不稳定网络长连接的处理

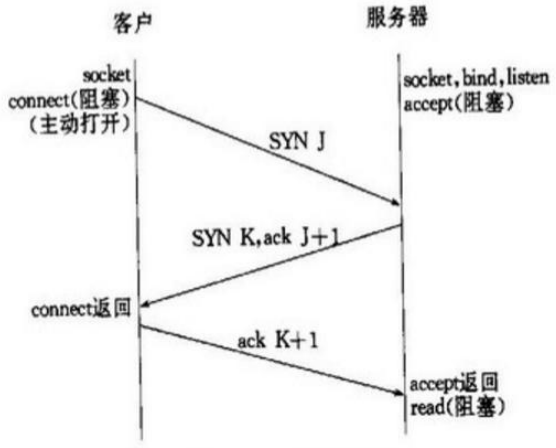
软件工程 潘舜达 1354366

一. 基础概念

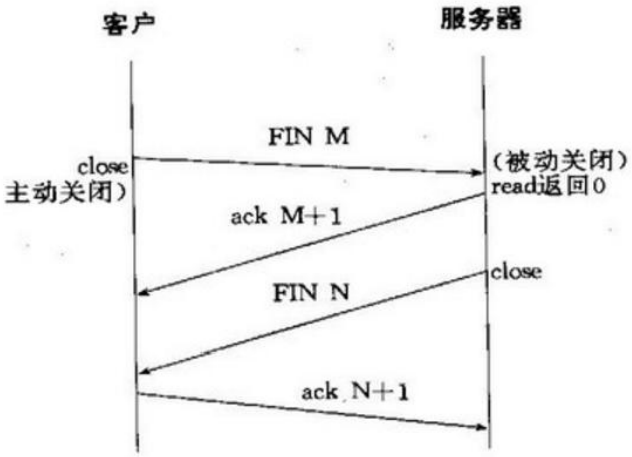
1. TCP 连接

当网络通信时采用 TCP 协议时，在真正的读写操作之前，server 与 client 之间必须建立一个连接，当读写操作完成后，双方不再需要这个连接时它们可以释放这个连接，连接的建立是需要三次握手的，而释放则需要 4 次握手，所以说每个连接的建立都是需要资源消耗和时间消耗的

经典的三次握手示意图：



经典的四次握手关闭图：



2. TCP 短连接

我们模拟一下 TCP 短连接的情况，client 向 server 发起连接请求，server 接到请求，然后双方建立连接。client 向 server 发送消息，server 回应 client，然后一次读写就完成了，这时候双方任何一个都可以发起 close 操作，不过一般都是 client 先发起 close 操作。为什么呢，一般的 server 不会回复完 client 后立即关闭连接的，当然不排除有特殊的情况。从上面的描述看，短连接一般只会在 client/server 间传递一次读写操作

3.TCP 长连接

接下来我们再模拟一下长连接的情况，client 向 server 发起连接，server 接受 client 连接，双方建立连接。Client 与 server 完成一次读写之后，它们之间的连接并不会主动关闭，后续的读写操作会继续使用这个连接。

首先说一下 TCP 保活功能，保活功能主要为服务器应用提供，服务器应用希望知道客户主机是否崩溃，从而可以代表客户使用资源。如果客户已经消失，使得服务器上保留一个半开放的连接，而服务器又在等待来自客户端的数据，则服务器将应远等待客户端的数据，保活功能就是试图在服务器端检测到这种半开放的连接。

从上面可以看出，TCP 保活功能主要为探测长连接的存活状况，不过这里存在一个问题，存活功能的探测周期太长，还有就是它只是探测 TCP 连接的存活，属于比较斯文的做法，遇到恶意的连接时，保活功能就不够使了。

