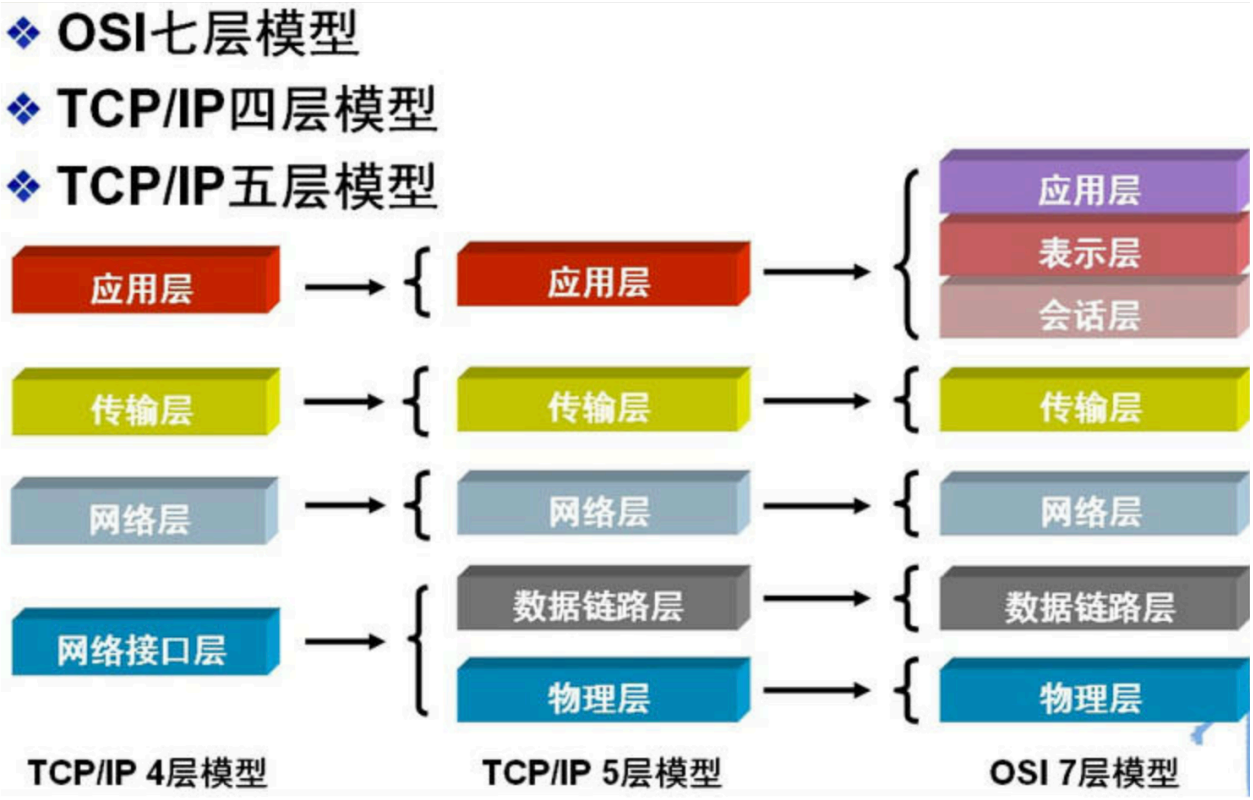
计算机网络知识点

各个层次和其中的协议

OSI七层模型和TCP/IP四层协议。它们之间的对应关系如下：



OSI七层模型	TCP/IP概念层模型	功能	TCP/IP协议族
应用层	应用层	文件传输，电子邮件，文件服务，虚拟终端	TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
表示层		数据格式化，代码转换，数据加密	没有协议
会话层		解除或建立与别的接点的联系	没有协议
传输层	传输层	提供端对端的接口	TCP，UDP
网络层	网络层	为数据包选择路由	IP，ICMP，RIP，OSPF，BGP，IGMP
数据链路层	链路层	传输有地址的帧以及错误检测功能	SLIP，CSLIP，PPP，ARP，RARP，MTU
物理层		以二进制数据形式在物理媒体上传输数据	ISO2110，IEEE802，IEEE802.2

硬件：

中继器、集线器——物理层

交换机、网桥——数据链路层

路由器——网络层

1、物理层：比特

主要定义物理设备标准，如网线的接口类型、光纤的接口类型、各种传输介质的传输速率等。它的主要作用是传输比特流（就是由1、0转化为电流强弱来进行传输,到达目的地后在转化为1、0，也就是我们常说的数模转换与模数转换）。这一层的数据叫做比特。

2、数据链路层：帧

定义了如何让格式化数据以进行传输，以及如何让控制对物理介质的访问。这一层通常还提供错误检测和纠正，以确保数据的可靠传输。

3、网络层：数据报

在位于不同地理位置的网络中的两个主机系统之间提供连接和路径选择。Internet的发展使得从世界各站点访问信息的用户数大大增加，而网络层正是管理这种连接的层。

4、运输层：报文段/用户数据报

定义了一些传输数据的协议和端口号（WWW端口80等），如： TCP（transmission control protocol –传输控制协议，传输效率低，可靠性强，用于传输可靠性要求高，数据量大的数据） UDP（user datagram protocol–用户数据报协议，与TCP特性恰恰相反，用于传输可靠性要求不高，数据量小的数据，如QQ聊天数据就是通过这种方式传输的）。 主要是将从下层接收的数据进行分段和传输，到达目的地址后再进行重组。常常把这一层数据叫做段。

5、会话层：

通过运输层（端口号：传输端口与接收端口）建立数据传输的通路。主要在你的系统之间发起会话或者接受会话请求（设备之间需要互相认识可以是IP也可以是MAC或者是主机名）

6、表示层：

可确保一个系统的应用层所发送的信息可以被另一个系统的应用层读取。例如，PC程序与另一台计算机进行通信，其中一台计算机使用扩展二一十进制交换码（EBCDIC），而另一台则使用美国信息交换标准码（ASCII）来表示相同的字符。如有必要，表示层会通过使用一种通格式来实现多种数据格式之间的转换。

7、应用层：报文

UDP、TCP

1、TCP、UDP的区别，举例运用的地方

- 区别
    1. 面向连接VS无连接 TCP建立一个连接需要3次握手IP数据包，断开连接需要4次握手。另外断开连接时发起方可能进入TIME_WAIT状态长达数分钟（视系统设置，windows一般为120秒），在此状态下连接（端口）无法被释放。 UDP不需要建立连接，可以直接发起。
    2. 可靠VS不可靠 TCP利用握手、ACK和重传机制，udp没有。 1，校验和（校验数据是否损坏）； 2，定时器（分组丢失则重传）； 3，序列号（用于检测丢失的分组和重复的分组）； 4，确认应答ACK（接收方告知发送方正确接收分组以及期望的下一个分组）； 5，否定确认（接收方通知发送方未被正确接收的分组）； 6，窗口和流水线（用于增加信道的吞吐量）。（窗口大小：无需等待确认应答而可以继续发送数据的最大值）

3. 有序性 TCP利用seq序列号对包进行排序，udp没有。
4. 面向字节流vs面向报文
    - 面向报文 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。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。这也就是说，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。（一个upd的最大报文长度2^16-1-20-8,20是ip报文头，8是udp报文头）
    - 面向字节流 面向字节流的话，虽然应用程序和TCP的交互是一次一个数据块（大小不等），但TCP把应用程序看成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也可以等待积累有足够多的字节后再构成报文段发送出去。
5. tcp有流量控制，udp没有
6. tcp的头部比20bytes，udp8byres

