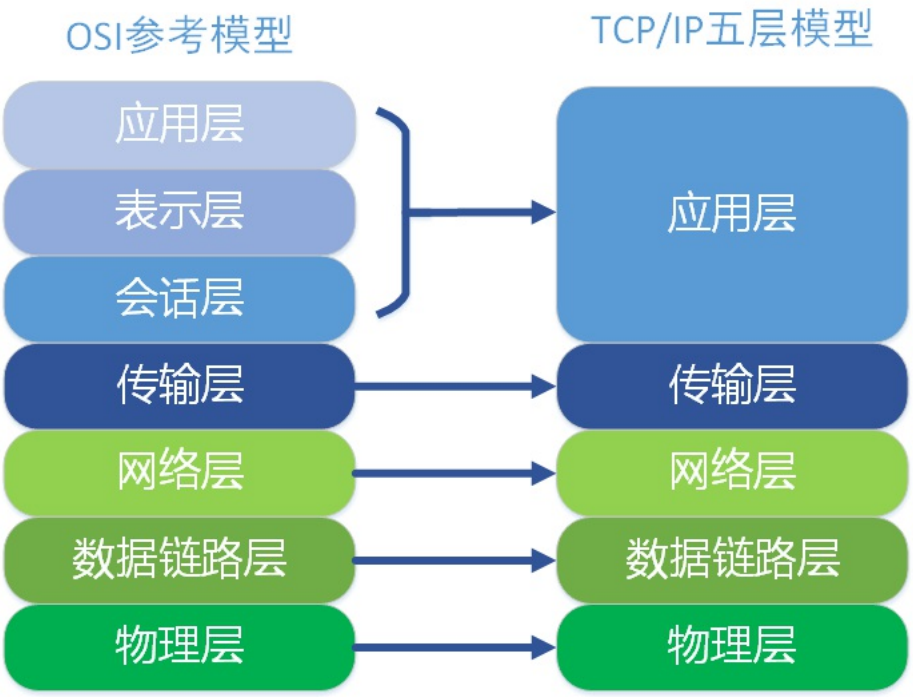
link: null
title: 珠峰架构师成长计划
description: OSI是Open System Interconnection的缩写，意为开放式系统互联。国际标准化组织（ISO）制定了OSI模型，该模型定义了不同计算机互联的标准，是设计和描述计算机网络通信的基本框架。OSI模型把网络通信的工作分为7层，分别是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。
首先来看看OSI的七层模型:
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=21 sentences=19, words=147

1. OSI七层网络模型

OSI是Open System Interconnection的缩写，意为开放式系统互联。国际标准化组织（ISO）制定了OSI模型，该模型定义了不同计算机互联的标准，是设计和描述计算机网络通信的基本框架。OSI模型把网络通信的工作分为7层，分别是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。 首先来看看OSI的七层模型:

2.TCP/IP 参考模型

TCP/IP是传输控制协议/网络互联协议的简称。早期的TCP/IP模型是一个四层结构，从下往上依次是网络接口层、互联网层、传输层和应用层。后来在使用过程中，借鉴OSI七层参考模型，将网络接口层划分为了物理层和数据链路层，形成五层结构。



3. 传输层

传输层是面向连接的、可靠的的进程到进程通信的协议。TCP提供全双工服务，即数据可在同一时间双向传播。TCP将若干个字节构成一个分组，此分组称为报文段(Segment)。提供了一种端到端的连接。 传输层的协议主要是TCP ， TCP(Transimision Control Protocal)是一种可靠的、面向连接的协议，传输效率低。

4. TCP格式

- 源端口号和目标端口号，计算机通过端口号识别访问哪个服务,比如http服务或ftp服务，发送方端口号是进行随机端口，目标端口号决定了接收方哪个程序来接收
- 32位序列号 TCP用序列号对数据包进行标记，以便在到达目的地后重新重装，假设当前的序列号为 s，发送数据长度为 l，则下次发送数据时的序列号为 s + l。在建立连接时通常由计算机生成一个随机数作为序列号的初始值
- 确认应答号 它等于下一次应该接收到的数据的序列号。假设发送端的序列号为 s，发送数据的长度为 l，那么接收端返回的确认应答号也是 s + l。发送端接收到这个确认应答后，可以认为这个位置以前所有的数据都已被正常接收。
- 首部长度：TCP 首部的长度，单位为 4 字节。如果没有可选字段，那么这里的值就是 5。表示 TCP 首部的长度为 20 字节。
- 控制位 TCP的连接、传输和断开都受这六个控制位的指挥
    - PSH(push急迫位) 缓存区将满，立刻传输速度
    - RST(reset重置位) 连接断了重新连接
    - URG(urgent紧急位) 紧急信号
    - ACK(acknowledgement 确认)为1表示确认号
    - SYN(synchronous建立联机) 同步序号位 TCP建立连接时要将这个值设为1
    - FIN发送端完成位，提出断开连接的一方把FIN置为1表示要断开连接
- 窗口值 说明本地可接收数据段的数目，这个值的大小是可变的。当网络通畅时将这个窗口值变大加快传输速度，当网络不稳定时减少这个值可以保证网络数据的可靠传输。它是来在TCP传输中进行流量控制的
- 窗口大小：用于表示从应答号开始能够接受多少个 8 位字节。如果窗口大小为 0，可以发送窗口探测。
- 效验和: 用来做差错控制，TCP校验和的计算包括TCP首部、数据和其它填充字节。在发送TCP数据段时，由发送端计算校验和，当到达目的地时又进行一次检验和计算。如果两次校验 和一致说明数据是正确的，否则 将认为数据被破坏，接收端将丢弃该数据
- 紧急指针：尽在 URG(urgent紧急) 控制位为 1 时有效。表示紧急数据的末尾在 TCP 数据部分中的位置。通常在暂时中断通信时使用（比如输入 Ctrl + C）。

