# 计算机网络

## socket 中的长连接和短连接

**短连接**

连接->传输数据->关闭连接

HTTP是无状态的，浏览器和服务器每进行一次HTTP操作，就建立一次连接，但任务结束就中断连接。

也可以这样说：短连接是指SOCKET连接后发送后接收完数据后马上断开连接。

**长连接**

连接->传输数据->保持连接 -> 传输数据-> 。。。 ->关闭连接。

长连接指建立SOCKET连接后不管是否使用都保持连接，但安全性较差。

**http的长连接**

HTTP也可以建立长连接的，使用Connection:keep-alive，HTTP 1.1默认进行持久连接。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(貌似最新的 http1.0 可以显示的指定 keep-alive),但还是无状态的，或者说是不可以信任的。

**什么时候用长连接，短连接？**

 长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况，。每个TCP连接都需要三步握手，这需要时间，如果每个操作都是先连接，再操作的话那么处理速度会降低很多，所以每个操作完后都不断开，次处理时直接发送数据包就OK了，不用建立TCP连接。例如：数据库的连接用长连接， 如果用短连接频繁的通信会造成socket错误，而且频繁的socket 创建也是对资源的浪费。

而像WEB网站的http服务一般都用短链接，因为长连接对于服务端来说会耗费一定的资源，而像WEB网站这么频繁的成千上万甚至上亿客户端的连接用短连接会更省一些资源，如果用长连接，而且同时有成千上万的用户，如果每个用户都占用一个连接的话，那可想而知吧。所以并发量大，但每个用户无需频繁操作情况下需用短连好。

总之，长连接和短连接的选择要视情况而定。

## 七层与四层协议

**OSI分层 （7层）**：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

**TCP/IP分层（4层）**：网络接口层、 网际层、运输层、 应用层。

**五层协议 （5层）**：物理层、数据链路层、网络层、运输层、 应用层。

**每一层的协议如下：**

物理层：RJ45、CLOCK、IEEE802.3 （中继器，集线器，网关）

数据链路：PPP、FR、HDLC、VLAN、MAC （网桥，交换机）

网络层：**IP、ICMP、ARP、RARP、OSPF、IPX、RIP、IGRP**（路由器）

传输层：TCP、UDP、SPX

会话层：NFS、SQL、NETBIOS、RPC

表示层：JPEG、MPEG、ASII

应用层：**FTP、DNS、Telnet、SMTP、HTTP、WWW、NFS**

**每一层的作用如下：**

**物理层：**通过媒介传输比特,确定机械及电气规范（比特Bit） 不可靠的

**数据链路层**：将比特组装成帧和点到点的传递（**帧Frame**） 可靠的

**网络层**：负责数据包从源到宿的传递和网际互连（**包PackeT**）

**传输层**：提供端到端的可靠报文传递和错误恢复（**段Segment**）

**会话层**：建立、管理和终止会话（会话协议数据单元SPDU）3

**表示层**：对数据进行翻译、加密和压缩（表示协议数据单元PPDU）

**应用层**：允许访问OSI环境的手段（应用协议数据单元APDU） 直接为用户的应用进程提供服务

为什么会有TCP/IP四层模型？

OSI七层模型是一种框架性的设计方法，建立七层模型的主要目的是解决异种网络互连时出现的兼容性问题。最主要的功能是使不同类型的主机实现数据传输。最大的优点是将服务 接口和协议三个概念区分开来。

服务：下层为上层提供的功能

接口：上层如何使用下层的服务

协议：如何实现本层的服务

便于研究和教学。

网络接口层：TCP/IP参考模型并没有真正描述这一层的实现。只是要求能够提供给其上层——网际层一个访问接口，以便在其上传递IP分组。

网际层：是整个TCP/IP协议栈的核心。将分组发到目标网络或主机。为了尽快发送分组，可能需要沿不同路径进行传递，这就需要上层进行排序。除了路由功能外，还能将不同类型的网络互连。

运输层：传输层的功能是使源端主机和目标端主机上的对等实体可以进行会话。在传输层定义了两种服务质量不同的协议。即：传输控制协议TCP（transmission control protocol）和用户数据报协议UDP（user datagram protocol）。　　  
　　TCP协议是一个面向连接的、可靠的协议。它将一台主机发出的字节流无差错地发往互联网上的其他主机。在发送端，它负责把上层传送下来的字节流分成报文段并传递给下层。在接收端，它负责把收到的报文进行重组后递交给上层。TCP协议还要处理端到端的流量控制，以避免缓慢接收的接收方没有足够的缓冲区接收发送方发送的大量数据。UDP协议是一个不可靠的、无连接协议，主要适用于不需要对报文进行排序和流量控制的场合。

应用层：TCP/IP模型将OSI参考模型中的会话层和表示层的功能合并到应用层实现。应用层面向不同的网络应用引入了不同的应用层协议。其中，有基于TCP协议的，如文件传输协议（File Transfer Protocol，FTP）、虚拟终端协议（TELNET）、超文本链接协议（Hyper Text Transfer Protocol，HTTP），也有基于UDP协议的。

**对于TCP/IP网络，更加关注的是IP和TCp层，故将其他层直接合并在一起了。Osi七层模型有点冗余。**

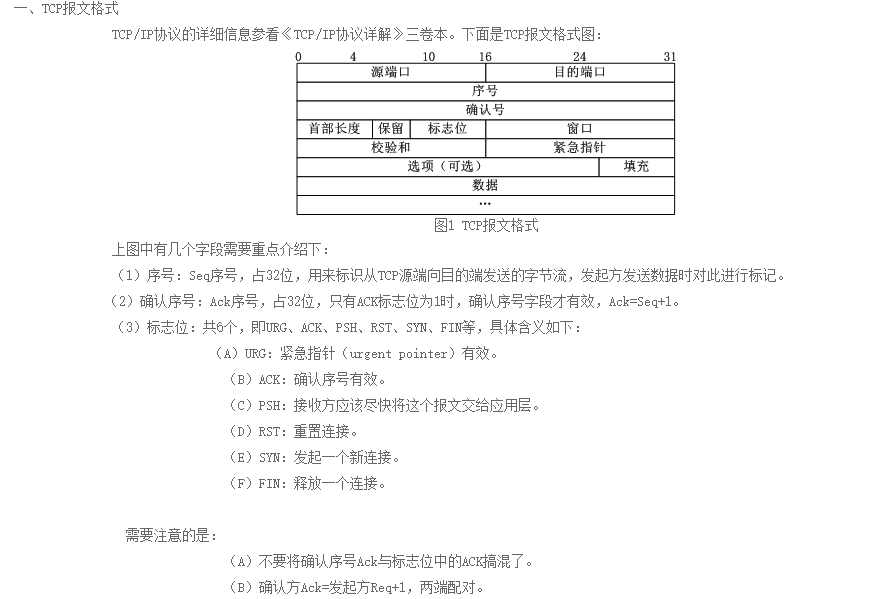
## TCP三次握手和四次挥手

<http://blog.csdn.net/whuslei/article/details/6667471/>

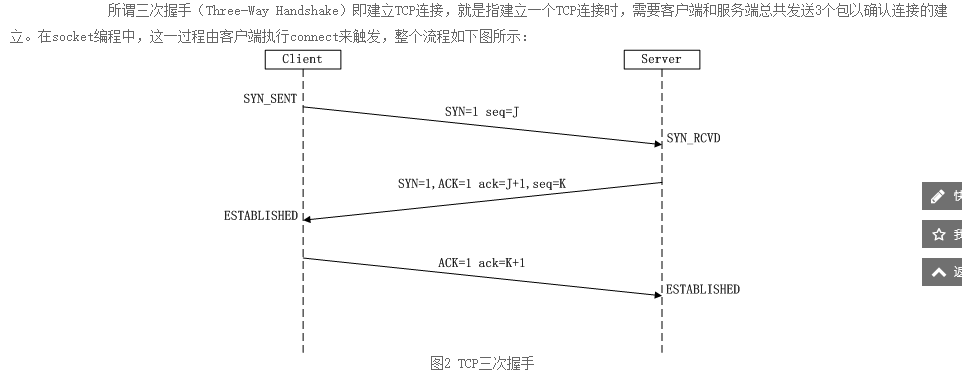
http://www.cnblogs.com/zmlctt/p/3690998.html

