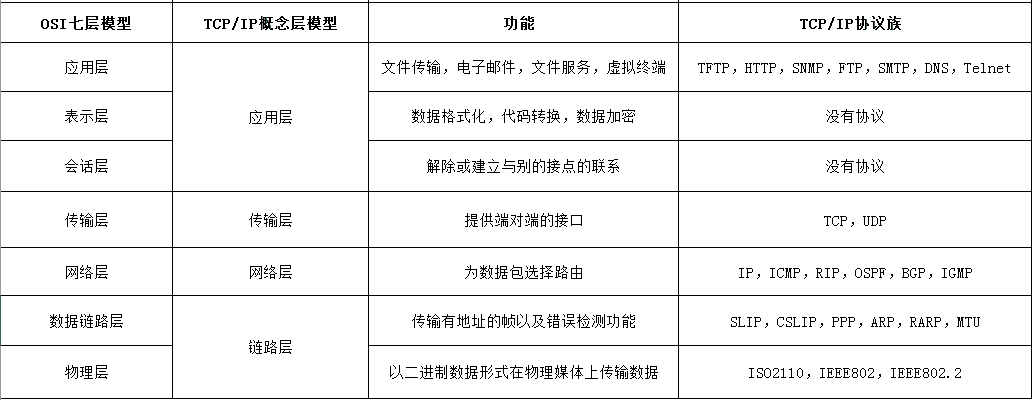
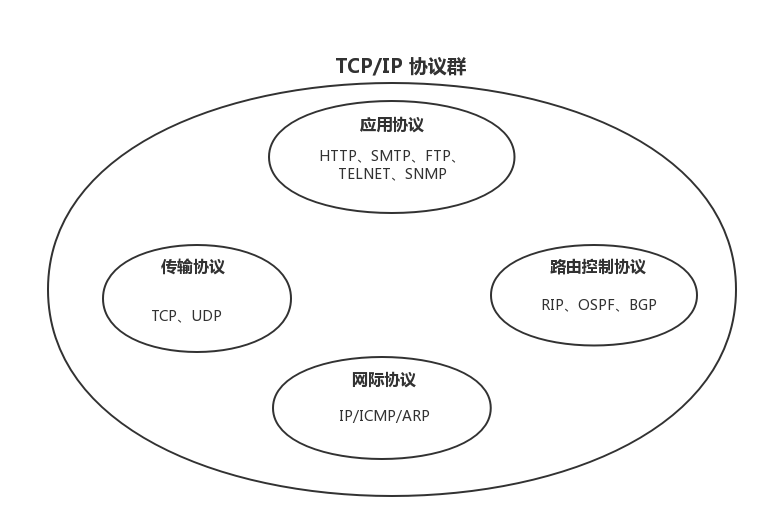
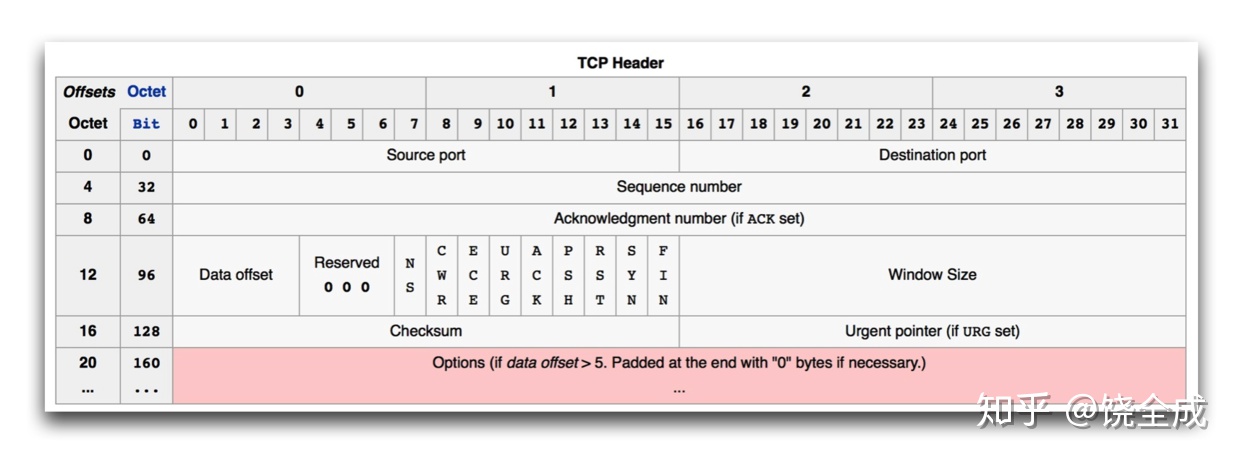
# 网络模型







