**网络协议： TCP/IP 和UDP/IP**

## TCP/IP

TCP/IP（Transmission Control Protocol/Internet Protocol）是一种可靠的网络数据传输控制协议。定义了主机如何连入因特网以及数据如何在他们之间传输的标准。

TCP/IP协议参考模型把所有TCP/IP系列协议归类到四个抽象层中；

每一个抽象层建立在低一层提供的服务上，并且为高一层提供服务

|  |  |
| --- | --- |
| http://images.cnitblog.com/blog/349217/201312/05230830-04807bb739954461a8bfc7513707f253.jpg | ICMP:控制报文协议  IGMP:internet组管理协议  ARP:地址解析协议  RARP:反向地址转化协议 |

OSI模型（开放式系统互联通信参考模型），它是由国际标准化组织提出的，试图使各种计算机在世界范围内互联为网络的标准框架

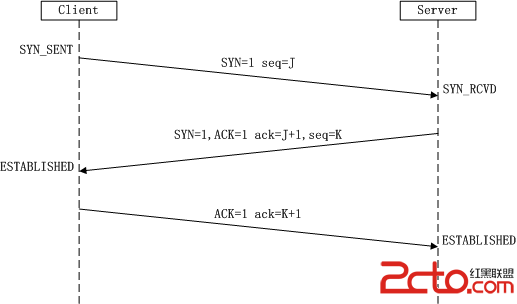
OSI模型多了表达层、会话层

### 3次握手协议

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

SYN：同步序列编号（Synchronize Sequence Numbers）,用于tcp/ip建立连接时的握手信号

ACK(acknowleggement): 确认字符，如果接收方接收到数据会回复一个ack字符



（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。

*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*

### 4次挥手协议

三次握手耳熟能详，四次挥手估计就听得比较少了，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

|  |  |
| --- | --- |
| http://www.2cto.com/uploadfile/2013/1022/20131022025350523.png | **单工**：数据传输只支持数据在一个方向上传输  **半双工**：数据传输允许数据在两个方向上传输，但是在某一时刻，只允许在一个方向上传输，实际上有点像切换方向的单工通信  **全双工**：数据通信允许数据同时在两个方向上传输，因此全双工是两个单工通信方式的结合，它要求发送设备和接收设备都有独立的接收和发送能力 |

