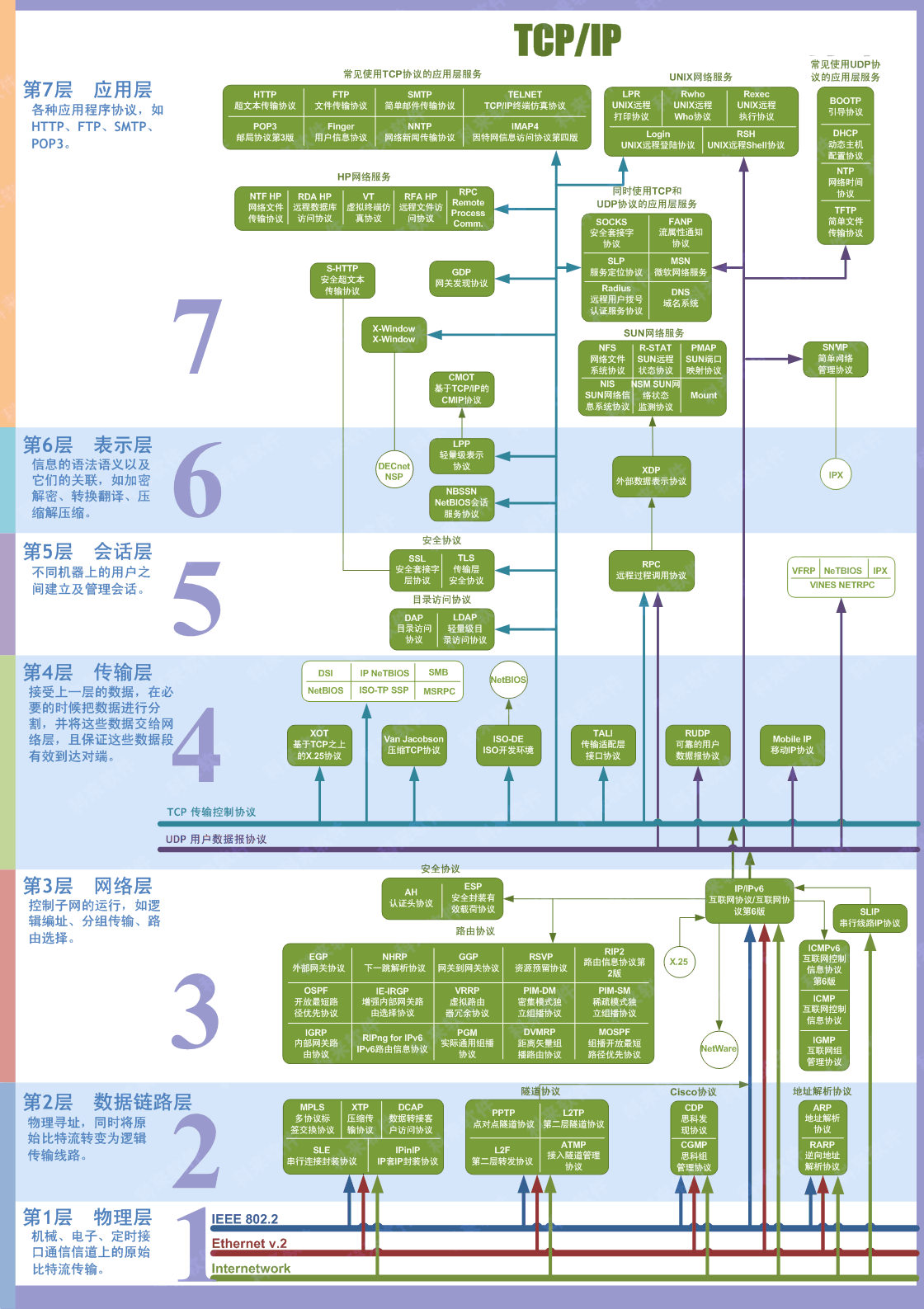
1. 网络知识
   1. [OSI七层模型详解](http://blog.csdn.net/yaopeng_2005/article/details/7064869)



OSI 七层模型通过七个层次化的结构模型使不同的系统不同的网络之间实现可靠的通讯，因此其最主要的功能就是帮助不同类型的主机实现数据传输 。

完成中继功能的节点通常称为中继系统。在OSI七层模型中，处于不同层的中继系统具有不同的名称。

一个设备工作在哪一层，关键看它工作时利用哪一层的数据头部信息。网桥工作时，是以MAC头部来决定转发端口的，因此显然它是数据链路层的设备。  
具体说:  
物理层：网卡，网线，集线器，中继器，调制解调器

数据链路层：网桥，交换机

网络层：路由器

网关工作在第四层传输层及其以上

集线器是物理层设备,采用广播的形式来传输信息。

交换机就是用来进行报文交换的机器。多为链路层设备(二层交换机)，能够进行地址学习，采用存储转发的形式来交换报文.。

路由器的一个作用是连通不同的网络，另一个作用是选择信息传送的线路。选择通畅快捷的近路，能大大提高通信速度，减轻网络系统通信负荷，节约网络系统资源，提高网络系统畅通率。

