软件复用讨论课

任子卓

1252976

一、长连接心跳问题

1、长连接心跳机制基本介绍

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。为了避免服务器发送数据和接受数据过程中产生的问题，就要引入一种机制，确保链接的有效性。这就引入了长连接心跳机制，该机制实现包括几个要素：

a、网络切换或者初始化时 server ip 的获取
b、连接前的 ip 筛选，出错之后 ip 的抛弃
c、维护长连接的心跳。
d、服务器通过长连 notify。
e、选择使用长连通道的业务。
f、断开后重连的策略。

2、业界对于该机制的实现

对于长连接心跳机制的实现，经过查阅一些资料，了解到业界微信对于这一部分的处理机制。心跳机制的目的很简单：通过定期的数据包，对抗 NAT 超时，因此微信心跳的实现主要通过以下几步：

a、连接后主动到服务器 Sync 拉取一次数据，确保连接过程的新消息。
b、心跳周期的 Alarm 唤醒后，一般有几秒的 cpu 时间，无需 wakelock。
c、心跳后的 Alarm 防止发送超时，如服务器正常回包，该 Alarm 取消。
d、如果服务器回包，系统通过网络唤醒，无需 wakelock。

3、基于项目对于该机制的实现

由于操作系统可以将长期不响应的连接关闭，因此我们的项目需要引入心跳机制。在我们的项目中，使用的是 Java 原生的 socket 实现服务端和客户端，参考业界的实现方法，我们可以新开一个线程对于其他线程进行监管，如果在心跳周期内服务正常回包则将该线程置为可靠线程。

二、消息不重复问题

1、基于项目对于该问题的解决

对于消息重复的问题通过参考业界的解决方案可以客户端的代码中增加一段消息过滤的代码，即开发一个消息过滤器，对于服务端发来的信息进行判断已解决消息重复的问题。

三、消息遗漏的问题

1、基于项目对于该问题的解决

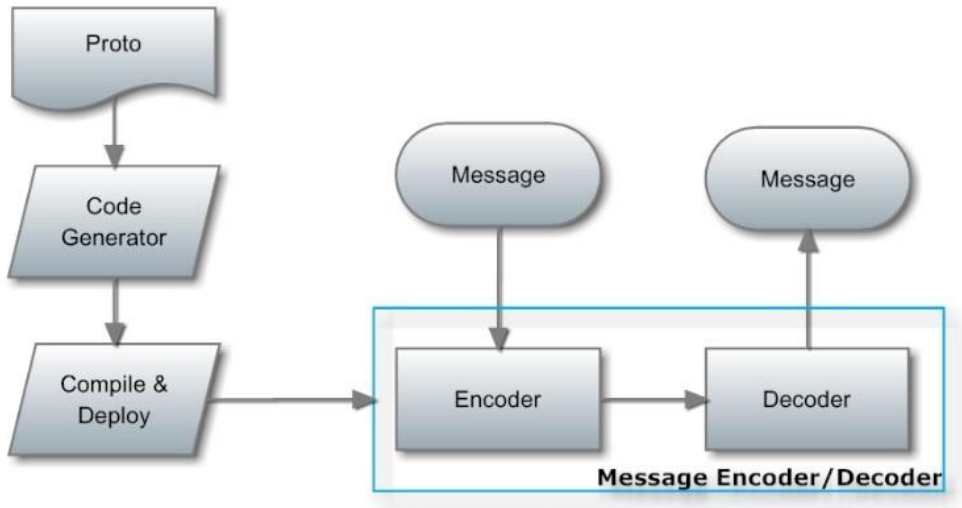
引用数据包机制，将单个消息划分为多个数据块，而这些数据块包含着该信息的属性信息，例如数据包头包括该消息的长度，通过对于这一部分的判断来判断消息。

四、消息压缩的问题

1、业界对于该问题的解决方法

经过查阅资料，业界普遍是通过某种方法进行数据的编在进行消息传输时，pomelo 实现了基于 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。在我们的 lordofpomelo 项目中，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。码与解码进而对于消息的处理压缩。例如我所查阅到的一种基于 protobuf 的传输数据压缩：

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：



2、基于项目对于该问题的解决

我们的项目中也可以引用类似的的机制，通过某种协议对于信息进行处理，对于所要发送的消息进行编码，进而达到信息压缩的目的。