物理层

虽然涉及到的知识点众多，但我觉得重点是调制解调器

解调器对收到的数据(data)，将模拟信号转成数字信号。

因为电话线路传输的是模拟信号，所以调制器对于将要发送的数据，将数字信号转成模拟信号。

至于模拟信号和数字信号的话，我的理解是，链路上传输的信号确实是模拟信号，但是这个模拟信号模拟的是数字信号，也就是说是把模拟信号当数字信号处理的。

此外还有几种信道复用方式。分别是时分复用，统计时分复用，码分复用，频分复用。时分复用有点像RR。

集线器：集线器的主要功能是对接收到的信号进行再生整形放大，以扩大网络的传输距离，同时把所有节点集中在以它为中心的节点上。它工作于物理层。也就是说他没法选择路径，他只是放大信号而已。功能很单一。工作很简单。集线器只能连接速率相同的机器。比如都是10M的或者都是100M的。

拓展：

数字信号在物理层传输时，本质上是数字信号还是模拟信号，为什么？

<https://www.zhihu.com/question/29244850/answer/107289428>

数据链路层

这一层发送的数据单元叫帧，帧其实就是数据链路层把从网络层拿到的数据的首尾各加上一段数据。分别是帧首部(左,SOH)和帧尾部(右,EOT)。

添加首尾的作用之一是区分帧的开始和结束

链路层的协议规定了：

帧的数据部分(上一层拿到的数据)的最大长度(MTU)。

透明传输：字符填充法

可靠传输：差错检测(冗余检验法CRC) 实际上数据部分被添加了一个冗余码，所以上面写的有问题，并不是在上层拿到的数据两端直接添加首尾，而是先分好组(因为最大长度有限制所以要先分组)，然后在数据末尾添加n位冗余码，然后再添加首尾，这时候就叫帧了。

但是具体的荣誉检测法我并没有理解。之后有需要再学习。

流量控制：

带宽差异，如果发送快接收慢数据会丢失。所以收方控制发方的速度。

确认和重传机制。接收方正确接收（冗余检验法）帧之后，会有一个反馈信号(ACK)，然后发送方再发下一帧。如果收到的帧是错误的返回一个NAK ，发送方收到NAK之后会拿缓存里面的数据从新发送一次。

如果一定时间内接收方没有反馈信号，说明数据丢失了，这时候自动重传。

返回信号丢失也重传，因为发送方也分不出来到底是返回信号丢了还是数据没发送过去。反正超时计时器到了没收到信号就重传。

为了解决重复帧的问题，给帧加序号(估计是加在帧首部了)，序号只用0和1表示(也就是说只占1位)

协议：PPP(点对点协议) 任意两个物理设备间适用

上面讲的帧封装，冗余校验，确认重传等在这个协议里面都有详细规定。比如冗余校验的那个占位符，在PPP协议里面就有详细规定，规定到底用什么转义字符作为占位符。并且还规定开始符号和结束符号具体都是什么。还有帧的最大长度，并且还隔几分钟就检测一下链路是否通畅(数据链路协议LCP--建立，配置，测试链路的方法，控制链路上的节点(开关？)，这种控制是物理上的，抱着链路在物理上是连通的。)。

16进制mark地址，标识上网设备用的。

数据压缩

协议：P2P(对等计算) 包括LCP和NCP

NCP是网络控制协议。逻辑上控制节点(路由器)。

ppp的帧，PPP是面向字节的(字节为最小单位)，所以所有帧都是整数字节



F：标志字段，相当于SOH。

A：地址字段，网络层使用的现在是空闲状态。

C：控制字段

协议：因为ppp可以向上兼容任何协议，所以这里具体写什么取决于上层吧。

FCS：检验序列，也就是冗余码。看来冗余码不是写在ip数据报里面的。

F：标志字段

ppp为了区分帧首尾和ip数据报(透明性)，想出了两种方法。分别是字符填充法和0比特填充法。

ppp的字符填充

由硬件完成，将0x7E转0x7D和0x5E。将0x7D转0x7D和0x5D。控制字符前面加0x7D并且将控制字符的编码加以改变。

ppp 0比特填充

发送端发现有5个连续1时替换成0.接收端发现五个连续1的时候就把1后面的0删除。

关键：802.3标准将数据链路层分为两层，分别是逻辑链路控制(LLC)和媒体接入控制(MAC)。

但是设备厂商不管LLC层，而是把逻辑控制融入到MAC层中。

网卡叫通信适配器

1进行串行并行转换，就是把运输层的ip数据报封装成帧。

2对数据进行缓存(流量控制，可靠传输)

3在windows里面安装驱动程序，其实就是给windows提供控制接口。

4实现以太网协议，其实就是实现ppp协议。

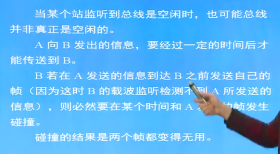
以太网：

站的概念。

数据链路层在以太网的情况下，没有重传，也就不需要对帧编号。

对数据的检测交给上一层来管理(网络层？运输层？应用层？)，如果上一层需要充传那就给发送方发送信号。在这种重传机制下，其实数据链路层就是个傻子，它只做数据发送和接收，即使数据重新发送了它也不知道，因为它不检测数据。

多点接入：多个计算机接一根总线上。

载波监听：发送数据前用电子技术检测总线上有没有其他计算机在发送的信号。有就不发，避免碰撞。

碰撞：碰撞发生时总线上的信号会失真。无法从信号中恢复出有效信息。碰撞之后会重传最多重传16次(二进制指数类型退避算法)

强化碰撞：发生碰撞后，发送干扰信号，让总线上所有计算机都知道发生了碰撞。

以太网规定凡是小于64字节的数据是由于发生碰撞而异常终止的无效帧

利用集线器扩展以太网

信道利用率的计算（帧越长，时延带宽积越低，a的值就越小。这时候信道利用率也就越高）

mac地址(传输过程中的物理设备的名字管理。)这里所说的地址其实是名字或标识符。802标准规定mac地址是48bit。为了方便，采用12个16进制表示mac地址。前三个16进制是厂家的标识符。后三个是拓展。最后6个是用来确定具体机器的?(“前6位用来标记特定厂商，后6位用来标记具体产品。”百度上有人这么说的)

路由有两个mac地址，其他设备是一个mac地址。一个网卡对应一个mac地址，路由有两个网卡。



网卡：收到mac帧之后，先看看是不是发给自己这个站的。如果不是就丢弃。如果是发给本站的(比如发给连到自己网络上的其他主机或者其他路由的)就收下。然后再进行其他处理。