5. 三次握手

TCP是面向连接的，无论哪一方向另一方发送数据之前，都必须先在双方之间建立一条连接。在TCP/IP协议中，TCP 协议提供可靠的连接服务，连接是通过三次握手进行初始化的。三次握手的目的是同步连接双方的序列号和确认号 并交换 TCP窗口大小信息。

为了方便描述我们将主动发起请求的172.16.17.94:8080 主机称为客户端，将返回数据的主机172.16.17.94:8080称为服务器，以下也是。

- 第一次握手: 建立连接。客户端发送连接请求，发送SYN报文，将seq设置为0。然后，客户端进入SYN_SEND状态，等待服务器的确认。
- 第二次握手: 服务器收到客户端的SYN报文段。需要对这个SYN报文段进行确认，发送ACK报文，将ack设置为1。同时，自己还要发送SYN请求信息，将seq为0。服务器端将上述所有信息一并发送给客户端，此时服务器进入SYN_RECV状态。

- 第三次握手: 客户端收到服务器的ACK和SYN报文后，进行确认，然后将ack设置为1，seq设置为1，向服务器发送ACK报文段，这个报文段发送完毕以后，客户端和服务器端都进入ESTABLISHED状态，完成TCP三次握手。**6. 数据传输#**」
- 客户端先向服务器发送数据，该数据报是长度为159的数据。
- 服务器收到报文后, 也向客户端发送了一个数据进行确认（ACK），并且返回客户端要请求的数据，数据的长度为111，将seq设置为1，ack设置为160（1 + 159）。
- 客户端收到服务器返回的数据后进行确认（ACK），将seq设置为160， ack设置为112（1 + 111）。

** 7. 四次挥手 #**

- 第一次挥手：客户端向服务器发送一个FIN报文段，将设置seq为160和ack为112，;此时，客户端进入 FIN_WAIT_1状态,这表示客户端没有数据要发送服务器了，请求关闭连接;
- 第二次挥手：服务器收到了客户端发送的FIN报文段，向客户端回一个ACK报文段，ack设置为1，seq设置为112;服务器进入了CLOSE_WAIT状态，客户端收到服务器返回的ACK报文后，进入FIN_WAIT_2状态;
- 第三次挥手：服务器会观察自己是否还有数据没有发送给客户端，如果有，先把数据发送给客户端，再发送FIN报文；如果没有，那么服务器直接发送FIN报文给客户端。请求关闭连接，同时服务器进入LAST_ACK状态;
- 第四次挥手：客户端收到服务器发送的FIN报文段，向服务器发送ACK报文段，将seq设置为161，将ack设置为113，然后客户端进入TIME_WAIT状态;服务器收到客户端的ACK报文段以后，就关闭连接;此时，客户端等待2MSL后依然没有收到回复，则证明Server端已正常关闭，客户端也可以关闭连接了。

> 注意：在握手和挥手时确认号应该是对方序列号加1,传输数据时则是对方序列号加上对方携带应用层数据的长度。

** 8. 问题 #**

1. 为什么需要三次握手? 确保双方收发都是正常的
2. 为什么需要四次挥手? 双方数据发送完毕，都认为可以断开
3. 为什么需要等待? A向B发的 FIN可能丢失
4. 为什么握手是三次，但挥手却是四次? 当Server端收到FIN报文时，很可能并不会立即关闭SOCKET

** 9.同学们的文章 #**

- 高春阳 (https://gcystar.github.io/2018/02/06/core/tcp%E7%9A%84%E8%AE%A4%E7%9F%A5%E5%92%8C%E5%BA%94%E7%94%A8/)
- whynotgonow (https://juejin.im/post/5a7c4ebaf265da4e81239431)
- 李斌 (https://juejin.im/post/5a7fea206fb9a06333151e99)