计算机网络

一. 概述

1.1 OSI（开放系统互联参考模型）标准模型

1）物理层

- **功能**：负责为数据端设备**透明地传输原始比特流**，并且定义了数据终端设备和数据通信设备的物理和逻辑链接方法。传输单位是**比特。**
- **协议**：RJ45、CLOCK、IEEE802.3
- **设备**：中继器，集线器

2）数据链路层

- **功能**:将网络层传下来的IP数据报组装成帧，并检测和矫正物理层产生的传输差错，使得链路对网络层显示一条无差错、可靠的数据传输线路。功能可以概括为成帧，差错控制、流量控制和传输管理
- **协议**：HDLC(高级数据链路控制协议),PPP,STP,SDLC,CSMA(载波监听多路访问)
- **设备**: 网桥，交换机
- **数据链路层协议可能提供的服务?**

  成帧、链路访问、透明传输、可靠交付、流量控制、差错检测、差错纠正、半双工和全双工。最重要的是帧定界（成帧）、透明传输以及差错检测。

3）网络层

- **功能**：负责在网络层上将数据**封装成数据报**，将数据报从源端传到目的端，同时进行**路由选择**，为分组交换网上的不同主机提供**通信服务。**关键问题是对分组进行选择，并实现流量控制、拥塞控制、差错控制和网际互联等功能。传输单位**数据报。**
- **协议**: IP,ICMP(因特网控制报文协议),IGMP(因特网组管理协议),ARP,RARP,OSPF(开放最短路径优先),IPX
- **设备**: 路由器

4）传输层

- **功能**：负责主机中两个进程之间的通信，为端到端连接提供可靠的传输服务。为端到端连接提供流量控制、差错控制、服务质量、数据传输管理等服务。
- **协议**：TCP,UDP

5）会话层

- **功能**：会话层允许不同主机上各个进程之间的**会话**，会话层利用传输层提供的端到端的服务，向表示层提供它的增值服务。这种服务主要是为表示层实体或用户进程**建立连接**并在连接上提供**有序**地传输数据。
- **协议**：SQL、RPC(远程调用协议)

6）表示层

- **功能**:用于处理两个通信系统中交换信息的表示方式。如数据压缩，加密和解密等。
- **协议**：JPEG、MPEG、ASII

7）应用层

- **功能**：是TCP/IP的最高层，它是直接为应用进程服务的一层。当不同的应用进程数据通信或数据交换时，就去调用应用层的**不同协议实体**，让这些实体去调用TCP或者UDP层服务来进行网络传输。
- **协议**：FTP(21) TELNET(23) SMTP(25) DNS(53) TFTP(69) HTTP(80) SNMP(161),DHCP(动态主机配置协议)

分层	作用	协议
物理层	通过媒介传输**比特**	IEEE802.3
数据链路层	将网络层的IP数据报封装成帧和点到点的传递（帧：Frame）	PPP、MAC（网桥，交换机）
网络层	数据包从源到宿的传递和网际互连（包：Packet）	IP（网际协议）, ICMP（网际控制报文协议）, ARP（地址解析协议）,
传输层（运输层）	提供端到端的可靠报文传递和错误恢复（段：segment）	TCP（传输控制协议）, UDP（用户数据报协议）
会话层		
表示层		
应用层		DNS, FTP, HTTP,

- 网络层：路由表包含：网络ID，子网掩码（判断IP所属网络），下一跳地址/接口

1.2 TCP/IP 分层

- **网络接口层（接收和发送数据报）**：负责将数据报发送到**网络介质**上，以及从网络上接收TCP/IP数据报，相当于OSI的物理层和数据层。
- **网际层（数据报封装和路由寻址功能）**：主要负责寻址和对数据报的封装以及重要的路由选择功能。
- **传输层**：**负责在应用进程之间的"端到端"的通信**，即从某个应用进程传输到另一个应用进程。
- **应用层**：是TCP/IP的最高层，它是直接为应用进程服务的一层。当不同的应用进程数据通信或数据交换时，就去调用应用层的**不同协议实体**，让这些实体去调用TCP或者UDP层服务来进行网络传输。

1.3 五层体系结构模型

物理层：考虑的是怎样在传输媒体上传输数据比特流，而不是指具体的传输媒体。物理层的作用是尽可能屏蔽传输媒体和通信手段的差异，使数据链路层感觉不到这些差异。

数据链路层：网络层针对的还是主机之间的数据传输服务，而主机之间可以有很多链路，链路层协议就是为同一链路的结点提供服务。数据链路层把网络层传来的分组封装成帧。

网络层：为主机之间提供数据传输服务，而运输层协议是为主机中的进程提供服务。网络层把运输层传递下来的报文段或者用户数据报封装成分组。

运输层：提供的是进程间的通用数据传输服务。由于应用层协议很多，定义通用的运输层协议就可以支持不断增多的应用层协议。运输层包括两种协议：传输控制协议 TCP，提供面向连接、可靠的数据传输服务，数据单位为报文段；用户数据报协议 UDP，提供无连接、尽最大努力的数据传输服务，数据单位为用户数据报。TCP 主要提供完整性服务，UDP 主要提供及时性服务。

应用层：为特定应用程序提供数据传输服务，例如 HTTP、DNS 等。数据单位为报文。

OSI 和TCP/IP的区别

1. OSI精确定义了服务、协议和接口，符合**面向对象**程序设计思想。而TCP/IP在这些概念上没有明确区分。
2. OSI在产生协议之前没有偏向任何特定的协议，通用性良好，但如果设计没有太多经验就不知道将哪些功能放到哪一层。TCP/IP,首先出现协议，模型实际上是对已有协议的描述。因此不会出现协议不能匹配的模式。
3. TCP/IP考虑了异构网的互联问题，而OSI只考虑用一种标准的公用数据网将各种不同系统互连。
4. OSI在网络层支持无连接和面向连接的通信，但在传输层仅有面向连接通信。而TCP/IP则相反，在网际层仅有无连接服务，在传输层支持无连接和面向连接两种模式。

二. 网络层

2.1 ARP是地址解析协议，简单语言解释一下工作原理

每个主机都存有一个ARP高速缓存，存放**本局域网上**各主机和路由器的**IP地址到MAC地址的映射表**，称为ARP表。使用ARP协议动态维护此表。

ARP工作在网络层中，其工作原理是：当主机A欲向**本局域网上**的某个主机B发送IP数据报时，就先在其ARP高速缓存中查看有无主机B的IP地址:

- 如果有可以查出其对应的硬件地址，再将此硬件地址写入**MAC帧**，然后通过局域网将该MAC帧发往此硬件地址。
- 如果没有，就通过使用目的MAC地址为本网络的广播地址即32个1的帧来封装并广播ARP请求分组，可以使同一个局域网里的所有主机收到ARP请求。当主机B收到该ARP请求后，就会向主机A发出响应ARP分组，分组中包含主机B的IP与MAC地址的映射关系，主机A在收到后将此映射写入ARP缓存中，然后按查询到的硬件地址发送MAC帧。

ARP是解决同一个局域网上主机与路由器的IP地址和硬件地址的映射问题。如果所要找的主机和源主机不在同一个局域网上，那么通过ARP协议找到一个位于本局域网上的某个路由器硬件地址，然后把分组发送给这个路由器，让这个路由器把分组转发给下一个网络。

2.2 描述RARP协议

RARP是逆地址解析协议，作用是完成**硬件地址到IP地址的映射**.(请求是广播，应答是单播)：

- 网络上的每台设备都会有一个独一无二的硬件地址，通常是由设备厂商分配的MAC地址。主机从网卡上读取MAC地址，然后在网络上发送一个RARP请求的广播数据包，请求的RARP服务器分配一个IP地址。
- 本地网段上的RARP服务器收到此请求后，检查其RARP列表，查找该MAC地址对应的IP地址。
- 如果存在，RARP服务器就给源主机发送一个响应数据包并将此IP地址提供给对方主机使用。
- 如果不存在，RARP服务器对此不做任何的响应。
- 源主机收到从RARP服务器的响应信息，就利用得到的IP地址进行通讯；如果一直没有收到RARP服务器的响应信息，表示初始化失败

