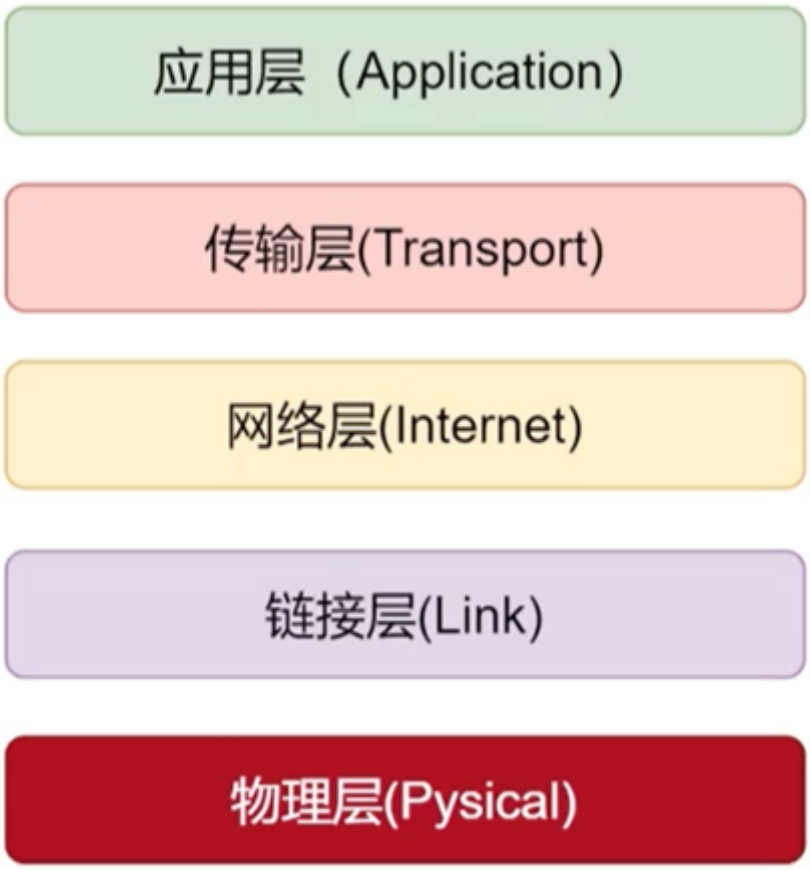
课程目标

TCP/IP协议群简介

TCP/IP协议群也称互联网协议群(Internet Protocol Suite)，TCP(Transmission Control Protocol)类似的OSI模型，一种网络协议的概念模型，对OSI有一定的简化，表现层和会话层，简化成了应用层。这样符合我架构的习惯，我们不习惯一层分的太细，应用层，表现层和会话层本来就内聚的，分成三成后，程序不太好写。精简下来后就是五层。

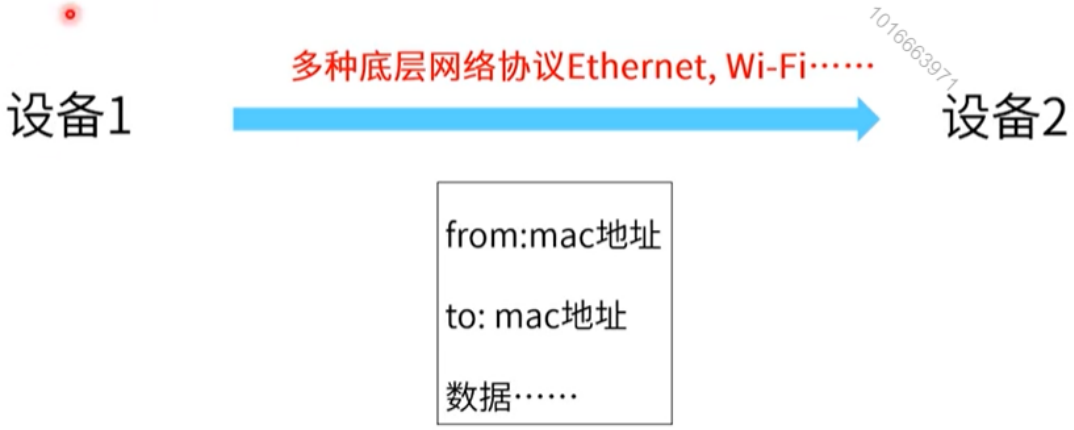


应用层：提供应用层之间的通信能力

传输层：提供主机到主机的通信能力(host-to-host)

