link: null
title: 珠峰架构师成长计划
description: Open System Interconnection适用于所有的网络
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=208 sentences=502, words=1949

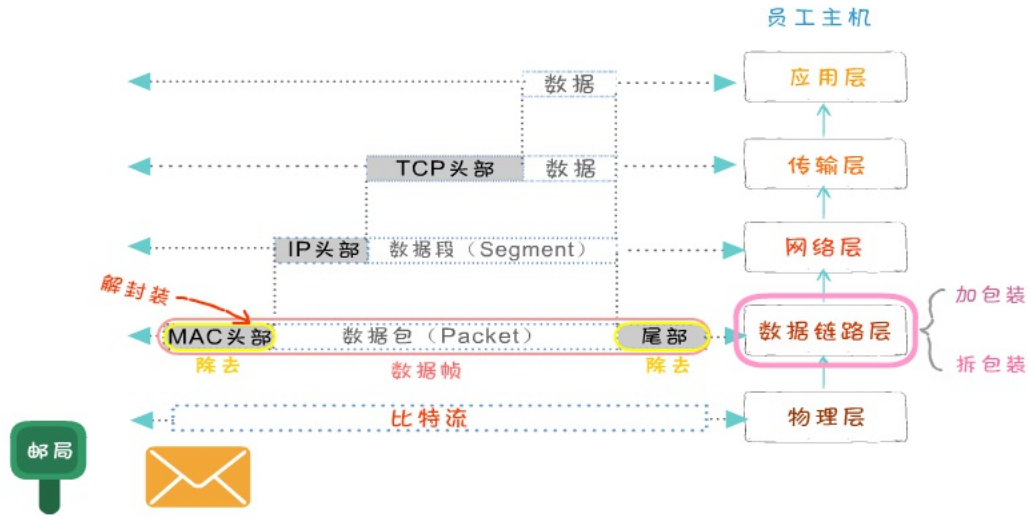
1. OSI七层模型

Open System Interconnection适用于所有的网络

- 分工带来效能
- 将复杂的流程分解为几个功能相对单一的子进程
- 整个流程更加清晰，复杂问题简单化
- 更容易发现问题并针对性的解决问题
    - 应用层(Application) 提供网络与用户应用软件之间的接口服务(HTTP)
    - 表示层(Presentation) 提供格式化的表示和转换数据服务，如加密和压缩()
    - 会话层(Session) 提供包括访问验证和会话管理在内的建立和维护应用之间通信的机制()
    - 传输层(Transimission) 提供建立、维护和取消传输连接功能，负责可靠地传输数据(TCP)
    - 网络层(Network) 处理网络间路由，确保数据及时传送(路由器)
    - 数据链路层(DataLink) 负责无错传输数据，确认帧、发错重传等(交换机)
    - 物理层(Physics) 提供机械、电气、功能和过程特性(网卡、网线、双绞线、同轴电缆、中继器)

1.1 分层模型

1.2 封装过程



2. TCP/IP参考模型

- TCP/IP是传输控制协议/网络互联协议的简称
- 早期的TCP/IP模型是一个四层结构，从下往上依次是网络接口层、互联网层、传输层和应用层
- 后来在使用过程中，借鉴OSI七层参考模型，将网络接口层划分为了物理层和数据链路层，形成五层结构

2.1 协议的概念和作用

- 为了让计算机能够通信，计算机需要定义通信规则，这些规则就是协议
- 规则是多种，协议也有多种
- 协议就是数据封装格式+传输

2.2 常用协议

- TCP/IP协议被称为传输控制协议/互联网协议，又称网络通讯协议
- 是由网络层的IP协议和传输层的TCP协议组成，是一个很大的协议集合
- 物理层和数据链路层没有定义任何特定协议，支持所有的标准和专用的协议

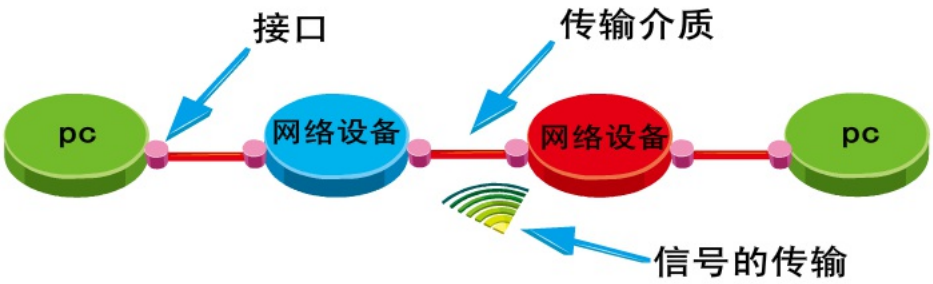
层级 名称 含义 应用层 HTTP 超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议 应用层 FTP 文件传输协议（英文：File Transfer Protocol，缩写：FTP）是用于在网络上进行文件传输的一套标准协议，使用客户/服务器模式 应用层 TFTP (Trivial File Transfer Protocol,简单文件传输协议)是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议 应用层 SMTP 简单邮件传输协议 (Simple Mail Transfer Protocol, SMTP) 是在Internet传输Email的事实标准 应用层 SNMP 简单网络管理协议（SNMP，Simple Network Management Protocol），由一组网络管理的标准组成，包含一个应用层协议（application layer protocol）、数据库模型（database schema）和一组资源对象,该协议能够支持网络管理系统，用以监测连接到网络上的设备是否有任何引起管理上关注的情况。 应用层 DNS 域名系统（英文：Domain Name System，缩写：DNS）是互联网的一项服务。它作为将域名和IP地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。 TCP TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议 传输层 UDP UDP 是User Datagram Protocol的简称， 中文名是用户数据报协议，是OSI（Open System Interconnection，开放式系统互联） 参考模型中一种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 传输层 ICMP ICMP（Internet Control Message Protocol）Internet控制报文协议。它是TCP/IP协议簇的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息 网络层 IGMP Internet 组管理协议称为IGMP协议（Internet Group Management Protocol），是因特网协议家族中的一个组播协议。该协议运行在主机和组播路由器之间 网络层 IP 互联网协议地址（英语：Internet Protocol Address，又译为网际协议地址），缩写为IP地址（英语：IP Address），是分配给用户上网使用的网际协议（英语：Internet Protocol, IP）的设备的数字标签 网络层 ARP 地址解析协议，即ARP（Address Resolution Protocol），是根据IP地址获取物理地址的一个TCP/IP协议 网络层 RARP 反向地址转换协议（RARP：Reverse Address Resolution Protocol） 反向地址转换协议（RARP）允许局域网的物理机器从网关服务器的 ARP 表或者缓存上请求其 IP 地址

3. 网络接口层

- 网络接口层是TCP/IP模型的最底层，负责接收从上一层交来的数据报并将数据报通过底层的物理网络发送出去，比较常见的就是设备的驱动程序，此层没有特定的协议
- 网络接口层又分为物理层和数据链路层

