# 一、输入系统方案整体架构

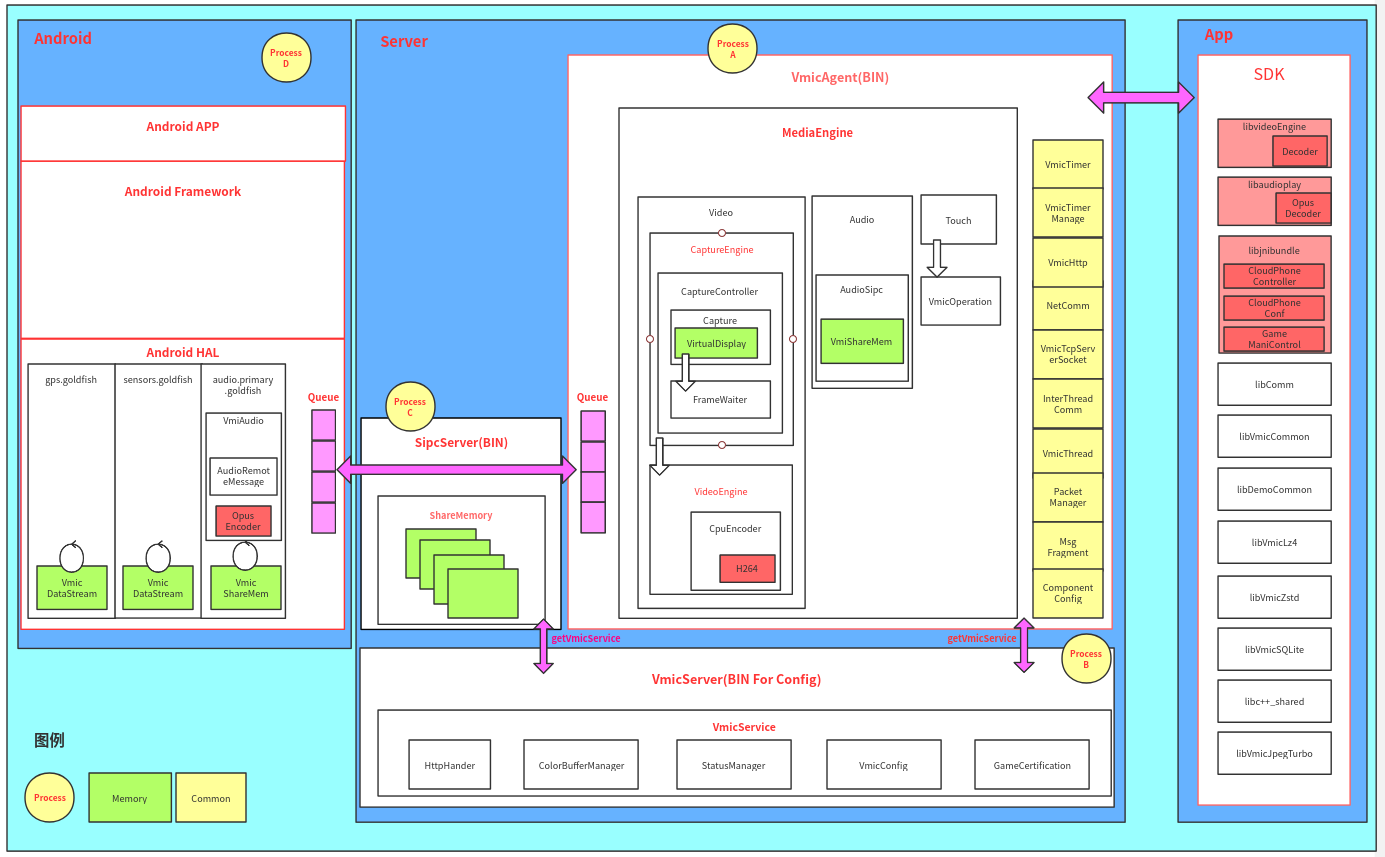
## 1.需求背景

借助5G高带宽、低延时的特性，结合云端大容量、高性能的运算及存储资源，终端的运算和存储能力逐渐迁移到云端，本地和云端只需进行交互，便可实现远程操控云端系统，畅想云端无限存储、计算资源的效果。

