

13 | 传输协议：应用程序之间对话的语言

2019-08-20 李玥



你好，我是李玥。

经过前面几课的学习，我们已经可以实现高性能的结构化数据传输了。不过，应用程序之间要想互相通信，一起配合来实现业务功能，还需要有一套传输协议来支持。

传输协议就是应用程序之间对话的语言。设计传输协议，并没有太多规范和要求，只要是通信双方的应用程序都能正确处理这个协议，并且没有歧义就好了。

这节课，我们就来说一下设计高性能传输协议的一些方法和技巧。

如何“断句”？

既然传输协议也是一种语言，那么在应用程序之间“通话”的过程中，与我们人类用自然语言沟通有很多相似之处，但是需要处理的问题却又不同。

现代语言，无论是汉语还是英语，都是通过标点符号来分隔句子的，这个叫“断句”。古代汉语是没有标点符号的，断句全靠上下文，但这种断句方式有的时候会出现歧义，比如很著名的那个段子“下雨天留客天天留我不留”，不同的断句方式，意思完全不一样。

我们在传输数据的时候，首先要解决的就是断句问题。对于传输层来说，收到的数据是什么样的？就是一段一段的字节，但是，因为网络的不确定性，你收到的分段并不一定是我们发出去的分段。比如我们发送的数据是这样的：

下雨天 留客天 天留 我不留

这样断句，意思就是，作为主人我不想让你在我这儿住。

经过网络传输，可能就变成这样了：

下雨天 留客天 天留我不 留

意思完全变了，客人想赖在这儿不走了。

所以，靠时间停顿来断句是不靠谱的。

你可能会想到，那我们在协议中也加上“标点符号”不就行了？而且，我们并不需要像自然语言中那么多种标点符号，只需要定义一个分隔符就可以了。

这个办法是可行的，也有很多传输协议采用这种方法，比如HTTP1协议，它的分隔符是换行（\r\n）。但是，这个办法有一个问题比较难处理，在自然语言中，标点符号是专用的，它没有别的含义，和文字是有天然区分的。

在数据传输的过程中，无论你定义什么字符作为分隔符，理论上，它都有可能会在传输的数据中出现。为了区分“数据内的分隔符”和真正的分隔符，你必须得在发送数据阶段，加上分隔符之前，把数据内的分隔符做转义，收到数据之后再转义回来。这是个比较麻烦的过程，还要损失一些性能。

更加实用的方法是，我们给每句话前面加一个表示这句话长度的数字，收到数据的时候，我们按照长度来读取就可以了。比如：

03下雨天03留客天02天留03我不留

这里面我们固定使用2位数字来存放长度，每句话最长可以支持到99个字。接收后的处理就比较简单了，我们先读取2位数字03，知道接下来的3个字是第一句话，那我们接下来就等着这3个字都收到了，就可以作为第一句话来处理了，接下来再按照这个方法读第二句话、第三句话。