发往本站的帧有三种：

单播帧：一对一 （我觉得应用SDN之后就会变成单播帧）

广播帧：一对全体

多播帧：一对多

mac帧和PPP帧的区别

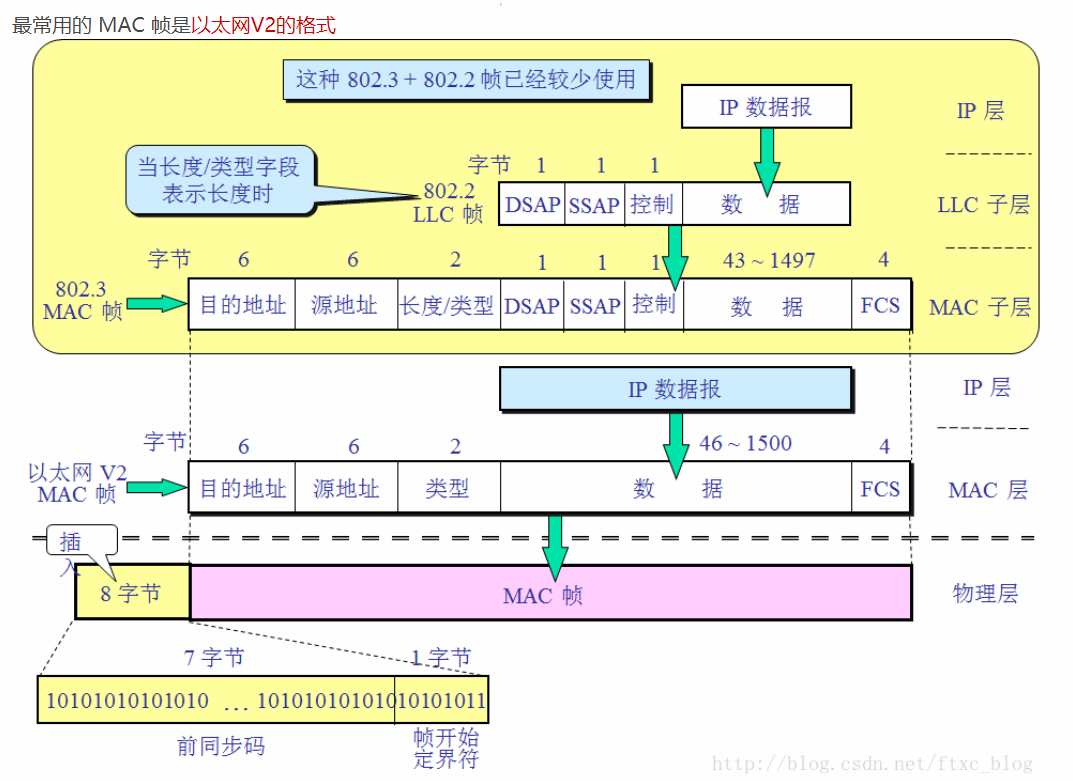
这里mac帧存在于以太网，而ppp帧是广域网。广域网是用ip地址寻址的(ip地址是网络层的寻址机制)，但是局域网内部有自己的寻址机制，局域网内部使用mac地址来做寻址(链路层)。

ppp属于广域网范畴，MAC是局域网范畴，按实际情况和环境就选用不同的协议，ppp支持的网络结构只能是点对点，mac支持多点对多点。

以太网中用mac，远程的话就用ppp（如ADSL拨号，就是基于ppp的）。

以太网封装和PPP封装都是对 IP 层的封装

这里mac帧其实是在ppp帧的基础上再进行添加首尾部分，从而组成mac帧。



类型字段 （2个字节）：

用来标志上一层使用的是什么协议，以便把收到的MAC帧的数据上交给上一层的这个 协议。其实就是为了向上兼容。(不过我猜测，它的作用只是让下一个接收这个帧的机 器以此来判断我这个帧是遵循哪个协议规范的帧格式。因为格式不同对于数据解释的规 则是不一样的。（就好像编码，不过这好像就是编码？））

数据字段 （46-1500）：

正式名称是MAC客户数据字段最小长度64 字节-18字节的首部和尾部 = 数据字段的 最小长度。最小长度就是46字节，因为规定传输的最小字节是64，首尾加起来有18了，所以数据字段最小是46。有效长度是64~1518。超过了也不行。

FCS字段 （4 字节）：

当数据字段的长度小于 46 字节时，应在数据字段的后面加入整数字节的填充字段，以 保证以太网的MAC帧长不小于64 字节(视频里面说这是冗余校验序列，我觉得这个比较靠谱，因为你填充不可能只有4字节啊，46加4也不等于64啊)

在帧的前面插入的 8 字节中的第一个字段共 7 个字节，是前同步码，用来迅速实现 MAC帧的比特同步。第二个字段是帧开始定界符，表示后面的信息就是MAC帧。一定要注意，8个字节不属于mac帧。

如果mac帧无效(不是整数字节、帧长不符合要求、FCS检测)，那就就把帧直接丢弃。以太网不需要接收端发送确认帧。也就是不进行确认。(那以太网没有流控吗？还是说重传就已经算流控了)

帧间最小间隔是9.6us，也就是争用期的时间。也是发送96bit数据的时间，一方面是用来确认对方是否正确接收数据(没有返回帧，所以到底怎么实现流控的呢？难道只要不发生碰撞就一定正确传输了？明显不是啊，那如果帧无效发送方怎么知道要不要重传呢？)，另一方面是给接收端一个处理帧的时间。

集线器：半双工(碰撞终止发送)

物理层 mac帧。多个集线器搭建网络的话，碰撞域就变大了，一个站发送信息，所有 集线器上面的站都要受到广播。其实多个集线器还是一根总线。

网桥：半双工(碰撞终止发送)