**交换机和路由器的区别**

交换机拥有一条很高带宽的背部总线和内部交换矩阵。交换机的所有的端口都挂接在这条总线上，控制电路收到数据包以后，处理端口会查找内存中的地址对照表以确定目的MAC（网卡的硬件地址）的NIC（网卡）挂接在哪个端口上，通过内部交换矩阵迅速将数据包传送到目的端口，目的MAC若不存在则广播到所有的端口，接收端口回应后交换机会“学习”新的地址，并把它添加入内部MAC地址表中。   
使用交换机也可以把网络“分段”，通过对照MAC地址表，交换机只允许必要的网络流量通过交换机。通过交换机的过滤和转发，可以有效的隔离广播风暴，减少误包和错包的出现，避免共享冲突。   
交换机在同一时刻可进行多个端口对之间的数据传输。每一端口都可视为独立的网段，连接在其上的网络设备独自享有全部的带宽，无须同其他设备竞争使用。当节点A向节点D发送数据时，节点B可同时向节点C发送数据，而且这两个传输都享有网络的全部带宽，都有着自己的虚拟连接。假使这里使用的是10Mbps的以太网交换机，那么该交换机这时的总流通量就等于2×10Mbps＝20Mbps，而使用10Mbps的共享式HUB时，一个HUB的总流通量也不会超出10Mbps。   
总之，交换机是一种基于MAC地址识别，能完成封装转发数据包功能的网络设备。交换机可以“学习”MAC地址，并把其存放在内部地址表中，通过在数据帧的始发者和目标接收者之间建立临时的交换路径，使数据帧直接由源地址到达目的地址。

从过滤网络流量的角度来看，路由器的作用与交换机和网桥非常相似。但是与工作在网络物理层，从物理上划分网段的交换机不同，路由器使用专门的软件协议从逻辑上对整个网络进行划分。例如，一台支持IP协议的路由器可以把网络划分成多个子网段，只有指向特殊IP地址的网络流量才可以通过路由器。对于每一个接收到的数据包，路由器都会重新计算其校验值，并写入新的物理地址。因此，使用路由器转发和过滤数据的速度往往要比只查看数据包物理地址的交换机慢。但是，对于那些结构复杂的网络，使用路由器可以提高网络的整体效率。路由器的另外一个明显优势就是可以自动过滤网络广播。

**集线器与路由器在功能上有什么不同?**  
首先说HUB,也就是集线器。它的作用可以简单的理解为将一些机器连接起来组成一个局域网。而交换机（又名交换式集线器）作用与集线器大体相同。但是两者在性能上有区别：集线器采用的式共享带宽的工作方式，而交换机是独享带宽。这样在机器很多或数据量很大时，两者将会有比较明显的。而路由器与以上两者有明显区别，它的作用在于连接不同的网段并且找到网络中数据传输最合适的路径。路由器是产生于交换机之后，就像交换机产生于集线器之后，所以路由器与交换机也有一定联系，不是完全独立的两种设备。路由器主要克服了交换机不能路由转发数据包的不足。

总的来说，路由器与交换机的主要区别体现在以下几个方面：   
（1）工作层次不同   
最初的的交换机是工作在数据链路层，而路由器一开始就设计工作在网络层。由于交换机工作在数据链路层，所以它的工作原理比较简单，而路由器工作在网络层，可以得到更多的协议信息，路由器可以做出更加智能的转发决策。   
  
（2）数据转发所依据的对象不同   
交换机是利用物理地址或者说MAC地址来确定转发数据的目的地址。而路由器则是利用IP地址来确定数据转发的地址。IP地址是在软件中实现的，描述的是设备所在的网络。MAC地址通常是硬件自带的，由网卡生产商来分配的，而且已经固化到了网卡中去，一般来说是不可更改的。而IP地址则通常由网络管理员或系统自动分配。   
  
（3）传统的交换机只能分割冲突域，不能分割广播域；而路由器可以分割广播域   
由交换机连接的网段仍属于同一个广播域，广播数据包会在交换机连接的所有网段上传播，在某些情况下会导致通信拥挤和安全漏洞。连接到路由器上的网段会被分配成不同的广播域，广播数据不会穿过路由器。虽然第三层以上交换机具有VLAN功能，也可以分割广播域，但是各子广播域之间是不能通信交流的，它们之间的交流仍然需要路由器。   
  
（4）路由器提供了防火墙的服务   
路由器仅仅转发特定地址的数据包，不传送不支持路由协议的数据包传送和未知目标网络数据包的传送，从而可以防止广播风暴。

**物理层**在OSI参考模型中，物理层（Physical Layer）是参考模型的最低层，也是OSI模型的第一层。  
物理层的主要功能是：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。  
物理层的作用是实现相邻计算机节点之间比特流的透明传送，尽可能屏蔽掉具体传输介质和物理设备的差异。使其上面的数据链路层不必考虑网络的具体传输介质是什么。“透明传送比特流”表示经实际电路传送后的比特流没有发生变化，对传送的比特流来说，这个电路好像是看不见的。

**数据链路层**数据链路层（Data Link Layer）是OSI模型的第二层，负责建立和管理节点间的链路。该层的主要功能是：通过各种控制协议，将有差错的物理信道变为无差错的、能可靠传输数据帧的数据链路。  
在计算机网络中由于各种干扰的存在，物理链路是不可靠的。因此，这一层的主要功能是在物理层提供的比特流的基础上，通过差错控制、流量控制方法，使有差错的物理线路变为无差错的数据链路，即提供可靠的通过物理介质传输数据的方法。  
该层通常又被分为介质访问控制（MAC）和逻辑链路控制（LLC）两个子层。

MAC子层的主要任务是解决共享型网络中多用户对信道竞争的问题，完成网络介质的访问控制；

LLC子层的主要任务是建立和维护网络连接，执行差错校验、流量控制和链路控制。  
数据链路层的具体工作是接收来自物理层的位流形式的数据，并封装成帧，传送到上一层；同样，也将来自上层的数据帧，拆装为位流形式的数据转发到物理层；并且，还负责处理接收端发回的确认帧的信息，以便提供可靠的数据传输。

**网络层**网络层（Network Layer）是OSI模型的第三层，它是OSI参考模型中最复杂的一层，也是通信子网的最高一层。它在下两层的基础上向资源子网提供服务。其主要任务是：通过路由选择算法，为报文或分组通过通信子网选择最适当的路径。该层控制数据链路层与传输层之间的信息转发，建立、维持和终止网络的连接。具体地说，数据链路层的数据在这一层被转换为数据包，然后通过路径选择、分段组合、顺序、进/出路由等控制，将信息从一个网络设备传送到另一个网络设备。  
一般地，数据链路层是解决同一网络内节点之间的通信，而网络层主要解决不同子网间的通信。例如在广域网之间通信时，必然会遇到路由（即两节点间可能有多条路径）选择问题。

