socket的实现机制

```
int port = 2018;
string host = "127.0.0.1";

IPAddress ip = IPAddress.Parse(host);
IPEndPoint ipe = new IPEndPoint(ip, port);

Socket sSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
sSocket.Bind(ipe);
sSocket.Listen(0);
Console.WriteLine("监听已经打开，请等待");

//收到消息 接受一个socket链接
Socket serverSocket = sSocket.Accept();
Console.WriteLine("连接已经建立。。。");
while (true)
{
string recStr = "";
byte[] recByte = new byte[4096];
int bytes = serverSocket.Receive(recByte, recByte.Length, 0);
recStr += Encoding.ASCII.GetString(recByte, 0, bytes);
Console.WriteLine("服务器端获得信息:{0}", recStr);

if (recStr.Equals("stop"))
{
serverSocket.Close();//关闭该socket对象
Console.WriteLine("关闭链接。。。。");
break;
}

//回发消息
Console.WriteLine("请输入回发消息。。。。");
string sendStr = Console.ReadLine(); //"send to client :hello world";
byte[] sendByte = Encoding.ASCII.GetBytes(sendStr);
serverSocket.Send(sendByte, sendByte.Length, 0);

}
```

sSocket.Close();//关闭server监听

心跳检测机制：
检测客户端是否掉线，客户端在线的话，每隔固定时间给服务器发送一个心跳包，服务器立即返回一个应答。

断线重连机制
心跳检测失败之后，再次contect一次，重新产生新的session实例。

Socket 粘包和分包问题

概念
Socket通信时会对发送的字节数据进行分包和粘包处理，属于一种Socket内部的优化机制。
粘包：
当发送的字节数据包比较小且频繁发送时，Socket内部会将字节数据进行粘包处理，既将频繁发送的小字节数据打包成 一个整包进行发送，降低内存的消耗。
分包：
当发送的字节数据包比较大时，Socket内部会将发送的字节数据进行分包处理，降低内存和性能的消耗。

例子解释
当前发送方发送了两个包，两个包的内容如下：
123456789
ABCDEFGH
1
2
3
我们希望接收方的情况是：收到两个包，第一个包为：123456789，第二个包为：ABCDEFGH。但是在粘包和分包出现的情况就达不到预期情况。

粘包情况
两个包在很短的时间间隔内发送，比如在0.1秒内发送了这两个包，如果包长度足够的话，那么接收方只会接收到一个包，如下：

123456789ABCDEFGH
1
分包情况

假设包的长度最长设置为5字节（较极端的假设，一般长度设置为1000到1500之间），那么在没有粘包的情况下，接收方就会收到4个包,如下：

12345
6789
ABCDE
FGH
1
2
3
4
处理方式
因为存在粘包和分包的情况，所以接收方需要对接收的数据进行一定的处理，主要解决的问题有两个：

在粘包产生时，要可以在同一个包内获取出多个包的内容。
在分包产生时，要保留上一个包的部分内容，与下一个包的部分内容组合。
目前处理方式主要两种：

一、给数据包的头尾加上标记。
比如在数据包的头部加上“START”字符串，尾部加上"END"字符串，这样可以解析出START和END之间的字符串就是接收方需要接收的内容。（当然真正处理的时候不可能使用START和END这种混效率较高的字符串，此处只是个例子）
上边两个包的例子就可以如下：

START123456789END
STARTABCDEFGHEND
1
2
二、在数据包头部加上内容的长度
发送方在发送的时候就可以在包头加上包的长度，接收方每次接收的时候都根据头部的长度去获取后面的内容。
上边两个包的例子就可以如下：

PACKAGELENGTH:0009123456789
PACKAGELENGTH:0008ABCDEFGH
1

2
处理例子
头尾标记处理
粘包
START123456789ENDSTARTABCDEFGHEND
1
获取第一个START和第一个END的位置，然后获取他们之间的内容，第二个包的内容就是获取第二个START和第二个END的位置。

分包
START1234567
89END
1
2
每个包要判断最后是否是END结尾，如果没有找到END，那么就保留上一个包START之后的内容，与下一个包第一个END之前的内容组合。

头部长度处理
粘包
PACKAGELENGTH:0009123456789PACKAGELENGTH:0008ABCDEFGH
1
获取“PACKAGELENGTH:”这个字符串后面4个字符，转化为数字就是包的长度，根据包的长度获取后面的内容，第二个内容的长度就是获取第二个“PACKAGELENGTH:”字符串后面的4个字符。

分包
PACKAGELENGTH:0009123456
789
1
2
获取“PACKAGELENGTH:”这个字符串后面4个字符，转化为数字就是包的长度，如果包结尾还没有获取完，那么就要获取下一个包前面的部分内容。

部分细节情况
看了前面的例子，比较善于思考的读者肯定已经想到了一些其他问题，这些问题处理起来方式和上面相似，笔者在此罗列一下，就不重复解释了，相信聪明的读者能够自己解决：

1、粘包和分包问题一起出现

START123456789ENDSTARTAB

CDEFGHEND

1

2

2、头尾标志由于分包获取不完整

START123456789E

ND