数据链路层 mac帧，拿到mac帧之后不是广播，而是先解析目的地址，然后根据目的 mac地址一对一的发送mac帧。

维护一个表

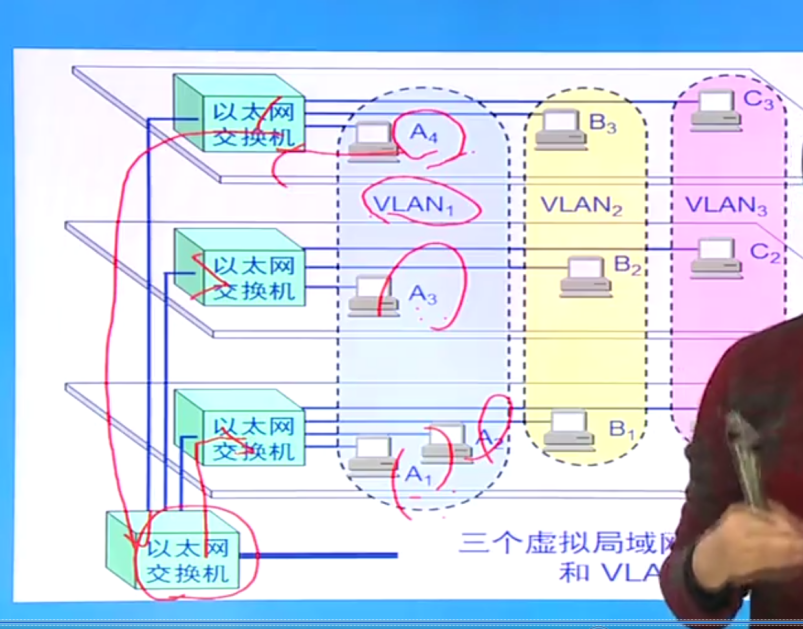
|  |  |
| --- | --- |
| 站地址(所有连接到总线上的机器的mac地址) | 端口(网桥一共俩端口，所以端口号不是1就是2) |
|  |  |

网桥替代集线器。一个集线器相当于一根总线，但是网桥拉出来的线都是总线。

交换机：全双工(无碰撞，每个站一条总线)

VLAN (虚拟局域网) 因为每台PC独占一条总线，所以用逻辑上构建VLAN。

详细解释的话，其实就是在mac帧原地址和类型之间再插入一个4字节的片段。叫做vlan标记(tag)，交换机通过这个tag可以知道这条信息属于哪个站。值得注意的是，即使这里插入了4字节的片段，也不和原来的46~1500冲突。最小字节数还是46~1500。



载波延伸(只是半双工1000M网络才用到，扩充mac帧，最小512字节(原来64))

分组突发(只延伸第一个帧，后面帧与前面的是0.96us就算是一起的数据。不需要单独进行 载波延伸)

网络层

虚电路服务：X.25协议

流控是由网络负责的(路由器负责？)

虚电路号存在X.25分组的首部

数据报服务：IP协议(TCP/IP协议体系之一)

流控是由发送方和接收方负责的。(也就是之前学的确认重传机制，做这件事情的不是路由而是两头的pc(的网卡)。？)

IP协议配套使用的有三个协议，分别是

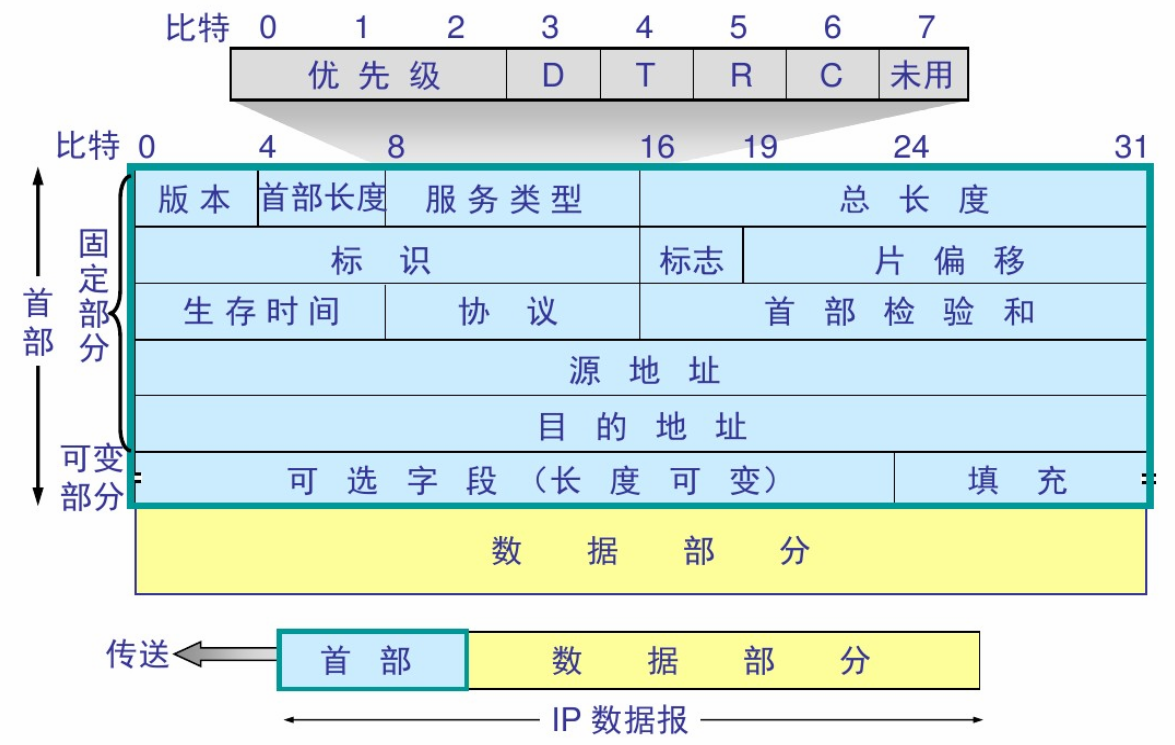
地址解析协议(ARP)、逆向地址解析协议(RARP)：为了向下兼容

网际控制报文协议(ICMP)、网际组管理协议(IGMP)：为了向上兼容



地址解析协议就是根据IP地址获取到mac地址。

网络层之上利用IP地址寻址，数据链路层之下用mac地址寻址。



IP数据报前20个字节是固定部分。

**版本**：

IP协议的版本，目前的IP协议版本号为4，下一代IP协议版本号为6。

**首部长度：**

IP报头的长度。固定部分的长度（20字节）和可变部分的长度之和。共占4位。最大 为1111，即10进制的15，代表IP报头的最大长度可以为15个32bits（4字节），也 就是最长可为15\*4=60字节，除去固定部分的长度20字节，可变部分的长度最大为40 字节。

**服务类型**：

Type Of Service。

**总长度**：

IP报文的总长度。报头的长度和数据部分的长度之和。

**标识**：

唯一的标识主机发送的每一分数据报。通常每发送一个报文，它的值加一。当IP报文 长度超过传输网络的MTU（最大传输单元）时必须分片，这个标识字段的值被复制到 所有数据分片的标识字段中，使得这些分片在达到最终目的地时可以依照标识字段的内 容重新组成原先的数据。

**标志**：

共3位。R、DF、MF三位。目前只有后两位有效，DF位：为1表示不分片，为0表示 分片。MF：为1表示“更多的片”，为0表示这是最后一片。

**片位移**：

本分片在原先数据报文中相对首位的偏移位。（需要再乘以8）。

较长的分组在分片后，某片在原分组中的相对位置。也就是说，相对**用户数据字段**的起 点，该片从何处开始。片偏移以8个字节为偏移单位。这就是说，每个分片的长度一定 是8字节（64位）的整数倍。

