****TCP（Transmission Control Protocol，传输控制协议）****

是面向连接的协议，也就是说，在收发数据前，必须和对方建立可靠(数据无丢失、数据无失序、数据无错误、数据无重复到达)的连接。

TCP/IP协议三次握手与四次握手流程解析

1. TCP报文格式  
     TCP/IP协议的详细信息参看《TCP/IP协议详解》三卷本。下面是TCP报文格式图：

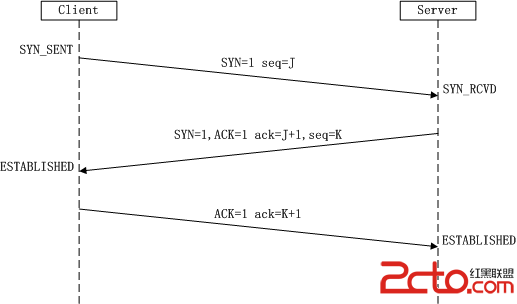


 上图中有几个字段需要重点介绍下：  
  （1）序号：seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。  
  （2）确认序号：ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。  
  （3）标志位：共6个，即URG、ACK、PSH、RST、SYN、FIN等，具体含义如下：  
  （A）URG：紧急指针（urgent pointer）有效。  
  （B）ACK：确认序号有效。  
  （C）PSH：接收方应该尽快将这个报文交给应用层。  
  （D）RST：重置连接。  
  （E）SYN： 同步序列号，发起一个新连接。  
  （F）FIN：释放一个连接。

**TCP的包头结构：**  
源端口 16位|目标端口 16位  
序列号 32位  
确认号 32位  
TCP首部长度 4位|保留 6位|标志 6位|窗口大小 16位  
偏移量(数据起始处的距离) 16位|校验和 16位  
选项  32位(可选)  
这样我们得出了TCP包头的最小长度，为**20字节**。（位=bit，字节=Byte，1Byte=8bit）

需要注意的是：  
  （A）不要将确认序号Ack与标志位中的ACK搞混了。  
  （B）确认方ack=发起方seq+1，两端配对。

1. 三次握手  
     所谓三次握手（Three-Way Handshake）即建立TCP连接，就是指建立一个TCP连接时，需要客户端和服务端总共发送3个包以确认连接的建立。在socket[编程](http://www.2cto.com/kf" \t "http://www.2cto.com/net/201310/_blank)中，这一过程由客户端执行connect来触发，整个流程如下图所示：

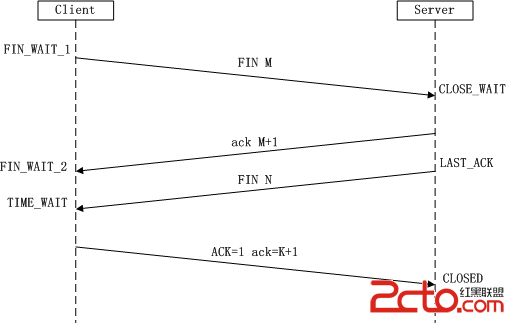


 （1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。  
  （2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。  
  （3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。

**三次握手过程通俗讲：**  
1 主机A通过向主机B 发送一个含有同步序列号的标志位的数据段给主机B ,向主机B 请求建立连接,通过这个数据段,  
主机A告诉主机B 两件事:我想要和你通信;你可以用哪个序列号作为起始数据段来回应我.  
2 主机B 收到主机A的请求后,用一个带有确认应答(ACK)和同步序列号(SYN)标志位的数据段响应主机A,也告诉主机A两件事:  
我已经收到你的请求了,你可以传输数据了;你要用哪佧序列号作为起始数据段来回应我  
3 主机A收到这个数据段后,再发送一个确认应答,确认已收到主机B 的数据段:"我已收到回复,我现在要开始传输实际数据了  
这样3次握手就完成了,主机A和主机B 就可以传输数据了.  
3次握手的特点  
没有应用层的数据  
SYN这个标志位只有在TCP建产连接时才会被置1  
握手完成后SYN标志位被置0

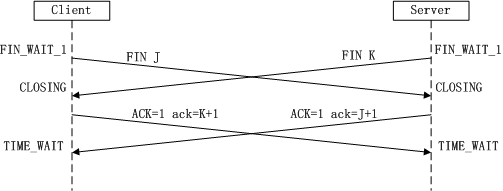
SYN攻击：  
  在三次握手过程中，Server发送SYN-ACK之后，收到Client的ACK之前的TCP连接称为半连接（half-open connect），此时Server处于SYN\_RCVD状态，当收到ACK后，Server转入ESTABLISHED状态。SYN攻击就是Client在短时间内伪造大量不存在的IP地址，并向Server不断地发送SYN包，Server回复确认包，并等待Client的确认，由于源地址是不存在的，因此，Server需要不断重发直至超时，这些伪造的SYN包将产时间占用未连接队列，导致正常的SYN请求因为队列满而被丢弃，从而引起网络堵塞甚至[系统](http://www.2cto.com/os/" \t "http://www.2cto.com/net/201310/_blank)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，使用如下命令可以让之现行：  
  #netstat -nap | grep SYN\_RECV