为什么需要三次握手和四次挥手

<http://blog.csdn.net/xifeijian/article/details/12777187>



**三次握手：**

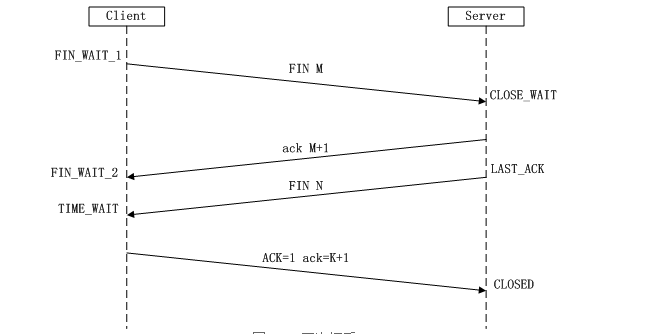


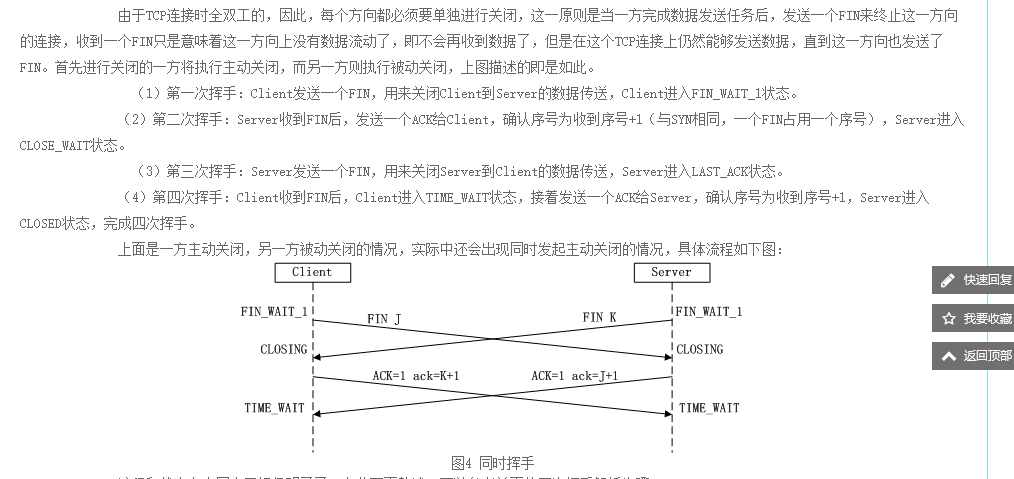
（1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。  
（2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。  
        （3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数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

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。

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

**四次挥手**





与建立连接的“三次握手”类似，断开一个TCP连接则需要“四次握手”。

第一次挥手：主动关闭方发送一个FIN，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不 会再给你发数据了(当然，在fin包之前发送出去的数据，如果没有收到对应的ack确认报文，主动关闭方依然会重发这些数据)，但是，此时主动关闭方还可 以接受数据。

第二次挥手：被动关闭方收到FIN包后，发送一个ACK给对方，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号）。

第三次挥手：被动关闭方发送一个FIN，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。

第四次挥手：主动关闭方收到FIN后，发送一个ACK给被动关闭方，确认序号为收到序号+1，至此，完成四次挥手。

## ARP地址解析协议

是根据IP地址获取物理地址的一个TCP/IP协议。

1：首先，每个主机都会在自己的ARP缓冲区中建立一个ARP列表，以表示IP地址和MAC地址之间的对应关系。

2：当源主机要发送数据时，首先检查ARP列表中是否有对应IP地址的目的主机的MAC地址，如果有，则直接发送数据，如果没有，就向本网段的所有主机发送ARP数据包，该数据包包括的内容有：**源主机** **IP地址，源主机MAC地址，目的主机的IP 地址**。

3：当本网络的所有主机收到该ARP数据包时，首先检查数据包中的IP地址是否是自己的IP地址，如果不是，则忽略该数据包，如果是，则首先从数据包中取出源主机的IP和MAC地址写入到ARP列表中，如果已经存在，则覆盖，然后将自己的MAC地址写入ARP响应包中，告诉源主机自己是它想要找的MAC地址。

4：源主机收到ARP响应包后。将目的主机的IP和MAC地址写入ARP列表，并利用此信息发送数据。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

**广播发送ARP请求，单播发送ARP响应**

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，主要用于无盘工作站，因为给无盘工作站配置的IP地址不能保存。工作流程：在网络中配置一台RARP服务器，里面保存着IP地址和MAC地址的映射关系，当无盘工作站启动后，就封装一个RARP数据包，里面有其MAC地址，然后广播到网络上去，当服务器收到请求包后，就查找对应的MAC地址的IP地址装入响应报文中发回给请求者。因为需要广播请求报文，因此RARP只能用于具有广播能力的网络。

## 在浏览器中输入www.baidu.com后执行的全部过程

<http://blog.csdn.net/qq_16681169/article/details/50866290>

https://www.zhihu.com/question/27199686

1、客户端浏览器通过DNS解析到www.baidu.com的IP地址220.181.27.48，通过这个IP地址找到客户端到服务器的路径。客户端浏览器发起一个HTTP会话到220.161.27.48，然后通过TCP进行封装数据包，输入到网络层。

2、在客户端的传输层，把HTTP会话请求分成报文段，添加源和目的端口，如服务器使用80端口监听客户端的请求，客户端由系统随机选择一个端口如5000，与服务器进行交换，服务器把相应的请求返回给客户端的5000端口。然后使用IP层的IP地址查找目的端。

3、客户端的网络层不用关系应用层或者传输层的东西，主要做的是通过查找路由表确定如何到达服务器，期间可能经过多个路由器，这些都是由路由器来完成的工作，我不作过多的描述，无非就是通过查找路由表决定通过那个路径到达服务器。

4、客户端的链路层，包通过链路层发送到路由器，通过邻居协议查找给定IP地址的MAC地址，然后发送ARP请求查找目的地址，如果得到回应后就可以使用ARP的请求应答交换的IP数据包现在就可以传输了，然后发送IP数据包到达服务器的地址。

