一次触摸，Android到底干了啥

原创 2017-11-16 任韬 腾讯WeTest

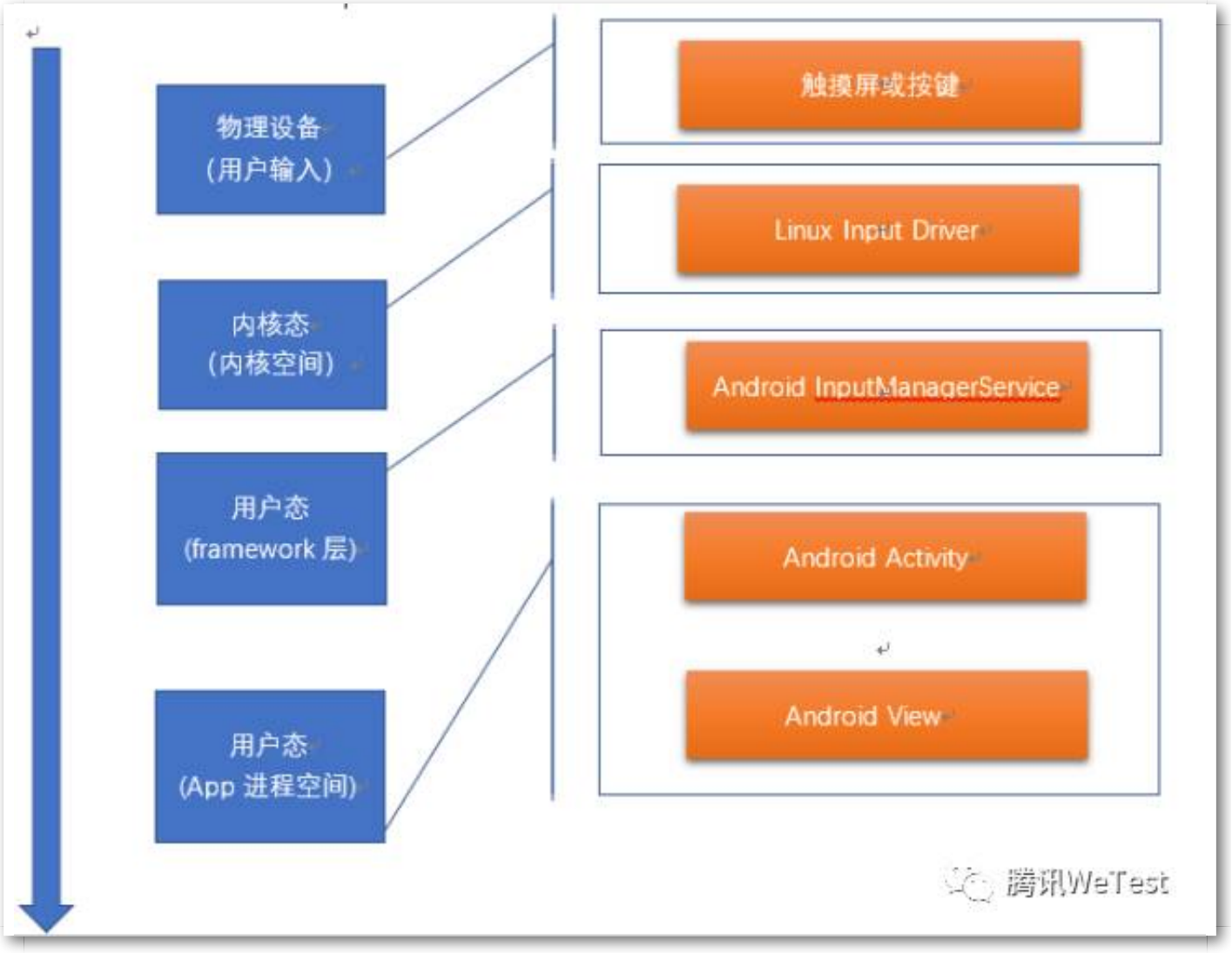
作者

任韬，腾讯移动客户端开发 高级工程师

WeTest 导读

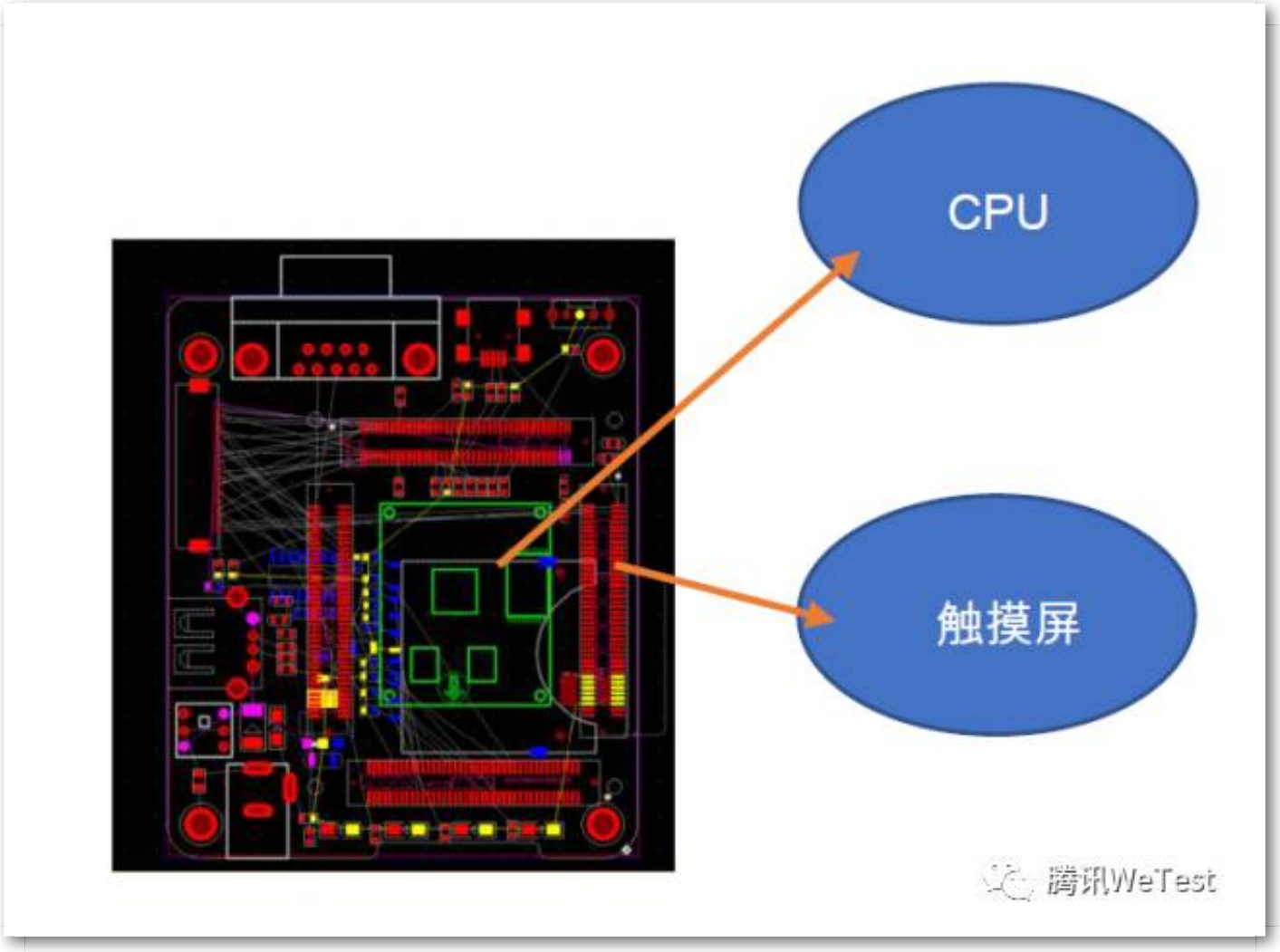
当我们在写带有UI的程序的时候，如果想获取输入事件，仅仅是写一个回调函数，比如(onKeyEvent,onTouchEvent....)，输入事件有可能来自按键的，来自触摸的，也有来自键盘的，其实软键盘也是一种独立的输入事件。那么为什么我能通过回调函数获取这些输入事件呢？系统是如何精确的让程序获得输入事件并去响应的呢？为什么系统只能同一时间有一个界面去获得触摸事件呢？ 下面我们通过Android系统输入子系统的分析来回答这些问题。