计算：每个数据报片的第一个字节是原数据报(运输层)的第几个字节。如果是第0个字节，那么片偏移就是0/8=0。如果是1400/8=175。

**生存时间**：

IP报文所允许通过的路由器的最大数量。每经过一个路由器，TTL减1，当为0时，路 由器将该数据报丢弃。TTL 字段是由发送端初始设置一个 8 bit字段.推荐的初始值由分 配数字 RFC 指定，当前值为 64。发送 ICMP 回显应答时经常把 TTL 设为最大值 255。

**协议**：

指出IP报文携带的数据使用的是那种协议，以便目的主机的IP层能知道要将数据报上 交到哪个进程（不同的协议有专门不同的进程处理）。和端口号类似，此处采用协议号， TCP的协议号为6，UDP的协议号为17。ICMP的协议号为1，IGMP的协议号为2.

**首部校验和**：

计算IP头部的校验和，检查IP报头的完整性。

**源IP地址**：

标识IP数据报的源端设备。

**目的IP地址**：

标识IP数据报的目的地址。

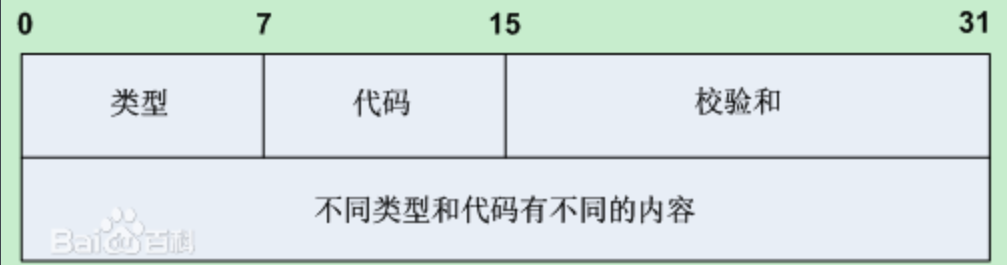
ICMP：是一种面向无连接的协议，网络层协议

<http://blog.csdn.net/tigerjibo/article/details/7356936>(必看)

ICMP是（Internet Control Message Protocol）Internet控制[报文](https://baike.baidu.com/item/%E6%8A%A5%E6%96%87" \t "https://baike.baidu.com/item/ICMP/_blank)协议。它是[TCP/IP协议族](https://baike.baidu.com/item/TCP/IP%E5%8D%8F%E8%AE%AE%E6%97%8F" \t "https://baike.baidu.com/item/ICMP/_blank)的一个子协议，用于在IP[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA" \t "https://baike.baidu.com/item/ICMP/_blank)、[路由](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1" \t "https://baike.baidu.com/item/ICMP/_blank)器之间传递控制消息。控制消息是指[网络通](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A" \t "https://baike.baidu.com/item/ICMP/_blank)不通、[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA" \t "https://baike.baidu.com/item/ICMP/_blank)是否可达、[路由](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1" \t "https://baike.baidu.com/item/ICMP/_blank)是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

当遇到IP数据无法访问目标、IP[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8" \t "https://baike.baidu.com/item/ICMP/_blank)无法按当前的传输速率转发[数据包](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%8C%85" \t "https://baike.baidu.com/item/ICMP/_blank)等情况时，会自动发送ICMP消息。ICMP报文在IP帧结构的首部**协议(上面的图里能找到，一共8bit)**类型字段（Protocol 8bit)的值=1.

如下图所示，ICMP包有一个8字节长的包头，其中前4个字节是固定的格式，包含8位类型字段，8位代码字段和16位的校验和；后4个字节根据ICMP包的类型而取不同的值。



我们在网络中经常会使用到ICMP协议，比如我们经常使用的用于检查网络通不通的[Ping](https://baike.baidu.com/item/Ping" \t "https://baike.baidu.com/item/ICMP/_blank)命令（Linux和Windows中均有），这个“Ping”的过程实际上就是ICMP协议工作的过程。还有其他的[网络命令](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%91%BD%E4%BB%A4" \t "https://baike.baidu.com/item/ICMP/_blank)如[跟踪路由](https://baike.baidu.com/item/%E8%B7%9F%E8%B8%AA%E8%B7%AF%E7%94%B1" \t "https://baike.baidu.com/item/ICMP/_blank)的Tracert命令也是基于ICMP协议的。

icmp传输错误时候，就作为普通的ip数据进行传输。只不过协议字段是1，接收端看到这个1就知道这个报文是icmp的报错报文， 而不是用户数据了。但无论是报错报文还是用户数据，他们在传输过程中都是没有区别的。只不过接收这个报文之后，对报文的解析过程有所区别。

类型分两种：差错报文、询问报文

差错报文：终点不可达，源站抑制，时间超过，参数问题，改变路由(重定向)。

内部网关协议(IGP):RIP，OSPF、IS-IS、IGRP、EIGRP。

外部网关协议(EGP):只有BGP-4

路由选择算法：

<http://blog.csdn.net/lpjishu/article/details/52413919>

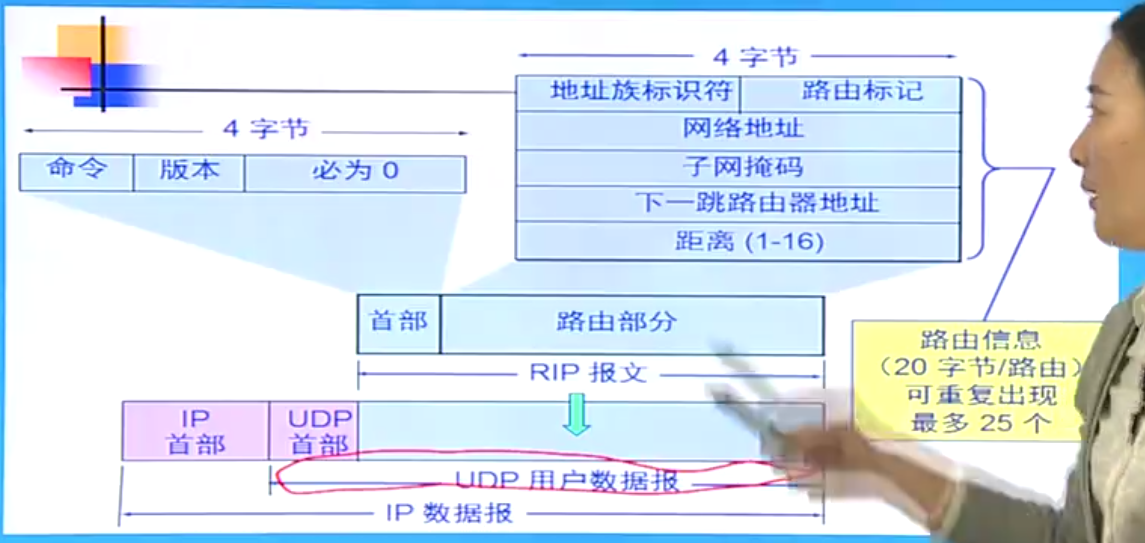
<http://blog.csdn.net/qq_33936481/article/details/54379818>

