# 一、输入系统方案整体架构

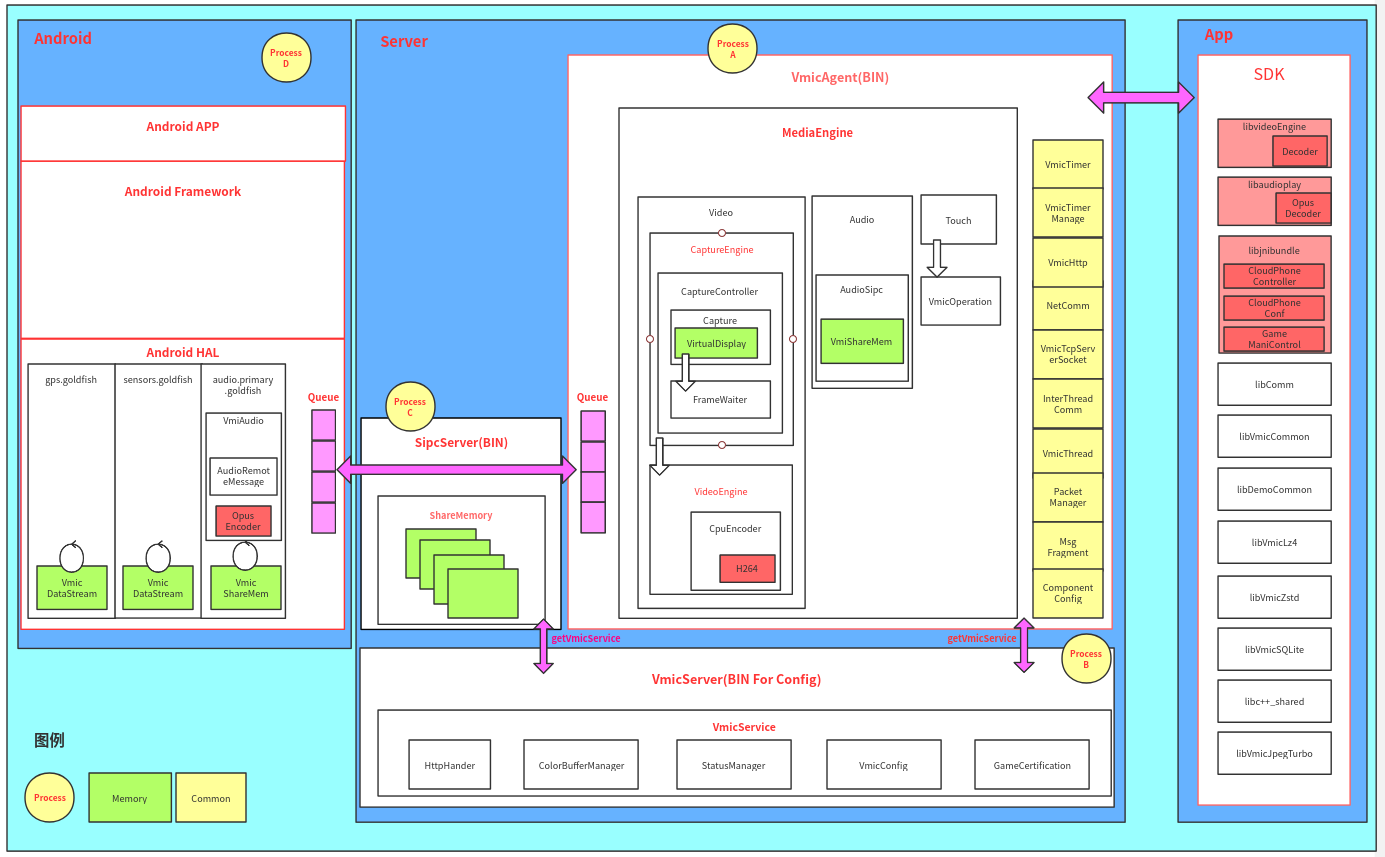
## 1．需求背景

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。

Sensor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小模块组成，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Sensor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Sensor数据，注入到Android Sensor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