- 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。
- 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）。

使用TCP协议的常见端口主要有以下几种：

（1） FTP：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。

（2） Telnet：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。

（3） SMTP：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。

（4） POP3：Post Office Protocol 3的简称,即邮局协议的第3个版本，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）。

（5）HTTP：这是大家用得最多的协议，它就是常说的"超文本传输协议"。上网浏览网页时，就得在提供网页资源的计算机上打开80号端口以提供服务。常说"WWW服-务"、"Web服务器"用的就是这个端口。

使用UDP协议端口常见的有：

（1）RIP：路由选择信息协议（RIP）是一种在网关与主机之间交换路由选择信息的标准

（2） DNS：用于域名解析服务，这种服务在Windows NT系统中用得最多的。因特网上的每一台计算机都有一个网络地址与之对应，这个地址是常说的IP地址，它以纯数字+"."的形式表示。然而这却不便记忆，于是出-现了域名，访问计算机的时候只需要知道域名，域名和IP地址之间的变换由DNS服务器来完成。DNS用的是53号端口。

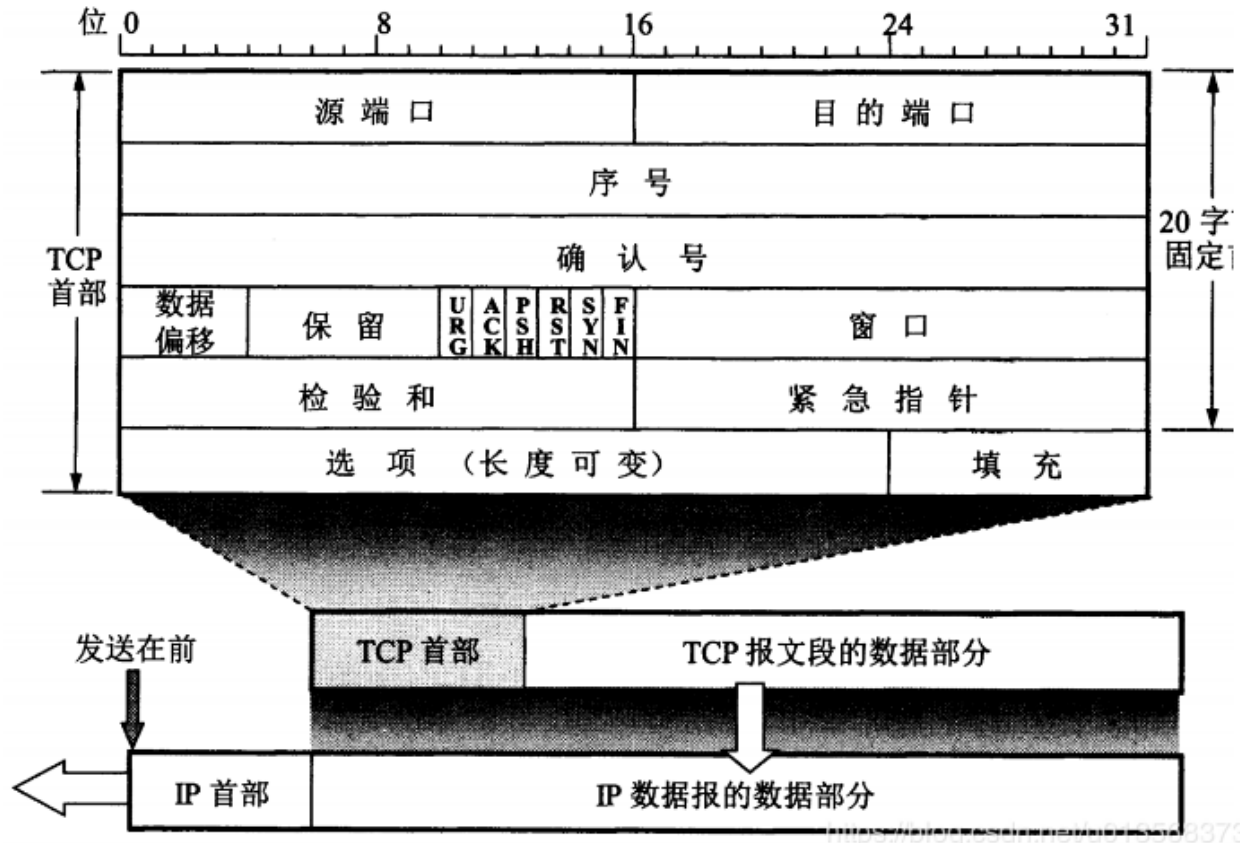
（3） SNMP：简单网络管理协议，使用161号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。

（4） OICQ：OICQ程序既接受服务，又提供服务，这样两个聊天的人才是平等的。OICQ用的是无连接的协议，也是说它用的是UDP协议。OICQ服务器是使用8-000号端口，侦听是否有信息到来，客户端使用4000号端口，向外发送信息。如果上述两个端口正在使用（有很多人同时和几个好友聊天），就顺序往上加。

2、TCP/UDP首部包含哪些信息?

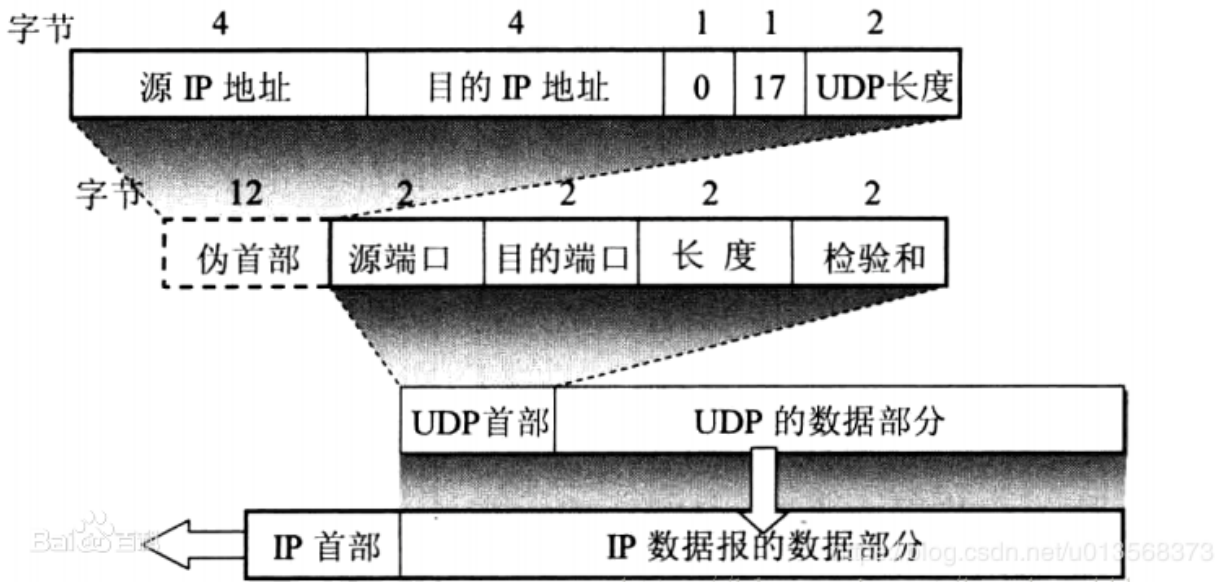
TCP头部信息

20个字节固定头部结构 40个字节头部选项字段



UDP头部信息

首部字段只有 8 个字节，包括源端口、目的端口、长度、检验和。12 字节的伪首部是为了计算检验和临时添加的



3、TCP保证传输的可靠性和数据的顺序性的原因

确保传输的可靠性方式：

TCP通过序列号、检验和、确认应答信号、重发控制、连接管理、窗口控制、流量控制、拥塞控制实现可靠性。

TCP提供一种面向连接的、可靠的字节流服务。 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：流量控制是指防止发送方发的太快，耗尽接收方的资源，从而使接收方来不及处理。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。