一、输入事件的转发流程



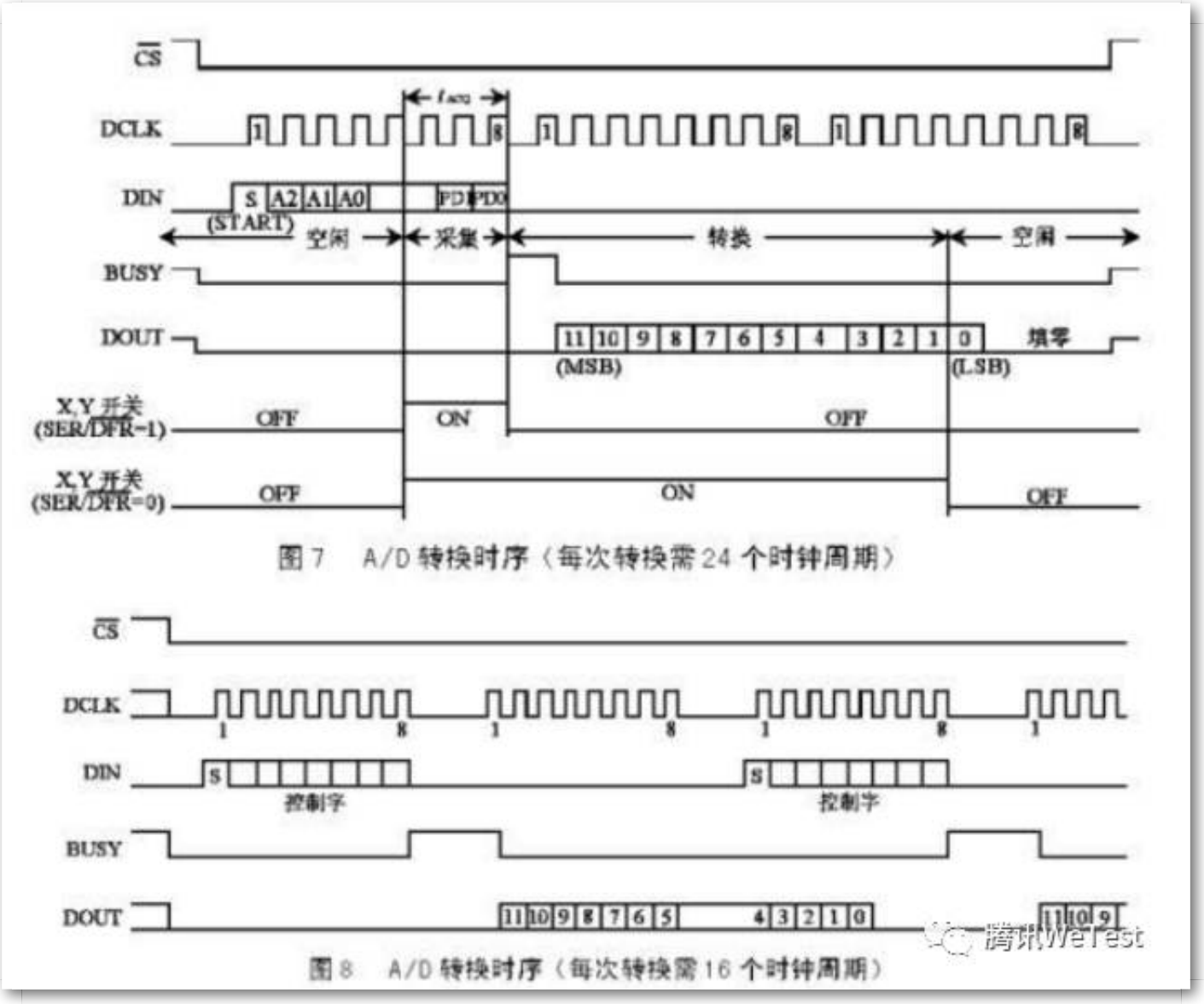
二、物理设备是如何将输入数据发送给内核的

物理设备将数据发送给内核是通过设备驱动传输的，在linux下的/dev/input/目录下有几个设备文件,event0,event1,event2..... 这些设备文件实际上是驱动创建的,他们共用一个主设备号，仅仅是次设备号不同,表示这是一类设备。比如触摸屏对应event0,触摸屏驱动被挂载后，驱动程序会进行初始化，主要是初始化CPU引脚，设置中断处理程序。



很好理解，触摸屏是一个物理设备，但是我们的驱动程序运行在CPU中，这是两个不同的设备，他们在物理上的连接是通过导线将对应的引脚相连接的，只不过导线在PCB板中很小，驱动程序就是初始化CPU中跟触摸屏连接的引脚，但让每个引脚都会对应寄存器，这个在CPU的芯片手册中很详细(DataSheet)。

