每个客户端client的应用缓冲区的处理

每一次从socket中read和write尽可能多的数据，然后判断是否能重组一个消息体，如果客户端提交的一个包，大小超过了client的缓冲区大小，则需要调整缓冲区的大小，server需要发送的一个包，要组成一个消息体，然后一次发送给客户端，如果超过了缓冲区大小，调整缓冲区大小，最好分多个消息处理。

天龙的做法是socketinputstream 最大为96\*1024 socketoutputstream 最大为100\*1024如果超过的话，就断开连接。

使用epoll分场景处理的问题

在client登陆的时候，一次性把有关玩家的所有数据提取到服务器内存中，在玩家进入场景之前保证他能拥有所有想拥有的数据即可。这样在client登陆的时候就会有一个等待过程了，

给在这里等待的玩家一个登陆的虚拟场景，收发使用一个网络模型epoll，等所有数据准备好了，让这个玩家登陆到场景，进入场景之中，而场景使用的是另外一个epoll模型了，当然进入场景的处理过程中就要把这个玩家的连接socket放到这个epoll中来。这样就按照场景分开了玩家了。

天龙做法是每个场景一个线程，每个线程在处理过程中各自使用一个select模型。每一次线程处理循环中，先read 来自client的发送消息并缓冲下来，然后进行消息处理，最后进行消息的发送。如果某个客户端没有发送，但是server仍然有消息要发送给他的话，异步read的时候会直接返回，并且进行消息处理，不会阻断，所以没有影响这个客户端和其他客户端。

但是这里的轮询情况如何规避？ 是select处理了没有消息发送来的情况，超时返回的时候进行轮询每个客户端是否有发出消息和客户端处理。

考虑一个客户端长连接但是不想发包的情形，一般情况下要让客户端发送心跳连接，并在心跳连接的处理中记录一个时间，在server某个心跳处理线程中来轮询所有客户端的心跳时间，如果超时就kick。该心跳处理线程是要间隔运行的。