4、TCP滑动窗口和拥塞控制的方法

TCP使用滑动窗口实现流量控制。

因为TCP是顺序发送的，操作系统将这些数据包一批一批的发送给对方，就像一个窗口，不停地往后移动，所以，我们称之为TCP滑动窗口协议。在TCP中，窗口的大小是在TCP三次握手后协定的，并且窗口的大小并不是固定的，而是会随着网络的情况进行调整。

一旦前面的数据已经得到服务端确认了，这个窗口就会慢慢地往后滑，如下图所示，P1,P2两个数据包被确认之后，窗口就往后移动，后面新的数据包就由不可发送待发送变成了可发送状态了。

TCP的滑动窗口协议有什么意义呢？首先当然是可靠性，滑动窗口只有在队列前部的被确认之后，才会往后移动，保证数据包被接收方确认并接收。其次是传输效率，假如没有窗口，服务端是杂乱无章地进行发包，因为TCP的队首效应，如果有前面的包没有发送成功，就会不停的重试，反而造成更差的传输效率。最后是稳定性，TCP的滑动窗口大小，是整个复杂网络商榷的结果，会进行动态调整，可以尽量地避免网络拥塞，更加稳定。

TCP拥塞控制

拥塞控制是为了防止过多的数据注入到网络中，这样可以使网络中的路由器或者链路不至于过载。

TCP进行拥塞控制的的过程涉及四种算法：慢开始、拥塞避免、快速重传、快速恢复。

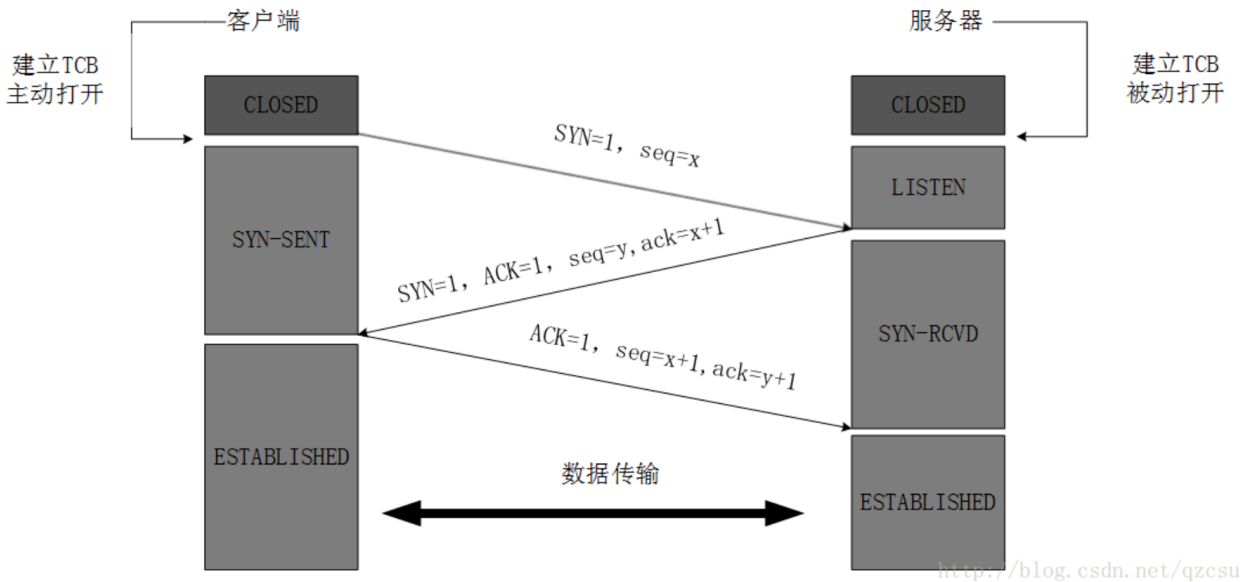
慢开始算法思想：由小到大逐渐增大发送窗口，也就是由小到大逐渐增大拥塞窗口数值，试探一下网络的拥塞情况。拥塞窗口的增长每次都是收到确认报文段数量的2倍。为了防止拥塞窗口增长多大引起网络阻塞，为其设置了一个慢启动门限，当到达门限时，就进入到拥塞避免阶段

拥塞避免算法的思路：不再以指数形式增长拥塞窗口，而是每经过一个往返时间RTT就将发送方的拥塞窗口+1，使其增长缓慢，按照线性方式增长，如果发生网络拥塞，比如丢包时，就将慢启动门限设为原来的一半，然后将拥塞窗口设置为1，开始执行慢启动算法。

快重传表示：接收方接收到一个失序的报文段后就立即发出重复确认。 当发送方一连接收到三个重复确认后就立即重传。而不是等到重传计时器到期。

快恢复的思想：将慢启动门限值设置为原来的一半，然后将拥塞窗口设置为现在的慢启动的门限值，不再执行慢启动而是直接进入拥塞避免阶段。使发送窗口成线性方式增长。

5、TCP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=x，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=x+1，随机产生一个值seq=y，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x+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=y+1，并将该数据包发送给Server，Server检查ack是否为y+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。

如果已经建立了连接，但是客户端突然出现故障了怎么办?

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

6、为什么tcp是三次而不是两次

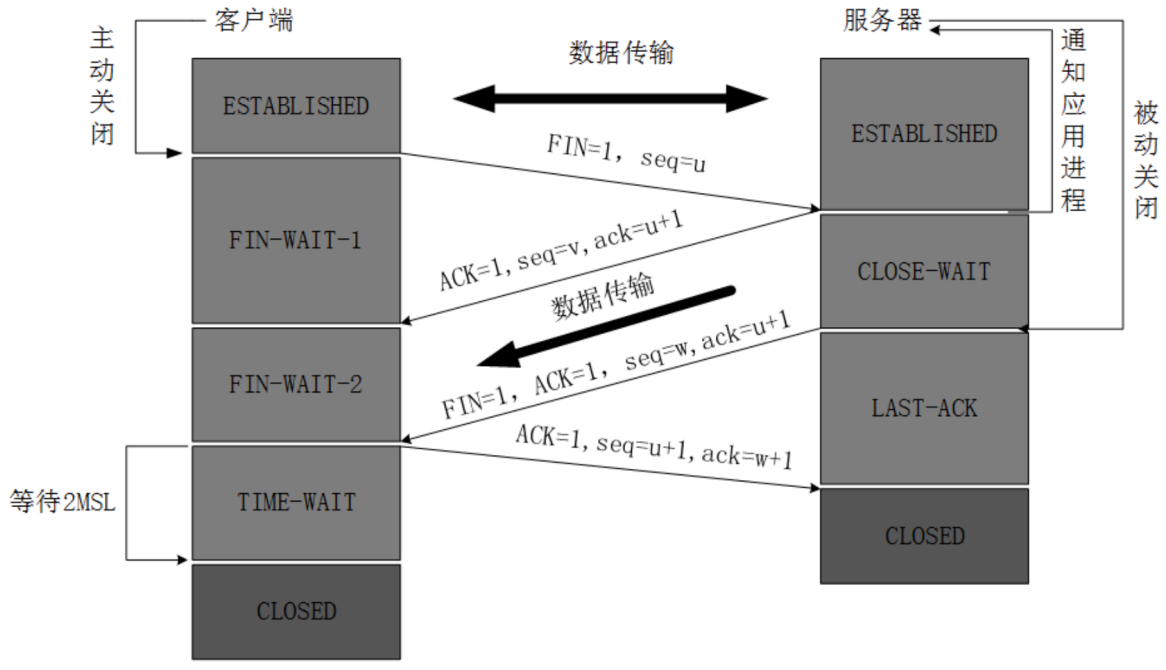
假定不采用三次握手，那么只要服务端发出确认，连接就建立了。

1、防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。

2、由于现在客户端并没有发出连接建立的请求，因此不会理会服务端的确认，也不会向服务端发送数据，但是服务端却以为新的传输连接已经建立了，并一直等待客户端发来数据，这样服务端的许多资源就这样白白浪费了。