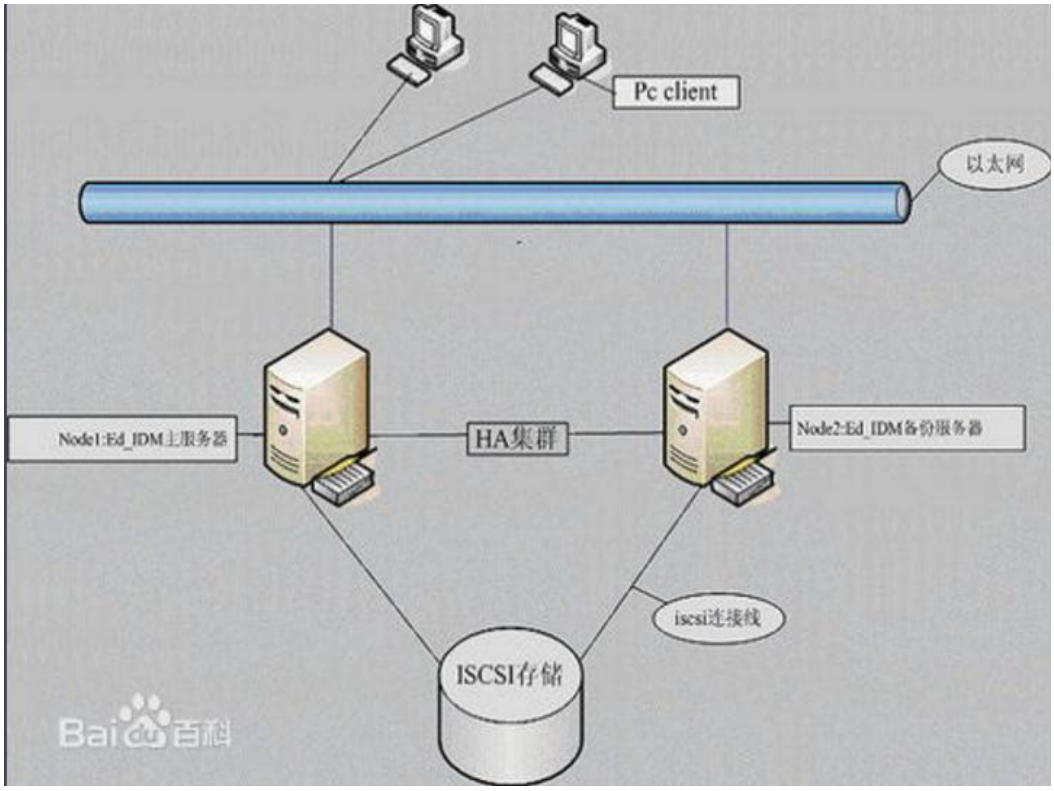
在长连接的应用场景下，client 端一般不会主动关闭它们之间的连接，Client 与 server 之间的连接如果一直不关闭的话，会存在一个问题，随着客户端连接越来越多，server 早晚有扛不住的时候，这时候 server 端需要采取一些策略，如关闭一些长时间没有读写事件发生的连接，这样可以避免一些恶意连接导致 server 端服务受损；如果条件再允许就可以以客户端机器为颗粒度，限制每个客户端的最大长连接数，这样可以完全避免某个蛋疼的客户端连累后端服务。

长连接和短连接的产生在于 client 和 server 采取的关闭策略，具体的应用场景采用具体的策略，没有十全十美的选择，只有合适的选择。

心跳机制

心跳机制是定时发送一个自定义的结构体(心跳包)，让对方知道自己还活着，以确保连接的有效性的机制。

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。



二．问题重述：

用户登录后始终在线，考虑低宽带/不稳定网络：

长连接心跳机制

消息不遗漏

消息不重复
消息压缩
参考业界现有解决方案分析

解决方案：

长连接心跳机制：

长连接与短连接的操作过程
通常的短连接操作步骤是：
连接→数据传输→关闭连接；
而长连接通常就是：
连接→数据传输→保持连接(心跳)→数据传输→保持连接(心跳)→……→关闭连接；
这就要求长连接在没有数据通信时，定时发送数据包(心跳)，以维持连接状态，
短连接在没有数据传输时直接关闭就行了

网络运营商的解决方案：

当一台智能手机连上移动网络时，其实并没有真正连接上 Internet，运营商分配给手机的 IP 其实是运营商的内网 IP，手机终端要连接上 Internet 还必须通过运营商的网关进行 IP 地址的转换，这个网关简称为 NAT(NetWork Address Translation)，简单来说就是手机终端连接 Internet 其实就是移动内网 IP，端口，外网 IP 之间相互映射。相当于在手机终端在移动无线网络这堵墙上打个洞与外面的 Internet 相连。
GGSN(GateWay GPRS Support Note 网关 GPRS 支持节点)模块就实现了 NAT 功能，由于大部分的移动无线网络运营商为了减少网关 NAT 映射表的负荷，如果一个链路有一段时间没有通信时就会删除其对应表，造成链路中断，正是这种刻意缩短空闲连接的释放超时，原本是想节省信道资源的作用，没想到让互联网的应用不得以远高于正常频率发送心跳来维护推送的长连接。这也是为什么会有之前的信令风暴，微信摇收费的传言，因为这类的应用发送心跳的频率是很短的，既造成了信道资源的浪费，也造成了手机电量的快速消耗。

消息不遗漏，不重复：

消息队列（也叫做报文队列）能够克服早期 unix 通信机制的一些缺点。消息队列就是一个消息的链表。可以把消息看作一个记录，具有特定的格式以及特定的优先级。对消息队列有写权限的进程可以向消息队列中按照一定的规则添加新消息；对消息队列有读权限的进程则可以从消息队列中读走消息。消息队列是随内核持续的。

