file *filp, **struct vm_area_struct *vma**)

It is too difference, so the second parameter struct vm_area_struct *vma in binder_mmap(), where is come from?

2. In binder, Process A send data to Process B, it need copy data between kernel space and user space, so how many times of copy data does it occur?

3. In binder driver, we use copy_from_user()/copy_to_user() to copy data between kernel space and user space, is it instead of using memcpy()? Why?

4. In binder_mmap() function, it reversed a vmalloc region for remap new pages by get_vm_area(), and the new page also remap to VMA region, why it remap two regions?