采用三次握手的办法可以防止上述现象的发生。

7、TCP四次挥手



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。

为什么需要“四次挥手” 原因是因为tcp是**全双工模式，接收到FIN时意味将没有数据再发来，但是还是可以继续发送数据。**

为什么客户端最后还要等待2MSL?

1、保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，接着给出回应报文，并且会重启2MSL计时器。

2、防止已经失效的连接请求报文段出现在本连接中。

8、为什么建立连接是三次握手，关闭连接确是四次挥手呢?

建立连接的时候， 服务器在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，服务器收到对方的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，而自己也未必全部数据都发送给对方了，所以己方可以立即关闭，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，从而导致多了一次。

9、TCP粘包现象原因和解决方法

什么是粘包现象?

TCP粘包是指发送方发送的若干数据到接收方时粘成一包，从接收缓冲区看，后一包数据的头紧接着前一包数据的尾。

为什么会发生 TCP 粘包、拆包

1. 要发送的数据大于TCP发送缓冲区剩余空间大小，将会发生拆包。
2. 待发送数据大于MSS（最大报文长度），TCP在传输前将进行拆包。
3. 要发送的数据小于TCP发送缓冲区的大小，TCP将多次写入缓冲区的数据一次发送出去，将会发生粘包。
4. 接收数据端的应用层没有及时读取接收缓冲区中的数据，将发生粘包。

为什么会出现粘包现象?

发送方原因：TCP默认会使用Nagle算法。而Nagle算法主要做两件事：1、只有上一个分组得到确认，才会发送下一个分组；2、收集多个小分组，在一个确认到来时一起发送。所以，正是Nagle算法造成了发送方有可能造成粘包现象。

接收方原因：TCP接收到分组时，并不会立刻送至应用层处理，或者说，应用层并不一定会立即处理；实际上，TCP将收到的分组保存至接收缓存里，然后应用程序主动从缓存里读收到的分组。这样一来，如果TCP接收分组的速度大于应用程序读分组的速度，多个包就会被存至缓存，应用程序读时，就会读到多个首尾相接粘到一起的包。

如何处理粘包、拆包

通常会有以下一些常用的方法：

1. 使用带消息头的协议、消息头存储消息开始标识及消息长度信息，服务端获取消息头的时候解析出消息长度，然后向后读取该长度的内容。
2. 设置定长消息，服务端每次读取既定长度的内容作为一条完整消息，当消息不够长时，空位补上固定字符。
3. 设置消息边界，服务端从网络流中按消息编辑分离出消息内容，一般使用'\n '。
4. 更为复杂的协议，例如车联网协议 808,809 协议。

10、CLOSE-WAIT和TIME-WAIT状态

1. LISTENING状态 服务启动后首先处于侦听(LISTENING)状态。
2. ESTABLISHED状态 ESTABLISHED的意思是建立连接。表示两台机器正在通信。
3. CLOSE_WAIT 对方主动关闭连接或者网络异常导致连接中断,这时我方的状态会变成CLOSE_WAIT 此时我方要调用close()来使得连接正确关闭
4. TIME_WAIT 我方主动调用close()断开连接,收到对方确认后状态变为TIME_WAIT,缺省为240秒。TCP协议规定TIME_WAIT状态会一直持续2MSL(即两倍的分段最大生存期),以此来确保旧的连接状态不会对新连接产生影响。处于TIME_WAIT状态的连接占用的资源不会被内核释放,所以作为服务器,在可能的情况下,尽量不要主动断开连接,以减少TIME_WAIT状态造成的资源浪费。

ESTABLISHED 表示正在通信，TIME_WAIT 表示主动关闭，CLOSE_WAIT 表示被动关闭。

在通信双方建立连接之后，**主动关闭**连接的一方会进入`TIME_WAIT`状态。Client端主动关闭连接时，会发送最后一个ACK，然后进入`TIME_WAIT`状态，再停留2个MSL时间，进入`CLOSED`状态。

11、如何查看TIME-WAIT状态的链接数量？为什么会TIME-WAIT过多？解决方法是怎样的?

TIME_WAIT状态存在的理由：

1. 可靠地实现TCP全双工连接的终止
2. 允许老的重复分节在网络中消逝

如何查看TIME-WAIT状态的链接数量?

netstat -an |grep TIME_WAIT|wc -l 查看连接数等待time_wait状态连接数

#netstat -n | awk '/^tcp/ {++S[$NF]} END { for(a in S) print(a,S[a])}'

为什么会TIME-WAIT过多?

在**高并发短连接**的TCP服务器上，当服务器处理完请求后立刻主动正常关闭连接。这就造成这类服务器上容易出现大量的TIME_WAIT状态的连接，而且并发量越大处于此种状态的连接越多。

另外，对于被动关闭连接的服务在主动关闭客户端非法请求或清理长时间不活动的连接时（这种情况很可能是客户端程序忘记关闭连接）也会出现TIME_WAIT的状态。

TIME_WAIT 对于web服务器来说，占用了一个socket 60秒，socket的数量是有限的，最多65535。

解决 TIME_WAIT 问题思路

打开系统的TIMEWAIT重用和快速回收。

如果配置调优后性能还不理想，可继续修改一下内核文件/etc/sysctl.conf的参数配置，

①解决发起端的IP地址，添加更多的IP(time_wait多的服务器) ②改用长链接方式 ③修改程序代码

HTTP超文本传输协议

1、HTTP1.0/1.1/2.0的区别

HTTP1.0 VS HTTP1.1

- 长连接：

  HTTP 1.0需要使用keep-alive参数来告知服务器端要建立一个长连接，而HTTP1.1默认支持长连接。

  HTTP是基于TCP/IP协议的，创建一个TCP连接是需要经过三次握手的,有一定的开销，如果每次通讯都要重新建立连接的话，对性能有影响。因此最好能维持一个长连接，可以用个长连接来发多个请求。

- 缓存：

  在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略例如Entity tag，If-Unmodified-Since, If-Match, If-None-Match等更多可供选择的缓存头来控制缓存策略

- 状态码：

  在HTTP1.1中新增了24个错误状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示服务器上的某个资源被永久性的删除

- 带宽优化：

  HTTP 1.1支持只发送header信息(不带任何body信息)，如果服务器认为客户端有权限请求服务器，则返回100，否则返回401。客户端如果接收到100，才开始把请求body发送到服务器。 这样当服务器返回401的时候，客户端就可以不用发送请求body了，节约了带宽。

- HOST:

  Host头处理，在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）。

  HTTP1.0是没有host域的，HTTP1.1才支持这个参数。

HTTP1.1 VS HTTP2.0

- 多路复用：

  在HTTP/1.1协议中，浏览器客户端在同一时间针对同一域名的请求有一定数据限制。超过限制数目的请求会被阻塞。 HTTP2.0使用了多路复用的技术，做到同一个连接并发处理多个请求，而且并发请求的数量比HTTP1.1大了好几个数量级。 当然HTTP1.1也可以多建立几个TCP连接，来支持处理更多并发的请求，但是创建TCP连接本身也是有开销的。 TCP连接有一个预热和保护的过程，先检查数据是否传送成功，一旦成功过，则慢慢加大传输速度。因此对应瞬时并发的连接，服务器的响应就会变慢。所以最好能使用一个建立好的连接，并且这个连接可以支持瞬时并发的请求。

- 首部压缩：

HTTP1.1不支持header数据的压缩，HTTP2.0使用HPACK算法对header的数据进行压缩，这样数据体积小了，在网络上传输就会更快。

- 服务器推送：

  当我们对支持HTTP2.0的web server请求数据的时候，服务器会顺便把一些客户端需要的资源一起推送到客户端，免得客户端再次创建连接发送请求到服务器端获取。这种方式非常合适加载静态资源。

2、HTTP常见状态码

HTTP协议的状态码由3位数字组成,第一个数字定义了响应的类别,共有5中类别: 1.1xx: 指示信息--表示请求已接收,继续处理 2.2xx: 成功--表示请求已被成功接收、理解、接受 3.3xx: 重定向--要完成请求必须进行更进一步的操作 4.4xx: 客户端错误--请求有语法错误或请求无法实现 5.5xx: 服务器端错误--服务器在处理请求的过程中发生了错误

常见状态码

200: 请求成功 301: 永久重定向。请求的资源已被永久的移动到新URI,返回信息会包括新的URI,浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替 302: 临时重定向。与301类似。但资源只是临时被移动。客户端应继续使用原有URI 403: 服务器理解请求客户端的请求,但是拒绝执行此请求.可能是没有权限访问或是人为设置不允许访问 404: 需要访问的文件不存在 499：客户端主动端开 500: 服务端代码异常,代码语法错误,连接不上数据库等 502: 代理情况下会出现,一般是后端服务器出现问题,如PHP-FPM挂掉 504: 后端服务器响应超时.如PHP-FPM的执行时间大于超时时间就会出现504

4、HTTP协议为什么是无状态的？如何让HTTP“有状态”?

HTTP是一种无状态协议，即服务器不保留与客户交易时的任何状态。也就是说，上一次的请求对这次的请求没有任何影响，服务端也不会对客户端上一次的请求进行任何记录处理。

带来的问题：用户登录后，切换到其他界面，进行操作，服务器端是无法判断是哪个用户登录的。 每次进行页面跳转的时候，得重新登录。

两种用于保持HTTP状态的技术就应运而生了，一个是 Cookie，而另一个则是 Session。cookie机制采用的是在客户端保持状态的方案，而session机制采用的是在服务器端保持状态的方案。

5、Cookie和Session的区别

1.存储位置不同

cookie的数据信息存放在本地。 session的数据信息存放在服务器上。

2.存储容量大小不同

cookie存储的容量较小，一般<=4KB。 session存储容量大小没有限制(但是为了服务器性能考虑，一般不能存放太多数据)。

3.存储有效期不同

cookie可以长期存储，只要不超过设置的过期时间，可以一直存储。 session在超过一定的操作时间(通常为30分钟)后会失效，但是当关闭浏览器时，为了保护用户信息，会自动调用session.invalidate()方法，该方法会清除掉session中的信息。

4.安全性不同

cookie存储在客户端，所以可以分析存放在本地的cookie并进行cookie欺骗，安全性较低。 session存储在服务器上，不存在敏感信息泄漏的风险，安全性较高。

5.域支持范围不同

cookie支持跨域名访问。例如，所有a.com的cookie在a.com下都能用。 session不支持跨域名访问。例如，www.a.com的session在api.a.com下不能用。

6.对服务器压力不同

cookie保存在客户端，不占用服务器资源。 session是保存在服务器端，每个用户都会产生一个session，session过多的时候会消耗服务器资源，所以大型网站会有专门的session服务器。

7.存储的数据类型不同

cookie中只能保管ASCII字符串，并需要通过编码方式存储为Unicode字符或者二进制数据。 session中能够存储任何类型的数据，包括且不限于string，integer，list，map等。

6、POST、GET区别，还有其他方式吗?

1. GET使用URL或Cookie传参，而POST将数据放在BODY中，这个是因为HTTP协议用法的约定。并非它们的本身区别。
2. GET方式提交的数据有长度限制，则POST的数据则可以非常大，这个是因为它们使用的操作系统和浏览器设置的不同引起的区别。也不是GET和POST本身的区别。
3. POST比GET安全，因为数据在地址栏上不可见。
4. GET和POST最大的区别主要是GET请求是幂等性的，POST请求不是。这个是它们本质区别，上面的只是在使用上的区别。幂等性是指一次和多次请求某一个资源应该具有同样的副作用。简单来说意味着对同一URL的多个请求应该返回同样的结果。因为get请求是幂等的，在网络不好的隧道中会尝试重试。如果用get请求增数据，会有重复操作的风险，而这种重复操作可能会导致副作用（浏览器和操作系统并不知道你会用get请求去做增操作）。

HTTP/1.1协议中共定义了八种方法（有时也叫“动作”），来表明Request-URL指定的资源不同的操作方式

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法

1. options：返回服务器针对特定资源所支持的HTML请求方法 或web服务器发送测试服务器功能（允许客户端查看服务器性能）
2. Get：向特定资源发出请求（请求指定页面信息，并返回尸体主题）
3. Post：向指定资源提交数据进行处理请求（提交表单、上传文件），有可能导致新的资源的建立或原有资源的修改
4. Put 向指定资源位置上上传其最新内容（从客户端向服务器传送的数据取代指定文档的内容）
5. Head 与服务器索与get请求一致的相应，响应体不会返回，获取包含在消息头中额度原信息（与get请求类似，返回的相应中没有具体内容，用于获取报头）
6. Delect 请求服务器删除request-URL所标示的资源*（请求服务器删除页面）
7. Trace 回显服务器收到的请求，用于测试和诊断
8. Connect HTTP/1.1协议中能够将连接改为管道方式的代理服务器

7、浏览器访问某网页的过程

假如我们输入baidu.com,回车

1. 先要解析出baidu.com对应的ip地址（DNS）
2. 需要知道默认网关的mac地址（ARP）
3. 拿到默认网关mac地址后，组织数据发送给默认网关（mac地址是默认网关的mac地址，但是ip是DNS服务器的地址）
4. 通过路由转发，到达DNS服务器所在的网关，再通过网关找DNS服务器，拿到baidu.com对应的IP
5. 取得IP后，用户和baidu.com的服务器tcp三次握手建立连接
6. 使用http协议响应用户的请求
7. 浏览器呈现
8. 浏览器关闭（tcp4次挥手）

8、HTTP和HTTPS的区别联系

HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了SSL（Secure Sockets Layer）协议用于对HTTP协议传输的数据进行加密，从而就诞生了HTTPS。