## TCP和UDP的区别

TCP提供面向连接的、可靠的数据流传输，而UDP提供的是非面向连接的、不可靠的数据流传输。

TCP传输单位称为TCP报文段，UDP传输单位称为用户数据报。

TCP注重数据安全性，UDP数据传输快，因为不需要连接等待，少了许多操作，但是其安全性却一般。

**TCP对应的协议和UDP对应的协议**

**TCP对应的协议：**

（1） **FTP**：定义了文件传输协议，使用**21**端口。

（2） **Telnet**：一种用于远程登陆的端口，使用**23**端口，用户可以以自己的身份远程连接到计算机上，可提供基于DOS模式下的通信服务。

（3） **SMTP**：邮件传送协议，用于发送邮件。服务器开放的是**25**号端口。

（4） **POP3**：它是和SMTP对应，POP3用于接收邮件。POP3协议所用的是**110**端口。

（5）**HTTP**：是从Web服务器传输超文本到本地浏览器的传送协议。

**UDP对应的协议：**

（1） **DNS**：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。（两种协议都使用了）

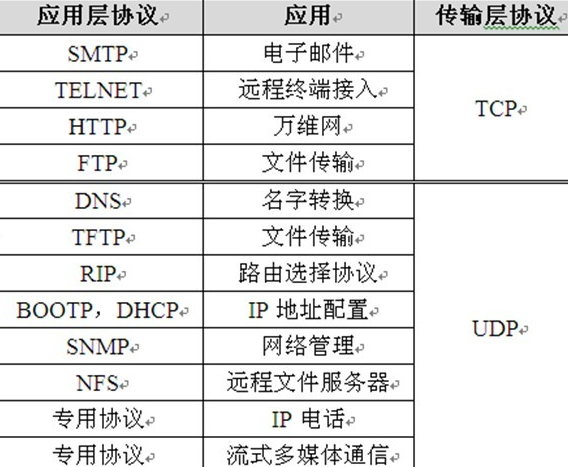
（2） **SNMP**：简单网络管理协议，使用**161**号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。

（3）**TFTP**(Trival File Tran敏感词er Protocal)，简单文件传输协议，该协议在熟知端口**69**上使用UDP服务。

TCP是面向字节流传输的，而UDP是面向报文传输。

面向报文的传输方式是应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。因此，应用程序必须选择合适大小的报文。若报文太长，则IP层需要分片，降低效率。如果太短，则IP会太小。UDP对应用层传下来的报文，不会合并也不会拆分，而是保留这些报文的边界。

面向字节流传输，虽然应用程序和TCP交互是一次一个数据块，但是TCP把应用程序堪称是一连串的无结构的字节流。TCP有一个缓冲，当应用程序传送的数据块太长，TCP就可以把它划分短一些再传送。如果应用程序一次只发送一个字节，TCP也会等待积累足够多的字节后再构成报文段发送出去。





TCP是端到端的传输，UDP是点到点的传输。

## TCP如何保证可靠

1 当TCP发出一个段之后，会启动一个定时器，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。（超时重传）

2 当TCP收到发自TCP连接另一端的数据时，它将发送一个确认。这个确认不是立即发送，通常推迟几分之一秒。（如果此时接收端要发送数据给发送端，可以把确认包放到要发送的数据包中）

3 TCP会保持它首部和数据的校验和。这是一个端到端的校验和，目的是检测数据在传输过程的任何变化。如果收到段的校验和有差错，TCP将丢弃这个报文段和不确认收到此报文段。（希望发端超时并重发）

4 由于IP数据报的到达可能失序，因此TCP报文段的到达也可能失序。如有必要，TCP将对收到的数据进行重新排序，将收到的数据以正确的顺序交给应用层。

TCP给发送的每个数据包进行编号，接收方对数据包进行排序，将有序数据传送给应用层。

5 IP数据报会发生重复，TCP接收端必须丢弃重复的数据

6 TCP提供流量控制。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区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。

可靠的机制保证：

1 差错检测：发送放发送额外的信息供接收方进行校验，接收方进行校验判断是否有错误发生。校验和

2 接收方反馈：接收方是否需要将有错误发生的信息反馈给发送方。ACK和NAK

3 发送方重传：在出现错误时，发送方需要重传出错的分组。

## TCP数据传输

当TCP连接建立之后，应用程序即可使用该连接进行数据收发。应用程序将数据提交给TCP，TCP将数据放入自己的缓存，并且在其认为合适的时候将数据发送出去。在TCP中，数据会被当做字节流并按照MSS的大小进行分段，然后加上TCP头部并提交给网络层。之后数据就会被网络层提交给目地主机，目地主机的IP层会将分组提交给TCP，TCP根据报文段的头部信息找到相应的socket，并将报文段提交给该socket，socket是和应用关联的，也就提交给了应用。

## TCP粘包和半包

TCP是数据流的传输，流，也就是说没有界限的。TCP并不了解上层业务数据的具体含义，它会根据TCP缓冲区的实际情况进行包的划分（MSS）。因此，在业务上任务，一个完整的包可能会被TCP拆分成多个包进行发送，也有可能把多个小包封装成一个大的数据包发送。这就是粘包和半包。

TCP粘包是指发送方发送的若干个数据包到达接收方时，粘成一个包。这样对于接收方来说，就无法区分这是一个包还是几个包。

出现粘包的原因：

可能由发送方引起。发送方引起的粘包是TCP协议本身造成的，TCP为了提高传输效率，发送方会收集足够多的数据包之后再一起发送，若连续发的几次数据包都比较小，TCP会根据优化算法Nagle将这些数据包合成一个包发送出去，这样接收方就收到了粘包数据。

接收方引起的粘包是因为接收方不及时接收数据，从而导致粘包。因为接收方将数据放在系统缓冲区中，用户进程从缓冲区中读取数据，若下一个数据包到达时前一个包还没被取出，那么下一包的数据放到系统缓冲区时就会接在前一个包数据之后，而用户进程根据设定的缓冲区从系统缓冲区中取出数据，这样就一次取得了多个包的数据。

（Nagel：在发送数据包时，如果发现数据包比较小，并不会立即发送，而是等待一段时间，看看还有没有数据要发送，如果有，则会将小的数据包封装成一个大的进行发送）

粘包、拆包原因：

1 应用程序写入的字节大小大于发送缓冲区的大小

2 进行MSS大小的分段

粘包解决：

1 发送的消息定长，如果不够，则补空格

2 在数据包的尾部添加换行符进行分割，例如FTP协议

3 将消息分为消息头和消息体，消息头中包含消息的长度（接收端获取到指定的长度时，才说明包完整）

什么时候考虑粘包？

1 在短连接中不考虑粘包。TCp每次发送数据，就与对方建立连接，然后双方发送完数据后，就会关闭连接，不会出现粘包问题（对于双方来说，知道只会接受一段数据，然后就关闭连接）

2 如果发送的数据无结构，比如文件传输，这样也不需要考虑粘包

3 双方在建立连接之后，需要发送不同结构的数据，需要考虑

## UDP不存在沾包 半包问题

UDP不会使用Nagle算法进行优化传书，而且，UDP支持的是一对多的模式，所以接收端的缓冲区采用了链接结构来记录每一个到达的数据包，在每个包中就有了消息头（消息来源 端口等），这样对于接收端来说，就比较容易处理。