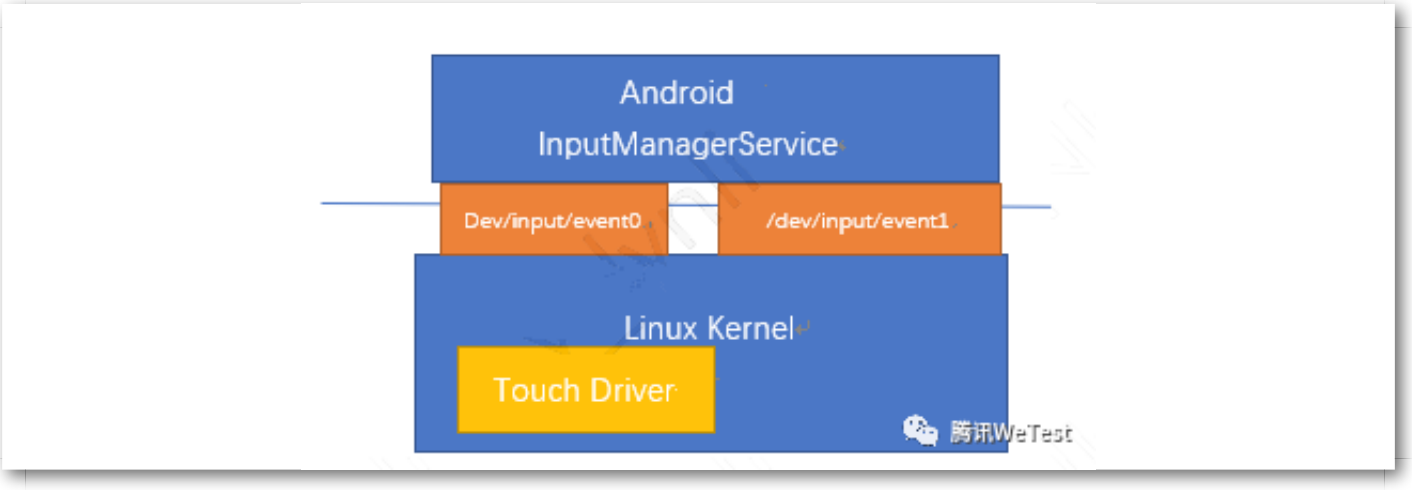
当按下触摸屏的时候触摸屏有个引脚电平变低了，相连的CPU引脚检查到这个连接的引脚电压变低了，那么就会触发中断，这个在触摸驱动中初始化好的，CPU有个中断向量表，这里就到了我们驱动中写好的中断处理函数，中断处理函数中就会读取触摸屏的数据，就是通过相连接的引脚组成的二进制数据比如(01011010)，这个时候我们的内核就拿到的触摸屏的数据。



触摸屏芯片的时序图

三、内核是如何把输入数据发送给用户空间Android framework的

内核拿到触摸屏的数据后，经过平滑处理，滤波，数据还是在内核空间，那么Android怎么拿到触摸数据呢？Android实际上是运行在linux内核上一组进程，这一组进程组合为用户提供UI,应用程序的安装等等服务。



手机开机流程是linux内核先启动，启动完成之后会将Android进程组启动起来，FrameWork属于这个进程组之中。Framewok中有个服务InputManagerService，**我们看Android源码它在哪里实例化的：**

SystemServer.java---------->

startOtherServices()------>

/*构造InputManagerService*/

inputManager = new InputManagerService(context);

/*将inputManager传递给WindowManagerService去

wm=WindowManagerService.main(context, inputManager,

mFactoryTestMode !=FactoryTest.FACTORY_TEST_LOW_LEVE !mFirstBoot, mOnlyCore);

/*给InputManagerService设置回调*/

inputManager.setWindowManagerCallbacks(wm.getInputMonitor());

/* 全初始化好后，SystemServer调用start()函数让InputManager中两个线程开始运行。先看InputReaderThread，它是事件在用户态处理过程的起点*/

inputManager.start();

所以可以看到它在SystemServer进程中实例化并且启动，所以我们首先需要看看InputManagerService的构造函数都做了什么?

构造函数会调用到jni创建NativeInputManager的c++对象, NativeInputManager构造函数中创建

Sp<EventHub> eventHub = new EventHub()

mInputManager = new InputManager(eventhub,this,this);

eventHub对象构造函数做了下面几件事情：

1. 创建epoll对象,之后就可以把各个输入设备的fd添加进来多路等待输入事件

2. 利用inotify机制监听/dev/input目录下的变更,如果有则意味着设备变换,需要处理,输入设备的增减删除操作的监听，将代表inotify的fd添加到epoll中

3. 创建pipe,管道只能用来在具有公共祖先的两个之间通信.读端添加epoll中

InputManager对象构造函数做了下面几件事：

1. 创建InputDispatcher

2. 创建InputReader(eventhub,inputdispatcher)，InputDispatcher继承InputListenerInterface
3. 创建InputReaderThread
4. 创建InputDispatcherThread