1. 四次挥手  
    三次握手耳熟能详，所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发，整个流程如下图所示：



  由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。  
 （1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。  
  （2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态。  
 （3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。  
  （4）第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

上面是一方主动关闭，另一方被动关闭的情况，实际中还会出现同时发起主动关闭的情况，具体流程如下图：



**【注意】中断连接端可以是Client端，也可以是Server端。**

假设Client端发起中断连接请求，也就是发送FIN报文。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据。所以你先发送ACK，"告诉Client端，你的请求我收到了，但是我还没准备好，请继续你等我的消息"。这个时候Client端就进入FIN\_WAIT状态，继续等待Server端的FIN报文。当Server端确定数据已发送完成，则向Client端发送FIN报文，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。Client端收到FIN报文后，"就知道可以关闭连接了，但是他还是不相信网络，怕Server端不知道要关闭，所以发送ACK后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。“，Server端收到ACK后，"就知道可以断开连接了"。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。Ok，TCP连接就这样关闭了！

1. **【注意】** 在TIME\_WAIT状态中，如果TCP client端最后一次发送的ACK丢失了，它将重新发送。TIME\_WAIT状态中所需要的时间是依赖于实现方法的。典型的值为30秒、1分钟和2分钟。等待之后连接正式关闭，并且所有的资源(包括端口号)都被释放。
2. 附注  
   **【问题1】**关于三次握手与四次挥手通常都会有典型的面试题  
     （1）三次握手是什么或者流程？四次握手呢？答案前面分析就是。  
     （2）为什么建立连接是三次握手，而关闭连接却是四次挥手呢？  
     这是因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。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，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送。

**【问题2】为什么TIME\_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？**

答：虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。

**【问题3】为什么不能用两次握手进行连接？**

三次握手”的目的是“为了防止已失效的连接请求报文段突然又传送到了服务端，因而产生错误”,即为了解决“网络中存在延迟的重复分组”

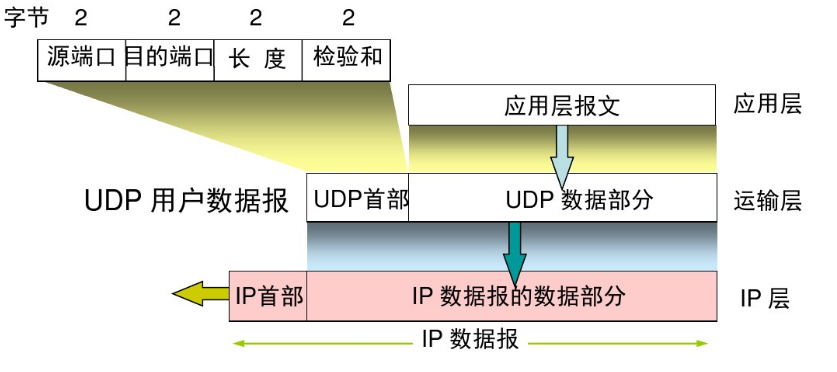
“已失效的连接请求报文段”的产生在这样一种情况下：client发出的第一个连接请求报文段并没有丢失，而是在某个网络结点长时间的滞留了，以致延误到连接释放以后的某个时间才到达server。本来这是一个早已失效的报文段。但server收到此失效的连接请求报文段后，就误认为是client再次发出的一个新的连接请求。于是就向client发出确认报文段，同意建立连接。假设不采用“三次握手”，那么只要server发出确认，新的连接就建立了。由于现在client并没有发出建立连接的请求，因此不会理睬server的确认，也不会向server发送ack包。但server却以为新的运输连接已经建立，并一直等待client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。

还有一种解释：现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机S和C之间的通信，假定C给S发送一个连接请求分组，S收到了这个分组，并发 送了确认应答分组。按照两次握手的协定，S认为连接已经成功地建立了，可以开始发送数据分组。可是，C在S的应答分组在传输中被丢失的情况下，将不知道S 是否已准备好，不知道S建立什么样的序列号，C甚至怀疑S是否收到自己的连接请求分组。在这种情况下，C认为连接还未建立成功，将忽略S发来的任何数据分 组，只等待连接确认应答分组。而S在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

