如何保持长连接

1352958 金敏

一. 不稳定网络中的解决方案

不稳定的网络下，最首先的假设是我们发送的数据包并不能保证正确或者完整到达目的地。将一次完整的连接过程中的两台机器简单地划分为主动请求方和被动请求方之后我们可以进行如下分别的讨论：

1. 发送方要确认已经和接受方正确地建立了一次连接，幸运的是传输层协议很有效地已经帮助我们保证了这一点。那么主动请求需要做的就是保留下来这次连接的描述符或者标识符，然后正常地进行数据地传输。至少到这里协议级别的长连接支持是必需的，哪怕是在一个私有协议栈上的长连接。就拿 HTTP 作为例子，从 HTTP/1.1 之后加入的 Keep-alive 选项就是对 WEB 上的会话的持久有效性的一个重要补充，从此才会有 Ajax 和高级 JS 应用。剩下来的事情就是判断什么时候去断开这个连接，简单地去划分，无非两种情况，一是本机主动断开连接，二是远程机主动断开连接，至于什么时候选择去断开一个连接，会在下面详述。

2. 接受方监听的端口上收到了连接的请求之后，服务端首先需要做的事情是保存下来这次连接的描述符，接下来在完成了正常数据的传输之后，就同样需要去判断这个长连接的状态，同样接受方也可以是这次连接的主动断开方，也可以是被动断开方。但是大多数情况下，接受方意味着连接的主动断开者。

考虑到主流传输层协议如 TCP，在不稳定的网络环境中，窗口大小会不断缩小，从而影响整个 IO 效率，一个“贪婪”的做法是当发现窗口大小收缩到一定程度的时候，主动刷新连接，所谓的刷新就是主动断开，再主动连接，这里先后要保留或者持久化窗口中的数据，以保证刷新连接后的数据流和正常不干预的数据流的数据是一致的。

二. 带宽优化的解决方案

带宽优化主要从以下 3 个方面去考虑：

1. 编码级的带宽优化，考虑消息本身的格式和字段特征，一是选择合适的二进制编码方式，二是对消息本身通过算法进行压缩。

2. 其中心跳消息可以只携带尽可能少的信息，比如 Cookie，比如版本号，Cookie 是为了防止伪造的心跳给网络 IO 带来负担，版本号可以帮助我们在不稳定的网络，心跳包丢失，错位的情况下，依然正确鉴别连接

的可靠性和连通性。

3. 消息丢失重发可以模仿四层协议中的做法，维护一个窗口去定时重发发送窗口中尚未得到确认的数据包。同样地，服务方有时候需要等待客户端特定的回复，可以维护回复等待窗口，定时发送回复确认请求。