输入系统是云侧和端侧协同的关键技术，同时可为云手机、云游戏、云办公等场景赋能，提供更流程、更安全、体验更佳的5G云服务。

## 2.整体架构

服务端（安卓侧+协议侧）+客户端整体架构图，概述方案架构。



如上图所示，本方案由客户端和服务端两个整体部分，而这两个部分由分别由四个模块组成，分别为：客户端Android侧、客户端协议侧、服务端协议侧、服务端Android侧。

**客户端Android侧：**此部分与用户交互最为紧密，它的目的是采集客户端所有有关输入的数据供服务端使用。

**客户端协议侧：**这个部分整个方案的核心之一，它是集数据封装，传送，数据完整性校验等功能以达到Android数据到服务端的完美迁移。

**服务端协议侧：**这个部分是另一个核心。与客户端协议侧一样，具有相同类似的功能。它的作用是把来自客户端协议侧的数据按照Android的方式更好更优的为服务端的展现提供数据的预处理。

**服务端Android侧：**将客户端传递过来的输入数据，注入到Android输入系统中，借此来使得整个云手机的组件或者功能更加完整。

# 二、技术调研

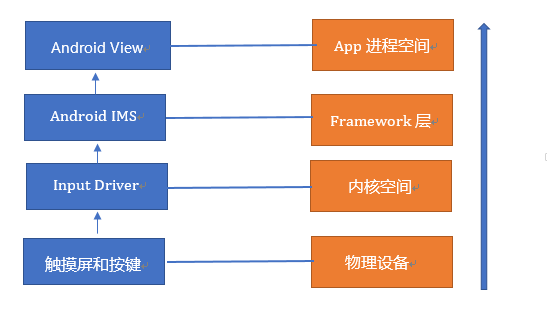
Android输入子系统支持各种不同类型的设备，包括键盘、操纵杆、游戏手柄、鼠标和触摸屏等，其中触摸屏和键盘是Android最普遍使用的输入设备。

Android设备可以同时连接多个输入设备，当输入设备可用时，Linux内核会在/dev/input/下创建对应的名为event0~n或其他名称的设备节点。而当输入设备不可用时，则会将对应的节点删除。

当用户操作输入设备时，会产生一个中断，Linux内核接收到相应的硬件中断，然后将中断加工成原始的输入事件数据并写入其对应的设备节点中，在用户空间可以通过read()函数将事件数据读出。

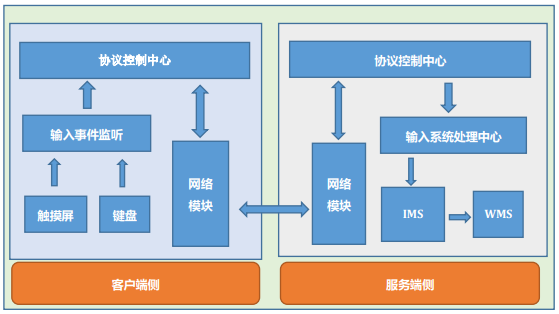
Android输入系统的工作原理概括来说，就是监控/dev/input/下的所有设备节点，当某个节点有数据可读时，将数据读出并进行一系列的翻译加工，然后在所有的窗口中寻找合适的事件接收者，并派发给它。

