1. tcp/ip的三次握手和四次挥手是什么概念，以及状态变化过程？

三次握手：所谓的三次握手，就是在建立 TCP 链接时，需要客户端和服务端总共发送 3 个包来确认连接的建立，在 socket 编程中，这个过程由客户端执行 connect 来触发。

第一次握手(SYN=1,seq=x)客户端发送一个TCP的SYN标志位置1的包，指明客户端打算连接的服务器的端口，以及初始序号X,保存在包头的序列号(SequenceNumber)字段里。发送完毕后，客户端进入SYN\_SEND状态。

第二次握手(SYN=1,ACK=1,seq=y,ACKnum=x+1):服务器发回确认包(ACK)应答。即SYN标志位和ACK标志位均为1。服务器端选择自己ISN序列号，放到Seq域里，同时将确认序号(AcknowledgementNumber)设置为客户的ISN加1，即X+1。发送完毕后，服务器端进入SYN\_RCVD状态。

第三次握手(ACK=1，ACKnum=y+1)客户端再次发送确认包(ACK)，SYN标志位为0，ACK标志位为1，并且把服务器发来ACK的序号字段+1，放在确定字段中发送给对方，并且在数据段放写ISN发完毕后，客户端进入ESTABLISHED状态，当服务器端接收到这个包时，也进入ESTABLISHED状态，TCP握手结束。

第一次挥手(FIN=1，seq=x) 假设客户端想要关闭连接，客户端发送一个 FIN 标志位置为 1 的包，表示自己已经没有数据 可以发送了，但是仍然可以接受数据。发送完毕后，客户端进入 FIN\_WAIT\_1 状态。

第二次挥手(ACK=1，ACKnum=x+1) 服务器端确认客户端的 FIN 包，发送一个确认包，表明自己接受到了客户端关闭连接的请求， 但还没有准备好关闭连接。发送完毕后，服务器端进入 CLOSE\_WAIT 状态，客户端接收到这 个确认包之后，进入 FIN\_WAIT\_2 状态，等待服务器端关闭连接。

第三次挥手(FIN=1，seq=w) 服务器端准备好关闭连接时，向客户端发送结束连接请求，FIN 置为 1。发送完毕后，服务器 端进入 LAST\_ACK 状态，等待来自客户端的最后一个 ACK。

第四次挥手(ACK=1，ACKnum=w+1) 客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入 TIME\_WAIT 状态，等待 可能出现的要求重传的 ACK 包。

服务器端接收到这个确认包之后，关闭连接，进入 CLOSED 状态。 客户端等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime） 之后，没有收到服务器端的 ACK，认为服务器端已经正常关闭连接，于是自己也关闭连接， 进入 CLOSED 状态。

1. 建立连接需要3次，为什么断开连接需要4次？

三次握手是因为因为当 Server 端收到 Client 端的 SYN 连接请求报文后，可以直接发送 SYN+ACK 报文。其中 ACK 报文是用来应答的，SYN 报文是用来同步的。但是关闭连接时， 当 Server 端收到 FIN 报文时，很可能并不会立即关闭 SOCKET（因为可能还有消息没处理 完），所以只能先回复一个 ACK 报文，告诉 Client 端，"你发的 FIN 报文我收到了"。只有等到 我 Server 端所有的报文都发送完了，我才能发送 FIN 报文，因此不能一起发送。故需要四步 握手。

1. 三次握手有哪些不安全性？

在三次握手过程中，Server发送SYN-ACK之后，收到Client的ACK之前的TCP连接称为半连接（half-openconnect），此时Server处于SYN\_RCVD状态，当收到ACK后，Server转入ESTABLISHED状态。SYN攻击就是Client在短时间内伪造大量不存在的IP地址，并向Server不断地发送SYN包，Server回复确认包，并等待Client的确认，由于源地址是不存在的，因此，Server需要不断重发直至超时，这些伪造的SYN包将产时间占用未连接队列，导致正常的SYN请求因为队列满而被丢弃，从而引起网络堵塞甚至系统瘫痪。SYN攻击时一种典型的DDOS攻击，检测SYN攻击的方式非常简单，即当Server上有大量半连接状态且源IP地址是随机的，则可以断定遭到SYN攻击了

1. TCP和UDP的区别？TCP是通过什么方式来保证可靠性的
2. TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接
3. TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付 Tcp通过校验和，重传控制，序号标识，滑动窗口、确认应答实现可靠传输。如丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。
4. UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

4.每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5、TCP对系统资源要求较多，UDP对系统资源要求较少。

5. tcp四层网络模型和osi七层网络模型分别是什么？以及每一层的作用

四层网络模型：应用层、传输层、网络层、数据链路层。

七层网络模型：应用层、表示层、会话层、传输层、网络层、数据链路层、物理层。

数据发送是，传输层在数据包前增加TCP头，网络层在TCP头前增加IP头，数据链路层在IP层前面增加MAC头；数据接收时，数据包拆分与分发。

1. 什么是滑动窗口协议？它的实现原理是什么？

滑动窗口是一种流量控制技术。在早期的网络通信中，通信双方不会考虑网络拥挤情况直接发送数据，导致发生了网络堵塞，谁也发送不了数据。

发送窗口：发送端可以不等待应答而连续发送的最大幀数称为发送窗口的尺寸。

接收窗口：接收方允许接收的幀的序号表，凡落在 接收窗口内的幀，接收方都必须处理，落在接收窗口 外的幀被丢弃。 接收方每次允许接收的幀数称为接收窗口的尺寸。

1. 服务器上TIME\_WAIT状态的连接过多，怎么解决？

1. 什么是NIO、BIO、AIO？他们的区别？

BIO：同步阻塞，获取连接请求只能等IO操作完成相应。

NIO：同步非阻塞，当请求IO操作无法完成时，异步去处理IO操作，一个线程轮训判断文件状态是否处理好，直到下次返回数据准备好，才可以去同步读取数据。

AIO：异步非阻塞，请求IO操作无法完成时，通过事件机制通知请求去处理IO操作。

1. 了解过多路复用吗？它是一个什么实现原理？

I/O 多路复用可以通过把多个 I/O 的阻塞复用到同一个 select 的阻塞上，从而使得系统在单 线程的情况下可以同时处理多个客户端请求。它的最大优势是系统开销小，并且不需要创建 新的进程或者线程，降低了系统的资源开销。

1. epool和select的区别是什么？

Select:进程可以通过把一个或者多个fd传递给select系统调用，进程会阻塞在select操作上，这样select可以帮我们检测多个fd是否处于就绪状态。Select有两个缺点，1 select需要去轮训所有的fd，当监听的fd越多，性能开销越大；2 select在单个线程能打开的fd是有限制的1024。

epoll：linux还提供了epoll的系统调用，epoll是基于事件驱动方式来代替顺序扫描，因此性能相对来说更高，主要原理是，当被监听的fd中，有fd就绪时，会告知当前进程具体哪一个fd就绪，那么当前进程只需要去从指定的fd上读取数据即可另外，epoll所能支持的fd上线是操作系统的最大文件句柄，这个数字要远远大于1024。