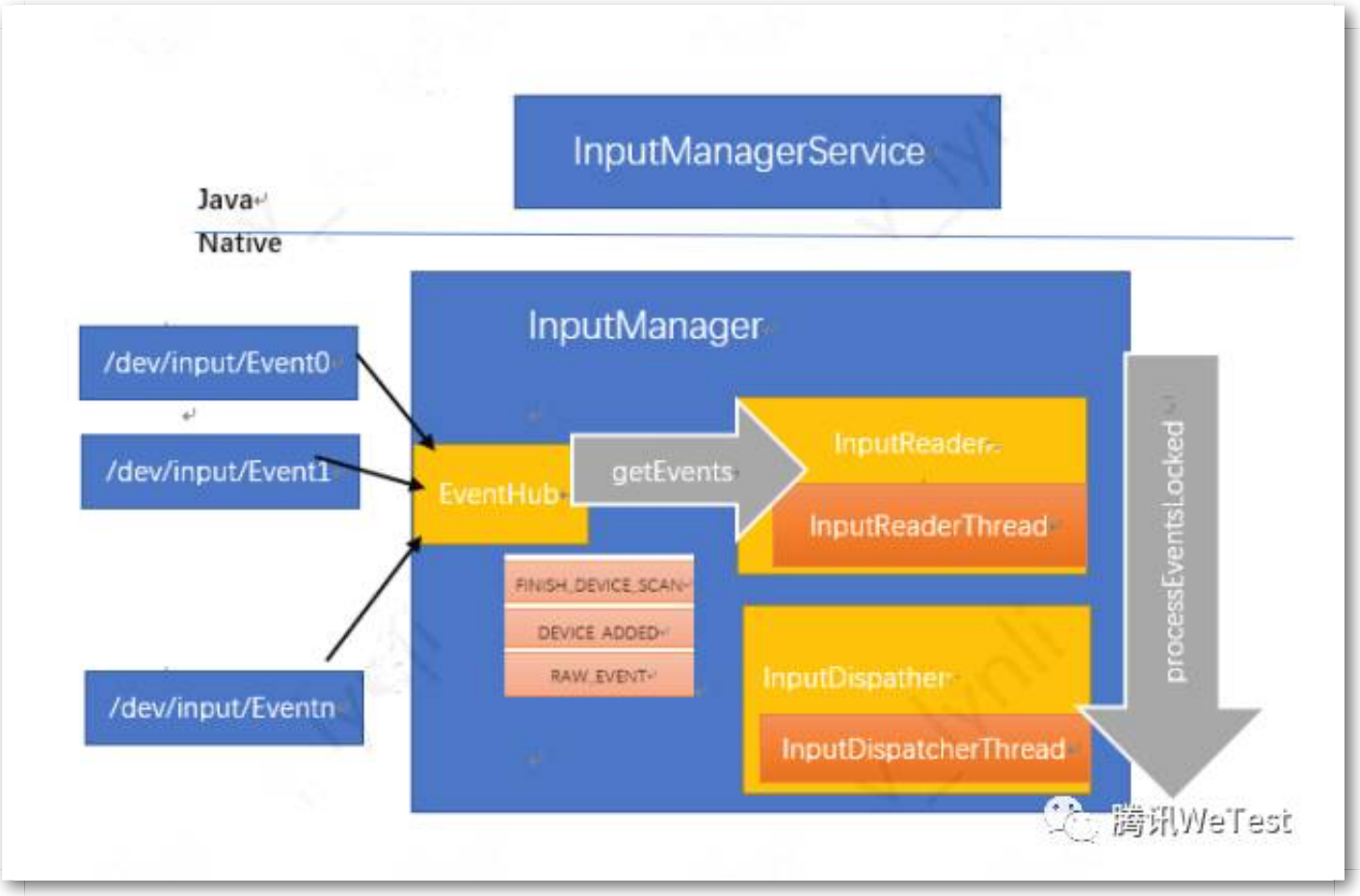
我们还记得最SystemServer.java中最后通过inputManager.start(); 来运行我们的InputManagerService,所以继续看start方法，实际上在native层的inputManager对象中,将上面创建的两个线程InputReaderThread和InputDispatcherThread的start方法中。

对于InputReaderThread的start方法：
1. 调用构造函数中保存的eventHub的getEvents方法获取input事件，在getEvent方法中做的事

1）判断是不是需要打开input设备驱动，如果需要打开设备驱动,扫描/dev/input目录下的设备文件并打开这些设备，同时会判断设备列表中有没有虚拟键盘,没有的话就创建一个device添加进去