2.3 常见的路由选择协议，以及它们的区别

常见的路由选择协议有：RIP协议、OSPF协议。

- **RIP协议（路由信息协议）**：底层是贝尔曼福特算法(Bellman-Ford)，**基于距离向量的路由选择协议**，它选择路由的度量标准（metric)是**跳数**，最大跳数是15跳，如果大于15跳，它就会丢弃数据包。仅和相邻路由器交换当前路由器所知道的全部信息。是应用层协议,使用UDP传输数据,端口520

- **OSPF协议（开放最短路由优先）**：底层是迪杰斯特拉(Dijkstra)算法，**是链路状态路由选择协议**，它选择路由的度量标准是**带宽**，延迟。向本自治系统中所有路由器发送与本路由器相邻的所有路由器的链路状态，但这只是路由器知道的部分信息。网络层协议,直接IP数据报传输． 端口89。

了解交换机、路由器、网关的概念，并知道各自的用途

两者简介：

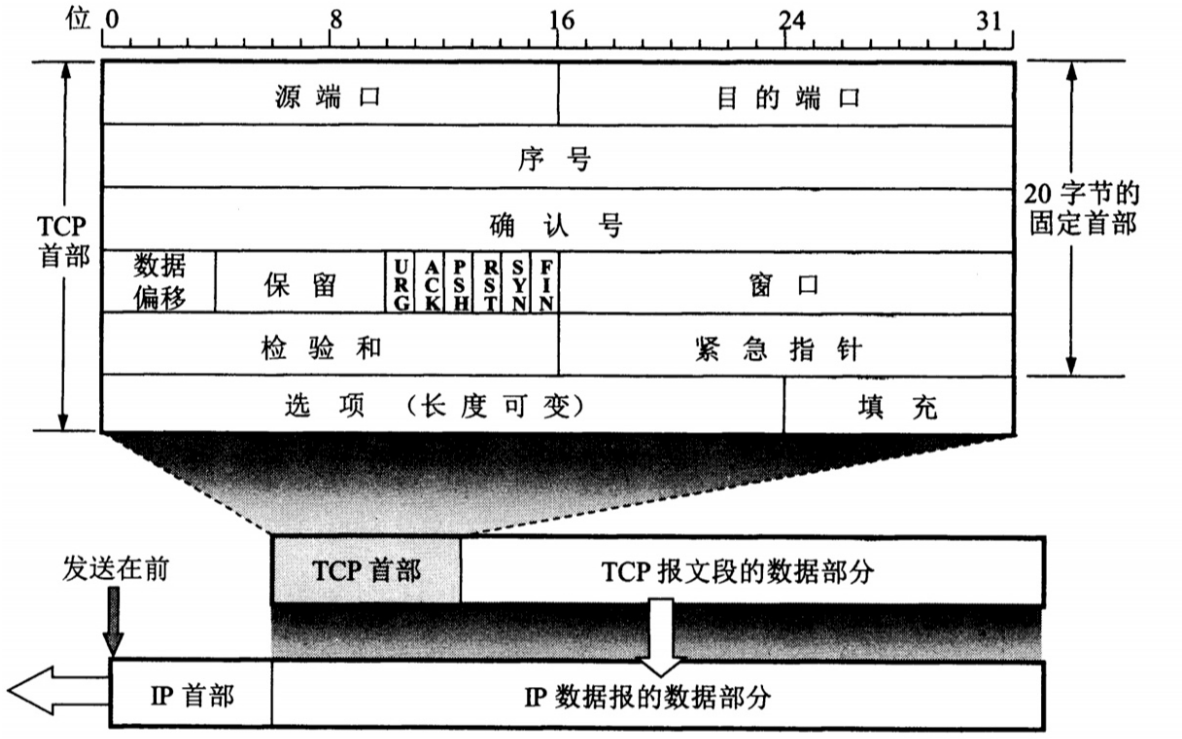
- **路由器**是具有多个输入输出端口的专用计算机，**其任务是连接不同的网络并完成路由转发。**当源主机向目标主机发送数据报时，路由器先检查源主机与目标主机是否连接在一个网络上。如果源主机和目标主机在一个网络上则直接交付而无需通过路由器。但如果源主机和目标主机不在同一个网络上，则路由器转发表指出路由将数据报转发给下一个路由器，即间接交付。路由器隔离了广播域。
- **交换机**：是多个端口的网桥，工作在数据链路上，**将两个或多个以太网连接起来成为更大的以太网。它能将网络分成小的冲突域，为每个工作站提供更高的带宽。**其原理是，检测从以太端口来的数据帧的源和目的地的MAC地址，然后与系统内部的动态查找表进行比较。若数据帧的MAC地址不在查找表中，则将该地址加入查找表中，并将数据帧发送给相应的端口。

区别：

- **路由器**：工作在网络层，是能够连接不同的广域网形成更大的广域网。连接的是异构网络。根据IP地址转发。
- **交换机**：工作在数据链路层，是将以太网连接形成更大的以太网，同一个网络。根据MAC地址进行转发。
- **网关(Gateway)**又称网间连接器、协议转换器。网关在网络层以上实现网络互连，是最复杂的网络互连设备，**仅用于两个高层协议不同的网络互连。**网关既可以用于广域网互连，也可以用于局域网互连。 网关是一种充当转换重任的计算机系统或设备。使用在不同的通信协议、数据格式或语言，甚至体系结构完全不同的两种系统之间，网关是一个翻译器。

三. 传输层

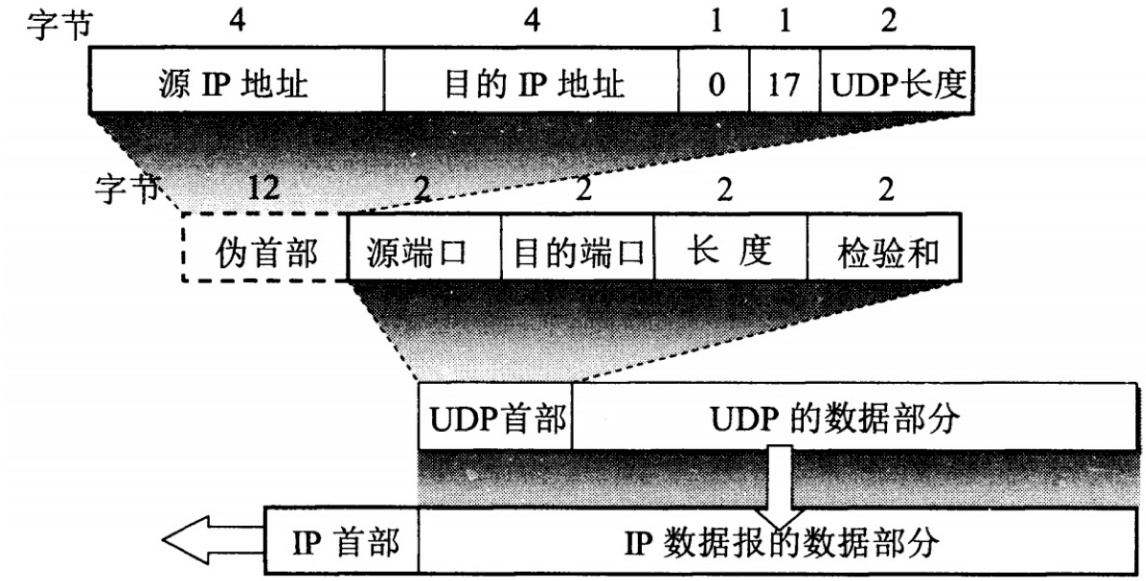
3.1 TCP报文段的首部格式



1. **源端口和目的端口。**各占2字节。

2. **序号**: TCP每个一个字节都是按顺序编号。
3. **确认号**: 期望收到对方下一个报文段的第一个数据字节的序号。
4. **数据偏移**：占用4字节，指出TCP报文段数据处距离TCP报文段的起始处有多远。
5. **保留**: 占用6位，留以后使用。
6. **紧急URG**: URG=1表示紧急，告诉系统有紧急数据，应尽快传送，而不按原来的排队顺序来传送。
7. **确认ACK**：只有ACK=1时，确认号ack才有效。
8. **推送PSH**: 希望立即能够收到对方的响应。
9. **复位RST**：RST=1,表明连接中出现严重差错，必须释放连接，然后再重新建立运输连接。
10. **同步SYN**: 在连接建立时用来同步序号。当SYN=1，ACK=0时表示连接请求报文段。如果对方同意则响应报文中使用SYN=1,ACK=1.
11. **终止FIN**: 用来释放连接。
12. **窗口**：占两字节，窗口值作为接收方让发送方设置发送窗口的依据。
13. **校验和**：占2字节，校验和字段校验的范围包括首部和数据两部分。
14. **紧急指针**：占两字节，只有URG=1时才有意义，紧急指针指出紧急数据的末尾在报文段中的位置。
15. **选项**：长度可变，最长40字节。