RIP协议(UDP报文传输)：

<http://blog.csdn.net/lycb_gz/article/details/1612254>

用**运输层**进行**路由表**的传输，**RIP报文格式**格式如下。

<http://blog.sina.com.cn/s/blog_c079d59e0102wgn9.html>



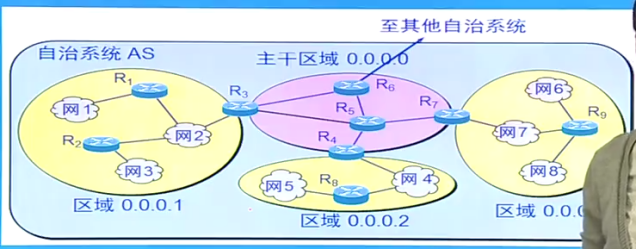
RIP报文最大长度为512字节。RIP报文头部占用4个字节,而每个路由条目占用20个八位组字节。因此,RIP消息最大为4+(25\*20)=504个字节,再加上8个字节的UDP头部,所以RIP数据报的大小(不含IP包的头部)最大可达512个字节。

路由标记是自治系统的号码，目的是过滤掉本自治系统外的路由选择信息。（其他自治系统的路由选择信息也有可能传进来。）

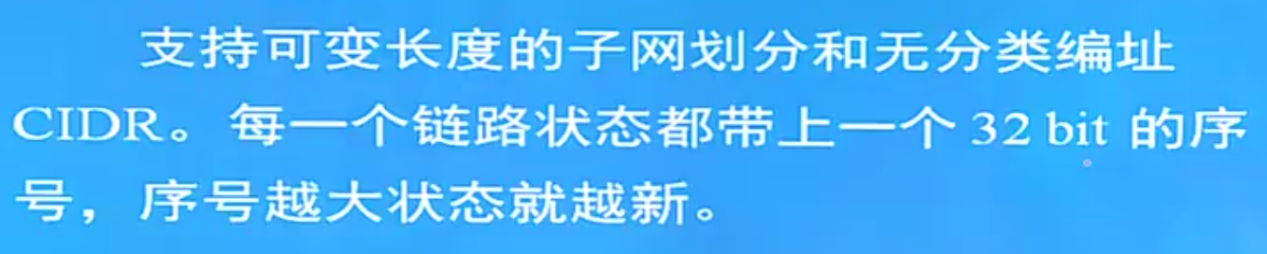
OSPF协议(IP数据报传输)：

<http://blog.csdn.net/qq_16811963/article/details/52144299>

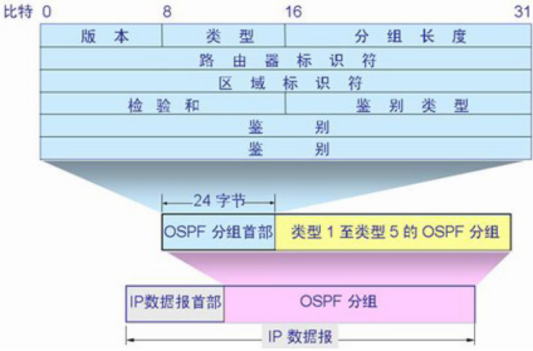
<http://blog.csdn.net/ztguang/article/details/70949781>



使用IP数据报传输。



OSPF数据报格式：



版本号:

这个字段表明了OSPF的当前版本号。最近的版本是2。版本1和版本2不兼容。

类型:

这个字段表明附加在分组头后面的是OSPF分组5种类型中的哪一种。

分组长度:

这个字段包含了整个OSPF分组的长度，这一长度包括了OSPF头在内。

路由器ID:

这个字段包含有4个字节的IP地址。路由器ID用于在自治系统中唯一地标识该路由器。 对一个Cisco路由器来说，这个字段包含此路由器的最大的（highest）IP地址。如果配 置了回环地址（loopback），那么最大的回环地址就成为了路由器ID。在路由器ID被 选择了之后，它将一直不改变，除非路由器重启、被选择作为路由器ID的接口关闭， 或者是这个接口上的IP地址已经被删除或取代。

区域ID:

这个字段指定分组所属的区域号。这也是一个4字节编号。为了形成邻居关系，它的值 在两方必须是相同的。这个字段的书写有两种方式：区域1或者区域0.0.0.1。这两种书 写方式没有区别。

校验和:

这个字段包含有除了对数据损坏的认证部分之外的整个OSPF分组的校验。

AuType:

这个字段包含有认证的类型码：0表示没有认证（null authentication）。1表示认证类 型为普通文本。2表示认证类型为MD5。

认证:

如果是普通文本认证则这个64比特的字段包含有认证密钥。如果是消息摘要认证，则 这个64比特的认证字段被重新定义为一些其他的参数。参见附录D，RFC2328，了解关 于MD5认证方案的更多细节。

BGP协议(TCP报文)：

<http://www.360doc.com/content/10/1230/16/2614615_82680038.shtml>

支持CIDR，所以BGP的路由表里面应该有网络前缀和下一跳路由，以及到达目的路由 要经过的各个自治系统序列

IP多播(IPv6)：借助网际组管理协议IGMP，多播路由选择协议。

<https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/6867723?fr=aladdin>

多播是IPv6数据包的3种基本目的地址类型之一，多播是一点对多点的通信，IPv6没有采用IPv4中的组播术语，而是将广播看成是多播的一个特殊例子。

**IP多播（也称多址广播或[组播](https://baike.baidu.com/item/%E7%BB%84%E6%92%AD" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)）技术，是一种允许一台或多台主机（多播源）发送单一[数据包](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%8C%85" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)到多台主机（一次的，同时的）的TCP/[IP网络技术](https://baike.baidu.com/item/IP%E7%BD%91%E7%BB%9C%E6%8A%80%E6%9C%AF" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)。多播作为一点对多点的通信，是节省[网络带宽](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%B8%A6%E5%AE%BD" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)的有效方法之一。在网络音频/视频广播的应用中，当需要将一个[节点](https://baike.baidu.com/item/%E8%8A%82%E7%82%B9" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)的信号传送到多个节点时，无论是采用重复点对点通信方式，还是采用广播方式，都会严重浪费网络带宽，只有多播才是最好的选择。多播能使一个或多个多播源只把数据包发送给特定的多播组，而只有加入该多播组的主机才能接收到数据包。目前，IP多播技术被广泛应用在网络音频/视频广播、[AOD](https://baike.baidu.com/item/AOD" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)/VOD、[网络视频会议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E8%A7%86%E9%A2%91%E4%BC%9A%E8%AE%AE" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)、多媒体远程教育、“push”技术（如股票行情等）和[虚拟现实](https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E7%8E%B0%E5%AE%9E" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)游戏等方面。**

