**软件复用 讨论课1**

**复用解决方案**

1352914 计鹏玥

用户登录后始终在线，考虑低带宽/不稳定网络

（参考业界现有的解决方案）

[一．长连接心跳机制 1](#_Toc23782)

[1.心跳机制 1](#_Toc15180)

[2.长连接 2](#_Toc24359)

[3.互联网推送服务原理：长连接+心跳机制（MQTT协议） 3](#_Toc159)

[二．消息不遗漏 4](#_Toc22154)

[1.网络丢包 4](#_Toc16769)

[2.解决方法 4](#_Toc10088)

[三．消息不重复 6](#_Toc31358)

[1.“粘包”问题 6](#_Toc18251)

[2.解决方法 6](#_Toc20833)

[四．消息压缩 8](#_Toc22936)

[1.HTTP压缩算法SDCH 8](#_Toc9659)

[2.案例：视频压缩 9](#_Toc19054)

# 一．长连接心跳机制

## 1.心跳机制

心跳机制是定时发送一个自定义的结构体，也就是心跳包，让对方知道自己还活着，以确保连接的有效性的机制。心跳包其实是为了保持长链接，至于包的内容并没有特别的规定，一般情况下都是很小的包，或者是只含有包头的一个空包。

在TCP的机制中本身就是含有心跳包的机制的，也就是TCP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。一般来说，如果只是用于保活，这种方式还是可以的。定时器每当过了一定时间间隔，就会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。只需要发送或者接收一下，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，用于维持长连接和保活。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和重新连接，这个是要由逻辑层根据需求去实现的。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。

## 2.长连接

长连接是指在一个连接上可以连续发送多个数据包，在连接保持期间，如果没有数据包的发送，需要双方发链路检测包。长连接多应用于手机推送，其优点是相对稳定安全，缺点是消耗的资源较多。与长连接相对应的是短连接。

短连接是指通讯双方有数据交互时，就建立一个连接，数据发送完成后，则断开此连接，即每次连接只完成一项业务的发送。

长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况。每个TCP连接都需要三步握手，这需要时间，如果每个操作都是短连接，再操作的话那么处理速度会降低很多，所以每个操作完后都不断开，下次处理时直接发送数据包就OK了，不用建立TCP连接。例如：数据库的连接用长连接，如果用短连接频繁的通信会造成socket错误，而且频繁的socket 创建也是对资源的浪费。

而像WEB网站的HTTP服务一般都用短链接，因为长连接对于服务端来说会耗费一定的资源，而像WEB网站这么频繁的成千上万甚至上亿客户端的连接用短连接会更省一些资源，如果用长连接，而且同时有成千上万的用户，如果每个用户都占用一个连接的话，那可想而知吧。所以并发量大，但每个用户无需频繁操作情况下需用短连好。

总之，长连接和短连接的选择要视情况而定。

## 3.互联网推送服务原理：长连接+心跳机制（MQTT协议）

手机推送消息的原理都是维护一个长连接，但是普通的socket连接对服务器的消耗太大，因而才出现了像MQTT这种轻量级低消耗的协议来维护长连接。

首先，智能手机在连接移动网络时，其实并没有真正连接上Internet，还需要运营商的网关NAT（Network Address Translation）来进行IP地址的转换。然而大部分的移动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。

其次，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。

最后，推送的实现方式分为以下三点：

a.客户端不断的查询服务器，检索新内容，也就是所谓的pull 或者轮询方式；

b.客户端和服务器之间维持一个TCP/IP长连接，服务器向客户端push；

c.服务器又新内容时，发送一条类似短信的信令给客户端，客户端收到后从服务器中下载新内容，也就是SMS的推送方式。

# 二．消息不遗漏

## 1.网络丢包

网络丢包是我们在使用ping对目站进行询问时，数据包由于各种原因在信道中丢失的现象。ping使用了ICMP回送请求与回送回答报文。ICMP回送请求报文是主机或路由器向一个特定的目的主机发出的询问，收到此报文的机器必须给源主机发送ICMP回送回答报文。这种询问报文用来测试目的站是否可到达以及了解其状态。需要指出的是，ping是直接使用网络层ICMP的一个例子，它没有通过运输层的UDP或TCP。

网络丢包的原因主要有物理线路故障、设备故障、病毒攻击、路由信息错误等。

## 2.解决方法

1. 物理线路故障

如果是物理线路故障所造成网络丢包现象，则说明故障是由线路供应商提供的线路引起的，需要与线路供应商联系尽快解决问题。联系你的服务商来解决网络丢包很严重的情况。

1. 设备故障

设备方面主要包括软件设置不当、网络设备接口及光纤收发器故障造成的。这种情况会导致交换机端口处于死机状态。那么可以将你的光纤模块更换掉，换一条新的模块替换掉。

1. 网络被堵塞、拥堵

当网络不给力的时候，在通过网络传输数据，就会将网络丢包更多，一般是路由器被占用大量资源造成的。解决方法就是这时应该show process cpu和show process mem，一般情况下发现IP input process占用过多的资源。接下来可以检查fast switching在大流量外出端口是否被禁用，如果是，则需要重新使用。用show interfaces和show interfaces switching命令识别大量包进出的端口。一旦确认进入端口后，打开IP accounting on the outgoing interface看其特征，如果是攻击，源地址会不断变化但是目的地址不变，可以用命令“access list”暂时解决此类问题。