3.1 物理层

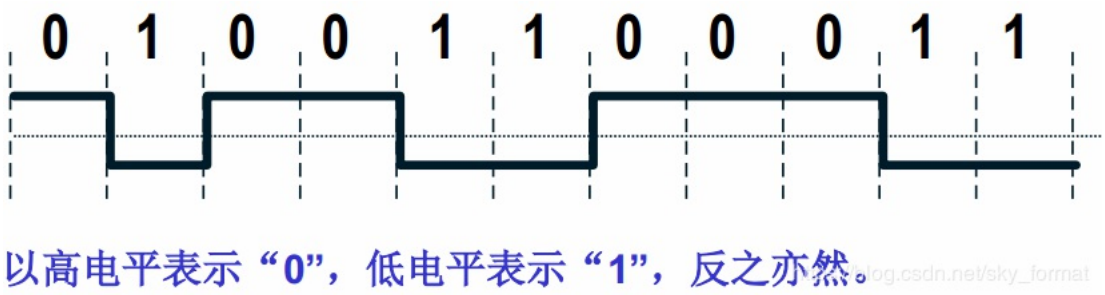
- 计算机在传递数据的时候传递的都是0和1的数字，而物理层关心的是用什么信号来表示0和1，是否可以双向通信，最初的连接如何建立以及完成连接如何终止,物理层是为数据传输提供可靠的环境
- 尽可能的屏蔽掉物理设备和传输媒介，使数据链路层不考虑这些差异，只考虑本层的协议和服务
- 为用户提供在一条物理传输媒体上提供传送和接收比特流的能力
- 需要解决物理连接、维护和释放的问题



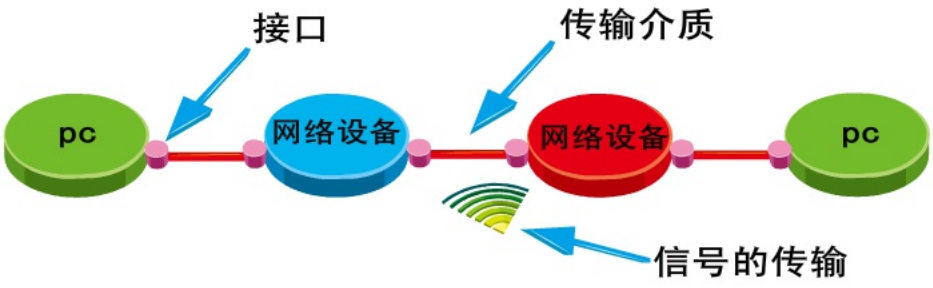
3.1.1 数字信号的编码

- 数字信号的编码：用何种物理信号来表示 0和 1

3.1.1.1 非归零编码



- 优点： 编/译码简单。
- 缺点： 内部不含时钟信号， 收/发端同步困难。
- 用途： 计算机内部， 或低速数据通信。*3.1.1.2 曼彻斯特编码#



- 优点：
  - 内部自含时钟， 收/发端同步容易。
  - 抗干扰能力强。
- 缺点：
  - 编/译码较复杂。
  - 占用更多的信道带宽， 在同样的波特率的情况下， 要比非归零编码多占用一倍信道带宽。
  - 用途： 802.3局域网（以太网）

3.2 数据链路层

- 数据链路层是OSI参考模型中的第二层，介乎于物理层和网络层之间
- 数据链路层在物理层提供的服务的基础上向网络层提供服务，其最基本的服务是将源自网络层来的数据可靠地传输到相邻节点的目标机网络层
- 如何将数据组合成数据块，在数据链路层中称这种数据块为帧 `frame`，帧是数据链路层的传送单位
- 如何控制帧在物理信道上的传输，包括如何处理传输差错，如何调节发送速率以使与接收方相匹配

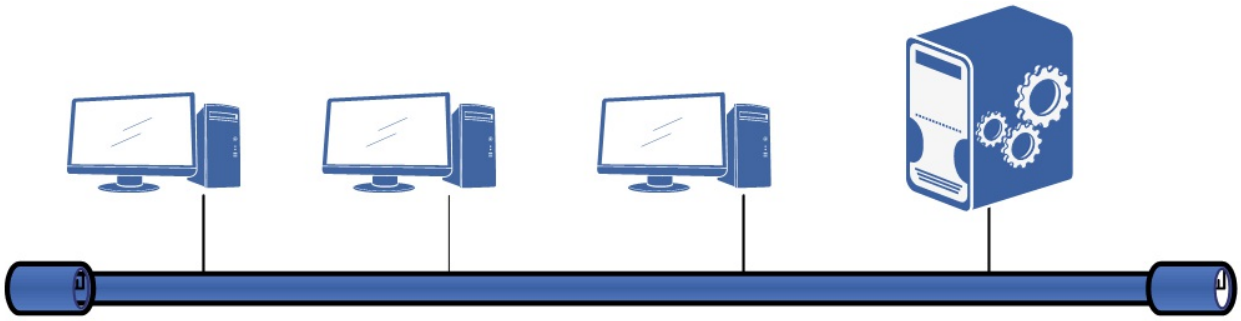
- 以及在两个网络实体之间提供数据链路通路的建立、维持和释放的管理

3.2.1 以太网

- 以太网（Ethernet）是一种计算机局域网技术。IEEE组织的IEEE 802.3标准制定了以太网的技术标准，它规定了包括物理层的连线、电子信号和介质访问层协议的内容
- 以太网的标准拓扑结构为总线型拓扑
- 以太网仍然使用总线型拓扑和CSMA/CD（Carrier Sense Multiple Access/Collision Detection，即载波多重访问/碰撞侦测）的总线技术
- 以太网实现了网络上无线电系统多个节点发送信息的想法，每个节点必须获取电缆或者信道的才能传送信息
- 每一个节点有全球唯一的48位地址也就是制造商分配给网卡的MAC地址，以保证以太网上所有节点能互相鉴别

3.2.2 总线型拓扑

- 总线型拓扑是采用单根传输作为共用的传输介质,将网络中所有的计算机通过相应的硬件接口和电缆直接连接到这根共享的总线上
- 使用总线型拓扑结构需解决的是确保端用户使用媒体发送数据时不能出现冲突。
- 总线型网络采用 载波监听多路访问/ 冲突检测协议（CSMA/CD)作为控制策略



** 3.2.2.1 载波监听多路访问 #**

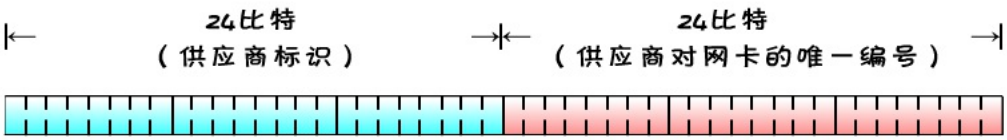
- 全称Carrier Sense Multiple Access (CSMA)，是一种允许多个设备在同一信道发送信号的协议，其中的设备监听其它设备是否忙碌，只有在线路空闲时才发送
- 在此种访问方式下，网络中的所有用户共享传输介质，信息通过广播传送到所有端口，网络中的工作站对接收到的信息进行确认，若是发给自己的便接收否则不理
- 从发送端情况看，当一个工作站有数据要发送时，他首先监听信道并检测网络上是否有其他的工作站正在发送DATA，如果检测到信道忙，工作站将继续WAIT若发现信道空闲，则开始发送数据，信息发送出去后，发送端还要继续对发送出去的信息进行确认，以了解接收端是否已经正确接收到数据，如果收到则发送结束，否则再次发送
- 核心思想
    - 先听后讲 信道空闲则发送，信道忙则等待。
    - 边听边讲 发送信号时不断检测信道是否碰撞。
    - 碰撞即停
    - 退避重传 二进制指数退避重传
    - 多次碰撞，放弃发送,最多16次

