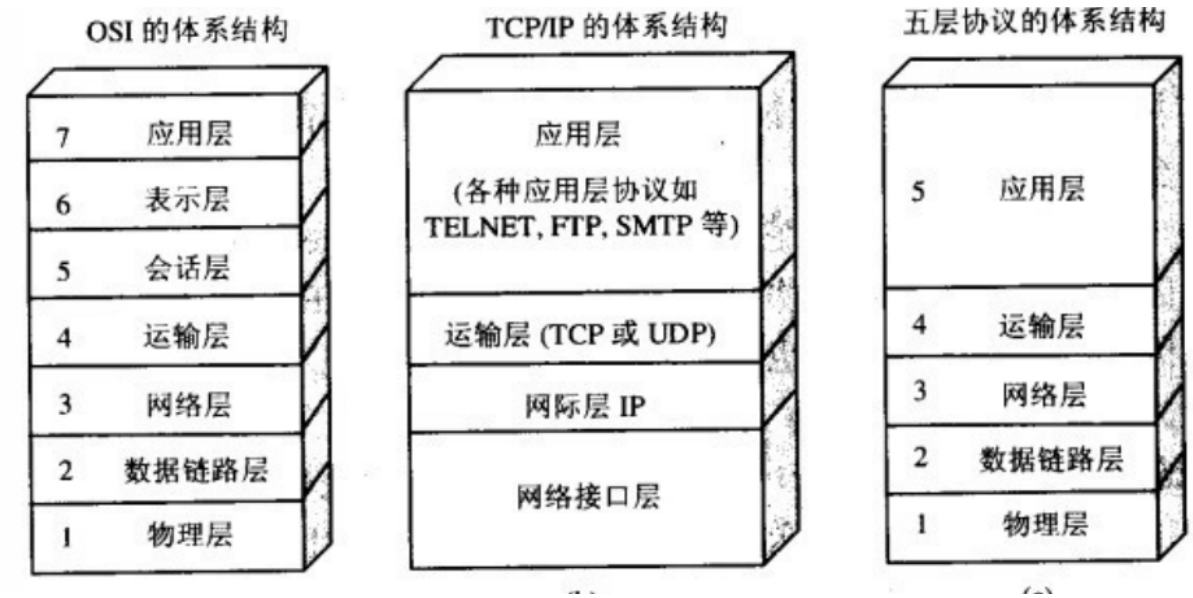
1.OSI 7层模型(5层协议体系)



应用层协议:

DNS,HTTP,HTTPS

传输层协议:

TCP-传输控制协议,面向连接的,可靠的数据传输服务

UDP-用户数据协议,无连接的,尽最大努力的传输(不可靠)

网络层:

IP:

数据链路层:

mac (ARP广播)

物理层

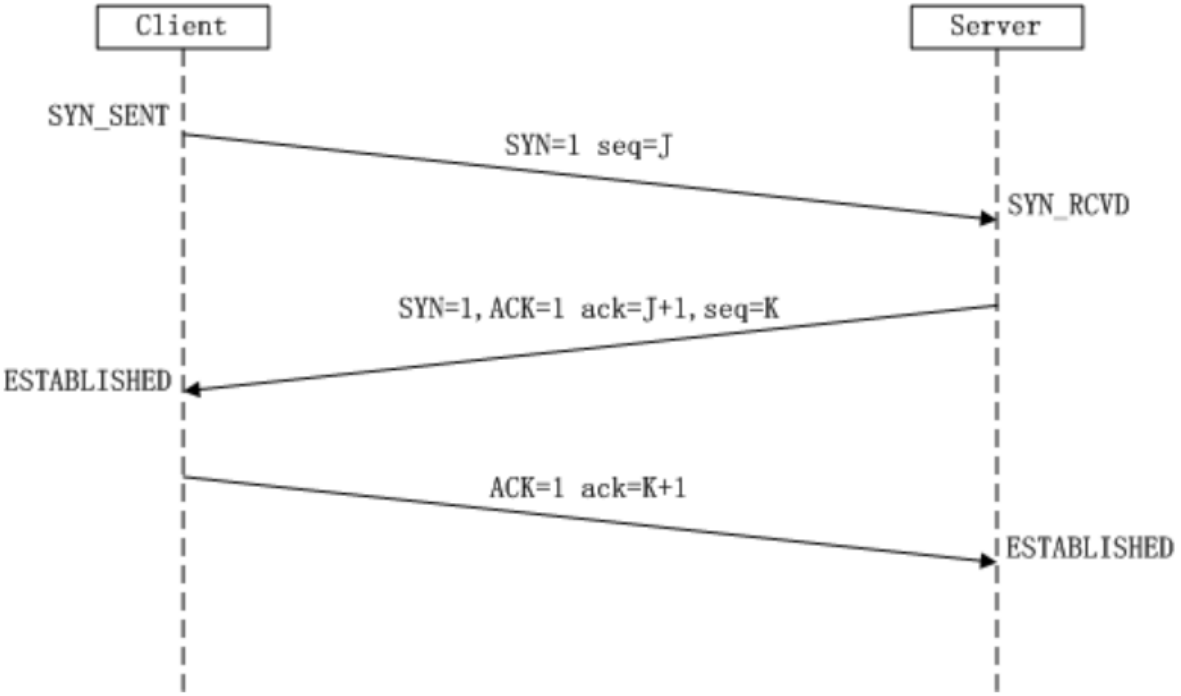
2.TCP的三次握手和四次挥手

2.1 三次握手

“我想给你发数据，可以吗？” （请提供序列号作为起始数据段）SYN：同步序列编号（Synchronize Sequence Numbers）

“可以，你什么时候发？” （已提供序列号）SYN+ACK应答

“我现在就发，你接着吧！” ACK消息响应



2.2 为什么需要三次握手

三次握手是需要发送方和接收方都要确认,自己和对方,接收和发送功能正常

第一次:client:什么都不能确认　　server:client发送正常,server接收正常

第二次:client:client接收,发送正常,server接收发送正常;