大多数Android设备都有内置传感器，这些传感器能够提供高度精确的原始数据，非常适合用来监测设备的三维移动或定位，或监测设备周围环境的变化。

Android平台支持三大类传感器：

**动态传感器：**

这类传感器测量三个轴向上的加速力和旋转力。这个类别中包含加速度计、重力传感器、陀螺仪和旋转矢量传感器。

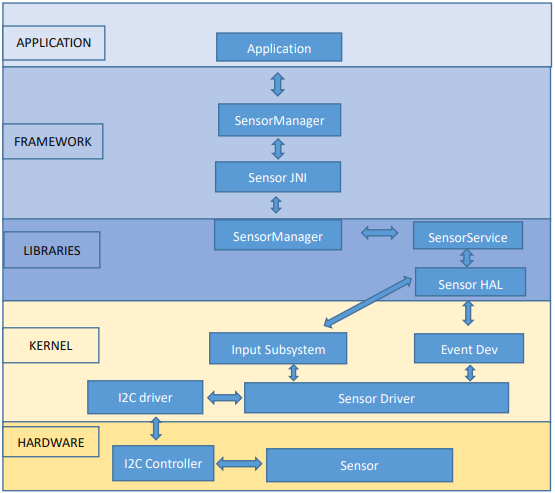
**环境传感器：**

这类传感器测量各种环境参数，如环境气温、气压、照度和湿度。这个类别中包含气压计、光度计和温度计。

**位置传感器：**

这类传感器测量设备的物理位置。这个类别中包含屏幕方向传感器和磁力计。

## Sensor子系统架构



**Framework：**

Sensor应用程序通过Sensor应用框架来获取sensor数据，应用框架层的SensorManager通过JNI与C++层进行通信。

**Sensor-Libraries**

Sensor中间层主要由SensorManager、Sensorservice和Sensor硬件抽象层组成。

**Input-Subsystem**

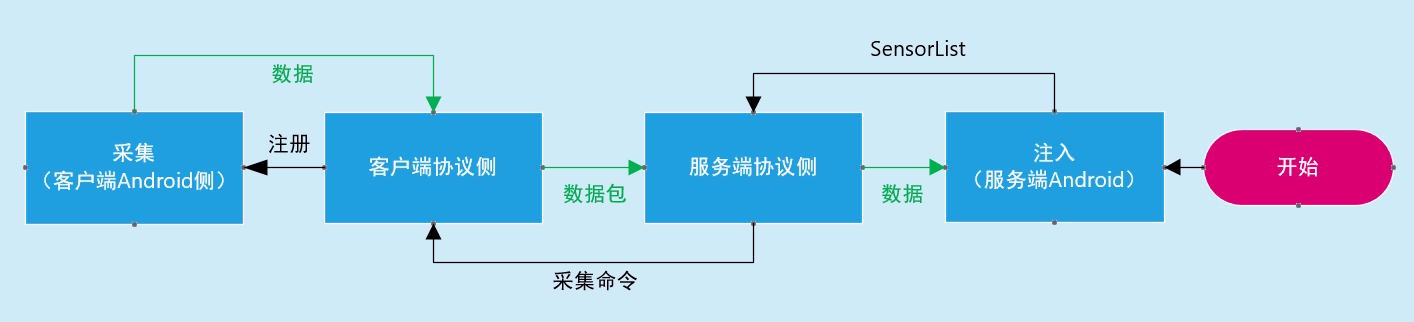
通用的Linux输入框架专为与键盘、鼠标和触摸屏等输入设备而设计，并定义了一套标准事件集合。Sensor输入子系统采用采用了通用的Linux输入框架，它通过/sys/class/input节点和用户空间进行交互。

**EventDev**

EventDev提供了一种访问/dev/input/eventX输入设备事件的通用方法。

# 三、客户端实现方案

## 1．客户端整体方案（架构）

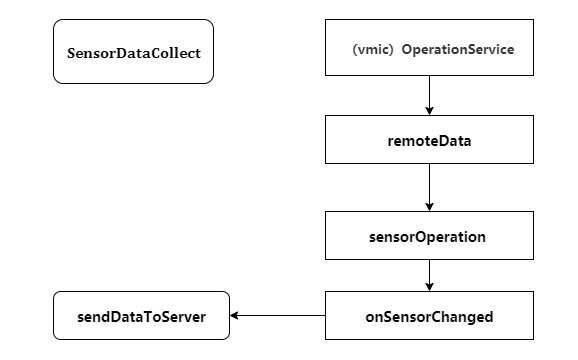


## 2．客户端实现

当应用使用到某个Sensor时的时候，系统会调用到SensorDevice中的SensorDevice::activate，该函数主要用于打开和关闭Sensor。原生的做法是通过调用HAL层的activate函数将打开和关闭相关的操作交由sensors\_qemu.c中的sensor\_device\_activate函数来处理。

云手机的实现流程是：当Sensor的activate操作通过OperationService，由协议传到Client后，交由SDK进行解析。SDK中Sensor的相关实现在SensorDataCollect中，当打开Sensor的命令通过OperationService发送到客户端，客户端会根据服务端的需求来进行收集相对应的Sensor数据。

**客户端实现的流程：**



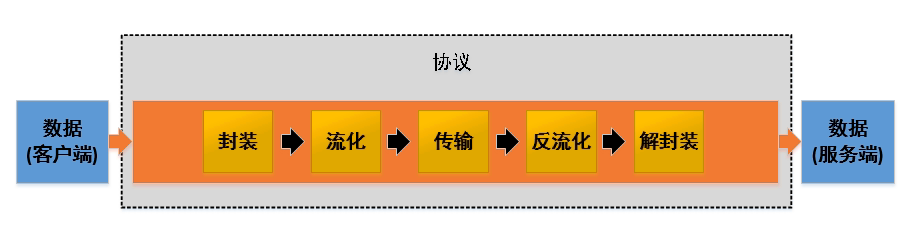
1、SensorDataCollect中的remoteData方法负责接收协议Server端发来的数据，在该方法中对String类型的数据进行了解析。