保护消息边界，就是指传输协议把数据当作一条独立的消息放在网络中传输，接收端只能接收独立的消息，接收端一次只能接收发送端发出的一个数据包。而面向流传输是没有边界的，接收端可以在一次接收动作中接收多个数据包。也就是说，对于UDP，发送端发送几次，接收端就要接收几次。

## 拥塞控制

当网络拥塞时数据报不能及时被转，在分组转发网络中，数据报就会被排队，甚至出现丢包因此说网络拥塞会带来网络销：

* 引入大的排队时延
* 当数据报被丢失时发送方必须重传，因此引入了重传开销
* 当数据报被丢失时，丢失路由器的上游路由器做的工作都变成了无用功

因此必须采取技术来尽可能避免拥塞。由于IP层不提供网络是否拥塞的信息，因而TCP必须自己来判断网络是否出现了拥塞。  
TCP将丢包（可能是超时也可能是收到了三个冗余的ACK）看做是网络拥塞的线索，将RTT增加看做是网络拥塞程度加重的线索。  
TCP让连接双方根据自己所判断的网络拥塞的程度来限制其发往网络的流量。TCP在连接的每一端都增加了一个变量cong\_win，它表示拥塞窗口，用于限制一端可以向网络发送的数据。TCP连接的每一端都保证它所已经发送的未被确认的报文段的总大小不会超过拥塞窗口和对方通告的窗口大小中的较小的那一个。  
TCP通过ACK到达的情况（即是否到达，到达的速率）来调整拥塞窗口的大小。

### 1.慢启动

在建立TCP连接时，拥塞窗口被初始化为 min (4\*SMSS, max (2\*SMSS, 4380 bytes)) 。但是TCP不是以线性的方式增大拥塞窗口，而是以指数的方式增加的，即：

1. 初始设置cwnd=1个 min (4\*SMSS, max (2\*SMSS, 4380 bytes)) ，发送一个报文，在RFC5861中有更新，但是总体就是一个较小的值（SMSS：SENDER MAXIMUM SEGMENT SIZE : The SMSS is the size of the largest segment that the sender can transmit）。(对于SCTP，cwnd初始值为min(4\*MTU)。
2. 收到对该报文的ack，则cwnd被设置为2个MSS，可以发送两个报文
3. 收到对2个报文的确认后，cwnd更新为4个MSS，可以发送四个报文，一依次类推。

该过程会一直持续直到遇到一个丢包事件为止（或者等于ssthresh时，ssthresh事实上是用于区分是进行的拥塞避免还是慢启动，初始化时它被设置为65536），事实上该算法相当于每收到对一个MSS的确认就将cwnd增大一个MSS（cwnd决定了当前可以发送cwnd/MSS个报文，当这个cwnd/MSS个报文段都被确认后，就将cwdn增大了cwnd/MSS × MSS即cwnd）。发送方取拥塞窗口与对端通告窗口中的最小值作为发送上限。由于初始的拥塞窗口很小的值，因为该启动方式被称为慢启动。

但是对于SCTP，它只有在满足三个条件时才增大cwnd，min(上次发送的所有数据的大小，MTU)：

1. cwnd已经被用完
2. 累积确认被更新了
3. 当前不处于快速恢复模式

### 2.拥塞避免（加性增，乘性减）

当遇到一个丢包事件时，TCP会将其拥塞窗口降低为原来值的一半，同时将ssthresh设置为 max (FlightSize / 2, 2\*SMSS) （FlightSize：The amount of data that has been sent but not yet acknowledged.）（对于SCTP为：max(cwnd/2, 4\*MTU），总体而言，SCTP和TCP的拥塞控制、避免算法是一致的，用MTU替换掉MSS即可。另外SCTP只在有大于等于cwnd的数据正在被发送时（onflight）才更新cwnd) 。其目的是通过降低发送速率来缓解、避免拥塞。但是拥塞窗口大小至少为1个MMS。  
在非启动期间，当TCP探测到没有拥塞时，即当连接的一端收到了对它已经发送但未被确认的报文段的确认时，它就会增大拥塞窗口，增大的方式是每收到一个ACK将拥塞窗口大小增大MSS×MSS/cwnd，因此在该算法下，经过一个RTT，cwnd最多增大一个MSS。  
因此TCP的拥塞控制方式又称为加性增，乘性减。拥塞窗口的增加受惠的只是自己，而拥塞窗口减少受益的是大家，当出现拥塞时，通过乘性减虽然损害了自己，但是可以让更多的其它网络参与者受益，这也证实TCP拥塞控制中的公平性的核心所在。

### 3.对超时事件的处理

虽然超时和收到三个冗余ACK（SCTP中不存在三个冗余ACK，对应的事件是三个SACK都不包含都某个报文段的确认，则认为该报文段丢失需要重传）都被认为是丢包事件，但是TCP在二者的处理上并不全相同。当收到三个冗余ACK时，TCP的处理就是“加性增，乘性减”。但是如果是超时事件，则TCP会更新ssthresh的值为max (FlightSize / 2, 2\*SMSS)（对于SCTP，max(cwnd/2, 4\*MTU)），然后进入“慢启动”过程，即将拥塞窗口设置为一个MSS，然后指数增加拥塞窗口大小。此时“慢启动”会持续到遇到一个丢包事件或者拥塞窗口被增大到了ssthresh，如果是增大到了ssthresh则进入“拥塞避免”的模式，即开始加性增。  
因此对于丢包事件来说，只要发生了丢包，则ssthresh都会更新max (FlightSize / 2, 2\*SMSS)（对于SCTP，max(cwnd/2, 4\*MTU)）。如果是超时，cwnd被设置为一个MSS（对于SCTP，1个MTU ）；如果是冗余ACK，则cwnd被更新为更新后的ssthresh。随后cwnd的更新方式取决于它和ssthresh之间的大小关系，如果cwdn小于或等于ssthresh，则就是在执行慢启动，否则就是在执行拥塞避免。

对超时时间和三个冗余ACK处理方式上存在区别的原因在于，收到冗余ACK表明了网络时可以交付报文段的，是可用的，而超时则就是明确的丢包。而收到冗余ACK至少表明网络还是可用的，只是出现了丢包，事实上，在出现三个丢包的时候，采用的是快速恢复+快速重传+拥塞避免。

### 4. 快速恢复

快速恢复一般和快速重传一起实现，其算法为：

1. 当收到第3个重复的ACK时（对于SCTP，有三个SACK都不包含某个报文的确认时。具体规则：1.只有报文TSN小于当前SACK中新被确认的最大TSN的被丢失的报文的丢失计数才会增加，2.如果已经处于快速重传模式，并且当前的SACK会更新累积确认点，则所有丢失的报文的丢失计数都会增加（这一点将保证报文会被尽快快速重传，从而使得尽快退出快速重传模式）。），把ssthresh设置为max (FlightSize / 2, 2\*SMSS)（对于SCTP，max(cwnd/2, 4\*MTU)），把cwnd设置为ssthresh的值加3个SMSS（对于SCTP，不增加）。然后重传丢失的报文段。因为收到3个重复的ACK表明有三个报文已经离开网络到达了接收断，被且被接收端给接收了。（同时，对于SCTP，还会将当前已经发出的最大的报文序号（TSN）作为退出快速恢复的序列号）
2. 再收到重复的ACK时，cwnd增加一个MSS。
3. 当收到确认新数据包的ACK时，把cwnd设置为第一步中的ssthresh的值。此时就重新进入到了第一步丢包时本应进入的拥塞避免。（对于SCTP，如果收到的SACK的累积确认确认了步骤1中的退出快速恢复的序列号，则退出快速恢复）

