不稳定网络下的长连接

1352911 曹琦

基础知识

TCP长连接和心跳机制

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。长连接可以省去较多的TCP建立和关闭的操作，减少浪费，节约时间。对于频繁请求资源的客户来说，较适用长连接。

TCP的Socket本身就是长连接的，而且本身具有心跳机制（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），也就是TCP的选项：SO_KEEPALIVE。系统默认是设置的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、对方进程挂掉、频繁丢包等异常情况。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。为了处理异常情况，可以在逻辑层发送空的echo包。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。

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。

HTTP长连接

HTTP持久连接（HTTP persistent connection，也称作HTTP keep-alive或HTTP connection reuse）是使用同一个TCP连接来发送和接收多个HTTP请求/应答，而不是为每一个新的请求/应答打开新的连接的方法。在 HTTP 1.1 中 所有的连接默认都是持续连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。

HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。

HTTP持久连接的优点如下：

- 较少的CPU和内存的使用（由于同时打开的连接的减少了）
- 允许请求和应答的HTTP管线化
- 降低网络阻塞 （TCP连接减少了）
- 减少了后续请求的延迟（无需再进行握手）
- 报告错误无需关闭TCP连接

HTTP协议的长连接，实质上是TCP协议的长连接。

解决方案

设置合理的心跳周期

以移动无线网络为例：

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。

GGSN(GateWay GPRS Support Note)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。以中移动的2.5G网络为例，大约5分钟左右的基带空闲，连接就会被释放。

由于移动无线网络的特点，推送服务的心跳周期并不能设置的太长，否则长连接会被释放，造成频繁的客户端重连，但是也不能设置太短，否则在当前缺乏统一心跳框架的机制下很容易导致信令风暴（例如微信心跳信令风暴问题，信令风暴指网络收到的终端信令请求超过了网络各项信令资源的处理能力而导致网络服务出现问题。一些手机应用程序会频繁地和服务器交换少量的数据以确保用户保持在线状态，这样会占用大量网络资源，当信令流量超出网络的信令处理能力后，网络便会失控并导致服务不可用）。具体的心跳周期并没有统一的标准，180S也许是个不错的选择，微信为300S。

减少消息丢失和重复

参考京东云推送相关技术，通过消息回执确认机制、高可用分布式服务器集群确保不丢消息。

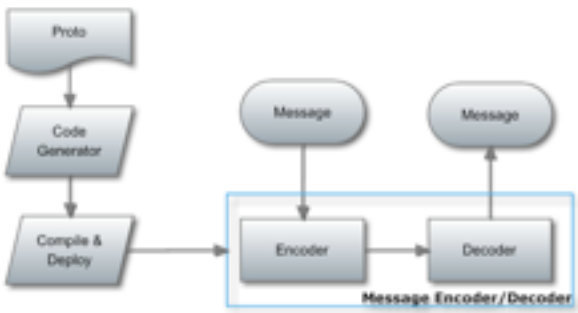
- **消息回执确认**。用户终端每收到一条消息，会给云端发送回执确认。消息发送给用户终端应用后，终端JDPushService服务发送收到该消息的回执确认给云端服务，云端把消息状态置为已送达。手机网络不稳定造成长连接中断时，JDPushService向云端服务发起重连时，如果未收到回执确认并且消息未过期，云端会重新发送该消息给用户终端，确保即使网络不稳定，用户始终能收到消息。
- **消息持久化**。为了防止应用重复收到同一条消息，终端JDPushService服务对最新消息做持久化，收到消息时判断是否已收过该消息，如果未收过就发送给应用，否则不再发送。同时发送回执给云端，有效解决重复推送问题。
- **高可用分布式服务器集群**。系统可用性差也会造成消息丢失，比如消息发送处理中，系统出现意外，就有消息丢失可能。分布式服务器集群架构，每个服务节点没有单点，确保系统的高可用。各个实例节点间通过MQ通信，对每个job是否成功的反馈确认，如果worker节点在执行job时崩溃或者退出，该job会再分发给其他woker节点执行，确保消息处理环节不丢失。数据存储集群提供完善的数据备份和故障转移支持，确保数据在存储环节不丢失。

消息压缩

在实际编程中，网络带宽的有效数据负载率是一个值得考虑的问题。特别地，对于移动客户端来说，网络资源往往并不是很丰富，为了尽可能地节省网络资源，往往需要尽大可能地增加数据包的有效数据率。

基于protobuf的数据编码协议，与其他的编码协议如xml，json相比，protobuf有着更好的传输效率和压缩比率。使用protobuf进行数据编码后的消息大小只有基于Json的编码的20%左右。

protobuf协议是由google制定的，主要用于其内部的rpc调用和文件编码。原生的protobuf包括两部分内容：基于二进制的数据编码协议和基于proto元数据的代码生成器。首先，需要根据每条消息来编写对应的proto文件，然后使用google提供的代码生成器，基于proto文件来生成相应的编码器和解码器，然后使用生成的编/解码器来进行编/解码操作，对应的流程如下图：



这种方式的优势是代码静态生成，运行时不需要proto文件信息，而且可以根据具体的信息内容对代码进行优化，编解码的时候不需要类型元信息，效率很高。

减少延时和加载开销

参考360消息系统的架构，我们可以结合服务端做一些策略，比如我们在选路时候，我们会将同一个用户尽量映射到同一个room service实例上（长连接网关）。断线时，客户端尽量对上次连接成功的地址进行重试。主要是方便服务端做闪断情况下策略，会暂存用户闪断时实例上的信息，重新连入的时候，做单实例内的迁移，减少延时与加载开销。

参考资料

[1] Netty系列之Netty百万级推送服务设计要点 http://www.infoq.com/cn/articles/netty-million-level-push-service-design-points

[2] 如何实现支持数亿用户的长连消息系统 | Golang高并发案例 http://chuansong.me/n/1641640

[3] 消息压缩 https://github.com/NetEase/pomelo/wiki/消息压缩

[4] 浅谈京东云推送----节省 稳定 精准 http://www.csdn.net/article/a/2014-03-18/15818234