在实现网络层功能时，需要解决的主要问题如下：  
 寻址：数据链路层中使用的物理地址（如MAC地址）仅解决网络内部的寻址问题。在不同子网之间通信时，为了识别和找到网络中的设备，每一子网中的设备都会被分配一个唯一的地址。由于各子网使用的物理技术可能不同，因此这个地址应当是逻辑地址（如IP地址）。  
 交换：规定不同的信息交换方式。常见的交换技术有：线路交换技术和存储转发技术，后者又包括报文交换技术和分组交换技术。  
 路由算法：当源节点和目的节点之间存在多条路径时，本层可以根据路由算法，通过网络为数据分组选择最佳路径，并将信息从最合适的路径由发送端传送到接收端。  
 连接服务：与数据链路层流量控制不同的是，前者控制的是网络相邻节点间的流量，后者控制的是从源节点到目的节点间的流量。其目的在于防止阻塞，并进行差错检测。  
  
**传输层**OSI下3层的主要任务是数据通信，上3层的任务是数据处理。而传输层（Transport Layer）是OSI模型的第4层。因此该层是通信子网和资源子网的接口和桥梁，起到承上启下的作用。  
该层的主要任务是：向用户提供可靠的端到端的差错和流量控制，保证报文的正确传输。传输层的作用是向高层屏蔽下层数据通信的细节，即向用户透明地传送报文。该层常见的协议：TCP/IP中的TCP协议、Novell网络中的SPX协议和微软的NetBIOS/NetBEUI协议。  
传输层提供会话层和网络层之间的传输服务，这种服务从会话层获得数据，并在必要时，对数据进行分割。然后，传输层将数据传递到网络层，并确保数据能正确无误地传送到网络层。因此，传输层负责提供两节点之间数据的可靠传送，当两节点的联系确定之后，传输层则负责监督工作。综上，传输层的主要功能如下：  
传输连接管理：提供建立、维护和拆除传输连接的功能。传输层在网络层的基础上为高层提供“面向连接”和“面向无接连”的两种服务。  
处理传输差错：提供可靠的“面向连接”和不太可靠的“面向无连接”的数据传输服务、差错控制和流量控制。在提供“面向连接”服务时，通过这一层传输的数据将由目标设备确认，如果在指定的时间内未收到确认信息，数据将被重发。  
监控服务质量。  
**会话层**会话层（Session Layer）是OSI模型的第5层，是用户应用程序和网络之间的接口，主要任务是：向两个实体的表示层提供建立和使用连接的方法。将不同实体之间的表示层的连接称为会话。因此会话层的任务就是组织和协调两个会话进程之间的通信，并对数据交换进行管理。  
用户可以按照半双工、单工和全双工的方式建立会话。当建立会话时，用户必须提供他们想要连接的远程地址。而这些地址与MAC（介质访问控制子层）地址或网络层的逻辑地址不同，它们是为用户专门设计的，更便于用户记忆。域名（DN）就是一种网络上使用的远程地址例如：[www.3721.com](http://student.csdn.net/link.php?url=http://www.3721.com%2F)就是一个域名。会话层的具体功能如下：  
会话管理：允许用户在两个实体设备之间建立、维持和终止会话，并支持它们之间的数据交换。例如提供单方向会话或双向同时会话，并管理会话中的发送顺序，以及会话所占用时间的长短。  
 会话流量控制：提供会话流量控制和交叉会话功能。  
寻址：使用远程地址建立会话连接。l  
出错控制：从逻辑上讲会话层主要负责数据交换的建立、保持和终止，但实际的工作却是接收来自传输层的数据，并负责纠正错误。会话控制和远程过程调用均属于这一层的功能。但应注意，此层检查的错误不是通信介质的错误，而是磁盘空间、打印机缺纸等类型的高级错误。  
**表示层**表示层（Presentation Layer）是OSI模型的第六层，它对来自应用层的命令和数据进行解释，对各种语法赋予相应的含义，并按照一定的格式传送给会话层。其主要功能是“处理用户信息的表示问题，如编码、数据格式转换和加密解密”等。表示层的具体功能如下：  
数据格式处理：协商和建立数据交换的格式，解决各应用程序之间在数据格式表示上的差异。  
数据的编码：处理字符集和数字的转换。例如由于用户程序中的数据类型（整型或实型、有符号或无符号等）、用户标识等都可以有不同的表示方式，因此，在设备之间需要具有在不同字符集或格式之间转换的功能。  
压缩和解压缩：为了减少数据的传输量，这一层还负责数据的压缩与恢复。  
数据的加密和解密：可以提高网络的安全性。

**应用层**应用层（Application Layer）是OSI参考模型的最高层，它是计算机用户，以及各种应用程序和网络之间的接口，其功能是直接向用户提供服务，完成用户希望在网络上完成的各种工作。它在其他6层工作的基础上，负责完成网络中应用程序与网络操作系统之间的联系，建立与结束使用者之间的联系，并完成网络用户提出的各种网络服务及应用所需的监督、管理和服务等各种协议。此外，该层还负责协调各个应用程序间的工作。  
应用层为用户提供的服务和协议有：文件服务、目录服务、文件传输服务（FTP）、远程登录服务（Telnet）、电子邮件服务（E-mail）、打印服务、安全服务、网络管理服务、数据库服务等。上述的各种网络服务由该层的不同应用协议和程序完成，不同的网络操作系统之间在功能、界面、实现技术、对硬件的支持、安全可靠性以及具有的各种应用程序接口等各个方面的差异是很大的。应用层的主要功能如下：  
用户接口：应用层是用户与网络，以及应用程序与网络间的直接接口，使得用户能够与网络进行交互式联系。  
实现各种服务：该层具有的各种应用程序可以完成和实现用户请求的各种服务。