下图为输入事件的流程：



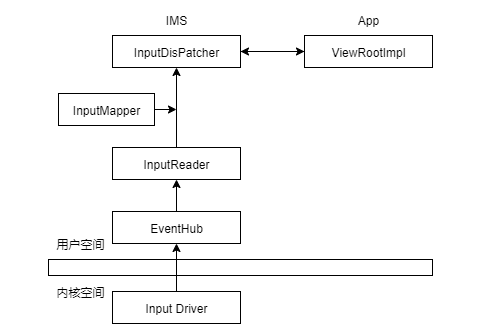
# 三、客户端实现方案

## 1.客户端整体方案（架构）



## 2.输入事件监听

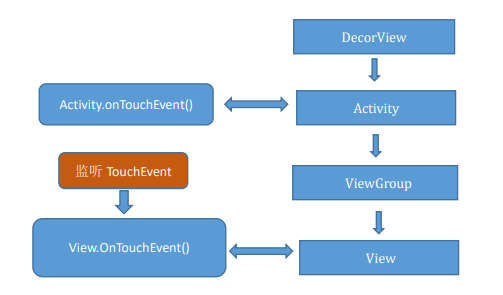
如下为Android客户端输入事件从Driver开始的处理流程：



最初的输入事件为内核生成的原始事件，而最终交付给窗口的则是MotionEvent或KeyEvent对象。客户端的监听是在APP的View中进行的，下面分别介绍触摸屏的输入事件（MotionEvent）和键盘的输入事件（KeyEvent）的监听。

### 2.1.触摸屏事件监听

事件由Activity的dispatchTouchEvent()开始，将事件传递给当前Activity的根ViewGroup：mDecorView，事件开始自上而下进行传递，直至被消费。View的onTouchEvent默认返回true。也就是如果事件流到View的onTouchEvent中，默认会消费这个事件。



触摸屏输入事件的监听是在客户端Android侧进行的，会在Android的对应Activity里面获取相对应的MotionEvent数据，MotionEvent数据主要包括以下几部分：

1、EventType：事件类型。

2、Pointer：一个pointer就代表一个触摸点，每个pointer有自己的EventType，也有自己的坐标值。

3、x:x坐标值。

4：y:y坐标值。

5、PointerId：PointerId用来区分是哪个触摸点。从按下、移动到离开屏幕，每个触摸点都会拥有一个固定PointerId。

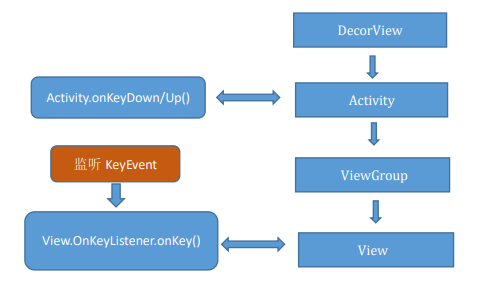
6、PointerIndex：每个触摸点从按下、移动到离开屏幕，每个触摸点在每一个事件的Index可能是不固定的，因为受到其它触摸点变化的影响。

### 2.2.键盘事件监听

在App中，键盘事件主要用于对键盘事件的监听，根据用户输入内容对键盘事件进行跟踪，键盘事件使用View.OnKeyListener接口进行事件处理。

事件由Activity的dispatchKeyEvent()开始，将事件传递给当前Activity的根ViewGroup：mDecorView，事件开始自上而下进行传递，ViewGroup递归寻找当前焦点的子View，将事件传给焦点子View的dispatchKeyEvent()分发。

KeyEvent事件的处理只有两个地方，一个是Activity，另一个则是具体的 View.ViewGroup只负责分发，不会消耗事件。同TouchEvent一样，返回true表示事件已消耗掉，返回false则表示事件还在。

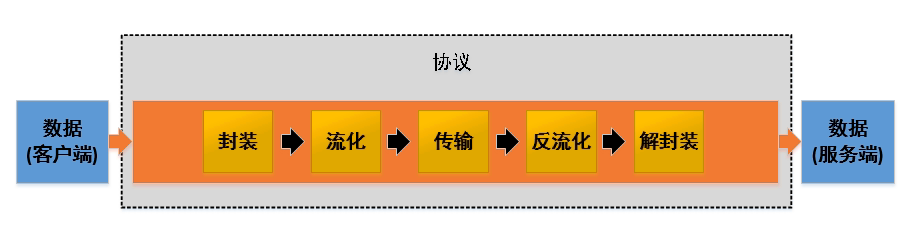


键盘输入事件的监听OnKeyListener，回调OnKeyListener.onKey()方法来监听，在此处获取相对应的KeyEvent数据，KeyEvent数据主要包括以下两部分：