1. 路由错误

网络中的路由器的路径错误也是会导致数据包不能正常传输到主机数据库上这种情况属于正常状况，它所丢失的数据也是很小的。所以用户可以忽略这些数据丢包，而且这也是避免不了的。

# **三．消息不重复**

## 1.“粘包”问题

TCP（Transport Control Protocol，传输控制协议）是面向连接的，面向流的，提供高可靠性服务。收发两端（客户端和服务器端）都要有成对的socket，因此，发送端为了将多个发往接收端的包，更有效的发到对方，使用了优化方法（Nagle算法），将多次间隔较小且数据量小的数据，合并成一个大的数据块，然后进行封包。这样，接收端，就难于分辨出来了，必须提供科学的拆包机制。即面向流的通信是无消息保护边界的。

UDP（User Datagram protocol，用户数据报协议）是无连接的，面向消息的，提供高效率服务。不会使用块的合并优化算法，, 由于UDP支持的是一对多的模式，所以接收端的skbuff(套接字缓冲区）采用了链式结构来记录每一个到达的UDP包，在每个UDP包中就有了消息头（消息来源地址，端口等信息），这样，对于接收端来说，就容易进行区分处理了。 即面向消息的通信是有消息保护边界的。

导致“粘包”出现会有两种情况：

1. 发送端需要等缓冲区满才发送出去，造成粘包；
2. 接收方不及时接收缓冲区的包，造成多个包接收。

## 2.解决方法

1. 对于发送方引起的粘包现象，用户可通过编程设置来避免，TCP提供了强制数据立即传送的操作指令push，TCP软件收到该操作指令后，就立即将本段数据发送出去，而不必等待发送缓冲区满。

这种编程设置方法虽然可以避免发送方引起的粘包，但它关闭了优化算法，降低了网络发送效率，影响应用程序的性能，一般不建议使用。

1. 对于接收方引起的粘包，则可通过优化程序设计、精简接收进程工作量、提高接收进程优先级等措施，使其及时接收数据，从而尽量避免出现粘包现象。

这种方法只能减少出现粘包的可能性，但并不能完全避免粘包，当发送频率较高时，或由于网络突发可能使某个时间段数据包到达接收方较快，接收方还是有可能来不及接收，从而导致粘包。

c.由接收方控制，将一包数据按结构字段，人为控制分多次接收，然后合并，通过这种手段来避免粘包。

最后这种方法虽然避免了粘包，但应用程序的效率较低，对实时应用的场合不适合。

# 四．消息压缩

## 1.HTTP压缩算法SDCH

a.Google发明的HTTP压缩算法SDCH

SDCH的全称是Shared Dictionary Compression over HTTP的简写。SDCH的基于以下的事实：

在HTTP的传输文件之间有大量重复冗余的数据，从服务器端一次又一次的传输到请求端；

服务器端可以对这些文件之间的大量相同数据建立字典；

客户端第一次访问时，请求字典与差异文件。之后每次访问，都只需要传输差异文件，结合本地的字典就可以还原相应文件。

b.HTTP中的压缩算法Gzip

Gzip中使用的算法是LZ77算法和哈夫曼编码，是单个文件内部文本信息的前后参照，共享信息实现压缩。

c.设计模式之单例模式、享元模式

单例模式是在系统中只包含一个被称为单例类的特殊类。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。

享元模式它使用共享物件，用来尽可能减少内存使用量以及分享资讯给尽可能多的相似物件；它适合用于当大量物件只是重复因而导致无法令人接受的使用大量内存。

d.数据库连接池

数据库连接是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。

数据库连接池正是针对这个问题提出来的。数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而再不是重新建立一个；释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。这项技术能明显提高对数据库操作的性能。

## 2.案例：视频压缩

视频压缩技术是计算机处理视频的前提。视频信号数字化后数据带宽很高，通常在20MB/秒以上，因此计算机很难对之进行保存和处理。采用压缩技术通常数据带宽降到1-10MB/秒，这样就可以将视频信号保存在计算机中并作相应的处理。常用的算法是由ISO制订的，即JPEG和MPEG算法。JPEG是静态图像压缩标准，适用于连续色调彩色或灰度图像，它包括两部分：一是基于DPCM（空间线性预测）技术的无失真编码，一是基于DCT（离散余弦变换）和哈夫曼编码的有失真算法，前者压缩比很小，主要应用的是后一种算法。在非线性编辑中最常用的是MJPEG算法，即Motion JPEG。它是将视频信号50帧/秒（PAL制式）变为25帧/秒，然后按照25帧/秒的速度使用JPEG算法对每一帧压缩。通常压缩倍数在3.5-5倍时可以达到Betacam的图像质量。MPEG算法是适用于动态视频的压缩算法，它除了对单幅图像进行编码外还利用图像序列中的相关原则，将冗余去掉，这样可以大大提高视频的压缩比。前MPEG-I用于VCD节目中，MPEG-II用于VOD、DVD节目中。

高质量图像压缩在传输视频图像时使用了1/2的带宽，而不佳的压缩技术不仅没能提供高质量图像，更是占用了你全部的带宽。良好的压缩技术可以提供给你低带宽下高帧率的高清图像。