**周志垒，从网络到分布式（集群内高并发）、、、**



**+++++++++++++++++++高并发负载均衡：网络协议原理 +++++++++++++++++++++++++++++++**

# 一、前言

未来几天讲的是：高并发，负载均衡，高可用。

其实这个知识点适用于架构师也适用于大数据，为什么要聊这个事情，不要因为技术而技术。

不要因为有这个技术了就要研究这个技术，我觉得这个技术怎么怎么样。你更应该弄明白为什么要有这个技术，以及这个技术是什么样的原理，我怎么样去使用它。这样才是一个正确饿的学习方式。

中国网民基数大，互联网项目带来高并发问题。

高并发怎么去实现？网上对高并发的解释一抓一大把，怎么学？一切从最基本的基础讲起，什么是网络。因为在解决负载均衡、高并发的时候，其实有多种解决方案，最常规的有基于四层的 ，和基于七层的解决方案。Nginx是属于几层？ 最终自己要知道，4层里面可以做什么事情，7层里面可以做什么事情。

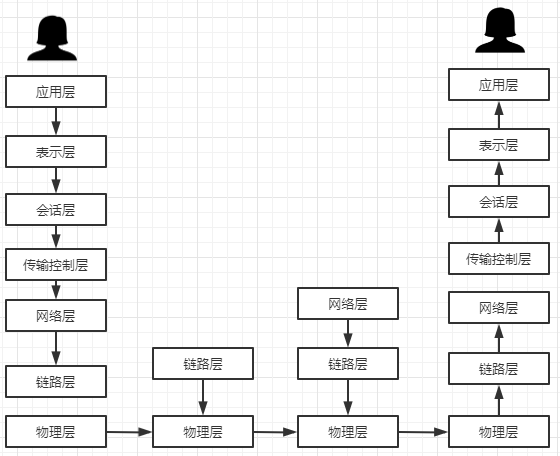
4层 7层，描述的是网络的术语。

# OSI七层参考模型、TCP/IP协议

## OSI七层参考模型

“参考模型”这个词，意思就是没有给出具体的层的定义，分了7个层，干七个事。7个层串起来就是个通信的过程。

OSI七层参考模型，哪七层呢？回归本质，本质是通信。大白话就是俩人一人拿一电脑，在这通信，通信可以写一软件从头到尾写完。你做的是软件工程学，是个工程，是有一定学问的，最重要的是 “分层解耦”



分层解耦，接口不变，每一层实现可以变



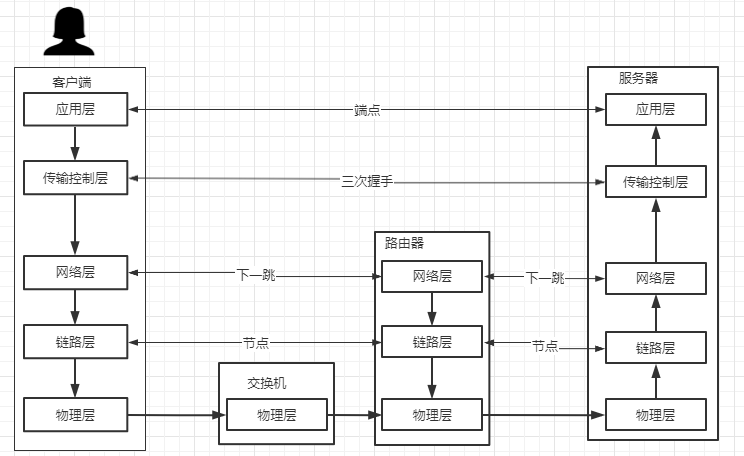
## TCP/IP协议tcp/ip五层或tcp/ip四层

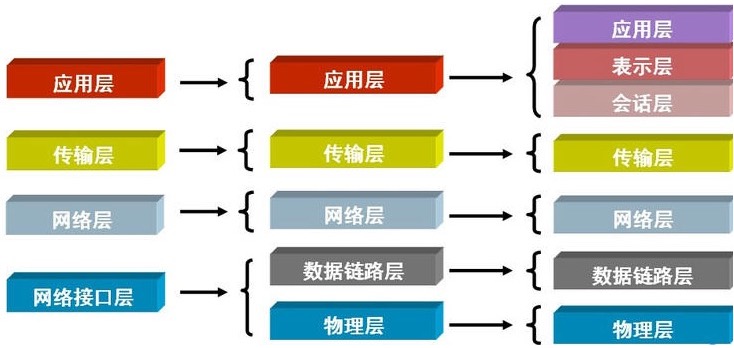
### 1定义：

相对于的参考模型，TCP/IP协议，协议：明确定义了双方怎么具体去实现。协议相对于参考模型就是一个具体的方案，具体的定义。所以接下来讨论的是TCP/IP精简完的4层或5层。