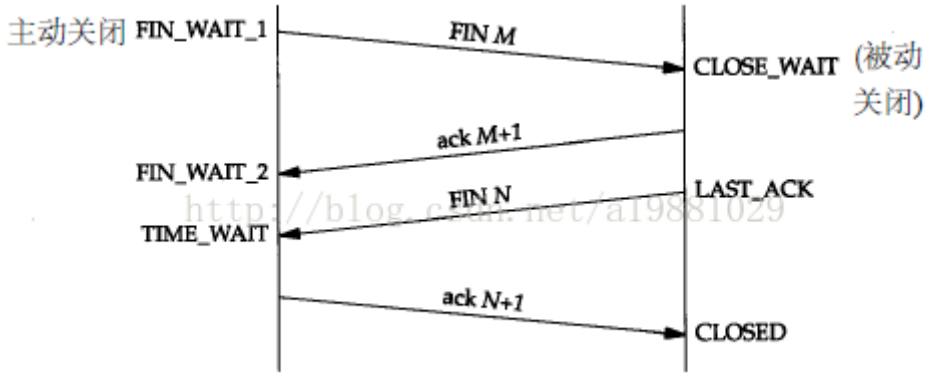
server:client发送正常,server接收正常

第三次:server:client接收,发送正常,server接收发送正常.

2.3 4次挥手

一般是server主动关闭

- 主动关闭连接的一方，调用close()；协议层发送FIN包
- 被动关闭的一方收到FIN包后，协议层回复ACK；然后被动关闭的一方，进入CLOSE_WAIT状态，主动关闭的一方等待对方关闭，则进入FIN_WAIT_2状态；此时，主动关闭的一方会等待被动关闭一方的应用程序调用close操作
- 被动关闭的一方在完成所有数据发送后，调用close()操作；此时，协议层发送FIN包给主动关闭的一方，等待对方的ACK，被动关闭的一方进入LAST_ACK状态；
- 主动关闭的一方收到FIN包，协议层回复ACK；此时，主动关闭连接的一方，进入TIME_WAIT状态；而被动关闭的一方，进入CLOSED状态
- 等待2MSL时间，主动关闭的一方，结束TIME_WAIT，进入CLOSED状态



2.4 TIME_WAIT问题

https://blog.csdn.net/fanren224/article/details/89849276

MSL 是Maximum Segment Lifetime英文的缩写，中文可以译为“报文最大生存时间”，他是任何报文在网络上存在的最长时间，

TIME_WAIT 存在是为了处理因为网络堵塞问题,主动方最后的ack可能未送达,处理被动方发来的FIN

占用内存,CPU和端口

解决方法:

通过优化相关的内核参数来解决这个问题

1.开启tw重用

2.开启tw回收 (更快速的回收)

1和2违背tcp原理,不推荐使用

1、客户端改用长连接

需要客户端的改动比较大，但能彻底解决问题，高并发的场景下，长连接的性能也明显好于短连接。

2、增加客户端的个数，避免在2MSL时间内使用到重复的端口

能够降低出问题概率，但需要增加成本，性价比不高。

3、降低net.ipv4.tcp_max_tw_buckets(有风险)

能够降低出问题概率，降低的程度视修改的参数值而定，设置为0可以完全解决。此方法在网络状况不好的情况下有风险，一般内网低延迟的网络风险不大。

4、客户端在断开连接时，不用quit的方式退出，直接发FIN或者RST

能够彻底解决问题，需要修改客户端底层库，有一定风险。

5、修改linux内核减小MSL时间

能够降低出问题的概率，需要修改linux内核，难度和风险都较大。

3.TCP如何保证可靠传输

1.大的应用数据被分割为多个数据块.并且编号保证顺序

2.首部校验和,如果校验和有问题,tcp会抛弃该包,并且不确认收到该包

3.丢弃重复数据包

4.流量控制

5.拥塞控制

6.超时重传

4.流量控制和拥塞控制

4.1 流量控制

TCP利用滑动窗口实现流量控制.

流量控制是为了控制发送方发送速率,保证接收方来得及接收.

接收方发送的确认报文中的窗口字可以用来控制发送窗口大小,进而影响发送方的发送速率.如果窗口字段为0,则发送方不能发送数据

4.2 拥塞控制

流量控制是点对点的控制,拥塞控制是一个全局性的控制

为了进行拥塞控制,TCP发送方要维持一个**拥塞窗口(cwnd)**.拥塞窗口的大小取决于网络的拥塞程度,并且动态变化.

方式：慢启动

拥塞避免(当出现拥塞的时候,重新进入慢启动,拥塞窗口重置)

解决方法：快速重传,快速恢复

5. ### 从URL输入到页面展现发生什么

1. 根据域名到DNS中找到IP
2. 根据IP建立TCP连接(三次握手)
3. 发起http请求
4. 服务器响应http请求
5. 浏览器解析HTML代码并请求html中的静态资源（js,css）

6. 关闭TCP连接（四次挥手）
7. 浏览器渲染页面