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和HTTP的区别主要如下：

1、https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

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。

3、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。

4、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，比http协议安全。

IP

1、IPv4/6的格式和数量

IPv4

地址长度：32位

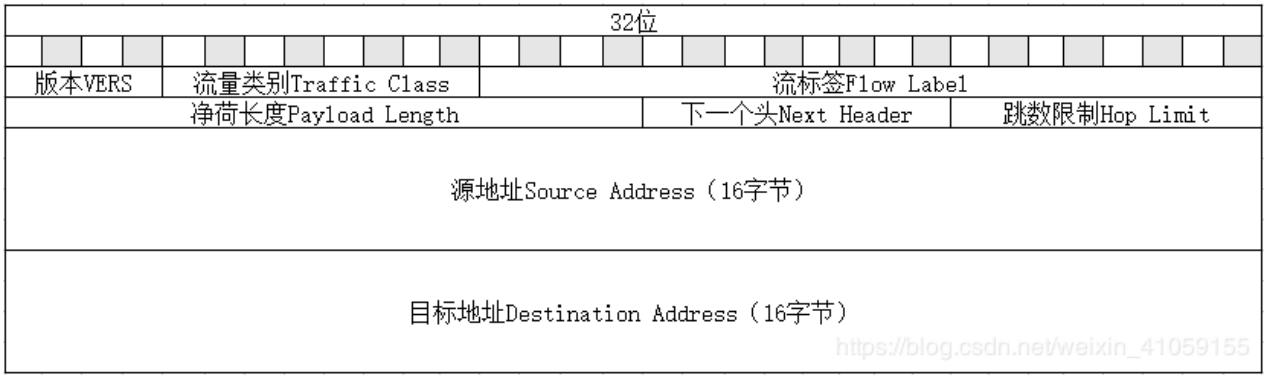
地址数量：2^{32}（约4×10^9)

<table>
<tr><td colspan="32">32位</td></tr>
<tr><td colspan="4">版本VERS</td><td colspan="4">头长度IHL</td><td colspan="8">服务类型TOS</td><td colspan="16">总长度Total Length</td></tr>
<tr><td colspan="24">标识Identification</td><td></td><td>DF</td><td>MF</td><td colspan="5">分段偏移Fragment Offset</td></tr>
<tr><td colspan="8">生命周期TTL</td><td colspan="8">协议Protocol</td><td colspan="16">头部校验和Header Checksum</td></tr>
<tr><td colspan="32">源地址</td></tr>
<tr><td colspan="32">目的地址</td></tr>
<tr><td colspan="32">选项（0或多个字）</td></tr>
<tr><td colspan="32">数据Data</td></tr>
</table>

IPv6

地址长度：128位

地址数量：2^{128}（约3.4×10^{38}）



IPv4与IPv6的区别：

1、扩展了路由和寻址的能力

2、报头格式的简化

3、对可选项更大的支持

4、QoS的功能

5、身份验证和保密：在IPv6中加入了关于身份验证、数据一致性和保密性的内容。

6、安全机制IPSec是必选的，IPv4的是可选的或者是需要付费支持的。

7、加强了对移动设备的支持：IPv6在设计之初有有着支持移动设备的思想，允许移动终端在切换接入点时保留相同的IP地址。

8、支持无状态自动地址配置，简化了地址配置过程。

2、IP数据报的首部有什么

版本号字段占4位： IP协议的版本号，一般有两个值，如果为4就代表是IPv4，6就代表是IPv6协议。
4→IPv4， 6 → IPv6

首部长度字段占4位： IP分组首部长度，这里是以四个字节为单位，如果值为5，则表示首部长度为20个字节（5×4）

服务类型(TOS)字段占8位：指示期望获得哪种类型的服务

总长度字段占16位： IP分组的总字节数(首部+数据)

生存时间（ TTL） 字段占8位： IP分组在网络中可以通过的路由器数（或跳步数）

协议字段占8位： 指示IP分组封装的是哪个协议的数据包

首部校验和字段占16位：实现对IP分组首部的差错检测

源IP地址、目的IP地址字段各占32位：分别标识发送分组的源主机/路由器(网络接口)和接收分组的目的主机/路由器（网络接口）的IP地址

选项字段占长度可变，范围在1~40B之间：携带安全、源选路径、时间戳和路由记录等内容 **实际上很少被使用**

填充字段占长度可变，范围在0~3B之间：目的是补齐整个 首部，符合32位对齐，即保证首部长度是4字节的倍数

3、IP子网划分

IP地址由32位二进制组成，32位二进制分成了4字节，每字节8位，字节之间用符.（点）分隔，为了方便人们记忆，经常需要转换成十进制数字显示，每字节最大为255（十进制）即二进制表示为11111111（8个1）。

A类网络的默认子网掩码为255.0.0.0，B类网络的默认子网掩码为255.255.0.0，C类网络的默认子网掩码为255.255.255.0

https://blog.csdn.net/vistas_fh/article/details/76167977

4、网络中的协议

ARP是地址解析协议

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。

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。

补充

1、TCP、UDP、IP、以太网报文格式以及重要字段。

以太帧

在链路层中，使用的最多的是以太网，而以太网帧因为历史原因存在多个版本，这里采用IEEE802.3以太网帧格式。

Preamble 7	SFD 1	dst MAC 6	src MAC 6	Length 2	Type 2	Data and Pad 46~1500	FCS 4

Preamble：前导码，7个字节，用于数据传输过程中的双方发送、接收的速率的同步

SFD：帧开始符，1个字节，表明下一个字节开始是真实数据（目的MAC地址）

dst MAC：目的MAC地址，6个字节，指明帧的接受者

src MAC：源MAC地址，6个字节，指明帧的发送者

Length：长度，2个字节，指明该帧数据字段的长度，但不代表数据字段长度能够达到（2^16）字节

Type：类型，2个字节，指明帧中数据的协议类型，比如常见的IPv4中ip协议采用0x0800

Data and Pad：数据与填充，46~1500个字节，包含了上层协议传递下来的数据，如果加入数据字段后帧长度不够64字节，会在数据字段加入“填充”至达到64字节

FCS：帧校验序列，4个字节，对接收网卡（主要是检测Data and Pad字段）提供判断是否传输错误的一种方法，如果发现错误，丢弃此帧。目前最为流行的用于FCS的算法是循环冗余校验（cyclic redundancy check –**CRC**）

ip帧

<table>
<tr><td colspan="32">32位</td></tr>
<tr><td colspan="4">版本VERS</td><td colspan="4">头长度IHL</td><td colspan="8">服务类型TOS</td><td colspan="16">总长度Total Length</td></tr>
<tr><td colspan="16">标识Identification</td><td></td><td>DF</td><td>MF</td><td colspan="13">分段偏移Fragment Offset</td></tr>
<tr><td colspan="8">生命周期TTL</td><td colspan="8">协议Protocol</td><td colspan="16">头部校验和Header Checksum</td></tr>
<tr><td colspan="32">源地址</td></tr>
<tr><td colspan="32">目的地址</td></tr>
<tr><td colspan="32">选项（0或多个字）</td></tr>
<tr><td colspan="32">数据Data</td></tr>
</table>

Version：版本，4位，用来表明IP协议实现的版本号，当前一般为IPv4，即0100，IPv6的为0110。这个字段确保可能运行不同IP版本的的设备之间的兼容性

IHL：报头长度，4位，以32比特的字来定义IP首部的长度，包括可选项。若该字段的值是5，即5*32=160比特=20字节。此字段最大值为60（*1532/8=60）字节 **TOS**：服务类型，8位，用于携带提供服务质量特征信息的字段，服务类型字段声明了数据报被网络系统传输时可以被怎样处理。其中前3比特为优先权子字段（Precedence，现已被忽略，各种终端都不采用）。第8比特保留未用。第4至第7比特分别代表延迟、吞吐量、可靠性和花费。当它们取值为1时分别代表要求最小时延、最大吞吐量、最高可靠性和最小费用。这4比特的服务类型中只能置其中1比特为1。可以全为0，若全为0则表示一般服务。大多数情况下该TOS会被忽略