网络层: 提供地址到地址的通讯能力

链路层:提供设备到设备的通信能力

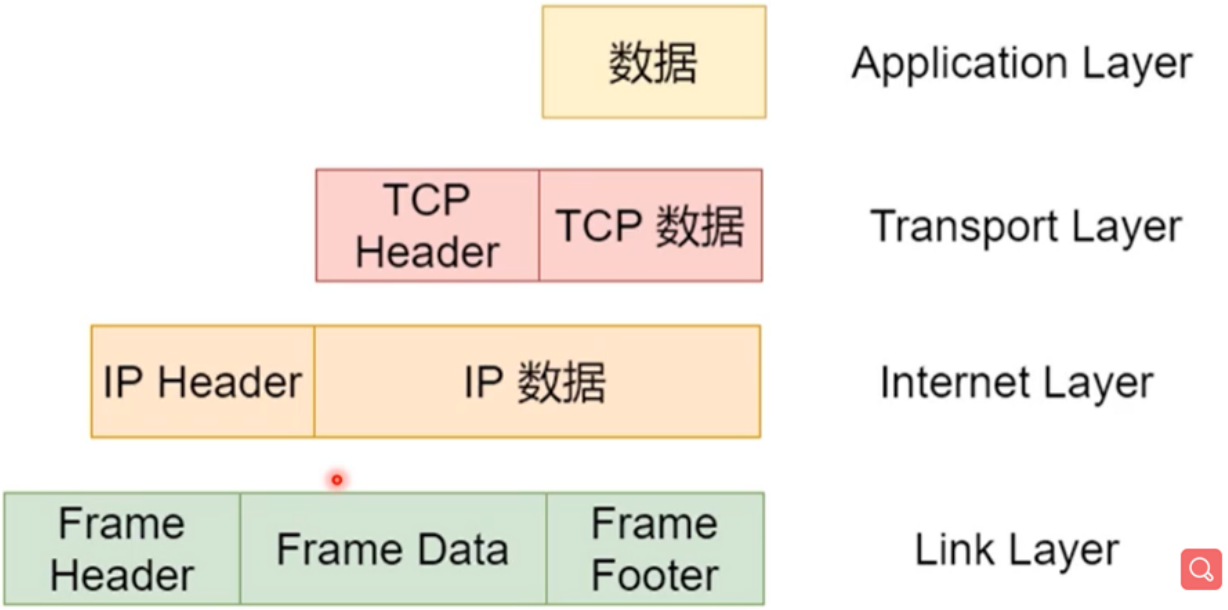


重要标识

应用的端口号:最容易理解，8080一般给服务器端java应用的，因为一台服务器里面有多应用，消息到主机，要确保消息发送给哪个应用，需要端口号进行标识。

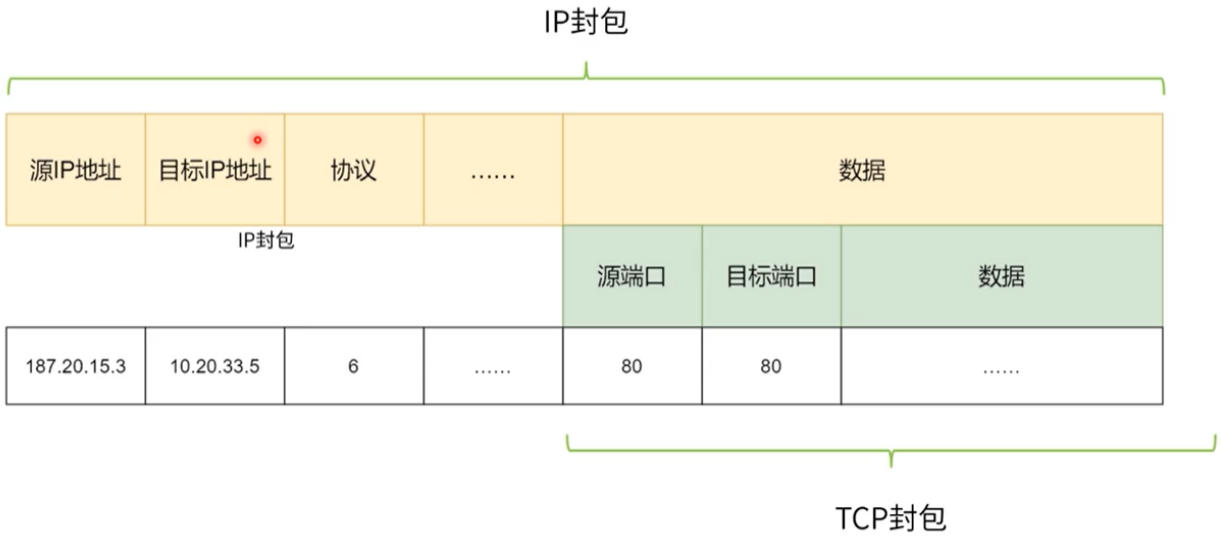
MAC编号：MAC编号是出厂的时候就存在，竟然MAC已经存在了，还要IP地址干什么呢?可以这么理解，你有一个身份证，身份证上有身份证编号，有居住地址。身份证编号可以去作为其他的作用，居住地址可以用来收物流。所以MAC编号是设备编号，IP地址是位置编号