3.2 UDP报文段首部格式



- **源端口**: 源端口号，在需要对方回信时选用，不需要时可用全0.
- **目的端口**: 目的端口号，在终点交付报文时必须使用
- **长度**: UDP用户数据报的长度，其最小值是8(仅有头部)。
- **校验和**: 检测UDP用户数据报在传输中是否有错。有错就丢弃。

3.3 TCP与UDP的区别

1. 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
2. TCP提供可靠**全双功**的通信服务。UDP是半双功，只能单向传播。
3. 通过TCP连接可靠传送的数据，**可靠的**、无差错，不丢失，不重复，且按序到达;UDP则是**不可靠信道**，尽最大努力交付，即不保证可靠交付.
4. TCP**面向字节流**，实际上是TCP把数据看成一连串无结构的字节流;UDP是**面向报文**的.
5. TCP具有**拥塞控制**，UDP**没有拥塞控制**，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）
6. 每一条TCP连接只能是**点到点**的;UDP比较灵活，支持一对一，一对多，多对一和多对多的交互通信

7. TCP首部开销20字节;UDP的首部开销小，只有8个字节.

3.4 TCP应用场景

效率要求相对低，但对准确性要求相对高的场景。因为传输中需要对数据确认、重发、排序等操作，相比之下效率没有UDP高。举几个例子：文件传输（准确高要求高、但是速度可以相对慢）、接受邮件、远程登录。 NSQ底层通讯就采用TCP。

TCP

- TCP：面向连接的、可靠的、基于**字节流**的传输层通信协议，传输的单位是报文段
- 特征：面向连接，只能一对一，可靠交互，全双工通信（发送的同时也可以接收数据）

3.5 UDP应用场景

效率要求相对高，对准确性要求相对低的场景。举几个例子：QQ聊天、在线视频、网络语音电话（即时通讯，速度要求高，但是出现偶尔断续不是太大问题，并且此处完全不可以使用重发机制）、广播通信（广播、多播）。

UDP

- UDP：无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，单位是用户**数据报**
- 特征：无连接，尽最大努力交付，面向报文，没有拥塞控制，支持一对一、一对多、多对一、多对多的交互通信，首部开销小

udp调用connect

- UDP中可以使用connect系统调用
- UDP中connect操作与TCP中connect操作有着本质区别.
    - TCP中调用connect会引起三次握手,client与server建立连结.UDP中调用connect内核仅仅把对端ip&port记录下来.
- UDP中可以多次调用connect,TCP只能调用一次connect.UDP多次调用connect.
- 有两种用途:
    1. 指定一个新的ip&port连结.
    2. 断开和之前的ip&port的连结.指定新连结,直接设置connect第二个参数即可.断开连结,需要将connect第二个参数中的sin_family设置成 AF_UNSPEC即可.
- UDP中使用connect可以提高效率.原因如下:
    - 普通的UDP发送两个报文内核做了如下:#1:建立连结#2:发送报文#3:断开连结#4:建立连结#5:发送报文#6:断开连结,
    - 采用connect方式的UDP发送两个报文内核如下处理:#1:建立连结#2:发送报文#

UDP中使用connect的好处:

- 会提升效率.前面已经描述了
- 高并发服务中会增加系统稳定性.原因:假设client A 通过非connect的UDP与serverB,C通信.B,C提供相同服务.为了负载均衡,我们让A与B,C交替通信.A 与 B通信IPa:PORTa<----> IPb:PORTbA 与 C通信IPa:PORTa'<---->IPc:PORTc
- 假设PORTa 与 PORTa'相同了(在大并发情况下会发生这种情况),那么就有可能出现A等待B的报文,却收到了C的报文.导致收报错误.解决方法内就是采用connect的UDP通信方式.在A中创建两个udp,然后分别connect到B,C.

3.6 TCP和UDP对比

TCP 与 UDP 的区别

1. TCP 面向连接，UDP 是无连接的；
2. 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
3. TCP 的逻辑通信信道是全双工的可靠信道；UDP 则是不可靠信道
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 面向字节流（可能出现黏包问题），实际上是 TCP 把数据看成一连串无结构的字节流；UDP 是面向报文的（不会出现黏包问题）
6. UDP 没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如 IP 电话，实时视频会议等）
7. TCP 首部开销20字节；UDP 的首部开销小，只有 8 个字节

- 路由器转发规则，一些常见的概念。tcp的三次握手四次挥手的过程，tcp和udp的区别
- tcp三次握手漏洞：收到一个恶意攻击的ip一直请求连接，然后服务器会发送ack确认，但是永远等不到回复，就会导致服务器的NIC，内存，cpu占用率超载，这种攻击方式叫做基于TCP半开回话的洪水攻击
- TCP黏包问题：因为TCP是基于字节流的，所以可能两个包没有边界；

  解决办法：包头加上包体长度，数据包之间设置边界
- TCP流量控制：可变窗口

  发送端窗口大小不能超过接收端窗口大小的值

3.7 TCP对应的协议和UDP对应的协议

1）TCP对应的协议

- **FTP(21)**：定义了**文件传输协议**，使用21端口。常说某某计算机开了FTP服务便是启动了文件传输服务。下载文件，上传主页，都要用到FTP服务。
- **ssh(22)**: 专为**远程登录会话**和其他网络服务提供安全性的协议
- **Telnet(23)**：(远程登陆协议)它是一种用于**远程登陆**的端口，用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于DOS模式下的通信服务。如以前的BBS是-纯字符界面的，支持BBS的服务器将23端口打开，对外提供服务。
- **SMTP(25)**：定义了**简单邮件传送协议**，现在很多邮件服务器都用的是这个协议，用于发送邮件。如常见的免费邮件服务中用的就是这个邮件服务端口，所以在电子邮件设置-中常看到有这么SMTP端口设置这个栏，服务器开放的是25号端口。
- **POP3(110)**：它是和SMTP对应，POP3用于**接收邮件**。通常情况下，POP3协议所用的是110端口。也是说，只要你有相应的使用POP3协议的程序（例如Fo-xmail或Outlook），就可以不以Web方式登陆进邮箱界面，直接用邮件程序就可以收到邮件（如是163邮箱就没有必要先进入网易网站，再进入自己的邮-箱来收信）。
- **HTTP(80)协议**：是从Web服务器传输超文本到本地浏览器的传送协议。

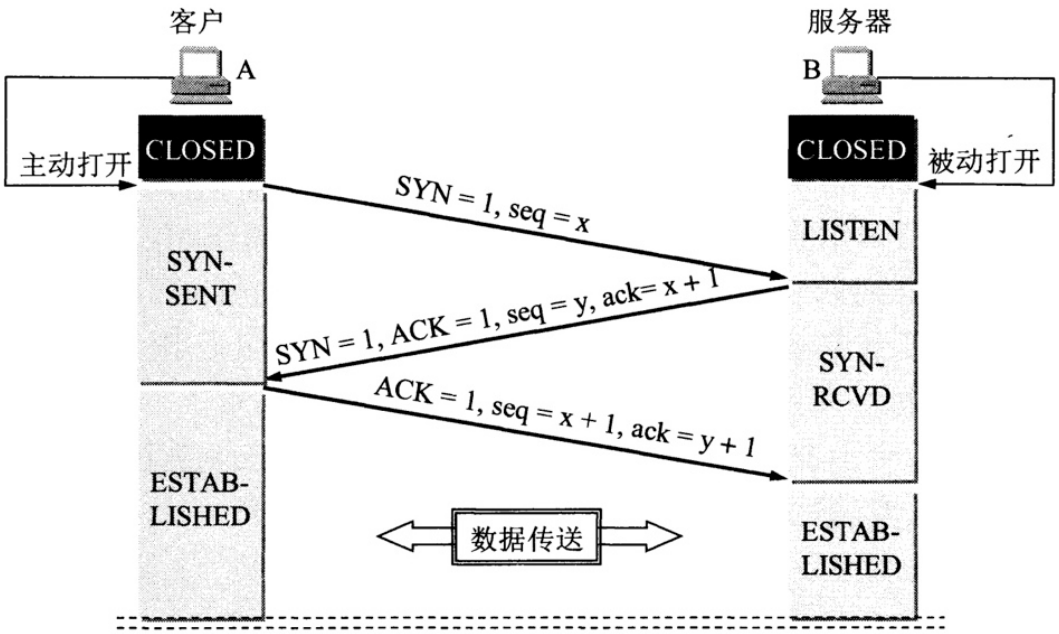
2）UDP对应的协议

- **DNS(53)**：用于域名解析服务，将域名地址转换为IP地址。DNS用的是53号端口。
- **RIP(520)**:路由信息协议，端口520
- **SNMP(161)**：简单网络管理协议，使用161号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。
- **TFTP(69)(Trival File Transfer Protocal)**，简单文件传输协议，该协议在熟知端口69上使用UDP服务。