# TCP VS UDP // todo

传输层协议

TCP

面向连接的、可靠的流协议。流是指不间断的数据结构。有重发机制

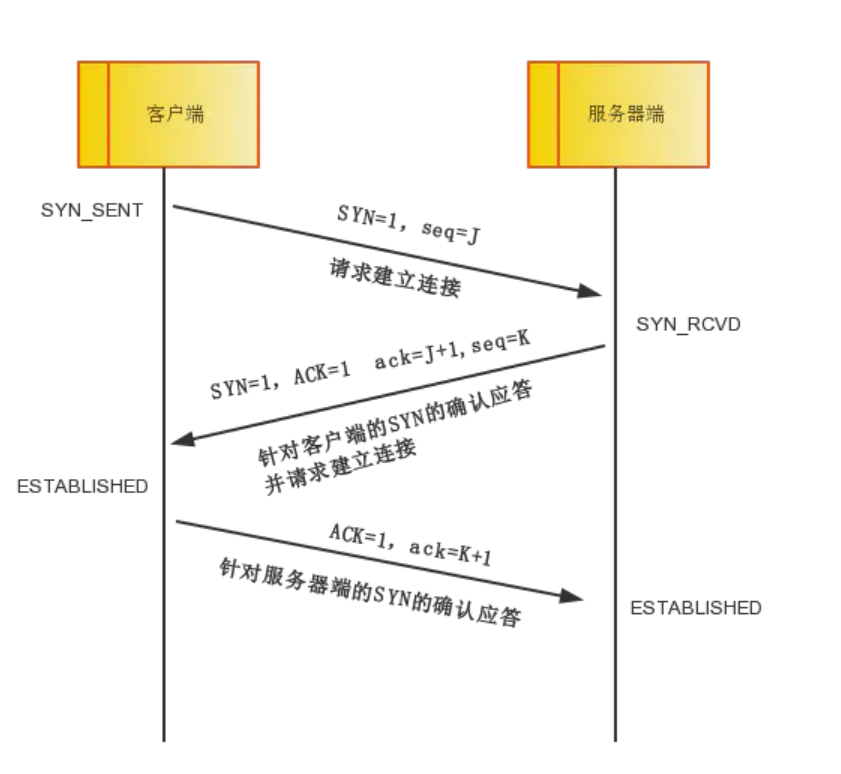
UDP

利用IP提供面向无连接的通信服务，不可靠的数据包协议，主要用于那些对高速传输和实时性有较高要求的通信或广播通信。视频/音频多媒体（即时通信）广播通信（广播多播）

# 三次握手 VS 四次挥手：

其实三次握手、四次挥手根本过程是一致的，都是双向的同步+确认，握手的时候同步的是ISN，即初始序列号，挥手时候同步关闭连接信息。只是建立连接时，server对client的ack和syn可以合为一次，但是关闭连接时，server收到client的fin包后，可能并不会立即关闭，还有需要发送的报文，那么就只能回复一个ack给client，等报文都发送完毕再发送fin给client。