** 3.2.2.2 冲突检测 #**

- 冲突检测即发送站点在发送数据时要边发送边监听信道，若监听到信道有干扰信号，则表示产生了冲突，于是就要停止发送数据，计算出退避等待时间，然后使用CSMA方法继续尝试发送
- 计算退避等待时间采用的是 二进制指数退避算法

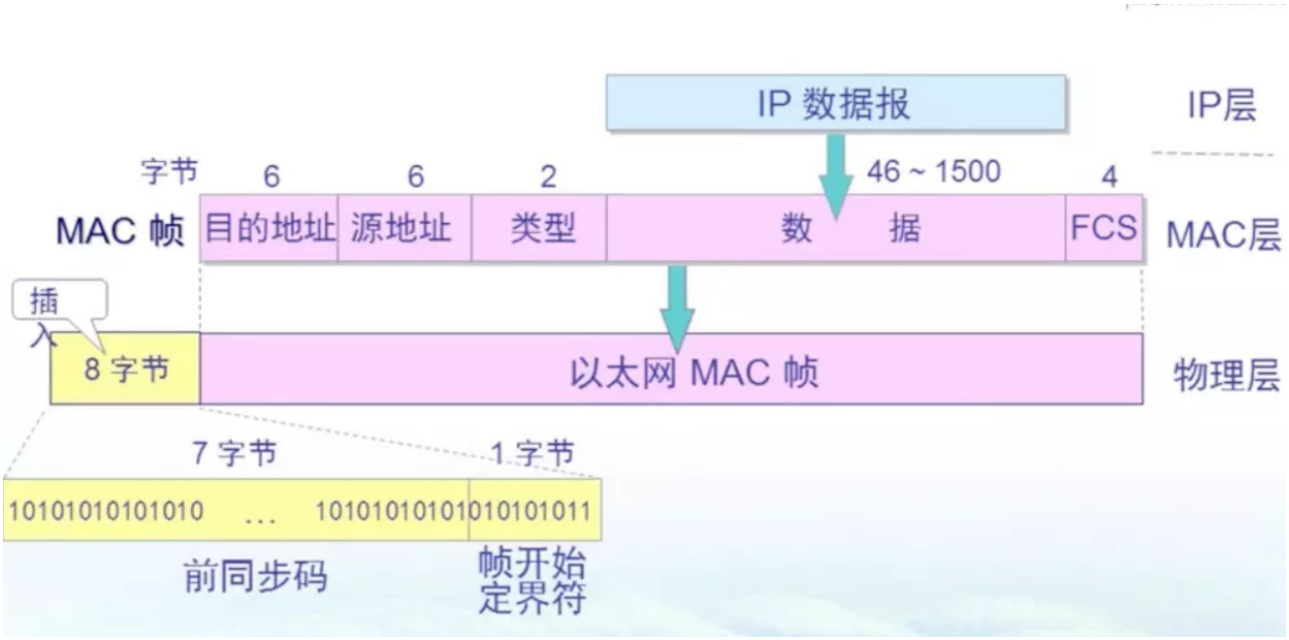
3.2.3 MAC地址

- 在通信过程中是用内置在网卡内的地址来标识计算机身份的
- 每个网卡都有一个全球唯一的地址来标识自己，不会重复
- MAC地址48位的二进制组成，通常分为6段，用16进制表示



3.2.4 以太网帧格式

- 在以太网链路上的数据包称作以太帧。以太帧起始部分由前导码和帧开始符组成
- 后面紧跟着一个以太网报头，以MAC地址说明目的地址和源地址
- 帧的中部是该帧负载的包含其他协议报头的数据包(例如IP协议)
- 以太帧由一个32位冗余校验码结尾。它用于检验数据传输是否出现损坏