http://www.jianshu.com/p/8be9b3204864
http://www.jianshu.com/p/eab86c0d1612

TCP的三次握手与四次挥手过程，各个状态名称与含义，TIMEWAIT的作用

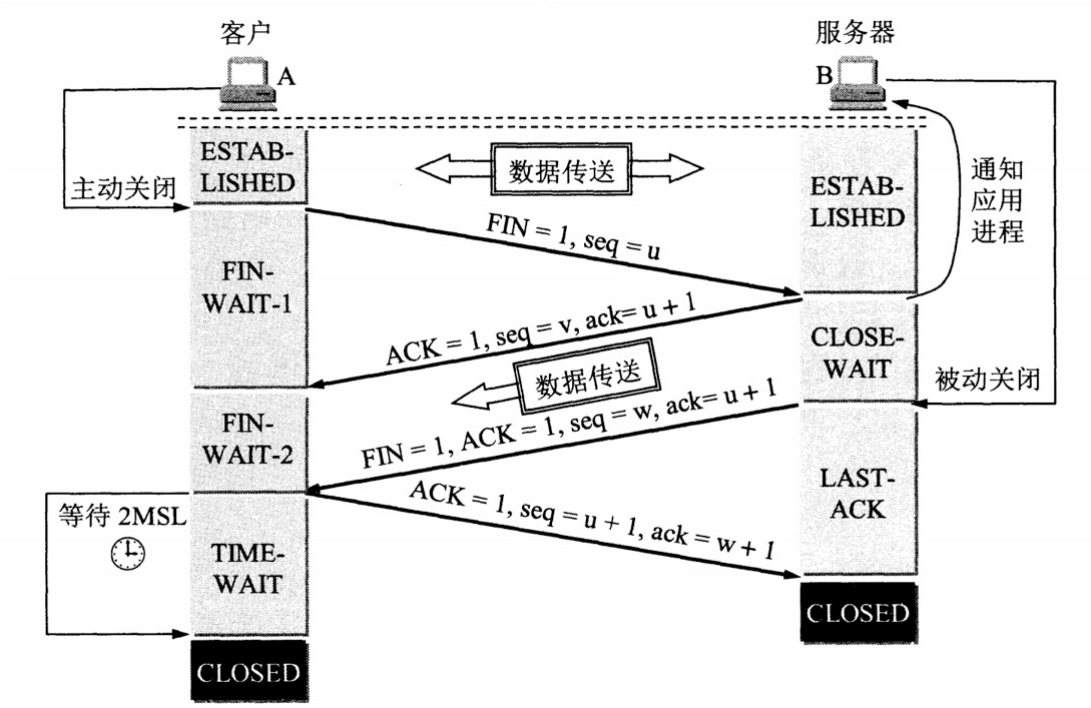
三次握手



- **第一次握手**：客户机首先向服务器的TCP发送一个连接请求报文段，这个特殊的报文段不含应用层数据，其首部中同步位SYN被设置为1。另外，客户机会随机选择一个起始序号seq=x(连接请求报文不携带数据，但要消耗一个序号)；
- **第二次握手**：服务器的TCP收到连接请求报文段后，如果同意建立连接，就向客户机发回确认，并为该TCP连接分配TCP缓存和变量。在确认报文段中，SYN和ACK位都被设置为1，确认号字段值为ack=x+1,并且服务器随机产生起始序号seq=y. 确认包同样不包含应用层数据。
- **第三次握手**：当客户机收到确认报文段后，还要向服务器给出确认，并且也要给该连接**分配缓存和变量。**这个报文段的确认位ACK被设置为1，序号段被设置为seq=x+1,确认号字段ack=y+1. 该报文段可以携带数据，如果不携带数据则不消耗序号。

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。因为TCP提供全双工通信，因此双方任何时候都可以发送数据。

四次挥手



- **第一次挥手**：客户机打算关闭连接，就向其TCP发送一个连接释放报文，并停止再发送数据，主动关闭TCP连接。该报文段的结束标志位FIN被设置为1，seq=u,它等于前面已经发送过的数据的最后一个字节的序号加1。
- **第二次挥手**：服务器收到连接释放报文段后即发出确认，确认号是ack=u+1,序号为v,等于它前面已经发送过的数据的最后一个字节序号加1.此时客户机到服务器这个方向的连接就释放了，TCP处于半关闭状态。ACK=1，seq=v,ack=u+1
- **第三次挥手**：若服务器已经没有要向客户机发送的数据，就通知TCP释放连接，此时发出FIN=1，确认号ack=u+1,序号seq=w,已经发送过的数据最后一个字节加1。确认为ACK=1.(FIN=1, ACK=1, seq=w, ack=u+1)
- **第四次挥手**：客户机收到连接释放报文段后，必须发出确认。在确认报文段中，确认位ACK=1，序号seq=u+1,确认号ack=w+1. 此时连接还没有释放掉，必须经过时间等待计时器设置的时间2MSL(Max Segment Lifetime),后，客户机才进入连接关闭状态。(ACK=1,seq=u+1,ack=w+1)

http://www.cnblogs.com/Jessy/p/3535612.html

为什么会采用三次握手，若采用二次握手可以吗?

采用三次握手是为了防止失效的连接请求报文段再次传到服务器，因而产生错误。如果由于网络不稳定，虽然客户端以前发送的连接请求以到达服务方，但服务方的同意连接的应答未能到达客户端。则客户方要重新发送连接请求，若采用二次握手，服务方收到客服端重传的请求连接后，会以为是新的请求，就会发送同意连接报文，并新开进程提供服务，这样会造成服务方资源的无谓浪费。

如果只采用一次的话，客户端不知道服务端是否已经收到自己发送的数据，则会不断地发送数据。为了保证服务端能收接受到客户端的信息并能做出正确的应答而进行前两次(第一次和第二次)握手，为了保证客户端能够接收到服务端的信息并能做出正确的应答而进行后两次(第二次和第三次)握手

为什么四次挥手，主动方要等待 2 MSL后才关闭连接.

一、保证TCP协议的全双工连接能够可靠关闭.

