问题重述

用户登陆后始终在线，考虑低带宽/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

解决方案

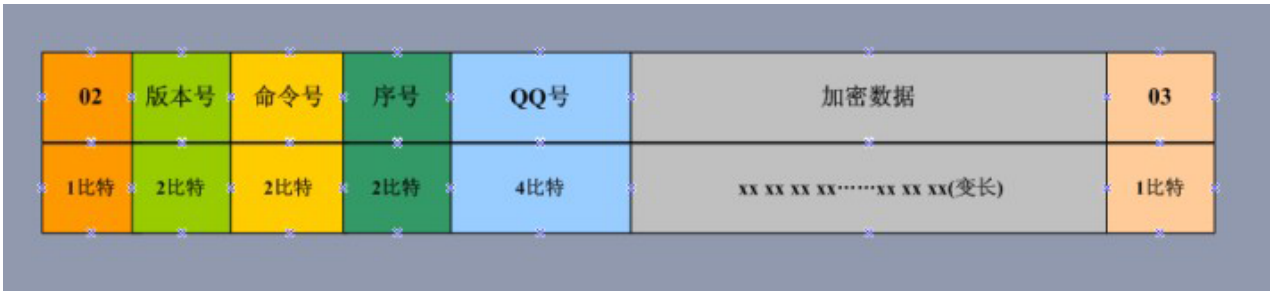
1. 长连接心跳机制

由于服务器资源有限，为了保证不浪费系统资源和网络开销，当客户端非正常退出或出现其他意外时服务器可以尽快终止连接以释放此用户所占用的资源。QQ 的客户端只需每隔一分钟向服务器发送一次“心跳”包就可以保持此连接。而 MSN 则在客户端和服务器两端都提供这种类型的消息。当客户端发送“心跳”消息后，服务器的响应中包含了下一次发送此消息的间隔时间。当服务器发送“心跳”消息时，其消息内包含 challenge 信息，客户端需回复此 challenge 信息的 MD5 值和用户账户。相比较而言，微信的保持连接的机制更为复杂，不但在两端都提供“心跳”包，而且在 TCP层和应用层都使用心跳机制。在应用层上，客户每隔 5 分 50 秒和 10 分钟发送两种不同类型的“心跳”消息；在 TCP 层上，客户端每隔 1 分钟发送，而服务器的发送间隔时间为 2 分30 秒。

在我们的项目里，我们使用的库为NIO2，它自带了长连接心跳机制，其中有一个选项就是 SO_KEEPALIVE 表示保持长连接。至于心跳包的时间应该设置为多少比较合适我认为这个应该具体情况具体分析。如果心跳包时间过短，这就变相成了 DDOS 攻击，过长则会因为没有及时检测出断线的客户端而浪费 socket 资源。一些文章指出应该根据客户端到服务端的RTT时间动态设置，我认为这也是比较好的一个方案。

2. 消息不遗漏

参考QQ数据包协议：



采用二进制的数据传输格式。其数据包都由0x02开头并由0x03结尾。其中，版本号表示正在通信的客户端的版本信息。命令号则代表一种具体的通信目的，序号由一个全局的整数表示，客户端每发送一个命令请求后序号加1，而服务器响应某一请求的应答数据包必须包含与客户请求一样的命令号和序号，这样可以保证实现异步通信。QQ号码即为当前与服务器通信的QQ用户号码。

具体在我们的项目里我们使用了队列来维护我们的消息：在发送的时候，每条消息都有一个序号和时间戳，将该消息插入到队伍中去。每条消息发送出去后，会又重新插入回队尾，当收到消息后，我们会检查它的ACK，并将队列里的序号相同的消息从队列中删除。每次会从队伍头取出一条消息，首先检查它是否是新消息，如果是的话就发送并插到队尾，如果不是的话就检查时间戳是否超时，如果未超时就直接插到队尾，如果超时的话就重发、更新时间戳并插回队尾。如果该条消息的重试次数超过预设的次数（我们设置为3次），则丢弃该消息。

3. 消息不重复

同样参考上面的解决方案，接受的时候会维护一个队列，每次取出队伍头的一条消息，如果该条消息的序号之前处理过，则丢弃；如果没有处理过，就取出并处理并标记序号。

3. 消息压缩

我们这个聊天室项目的特点是时间大多都花在了数据传输上。故而相比于计算时间，网络传输时间要远大于计算时间。基于以上的特点，我们应该最大限度的压缩数据，以计算时间换取网络传输时间。

根据Google几年前发布的一组测试数据（数据有些老了，有人近期做过测试的话希望能共享出来）：

Algorithm	% remaining	Encoding	Decoding
GZIP	13.4%	21MB/s	118MB/s
LZO	20.5%	135MB/s	410MB/s

Zippy/Snappy	22.2%	172MB/s	409MB/s

其中：
1. GZIP的压缩率最高，但是其实CPU密集型的，对CPU的消耗比其他算法要多，压缩和解压速度也慢；
2. LZO的压缩率居中，比GZIP要低一些，但是压缩和解压速度明显要比GZIP快很多，其中解压速度快的更多；
3. Zippy/Snappy的压缩率最低，而压缩和解压速度要稍微比LZO要快一些。

我们可以选用Gzip进行压缩。理由如上。