软件复用讨论课

> 场景： 在低带宽/不稳定网络下，保持用户在登陆后始终在线和保证消息准确收发

客观分析

> TCP 协议虽然提供了 `KEEPALIVE` 的设置，允许客户端每隔一段固定的时间发送心跳包，但是由于 TCP 其实无法感知网络中断等问题，例如：网络路由、网线故障等。而且使用 `KEEPALIVE` 会对服务器造成较大的压力。同时在网络较差的情况下，由于 TCP 具有拥塞控制、保证有序等特性的要求，在环境处于低带宽/不稳定网络下时，TCP 对带宽的利用率很低。另外，因为网络的抖动等原因，连接断掉之后，TCP 需要三次握手才能重新建立连接，一旦出现频繁的小抖动就会使得带宽利用更低。所以一般在保持用户始终在线的要求下，一般不直接使用 `KEEPALIVE` ，而是在应用层实现心跳包机制。

业界常用方案

> - 在移动应用中，常用的是谷歌的消息推送服务 `GCM` 。
>   `GCM` 使用的是 TCP 长连接，使用心跳包机制，采用固定的时间间隔，在网络环境较好的情况下，GCM的连接极其稳定，断开次数很少。但是由于它的心跳时间间隔固定，WIFI 下是15分钟，数据网络下是28分钟，在 `NAT Aging-time` 设置较小的网络下，会导致TCP长连接在下一次心跳前被网关释放，造成延迟接收消息。
> - 微信（Android）采用的自适应心跳时间间隔。
>   同样采用 TCP 长连接的情况下，微信的心跳机制使用了自适应的心跳时间间隔，根据手机状态的不同，自动调整心跳时间间隔，这样在尽量不影响用户收消息及时性的前提下，根据网络类型自适应地找出保持 TCP 连接的尽可能大的心跳间隔，从而达到减少安卓微信因心跳引起的带宽消耗，减少服务器的负载，以及减少部分因心跳引起的耗电。
> - 桌面端 QQ
>   桌面端 QQ 在消息收发方面是使用 UDP 协议收发，由于是无连接的协议，效率高，速度快，占资源少，同时在网络环境较差的时候，建立 TCP 连接的概率很小，会影响消息传送的效率。但是因为 UDP 的不可靠的方式，又需要在上层来实现可靠性。

方案提出

所以在低带宽/不稳定网络条件下，实现保持用户始终在线，并保证消息准确收发的一个解决方案是：

1. 使用 TCP 保持长连接，心跳机制采用动态调整的心跳时间间隔。
2. 消息收发过程采用 UDP。消息收发过程中，发送端使用 UDP 协议发送消息，接收端收到之后，同样使用 UDP 协议发送消息给发送端，表明已经收到消息，保证了消息可以不遗漏地传输。发送端未接收到确认消息时，会主动重发消息，为了避免消息重复，一般的做法是每条消息都带上自己唯一ID，由接收端进行去重操作。
3. 消息协议方面，由于 XML 的冗余信息较多，不利于节省传输中的流量消耗，可以采用 Protocol Buffers 或者 MessagePack 这些更加轻量化的消息协议，在保证较好易读性的同时，也可以减少消息传输过程中的带宽消耗。