**一、為什麼要有ANR**

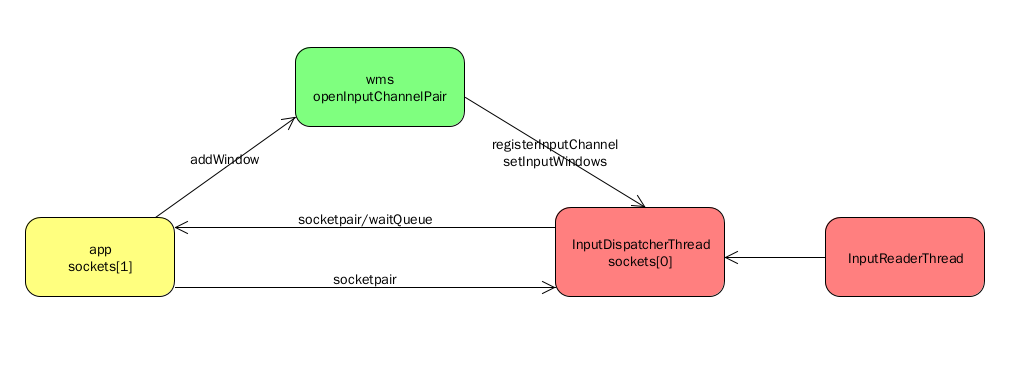
因為android中界面的繪製(vsync 信號的處理)是在主線程中完成的，假如主線程處理某個消息花費時間很長，繪製的工作就會被delay，這樣導致界面卡頓，用戶體驗會很差。

**二、ANR 產生的詳細過程：**

**Fg ANR:**

**Reason: Input dispatching timed out**

1. GUI 中input的作用和簡要的講解：



**線程：**

InputReaderThread、InputDispatcherThread

**InputDispatcher 中主要對象：**

sp<InputWindowHandle> mFocusedWindowHandle

KeyedVector<int, sp<Connection> > mConnectionsByFd

const nsecs\_t DEFAULT\_INPUT\_DISPATCHING\_TIMEOUT = 8000 \* 1000000LL;

nsecs\_t mInputTargetWaitStartTime;

nsecs\_t mInputTargetWaitTimeoutTime;

**app和InputDispatcher之間通信管道的創建：**

InputChannel 的創建：

String name = win.makeInputChannelName();

InputChannel[] inputChannels = InputChannel.openInputChannelPair(name);

win.setInputChannel(inputChannels[0]);

inputChannels[1].transferTo(outInputChannel);

mInputManager.registerInputChannel(win.mInputChannel, win.mInputWindowHandle);

int sockets[2];

socketpair(AF\_UNIX, SOCK\_SEQPACKET, 0, sockets)

InputDispatcher 端創建接收的監聽：

mLooper->addFd(fd, 0, ALOOPER\_EVENT\_INPUT, handleReceiveCallback, this);

**app端接收監聽的創建：**

mInputEventReceiver = new WindowInputEventReceiver(mInputChannel,

Looper.myLooper());