TCP/IP封包



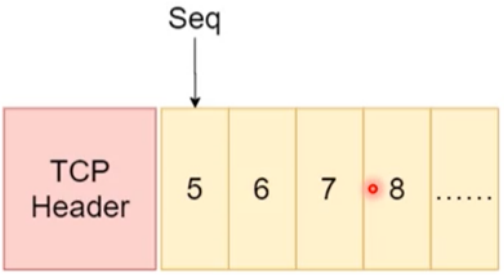
应用层的数据(程序员定义的格式)==》会进行一层数据转换。当转化成传输层的数据时候，TCP协议为例，TCP是的传输层协议，会加上一个Header头部，这个头部就是描述这次传输的数据，其中有一个重要的数据，就是源端口号和目标端口号==>网络层，添加一个header头部，作用同上。TCP数据和IP数据的差别，IP数据分割成更小的帧的小块了===》到链路层，添加header头部， 并且添加一个footer尾部，到达链路层数据都是二进制，这样更能保证数据的安全。

IP封包



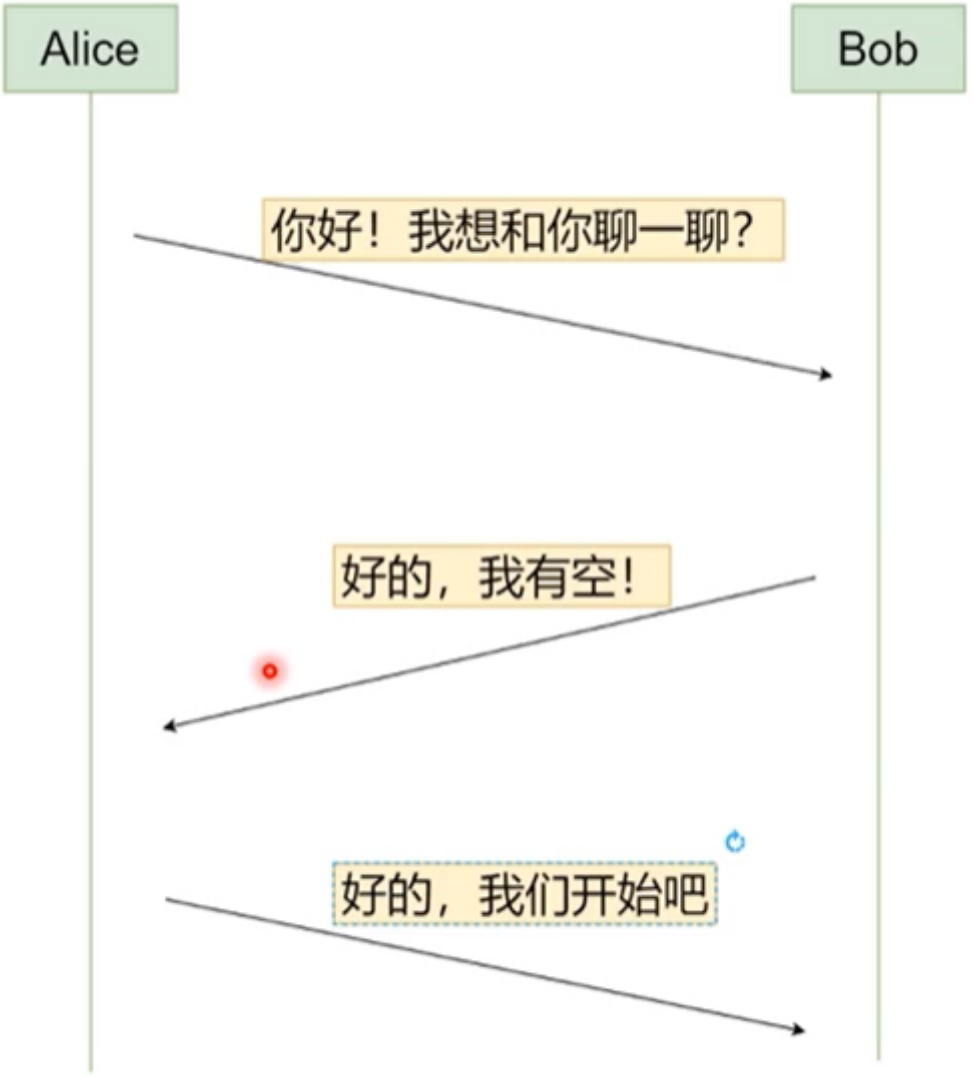
TCP协议封包

- 每个封包称作一个TCP信息段(TCP Segment)
- Header用于描述传输行为（如源端口，目标端口等）
- Header后面跟若干个byte数据，每个byte拥有自己的序列号

