UDP是以报文方式传输的，接收方接收到的报文会以一个完整的报文形式传递给传输层。在发送UDP报文时，或者在UDP报文的传送过程中，即使被分片了，但接收端的IP层也会将分片的报文组合成一个完整的UDP报文，然后传递给传输层。

TCP是基于流传输的，不存在数据边界。因此所谓的TCP粘包和拆包的概念是不存在的，而之所以有这个问题是因为程序没有在TCP的基础上设计应用层的协议。

1. 拆包的产生：

发送端要发送数据abcdefg，而接收端的窗口为5，所以发送端只能先发送abcde。接收端取走接收到的数据以后通知发送端窗口大于0可以继续发送了，然后发送端才继续发送剩下的数据。发送端调用一次send()函数来发送TCP数据，而接收端分两次都能接收完数据。

1. 发送端产生的粘包：

接收端的窗口很大，而此时发送端要连续发送多个较小的报文段，发送端由于Nagle算法而将数据缓存在发送缓存中然后一起发送，这样接收端就一次性接收到多个数据报文段的数据而无法区分数据边界。

1. 接收端产生的粘包：

接收端的主机繁忙，接收到TCP报文段后没能及时取走数据。这时又来了新的数据报文段，由于TCP数据流没有边界，两次接接收的报文段的数据粘在一块无法区别边界。

1. 其它拆包和粘包的情况
2. 两个报文段，第一个报文段被拆包，第一个报文段剩下的数据和第二个报文段产生粘包。
3. 两个报文段，第二个报文段被拆包，第一个报文段和第二个报文段的前部分粘包。

解决TCP拆包和粘包的方法是应用层定义自己的报文格式。比如，应用层的每个报文都以某一个magicNumber作为开头，然后紧跟着应用层报文的长度，再接下来是应用层报文的数据。也就是说只要应该层实现自己的封包和解包过程，就可以解决TCP协议的粘包和拆包问题。