目前主要有两种类型的消息队列：POSIX 消息队列以及系统 V 消息队列，系统 V 消息队列目前被大量使用。考虑到程序的可移植性，新开发的应用程序应尽量使用 POSIX 消息队列。
系统 V 消息队列是随内核持续的，只有在内核重起或者显示删除一个消息队列时，该消息队列才会真正被删除。因此系统中记录消息队列的数据结构（struct ipc_ids msg_ids）位于内核中，系统中的所有消息队列都可以在结构 msg_ids 中找到访问入口。 消息队列就是一个消息的链表。每个消息队列都有一个队列头，用结构 struct msg_queue 来描述。队列头中包含了该消息队列的大量信息，包括消息队列键值、用户 ID、组 ID、消息队列中消息数目等等，甚至记录了最近对消息队列读写进程的 ID。读者可以访问这些信息，也可以设置其中的某些信息。

Socket 套接字

套接字，是支持 TCP/IP 的网络通信的基本操作单元，可以看做是不同主机之间的进程进行双向通信的端点，简单的说就是通信的两方的一种约定，用套接字中的相关函数来完成通信过程。
常用的 TCP/IP 协议中套接字有 3 种，包括流套接字，数据包套接字，原始套接字。其中流套接字用于提供面向连接、可靠的数据传输服务。该服务将保证数据能够实现无差错、无重复发送，并按顺序接收。流

套接字之所以能够实现可靠的数据服务，原因在于其使用了传输控制协议，即 TCP（The Transmission Control Protocol）协议。

流式套接字提供没有记录边界的数据流：可以是双向的字节流（应用程序是全双工：可以通过套接字同时传输和接收）。可依赖流传递有序的、不重复的数据。（“有序”指数据包按发送顺序送达。“不重复”指一个特定的数据包只能获取一次。）这能确保收到流消息，而流非常适合处理大量数据。

流基于显式连接：套接字 A 请求与套接字 B 建立连接；套接字 B 接受或拒绝此连接请求。

消息压缩：

消息压缩在 WCF 中的实现其实很简单，我们只需要在消息（请求消息/回复消息）被序列化之后，发送之前进行压缩；在接收之后，反序列化之前进行解压缩即可。针对压缩/解压缩使用的时机，我们具有三种典型的解决方案。

1.通过自定义 MessageEncoder 和 MessageEncodingBindingElement 来完成。

2.直接创建用于压缩和解压缩的信道，在 CodePlex 中具有这么一个 WCF Extensions；

3.自定义 MessageFormatter 实现序列化后的压缩和法序列化前的解压缩

实际应用：

Android 系统的推送和 IOS 的推送的特点与区别：

iOS 的推送：就是 Apple 官方的 APNs （Apple Push Notification service）。

Android 的推送：Google 官方的是 GCM (Google Cloud Messaging）。

本质上，APNs 与 GCM 是类似的技术实现原理：即系统层有一个常驻的 TCP 长连接，一直保持的长连接，即使手机休眠的时候也在保持的长连接。

IOS 长连接是由系统来维护的，也就是说苹果的 IOS 系统在系统级别维护了一个客户端和苹果服务器的长链接，IOS 上的所有应用上的推送都是先将消息推送到苹果的服务器然后将苹果服务器通过这个系统级别的长链接推送到手机终端上，这样的的几个好处为：

1.在手机终端始终只要维护一个长连接即可，而且由于这个长链接是系统级别的不会出现被杀死而无法推送的情况。

2.省电，不会出现每个应用都各自维护一个自己的长连接。

3.安全，只有在苹果注册的开发者才能够进行推送，等等。

android 的长连接是由每个应用各自维护的，但是 google 也推出了和苹果技术架构相似的推送框架，C2DM,云端推送功能，但是由于 google 的服务器不在中国境内，所以导致这个推送无法使用，android 的开发者不得不自己去维护一个长链接，于是每个应用如果都 24 小时在线，那么都得各自维护一个长连接，这种电量和流量的消耗是可想而知的。虽然国内也出现了各种推送平台，但是都无法达到只维护一个长连接这种消耗的级别。

参考资料

互联网推送服务原理：长连接+心跳机制(MQTT 协议)

http://blog.csdn.net/clh604/article/details/20167263

心跳机制详解

http://www.tuicool.com/articles/uQR3y2

TCP 长连接和短连接的区别

http://www.cnblogs.com/liuyong/archive/2011/07/01/2095487.html

消息队列简介及其应用

http://www.cnblogs.com/sk-net/archive/2011/11/25/2232341.html

套接字详解

http://blog.csdn.net/abc78400123/article/details/6753867