2、通过sensorOperation选择操作的方法，如activate相关的操作会交由setPower来处理，setPower方法负责相关Sensor类型的打开和关闭（通过调用updateRegister方法来实现）。

3、当某种类型的Sensor被register后，onSensorChanged方法中将会开始监听相关类型的Sensor数据。

4、对Sensor数据进行了简单的拼接处理，主要是为了与服务端的sensor\_qemu.c中的Sensor数据格式一致，然后通过sendDataToServer方法将数据交由协议发送给云端。

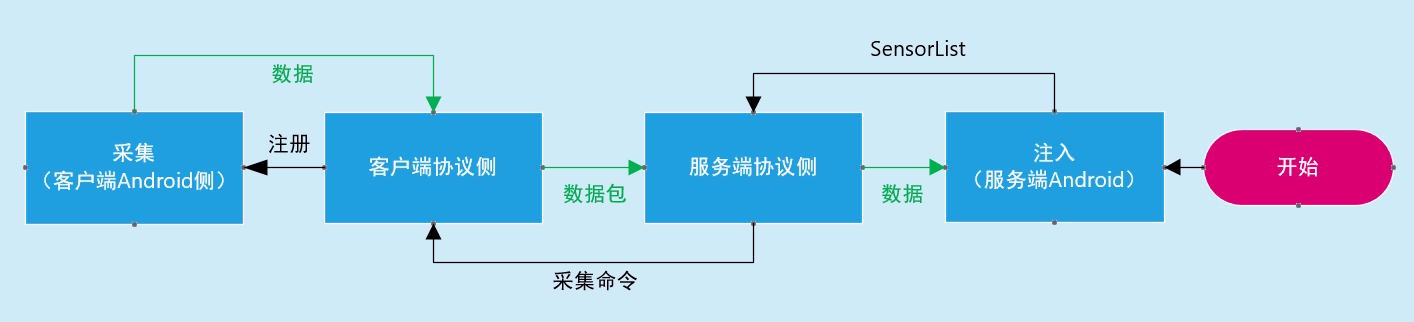
# 四、服务端客户端协议



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．服务端实现

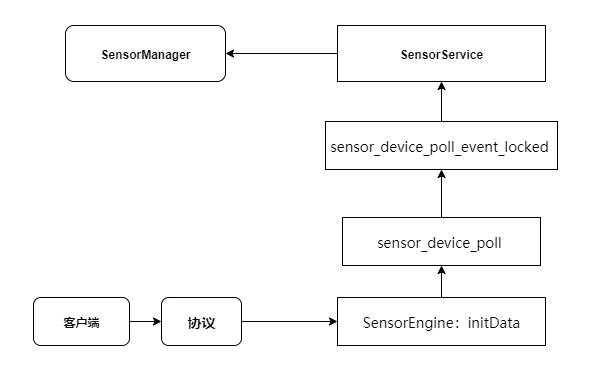
当应用使用某个Sensor时，会调用到SensorDevice.cpp中的SensorDevice::activate，以打开Sensor为例，服务端通过OperationService将对应的命令发送给客户端，从客户端获取数据并返回给服务端，最终返回给应用使用。

整个流程服务端需要实现两个部分 ：

1、注册Sensor时，将对应的命令发送给客户端，主要实现是通过协议来完成的，具体的实现由OperationService的sendOperationMsg来完成。

2、通过协议从客户端接收获取到的Sensor数据，并传给应用的过程。本节主要介绍这一部分。

整个流程如下图所示：



1、当协议的Server端拿到Sensor数据后，会交由SensorEngine进行处理；SensorEngine::inData函数负责处理Client端传输过来的Sensor数据。主要是将数据交由SensorEngine::writeData存储到共享内存中。

2、在Anbox机制中Android有关Sensor的操作通过HAL后最后会交由sensors\_qemu.c的dev->device.poll处理。

3、从对应关系中可以知道SensorDevice中对于数据的poll操作交由sensor\_device\_poll函数，sensor\_device\_poll函数调用sensor\_device\_poll\_event\_locked函数。

4、通过sensors\_read\_sharemem函数获取Sensor数据，然后根据Sensor数据类型进行解析，可以看到Client传递过来的Sensor数据格式为：sensor\_name:param1:param2:param3。当对Sensor数据解析完成后，将会有一个sync指令，接受到sync后将会跳转到out，最终结束sensor\_device\_poll\_event\_locked函数操作，返回数据。

5、SensorService在onFirstRef时创建了一个Looper，该Looper的执行线程会调用poll接口，并阻塞在sensor的数据管道，当sensor有数据返回时，SensorService会通过SensorEventQueue 发送到上层，并最终分发到各个 listener。