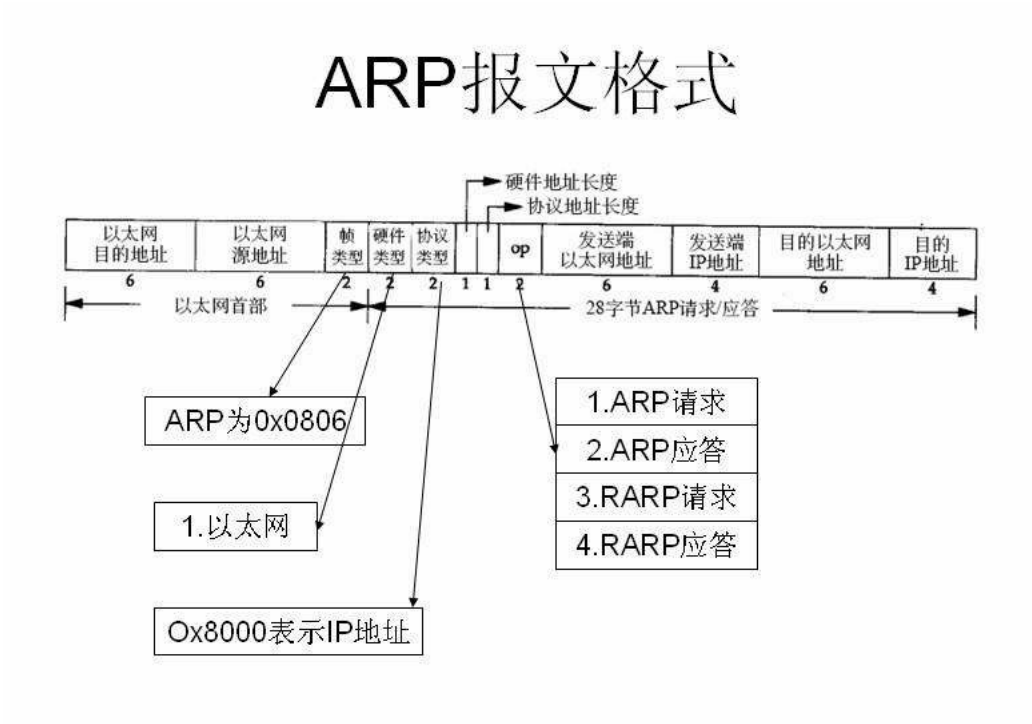


字段 说明 前导符(Preamble) 由1和0交互构成(10101010.....)，共占7个字节，用于使PLS子层电路与收到的帧达成时钟同步 帧起始(Start-of-Frame Delimiter, SFD) 为10101011，共占1个字节，表示一个帧的开始。它和前导符共同使接收方能根据1、0交替来迅速实现比特同步，当检测到连续的两位1时，将后续信息交给MAC子层。通常来说，Pre和SFD这两个字段只用于提醒接收端新帧到达，并不计入MAC帧大小，也不算作是MAC帧头的组成部分 目的MAC(Destination Address, DA)/源MAC(Source Address, SA) 分别用于标识目的MAC地址和源MAC地址，两个字段各占6个字节。它们可以是单播地址也可以是广播地址。当地址的最高位为0时表示单播，最高位为1时为组播，全为1时为广播 长度(Length)/类型(Type) 这是一个二选一的字段，共占2个字节，对于不同的网络协议，它有不同的含义。但是，作为类型使用时，如上表所示，最小值也总是大于1536（十六进制0x600）；所以不会产生冲突。另外，在IEEE 802.3中，数据字段的长度为38~1500个字节 数据(Data) 该字段对于不同的以太网帧包含的内容不一，对于较老的以太网标准，它是网络层来的数据报；而较新的标准，则是一个LLC帧的全部内容 帧校验序列(FCS) 它是一个包含32位CRC校验值的字段，一共占4个字节。由发送端对MAC帧的DA字段到Data字段间（不包含前导符和帧起始）的二进制序列进行计算

3.2.5 ARP协议 #

- 地址解析协议，即ARP（Address Resolution Protocol），是根据IP地址获取物理地址的一个TCP/IP协议
- 主机发送信息时将包含目标IP地址的ARP请求广播到网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源
- 地址解析协议是建立在网络中各个主机互相信任的基础上的，网络上的主机可以自主发送ARP应答消息，其他主机收到应答报文时不会检测该报文的真实性就会将其记入本机ARP缓存
- 由此攻击者就可以向某一主机发送伪ARP应答报文，使其发送的信息无法到达预期的主机或到达错误的主机，这就构成了一个ARP欺骗

** 3.2.5.1 ARP协议报文 #**



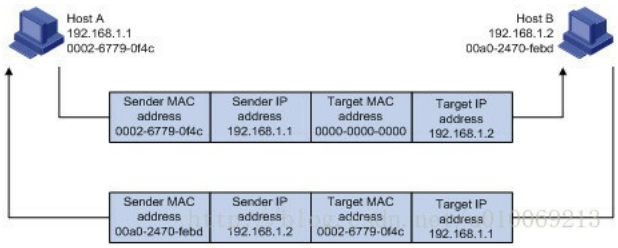
字段 说明 硬件类型 表示硬件地址的类型，值为1表示以太网地址 协议类型 表示要映射的协议地址类型。它的值为0x0800表示IP地址类型 硬件地址长度和协议长度 以字节为单位，对于以太网上的IP地址的ARP请求或应答来说，他们的值分别为6和4 操作类型 用来表示这个报文的类型，ARP请求为1，ARP响应为2，RARP请求为3，RARP响应为4 发送端MAC地址 发送方设备的硬件地址 发送端IP地址 发送方设备的IP地址 目标MAC地址 接收方设备的硬件地址 目标IP地址 接收方设备的IP地址

arp

No.	Time	Source	Destination	Protocol	Length	Info
32	10.418670	AsixElec_b1:ae:65	TendaTec_62:46:d0	ARP	42	Who has 192.168.2.1? Tell 192.168.2.100

```
> Frame 32: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
v Ethernet II, Src: AsixElec_b1:ae:65 (00:0e:c6:b1:ae:65), Dst: TendaTec_62:46:d0 (cc:2d:21:62:46:d0)
  > Destination: TendaTec_62:46:d0 (cc:2d:21:62:46:d0)
  > Source: AsixElec_b1:ae:65 (00:0e:c6:b1:ae:65)
    Type: ARP (0x0806)
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: AsixElec_b1:ae:65 (00:0e:c6:b1:ae:65)
    Sender IP address: 192.168.2.100
    Target MAC address: TendaTec_62:46:d0 (cc:2d:21:62:46:d0)
    Target IP address: 192.168.2.1
```

** 3.2.5.2 ARP地址解析过程 #**

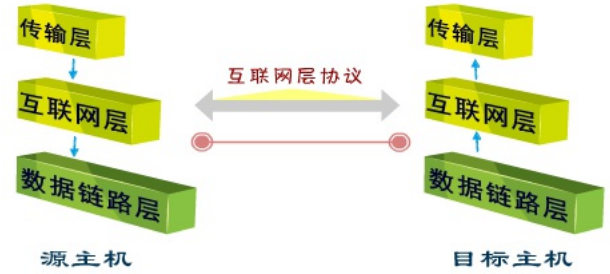


主机A和B在同一个网段，主机A要向主机B发送信息

- 1. 主机A首先查看自己的ARP表，确定其中是否包含有主机B对应的ARP表项。如果找到了对应的MAC地址，则主机A直接利用ARP表中的MAC地址，对IP数据包进行帧封装，并将数据包发送给主机B。
- 1. 如果主机A在ARP表中找不到对应的MAC地址，则将缓存该数据报文，然后以广播方式发送一个ARP请求报文。ARP请求报文中的发送端IP地址和发送端MAC地址为主机A的IP地址和MAC地址，目标IP地址和目标MAC地址为主机B的IP地址和全0的MAC地址。由于ARP请求报文以广播方式发送，该网段上的所有主机都可以接收到该请求，但只有被请求的主机（即主机B）会对该请求进行处理。
- 1. 主机B比较自己的IP地址和ARP请求报文中的目标IP地址，当两者相同时进行如下处理：将ARP请求报文中的发送端（即主机A）的IP地址和MAC地址存入自己的ARP表中。之后以单播方式发送ARP响应报文给主机A，其中包含了自己的MAC地址。
- 1. 主机A收到ARP响应报文后，将主机B的MAC地址加入到自己的ARP表中以用于后续报文的转发，同时将IP数据包进行封装后发送出去。

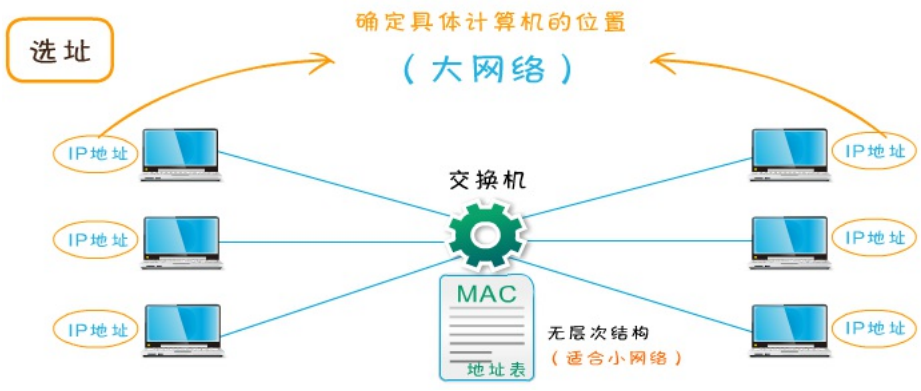
4. 互联网层(网络层)