快速恢复意在通过快速重传丢失的报文，使得接收端可以将累积确认的最后一个报文和乱序到达的报文之间的“间隙”填充起来，从而尽快进行新的确认。在这个过程中，每当接收断收到一个新的乱序的报文，发送端就将自己的cwnd增大一个MSS，使得发送端可以尽快填充接收端的“间隙”，直到累积确认的报文段之后有新的连续的报文段被接收端接收到了，这个时候接收端会更新一个新的ACK，发送端在收到该ACK后就退出快速恢复并进入拥塞避免。

需要注意的是无论是拥塞避免还是慢启动，SCTP规定，如果某个地址不用于发送数据，则每个RTO都要对cwnd做一次调整，调整后的值为max(cwnd/2, 4\*MTU).

## 流量控制

TCP提供了流量控制功能，来防止发送方发送过快而导致接收方缓存溢出的情形出现。这是通过让接收方通告一个接收窗口大小来实现的。接收窗口的大小包含在TCP头部的窗口大小字段中。其工作原理为：  
接收方通过窗口大小通告本地可以接收的报文段的总大小。发送方将根据该信息来判断自己可以发送多少数据。发送方保证自己发送的未被确认的报文段的总大小不超过接收方通告的窗口大小。对应到我们所描述的GBN和选择重传协议中，就是发送方会用接收方通告的窗口大小更新本地的窗口大小N的值。

但是TCP连接的一端可能通告一个大小为0的窗口，这时候接收到对端通告大小为0的窗口的一端并不会停止发送，而是会启动一个定时器来发送窗口探测报文段，该报文段只包含一个字节，该报文段会被接收方确认，该定时器会一直重启自身来发送窗口探测包直到对端通告了一个大小不为0的窗口为止。定时器的超时值会逐渐增大到一个最大值，然后固定以该值重发窗口探测包。

## 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，通信双方在进行通信之前，要先在双方建立起一个完整的可以彼此沟通的通道，在通信过程中，整个连接的情况一直可以被实时地监控和管理。

非面向连接的服务，不需要预先建立一个联络两个通信节点的连接，需要通信的时候，发送节点就可以往网络上发送信息，让信息自主地在网络上去传，一般在传输的过程中不再加以监控。

## 端口及对应的服务

|  |  |  |  |
| --- | --- | --- | --- |
| 服务 | 端口号 | 服务 | 端口号 |
| 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 | Shell或 cmd | 514 |
| POP-2 | 109 | SQL Server | 1433 |

## 交换机 路由器 网关

1）交换机

在计算机网络系统中，交换机是针对共享工作模式的弱点而推出的。交换机拥有一条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背 部总线上，当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，交换机才广播到所有的端口，接收端口回应后交换机会“学习”新的地址，并把它添加入内部地址表 中。

交换机工作于OSI参考模型的第二层，即数据链路层。交换机内部的CPU会在每个端口成功连接时，通过ARP协议学习它的MAC地址，保存成一张 ARP表。在今后的通讯中，发往该MAC地址的数据包将仅送往其对应的端口，而不是所有的端口。因此，交换机可用于划分数据链路层广播，即冲突域；但它不 能划分网络层广播，即广播域。

交换机被广泛应用于二层网络交换，俗称“二层交换机”。

交换机的种类有：二层交换机、三层交换机、四层交换机、七层交换机分别工作在OSI七层模型中的第二层、第三层、第四层盒第七层，并因此而得名。

2）路由器

**路由器**（**Router**）是一种计算机网络设备，提供了路由与转送两种重要机制，可以决定数据包从来源端到目的端所经过 的路由路径（host到host之间的传输路径），这个过程称为路由；将路由器输入端的数据包移送至适当的路由器输出端(在路由器内部进行)，这称为转 送。路由工作在OSI模型的第三层——即网络层，例如网际协议。

**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。** 路由器与交换器的差别，路由器是属于OSI第三层的产品，交换器是OSI第二层的产品(这里特指二层交换机)。

3）网关

**网关**（Gateway），**网关**顾名思义就是连接两个网络的设备，区别于路由器（由于历史的原因，许多有关TCP/IP 的文献曾经把网络层使用的路由器（Router）称为网关，在今天很多局域网采用都是路由来接入网络，因此现在通常指的网关就是路由器的IP），经常在家 庭中或者小型企业网络中使用，用于连接局域网和Internet。 网关也经常指把一种协议转成另一种协议的设备，比如语音网关。

在传统TCP/IP术语中，网络设备只分成两种，一种为网关（gateway），另一种为主机（host）。网关能在网络间转递数据包，但主机不能 转送数据包。在主机（又称终端系统，end system）中，数据包需经过TCP/IP四层协议处理，但是在网关（又称中介系 统，intermediate system）只需要到达网际层（Internet layer），决定路径之后就可以转送。在当时，网关 （gateway）与路由器（router）还没有区别。

在现代网络术语中，网关（gateway）与路由器（router）的定义不同。网关（gateway）能在不同协议间移动数据，而路由器（router）是在不同网络间移动数据，相当于传统所说的IP网关（IP gateway）。

**网关**是连接两个网络的设备，对于语音网关来说，他可以连接PSTN网络和以太网，这就相当于VOIP，把不同电话中的模拟信号通过网关而转换成数字信号，而且加入协议再去传输。在到了接收端的时候再通过**网关**还原成模拟的电话信号，最后才能在电话机上听到。

对于以太网中的**网关**只能转发三层以上数据包，这一点和路由是一样的。而不同的是**网关**中并没有路由表，他只能按照预先设定的不同网段来进行转发。网关最重要的一点就是端口映射，子网内用户在外网看来只是外网的IP地址对应着不同的端口，这样看来就会保护子网内的用户。

## 流量控制和拥塞控制

**流量控制**

所谓流量控制就是让发送方的发送速率不要太快，要让对方来得及接收。

利用滑动窗口机制可以很方便地在 TCP 连接上实现对发送方的流量控制。

发送方的发送窗口不能超过接收方给出的接收窗口的数值。

注意，TCP 的窗口单位是字节，不是报文段。

**拥塞控制**

在计算机网络中的链路容量（即带宽）、交换节点中的缓存和处理机等，都是网络的资源。在某段时间，若对网络中的某一资源的需求超过了该资源所能提供的可用部分，网络中的性能就要变坏。这种情况就叫做拥塞。

若网络中的许多资源同时出现供应不足，网络的性能就要明显变坏，整个网络德吞吐量将随输入负荷的增大而下降。

拥塞控制的四种算法：慢开始(slow start)、拥塞避免(congestion avoidance)、快重传(fast retransmit)、快恢复(fast recovery)。

**拥塞控制与流量控制**

所谓拥塞控制就是防止过多的数据注入到网络中，这样可以使网络中德路由器或链路不致过载。拥塞控制所要做的都有一个前提，就是网路能够承受现有的网络负荷。拥塞控制是一个全局性德过程，涉及到所有的主机、所有的路由器、以及与降低网路传输性能有关的所有因素。