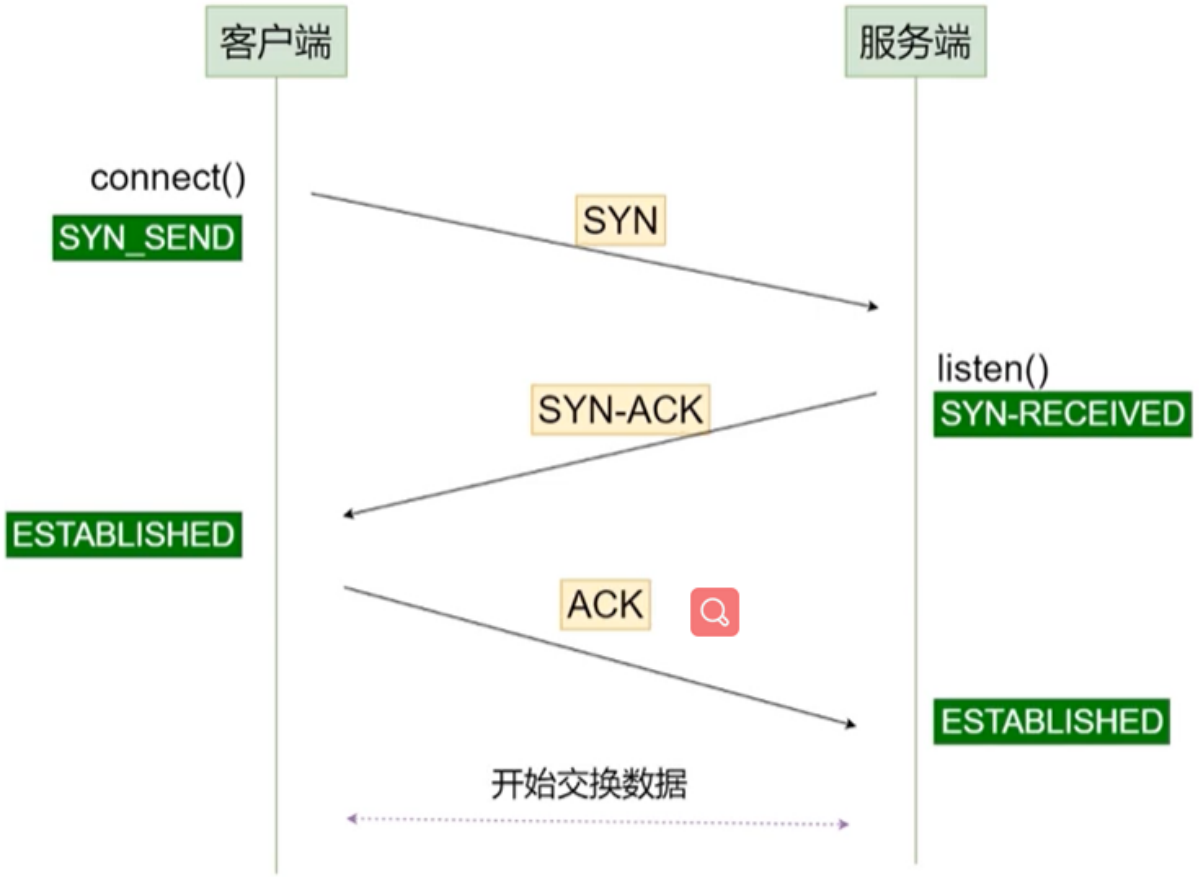


TCP/IP三次握手

为什么要建立三次握手？比如



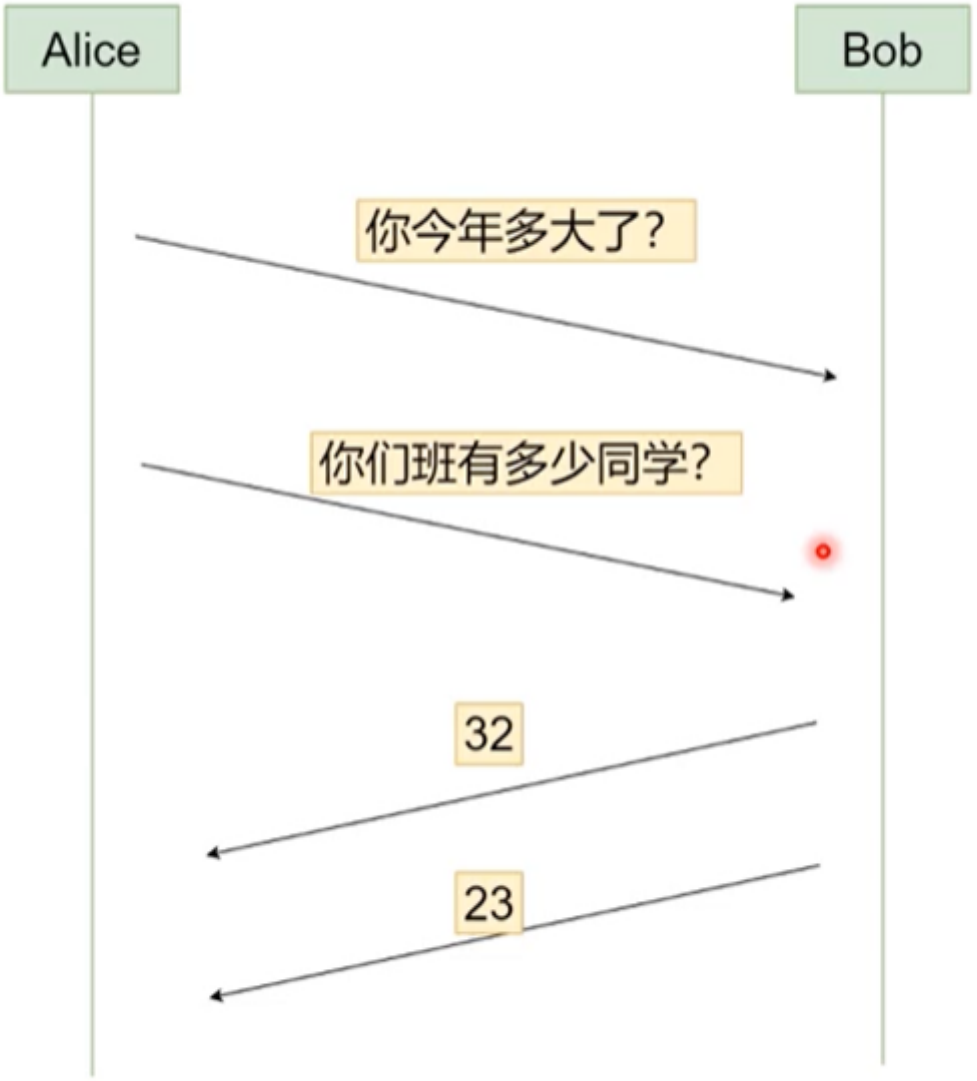
Alice和Bob聊天，如果Alice和Bob是有思想的人类，当的Aclie发一条’ 你好!我想和你聊聊 ‘给到Bob，Bob回复一句'好的，我有空'给到Alice，Alice即使走开，或者去厕所，不回复下面的那句’ 好的，我们开始吧 ‘给到Bob，人类也会达成一个共识，Bob会继续等等，他会猜想Alice有其他的事情去了。但是的如果Alice和Bob是机器的会话，一但是Alice即使走开，或者去厕所，不回复下面的那句’ 好的，我们开始吧 ‘，Bob机器会误以为Alice没有收到上一条’ 好的，我有空 ‘的消息，会重新发送。直到断开连接。所以在计算机中，需要更严谨的协议。需要以上三次握手。迁移到机器的三次握手如图所示