主要为了确保对方能受到ACK信息. 如果Client直接CLOSED了，那么由于IP协议的不可靠性或者是其它网络原因，导致Server没有收到Client最后回复的ACK。那么Server就会在超时之后继续发送FIN，此时由于Client已经CLOSED了，就找不到与重发的FIN对应的连接，最后Server就会收到RST而不是ACK，Server就会以为是连接错误把问题报告给高层。所以，Client不是直接进入CLOSED，而是要保持2MSL,

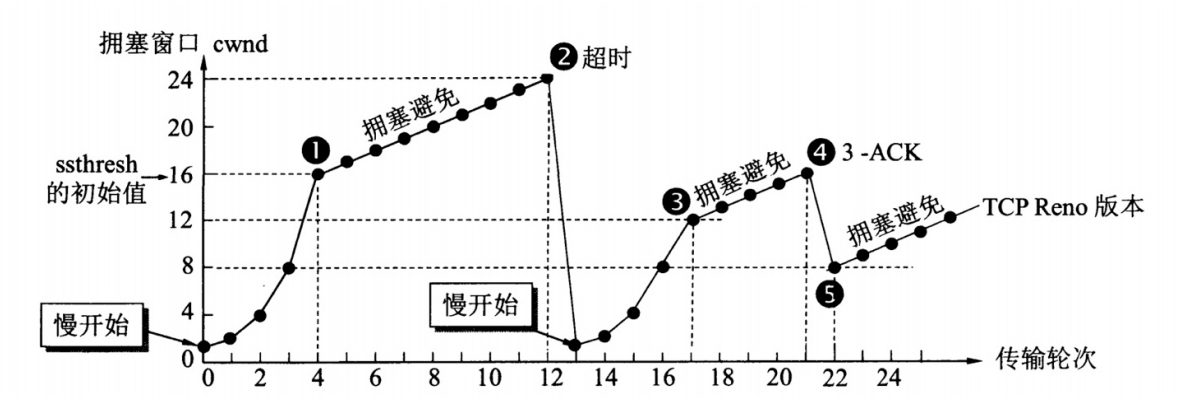
如果在这个时间内又收到了server的关闭请求时可以进行重传，否则说明server已经受到确认包则可以关闭.

https://www.zhihu.com/question/36930631

TCP拥塞控制

为了更好对TCP进行拥塞控制，因特网建议标准定义了以下四种算法：**慢开始，拥塞避免，快重传，快恢复。**
首先在TCP要求发送端维护两个窗口：

- **接收窗口 rwnd**: 接收方根据当前缓存大小所许诺的最新窗口值.
- **拥塞窗口 cwnd**: 发送方根据自己估算的网络拥塞程度而设置的窗口值。



发送窗口的上限是取这两者(接收窗口、拥塞窗口)的最小值。

- **慢开始**： TCP刚连接好时，先令拥塞窗口cwnd =1 ,在每次收到一个对新报文段的确认时将cwnd加1（因此是成倍增长2的n次方). Cwnd的大小呈指数增长。
- **拥塞避免算法**： 当cwnd大于等于慢开始门限**ssthresh**时，cwnd窗口每次加1而不是加倍。当发送方检测到**超时事件**的发生时，**就将慢开始门限ssthresh设置为当前cwnd的一半，同时将cwnd设置为1**. 这样的目的是迅速减少主机发送到网络的分组数，使得发生拥塞的路由器有足够的时间吧队列中积压的分组处理完毕。
- **快重传**：当发送方连续收到**三个重复的ACK报文时**，就立即进行重传。直接重传对方尚未收到的报文段，而不必等待那个报文段设置的重传计时器超时。
- **快恢复**：当发送端收到连续三个冗余的ACK时，发送端调整门限值 **cwnd = ssthresh = cwnd / 2**，即发送端口、阈值都等于当前窗口的一半。

TCP滑动窗口与回退N针协议

滑动窗口

发送方都维持一组连续的允许发送的帧的序号称为**发送窗口。**同时接收方也维持一组连续的允许接收的帧序号，称为**接收窗口。**发送窗口是用来对发送方进行流量控制，接收窗口是用来控制接收那些数据帧不可以接收那些帧。

在发送端，收到一个确认帧，发送窗口就向前滑动一个帧位置，当发送窗口没有可以发送的帧时，发送方就停止发送。直到接收方发送的确认帧使发送窗口向前移动。

在接收端，只有收到数据帧的序号落在接收窗口内才将该帧收下，否则一律丢弃。每收到一个帧后就发送回确认帧。

后退N帧协议

发送窗口大于1，接收窗口等于1.在后退N帧中，发送方不需要收到上一帧的ACK后才能发送下一帧，而是可以连续发送帧。

当接收方检测出失序信息帧后，要求发送方重发最后一个正确接收的帧之后的所有未被确认的帧。源站每发完一帧就要为该帧设置超时计时器，如果在超时时间内没有收到确认帧则进行重发。服务端会采用累积确认的方式，不是每个帧都发确认，可以连续收到好几个正确帧后发回一个确认信息。接收方因为窗口为1，所以必须按序接收数据帧，如果某个序大于当前所期望的序号时就会连续发送3个ACK确认帧，要求客户端重传失序帧。

TCP的可靠性如何保证

在TCP的连接中，数据流必须以正确的顺序送达对方。TCP的可靠性是通过确认机制来保证 安全的到达，即采用顺序编号、确认（ACK）、超时重传等来实现的。TCP在开始传送一个段时，为准备重传而首先将该段插入到发送队列之中，同时启动时钟。其后，如果收到了接受端对该段的ACK信息，就将该段从队列中删去。如果在时钟规定的时间内，ACK未返回，那么就从发送队列中再次送出这个段。TCP在协议中就对数据可靠传输做了保障，握手与断开都需要通讯双方确认，数据传输也需要双方确认成功，在协议中还规定了：分包、重组、重传等规则；而UDP主要是面向不可靠连接的，不能保证数据正确到达目的地。

如何保证可靠性：

- 确认和超时重传
- 数据合理分片和排序
- 流量控制
- 拥塞控制
- 数据校验
- 对于可靠性，TCP通过以下方式进行保证：
    - 数据包校验：目的是检测数据在传输过程中的任何变化，若校验出包有错，则丢弃报文段并且不给出响应，这时TCP发送数据端超时后会重发数据；
    - 对失序数据包重排序：既然TCP报文段作为IP数据报来传输，而IP数据报的到达可能会失序，因此TCP报文段的到达也可能会失序。TCP将对失序数据进行重新排序，然后才交给应用层；
    - 丢弃重复数据：对于重复数据，能够丢弃重复数据；
    - 应答机制：当TCP收到发自TCP连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒；
    - 超时重发：当TCP发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段；
    - 流量控制：TCP连接的每一方都有固定大小的缓冲空间。TCP的接收端只允许另一端发送接收端缓冲区所能接纳的数据，这可以防止较快主机致使较慢主机的缓冲区溢出，这就是流量控制。TCP使用的流量控制协议是可变大小的滑动窗口协议。

四. 应用层

Http的报文结构

http请求由三部分组成，分别是：请求行、消息报头、请求正文:

```
方法 [空格] URL [空格] 版本
首部字段名: [空格] 值
…
首部字段名: [空格] 值
[空一行]
实体主体
```

HTTP响应也是由三个部分组成，分别是：状态行、消息报头、响应正文

```
版本[空格] 状态码 [空格] 短语
首部字段名：[空格] 值
…
首部字段名：[空格] 值
[空一行]
实体主体
```

更多http内容参考：http://www.cnblogs.com/maanshancss/p/4503385.html

Http request的几种类型

HTTP协议中共定义了八种方法或者叫"动作"来表明对Request-URI指定的资源的不同操作方式，具体介绍如下：

- **GET**：向特定的资源发出请求。
- **POST**：向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的创建和/或已有资源的修改。
- **PUT**：向指定资源位置上传其最新内容。
- **DELETE**：请求服务器删除Request-URI所标识的资源。
- **HEAD**：请求读取由URL所标志的信息的首部。
- **OPTIONS**：返回服务器针对特定资源**所支持的HTTP请求方法。**也可以利用向Web服务器发送'*'的请求来测试服务器的功能性。
- **TRACE**：回显服务器收到的请求，主要用于测试或诊断。
- **CONNECT**：HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

get提交和post提交的区别

1. 根据HTTP规范，**GET用于信息获取，而且应该是安全的和幂等的。**

- 所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET 请求一般不应产生副作用。就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改，增加数据，不会影响资源的状态。
- 幂等的意味着对同一URL的多个请求应该返回同样的结果。

2. 根据HTTP规范，**POST表示可能修改变服务器上的资源的请求。**还是新闻以网站为例，读者对新闻发表自己的评论应该通过POST实现，因为在评论提交后站点的资源已经不同了，或者说资源被修改了。
3. GET请求的数据会附在URL之后（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，参数之间以&相连.POST把提交的数据则放置在是HTTP包的包体中。
4. **GET安全性较低，POST安全性较高。**因为GET在传输过程，数据被放在请求的URL中，而如今现有的很多服务器、代理服务器或者用户代理都会将请求URL记录到日志文件中，然后放在某个地方，这样就可能会有一些隐私的信息被第三方看到。另外，用户也可以在浏览器上直接看到提交的数据，一些系统内部消息将会一同显示在用户面前。
5. **get传送的数据量较小，最多2083个字符。post传送的数据量较大**，一般被默认为不受限制。
6. 在FORM（表单）中，Method默认为"GET"