相反，流量控制往往指点对点通信量德控制，是个端到端问题（接收端控制发送端）。流量控制所要做的就是抑制发送端发送数据的速率，以便使接收端来得及接收。

拥塞控制和流量控制之所以常常被搞混，是因为某些拥塞控制算法是向发送端发送控制报文，并告诉发送端，网络出现麻烦，必须放慢发送速率。这点又和流量控制很相似。

## 网络层协议简介

**ARP地址解析协议**

网络层使用的是IP地址，但在实际网络的链路上传送数据帧时，最终还是必须使用该网络德硬件地址。但IP地址和下面的网络的硬件地址之间由于格式不同而不存在简单的映射关系（例如，IP地址有32位，而局域网的硬件地址是48位）。此外，在一个网络中，经常会有新的主机加进来，或撤走一些主机。更换网络适配器也会使主机的硬件地址改变。地址解析协议ARP解决这个问题的方法是在主机ARP高速缓存中应存放一个从IP地址到硬件地址的映射表，并且这个映射表还经常动态更新（新增或超时删除）。

每一个主机都设有一个ARP告诉缓存，里面有本局域网上的各主机和路由器的IP地址到硬件地址的映射表，这些都是该主机目前知道的一些地址。

注意，ARP是解决同一个局域网上的主机或路由器的IP地址和硬件地址的映射问题。从IP地址到硬件地址的解析是自动进行的，主机的用户对这种地址解析过程是不知道的。

**ICMP网际控制报文协议**

ICMP报文的种类有两种，即ICMP差错报告报文和ICMP询问报文。

ICMP差错报告报文共有5种，即：

（1）终点不可达    当路由器或主机不能交付数据报时就向源点发送终点不可达报文。

（2）源点抑制    当路由器或主机由于拥塞而丢弃数据报时，就向源点发送源点抑制报文，使源点知道应当把数据报的发送速率放慢。

（3）时间超过    当路由器收到生存时间为零的数据报时，除丢弃该数据报外，还要向源点发送时间超过报文。当终点在预先规定时间内不能收到一个数据报的全部数据报片时，就把已收到的数据报片都丢弃，并向源点发送时间超过报文。

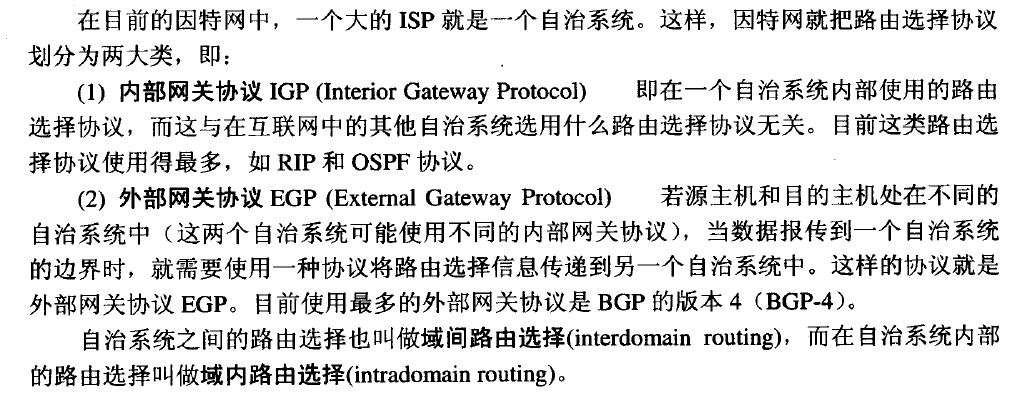
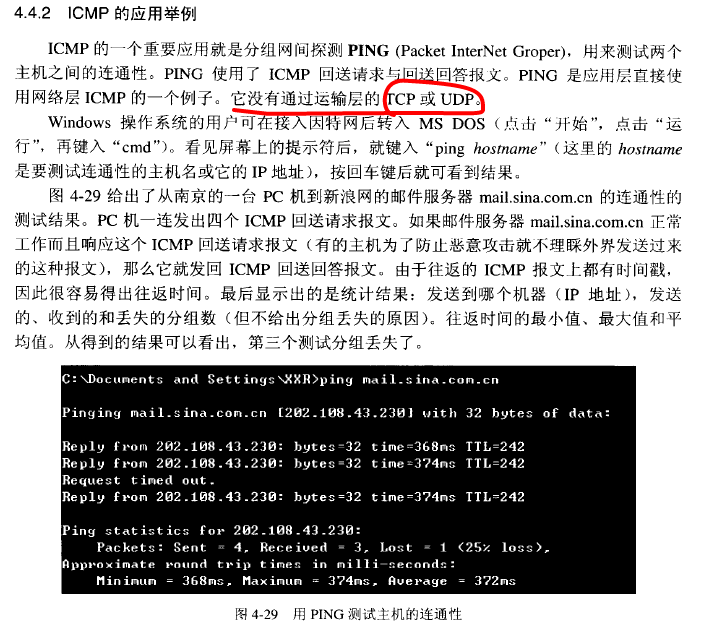
（4）参数问题    当路由器或目的主机收到的数据报的首部中有的字段值不正确时，就丢弃该数据报，并向源点发送参数问题报文。

（5）改变路由（重定向）    路由器把改变路由报文发送给主机，让主机知道下次应将数据报发送给另外的路由器（可通过更好的路由）。

ICMP询问报文有两种，即：

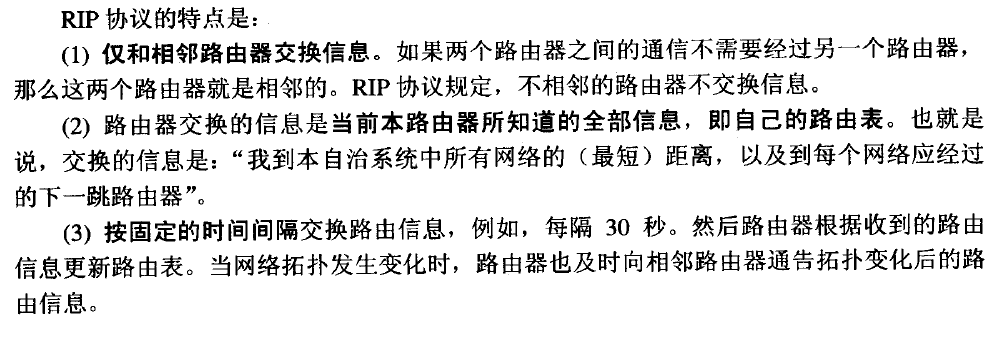
（1）回送请求和回答    ICMP回送请求报文是由主机和路由器向一个特定的目的主机发出的询问。收到此报文的主机必须给源主机或路由器发送ICMP回送回答报文。这种询问报文用来测试目的站是否可达以及了解其有关状态。

（2）时间戳请求和回答    ICMP时间戳请求报文是请某个主机或路由器回答当前的日期和时间。在ICMP时间戳回答报文中有一个32位的字段，其中写入的整数代表从1900年1月1日到当前时刻一共多少秒。时间戳请求与回答可用来进行时钟同步和测量时延。