这种预置长度的方法就很好解决了断句的问题，并且它实现起来要比分隔符的方法简单很多，性能也更好，是目前普遍采用的一种分隔数据的方法。

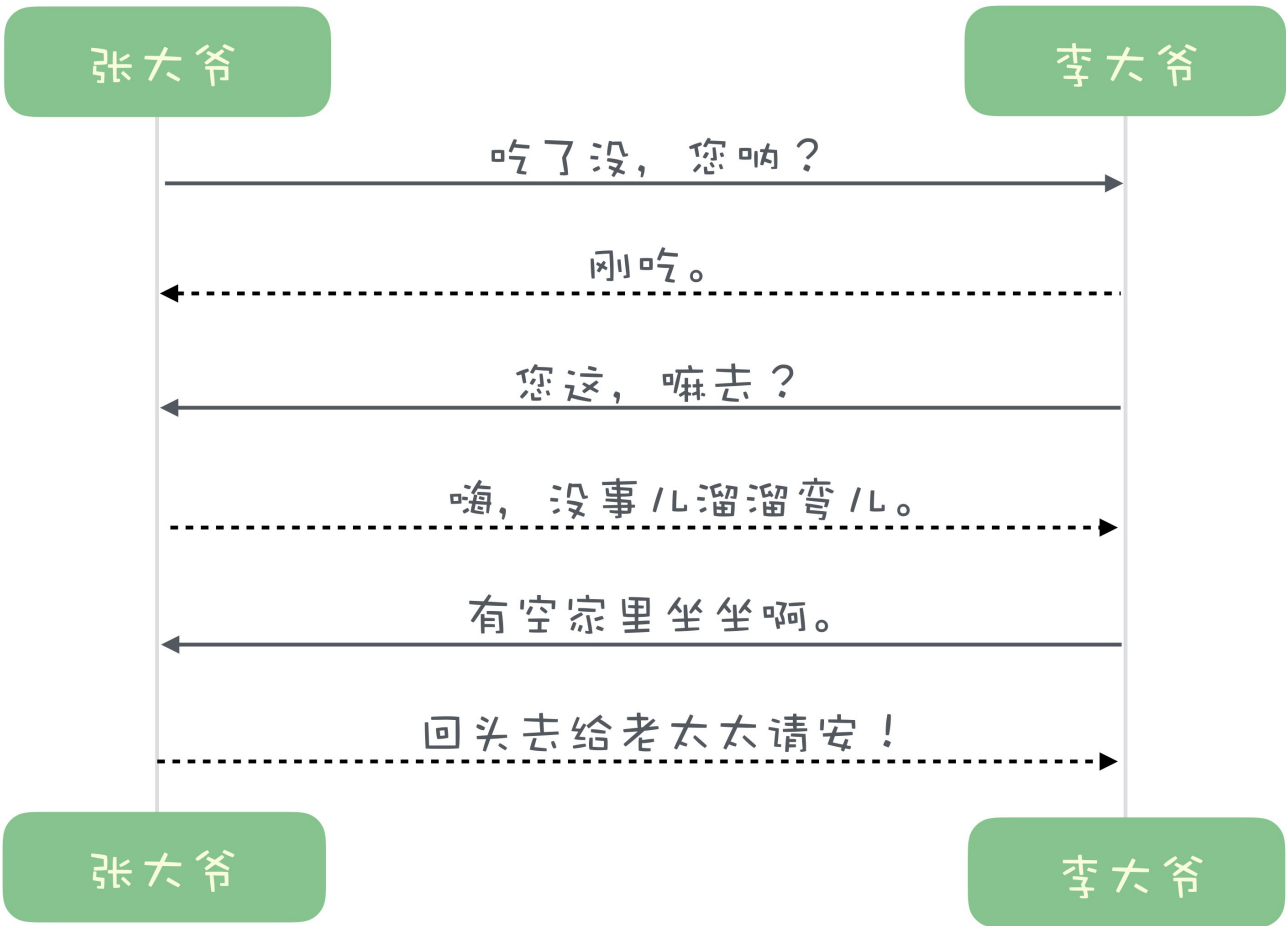
掌握了断句的方法之后，我们再来看一下实现高性能协议还需要解决什么问题。

用双工收发协议提升吞吐量

人类之间通过语言来交流时，基本上处于一种单工通信的状态，也就是我说你听，然后再你说我听这样。如果俩人同时说，那就不是交流了，那是两个外国人在吵架。所谓的单工通信就是，任何一个时刻，数据只能单向传输，一个人说的时候，另外一个人只能听。

HTTP1协议，就是这样一种单工协议，客户端与服务端建立一个连接后，客户端发送一个请求，直到服务端返回响应或者请求超时，这段时间内，这个连接通道上是不能再发送其他请求的。这种单工通信的效率是比较低的，很多浏览器和App为了解决这个问题，只能同时在服务端和客户端之间创建多个连接，这也是没有办法的办法。

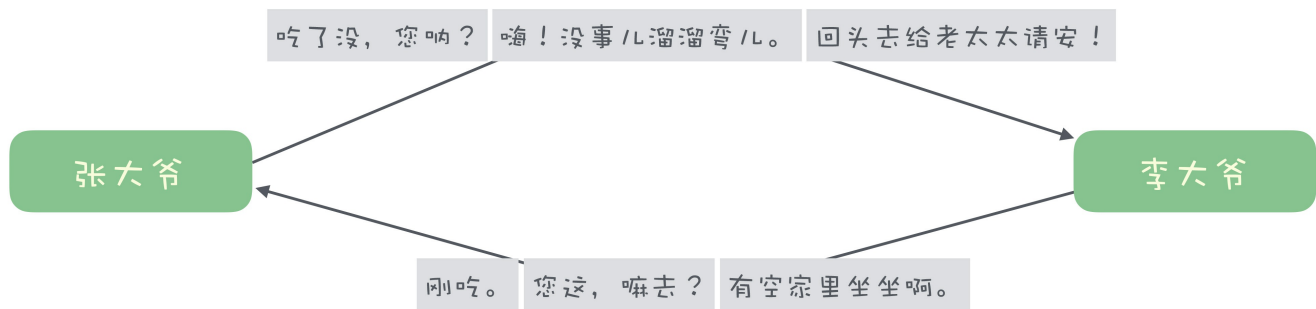
单工通信时，一句对一句，请求和响应是按照顺序依次收发，有一个天然的对应关系。比如说，胡同口张大爷和李大爷俩大爷碰上了：