 OSI7层模型的小结  
由于OSI是一个理想的模型，因此一般网络系统只涉及其中的几层，很少有系统能够具有所有的7层，并完全遵循它的规定。  
在7层模型中，每一层都提供一个特殊的网络功能。从网络功能的角度观察：下面4层（物理层、数据链路层、网络层和传输层）主要提供数据传输和交换功能，即以节点到节点之间的通信为主；第4层作为上下两部分的桥梁，是整个网络体系结构中最关键的部分；而上3层（会话层、表示层和应用层）则以提供用户与应用程序之间的信息和数据处理功能为主。简言之，下4层主要完成通信子网的功能，上3层主要完成资源子网的功能。

以下是TCP/IP分层模型  
        ┌────------────┐┌─┬─┬─-┬─┬─-┬─┬─-┬─┬─-┬─┬─-┐  
　　│　　　　　　　　││Ｄ│Ｆ│Ｗ│Ｆ│Ｈ│Ｇ│Ｔ│Ｉ│Ｓ│Ｕ│　│  
　　│　　　　　　　　││Ｎ│Ｉ│Ｈ│Ｔ│Ｔ│Ｏ│Ｅ│Ｒ│Ｍ│Ｓ│其│  
　　│第四层，应用层　││Ｓ│Ｎ│Ｏ│Ｐ│Ｔ│Ｐ│Ｌ│Ｃ│Ｔ│Ｅ│　│  
　　│　　　　　　　　││　│Ｇ│Ｉ│　│Ｐ│Ｈ│Ｎ│　│Ｐ│Ｎ│　│  
　　│　　　　　　　　││　│Ｅ│Ｓ│　│　│Ｅ│Ｅ│　│　│Ｅ│它│  
　　│　　　　　　　　││　│Ｒ│　│　│　│Ｒ│Ｔ│　│　│Ｔ│　│  
　　└───────------─┘└─┴─┴─-┴─┴─-┴─┴─-┴─┴─-┴─┴-─┘  
　　┌───────-----─┐┌─────────-------┬──--------─────────┐  
　　│第三层，传输层　││　　　ＴＣＰ　　　│　　　　ＵＤＰ　　　　│  
　　└───────-----─┘└────────-------─┴──────────--------─┘  
　　┌───────-----─┐┌───----──┬───---─┬────────-------──┐  
　　│　　　　　　　　││　　　　　│ＩＣＭＰ│　　　　　　　　　　│  
　　│第二层，网间层　││　　　　　└──---──┘　　　　　　　　　　│  
　　│　　　　　　　　││　　　　　　　ＩＰ　　　　　　　　　　　 │  
　　└────────-----┘└────────────────────-------------─-┘  
　　┌────────-----┐┌─────────-------┬──────--------─────┐  
　　│第一层，网络接口││ＡＲＰ／ＲＡＲＰ　│　　　　其它　　　　　│  
　　└────────------┘└─────────------┴─────--------──────┘  
　　　　　　 TCP/IP四层参考模型   
  
　　TCP/IP协议被组织成四个概念层，其中有三层对应于ISO参考模型中的相应层。ICP/IP协议族并不包含物理层和数据链路层，因此它不能独立完成整个计算机网络系统的功能，必须与许多其他的协议协同工作。  
　　TCP/IP分层模型的四个协议层分别完成以下的功能：  
　　第一层:网络接口层  
　　包括用于协作IP数据在已有网络介质上传输的协议。实际上TCP/IP标准并不定义与ISO数据链路层和物理层相对应的功能。相反，它定义像地址解析协议(Address Resolution Protocol,ARP)这样的协议，提供TCP/IP协议的数据结构和实际物理硬件之间的接口。  
　　第二层:网间层  
　　对应于OSI七层参考模型的网络层。本层包含IP协议、RIP协议(Routing Information Protocol，路由信息协议)，负责数据的包装、寻址和路由。同时还包含网间控制报文协议(Internet Control Message Protocol,ICMP)用来提供网络诊断信息。  
　　第三层:传输层  
　　对应于OSI七层参考模型的传输层，它提供两种端到端的通信服务。其中TCP协议(Transmission Control Protocol)提供可靠的数据流运输服务，UDP协议(Use Datagram Protocol)提供不可靠的用户数据报服务。  
　　第四层:应用层  
　　对应于OSI七层参考模型的应用层和表达层。因特网的应用层协议包括Finger、Whois、FTP(文件传输协议)、Gopher、HTTP(超文本传输协议)、Telent(远程终端协议)、SMTP(简单邮件传送协议)、IRC(因特网中继会话)、NNTP（网络新闻传输协议）等，这也是本书将要讨论的重点。