- 位于传输层和网络接口层之间,用于把数据从源主机经过若干个中间节点传送到目标主机,并向传输层提供最基础的数据传输服务,它要提供路由和选址的工作



4.1 选址

交换机是靠MAC来寻址的，而因为MAC地址是无层次的,所以要靠IP地址来确认计算机的位置,这就是选址



4.2 路由

在能够选择的多条道路之间选择一条最短的路径就是路由的工作

4.3 IP

在网络中，每台计算机都有一个唯一的地址，方便别人找到它，这个地址称为IP地址。

4.3.1 IP头部

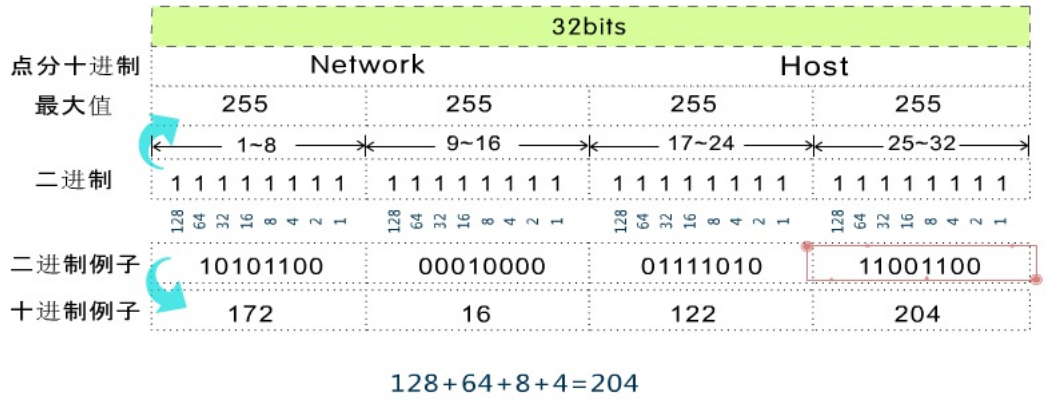
字段 说明 版本 Version）字段：占4比特。用来表明IP协议实现的版本号，当前一般为IPv4，即0100 首部长部 报头长度（Internet Header Length，IHL）字段：占4比特。是头部占32比特的数字，包括可选项。普通IP数据报（没有任何选项），该字段的值是5，即160比特=20字节。此字段最大值为60字节 优先级与服务类型 占8比特。其中前3比特为优先权子字段（Precedence，现已被忽略）。第8比特保留未用。第4至第7比特分别代表延迟、吞吐量、可靠性和花费。当它们取值为1时分别代表要求最小时延、最大吞吐量、最高可靠性和最小费用。这4比特的服务类型中只能置其中1比特为1。可以全为0，若全为0则表示一般服务。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。例如：TELNET协议可能要求有最小的延迟，FTP协议（数据）可能要求有最大吞吐量，SNMP协议可能要求有最高可靠性，NNTP（Network News Transfer Protocol，网络新闻传输协议）可能要求最小费用，而ICMP协议可能无特殊要求（4比特全为0 总长度 占16比特。指明整个数据报的长度（以字节为单位）。最大长度为65535字节 标识符 占16比特。用来唯一地标识主机发送的每一份数据报。通常每发一份报文，它的值会加1 标志 分为3个字段，依次为保留位、不分片位和更多片位 标志 保留位：一般被置为0 标志 不分片位：表示该数据报是否被分片，如果被置为1，则不能对数据报进行分片，如果要对其进行分片处理，就应将其置为0 标志 更多片位：除了最后一个分片，其他每个组成数据报的片都要将该位置设置为1. 段偏移量 占13比特。如果一份数据报要求分段的话，此字段指明该段偏移距原始数据报开始的位置 TTL(Time to Live生存时间) 该字段用于表示IP数据包的生命周期，可以防止一个数据包在网络中无限循环地发下去。TTL的意思是一个数据包在被丢弃之前在网络中的最大周转时间。该数据包经过的每一个路由器都会检查该字段中的值，当TTL的值为0时此数据包会被丢弃。TTL对应于一个数据包通过路由器的数目，一个数据包每经过一个路由器，TTL将减去1 协议号 占8比特。指明IP层所封装的上层协议类型，如ICMP（1）、IGMP（2）、TCP（6）、UDP（17）等 首部校验和 检验和是16位的错误检测字段。目的主机和网络中的每个网关都要重新计算报头的校验和，一样表示没有改动过,计算方法是：对头部中每个16比特进行二进制反码求和 源IP地址 该字段用于表示数据包的源地址，指的是发送该数据包的设备的网络地址 目标IP地址 该字段用于表示数据包的目标的地址，指的是接收节点的网络地址

```
tcp
No.  Time      Source          Destination     Protocol  Length  Info
  31 10.049692 192.168.2.100   199.59.148.14   TCP         66 [TCP Retransmission] 55193 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

> Frame 31: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
v Ethernet II, Src: AsixElec_b1:ae:65 (00:0e:c6:b1:ae:65), Dst: TendaTec_62:46:d0 (cc:2d:21:62:46:d0)
    > Destination: TendaTec_62:46:d0 (cc:2d:21:62:46:d0)
    > Source: AsixElec_b1:ae:65 (00:0e:c6:b1:ae:65)
      Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.2.100, Dst: 199.59.148.14
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    v Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
      Total Length: 52
      Identification: 0x1ca1 (7329)
    v Flags: 0x4000, Don't fragment
        0... .... .... .... = Reserved bit: Not set
        .1.. .... .... .... = Don't fragment: Set
        ..0. .... .... .... = More fragments: Not set
        ...0 0000 0000 0000 = Fragment offset: 0
      Time to live: 64
      Protocol: TCP (6)
      Header checksum: 0x0000 [validation disabled]
      [Header checksum status: Unverified]
      Source: 192.168.2.100
      Destination: 199.59.148.14
```

4.3.2 IP地址格式

- IP地址是一个网络编码，用来确定网络中的一个节点。
- IP地址是由32位二进制(32bit)组成

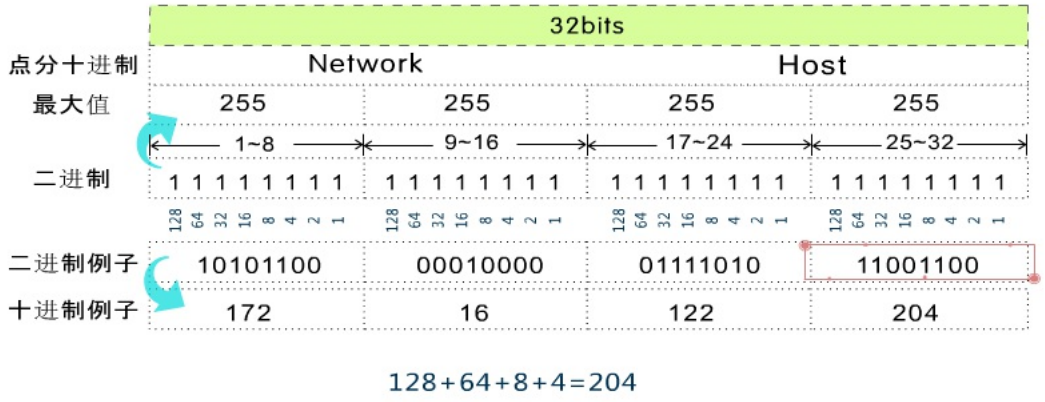


128+64+8+4=204

4.3.3 IP地址组成

- 网络部分(NETWORK)
- 主机部分(HOST)

