# 【第04章-TCP粘包拆包问题的解决之道】【01-TCP粘包拆包】

## TCP粘包拆包

TCP 是个“流”协议，所谓流，就是没有界限的一串数据。大家可以想想洞里的流水，它们是连成一片的，其间并没有分界线。TCP 底层并不了解上层业务数据的具体含义， 它会根据TC P 缓冲区的实际情况进行包的划分，所以在业务上认为，一个完整的包可能会被TCP 拆分成多个包进行发送，也有可能把多个小的包封装成一个大的数据包发迭， 这就是所谓的TCP 粘包和拆包问题。

## TCP 粘包/拆包问题说明

通过图解对TCP 粘包和拆包问题进行说明



图1 TCP粘包拆包问题

假设客户端分别发送了两个数据包D l 和D2 给服务端，由于服务端一次读取到的字节数是不确定的，故可能存在以下4 种情况。

（1）服务端分两次读取到了两个独立的数据包，分别是D1和D2 ，没有粘包和拆包：

（2）服务端一次接收到了两个数据包， D1和D2 粘合在一起，被称为TC P 粘包；

（3）服务端分两次读取到了两个数据包，第一次读取到了完整的D1 包和D2 包的部分内容，第二次读取了D2 包的剩余内容，这被称为TCP 拆包；

（4）服务端分两次读取到了两个数据包，第一次读取到了D1包的部分内容Dl-1，第二次读取到了D l 包的剩余内容D1-2 和D2包的整包。

如果此时服务端TCP 接收滑动窗口非常小，而数据包D1 和D2 比较大，很有可能会发生第5 种可能， 即服务端分多次才能将Dl和D2 包接收完全，期间发生多次拆包。

## TCP 粘包/拆包发生的原因

问题产生的原因有三个：

（1）应用程序write 写入的字节大小大于套接口发送缓冲区大小：

（2）进行MSS 大小的TCP 分段；

（3）以太网帧的payload 大于MTU 进行IP 分片。

如图2所示：



图2 TCP 粘包拆包问题原因

## 粘包问题的解决策略

由于底层的TCP 无法理解上层的业务数据，所以在底层是无法保证数据包不被拆分和重组的，这个问题只能通过上层的应用协议战设计来解决，根据业界的主流协议的解决方案，可以归纳如下。

（1）消息定长，例如每个报文的大小为固定长度200 字节，如果不够，空位补空格：

（2）在包尾增加回车换行符进行分剖，例如FTP 协议：

（3）将消息分为消息头和消息体，消息头中包含表示消息总长度（或者消息体长度）的字段，通常设计思路为消息头的第一个字段使用int32 来表示消息的总长度；

（4）更复杂的应用层协议。