每一层都有自己的协议。

### 2图示





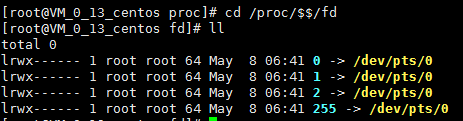
真正的网络这一块是从传输控制层往下走的（不带应用层），每一层有具体的协议。

应用层的协议有：http协议，ssh协议，smtp协议，ftp协议等。

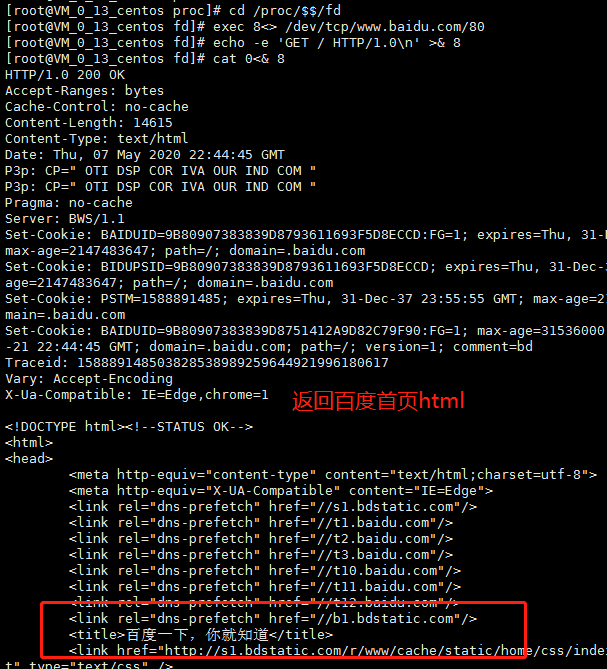
### 3实验演示http协议

实验：通过linux命令行模拟应用层的软件（浏览器）演示http协议







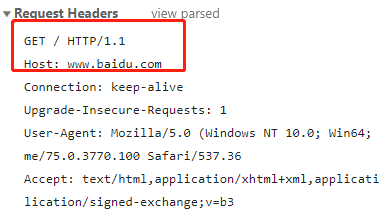


第一步：建立连接

第二步：传输数据（http协议：规范标准）

最终展现的是一个应用层协议

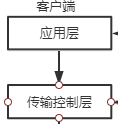
浏览器访问百度：http请求



http响应：



协议规范完的是数据，浏览器准备好数据最终是要发给对方的，不是浏览器把数据发出去的，浏览器只需要按http协议把数据规范好，OSI模型是分层的，接下来传输控制不归我管，浏览器（应用层）不管，它接着往下调用。



### 4应用层（http协议）

【应用层】访问百度过程：（命令行或者浏览器为例）

1. 浏览器先调用【传输控制层】，告诉他我想和百度传输数据
2. 传输控制层和百度建立连接（Socket）。然后我把请求头给百度，你在帮我传过去。对方再给我输送回来，我在渲染图形。