1. 客户端(可以是js java php....)通过connect()方法客户端的状态变成SYN_SEND，并发送消息SYN(请求同步)给到服务端,第一次握手
2. 服务端接收到SYN后，服务端会变成请求监听的状态，SYN-RECEIVED,同时发送SYN-ACK(请求同步已确认)消息给到客户端,第二次握手
3. 客户端收到消息之后，就变成ESTABLISHED连接状态，但是需要发送一个ACK确认给到服务端，服务端收到消息后次才变成ESTABLISHED连接状态。第三次握手。

TCP/IP协议保证了数据传输的可靠性，如果不需要保证的数据传输的可靠性，可以交给应用层UDP

传输和处理数据顺序



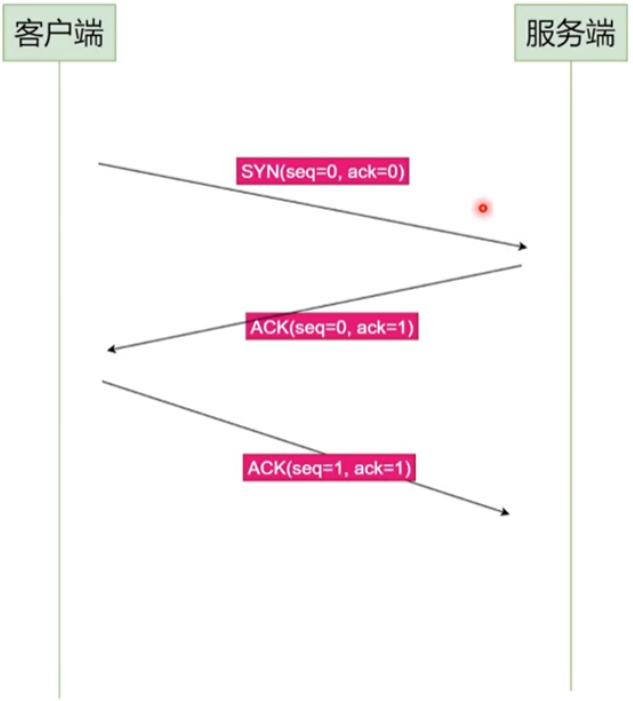
Alice发送了'你今年多大了'和'你们班有多少同学?'发送的两组消息，经过了复杂的网络路由，是一个网状结构，谁先到，谁后到无法确定。Bob发送的两句也无法确认谁先到谁后到。如何解决数据顺序问题?

TCP/IP协议的处理方法，消息的绝对顺序用SEQ,ACK这一对元组描述。

SEQ(sequence):这个消息发送前一共发送了多少字节

ACK(acknowledge):这个消息发送前一共收到多少字节

三次握手



三次握手seq,ack计算解析：

1、

- seq：client端第一次发送packet，即：first-way handshake。所以按照上面的准则，它的数据应该从第一个开始，也即是第0位开始，所以seq为0。
- ack：而server端之前并未发送过数据，所以期望的是server端回传时的packet的seq应该从第一个开始，即是第0位开始，所以ack为0。