2）到下一步中至少系统存在两个输入设备,一个是触摸屏,一个是虚拟键盘,因为上面这次getEvent的调用需要打开设备，所有就将这些动作封装成RawEvent事件，这里两个DEVICE_ADDED事件
+FINISH_DEVICE_SCAN事件,将这些事件返回，不会往下走了

3）如果第二次进入getEvents方法中就会等待读取输入事件，将读取的touch事件发送返回



到这里我们就知道了内核空间的触摸输入数据是如何传递到了用户空间的Android framework中的,实际上就是通过/dev/input目录下，去扫描这个目录，如果有device就打开这个device ,并添加到epoll对象中,多路等待输入事件，在loop中获取数据。

四、Android framework是怎样将输入数据发送给APP进程的

Android framework获取了触摸输入的数据，但是在系统中有那么多进程，那么多进程都在获取输入，它是如何进一步处理，准确的分发事件的呢?

InputReaderThread的start方法中做的第二件事情:
调用processEventsLocked方法处理上面的getEvents方法返回的的RawEvent
1）根据RawEvent的类型不同，调用不同的方法处理,有 ● 普通的touch事件
.● 添加设备的事件
.● 删除设备的事件
.● FiNISHED_DEVICE_SCAN

2）对于touch事件: 调用这个touch事件对应的输入设备(之间创建的InputDevice)的process方法，该方法内部调用内部的InputMapper的process方法,一个输入设备有很多个Mapper,遍历所有的Mapper,并调用process,假定我们是一个支持多点触摸的touch screen，它的mapper是MultiTouchInputMapper,调用它的process方法。

3）MultiTouchInputMapper的process方法内部会这样处理：
首先每次一个touchEvent获取Slot,在没有收到EV_SYN之前对应的Slot都是相同的,然后依次处理x,y,pressure,touch_major，这些值初始化slot的各个变量；

当收到ev.type== EV_SYN并且ev.code = SYN_MT_REPORT那么当前的slot的index加1，给下一次触摸事件去记录，同时sync函数处理这次触摸事件；

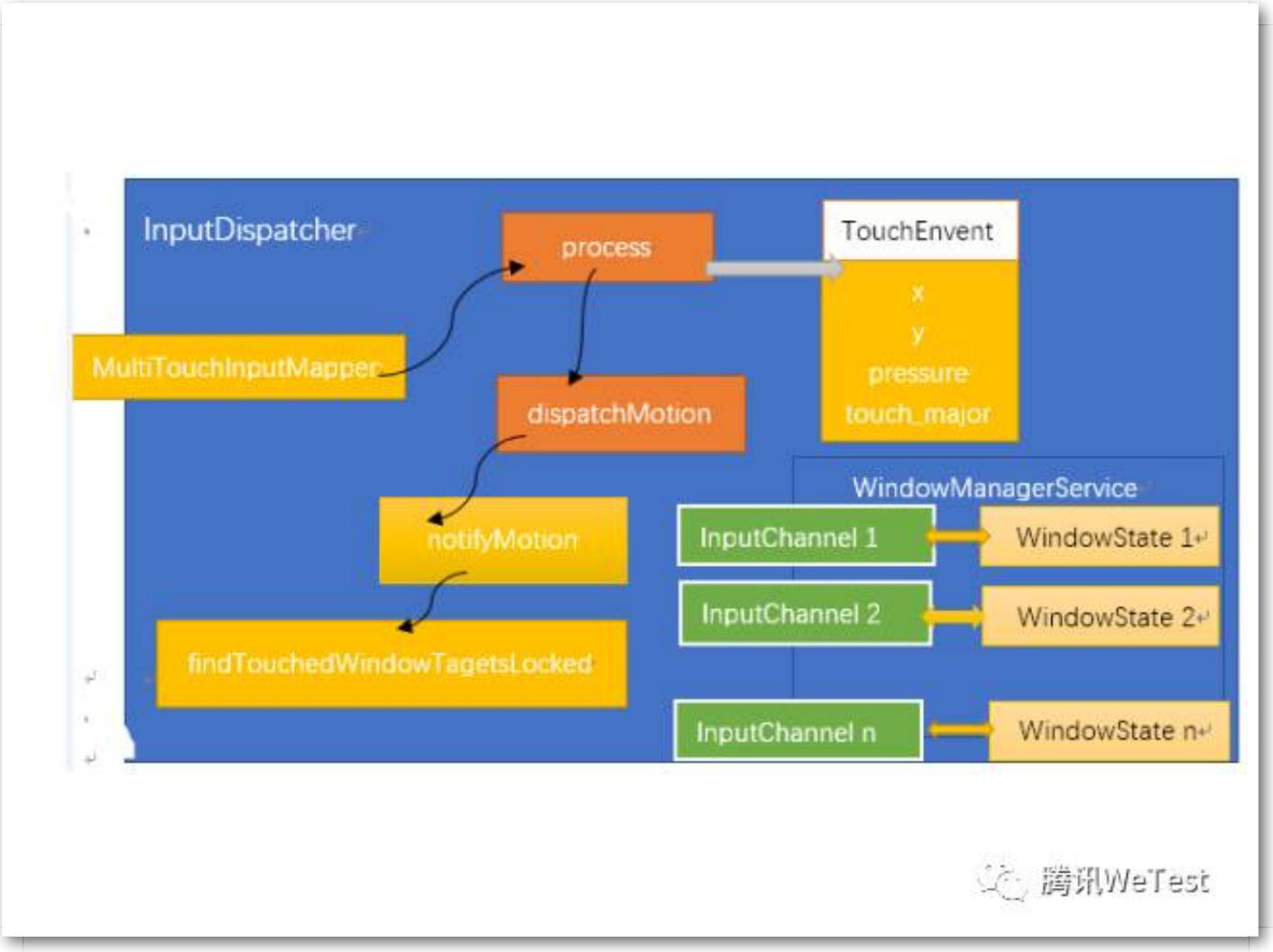
然后CurrentCookedPointerData和LastCookedPointerData进行一些列的操作，up,down还是move事件，然后对应的不同事件，调用dispatchMotion,内部调用InputDispatcher的notifyMotion

