讨论课内容
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

在这一次的项目中，我们要实现一个长连接的用户登录系统。在项目讨论的初级阶段我们小组产生了两种实现方案。

方案一：使用 Socket 作为通信机制

方案二：使用 HTTP 协议

使用 HTTP 的特点就是，由于 HTTP 在每次请求结束后都会主动释放连接，因此 HTTP 连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即时不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

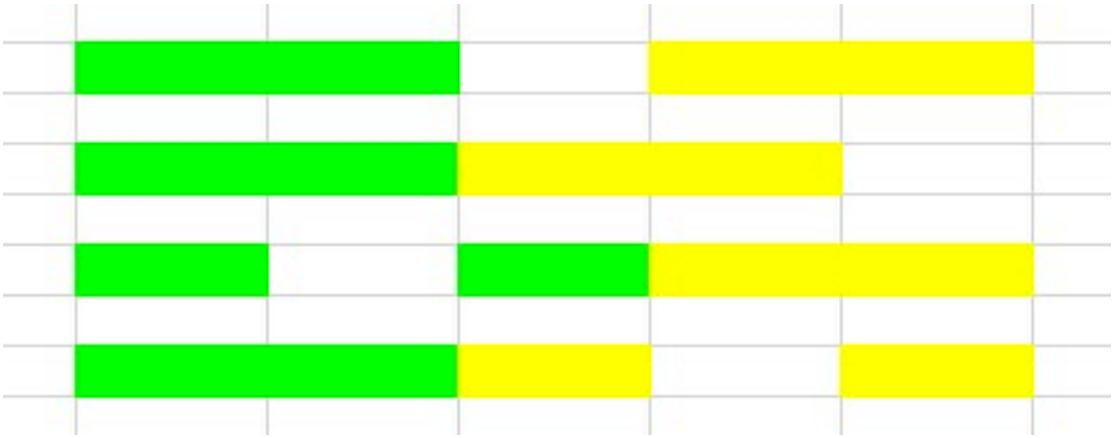
但是如若使用 Socket，连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。在这个过程中，服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。也就是说，Socket 实际上是一直在连接。

但在实际网络应用中，客户端到服务器之间的通信往往需要穿越多个中间节点，例如路由器、网关、防火墙等，大部分防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态。基于这一点，我们选择了 Netty 进行开发。Netty 是一个一步的、事件驱动的网络应用程序框架，可以简化网络应用的开发，例如提供内部的心跳长连接机制。

对于消息的发送，首先将消息转化成二进制流，客户端进行发送，服务器进行接收。其中包含缓冲区，以防止消息过长，分次发送，接收不完整。这个缓冲区在网络不稳定时会发挥作用。

对于 NIO 的 SocketChannel，在非阻塞模式下，它会直接返回连接结果，如果没有连接成功，也没有发生 IO 异常，则需要将 SocketChannel 注册到 Selector 上监听连接结果。所以，异步连接的超时无法在 API 层面直接设置，而是需要通过定时器来主动监测。创建连接超时定时任务之后，会由 NioEventLoop 负责执行。如果已经连接超时，但是服务端仍然没有返回 TCP 握手应答，则关闭连接。

TCP 是个"流"协议，所谓流，就是没有界限没有分割的一串数据。TCP 会根据缓冲区的实际情况进行包划分，一个完整的包可能会拆分成多个包进行发送，也用可能把多个小包封装成一个大的数据包发送。这就是 TCP 粘包/拆包。



在本项目中粘包/拆包解决办法

1．固定格式，例如每个报文长度固定，不够补空格

2．使用回车换行符分割，在包尾加上分割符，例如 Ftp 协议

3．添加解码器。例如 LineBasedFrameDecoder，StringDecoder