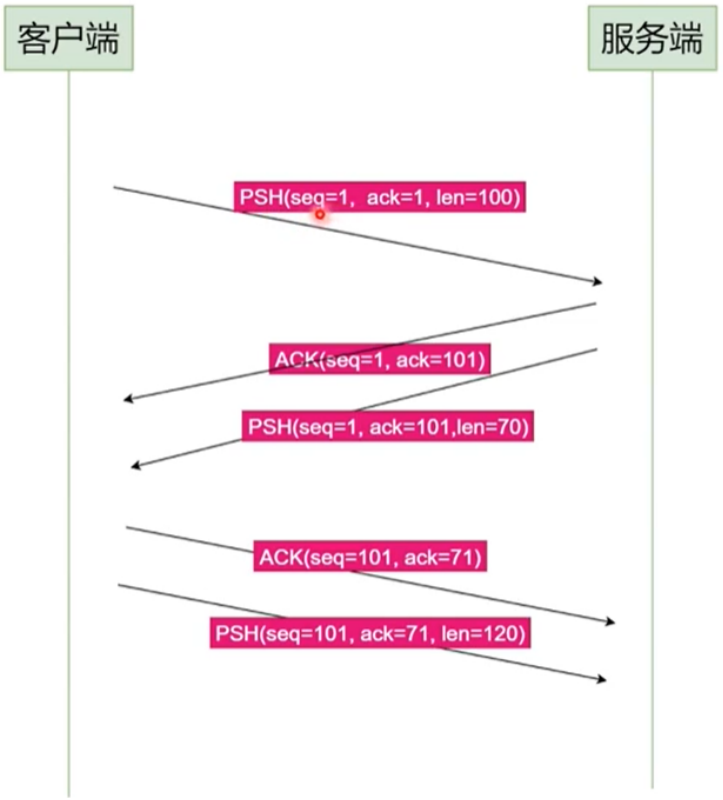
2、

- seq：server端第一次发送packet，即：second-way handshake。所以，这个packet的seq为0。
- ack：由于在【1】中接收到的是client端的SYN数据包，且它的seq为1，所以client端会让它自己的seq增加1。由此可预计（expect），client端的下一次packet传输时，它的seq是1（0增加1）。所以，ACK为1。

3、

- seq：third-way handshake。上一次发送时为【1】，【1】中seq为0且为SYN数据包，所以这一次的seq为1（0增加1）。
- ack：上次接收到时为【2】，【2】中seq为0，且为SYN数据包（虽然在flag上同时设定为SYN/ACK，但只要flag是SYN，就会驱使seq加一），所以可预计，server端下一次seq为1（0增加1）。

数据传送



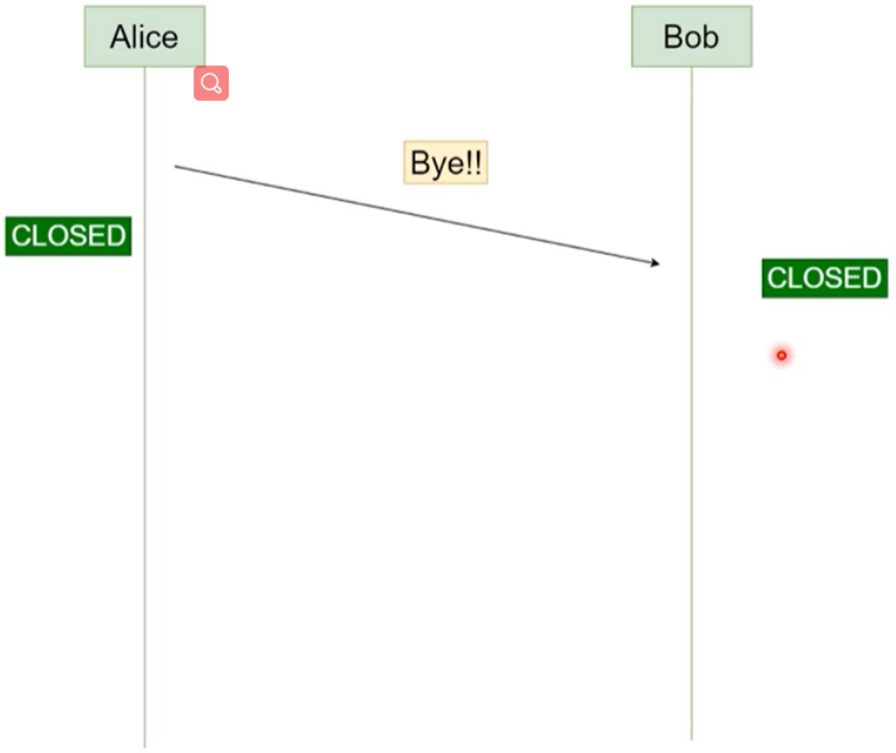
数据传送seq，ack解析

1. 三次握手后，seq，ack都为1，client端现在要PSH指令(也在头部)发送一个长度为100的数据
2. sever端 ack确认，seq1，ack已经收到长度为100的数据，ack为101，返回给client（第一组消息完毕)
3. server端发送消息前，seq为1，ack为101，PSH发送长度为70的数据到客户端
4. 客户端收到消息前，ack确认 seq为101，ack71(第二组消息完毕)
5. .....