Total Length：总长度，16位，指明整个数据报的长度，按字节为计算。最大长度为65535（2^16=65536）字节

Identification：标识，16位，用来唯一地标识主机发送的每一份数据报。IP软件会在存储器中维持一个计数器，每产生一个数据段，计数器就加1，并将此值赋给标识字段。但这个“标识”并不是序号，因为IP是无连接服务，数据报不存在按序接收的问题。如数据报由于超过网络的MTU而必须分片时，这个标识字段的值就被复制到所有的数据报的标识字段中。相同的标识字段的值使分片后各数据报片最后能正确的重装成为原来的数据报

Flags：标志，3位，分别是（RF, DF, MF），目前只有DF, MF有效。DF（don't fragment），置为0时表示可以分段，置为1是不能被分段；MF（more fragment），置为0时表示该数据段为最后一个数据段，置为1时表示后面还有被分割分段

Fragment offset：段偏移量，13位，指出较长的分组在分段后，某段在原分组的相对位置。也就是说相对用户字段的起点，该片从何处开始。段偏移以**8个字节**（有3位被flags占据）为偏移单位。这就是，每个分片的长度一定是8字节（64位）的整数倍

Time to live：生存期（TTL），8位，用来设置数据报最多可以经过的路由器数。由发送数据的源主机设置，通常为32、64、128等。每经过一个路由器，其值减1，直到0时该数据报被丢弃

Protcol：协议，8位，指明ip数据字段中的数据采用上层什么协议封装的。常见的有ICMP（1）、IGMP（2）、TCP（6）、UDP（17）

Header Checksum：头部校验和，16位，填充根据IP头部计算得到的校验和码。计算方法是：对头部中每个16比特进行二进制反码求和，但不和涉及头部后的数据字段

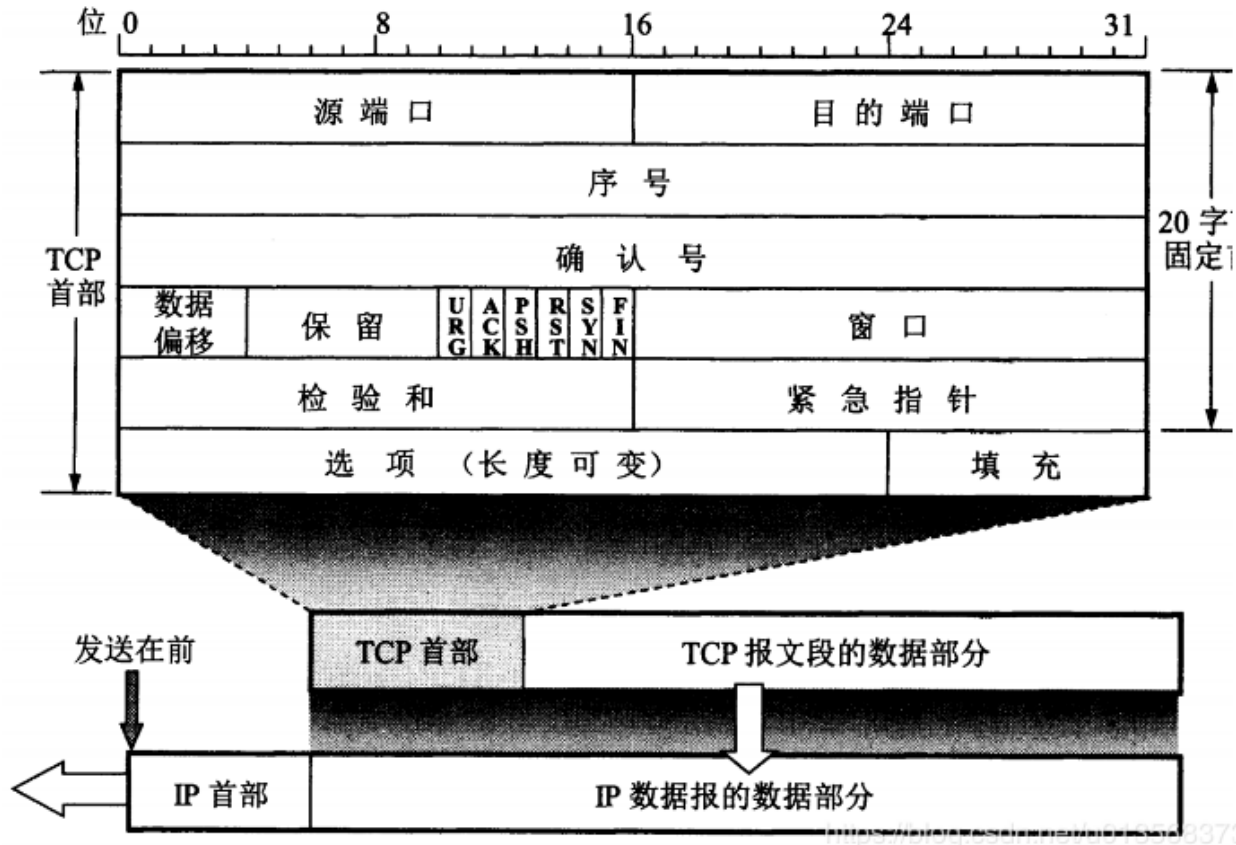
Source Address：源ip地址，32位，如（192.168.1.2）

Destination Address：目的ip地址，32位，如（192.168.1.3）

Option：选项，n*32位。用来定义一些可选项：如记录路径、时间戳等。但这些选项很少被使用，同时并不是所有主机和路由器都支持这些选项。可选项字段的长度必须是32比特的整数倍，如果不足，必须填充0以达到此长度要求。根据IHL可以得到option的长度

Data：数据，不定长度，但受限于数据报的最大长度（65535）。这是在数据报中要传输的数据。它是一个完整的较高层报文或报文的一个分片

TCP帧



src port：源端口，2个字节，是一个大于1023的16位数字，由基于TCP应用程序的用户进程随机选择

dst port：目的端口，2个字节，指明接收者所用的端口号，一般由应用程序来指定

Sequence number：顺序号，4个字节，用来标识从 TCP 源端向 TCP 目的端发送的数据字节流，它表示在这个报文段中的**第一个数据字节**的顺序号。如果将字节流看作在两个应用程序间的单向流动，则 TCP 用顺序号对每个字节进行计数。序号是 32bit 的无符号数，**序号到达 (2^32) − 1 后又从 0 开始。**当建立一个新的连接时， SYN 标志变 1 ，顺序号字段包含由这个主机选择的该连接的初始顺序号 ISN （ Initial Sequence Number ）** **

Acknowledgement number：确认号，4个字节，包含发送确认的一端**所期望收到**的下一个顺序号。因此，确认序号应当是上次已成功收到数据字节顺序号加 1 。只有 ACK 标志为 1 时确认序号字段才有效

Offset：报头长度，4位，给出报头中 32bit 字的数目。需要这个值是因为任选字段的长度是可变的。这个字段占 4bit ， 即TCP 最多有 60（15*4） 字节的首部

Resrvd：保留区域，6位，保留给将来使用，目前必须置为 0

Control Flags（6位）控制位包括

URG：为 1 表示紧急指针有效，为 0 则忽略紧急指针值

ACK：为 1 表示确认号有效，为 0 表示报文中不包含确认信息，忽略确认号字段

PSH：为 1 表示是带有 PUSH 标志的数据，指示接收方应该尽快将这个报文段交给应用层而不用等待缓冲区装满

RST：用于复位由于主机崩溃或其他原因而出现错误的连接。它还可以用于拒绝非法的报文段和拒绝连接请求。一般情况下，如果收到一个 RST 为 1 的报文，那么一定发生了某些问题