要实现IP多播通信，要求介于多播源和接收者之间的[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)、[集线器](https://baike.baidu.com/item/%E9%9B%86%E7%BA%BF%E5%99%A8" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)、[交换机](https://baike.baidu.com/item/%E4%BA%A4%E6%8D%A2%E6%9C%BA" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)以及主机均需支持IP多播。目前，IP多播技术已得到硬件、软件厂商的广泛支持。

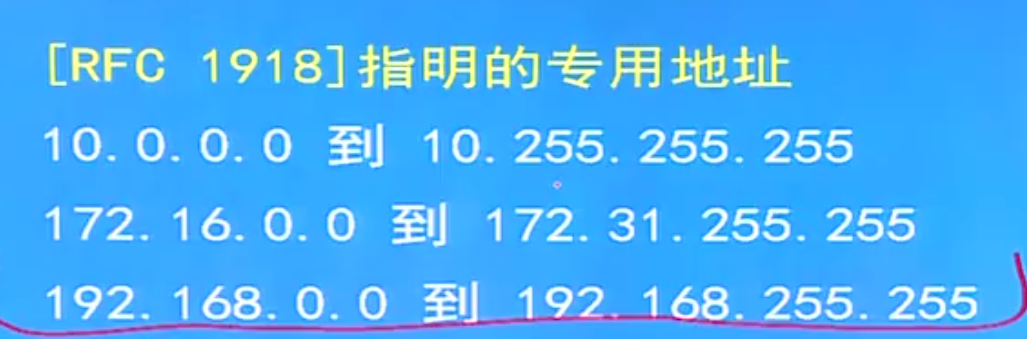
**多播通信要求多播源[节点](https://baike.baidu.com/item/%E8%8A%82%E7%82%B9" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)和目的节点之间的所有路由器必须提供对Internet组管理协议（IGMP）、多播[路由协议](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%8D%8F%E8%AE%AE" \t "https://baike.baidu.com/item/%E5%A4%9A%E6%92%AD/_blank)（如PIM、DVMRP等）的支持。**

其实看了这么多也不懂到底怎么实现的，全球那么多电脑，难道要全世界共享多播组的列表吗？如果是这样那多播不见得多实用。

隧道技术：

有一些路由不支持多播，所以把多播数据报前面加一个单播数据报的首部，装作是单播数据报通过这个路由传输。

虚拟地址：这些地址路由器不会转发。



VPN:想和因特网上的主机通信（需要加密）时

<https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E4%B8%93%E7%94%A8%E7%BD%91%E7%BB%9C/8747869?fromtitle=VPN&fromid=382304&fr=aladdin>

NAT:想和因特网上的主机通信（并不需要加密）时

<https://www.cnblogs.com/bo083/articles/2170189.html>

**运输层**

**TCP(面向连接，可靠)：**

TCP是因特网中的传输层协议，使用[三次握手协议](https://baike.baidu.com/item/%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B%E5%8D%8F%E8%AE%AE" \t "https://baike.baidu.com/item/TCP/_blank)建立连接。

这里所说的连接，是逻辑上的连接，**路由器其实并不知道创建了连接**。

TCP的连接并不是指一条实际的或虚拟的链路存在于数据交换的两端，而是指连接的双方都维护了一些资源（如输入输出缓冲区、多种定时器）以及链路状态的信息，并通过双方的控制报文交互管理状态、向用户提供接口修改这些资源的分配。

<https://baike.baidu.com/item/TCP/33012?fr=aladdin>(百度上的TCP基础知识)

<http://network.51cto.com/art/201411/456783.htm>(关于三次握手和四次分手)

<https://www.jianshu.com/p/ef892323e68f>(关于一些其他细节)

<https://www.cnblogs.com/yueminghai/p/6646043.html>(流控)(传输层2, 1时14分左右)

<https://baike.baidu.com/item/Nagle%E7%AE%97%E6%B3%95/5645172?fr=aladdin>(百度Nagle)

<http://blog.csdn.net/yuan1125/article/details/51536490>(Nagle算法详解,结合了实际代码)

<https://baike.baidu.com/item/%E7%B3%8A%E6%B6%82%E7%AA%97%E5%8F%A3%E7%BB%BC%E5%90%88%E5%BE%81/6167377?fr=aladdin>(糊涂窗口综合征的解释以及解决方法)

<https://www.cnblogs.com/zhaoyl/archive/2012/09/20/2695799.html>(糊涂窗口和Nagle参考1)

<http://blog.csdn.net/wdscq1234/article/details/52463952>(糊涂窗口以及Nagle参考2)

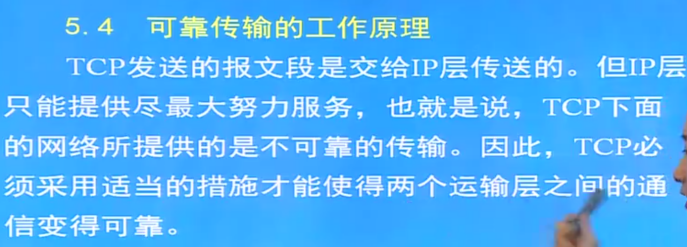
<http://blog.csdn.net/wdscq1234/article/details/52432095>(糊涂窗口以及Nagle参考3)

<https://www.cnblogs.com/yueminghai/p/6646043.html>(TCP知识补全)

它传输的数据叫 TCP报文段

不提供广播和多播

面向字节流



每发送一个分组之后，等待对方的确认信息，收到确认信息之后再发送下一个分组。（有点像流控）

也有超时重传。

首部有20字节。

当应用层向TCP层发送用于网间传输的、用8位字节表示的[数据流](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%B5%81" \t "https://baike.baidu.com/item/TCP/_blank)，TCP则把数据流分割成适当长度的报文段，最大传输段大小（[MSS](https://baike.baidu.com/item/MSS/3567770" \t "https://baike.baidu.com/item/TCP/_blank)）通常受该计算机连接的网络的数据链路层的最大传送单元（[MTU](https://baike.baidu.com/item/MTU" \t "https://baike.baidu.com/item/TCP/_blank)）限制。之后TCP把数据包传给IP层，由它来通过网络将包传送给接收端实体的TCP层。