4.3.4 IP地址表示



128+64+8+4=204

4.3.5 IP地址的分类

- IP地址的网络部分是由Internet地址分配机构来统一分配的，这样可以保证IP的唯一性。
- ip地址中全为1的ip即255.255.255.255，它称为限制广播地址，如果将其作为数据包的目标地址可以理解为发送到所有网络的所有主机
- ip地址中全为0的ip即0.0.0.0，它表示启动时的ip地址，其含义就是尚未未分配时的ip地址
- 127是用来进行本机测试的，除了127.255.255.255外，其它的127开头的地址都代表本机

4.3.6 公有地址和私有地址

分类 范围 A类私有IP 10.0.0.0 ~ 10.255.255.255 B类私有IP 172.16.0.0 ~ 172.31.255.255 C类私有IP 192.168.0.0 ~ 192.168.255.255

其他范围的IP均为公有IP地址

4.3.7 子网掩码

- 子网掩码(subnet mask)又叫子网络遮罩，它是一种用来指明一个IP地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机位的掩码。
- 子网掩码不能单独存在，它必须结合IP地址一起使用。
- 子网掩码只有一个作用，就是将某个IP地址划分成网络地址和主机地址两部分。
- 子网掩码也是32个二进制位
- 对应IP的网络部分用1表示
- 对应IP地址的主机部分用0表示
- IP地址和子网掩码做逻辑与运算得到网络地址
    - 0和任何数相与都是0
    - 1和任何数相与都等于任何数本身
- A B C 三类地址都有自己默认的子网掩码
    - A类 255.0.0.0
    - B类 255.255.0.0
    - C类 255.255.255.0

```
let ip1 = '192.168.0.1';
let ip2 = '192.168.0.4';
let mask = '255.255.255.0';
let ip1s = ip1.split('.').map(item=>(+item).toString(2).padStart(8,'0')).join('');
let ip2s = ip2.split('.').map(item=>(+item).toString(2).padStart(8,'0')).join('');;
let masks = mask.split('.').map(item=>(+item).toString(2).padStart(8,'0')).join('');

console.log(ip1s,parseInt(ip1s,2),parseInt(masks,2),parseInt(ip1s,2)&parseInt(masks,2));
console.log(ip2s,parseInt(ip2s,2),parseInt(masks,2),parseInt(ip2s,2)&parseInt(masks,2));
console.log((parseInt(ip1s,2)&parseInt(masks,2)) === (parseInt(ip2s,2)&parseInt(masks,2)));
```

5. 传输层

- 位于应用层和网络接口层之间
- 是面向连接的、可靠的的进程到进程通信的协议
- TCP提供全双工服务，即数据可在同一时间双向传播

- TCP将若干个字节构成一个分组，此分组称为报文段(Segment)
- 对可靠性要求高的上层协议，实现可靠性的保证,如果数据丢失、损坏的情况下如何保证可靠性,网络层只管传递数据，成功与否并不关心

5.1 传输层的功能

- 提供了一种端到端的连接

5.2 协议分类

- TCP(Transimision Control Protocal)
    - 传输控制协议
    - 可靠的、面向连接的协议
    - 传输效率低
- UDP(User Datagram Protocal)
    - 用户数据报协议
    - 不可靠的、无连接的服务
    - 传输效率高

5.3 TCP协议

- 将数据进行分段打包传输
- 对每个数据包编号控制顺序
- 运输中丢失、重发和丢弃处理
- 流量控制避免拥塞

5.3.1 TCP数据包封装

** 5.3.1.1 格式 #**

- 源端口号和目标端口号，计算机通过端口号识别访问哪个服务,比如http服务或ftp服务，发送方端口号是进行随机端口，目标端口号决定了接收方哪个程序来接收

** 5.3.1.2 32位序列号 #**

- 32位序列号 TCP用序列号对数据包进行标记，以便在到达目的地后重新重装，假设当前的序列号为 s，发送数据长度为 l，则下次发送数据时的序列号为 s + l。在建立连接时通常由计算机生成一个随机数作为序列号的初始值

** 5.3.1.3 确认应答号 #**

- 确认应答号 它等于下一次应该接收到的数据的序列号。假设发送端的序列号为 s，发送数据的长度为 l，那么接收端返回的确认应答号也是 s + l。发送端接收到这个确认应答后，可以认为这个位置以前所有的数据都已被正常接收。

** 5.3.1.4 首部长度 #**

- 首部长度：TCP 首部的长度，单位为 4 字节。如果没有可选字段，那么这里的值就是 5。表示 TCP 首部的长度为 20 字节。

** 5.3.1.5 控制位 #**

- 控制位 TCP的连接、传输和断开都受这六个控制位的指挥
    - PSH(push急迫位) 缓存区将满，立刻传输速度
    - RST(reset重置位) 连接断了重新连接
    - URG(urgent紧急位) 紧急信号
- 紧急指针：尽在 URG(urgent紧急) 控制位为 1 时有效。表示紧急数据的末尾在 TCP 数据部分中的位置。通常在暂时中断通信时使用（比如输入 Ctrl + C）。

5.3.1.5.1 SYN

- SYN(synchronous建立联机) 同步序号位 TCP建立连接时要将这个值设为1

5.3.1.5.2 ACK

- ACK(acknowledgement 确认)为1表示确认号

5.3.1.5.3 FIN

- FIN发送端完成位，提出断开连接的一方把FIN置为1表示要断开连接

** 5.3.1.6 窗口值 #**

- 窗口值 说明本地可接收数据段的数目，这个值的大小是可变的。当网络通畅时将这个窗口值变大加快传输速度，当网络不稳定时减少这个值可以保证网络数据的可靠传输。它是来在TCP传输中进行流量控制的
- 窗口大小：用于表示从应答号开始能够接受多少个 8 位字节。如果窗口大小为 0，可以发送窗口探测。

** 5.3.1.7 差错控制 #**

- 校验和用来做差错控制，TCP校验和的计算包括TCP首部、数据和其它填充字节。在发送TCP数据段时，由发送端计算校验和，当到达目的地时又进行一次检验和计算。如果两次校验 和一致说明数据是正确的，否则 将认为数据被破坏，接收端将丢弃该数据

5.3.2 握手和断开

- TCP是面向连接的协议，它在源点和终点之间建立虚拟连接，而不是物理连接
- 在数据通信之前，发送端与接收端要先建立连接，等数据发送结束后，双方再断开连接
- TCP连接的每一方都是由一个IP地址和一个端口组成