SYN：同步序号，为 1 表示连接请求，用于建立连接和使顺序号同步（ synchronize ）

FIN：用于释放连接，为 1 表示发送方已经没有数据发送了，即关闭本方数据流

Window Size：窗口大小，2个字节，表示**从确认号开始**，本报文的源方可以接收的字节数，即源方接收窗口大小。窗口大小是一个 16bit 字段，因而窗口大小最大为 65535（2^16 - 1）

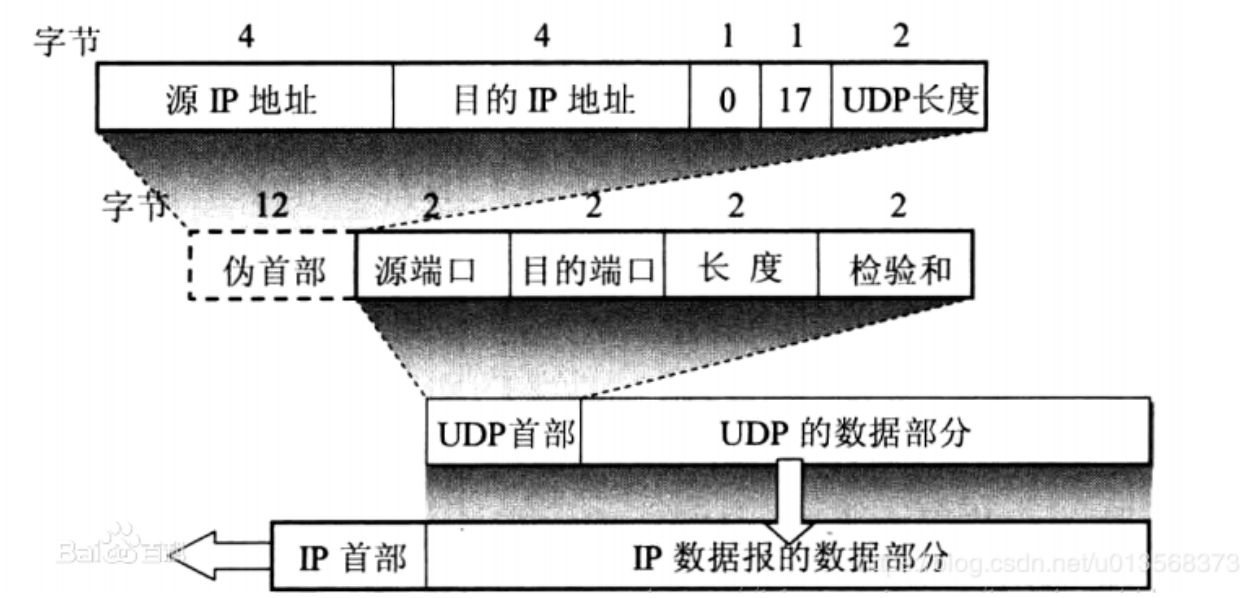
Checksum：校验和，2个字节，对整个的 TCP 报文段**（包括 TCP 头部和 TCP 数据）**，以 16 位字进行计算所得。这是一个强制性的字段，要求由发送端计算和存储，并由接收端进行验证

Urgent Pointer：紧急指针，2个字节，是一个正的偏移量，和顺序号字段中的值相加表示紧急数据最后一个字节的序号。 TCP 的紧急方式是发送端向另一端发送紧急数据的一种方式。 只有当URG 标志置 1 时紧急指针才有效

Option and Pad：选项和填充，n*4字节，常见的可选字段是最长报文大小 MSS(Maximum Segment Size) 。每个连接方通常都在通信的第一个报文段（为建立连接而设置 SYN 标志的那个段）中指明这个选项，它指明本端所能接收的最大长度的报文段。选项长度不一定是 32 位字的整数倍，所以要加填充位，使得报头长度成为整字数

Data：数据，不定长度，为上层协议封装好的数据

UDP帧



src port：源端口，2个字节，是一个大于1023的16位数字，由基于UDP应用程序的用户进程随机选择。

dst port：目的端口，2个字节，指明接收者所用的端口号，一般由应用程序来指定

Length：数据长度，2个字节，指明了**包括首部在内**的UDP报文段长度

Checksum：检验和，2个字节，指整个**UDP报文头和UDP所带的数据的校验和（也包括伪报文头）**。伪报文头不包括在真正的UDP报文头中，但是它可以保证UDP数据被正确的主机收到了

Data：数据字段，不定长度，为上层协议封装好的数据

2、网络数据包传送的过程

在整个数据报传输过程当中，

发送： 发送端进程首先调用系统调用，然后把数据发送给了socket，然后socket检查数据类型，调用系统调用send函数，send函数检查socket的状态，协议类型，传给了传输层，传输层对应的协议（UDP或者是TCP为这些数据创建数据结构），然后加入对应的传输层协议头部，然后交付给网际层，IP层，IP层加上它的头部，例如ip地址和检验和。然后决定是否分片，然后向下交付给数据链路层，数据链路层进行封装目的MAC和源MAC以及CRC校验。然后网卡调用中断驱动程序，发送到网络当中去。

接受： 数据报从网络中到达网卡，然后网卡接收到数据帧，放入网卡的缓存当中，向系统发送中断请求，执行中断处理程序，从网卡缓存当中读取到数据放入内存当中，然后把数据交给数据链路层，数据链路层进行解包，向上传递，IP层在对这个数据包进行差错检验等，此时如果是要接受的就向上层进行传递，如果不是，那么就丢弃或者转发，到达传输层，进行对应协议交付解包，然后向上到达应用层，交付给对应的协议，放入socket接受队列当中，然后接收的进程进行系统调用，获得数据，拷贝至进程缓冲区。然后返回用户态。

3、对称加密算法和非对称加密算法

对称加密算法：

这类算法在加密和解密时使用相同的密钥，或是使用两个可以简单地相互推算的密钥。

为什么叫对称加密呢，你可以这么理解，一方通过密钥将信息加密后，把密文传给另一方，另一方通过这个相同的密钥将密文解密，转换成可以理解的明文。他们之间的关系如下：

```
明文 <-> 密钥 <-> 密文
```

> 常见的对称加密算法有DES、3DES、AES、Blowfish、IDEA、RC5、RC6。

非对称加密算法：

非对称加密是一种比对称加密更加优秀的加密算法，当然算法有利有弊，对称加密速度快但是安全性相对于非对称加密来说低。

在这种密码学方法中，需要一对密钥(其实这里密钥说法不好，就是“钥”)，一个是私人密钥，另一个则是公开密钥。这两个密钥是数学相关，用某用户密钥加密后所得的信息，只能用该用户的解密密钥才能解密。如果知道了其中一个，并不能计算出另外一个。因此如果公开了一对密钥中的一个，并不会危害到另外一个的秘密性质。称公开的密钥为公钥；不公开的密钥为私钥。

这种加密算法应用非常广泛，SSH, HTTPS, TLS，电子证书，电子签名，电子身份证等等。

4、重定向和转发的区别

重定向的特点:redirect

1. 地址栏发生变化
2. 重定向可以访问其他站点(服务器器)的资源
3. 重定向是两次请求。不能使用request对象来共享数据

转发的特点:forward

1. 转发地址栏路径不变
2. 转发只能访问当前服务器下的资源
3. 转发是一次请求，可以使用request对象来共享数据