以此类推，发送一个消息，一定有个ack 确认返回，如果发一个消息没有返回，TCP会认为这条消息没有送达，或者连接中断，或者连接异常的原因。通过这seq，ack解决数据的顺序问题。

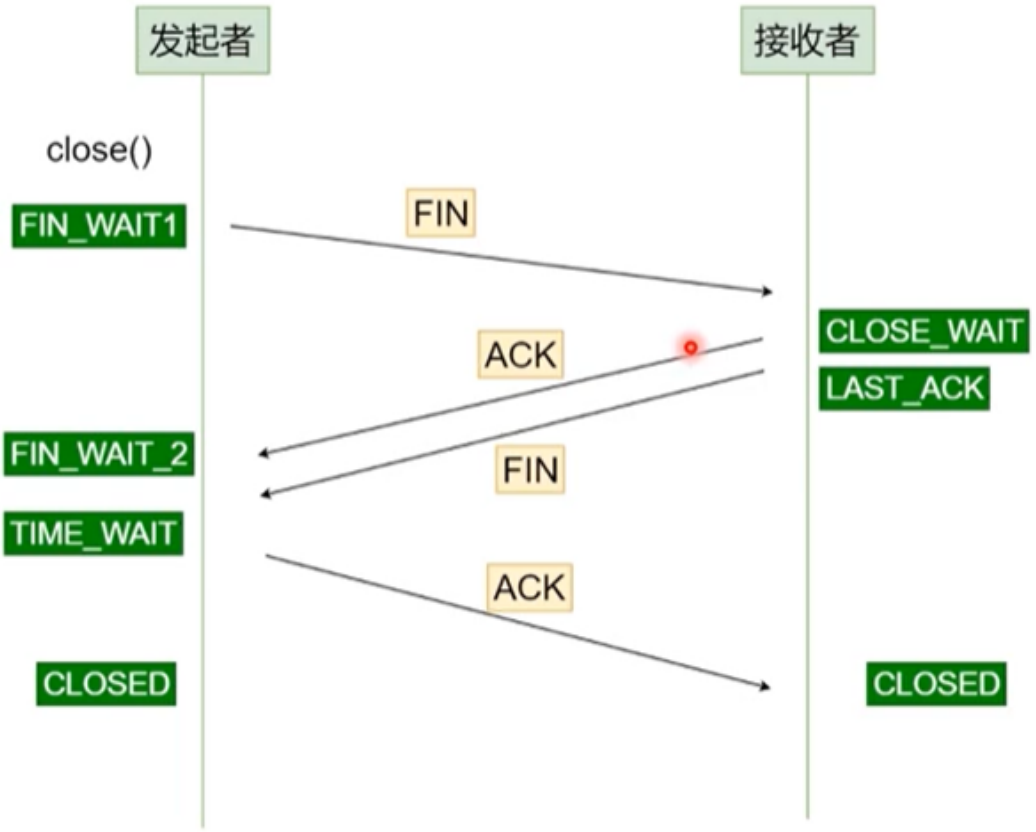
四次挥手(中断连接)

这样可不可以?



Alice说Bye后，Bob就不回消息了，这样可以吗？人与人之间有这样的context是可以的，但是机器是不可以的。因为是Alice发送bye的时候，Bob可能还在发消息，要保持整体连接中断，希望所有的消息都处理完毕了。这个叫稳定性。TCP协议是要保证可靠性，保证每发出去的消息，都能收到反馈。

完整的四次挥手



四次挥手解析:

1. 发起者发送FIN消息给接收者，接收者有两件事情要做，第一件ACK收到发送者的FIN消息(第一次，第二次挥手)
2. 第二件事情，接收者处理完自己的事情后，给发起者发送FIN消息(第三次挥手)
3. 发起者收到FIN消息后，ACK给接收者(第四次挥手)
4. TCP结束，关闭连接

总结和思考

- 最简化原则：没有足够的事情要做，就不必分层
- 思考计算机对话和人对话的区别？为什么需要三次握手？
- 网络中的顺序问题，TCP协议给了完美的解法，这个方法是可以迁移的。（学算法的价值）