1、event.getAction():对应Action的ACTION\_DOWN,ACTION\_UP。

2、event.getScanCode()：kernel中的按键驱动通过input子系统上报的键值叫做扫描码(ScanCode)。

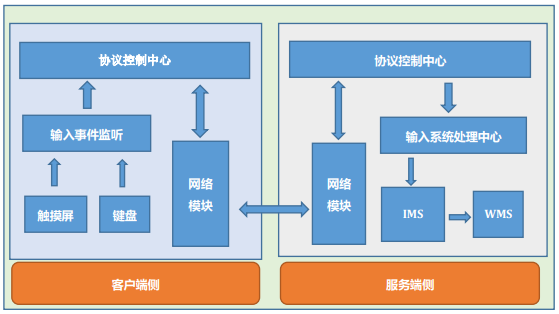
# 四、服务端客户端协议



如上图，整个架构的协议模块就是用来保证客户端产生的数据能够顺利并且‘毫发无损’的抵达服务端，并传送到事件的初始注入模块。为了提升整体的传输效率，协议端会在接收数据和发送数据的地方做相关的封装和解析以及流化和反流化，既省去了其他模块二次解析的时间，也做到了网络层传输的高效性。

# 五、服务端实现方案

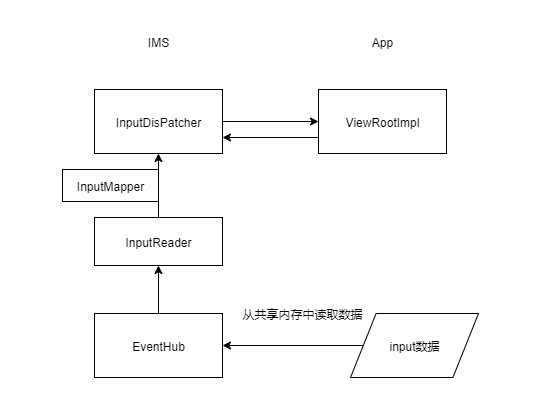
## 1. 服务端整体方案（架构）



## 2.服务端-Android侧

### 2.1 实现概述

云手机需要读取从客户端获取的数据，并进行处理分发，Android侧的处理过程大致如下：



简单来说，服务端协议侧将输入事件的数据不断写入共享内存中，InputReader通过EventHub::getEvents将其读出并加工成Android输入事件， InputDispatcher根据WMS提供的窗口信息将输入事件分发给合适的窗口。窗口的ViewRootImpl对象再将其分给合适的控件。控件则对输入事件作出响应，更新自己的画面、执行特定的动作等。

下面介绍一下输入事件的处理流程以及输入系统中最基本的参与者。

1、IMS：输入管理服务，Android系统服务，它分为Java层和Native层两部分。Java层负责与WMS的通信。而Native层则是InputReader和InputDispatcher两个输入系统关键组件的运行容器。

2、WMS：是窗口管理服务，核心维护了一个有序的窗口堆栈，对InputDispatcher来说是非常重要的一部分。

3、EventHub：输入系统组件，向使用者提供抽取设备事件与原始输入事件的功能，其主要功能在getEvents()中完成。

4、InputReader：输入管理服务的最重要的组件之一。运行在一个独立的线程中，负责管理输入设备的列表与配置，以及进行输入事件的加工处理。

5、InputDispatcher：也是输入管理服务中重要的组件之一，收到来自InputReader的输入事件后，需找合适的窗口，分发事件。

6、ViewRootImpl：将窗口所接收到的输入事件沿着控件树将事件派发给合适的控件。