> 幂等（idempotent、idempotence）是一个数学或计算机学概念，常见于抽象代数中。幂等有一下几种定义：
>
> - 对于单目运算，如果一个运算对于在范围内的所有的一个数多次进行该运算所得的结果和进行一次该运算所得的结果是一样的，那么我们就称该运算是幂等的。比如绝对值运算就是一个例子，在实数集中，有abs(a)=abs(abs(a))。
> - 对于双目运算，则要求当参与运算的两个值是等值的情况下，如果满足运算结果与参与运算的两个值相等，则称该运算幂等，如求两个数的最大值的函数，有在在实数集中幂等，即max(x,x) = x。

详见: http://www.cnblogs.com/hyddd/archive/2009/03/31/1426026.html
http://www.cnblogs.com/skynet/archive/2010/05/18/1738301.html

http常见错误码

- 2xx 成功：
  - **200 OK**：一切正常
- 3xx 重定向：
  - 301 Moved Permanently: 客户请求的文档在其他地方
- 4xx 客户机中出现的错误
  - **400 Bad Request**: 请求出现语法错误。
  - **401 Unauthorized**: 客户试图未经授权访问受密码保护的页面。
  - **403 Forbidden**: 资源不可用,服务器理解客户的请求，但拒绝处理它。通常由于服务器上文件或目录的权限设置导致。
  - **404 Not Found**: 无法找到指定位置的资源。这也是一个常用的应答。
- 5xx 服务器中出现的错误:
  - **500 Internal Server Error**: 服务器遇到了意料不到的情况，不能完成客户的请求。
  - **501 Not Implemented** 服务器不支持实现请求所需要的功能。例如，客户发出了一个服务器不支持的PUT请求。
  - **502 Bad Gateway**: 服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答。
  - **503 Service Unavailable**: 服务器由于维护或者负载过重未能应答。
  - **504 Gateway Timeout**: 由作为代理或网关的服务器使用，表示不能及时地从远程服务器获得应答。
  - **505 HTTP Version Not Supported**:服务器不支持请求中所指明的HTTP版本。（HTTP 1.1新）

Http1.1和Http1.0的区别

- **HTTP1.0 是短连接，HTTP1.1是长连接。** HTTP 1.0规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。如果一个html包含多个图片或资源时需要多次与服务器建立连接。 HTTP 1.1支持持久连接，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟。一个包含有许多图像的网页文件的多个请求和应答可以在一个连接中传输，但每个单独的网页文件的请求和应答仍然需要使用各自的连接.**HTTP 1.1采用了流水线的持久连接**，即客户端不用等待上一次请求结果返回，就可以发出下一次请求，但服务器端必须按照接收到客户端请求的先后顺序依次回送响应结果，以保证客户端能够区分出每次请求的响应内容，这样也显著地减少了整个下载过程所需要的时间。
- **HTTP 1.0不支持Host请求头字段**，WEB浏览器无法使用主机头名来明确表示要访问服务器上的哪个WEB站点，这样就无法使用WEB服务器在同一个IP地址和端口号上配置多个虚拟WEB站点。在HTTP 1.1中增加Host请求头字段后，WEB浏览器可以使用主机头名来明确表示要访问服务器上的哪个WEB站点，这才实现了在一台WEB服务器上可以在同一个IP地址和端口号上使用不同的主机名来创建多个虚拟WEB站点。
- HTTP 1.1还提供了与**身份认证、状态管理和Cache缓存等机制**相关的请求头和响应头
- **带宽优化。**HTTP/1.0中，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了。例如，客户端只需要显示一个文档的部分内容，又比如下载大文件时需要支持断点续传功能，而不是在发生断连后不得不重新下载完整的包。HTTP/1.1中在请求消息中引入了range头域，它允许只请求资源的某个部分。
- HTTP/1.1增加了**OPTIONS**方法，它允许客户端获取一个服务器支持的方法列表

http://blog.csdn.net/forgotaboutgirl/article/details/6936982

Http怎么处理长连接

HTTP1.1和HTTP1.0相比较而言，最大的区别就是增加了持久连接支持。TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了带宽。

使用长连接的HTTP协议，会在响应头有加入这行代码：**Connection:keep-alive**。在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的 TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现长连接要客户端和服务端都支持长连接。

http://www.codeceo.com/article/http-long-connect.html

Cookie与Session的作用于原理

Cookie

会话跟踪是Web程序中常用的技术，用来跟踪用户的整个会话。常用的会话跟踪技术是Cookie与Session。Cookie通过在客户端记录信息确定用户身份，Session通过在服务器端记录信息确定用户身份。

Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要**记录该用户状态**，就产生一个用户身份标识，然后在响应消息中将该标识号以Cookie的形式传递给浏览器.客户端浏览器会把Cookie保存起来。浏览器在以后每次访问该web服务器时，浏览器把请求的网址连同该Cookie一同提交给服务器。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容. Cookie不能被浏览器共享
Cookie具有不可跨域名性，例如浏览器访问Google只会携带Google的Cookie，而不会携带Baidu的Cookie。Cookie的maxAge决定着Cookie的有效期，单位为秒（Second）.

默认情况下,cookie是一个会话级别的,用户退出浏览器后被删除

Session

Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。
如果说Cookie机制是通过检查客户身上的“通行证”来确定客户身份的话，那么Session机制就是通过检查服务器上的“客户明细表”来确认客户身份。Session相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。

Session保存在服务器端。为了获得更高的存取速度，服务器一般把Session放在内存里。每个用户都会有一个独立的Session。如果Session内容过于复杂，当大量客户访问服务器时可能会导致内存溢出。因此，Session里的信息应该尽量精简。

session工作原理:

- web不会在客户端开始访问它时创建session,在访问特殊的程序并且该程序(servlet)决定与客户端开启会话时,服务器生成一个唯一值,称为Session ID（好像是通过取进程ID的方式取得的）。服务器开辟一块内存，对应于该Session ID。
- 服务器再将该Session ID写入浏览器的cookie。
- 服务器内有一进程，监视所有Session的活动状况，如果有Session超时或是主动关闭，服务器就释放改内存块。
- 当浏览器连入服务器时并请求Session时，服务器就读浏览器Cookie中的Session ID。
- 然后，服务检查该Session ID所对应的内存是否有效。
- 如果有效，就读出内存中的值。

- 如果无效，就建立新的Session。

tomcat默认设置是30分钟,具体设置很简单，方法有三种：

（1）在主页面或者公共页面中加入：session.setMaxInactiveInterval(900);
参数900单位是秒，即在没有活动15分钟后，session将失效。设置为-1将永不关闭。
这里要注意这个session设置的时间是根据服务器来计算的，而不是客户端。所以如果是在调试程序，应该是修改服务器端时间来测试，而不是客户端。

（2）也是比较通用的设置session失效时间的方法，就是在项目的web.xml中设置

```
<session-config>
<session-timeout>15</session-timeout>
</session-config>
```

这里的15也就是15分钟失效.

（3）直接在应用服务器中设置，如果是tomcat，可以在tomcat目录下conf/web.xml中
找到元素，tomcat默认设置是30分钟，只要修改这个值就可以了。
需要注意的是如果上述三个地方如果都设置了，有个优先级的问题，从高到低：

（1）-->(2)--->(3)

生命周期:

- 1)tomcat默认是30分钟
- 2)因为session和cookie有关系，客户端浏览器通过cookie维持该会话，cookie不支持跨浏览器共享，意味着如果把浏览器关该会闭的话该会话就会失效(但服务端还保存着)。因此同一机器的两个浏览器窗口访问服务器时，会生成两个不同的Session。

http://qiusuoge.com/10576.html
http://blog.csdn.net/colzer/article/details/8686966
http://www.cnblogs.com/binger/archive/2013/03/19/2970171.html

电脑上访问一个网页，整个过程是怎么样的：DNS、HTTP、TCP、OSPF、IP、ARP

- 1) 浏览器分析连接指向的页面URL(http://www.baidu.com)
- 2) 浏览器向DNS请求www.baidu.com.的IP地址
- 3) 域名系统DNS解析出百度官网的服务器IP地址
- 4) 浏览器与该服务器建立TCP连接（默认端口80）
- 5) 浏览器发出HTTP请求获取指定页面。
- 6) 服务器通过HTTP响应把文件对应页面发送给浏览器。
- 7) TCP连接释放。
- 8) 浏览器将文件进行解析，并将web网页显示给用户。

https如何加密的

HTTPS介绍：
HTTPS(Secure Hypertext Transfer Protocol)安全超文本传输协议 它是一个安全通信通道，它基于HTTP开发，用于在客户计算机和服务器之间交换信息。**它使用安全套接字层(SSL)进行信息交换**，简单来说它是HTTP的安全版。 它是由Netscape开发并内置于其浏览器中，使用SSL在发送方把原始数据进行加密，然后在接受方进行解密保证数据的安全性．然而，加密和解密过程需要耗费系统大量的开销，严重降低机器的性能.