建立一个TCP连接时，客户端和服务器端总共发送三个包以确认连接的建立。



建立连接需要

1. A—>B SYN my number X
2. A<—B ACK your number X+1
3. A<—B SYN my number Y
4. A—>B ACK your number Y+1

3 4合在一起就是三次握手

*如果A发送给B的包丢失，A会周期性超时重传，直到收到B的确认。*

*如果B发送的sync+ack没有到A，B会周期性超时重传直到收到A的确认。*

*如果A给B的ack没有到B，A发送完ACK会进入established状态，假定双发没有数据，那么B会周期性超时重传，直到收到A的确认，如果A有数据发送给B，B收到A的data+ACK，会自动切换到established状态。B不能发送数据。*

三次握手可以确定双方的初始序列号

TCP需要序列号来做可靠的传输，为了避免连接复用时无法分辨出是延迟或是旧的seq，需要三次握手双方约定确认初始序列号。

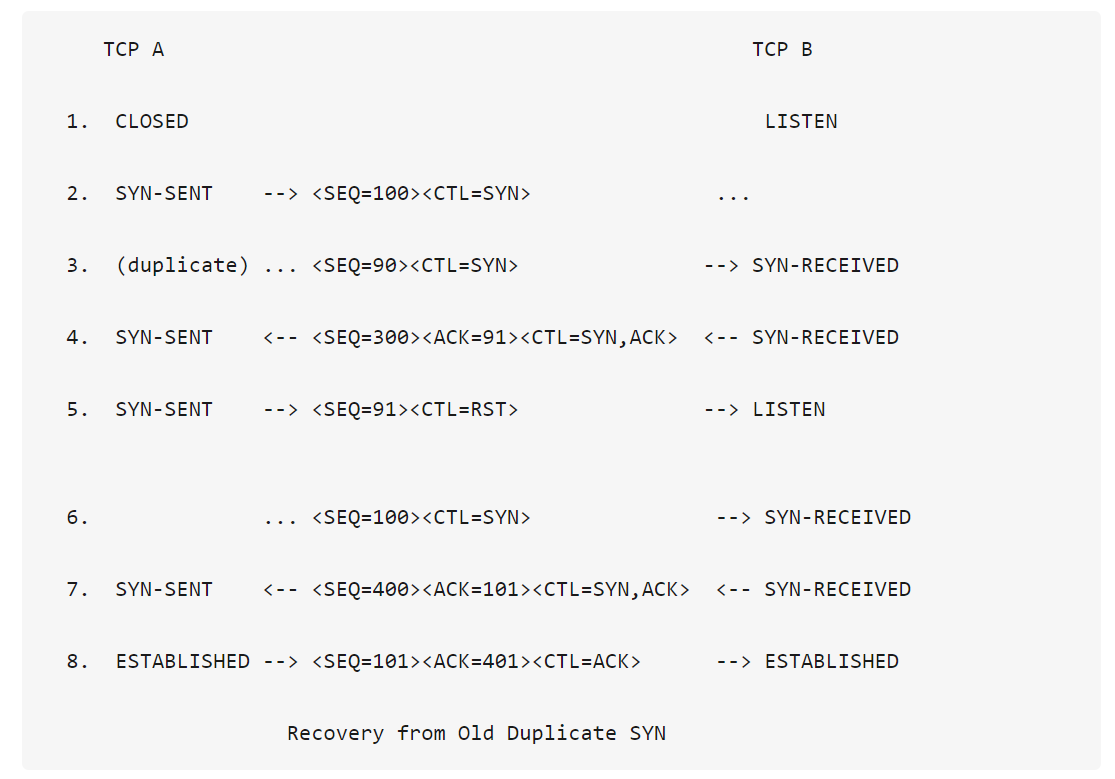
Tcp的设计中，发送的每一个包都会有一个seq number，确认机制是累加的，对一个seq X的确认意味着X序列号之前的包都是被确认接收到的。