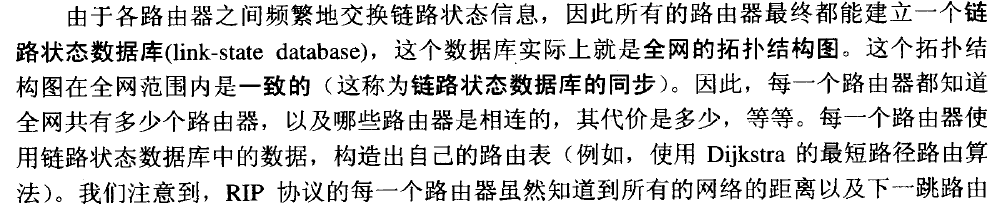
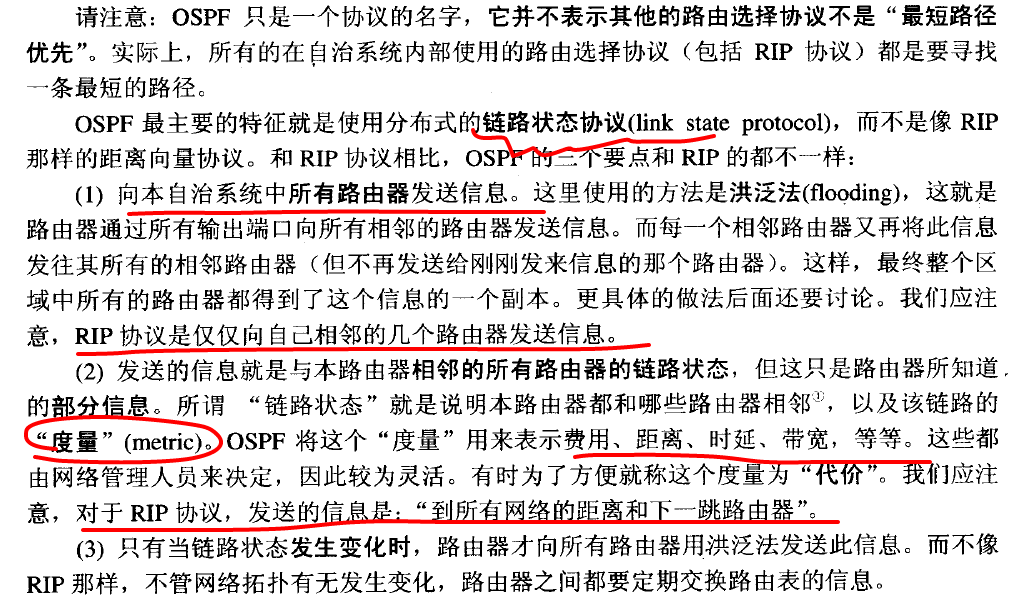


**RIP路由信息协议**

RIP是一种分布式的基于距离向量的路由选择协议。



**OSPF最短路径优先协议**



## DNS服务

DNS 域名解析系统。

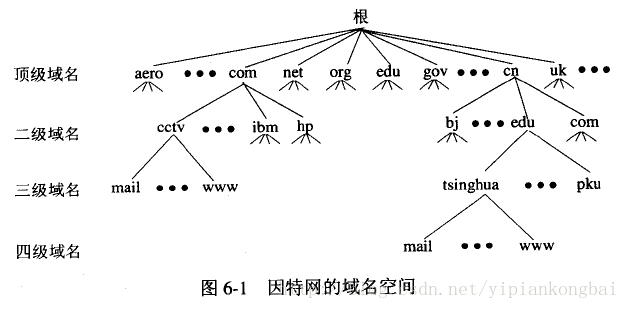
将域名解析为对应的IP地址。 基于UDP协议。 工作在应用层。

域名结构：



cn代表中国 us代表美国 uk代表英国

com代表企业 edu代表教育机构 gov代表政府组织 org代表非营利性组织 int代表国际组织



 如果采用上述的树状结构，每一个节点都采用一个域名服务器，这样会使得域名服务器的数量太多，使域名服务器系统的运行效率降低。所以在DNS中，采用划分区的方法来解决。

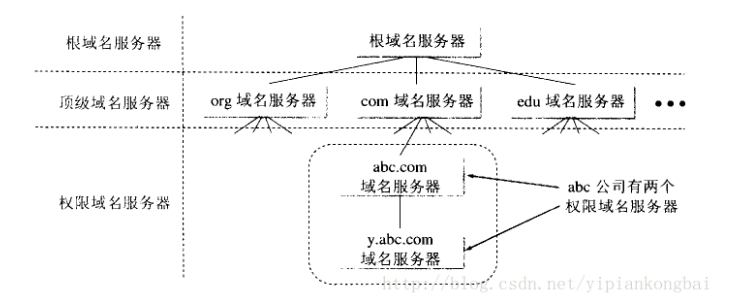
 一个服务器所负责管辖(或有权限)的范围叫做区(zone)。各单位根据具体情况来划分自己管辖范围的区。但在一个区中的所有节点必须是能够连通的。每一个区设置相应的权限域名服务器，用来保存该区中的所有主机到域名IP地址的映射。总之，DNS服务器的管辖范围不是以“域”为单位，而是以“区”为单位。区是DNS服务器实际管辖的范围。区 <= 域。

**根域名服务器**：最高层次的域名服务器，也是最重要的域名服务器。所有的根域名服务器都知道所有的顶级域名服务器的域名和IP地址。不管是哪一个本地域名服务器，若要对因特网上任何一个域名进行解析，只要自己无法解析，就首先求助根域名服务器。所以根域名服务器是最重要的域名服务器。假定所有的根域名服务器都瘫痪了，那么整个DNS系统就无法工作。需要注意的是，在很多情况下，根域名服务器并不直接把待查询的域名直接解析出IP地址，而是告诉本地域名服务器下一步应当找哪一个顶级域名服务器进行查询。

**顶级域名服务器**：负责管理在该顶级域名服务器注册的二级域名。

**权限域名服务器**：负责一个“区”的域名服务器。

**本地域名服务器**：本地服务器不属于下图的域名服务器的层次结构，但是它对域名系统非常重要。当一个主机发出DNS查询请求时，这个查询请求报文就发送给本地域名服务器。



**域名解析过程**

**注意：1** 主机向本地域名服务器的查询一般都是采用递归查询。所谓递归查询就是：如果主机所询问的本地域名服务器不知道被查询的域名的IP地址，那么本地域名服务器就以DNS客户的身份，向其它根域名服务器继续发出查询请求报文(即替主机继续查询)，而不是让主机自己进行下一步查询。因此，递归查询返回的查询结果或者是所要查询的IP地址，或者是报错，表示无法查询到所需的IP地址。

2 本地域名服务器向根域名服务器的查询的迭代查询。迭代查询的特点：当根域名服务器收到本地域名服务器发出的迭代查询请求报文时，要么给出所要查询的IP地址，要么告诉本地服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地服务器进行后续的查询。根域名服务器通常是把自己知道的顶级域名服务器的IP地址告诉本地域名服务器，让本地域名服务器再向顶级域名服务器查询。顶级域名服务器在收到本地域名服务器的查询请求后，要么给出所要查询的IP地址，要么告诉本地服务器下一步应当向哪一个权限域名服务器进行查询。最后，知道了所要解析的IP地址或报错，然后把这个结果返回给发起查询的主机。