HTTPS实际上就是SSL over HTTP，它使用默认端口**443**，而不是像HTTP那样使用端口80来和TCP/IP进行通信。HTTPS协议使用SSL在发送方把原始数据进行加密，然 后在接受方进行解密，加密和解密需要发送方和接受方通过交换共知的密钥来实现，因此，所传送的数据不容易被网络黑客截获和解密.然而，加密和解密过程需要耗费系统大量的开销，严重降低机器的性能，相关测试数据表明使用HTTPS协议传输数据的工作效率只有使用HTTP协议传输的十分之一。

如何实现加密：（其实就是sssl的交互过程）
HTTPS其实是有两部分组成：HTTP + SSL / TLS，也就是在HTTP上又加了一层处理加密信息的模块。服务端和客户端的信息传输都会通过TLS进行加密，所以传输的数据都是加密后的数据。具体是如何进行加密，解密，验证的

http://blog.csdn.net/binyao02123202/article/details/8049446
http://www.codeceo.com/article/why-http-better-than-https.html

http和https的区别

1. https协议需要到ca申请证书，一般免费证书很少，需要交费.
2. 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。
3. 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 要比http协议安全 HTTPS解决的问题.

http://www.admin5.com/article/20150708/608526.shtml

SSL协议及完整交互过程

SSL是Netscape公司所提出的安全保密协议，在浏览器(如Internet Explorer、Netscape Navigator)和Web服务器(如Netscape的Netscape Enterprise Server、ColdFusion Server等等)之间构造安全通道来进行数据传输，SSL运行在TCP/IP层之上、应用层之下，为应用程序提供加密数据通道，它采用了RC4、MD5 以及RSA等加密算法，使用40 位的密钥，适用于商业信息的加密。

开始加密通信之前，客户端和服务器首先必须建立连接和交换参数，这个过程叫做握手（handshake）。

1 客户端发出请求（ClientHello）

首先，客户端（通常是浏览器）先向服务器发出加密通信的请求，这被叫做ClientHello请求。在这一步，客户端主要向服务器提供以下信息。

- （1） 支持的协议版本，比如TLS 1.0版。
- （2） 一个客户端生成的随机数，稍后用于生成"对话密钥"。
- （3） 支持的加密方法，比如RSA公钥加密。
- （4） 支持的压缩方法。

2 服务器回应（SeverHello）

服务器收到客户端请求后，向客户端发出回应，这叫做SeverHello。服务器的回应包含以下内容：

- （1） 确认使用的加密通信协议版本，比如TLS 1.0版本。如果浏览器与服务器支持的版本不一致，服务器关闭加密通信。
- （2） 一个服务器生成的随机数，稍后用于生成"对话密钥"。
- （3） 确认使用的加密方法，比如RSA公钥加密。
- （4） 服务器证书。（公钥）

3 客户端回应

客户端收到服务器回应以后，首先验证服务器证书。如果证书不是可信机构颁布、或者证书中的域名与实际域名不一致、或者证书已经过期，就会向访问者显示一个警告，由其选择是否还要继续通信。
如果证书没有问题，客户端就会从证书中取出服务器的公钥。然后，向服务器发送下面三项信息。

- 一个随机数。该随机数用服务器公钥加密，防止被窃听。
- 编码改变通知，表示随后的信息都将用双方商定的加密方法和密钥发送.
- 客户端握手结束通知，表示客户端的握手阶段已经结束。这一项同时也是前面发送的所有内容的hash值，用来供服务器校验。

上面第一项的随机数，是整个握手阶段出现的第三个随机数，又称"pre-master key"。有了它以后，客户端和服务器就同时有了三个随机数，接着双方就用事先商定的加密方法，各自生成本次会话所用的同一把"会话密钥"。（会话秘钥是采用对称加密方式，而这里的公钥是采用非对称加密）

4 服务器的最后回应

服务器通过私钥解密收到客户端的第三个随机数pre-master key之后，计算生成本次会话所用的"会话密钥"。然后，向客户端最后发送下面信息。

- 编码改变通知，表示随后的信息都将用双方商定的加密方法和密钥发送。
- 服务器握手结束通知，表示服务器的握手阶段已经结束。这一项同时也是前面发送的所有内容的hash值，用来供客户端校验。

至此，整个握手阶段全部结束。接下来，客户端与服务器进入加密通信，就完全是使用普通的HTTP协议，只不过用"会话密钥"加密内容。

至于为什么一定要用三个随机数，来生成"会话密钥"，dog250解释得很好：

"不管是客户端还是服务器，都需要随机数，这样生成的密钥才不会每次都一样。由于SSL协议中证书是静态的，因此十分有必要引入一种随机因素来保证协商出来的密钥的随机性。

对于RSA密钥交换算法来说，pre-master-key本身就是一个随机数，再加上hello消息中的随机，三个随机数通过一个密钥导出器最终导出一个对称密钥。

pre master的存在在于SSL协议不信任每个主机都能产生完全随机的随机数，如果随机数不随机，那么pre master secret就有可能被猜出来，那么仅适用pre master secret作为密钥就不合适了，因此必须引入新的随机因素，那么客户端和服务器加上pre master secret三个随机数一同生成的密钥就不容易被猜出了，一个伪随机可能完全不随机，可是是三个伪随机就十分接近随机了，每增加一个自由度，随机性增加的可不是一。"

http://limboy.me/tech/2011/02/19/https-workflow.html
http://www.ruanyifeng.com/blog/2014/02/ssl_tls.html
http://www.ruanyifeng.com/blog/2014/09/illustration-ssl.html
http://kb.cnblogs.com/page/162080/

https处理的一个过程，对称加密和非对称加密（参考上一条，这一条就是通俗的表示）

1. SSL原理很简单。当你的浏览器向服务器请求一个安全的网页(通常是 https://)
2. 服务器就把它的证书和公匙发回来
3. 浏览器检查证书是不是由可以信赖的机构颁发的，确认证书有效和此证书是此网站的。
4. 浏览器使用公钥加密了一个随机对称密钥，包括加密的URL一起发送到服务器。
5. 服务器用自己的私匙解密了你发送的钥匙。然后用这把对称加密的钥匙给你请求的URL链接解密。
6. 服务器用你发的对称钥匙给你请求的网页加密。你也有相同的钥匙就可以解密发回来的网页了

加密算法绝大部分都属于以下两种加密类型之一：

- 对称加密：加密解密用的是同样的“钥匙”

- 非对称加密：加密解密用的是不同的“钥匙”

对称加密

Alice 在盒子里放有信息，盒子上有挂锁，她有钥匙。通过邮局她把这个盒子寄给Bob。Bob收到盒子后，用相同的钥匙打开盒子（钥匙之前就得到了，可能是Alice面对面给他的）。然后Bob可以用同样的方法回复。

非对称加密

Bob和Alice各有自己的盒子。Alice要跟Bob秘密通信，她先让Bob把开着的盒子通过邮局发给她。Alice拿到盒子后放入信息锁上，然后发给Bob。Bob就可以用他自己的钥匙打开了。回复的话就用同样的方法。
非对称算法在加密和解密时用的是不同的钥匙。信息接受者有两把钥匙：一把“公匙”，一把“私匙”。公匙是给信息发送者用来加密的，私匙是自己用来解密的。这样最大的好处是：不必通过不安全的渠道发送私密的东西。公匙本来就是给别人用的，不用藏好。你的私匙在你产生私匙的电脑里保存着。

http://article.yeeyan.org/view/90729/174903/

Http和https的区别

- http是HTTP协议运行在TCP之上。所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份
- https是HTTP运行在SSL/TLS之上，SSL/TLS运行在TCP之上。所有传输的内容都经过加密，加密采用对
- 加密，但对称加密的密钥用服务器方的证书进行了非对称加密。此外客户端可以验证服务器端的身份
- 如果配置了客户端验证，服务器方也可以验证客户端的身份。
- https协议需要到ca申请证书，一般免费证书很少，需要交费。
- 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 要比http协议安全

https://github.com/Wl201314/MartinBlog/tree/master/Blog/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%9F%BA%E7%A1%80