**UDP（User Data Protocol，用户数据报协议）**

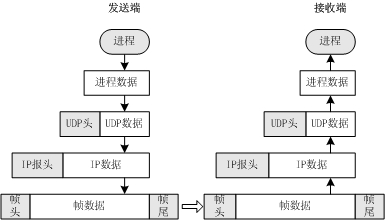
**端口**：UDP协议必须通过某个标志用以区分不同的程序所需要的数据包。端口号的功能就在于此，例如某一个UDP程序A在系统中注册了3000端口，那么，以后从外面传进来的目的端口号为3000的UDP包都会交给该程序。端口号理论上可以有2^16这么多。因为它的长度是16个bit。  
 UDP是一个非连接的协议，传输数据之前源端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP把每个消息段放在队列中，应用程序每次从队列中读一个消息段。  
（2） 由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一 台服务机可同时向多个客户机传输相同的消息。  
（3） UDP信息包的标题很短，只有8个字节，相对于TCP的20个字节信息包的额外开销很小。  
（4） 吞吐量不受拥挤控制算法的调节，只受应用软件生成数据的速率、传输带宽、源端和终端主机性能的限制。很适合多媒体通讯  
（5）UDP使用**尽最大努力交付，**即不保证可靠交付，因此主机不需要维持复杂的链接状态表（这里面有许多参数）。  
（6）UDP是**面向报文**的。发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给IP层。既不拆分，也不合并，而是保留这些报文的边界，应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文，接收方UDP对IP层交上来的UDP用户数据报，在去除首部后就原封不动的交给上层的应用进程，即以此交付一个完整的报文，因此，应用程序需要选择合适的报文大小。

1. UDP支持一对一，一对多，多对一，多对多的交互通信  
   我们经常使用“ping”命令来测试两台主机之间TCP/IP通信是否正常，其实“ping”命令的原理就是向对方主机发送UDP数据包，然后对方主机确认收到数据包，如果数据包是否到达的消息及时反馈回来，那么网络就是通的。



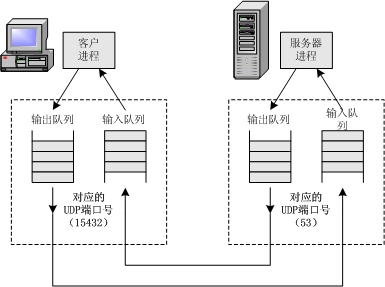
**UDP的包头结构：**  
源端口 16位  
目的端口 16位  
长度 16位，UDP整个大小，注意该长度不是报文的总长度，而只是UDP（包括UDP头和数 据部分）的总长度，最小为8个字节（仅为首部）  
校验和 16位：在进行检验和计算时，会添加一个**伪首部**一起进行运算。伪首部（占用12个字节）为：4个字节的源IP地址、4个字节的目的IP地址、1个字节的0、一个字节的数字17、以及占用2个字节UDP长度。这个伪首部不是报文的真正首部，只是引入为了计算校验和。相对于IP协议的只计算首部，UDP检验和会把首部和数据一起进行校验。接收端进行的校验和与UDP报文中的校验和相与，如果无差错应该全为1。如果有误，则将报文丢弃或者发给应用层、并附上差错警告

UDP用户数据报传输过程中的封装与拆封

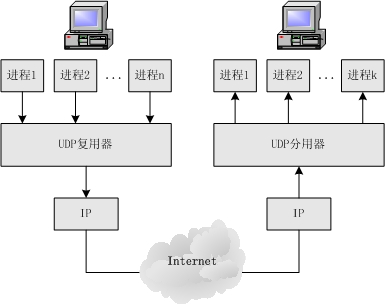


在TCP/IP协议层次模型中，UDP位于IP层之上。应用程序访问UDP层然后使用IP层传送数据报。**IP层的报头指明了源主机和目的主机地址**，而**UDP层的报头指明了主机上的源端口和目的端口。**

UDP报文传输队列



UDP的复用和分用



TCP/IP协议族中用端口号来标识进程；  
端口号是在0到65535之间的整数；  
客户程序随机选取的临时端口号；  
每一种服务器程序被分配了确定的全局一致的熟知端口号；  
每一个客户进程都知道相应的服务器进程的熟知端口号。  
  
**小结TCP与UDP的区别：**  
1.基于连接与无连接；  
2.对系统资源的要求（TCP较多，UDP少）；  
3.UDP程序结构较简单；  
4.流模式与数据报模式 ；  
5.TCP保证数据正确性，UDP可能丢包，TCP保证数据顺序，UDP不保证。

网络相关概念

网络的功能：

数据通信，资源共享（包括硬件资源和软件资源），负载均衡，高可靠性