备注：访问百度之前，需要先和百度建立了一个Socket连接（就行java一样，第一步先new 一个Socket，拿到一个IO，才能把你的数据发出去）

每一层都有协议，在传输控制层就会涉及到Socket等概念了。

#### 用户态内核态解释

|  |
| --- |
| 为什么叫应用层：  这和计算机的一个术语有关系，计算机内存中分为两个区域：内核区-内核态，用户空间-用户态。浏览器、tomcat都是应用程序，传输控制层往下，都是内核的东西。  一个计算机为什么要有内核： 比方说你放着直播，登着QQ，很多程序都需要联网，这么多软件要是都控制网卡一定会控制乱。这些软件不需要关注怎么去控制网卡，怎么去传输数据包，他们只需要调用一下内核，内核里面有一套传输的标准，轮流会让他们把数据发出去或接收回来。  浏览器、Tomcat都是应用程序，都是应用层。传输控制层、网络层往下都是操作系统内核的东西了。  一个JVM说想建立连接的话，调用JVM所谓的Socket，new了一个对象，socket底层一定是native的，最终JVM是掉了操作系统内核的系统调用，等着操作系统跑完从  传输控制层----》网络层----》链路层-----》物理层 ---》链路层-----》网络层-----》控制层，这几层之后，建立连接之后，返回连接建立成功。之后才能走下一步代码，代码在new Socket这一步会出现小小的阻塞。 |

### 5 传输控制层

传输控制层是和某一个具体的应用无关的。

你的qq可以调传输控制层，你的远程登录可以调传输控制层，什么都可以调传输控制层。

每一层都有自己的协议，应用层是http协议，传输控制层必须知道的是

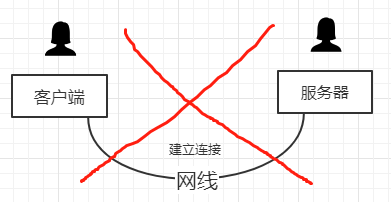
#### TCP协议

是一种面向连接的、可靠的、基于字节流的传输层通信协议

##### TCP的面向连接

###### 错误理解：

一个计算机想使用TCP去访问其它节点的时候，不是说你这边拿着一根网线Duang 顶上，Duang插到那边，给你一个屋里的连接么？不是这样的。



###### 正确理解：

有一首歌很火，确认过眼神 你是我想要的人。什么叫确认过眼神？有一美女在大马路上走着呢，你怎么知道她对你有感觉，你就能追到手呢？ ，一定是 你看她了一眼，她回看了你一眼，俩人眼睛一放电，她心里也这么想的。总之，**面向连接就是**：我发一个，你给我确认一个，我发你一个你给我确认一个。这个确认的过程，保证他它的连接是可靠的传输方式。

这只是说了什么是“面向连接”，但是还没设计到连接的概念，这时候再引入“三次握手”，因为“三次握手”里面要用到这个确认的过程。

##### TCP的连接

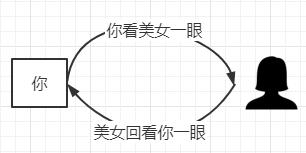
先不看七层模型中的其他层，只看传输控制层，这个传输控制，只讲TCP，如何制造出一个所谓的连接？这时候他有一个概念 三次握手 ，四次分手，以及加上他的确认机制。



###### TCP的三次握手：

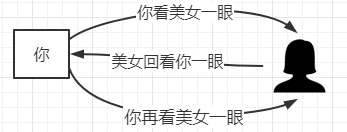
为什么需要三次握手才能建立一个连接？

比如说上层应用层 调用传输控制层，要你跟百度建立一连接，传输控制层创建一个数据包 sync，就是想握手，也就是你看了美女一眼，看完之后，美女喯看了你一眼。按理说这时候双方已经确认了，可以建立通信所谓的连接了。



###### 为什么是三次的？

你需要再回看美女一眼。



**站在方向层面：**

站在客户端一边，我给你发送想握手，你给我回SYNC+ACK确认，这个回送是一个确认，服务端确认你发的东西了。两次握手时，站在客户端，其实已经确认我的output和input都是OK的。



站在服务端，只能确定的是，接收到了客户端来握手的请求，也给客户端回了一个确认，但是回的这个确认，是不知道客户端是否接收到了的。所以需要客户端再回一个确认。

**站在IO层面：**

三趟数据包走完之后，双方才会在内存开辟线程，开辟对象，开辟所有的描述符，建立相应的资源。如果三次没完成，这些资源是不会开辟的。

#### UDP协议

 参考模型中一种无连接的[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "https://baike.baidu.com/item/_blank)协议，提供面向事务的简单不可靠信息传送服务

# 高并发负载均衡：LVS的DR,TUN,NAT模型推导

# 高并发负载均衡：LVS的DR模型试验搭建

# 高并发负载均衡：基于keepalived的LVS高可用搭建