4）对于InputDispatcher的notifyMotion：
● 如果InputDispatcher设置了inputFilter,那么首先调用inputFilter来消费这些事件
● 如果没有inputFiler,或者inputFilter对这些事件不感兴趣,那么就会构造一个MotionEntry,添加到mInboundQueue,并唤醒InputDispatcher线程处理

5）对于InputDispatcher的线程处理循环：
● 优化app切换延迟,当切换超时，则抢占分发，丢弃其他所有即将要处理的事件；
● 分发事件：
首先调用findTouchedWindowTagetsLocked寻找有focus的window窗口， 并把这些创建保存在inputTargets数组中；

之前注册的monitor的InputChannel这里也会添加到inputTargets数组中；

然后向inputTargets数组一一分发事件。



到这里我们就知道了是如何找到这个APP进程的了。

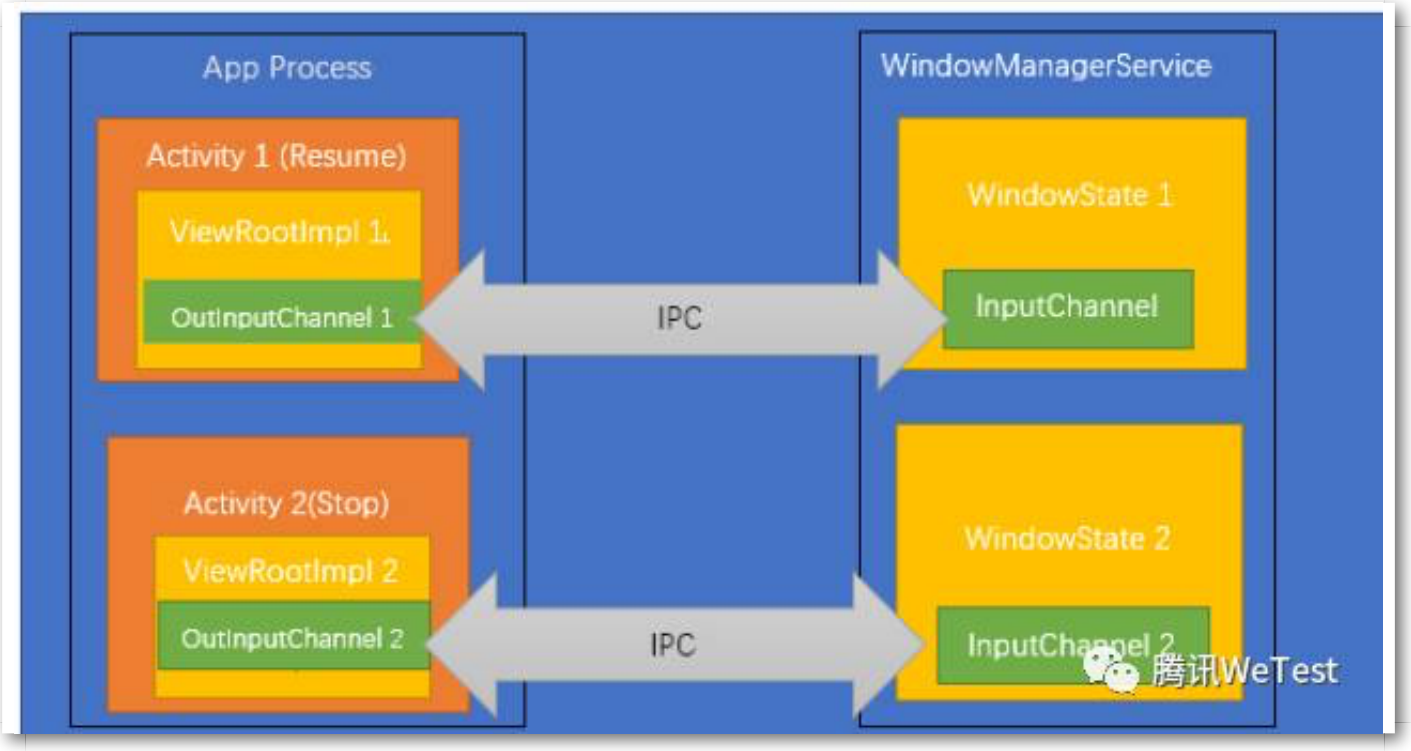
五、APP进程是如何将输入数据发送给它对应的Activity的

Activity是一个进程的基本组件，可以认为它代表了一个界面，是一堆View的集合，每次Activity启动的时候都做了什么呢?

1、实际上取决于它背后的ViewRootImpl做了什么，在ViewRootImpl.java中的setView方法中,实例化InputChannel,当然会判断当前的窗口能不能接受输入事件,接着在调用到session.java中的addToDisplay方法传递给WindowManagerService,实际上是调用WindowManagerService的addWindow方法,在WindowManagerService中会创建一对InputChannel[],然后InputChannel[1]转移到这个inputChannel， 然后setView方法继续创建一个WindowInputEventReceiver对象,然后将上面创建好的InputChannel

2、WindowManagerService中的addWindow方法:

```
InputChannel[] inputChannels = InputChannel.openInputChannelPair(name)
/*Channel[0]保存在server端*/
win.setInputChannel[inputChannels[0]]