** 5.3.2.1 tcp服务器 #**

5.3.2.1.1 tcp_server.js

```
var net = require("net");
var server = new net.Server();
server.on("connection", function (socket) {
    console.log("connected");
    socket.on('data',function(data){
        console.log(data.toString());
        socket.write("server:"+data);
    })
    socket.on('end',function(data){
        console.log('end');
    })
    socket.on('error',function(error){
        console.log(error);
    })
});
server.listen(8000);
server.on("listening", function () {
    console.log("Created server on http://127.0.0.1:8000/");

})
server.on("close", function () {
    console.log("server closed!");
})
server.on("error", function (err) {
    console.log(err);
})
```

5.3.2.1.2 tcp_client.js

```
var net = require("net");
var socket = net.Socket();
socket.connect(8000, '127.0.0.1', function () {
    console.log("connect the server");
    socket.write("hello");
})
socket.on("data", function (data) {
    console.log(data.toString());
    socket.destroy();
})
socket.on("error", function (err) {
    console.log(err);
})
socket.on("end", function () {
    console.log("data end");
})
```

** 5.3.2.1 三次握手 #**

- 第一次握手主机A通过一个标识为SYN标识位的数据段发送给主机B请求连接，通过该数据段告诉主机B希望建立连接，需要B应答，并告诉主机B传输的起始序列号
- 第二次握手是主机B用一个确认应答ACK和同步序列号SYNC标志位的数据段来响应主机A，一是发送ACK告诉主机A收到了数据段，二是通知主机A从哪个序列号做标记。
- 第三次握手是主机A确认收到了主机B的数据段并可以开始传输实际数据。

** 5.3.2.2 收发数据 #**

** 5.3.2.3 四次断开 #**

- 主机A发送FIN控制位发出断开连接的请求
- 主机B进行响应，确认收到断开连接请求
- 主机B提出反方向的关闭要求
- 主机A确认收到的主机B的关闭连接请求

5.3.3 滑动窗口

- 滑动窗口（Sliding window）是一种流量控制技术
- 早期的网络通信中，通信双方不会考虑网络的拥挤情况直接发送数据。由于大家不知道网络拥塞状况，同时发送数据，导致中间节点阻塞掉包，谁也发不了数据，所以就有了滑动窗口机制来解决此问题
- TCP中采用滑动窗口来进行传输控制，滑动窗口的大小意味着接收方还有多大的缓冲区可以用于接收数据。发送方可以通过滑动窗口的大小来确定应该发送多少字节的数据
- 当滑动窗口为0时，发送方一般不能再发送数据报，但有两种情况除外，一种情况是可以发送紧急数据，例如，允许用户终止在远端机上的运行进程。另一种情况是发送方可以发送一个1字节的数据报来通知接收方重新声明它希望接收的下一字节及发送方的滑动窗口大小

** 5.3.3.1 窗口机制 #**

- 滑动窗口协议的基本原理就是在任意时刻，发送方都维持了一个连续的允许发送的帧的序号，称为发送窗口；同时，接收方也维持了一个连续的允许接收的帧的序号，称为接收窗口
- 发送窗口和接收窗口的序号的上下界不一定要一样，甚至大小也可以不同
- 不同的滑动窗口协议窗口大小一般不同
- 发送方窗口内的序列号代表了那些已经被发送，但是还没有被确认的帧，或者是那些可以被发送的帧

** 5.3.3.2 拥塞控制 #**

- TCP拥塞控制是传输控制协议（英语：Transmission Control Protocol，缩写TCP）避免网络拥塞的算法，是互联网上主要的一个拥塞控制措施
- TCP使用多种拥塞控制策略来避免雪崩式拥塞。TCP会为每条连接维护一个"拥塞窗口"来限制可能在端对端间传输的未确认分组总数量
- 这类似TCP流量控制机制中使用的滑动窗口，是由发送方控制的
- TCP在一个连接初始化或超时后使用一种"慢启动"机制来增加拥塞窗口的大小。它的起始值一般为最大分段大小（Maximum segment size，MSS）的两倍，虽然名为"慢启动"，初始值也相当低，但其增长极快：当每个分段得到确认时，拥塞窗口会增加一个MSS，使得在每次往返时间（round-trip time，RTT）内拥塞窗口能高效地双倍增长
- 在流量控制中，接收方通过TCP的"窗口"值（Window Size）来告知发送方，由发送方通过对拥塞窗口和接收窗口的大小比较，来确定任何时刻内需要传输的数据量
- 和式增加，积式减少（additive-increase/multiplicative-decrease，AIMD，这里简称"线增积减"）是一种反馈控制算法，其包含了对拥塞窗口线性增加，和当发生拥塞时对窗口积式减少。多个使用AIMD控制的TCP流最终会收敛到对线路的等量竞争使用。
- 未确认的数据包刚好等于带宽等于延迟
- 当发现丢包的时候立刻减半

5.4 UDP

- UDP是一个无连接、不保证可靠性的传输层协议，也就是说发送端不关心发送的数据是否到达目标主机、数据是否出错等，收到数据的主机也不会告诉 发送方是否收到了数据，它的可靠性由上层协议来保障
- 首部结构简单，在数据传输时能实现最小的开销，如果进程想发送很短的报文而对可靠性要求不高可以使用

5.4.1 UDP的封装格式

** 5.4.1.1 数据包 #**

** 5.4.1.2 数据长度 #**

** 5.4.1.3 差错控制 #**

5.4.2 UDP的应用

- QQ
- 视频软件
- TFTP 简单文件传输协议(短信)

5.4.3 UDP服务器

** 5.4.3.1 点对点 #**

5.4.3.1.1 udp_server.js

```
var dgram = require('dgram');
var socket = dgram.createSocket('udp4');
socket.on('message',function(msg,rinfo){
  console.log(msg.toString());
  console.log(rinfo);
   socket.send(msg,0,msg.length,rinfo.port,rinfo.address);
});
socket.bind(41234,'localhost');
```

5.4.3.1.2 udp_client.js

```
var dgram = require('dgram');
var socket = dgram.createSocket('udp4');
socket.on('message',function(msg,rinfo){
    console.log(msg.toString());
    console.log(rinfo);
});
socket.send(new Buffer('helloworld'),0,5,41234,'localhost',function(err,bytes){
    console.log('发送了个%d字节',bytes);
});
socket.on('error',function(err){
    console.error(err);
});
```

** 5.4.3.2 广播 #**

- 创建一个UDP服务器并通过该服务器进行数据的广播**5.4.3.2.1 udp_server.js #**

```
let dgram = require('dgram');
let server = dgram.createSocket('udp4');
server.on('message',function(msg){
let buf = new Bufffer('已经接收客户端发送的数据'+msg);
server.setBroadcast(true);
server.send(buf,0,buf.length,41235,"192.168.1.255");
});
server.bind(41234,'192.168.1.100');
```

5.4.3.2.2 udp_client.js #

```
let dgram = require('dgram');
let client = dgram.createSocket('udp4');
client.bind(41235,'192.168.1.102);
let buf = new Buffer('hello');
client.send(buf,0,buf.length,41234,'192.168.1.100');
client.on('message',function(msg,rinfo){
console.log('received : ',msg);
});
```