假定域名为m.xyz.com的主机想知道另一个主机y.abc.com的IP地址。例如，主机m.xyz.com打算发送邮件给y.abc.com。这时就必须知道主机y.abc.com的IP地址。下面是上图a的几个查询步骤：

        1、主机m.abc.com先向本地服务器dns.xyz.com进行递归查询。

        2、本地服务器采用迭代查询。它先向一个根域名服务器查询。

        3、根域名服务器告诉本地服务器，下一次应查询的顶级域名服务器dns.com的IP地址。

        4、本地域名服务器向顶级域名服务器dns.com进行查询。

        5、顶级域名服务器dns.com告诉本地域名服务器，下一步应查询的权限服务器dns.abc.com的IP地址。

        6、本地域名服务器向权限域名服务器dns.abc.com进行查询。

        7、权限域名服务器dns.abc.com告诉本地域名服务器，所查询的主机的IP地址。

        8、本地域名服务器最后把查询结果告诉m.xyz.com。

        整个查询过程共用到了8个UDP报文。

        为了提高DNS查询效率，并减轻服务器的负荷和减少因特网上的DNS查询报文数量，在域名服务器中广泛使用了高速缓存，用来存放最近查询过的域名以及从何处获得域名映射信息的记录。

        例如，在上面的查询过程中，如果在m.xyz.com的主机上不久前已经有用户查询过y.abc.com的IP地址，那么本地域名服务器就不必向根域名服务器重新查询y.abc.com的IP地址，而是直接把告诉缓存中存放的上次查询结果(即y.abc.com的IP地址)告诉用户。

        由于名字到地址的绑定并不经常改变，为保持告诉缓存中的内容正确，域名服务器应为每项内容设置计时器并处理超过合理时间的项(例如每个项目两天)，即TTL，time to live。当域名服务器已从缓存中删去某项信息后又被请求查询该项信息，就必须重新到授权管理该项的域名服务器绑定信息。当权限服务器回答一个查询请求时，在响应中都指明绑定有效存在的时间值。增加此时间值可减少网络开销，而减少此时间值可提高域名解析的正确性。

        不仅在本地域名服务器中需要高速缓存，在主机中也需要。许多主机在启动时从本地服务器下载名字和地址的全部数据库，维护存放自己最近使用的域名的高速缓存，并且只在从缓存中找不到名字时才使用域名服务器。维护本地域名服务器数据库的主机应当定期地检查域名服务器以获取新的映射信息，而且主机必须从缓存中删除无效的项。由于域名改动并不频繁，大多数网点不需花精力就能维护数据库的一致性。

## NAT

网络地址转换。将私有网络地址转换为公有网络地址的技术。

解决的问题： 1 IP地址短缺的问题 2 保证了内网的安全性

类型 1 静态NAT 2 动态NAT 3 网络地址端口转换NAPT

NAPT是将内部网络地址映射到外部网络的一个IP地址的不同端口上。

分为SNAT和DNAT。

(1)源NAT（Source NAT，SNAT）：修改数据包的源地址。源NAT改变第一个数据包的来源地址，它永远会在数据包发送到网络之前完成，数据包伪装就是一具SNAT的例子。实现了具有私有地址的用户对公网的访问。

(2)目的NAT（Destination NAT，DNAT）：修改数据包的目的地址。Destination NAT刚好与SNAT相反，它是改变第一个数据懈的目的地地址，如平衡负载、端口转发和透明代理就是属于DNAT。实现了公网上的用户对位于内网的具有私有地址的服务器的访问。

**应用**

数据伪装: 可以将内网数据包中的地址信息更改成统一的对外地址信息，不让内网主机直接暴露在因特网上，保证内网主机的安全。同时，该功能也常用来实现共享上网。

端口转发: 当内网主机对外提供服务时，由于使用的是内部私有IP地址，外网无法直接访问。因此，需要在网关上进行端口转发，将特定服务的数据包转发给内网主机。

负载平衡: 目的地址转换NAT可以重定向一些服务器的连接到其他随机选定的服务器。

失效终结: 目的地址转换NAT可以用来提供高可靠性的服务。如果一个系统有一台通过路由器访问的关键服务器，一旦路由器检测到该服务器当机，它可以使用目的地址转换NAT透明的把连接转移到一个备份服务器上。

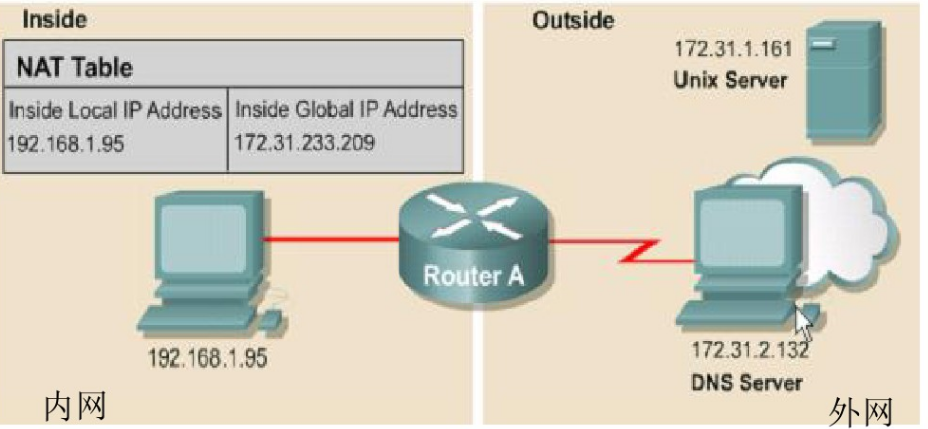
透明代理: NAT可以把连接到因特网的HTTP连接重定向到一个指定的HTTP代理服务器以缓存数据和过滤请求。一些因特网服务提供商就使用这种技术来减少带宽的使用而不用让他们的客户配置他们的浏览器支持代理连接。

**原理**

当私有网主机和公共网主机通信的IP包经过NAT网关时，将IP包中的源IP或目的IP在私有IP和NAT的公共IP之间进行转换。

**静态NAT**：私有地址和公有地址一一对应。

在NAT网关处，有一个NAT地址池。包含内部局部地址和内部全局地址。



**动态NAT**：在外部网络中定义了一系列的公有IP，采用动态分配的方法映射到内部网络。

动态NAT只是转换IP地址，为每一个内部的IP地址分配一个临时的外部IP地址，主要应用于拨号。当远程用户连接上之后，动态NAT就会分配给他一个IP地址，用户断开时，这个IP地址就会被释放而留待以后使用。

NAT池中存放的是一系列合法的公有IP。

**NAPT：**将内部IP地址映射到外部网络中的一个单独的IP地址上，同时在该地址上加上一个由NAT设备选定的TCP端口号。路由器支持的PAT会话数是有限制的。

**复用NAT池**：将多个合法的IP地址构成一个NAT池，使用复用技术映射其中的IP，每个地址可以对应多台主机，各主机用端口进行区分。是NAT池和PAT技术的结合，可用于大规模局域网。

在端口复用中，用端口区分的不是一台主机，而是一个网络连接（会话），当一台主机同时建立了多个会话时，每个会话会占用一个端口映射。