讨论课一

1352837 倪雨婷

- **长连接心跳机制**

维护任何一个长连接都需要心跳机制，客户端发送一个心跳给服务器，服务器给客户端一个心跳应答，这样就形成客户端服务器的一次完整的握手，这个握手是让双方都知道他们之间的连接是没有断开，客户端是在线的。如果超过一个时间的阈值，客户端没有收到服务器的应答，或者服务器没有收到客户端的心跳，那么对客户端来说则断开与服务器的连接重新建立一个连接，对服务器来说只要断开这个连接即可。

网络中的接收和发送数据都是使用操作系统中的SOCKET进行实现。但是如果此套接字已经断开，那发送数据和接收数据的时候就一定会有问题。可是如何判断这个套接字是否还可以使用呢？这个就需要在系统中创建心跳机制。其实TCP中已经为我们实现了一个叫做心跳的机制。如果你设置了心跳，那TCP就会在一定的时间（比如你设置的是3秒钟）内发送你设置的次数的心跳（比如说2次），并且此信息不会影响你自己定义的协议。所谓“心跳”就是定时发送一个自定义的结构体（心跳包或心跳帧），让对方知道自己“在线”。 以确保链接的有效性。

所谓的心跳包就是客户端定时发送简单的信息给服务器端告诉它我还在而已。代码就是每隔几分钟发送一个固定信息给服务端，服务端收到后回复一个固定信息如果服务端几分钟内没有收到客户端信息则视客户端断开。比如有些通信软件长时间不使用，要想知道它的状态是在线还是离线就需要心跳包，定时发包收包。发包方：可以是客户也可以是服务端，看哪边实现方便合理。一般是客户端。服务器也可以定时轮询发心跳下去。心跳包之所以叫心跳包是因为：它像心跳一样每隔固定时间发一次，以此来告诉服务器，这个客户端还活着。事实上这是为了保持长连接，至于这个包的内容，是没有什么特别规定的，不过一般都是很小的包，或者只包含包头的一个空包。

在TCP的机制里面，本身是存在有心跳包的机制的，也就是TCP的选项。系统默认是设置的是2小时的心跳频率。但是它检查不到机器断电、网线拔出、防火墙这些断线。而且逻辑层处理断线可能也不是那么好处理。一般，如果只是用于保活还是可以的。心跳包一般来说都是在逻辑层发送空的包来实现的。下一个定时器，在一定时间间隔下发送一个空包给客户端，然后客户端反馈一个同样的空包回来，服务器如果在一定时间内收不到客户端发送过来的反馈包，那就只有认定说掉线了。只需要send或者recv一下，如果结果为零，则为掉线。

但是，在长连接下，有可能很长一段时间都没有数据往来。理论上说，这个连接是一直保持连接的，但是实际情况中，如果中间节点出现什么故障是难以知道的。更要命的是，有的节点（防火墙）会自动把一定时间之内没有数据交互的连接给断掉。在这个时候，就需要我们的心跳包了，用于维持长连接，保活。在获知了断线之后，服务器逻辑可能需要做一些事情，比如断线后的数据清理呀，重新连接呀当然，这个自然是要由逻辑层根据需求去做了。总的来说，心跳包主要也就是用于长连接的保活和断线处理。一般的应用下，判定时间在30-40秒比较不错。如果实在要求高，那就在6-9秒。

考虑低带宽情况，出现了MQTT协议。

MQTT

普通的socket连接对服务器的消耗太大了，所以才会出现像MQTT这种轻量级低消耗的协议来维护长连接。

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输）是IBM开发的一个即时通讯协议。MQTT协议是为大量计算能力有限，且工作在低带宽、不可靠的网络的远程传感器和控制设备通讯而设计的协议，它具有以下主要的几项特性：

1、使用发布/订阅消息模式，提供一对多的消息发布，解除应用程序耦合；

2、对负载内容屏蔽的消息传输；

3、使用 TCP/IP 提供网络连接；

4、有三种消息发布服务质量：

“至多一次”，消息发布完全依赖底层 TCP/IP 网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。

“至少一次”，确保消息到达，但消息重复可能会发生。

只有一次”，确保消息到达一次。这一级别可用于如下情况，在计费系统中，消息重复或丢失会导致不正确的结果。

5、小型传输，开销很小（固定长度的头部是 2 字节），协议交换最小化，以降低网络流量；

6、使用 Last Will 和 Testament 特性通知有关各方客户端异常中断的机制；

消息不遗漏

QQ客户端请求一个连接给服务器，服务器接收后，知道QQ端a上线，把QQ端a的账号跟目前的ip会记录下来，放在在线列表里或者其他的地方，然后每隔几分钟或者几秒钟给QQ端a发送心跳包，问他是否还在线，来确保QQ端a的最新状态。这里客户端与服务器的通信方式是UDP。而不会时时刻刻都在用TCP连接。

QQ聊天消息通信采用UDP协议，通过服务器中转方式。因此，现在的IP侦探在你仅仅跟对方发送聊天消息的时候是无法获取到IP的。大家都知道，UDP 协议是不可靠协议，它只管发送，不管对方是否收到的，但它的传输很高效。但是，作为聊天软件，必须保证可靠传输，于是腾讯采用了上层协议来保证可靠传输：如果客户端使用UDP协议发出消息后，服务器收到该包，需要使用UDP协议发回一个**应答包**。如此来保证消息可以无遗漏传输。之所以会发生在客户端明明看到“消息发送失败”但对方又收到了这个消息的情况，就是因为客户端发出的消息服务器已经收到并转发成功，但客户端由于网络原因没有收到服务器的应答包引起的。

- **消息不重复**

通常在普通的操作当中，我们不需要处理重复提交的，而且有很多方法来防止重复提交。比如在登陆过程中，通过使用redirect，可以让用户登陆之上重定向到后台首页界面，当用户刷新界面时就不会触发重复提交了。或者使用token，隐藏在表单中，当提交时进行token验证，验证失败也不让提交。这都是一般的做法。

通过使用session以及在session中加入token，来验证同一个操作人员是否进行了并发重复的请求，在后一个请求到来时，使用session中的token验证请求中的token是否一致，当不一致时，被认为是重复提交，将不准许通过。

整个流程可以由如下流程来表述：

1 客户端申请token

2 服务器端生成token，并存放在session中，同时将token发送到客户端

3 客户端存储token，在请求提交时，同时发送token信息

4 服务器端统一拦截用户的所有请求，验证当前请求是否需要被验证（不是所有请求都验证重复提交）

5 验证session中token是否和用户请求中的token一致，如果一致则放行

6 session清除会话中的token，为下一次的token生成作准备

7 并发重复请求到来，验证token和请求token不一致，请求被拒绝

由以上的流程，我们整个实现需要以下几个东西

1 token生成器，负责生成token

2 客户token请求处理action，负责处理客户请求，并返回token信息

3 token拦截器，用于拦截指定的请求是否需要验证token

4 token请求拦截标识，用于标识哪些请求是需要被拦截的

5 客户端token请求处理方法，用于请求token，并存放于特定操作中，并在提交时发送到请求中

- **消息压缩**

用Java Socket来传输对象，但是在有些情况下比如网络环境不好或者对象比较大的情况下需要把数据对象进行压缩然后在传输，此时就需要压缩这些对象流，此时就可以GZIPInputStream和GZIPOutputStream来处理一下socket的InputStream和OutputStream。

参考：http://blog.csdn.net/kongxx/article/details/7259834