NativeInputEventReceiver：

getLooper()->addFd

**正常的input事件的流程：**

InputDispatcher 分發input事件通過保存input事件到waitQueue。

通過sockets[0] 將input事件發送到app端的sockets[1] 。

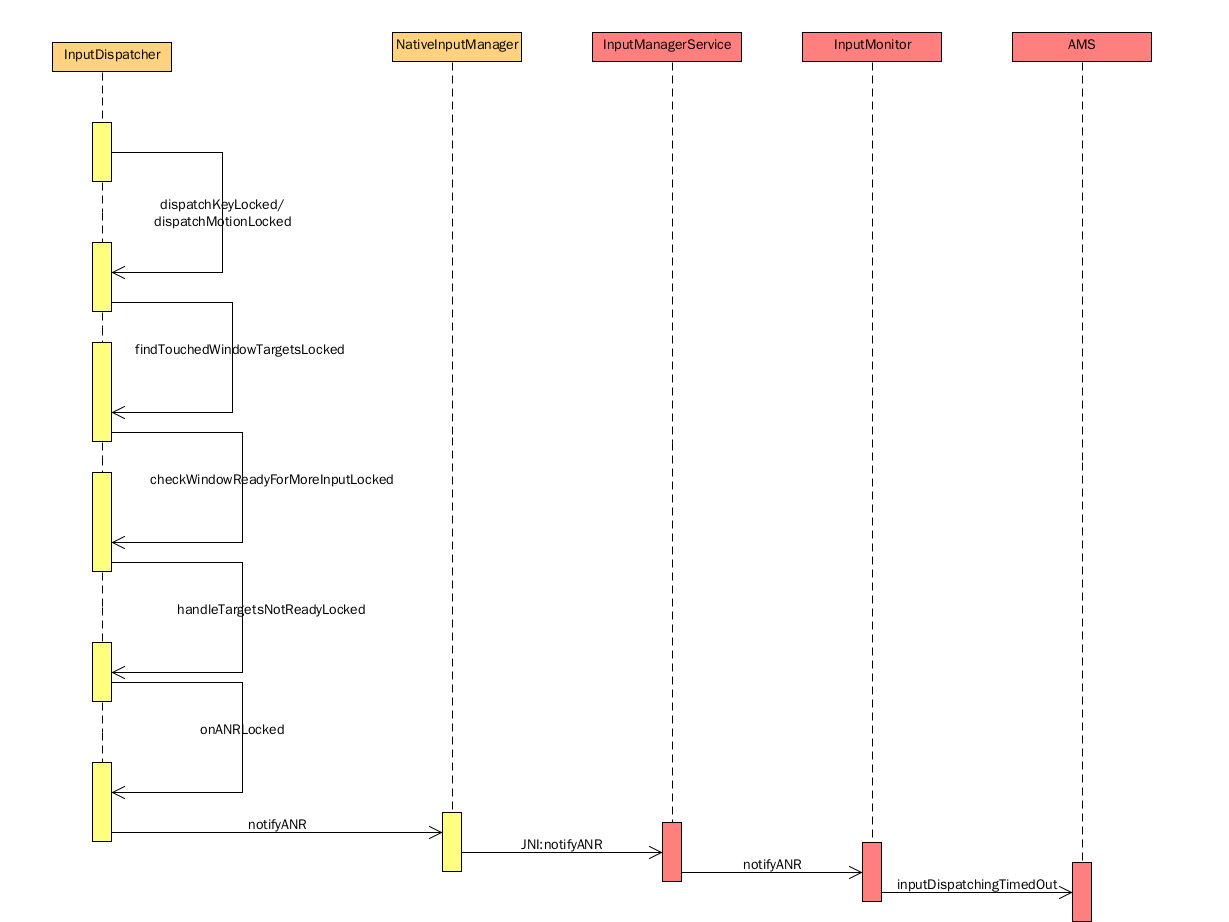
App端接收input事件並且將事件分發到相應的控件進行處理。

處理完成後調用InputEventReceiver .finishInputEvent函數將input事件處理完成的消息通過sockets[1] 發送到InputDispatcher的sockets[0] 。

在InputDispatcher線程中通過sockets[0]接收完成事件。

handleReceiveCallback處理該input完成事件，從waitQueue中移除該input事件，這樣一個input事件的生命流程走完。

**有ANR時input的流程：**

****

InputDispatcher 在分發input事件前函數會檢查connection的waitQueue（checkWindowReadyForMoreInputLocked），當等待隊列中有app沒有處理完的input事件時調用handleTargetsNotReadyLocked函數檢查第一次沒有處理完的input事件分發時設置的mInputTargetWaitTimeoutTime 和當前時間進行比較，當當前時間大於mInputTargetWaitTimeoutTime值說明app主線程發生了ANR，進行anr的處理流程。

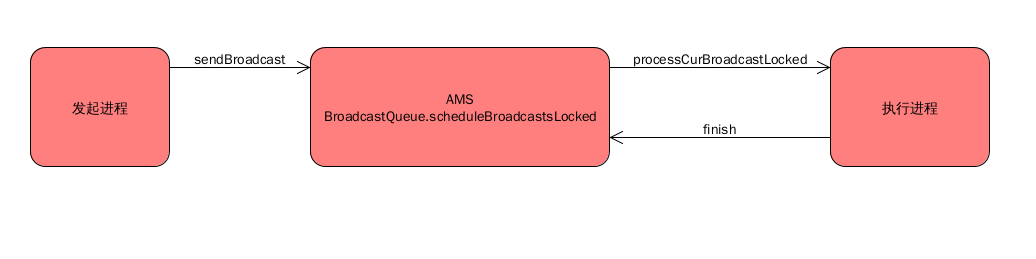
**Bg ANR:**

Reason: broadcastTimeout

Reason: serviceTimeout

<http://blog.csdn.net/siobhan/article/details/8969404>

broadcast 引起的ANR：

****

setBroadcastTimeoutLocked：sendMessageAtTime

cancelBroadcastTimeoutLocked：removeMessages

anr時：執行進程中的onReceive函數執行時間過長導致AMS沒有cancle 這個recever的BROADCAST\_TIMEOUT\_MSG 消息，同時AMS執行broadcastTimeoutLocked 調用appNotResponding函數執行ANR流程。

**針對發生ANR時trace堆棧不是耗時消息處理時主線程堆棧的問題：**

MTK 的MessageLogger、dumpMessageQueue

Looper 保存Message的處理信息。