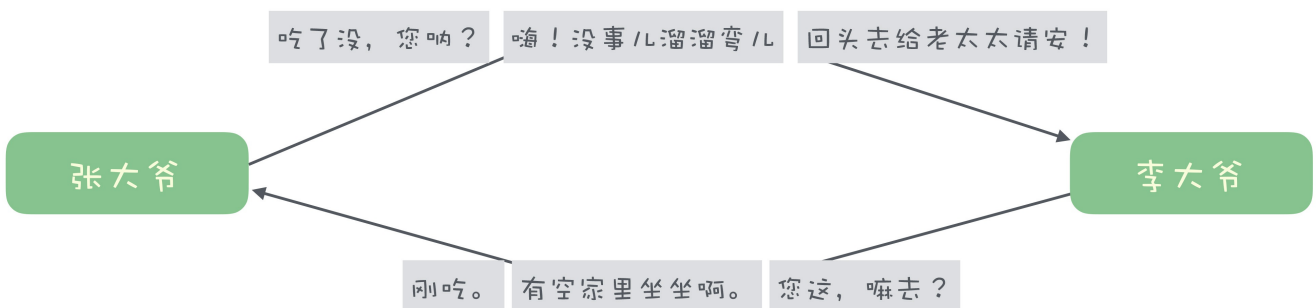
这个图里面，实线是请求，虚线是响应，一问一答，这是单工协议。

我们知道，TCP连接它是一个全双工的通道，你可以同时进行数据的双向收发，互相是不会受到任何影响的。要提高吞吐量，应用层的协议也必须支持双工通信。

如果说俩大爷有边听边说的本事，换成双工协议后，是这样的：



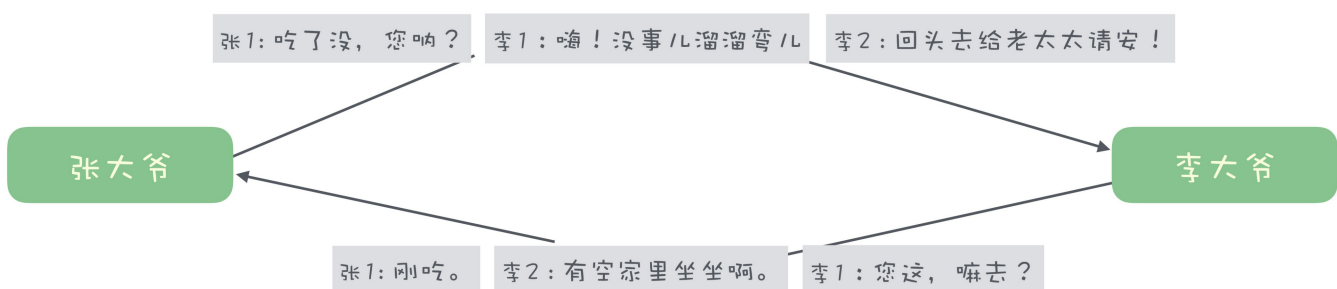
这时候就出现一个问题, 即使俩大爷有这个边听边说的本事, 问题和答案可能已经对不上了。在多线程并发的环境下, 顺序也没有办法保证, 这个对话就有可能变成这样:



在实际上设计协议的时候, 我们一般不关心顺序, 只要需要确保请求和响应能够正确对应上就可以了。

这个问题我们可以这样解决: 发送请求的时候, 给每个请求加一个序号, 这个序号在本次会话内保证唯一, 然后在响应中带上请求的序号, 这样就可以把请求和响应对应上了。

加上序号后, 俩大爷的就可以实现双工通信了:



张大爷和李大爷可以对自己发出去请求来编号, 回复对方响应的时候, 带上对方请求的编号就可以了。这样就解决了双工通信的问题。

小结

这节课我们主要讲了传输协议, 在设计传输协议的时候, 只要双方应用程序能够识别传输协议, 互相交流就可以了, 并没有什么一定要遵循的规范。

在设计传输协议的时候, 需要解决如何断句的问题, 我们给大家提供了“分隔符”和“前置长度”两

种断句的方法，你可以选择使用。

另外，我给大家介绍的这种“使用ID来标识请求与响应对应关系”的方法，是一种比较通用的实现双工通信的方法，可以有效提升数据传输的吞吐量。

解决了断句问题，实现了双工通信，配合专用的序列化方法，你就可以实现一套高性能的网络通信协议，实现高性能的进程间通信。很多的消息队列、RPC框架都是用这种方式来实现它们自己的私有应用层传输协议。

思考题

课后，我希望你能真正动手去写代码，用我们这四节课讲到的方法，来实现一个简单的高性能通信程序。功能就是上面两个大爷那三组对话，服务端是张大爷，客户端是李大爷，我们让俩人在胡同口碰见一百万次，记录下总共的耗时。欢迎你在评论区秀出你的总耗时。

在实现过程中，有任何问题，也欢迎你在评论区留言来提问。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

 极客时间

消息队列高手课

从源码角度全面解析 MQ 的设计与实现

李玥

京东零售技术架构部资深架构师



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言



滴流乱转小胖子

没想到老师居然是个相声演员，通俗易懂，点赞

2019-08-20

9



A9

👍 2

看了直播，没想到老师你是这样的人 所以，到底谁快？

2019-08-20

作者回复

同学当然是你最快呀！👍

2019-08-21



chon

👍 2

第一篇的内容质量很高，第二篇的目前这几篇文章的内容实在是太基础了。不用动脑，跳着看

2019-08-20



刘天鹏

👍 1

<https://gist.github.com/liutianpeng/85ce524452c8206396c94ab93506deda>

一个"胡同"做中转 两个"大爷"TCP连接到胡同

我这个版本的胡同效率有点低 大爷相遇1万次就用了 3.8s

2019-08-20

作者回复

赞分享代码的同学👍

2019-08-21



许童童

👍 1

跟着老师把这些基础知识打牢，很喜欢老师这种讲课节奏。

2019-08-20



linqw

👍 1

课后习题用netty做，耗时大致3秒左右，使用LengthFieldBasedFrameDecoder。

学习完这篇也写下自己的理解，字节流就像流水，为此我们在接收和发送字节流的时候，需要对此进行编码和解码，常见的几种形式1、定长，比如指定固定的长度，解析的时候获取固定的长度为一个完整的语句2、分隔符，比如在发送时，对字节流中使用分隔符分隔完整的语句。3、最常用的一种就是在发送的字节流中，有固定的字节表示长度

有个疑问，在接收的过程中接收到半包，如何处理了，等待剩余的接收完成么？老师帮忙解答下哦

2019-08-20

作者回复

是需要等待的，如果你使用Netty，这个问题Netty会帮你处理好。

2019-08-21



Hurt

👍 1

继续 打卡

2019-08-20



godtrue

👍 0

打卡，已经习惯打卡

这节明白没问题，不过给我的感受是还是基础最重要。

如果计算机组成原理、操作系统原理、计算机网络原理、数据结构与算法、编译原理这几门课我的基础打的更牢靠一些，极客时间的一些课程就不用买了，至少有些课程听起来就简单多啦！这些基础知识才是构建计算机知识大厦的根基。

总是，感谢玥哥分享，突然想到为啥公司内部JMQ分享没听过你的呢？

2019-08-23



oldman

0

老师，我理解的双工通信，是不是说不管是客户端还是服务端建立好链接之后，双方都可以基于该socket进行收发消息就好了，而不是说服务器只能accept到message之后再做一些处理。

2019-08-23



coffee

0

代码见 [git@github.com:swgithub1006/-geektime-mqstudy.git](https://github.com/swgithub1006/geektime-mqstudy) ,采用netty实现。

机器是4核，server端 bossGroup = new NioEventLoopGroup(8); workerGroup = new NioEventLoopGroup(1); client端有8个channel.机器上同时跑server和client。以下是相遇次数和执行耗时。