/* Channel[1]返回给ViewRootImpl端*/
inputChannels[1].transferTo(outInputChannel)
/*注册到inputManagerService中*/
mInputManager.registerInputChannel(win.mInputChannel,win.mInputWindowHandle)
```



到这里我们就能明白如何将时间分发给对应的Activity了，其实是给了它背后的ViewRootImpl。

六、Activity又是如何将输入数据发送给具体的View的

最后一步就是将事件分发到Activity中具体的View了,从ViewRootImpl中将事件分发给具体的View,很好理解，因为触摸的范围在到这里是知道的，每个View的位置以及状态到这里也是知道的，因为View要正确渲染的话，Android图形框架会搞定这一切，测量每个View的大小，确定每个View的位置，ViewRootImpl会一层一层将数据分发到自己每个View中，但是每个View自己知道这个触摸事件是不是作用在自己身上的，如果不是就丢弃了，往下面分发。

总结

触摸事件的分发流程看起来挺复杂，但是Android实现的还是很优雅的，我们去分析它的流程，对于我们想实现一些比较的酷的功能是有帮助的。当然对于我们调试代码也会有帮助，当发现触摸后，系统无响应，将上

面的流程分解，总是能分析出原因。

腾讯WeTest提供上千台真实手机，随时随地进行测试，保障应用/手游品质。节省百万硬件费用，加速敏捷研发流程。

同时腾讯WeTest兼容性测试团队积累了10年的手游测试经验，旨在通过制定针对性的测试方案，精准选取目标机型，执行专业、完整的测试用例，来提前发现游戏版本的兼容性问题，针对性地做出修正和优化，来保障手游产品的质量。目前该团队已经支持所有腾讯在研和运营的手游项目。

欢迎点击左下角**“阅读原文”**使用专家兼容测试服务。WeTest兼容性测试团队期待与您交流！You Create，We Test！

如果对使用当中有任何疑问，欢迎联系腾讯WeTest企业qq：800024531

点击左下角“阅读原文”体验专家兼容测试吧

⭐如果你喜欢这篇文章，欢迎分享到朋友圈⭐

关于腾讯WeTest

腾讯WeTest是腾讯游戏官方推出的一站式游戏测试平台，用十年腾讯游戏测试经验帮助广大开发者对游戏开发全生命周期进行质量保障。
腾讯WeTest提供：兼容适配测试；云端真机调试；安全测试；耗电量测试；服务器压力测试；舆情监控等服务。

阅读原文