知识点1【心跳包的概述】（了解）

1、举个例子

比如说，客户端程序断线了，服务端的TCP连接不会检测到断线，而是一直处于连接状态。这就带来了很大的麻烦，明明客户端已经断了，服务端还维护着客户端的连接，照常执行着该玩家的游戏逻辑……

2、心跳包的意义

心跳包就是用来及时检测是否断线的一种机制，通过每间隔一定时间发送心跳数据，来检测对方是否连接。是属于应用程序协议的一部分

知识点2【心跳包的实现方式】（了解）

1、SO_KEEPALIVE　全局方式设置

操作系统的 TCP/IP 协议栈其实提供了这个的功能，即 keepalive 选项。在 Linux 操作系统中，我们可以通过代码启用一个 socket 的心跳检测（即每隔一定时间间隔发送一个心跳检测包给对端），代码如下

```
//on 是 1 表示打开 keepalive 选项，为 0 表示关闭，0 是默认值
int on = 1;
setsockopt(fd, SOL_SOCKET, SO_KEEPALIVE, &on, sizeof(on));
```

但是，即使开启了这个选项，这个选项默认发送心跳检测数据包的时间间隔是 7200 秒（2 小时），这时间间隔实在是太长了，一定也不使用。

问题：

由于 keepalive 选项需要为每个连接中的 socket 开启，这不一定是必须的，可能会产生大量无意义的带宽浪费，且 keepalive 选项不能与应用层很好地交互，因此一般实际的服务开发中，还是建议读者在应用层设计自己的心跳包机制。那么如何设计呢？

2、用户自己在应用层设计自己的心跳包

方式一：启动定时器 应用层 发送自定义的心跳包

心跳包其实就是一个预先规定好格式的数据包，在程序中启动一个定时器，定时发送即可，这是最简单的实现思路。但是，如果通信的两端有频繁的数据来往，此时到了下一个发心跳包的时间点了，此时发送一个心跳包。这其实是一个流量的浪费，既然通信双方不断有正常的业务数据包来往，这些数据包本身就可以起到保活作用，为什么还要浪费流量去发送这些心跳包呢？

方式二：纪录上一次包的时间（推荐）

设置一个上次包时间，每次收数据和发数据时，都更新一下这个包时间，而心跳检测计时器每次检测时，将这个包时间与当前系统时间做一个对比，如果时间间隔大于允许的最大时间间隔（实际开发中根据需求设置成 15 ~ 45 秒不等），则发送一次心跳包。总而言之，就是在与对端之间，没有数据来往达到一定时间间隔时才发送一次心跳包。