* 1. TCP/IP协议详解

用户发送请求

传输层 TCP

网络层

数据链路层

物理层

* 1. TCP的三次握手和四次挥手

**一、TCP报文格式**



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：发起一个新连接。

（F）FIN：释放一个连接。

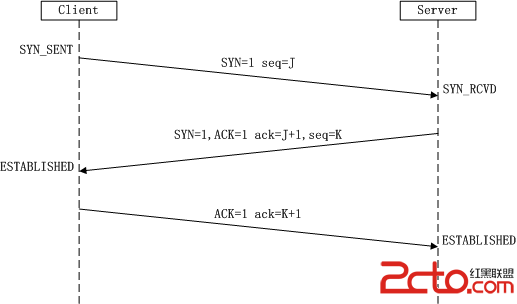
需要注意的是：

（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Req+1，两端配对。

**二、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。

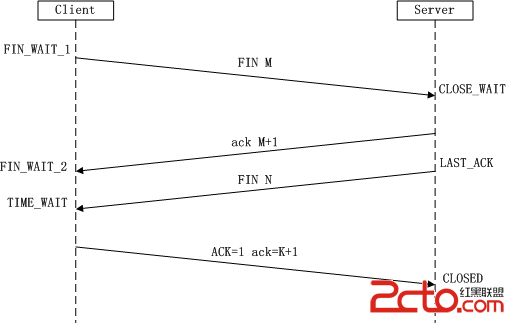
SYN攻击：

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

**三、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连接是全双工的，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。

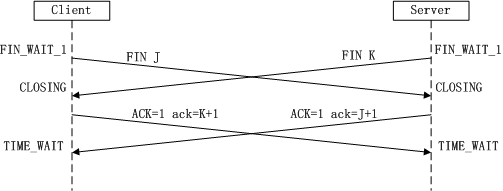
（1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。

（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：



流程和状态在上图中已经很明了了，在此不再赘述，可以参考前面的四次挥手解析步骤。

四、附注

关于三次握手与四次挥手通常都会有典型的面试题，在此提出供有需求的XDJM们参考：

（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。而关闭连接时，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。

* 1. 滑动窗口
  2. 同步和非同步 阻塞和非阻塞的区别
  3. [CLOSE\_WAIT状态的原因与解决方法](https://www.baidu.com/link?url=V91xPsm7KDruTjbDKanjmjb9J1NVklsacfH3bjvCWXCsAzljcanoB57GDdsZmIEG3wYlPwWRaok4ROAxSYI7jaNKHnxZpahV6hBCfp6ebRS&wd=&eqid=d6a5a7350000f2ad000000035b096933)
  4. [HTTP协议的特点](https://blog.csdn.net/whatday/article/details/38147103)

（1）Http协议时无状态的

Cookie

（2）传输是明文

* 1. [HTTPS协议](https://blog.csdn.net/whatday/article/details/38147103)

通信原理

怎么实现安全传输的

SSL、TLS

非对称加密

公钥怎么去拿

* 1. [HTTPS和HTTP的区别](https://blog.csdn.net/whatday/article/details/38147103)
     1. 什么是 HTTPS?

HTTPS (基于安全套接字层的超文本传输协议 或者是 HTTP over SSL) 是一个 Netscape 开发的 Web 协议。

你也可以说：HTTPS = HTTP + SSL

HTTPS 在 HTTP 应用层的基础上使用安全套接字层作为子层。

* + 1. 为什么需要 HTTPS ？

超文本传输协议 (HTTP) 是一个用来通过互联网传输和接收信息的协议。HTTP 使用请求/响应的过程，因此信息可在服务器间快速、轻松而且精确的进行传输。当你访问 Web 页面的时候你就是在使用 HTTP 协议，但 HTTP 是不安全的，可以轻松对窃听你跟 Web 服务器之间的数据传输。在很多情况下，客户和服务器之间传输的是敏感歇息，需要防止未经授权的访问。为了满足这个要求，网景公司(Netscape)推出了[HTTPS](http://www.nowamagic.net/librarys/veda/tag/https)，也就是基于安全套接字层的 HTTP 协议。

* + 1. HTTP 和 HTTPS 的相同点

大多数情况下，HTTP 和 HTTPS 是相同的，因为都是采用同一个基础的协议，作为 HTTP 或 HTTPS 客户端——浏览器，设立一个连接到 Web 服务器指定的端口。当服务器接收到请求，它会返回一个状态码以及消息，这个回应可能是请求信息、或者指示某个错误发送的错误信息。系统使用统一资源定位器 URI 模式，因此资源可以被唯一指定。而 HTTPS 和 HTTP 唯一不同的只是一个协议头(https)的说明，其他都是一样的。

* + 1. HTTP 和 HTTPS 的不同之处

1. HTTP 的 URL 以 http:// 开头，而 HTTPS 的 URL 以 https:// 开头
2. HTTP 是不安全的，而 HTTPS 是安全的
3. HTTP 标准端口是 80 ，而 HTTPS 的标准端口是 443
4. 在 OSI 网络模型中，HTTP 工作于应用层，而 HTTPS 工作在传输层
5. HTTP 无需加密，而 HTTPS 对传输的数据进行加密
6. HTTP 无需证书，而 HTTPS 需要认证证书
   * 1. HTTPS 如何工作?

使用 HTTPS 连接时，服务器要求有公钥和签名的证书。

当使用 https 连接，服务器响应初始连接，并提供它所支持的加密方法。作为回应，客户端选择一个连接方法，并且客户端和服务器端交换证书验证彼此身份。完成之后，在确保使用相同密钥的情况下传输加密信息，然后关闭连接。为了提供 https 连接支持，服务器必须有一个公钥证书，该证书包含经过证书机构认证的密钥信息，大部分证书都是通过第三方机构授权的，以保证证书是安全的。

换句话说，HTTPS 跟 HTTP 一样，只不过增加了 [SSL](http://www.nowamagic.net/librarys/veda/tag/SSL)。

HTTP 包含如下动作：

1. 浏览器打开一个 TCP 连接
2. 浏览器发送 HTTP 请求到服务器端
3. 服务器发送 HTTP 回应信息到浏览器
4. TCP 连接关闭

SSL 包含如下动作：

1. 验证服务器端
2. 允许客户端和服务器端选择加密算法和密码，确保双方都支持
3. 验证客户端(可选)
4. 使用公钥加密技术来生成共享加密数据
5. 创建一个加密的 SSL 连接
6. 基于该 SSL 连接传递 HTTP 请求
   * 1. 什么时候该使用 HTTPS?