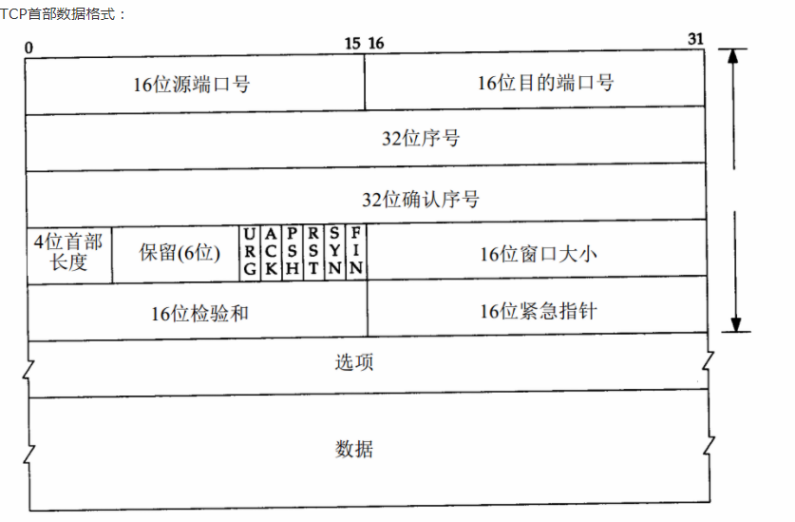
TCP为了保证报文传输的可靠[1]  ，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的字节发回一个相应的确认(ACK)；如果发送端实体在合理的往返时延(RTT)内未收到确认，那么对应的数据（假设丢失了）将会被重传。

在数据正确性与合法性上，TCP用一个校验和函数来检验数据是否有错误，在发送和接收时都要计算校验和；同时可以使用md5认证对数据进行加密。

在保证可靠性上，采用超时重传和捎带确认机制。

在流量控制上，采用滑动窗口[1]  协议，协议中规定，对于窗口内未经确认的分组需要重传。https://baike.baidu.com/item/ARQ/7402812

在拥塞控制上，采用广受好评的TCP拥塞控制算法（也称AIMD算法）。该算法主要包括三个主要部分：1）加性增、乘性减；2）慢启动；3）对超时事件做出反应。



<http://blog.csdn.net/mary19920410/article/details/58030147>

<https://www.jianshu.com/p/e2dd554279d7>

* 16位源都口号，16为目的端口号用于寻找发送端和接收端的应用进程，加上IP首部的源端IP及终端IP，唯一的确认一个TCP连接。
* 32位序号：标识发送的数据字节流，标识在这个报文段中的第一个数据字节，2^3 - 1后重新从0开始。包含该主机选择的连接的ISN(Initial Sequence Number)，要发送的第一个数据字节序号为ISN+1.
* 32位确认序号：ACK为1时有效，上次成功收到的数据字节序号+1（如接收到的为1024--2048，则返回2049）。
* 4位首部长度：首部中32bits字的数目，TCP最多有60字节的长度，除去任选字段，正常为20字节。
* 6bits：URG紧急指针；ACK确认序号有效；PSH接收方应尽快将此报文段交给应用层；RST重建连接；SYN同步序号，用来发起一个新连接；FIN发端完成发送任务。
* 16位窗口大小：TCP流量控制，字节数，起始于确认序列号指明的值，接收端期望收到的字节，最大为65535.
* 16位检验和：包括计算TCP首部和数据综合的二进制反码和检验和。
* 16位紧急指针：URG为1时有效，正向的偏移量，加上序号字段值表示最后一个字节的序号。
* 可选字段：例：MSS.

## **超时重传时间选择(传输层2,60:00左右)**

<https://www.cnblogs.com/yueminghai/p/6646043.html>

**具体实现:**

TCP 每发送一个报文段,就对这个报文段设置一次计时器.只要计时器设置的重传时间到但还没有收到确认,就要重传这一报文段

**加权平均往返时间:**

**做法:**

TCP 保留了 RTT 的一个加权平均往返时间 RTTS（这又称为平滑的往返时间）,第一次测量到 RTT 样本时,RTTS 值就取为所测量到的 RTT 样本值.以后每测量到一个新的 RTT 样本,就按下式重新计算一次 RTTS：

**公式:**

### ****新的 RTTS = ( 1 - α)×(旧的 RTTS)＋α(新的 RTT 样本)****

**说明:**

式中,0 ≤ α＜ 1.若α很接近于零,表示 RTT 值更新较慢若选择 α 接近于1,则表示 RTT 值更新较快 RFC 2988 推荐的 α 值为 1/8,即 0.125

**UDP(无连接，不可靠)：**

**udp没有拥塞控制。**

**<https://baike.baidu.com/item/UDP/571511?fr=aladdin>**

使用UDP协议包括：[TFTP](https://baike.baidu.com/item/TFTP" \t "https://baike.baidu.com/item/UDP/_blank)、[SNMP](https://baike.baidu.com/item/SNMP" \t "https://baike.baidu.com/item/UDP/_blank)、NFS、DNS、BOOTP

它传输的数据叫UDP报文或用户数据报。

支持1:1 1:n n:n通信。

只支持交替式的半双工通信

首部只有8字节

UDP有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。

它不提供报文到达确认、排序、及流量控制等功能。

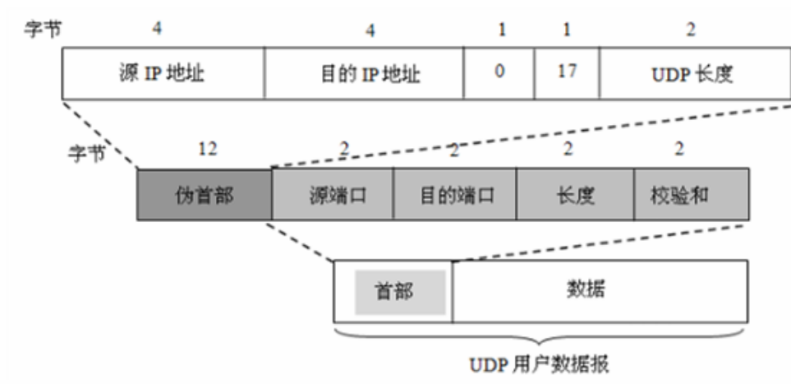
UDP是[OSI](https://baike.baidu.com/item/OSI" \t "https://baike.baidu.com/item/UDP/_blank)参考模型中一种无连接的传输层协议，它主要用于不要求分组顺序到达的传输中，分组传输顺序的检查与排序由应用层完成[2]  ，