相遇次数 执行耗时（秒）

10000 2

100000 5

1000000 不稳定（min 65 , max 100）

2019-08-22

作者回复

协议部分设计的非常好。

Netty的使用也非常熟练。

使用了8组连接并发，实际的性能要除以8哦。

这是单工通信还是双工通信呢？

最后，使用减号开头的项目名称对*nix用户严重不友好啊。

2019-08-23



奇奇

0

03 然后天留我先到达 这不就完了？

2019-08-22



奇奇

0

如果发送网络时序出现问题，怎么办？

2019-08-22

作者回复

一般来说，只要请求和响应能对上就可以了，在网络上传输的这些请求和响应，并不需要严格有序。

2019-08-23



大白先生

👍 0

老师，那在一次会话过程中，开头的先是唯一序列号么。然后后面跟的是数据长度，再然后是内容么。那接到消息的一方，该如何分辨序列号的长度大小，做到区分序列号和内容前的数据长度信息？

2019-08-21

作者回复

开头是数据长度，序号也是数据的一部分，所以应该在长度之后。

2019-08-22



A9

👍 0

<https://github.com/WangYangA9/netty-FullDuplex-example> 作业写完啦，大概时长5秒左右，有待优化，使用netty框架，协议使用Kryo序列化协议（类似上面的例子，4字节表示数据长度，后续记录对应长度数据）。

为了模拟真实的顺序相应，做了很多同步等待，包括每次tcp连接的断开也进行了同步。

客户端什么时候断开连接稍微想了一下。开始的时候，由于消息没收完就断开了连接导致报错。最后实现方案是，收到三条该收的信息后，断开连接。但是这样的代价是，没找到ChannelFuture有类似then的毁掉方法，addListener反而导致性能下降，因此选择同步等待后，再将任务状态置为true，这些等待应该是可以后续优化的点。

请问老师，用什么方式判断一次碰面的3次收发都结束了比较好呢？

2019-08-21

作者回复

可以考虑用Barrier或者更简单的CountDownLatch来解决你的问题。

2019-08-22



宋晓明

👍 0

老师，昨天您的直播我看了 但心中一直有个疑问：一般架构师coding能力非常强，尤其是java，现在很多招聘要求都是java架构师，目前本人擅长的语言是python和go，说实话 java虽好，但本人很不喜欢，是不是我与架构师就无缘了？？

2019-08-21

作者回复

跟语言关系不大，其实各种编程语言背后的实现原理都是差不多的。一般的大厂的架构师职位对语言也没有强要求，而且很多架构师都是掌握多门编程语言的。

2019-08-22



nimil

👍 0

赞👍，通俗易懂

2019-08-21



朱振光

👍 0

前置长度是不是也有类似的问题，03也可能是正常文字里的内容，也是需要转义吧

2019-08-20

作者回复

你可以想一下最好自己实现一下接收数据进行解析的代码，你就会明白，前置长度是不需要转

义的。

因为在解析的时候，可以明确的知道当前读到的这个位置应该是长度还是真正的数据，它是不需要根据数据流中的内容来确定的。

2019-08-21



learn more

👍 0

redis 的 aof 文件好像就是老师说的 前置长度，瞬间觉得经典无处不在

2019-08-20



张三

👍 0

这节的重点是实现

2019-08-20



知己逢知遇

👍 0

多线程下异步处理一次会话的结果消息体，除了对这次会话的结果消息进行编号，是否也要对分割的消息体进行顺序编号？

电驴，迅雷，p2p这种软件的消息协议大概是什么样的呢？是不是部分协议跟今天老师讲的情况类似？

我是不是可以理解为，在双全功下，我和一个网站就可以建立一条长链接，然后所有的资源请求都通过这条链接进行交换，只要协议正确，就可以保证数据的完整性，正常访问并解析打开的该网站的所有页面。

多线程，异步的io操作，是不是也有自己的协议？

2019-08-20

作者回复

你需要了解，协议是分层的，就像我们发快递，我发给你的可能是个手机，对于快递小哥来说，这就是个小包裹，他不关心里面是什么，总之我保证给你把包裹安全的送到站点儿就行了。包裹到了站点儿会分拣装箱，然后用小货车运到机场，对于小货车司机来说，它也不关心车里装的是什么，只要把车安全开到机场就行了。

对于协议来说已是这样在发送的时候一层一层的封装，然后接收的时候再一层一层解封，对于每一层协议来说，他是不知道底层是什么协议的，也不知道上层协议是如何封装的（送件的快递小哥不知道也不关心这个件是怎么到配送站的，也不知道包裹里到底装的啥），只在自己的协议层完成处理即可。

2019-08-20