银行网站、支付网关、购物网站、登录页、电子邮件以及一些企业部门的网站应该使用 HTTPS，例如：

* PayPal: https://www.paypal.com
* Google AdSense: https://www.google.com/adsense/

如果某个网站要求你填写信用卡信息，首先你要检查该网页是否使用 https 加密连接，如果没有，那么请不要输入任何敏感信息如信用卡号。

* + 1. 浏览器集成

多数浏览器在收到一个无效证书的时候都会显示警告信息，而一些老的浏览器会弹出对话框让用户选择是否继续浏览。新的浏览器一般在整个窗口显示横幅的警告信息，同时在地址栏上显示该网站的安全信息。如果网站中包含加密和非加密的混合内容，多数浏览器会提示警告信息。

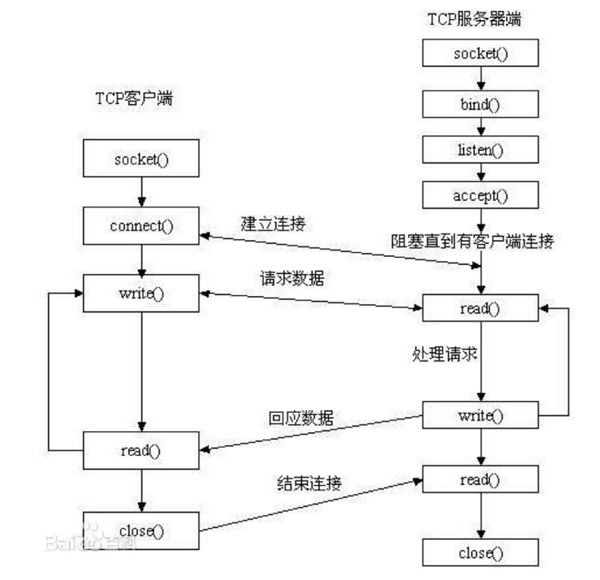
* 1. Http、TCP/IP协议与Socket之间的区别(1)

socket（套接字）是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

多个TCP连接或多个应用程序进程可能需要通过同一个TCP协议端口传输数据。为了区别不同的应用程序进程和连接，计算机操作系统为应用程序与TCP/IP协议交互提供了套接字(Socket)接口。应用层可以和传输层通过Socket接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

建立Socket连接至少需要一对套接字，其中一个运行于客户端，称为ClientSocket，另一个运行于服务器端，称为ServerSocket。套接字之间的连接过程分为三个步骤：服务器监听，客户端请求，连接确认。

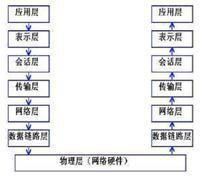
Socket可以支持不同的传输层协议（TCP或UDP），当使用TCP协议进行连接时，该Socket连接就是一个TCP连接,UDP连接同理。



**网络七层协议**

网络七层协议由下往上分别为物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。其中物理层、数据链路层和网络层通常被称作媒体层，是网络工程师所研究的对象；传输层、会话层、表示层和应用层则被称作主机层，是用户所面向和关心的内容。

HTTP协议对应于应用层，TCP协议对应于传输层，IP协议对应于网络层，HTTP协议是基于TCP连接的,三者本质上没有可比性。 TCP/IP是传输层协议，主要解决数据如何在网络中传输；而HTTP是应用层协议，主要解决如何包装数据。**Socket是应用层与TCP/IP协议族通信的中间软件抽象层，是它的一组接口。**

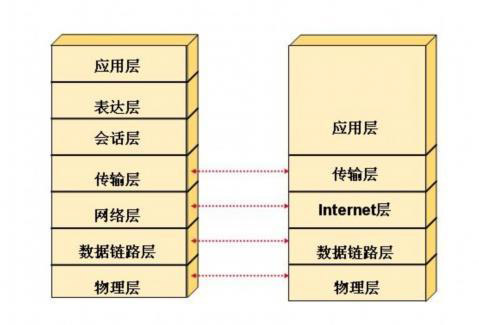
****

网络七层协议

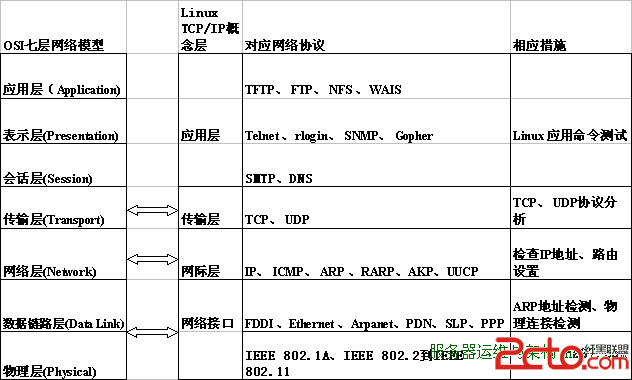
**TCP/IP五层模型**

TCP/IP五层模型的协议分为：应用层、传输层、网络层、数据链路层和物理层。中继器、集线器、还有我们通常说的双绞线也工作在物理层；网桥（现已很少使用）、以太网交换机（二层交换机）、网卡（其实网卡是一半工作在物理层、一半工作在数据链路层）在数据链路层；路由器、三层交换机在网络层；传输层主要是四层交换机、也有工作在四层的路由器。

TCP/IP协议中的应用层处理七层模型中的第五层、第六层和第七层的功能。TCP/IP协议中的传输层并不能总是保证在传输层可靠地传输数据包，而七层模型可以做到。TCP/IP协议还提供一项名为UDP（用户数据报协议）的选择。UDP不能保证可靠的数据包传输。