新连接建立时，初始序列号会生成一个新的32为的ISN。双方都会有自己的ISN来做双方互发通信，每个包都会有sync+ack

*client 发出的第一个连接请求报文段并没有丢失，而是在某个网络结点长时间的滞留了，以致延误到连接释放以后的某个时间才到达 server。本来这是一个早已失效的报文段。但 server 收到此失效的连接请求报文段后，就误认为是 client 再次发出的一个新的连接请求。于是就向 client 发出确认报文段，同意建立连接。假设不采用 “三次握手”，那么只要 server 发出确认，新的连接就建立了。由于现在 client 并没有发出建立连接的请求，因此不会理睬 server 的确认，也不会向 server 发送数据。但 server 却以为新的运输连接已经建立，并一直等待 client 发来数据。这样，server 的很多资源就白白浪费掉了。采用 “三次握手” 的办法可以防止上述现象发生。例如刚才那种情况，client 不会向 server 的确认发出确认。server 由于收不到确认，就知道 client 并没有要求建立连接。”*

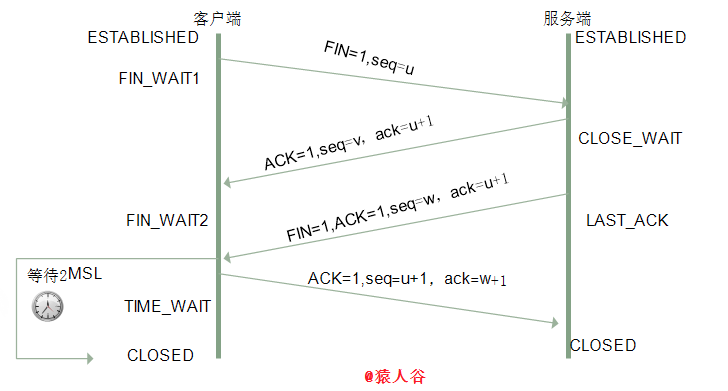
假设A的初始序列号是1000，以该序列号为起点，对要发送的每个字节的数据进行编号，B知道什么样子的编号是合法的，比如900就是非法的，同时B还可以对A的每一个编号的字节数据进行确认，如果A收到B的2001的确认，则意味着1000～2000共1000个字节已安全到达。

三次握手的原则设计是防止旧复用链接的初始化导致问题，为了解决此问题，我们设计了reset这个特别的控制信号来处理。



3中一个旧的sync到达B，B区分不出正常回包，A检测接收到的ack不正确，返回RST。B再次接受到seq=100的sync包，并ack，A收到正确的ack，连接建立。

# 四次挥手



客户端主动执行关闭并进入TIME\_WAIT是正常的，服务端通常执行被动关闭，不会进入TIME\_WAIT状态。

## Time\_wait：

2msl(max segment lifetime)等待。

msl：报文段最大生存时间。是任何报文段被丢弃前在网络内的最长时间。

用于重发最后一个ack，保证最后一个ack能到达server，从而都能够进入连接关闭状态。

当TCP执行一个主动关闭并发回最后一个ACK，该连接必须在TIME\_WAIT状态停留的时间为2倍的MSL，这样可以保证最后的ACK能够到达服务器。因为ack可能会丢失，从而导致处于last\_ack的server接收不到fin\_ack报文而无法进入正常的关闭状态，引入等待机制，那么服务器会超时重传，重新启动时间等待计时器，最后客户端和服务器都能正常关闭。

1st: 1 select 加S锁  
2nd: 2 select 加S锁  
3rn： 1 delete  
4th： 2 delete  
5th: 1 insert —— 产生锁等待  
6th: 2 insert 再commit后都能插入，这种情况是X锁和S锁  
  
t1 delete 没数据  
t2 delete 没数据  
t1 insert 锁等待  
t2 insert deadlock   这种情况有XGAP锁，才会有死锁  
  
所以不把查找放进session也会死锁，得把这块逻辑改一下