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。

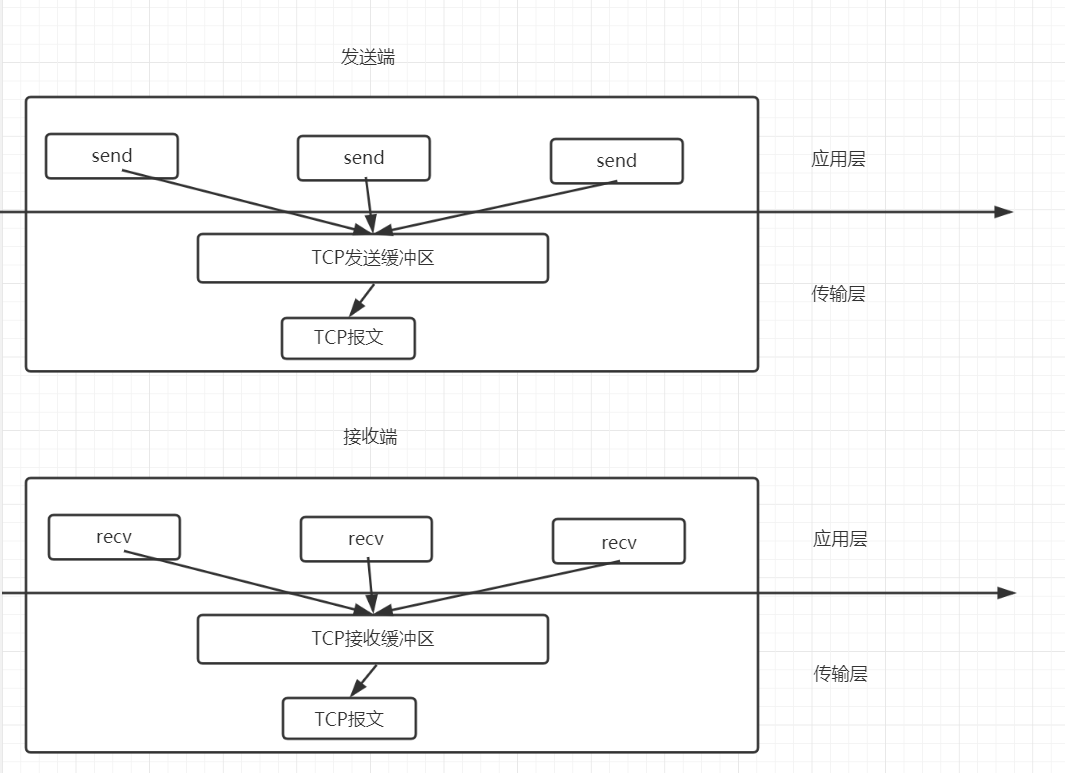
**TCP通信原理**

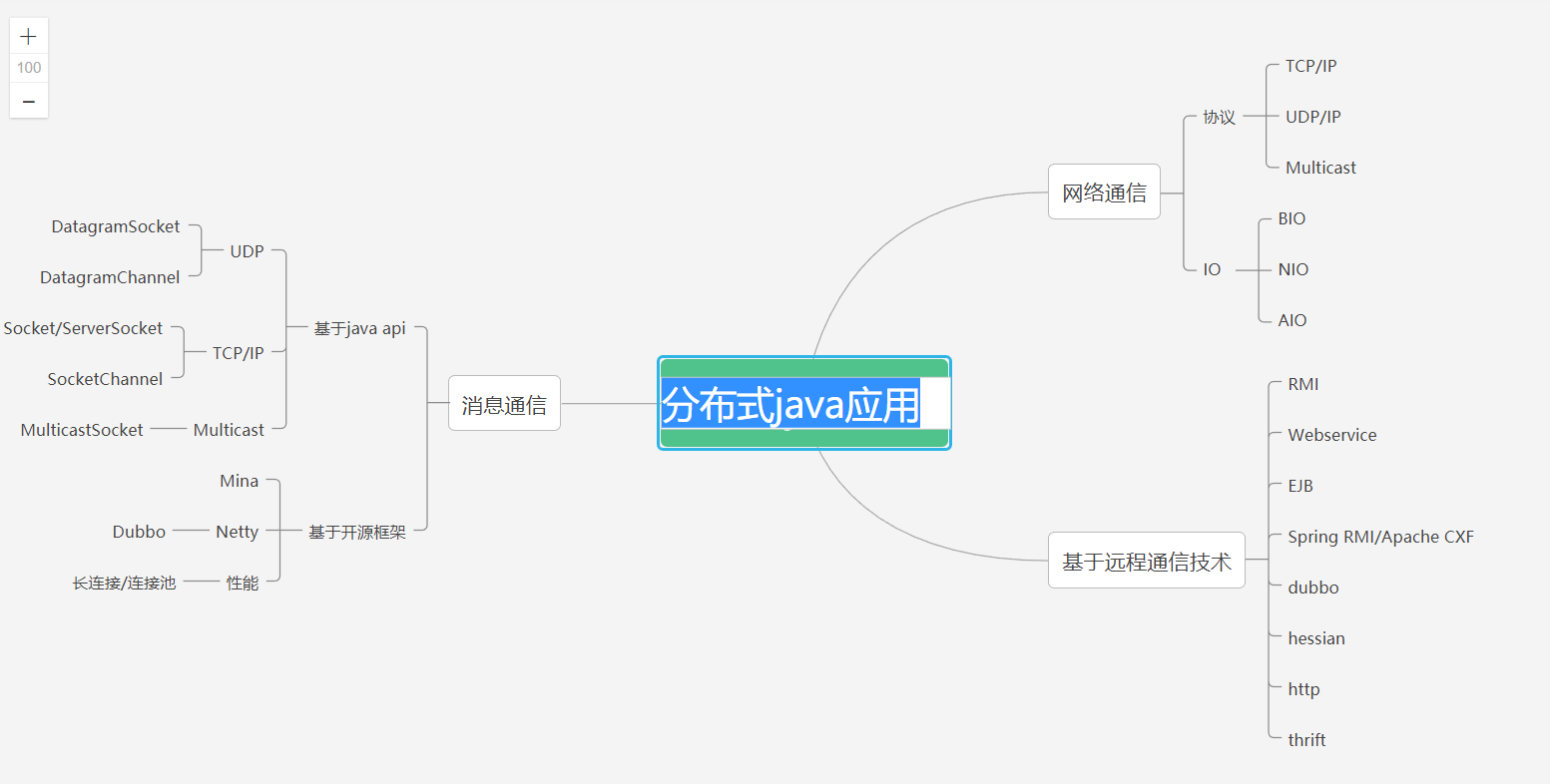
首先，对于TCP通信来说，每个TCP Socket的内核中都有一个发送缓冲区和一个接收缓冲区，TCP的全双工的工作模式及TCP的滑动窗口就是依赖于这两个独立的Buffer和该Buffer的填充状态。