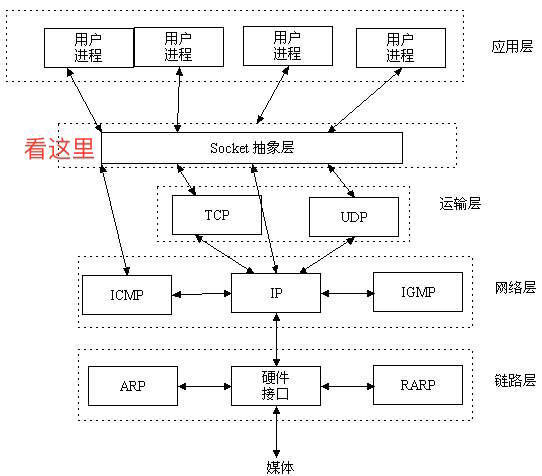
对应关系

* TCP：面向连接、传输可靠(保证数据正确性,保证数据顺序)、用于传输大量数据(流模式)、速度慢，建立连接需要开销较多(时间，系统资源)。
* UDP：面向非连接、传输不可靠、用于传输少量数据(数据包模式)、速度快。

TCP是一种流模式的协议，UDP是一种数据报模式的协议。

在传输数据时，可以只使用传输层（TCP/IP），但是那样的话，由于没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用应用层协议（HTTP、FTP、TELNET等），也可以自己定义应用层协议。

WEB使用HTTP作传输层协议，以封装HTTP文本信息，然后使用TCP/IP做传输层协议将它发送到网络上。Socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，我们才能使用TCP/IP协议。



Socket的位置

**TCP连接**

要想明白Socket连接，先要明白TCP连接。手机能够使用联网功能是因为手机底层实现了TCP/IP协议，可以使手机终端通过无线网络建立TCP连接。TCP协议可以对上层网络提供接口，使上层网络数据的传输建立在“无差别”的网络之上。

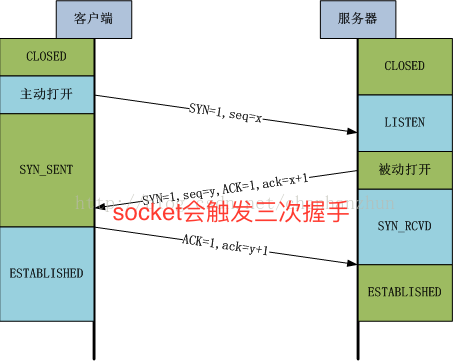
建立起一个TCP连接需要经过“三次握手”：

第一次握手：客户端发送syn包(syn=j)到服务器，并进入SYN\_SEND状态，等待服务器确认；

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN＋ACK包，向服务器发送确认包ACK(ack=k+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

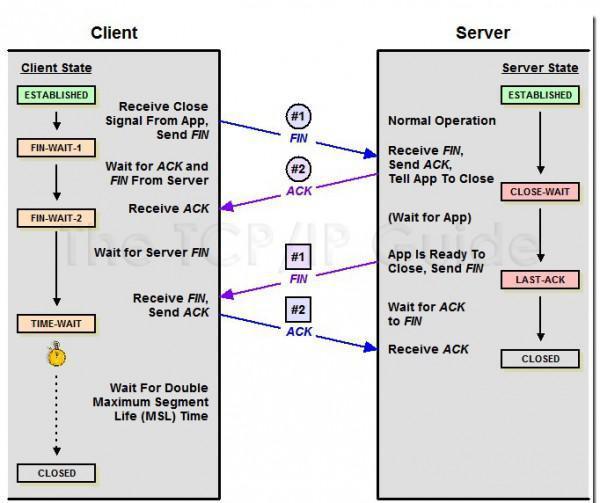
三次握手(Three-way Handshake)即建立一个TCP连接时，需要客户端和服务器总共发送3个包。三次握手的目的是连接服务器指定端口，建立TCP连接,并同步连接双方的序列号和确认号并交换TCP 窗口大小信息。在socket编程中，客户端执行connect()时,将触发三次握手。



三次握手

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP 连接都将被一直保持下去。断开连接时服务器和客户端均可以主动发起断开TCP连接的请求，断开过程需要经过“四次握手”。

TCP连接的拆除需要发送四个包，因此称为四次握手(four-way handshake)。在socket编程中，任何一方执行close()操作即可产生握手（有地方称为“挥手”）操作。



TCP连接的拆除

之所以有“三次握手”和“四次握手”的区别，是因为连接时当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，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。

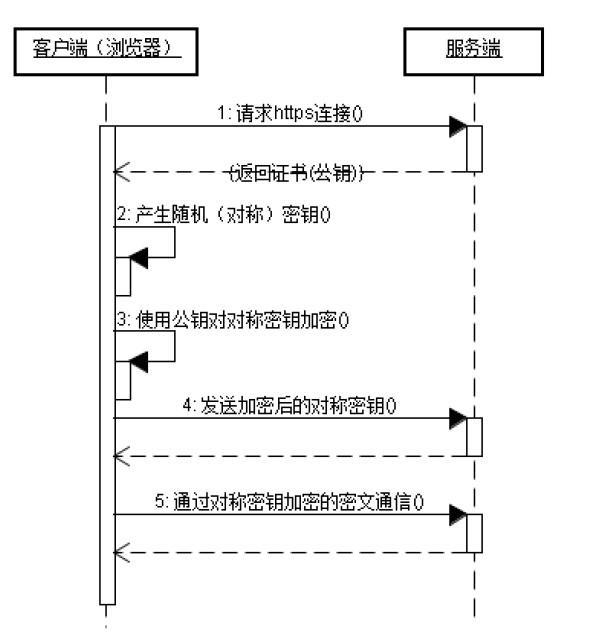
**HTTP连接**

HTTP协议即超文本传送协议(HypertextTransfer Protocol )，是Web联网的基础，也是手机联网常用的协议之一，HTTP协议是建立在TCP协议之上的一种应用。

HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。因此HTTP连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。在HTTP 1.0中，客户端的每次请求都要求建立一次单独的连接，在处理完本次请求后，就自动释放连接。在HTTP 1.1中则可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再发送下一个请求。

HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，是HTTP的安全版。 在HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。 HTTPS存在不同于HTTP的默认端口及一个加密/身份验证层（在HTTP与TCP之间）。HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息，因此HTTP协议不适合传输一些敏感信息。

https协议需要到ca申请证书；http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议；http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443；http的连接很简单，是无状态的，HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议。



HTTPS

**Socket连接与HTTP连接的不同**

**通常情况下Socket连接就是TCP连接**，因此Socket连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。但在实际应用中，客户端到服务器之间的通信防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态。

而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。

* 1. Http、TCP/IP协议与Socket之间的区别(2)

**1、TCP/IP连接**

手机能够使用联网功能是因为手机底层实现了TCP/IP协议，可以使手机终端通过无线网络建立TCP连接。TCP协议可以对上层网络提供接口，使上层网络数据的传输建立在“无差别”的网络之上。

建立起一个TCP连接需要经过“三次握手”：

第一次握手：客户端发送syn包(syn=j)到服务器，并进入SYN\_SEND状态，等待服务器确认;

第二次握手：服务器收到syn包，必须确认客户的SYN(ack=j+1)，同时自己也发送一个SYN包(syn=k)，即SYN+ACK包，此时服务器进入SYN\_RECV状态;

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP 连接都将被一直保持下去。断开连接时服务器和客户端均可以主动发起断开TCP连接的请求，断开过程需要经过“四次握手”(过程就不细写了，就是服务器和客户端交互，最终确定断开).

**2、HTTP连接**

HTTP协议即超文本传送协议(Hypertext Transfer Protocol )，是Web联网的基础，也是手机联网常用的协议之一，HTTP协议是建立在TCP协议之上的一种应用。

HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。

1)在HTTP 1.0中，客户端的每次请求都要求建立一次单独的连接，在处理完本次请求后，就自动释放连接。

2)在HTTP 1.1中则可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再发送下一个请求。

由于HTTP在每次请求结束后都会主动释放连接，因此HTTP连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即时不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

**3、SOCKET原理**

**3.1套接字(socket)概念**

套接字(socket)是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元。它是网络通信过程中端点的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

应用层通过传输层进行数据通信时，TCP会遇到同时为多个应用程序进程提供并发服务的问题。多个TCP连接或多个应用程序进程可能需要通过同一个 TCP协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机操作系统为应用程序与TCP/IP协议交互提供了套接字(Socket)接口。应用层可以和传输层通过Socket接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

**3.2 建立socket连接**

建立Socket连接至少需要一对套接字，其中一个运行于客户端，称为ClientSocket ，另一个运行于服务器端，称为ServerSocket 。

套接字之间的连接过程分为三个步骤：服务器监听，客户端请求，连接确认。

服务器监听：服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态，等待客户端的连接请求。

客户端请求：指客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

连接确认：当服务器端套接字监听到或者说接收到客户端套接字的连接请求时，就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，双方就正式建立连接。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

**4、SOCKET连接与TCP/IP连接**

创建Socket连接时，可以指定使用的传输层协议，Socket可以支持不同的传输层协议(TCP或UDP)，当使用TCP协议进行连接时，该Socket连接就是一个TCP连接。

socket则是对TCP/IP协议的封装和应用(程序员层面上)。也可以说，TPC/IP协议是传输层协议，主要解决数据 如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。关于TCP/IP和HTTP协议的关系，网络有一段比较容易理解的介绍：

“我们在传输数据时，可以只使用(传输层)TCP/IP协议，但是那样的话，如 果没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用到应用层协议，应用层协议有很多，比如HTTP、FTP、TELNET等，也 可以自己定义应用层协议。WEB使用HTTP协议作应用层协议，以封装HTTP文本信息，然后使用TCP/IP做传输层协议将它发到网络上。”

我们平时说的最多的socket是什么呢，实际上socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口(API)，通过Socket，我们才能使用TCP/IP协议。 实际上，Socket跟TCP/IP协议没有必然的联系。Socket编程接口在设计的时候，就希望也能适应其他的网络协议。所以说，Socket的出现 只是使得程序员更方便地使用TCP/IP协议栈而已，是对TCP/IP协议的抽象，从而形成了我们知道的一些最基本的函数接口，比如create、 listen、connect、accept、send、read和write等等。网络有一段关于socket和TCP/IP协议关系的说法比较容易理解：

“TCP/IP只是一个协议栈，就像操作系统的运行机制一样，必须要具体实现，同时还要提供对外的操作接口。这个就像操作系统会提供标准的编程接口，比如win32编程接口一样，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口。”

实际上，传输层的TCP是基于网络层的IP协议的，而应用层的HTTP协议又是基于传输层的TCP协议的，而Socket本身不算是协议，就像上面所说，它只是提供了一个针对TCP或者UDP编程的接口。socket是对端口通信开发的工具,它要更底层一些.

**5、Socket连接与HTTP连接**

由于通常情况下Socket连接就是TCP连接，因此Socket连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。但在实际网络应用中，客户端到服务器之间的通信往往需要穿越多个中间节点，例如路由器、网关、防火墙等，大部分防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态。

而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。

很多情况下，需要服务器端主动向客户端推送数据，保持客户端与服务器数据的实时与同步。此时若双方建立的是Socket连接，服务器就可以直接将数据传送给客户端;若双方建立的是HTTP连接，则服务器需要等到客户端发送一次请求后才能将数据传回给客户端，因此，客户端定时向服务器端发送连接请求，不仅可以保持在线，同时也是在“询问”服务器是否有新的数据，如果有就将数据传给客户端。

**http协议是应用层的协议**

有个比较形象的描述：HTTP是轿车，提供了封装或者显示数据的具体形式;Socket是发动机，提供了网络通信的能力。

两个计算机之间的交流无非是两个端口之间的数据通信,具体的数据会以什么样的形式展现`是以不同的应用层协议来定义的`如HTTP`FTP`...