UDP报文没有可靠性保证、顺序保证和流量控制字段等，可靠性较差。但是正因为UDP协议的控制选项较少，在数据传输过程中延迟小、数据传输效率高，适合对可靠性要求不高的应用程序，或者可以保障可靠性的应用程序，如DNS、TFTP、SNMP等。

数据报的最大长度根据操作环境的不同而各异。从理论上说，包含报头在内的数据报的最大长度为65535字节。不过，一些实际应用往往会限制数据报的大小，有时会降低到8192字节。

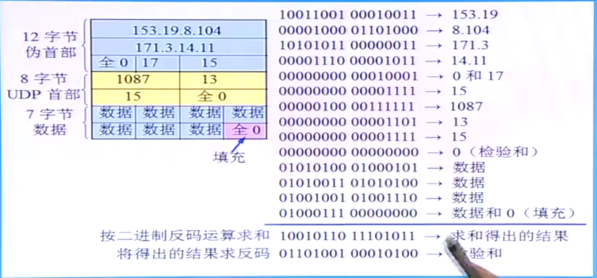
事实上，UDP协议的这种乱序性基本上很少出现，通常只会在网络非常拥挤的情况下才有可能发生。





1. ****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.   
   ****2. 目的端口：****目的端口号。这在终点交付报文时必须要使用到。   
   ****3. 长度：**** UDP用户数据报的长度，其最小值是8（仅有首部）。   
   ****4. 检验和：****检测UDP用户数据报在传输中是否有错。有错就丢弃。

伪首部：



端口号：16bit

端口号的范围从0到65535

大多数TCP/IP实现给临时端口号分配1024---5000之间的端口号。大于5000的端口号是为其他服务器预留的。

套接字(socket，只存在于TCP):

IP地址：端口号

网络编程的api接口简称也是soket。

TCP拥塞控制:慢开始、拥塞避免、快重传、快恢复

<http://blog.csdn.net/sinat_21112393/article/details/50810053>(参考1)

<https://www.zhihu.com/question/23055347/answer/197672107>(结合现实情况的一些科普,大部分内容是可信的。)

当cwnd<ssthresh时，使用慢开始算法。

当cwnd>ssthresh时，改用拥塞避免算法。

当cwnd=ssthresh时，慢开始与拥塞避免算法任意。

拥塞避免：

拥塞窗口缓慢增长。即每经过一个往返时间RTT就把发送方的拥塞窗口cwnd 加1，而不是加倍。这样拥塞窗口按线性规律缓慢增长。

慢开始：

拥塞窗口缓慢增长。一次传输轮次之后拥塞窗口就加倍。这就是乘法增长，和后面的拥塞避免算法的加法增长比较。

1.初始值是最大报文段MSS(这是视频的说法,和网上说法不一样)

快重传(接收方)：

要求接收方在收到一个失序的报文段后就立即发出重复确认。(会优先于超时重传的反 应时间，也就是在超时计时器结束之前，发送端就可以通过三条重复确认报文得知数据 报丢失)

快恢复(发送方)

①当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把ssthresh门限减半。(视频说是门限值+3MSS)但是接下去并不执行慢开始算法。

②考虑到如果网络出现拥塞的话就不会收到好几个重复的确认，所以发送方现在认为网络可能没有出现拥塞。所以此时不执行慢开始算法，而是将cwnd设置为ssthresh的大小，然后执行拥塞避免算法。

乘法减小：

无论是在****慢开始阶段****还是在****拥塞避免阶段****，只要发送方判断网络出现拥塞（其根据就是 没有收到确认，虽然没有收到确认可能是其他原因的分组丢失，但是因为无法判定，所 以都当做拥塞来处理），就把慢开始门限设置为出现拥塞时的发送窗口大小的一半。然 后把拥塞窗口设置为1，执行慢开始算法

**应用层**

域名。

<https://baike.baidu.com/item/%E5%9F%9F%E5%90%8D/86062?fr=aladdin>

文件传输协议FTP(使用TCP)

安装FTP软件(默认21端口)

TFTP协议(使用UDP)

<https://baike.baidu.com/item/tftp/455170?fr=aladdin>

<http://blog.csdn.net/zlj_fly/article/details/40227623>

远程终端协议 TELNET协议(其实就是远程协助协议,使用tcp协议)

NVT格式

WWW协议

1. HTTP协议

<https://www.cnblogs.com/ranyonsue/p/5984001.html>(详解,写的很好，我觉得比较全面，但还是有可以补全的地方。)

<https://www.cnblogs.com/li0803/archive/2008/11/03/1324746.html>(报头参数更详细的解释)

<https://www.cnblogs.com/TankXiao/archive/2012/02/13/2342672.html>(详解2)

<http://blog.sina.com.cn/s/blog_4b9b714a01017lwr.html>(拓展：SQL注入)

<http://blog.csdn.net/myfuturein/article/details/8046899>(拓展：跨站请求伪造)

CGI：

<http://www.jdon.com/idea/cgi.htm#I3>(有一些基本的解释)

其实对于客户端发来的请求，我都是根据请求类型和参数来做不同的反馈的，对于任何的请求我的处理过程都是在服务器这一个脚本内执行的，并没有说为了某一个功能单独写一个脚本然后用服务器脚本去调用它。总之我不太理解CGI这个东西，因为没用过。

不过网上也有说法是“Common Gateway Interface，简称CGI。在物理上是一段程序，运行在服务器上，提供同客户端 HTML页面的接口”。所以其实我之前写过的HTML服务器脚本就是一个CGI，这种理解是比较广义的。而第一段的理解是狭义的。

静态页面、动态页面、活动万维网文档

其实静态页面，动态页面和活动万维网文档这三种区分的不是很明显。现在一个页面它即是动态的也是活动的。

万维网信息检索

就是喜闻乐见的爬虫，原理很简单，所以几十行代码就可以写一个简单的爬虫。但是优秀的爬虫算法很复杂，比如谷歌的底层。

SMTP协议(用于发送邮件，TCP协议)

<http://blog.csdn.net/kerry0071/article/details/28604267>

SMTP使用TCP端口25

不能发送.exe

邮件内容只能是ASCII码

限定邮件长度(包括附件)

MIME协议(补充SMTP协议)



POP协议(用于接收邮件，TCP协议)

<https://baike.baidu.com/item/pop%E5%8D%8F%E8%AE%AE/734244?fr=aladdin>

SMTP使用TCP端口110

IMAP协议

DHCP协议(动态主机配置协议，UDP)