接收缓冲区把数据缓存到内核，若应用进程一直没有调用Socket的read方法进行读取，那么该数据会一直被缓存在接收缓冲区内。不管进程是否读取Socket，对端发来的数据都会经过内核接收并缓存到Socket的内核接收缓冲区。

read索要做的工作，就是把内核接收缓冲区中的数据复制到应用层用户的Buffer里。

进程调用Socket的send发送数据的时候，一般情况下是讲数据从应用层用户的Buffer里复制到Socket的内核发送缓冲区，然后send就会在上层返回。换句话说，send返回时，数据不一定会被发送到对端。

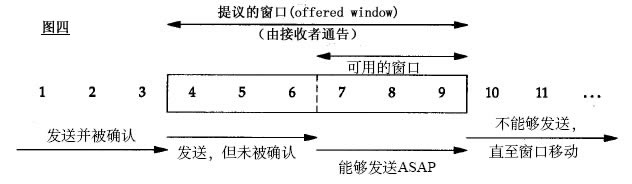




## 什么是滑动窗口协议

发送方和接收方都会维护一个数据帧的序列，这个序列被称作窗口。发送方的窗口大小由接收方确认，目的是控制发送速度，以免接收方的缓存不够大导致溢出，同时控制流量也可以避免网络拥塞。

下面图中的4,5,6号数据帧已经被发送出去，但是未收到关联的ACK，7,8,9帧则是等待发送。可以看出发送端的窗口大小为6，这是由接受端告知的（事实上必须考虑拥塞窗口cwnd，这里暂且考虑cwnd>rwnd）。此时如果发送端收到4号ACK，则窗口的左边缘向右收缩，窗口的右边缘则向右扩展，此时窗口就向前“滑动了”，即数据帧10也可以被发送



明白了Socket读写数据的底层原理，我们就很容易理解“阻塞模式”：对于读取Socket数据的过程而言，如果接收缓冲区为空，则调用Socket的read方法的线程会阻塞，知道有数据进入接收缓冲区；而对于写数据到Socket中的线程来说，如果待发送的数据长度大于发送缓冲区空余长度，则会阻塞在write方法上，等待发送缓冲区的报文被发送到网络上，然后继续发送下一段数据，循环上述过程直到数据都被写入到发送缓冲区为止

从前面分析的过程来看，传统的Socket阻塞模式直接导致每个Socket都必须绑定一个线程来操作数据，参与通信的任意一方如果处理数据的速度较慢，会直接拖累到另一方，导致另一方的线程不得不浪费大量的时间在I/O等待上，所以这就是Socket阻塞模式的“缺陷”。但是这种模式在少量的TCP连接通信的情况下，双方都可以快速的传输数据，这个时候的性能是最高的。