DNS域名系统，简单描述其工作原理。

域名系统DNS是因特网使用的命名系统，用于把便于人们记忆的含有特定含义的主机名转换为便于机器处理的IP地址。DNS系统采用C/S模式，其协议运行在UDP上，使用53号端口。从概念上讲DNS可以分为3个部分：层次域名空间，域名服务器和解析器。

域名解析是把域名映射成IP地址或者把IP地址映射成域名的过程。域名解析有两种方式：递归查询和迭代查询。

递归查询：如果本地主机询问的本地域名服务器不知道被查询的域名IP地址，那么**本地域名服务器**就以DNS客户的身份，向**根域名服务器**继续发出查询请求报文，而不是自己进行下一步非查询。根域名服务器会继续向**顶级域名服务器**查询请求，如果存在则直接返回给根服务器，否则顶级域名服务器向对应的**权限域名服务器**查询。是一种递归的查询方式。

迭代查询：本地域名服务器负责转发和调用功能。如果本地主机询问的本地域名服务器不知道被查询的域名IP地址，那么本地域名服务器就以DNS客户的身份，向根域名服务器继续发出查询请求报文。跟服务器收到请求报文时，要么给出所有查询的IP地址，要么告诉本地域名服务器下一步需要查询的顶级域名服务器。然后本地服务器向这个顶级域名服务器进行后续的查询。同样的顶级域名服务器收到查询报文后，要么给出所要查询的IP地址，要么告诉本地域名服务器下一步应该查找那个权限域名服务器。

面向连接和非面向连接的服务的特点是什么?

面向连接的服务：通信双方进行通信之前，必须先建立连接，在通信过程中，整个连接情况一直都是被实时地监控和管理。当通信结束后，则应释放这个连接。
无连接服务：两个实体之间的通信不需要先建立好连接，需要通信的时候，直接将信息发送到"网络"中，让该信息的传递在网上尽力而为地往目的地传送。

端口及对应的服务?

- FTP 21
- SSH 22
- telnet 23
- SMTP 25
- Domain(域名服务器) 53
- HTTP 80
- POP3 110
- NTP（网络时间协议）123
- MySQL数据库服务3306
- https 443
- SNMP(161)
- TFTP(69)
- Shell或 cmd 514
- POP-2 109
- SQL Server 1433

各种协议

- **ICMP协议**： 因特网控制报文协议。它是TCP/IP协议族的一个子协议，用于在IP主机、路由器之间传递控制消息。
- **TFTP协议**： 是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。
- **HTTP协议**： 超文本传输协议，是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。
- **DHCP协议**： 动态主机配置协议，是一种让系统得以连接到网络上，并获取所需要的配置参数手段。
- **NAT协议**：网络地址转换属接入广域网(WAN)技术，是一种将私有（保留）地址转化为合法IP地址的转换技术，
- **DHCP协议**：一个局域网的网络协议，使用UDP协议工作，用途：给内部网络或网络服务供应商自动分配IP地址，给用户或者内部网络管理员作为对所有计算机作中央管理的手段。

Ping的整个过程

同一个局域网中：

1. Pc1在应用层发起个目标IP位IP2的Ping请求。
2. 传输层接到上层请求的数据，将数据分段并加上UDP报头。下传到Internet层。
3. 网际层接收来处上层的数据后，根据ICMP协议进行封装，添加PC1的IP为源IP为和PC2IP为目标IP后封装成数据包。下传到网络接口层。
4. 网络接口层接收数据包后，进行封装，源MAC地址为PC1的MAC地址，目标MAC地址则查询自己的ARP缓存表获取。如果PC1 arp缓存表中没有目标IP对应的MAC地址，则PC1发出一个ARP广播报文。ARP报文中源MAC地址为Pc1mac地址，源IP地址为pc1 IP，所要请求的是PC2的IP对应的mac地址。

5. PC2收到ARP广播后，进行解封装，发现所请求的MAC地址是自己的。则PC2将PC1的mac地址写入arp缓存表中。然后向PC1发送一个 ARP应答单播。该单播消息包括目标IP为PC1ip，目标Mac为pc1mac地址，源IP为PC2的IP，源Mac为pc2的Mac。
6. Pc1接收到PC2的arp应答报文后，将Pc2的MAC地址存入arp缓存中，并将Pc2的Mac地址作为目标地址封装到数据帧中。发给下层进行网络传输。
7. PC2接收这个帧后，在网络接口层查看目标mac地址是否指向自己。是，PC2则将帧头去掉，向上层传输。
8. Pc2网际层接收到这个信息包，查看包头，发现目标IP和自己匹配，则解封装，将数据向上层传输。
9. 传输层接收来自下层的Ping请求的UDP报文，则去掉UDP报头，向应用层传送。
10. 应用层收到ping请求后，发送一个PIng回应报文给PC1

http://blog.chinaunix.net/uid-26758209-id-3146224.html

五. 补充

对于socket编程，accept方法是干什么的

三次握手：

- 服务器调用listen进行监听
- 客户端调用connect来发送syn报文
- 服务器协议栈负责三次握手的交互过程。连接建立后，往listen队列中添加一个成功的连接，直到队列的最大长度。
- 服务器调用accept从listen队列中取出一条成功的tcp连接，listen队列中的连接个数就少一个，然后程序就可以看到自己与用户的通信

select, epoll（IO多路复用）

- 阻塞IO模型

  最传统的一种IO模型，即在读写数据过程中会发生阻塞现象。

  当用户线程发出IO请求之后，内核会去查看数据是否就绪，如果没有就绪就会等待数据就绪，而用户线程就会处于阻塞状态，用户线程交出CPU。当数据就绪之后，内核会将数据拷贝到用户线程，并返回结果给用户线程，用户线程才解除block状态。

- select：同时监听多个socket，从用户空间将fd_set拷贝到内核，对所有的fd进行一次poll操作，即把当前进程挂载到fd上，检查是否有事件触发，无则休眠，再恢复做poll，直到有设备有事件触发，返回用户空间，对相关fd进行读或者写操作

- epoll：epoll不需要通过遍历的方式，而是在内核中建立了file节点，并且通过注册响应事件的方式，当有响应事件发生时采取相应的措施，并把准备就绪的事件放入链表中，从而epoll只关心链表中是否有数据即可。

  epoll的触发模式

  - 水平触发：就是与select和poll类似，当被监控的文件描述符上有可读写事件发生时，epoll_wait()会通知处理程序去读写。如果这次没有把数据一次性全部读写完(如读写缓冲区太小)，那么下次调用 epoll_wait()时，它还会通知你在上次没读写完的文件描述符上继续读写
  - 边缘触发：只告诉进程哪些文件描述符刚刚变为就绪状态，它只说一遍，如果我们没有采取行动，那么它将不会再次告知，这种方式称为边缘触发，理论上边缘触发的性能要更高一些，但是代码实现相当复杂。

socket中recv函数的返回值及意义

1. recv先等待socket的发送缓冲中的数据被协议传送完毕，如果协议在传送socket的发送缓冲中的数据时出现网络错误，那 么recv函数返回SOCKET_ERROR。

2. 如果socket的发送缓冲中没有数据或者数据被协议成功发送完毕后，recv先检查套接字socket的接收缓冲区，如果接收缓冲区中没有数据或者协议正在接收数据，那么recv就一直等待，直到协议把数据接收完毕。当协议把数据接收完毕，rec函数就把socket的接收缓冲中的数据copy到buffer中（注意协议接收到的数据可能大于buf的长度，所以 在这种情况下要调用几次recv函数才能把socket的接收缓冲中的数据copy完。recv函数仅仅是copy数据，真正的接收数据是协议来完成的），rec函数返回其实际copy的字节数。如果recv在copy时出错，那么它返回SOCKET_ERROR；如果recv函数在等待协议接收数据时网络中断了，那么它返回0。