5.4.3.3 组播#

- 所谓的组播，就是将网络中同一业务类型进行逻辑上的分组，从某个socket端口上发送的数据只能被该组中的其他主机所接收，不被组外的任何主机接收。
- 实现组播时，并不直接把数据发送给目标地址，而是将数据发送到组播主机，操作系统将把该数据组播给组内的其他所有成员。
- 在网络中，使用D类地址作为组播地址。范围是指 224.0.0.0 ~ 239.255.255.255,分为三类
    - 局部组播地址: 224.0.0.0 ~ 224.0.0.255 为路由协议和其他用途保留
    - 预留组播地址: 224.0.1.0 ~ 238.255.255.255 可用于全球范围或网络协议
    - 管理权限组播地址： 239.0.0.0 ～ 239.255.255.255 组织内部使用，不可用于Internet **5.4.3.3.1 udp_server.js #**

```
let dgram = require('dgram');
let server = dgram.createSocket('udp4');
server.on('listening',function(){
server.MulticastTTL(128);
server.setMulticastLoopback(true);
server.addMembership('230.185.192.108');
});
setInterval(broadcast,1000);
function broadcast(){
let buffer = Buffer.from(new Date().toLocaleString());
server.send(buffer,0,buffer.length,8080,"230.185.192.108");
}
```

5.4.3.3.2 udp_client.js #

```
let dgram = require('dgram');
let client = dgram.createSocket('udp4');
client.on('listening',function(){
client.addMembership('230.185.192.108');
});
client.on('message',function(message,remote){
console.log(message.toString());
});
client.bind(8080,'192.168.1.103');
```

5.4.3 DNS服务器

** 3.4.3.1 域名 #**

- 域名空间结构
- 根域
- 顶级域
    - 组织域
    - 国家/地区域名
- 二级域名

** 3.4.3.2 DNS服务器 #**

DNS是Domain Name Service的缩写，DNS服务器进行域名和与之对应的IP地址转换的服务器

- IP地址不易记忆
- 早期使用Hosts文件解析域名
    - 主要名称重复
    - 主机维护困难
- DNS(Domain Name System 域名系统)
    - 分布式
    - 层次性

** 3.4.3.3 查找过程 #**

- 客户端向本地域名服务器发出请求，我要访问www.163.com，请告诉我它的IP地址 (http://www.163.com，请告诉我它的IP地址)
- 本地DNS服务器向DNS根服务器发出请求，根DNS服务器会告诉本地服务器(.com)的服务器地址
- 本地DNS服务器会向(.com域)发请求，会得到(163.com)的服务器地址
- 本地DNS服务器会向(163.com)发请求,会得到(www.163.com)的IP地址1.1.1.1 (http://www.163.com)的IP地址1.1.1.1)
- 本地DNS服务器向客户端回复域名(www.163.com)对应的IP地址是1.1.1.1 (http://www.163.com)对应的IP地址是1.1.1.1)

5.4.4 DHCP服务器

- 保证任何IP地址在同一时刻只能由一台DHCP客户机所使用。
- DHCP应当可以给用户分配永久固定的IP地址。
- DHCP应当可以同用其他方法获得IP地址的主机共存（如手工配置IP地址的主机）
- DHCP服务器应当向现有的BOOTP客户端提供服务。

** 5.4.4.1 工作流程 #**

- 主机发送 DHCPDISCOVER广播包在网络上寻找DHCP服务器；
- DHCP服务器向主机发送 DHCPOFFER单播数据包，包含IP地址、MAC地址、域名信息以及地址租期；
- 主机发送 DHCPREQUEST广播包，正式向服务器请求分配已提供的IP地址；
- DHCP服务器向主机发送DHCPACK单播包，确认主机的请求

** 5.4.4.2 抓包 #**

6. 应用层

6.1 协议

6.2 应用层常见协议

- HTTP 超文件传输协议
- FTP 文件传输协议
- SMTP(发送邮件)和POP3(接收邮件)

6.3 案例

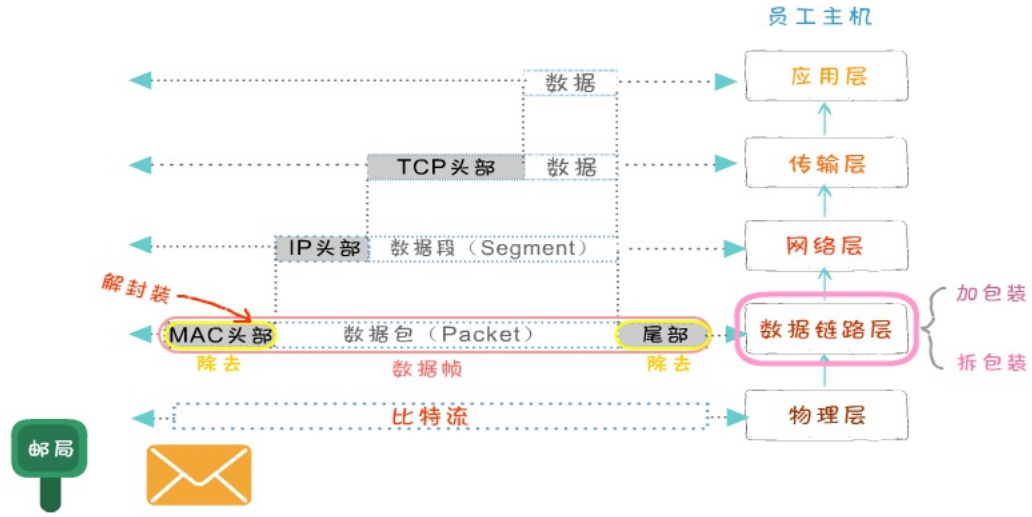
数据->传输层(包)->网络层(段Segment)->数据链路层(帧)

6.3.1 发送方是从高层到低层封装数据

- 在应用层要把各式各样的数据如字母、数字、汉字、图片等转换成二进制
- 在TCP传输层中，上层的数据被分割成小的数据段，并为每个分段后的数据封装TCP报文头部
- 在TCP头部有一个关键的字段信息端口号，它用于标识上层的协议或应用程序，确保上层数据的正常通信
- 计算机可以多进程并发运行，例如在发邮件的同时也可以通过浏览器浏览网页，这两种应用通过端口号进行区分
- 在网络层，上层数据被封装上亲的报文头部(IP头部)，上层的数据是包括TCP头部的。IP地址包括的最关键字段信息就是IP地址，用于标识网络的逻辑地址。
- 数据链路径层，上层数据成一个MAC头部，内部有最关键的是MAC地址。MAC地址就是固化在硬件设备内部的全球唯一的物理地址。
- 在物理层，无论在之前哪一层封装的报文头和还是上层数据都是由二进制组成的，物理将这些二进制数字比特流转换成电信号在网络中传输

6.3.2 接收方是从低层到高层解封装

- 数据封装完毕传输到接收方后，将数据要进行解封装
- 在物理层，先把电信号转成二进制数据，并将数据传送至数据链路层
- 在数据链路层，把MAC头部拆掉，并将剩余的数据传送至上一层
- 在网络层，数据的IP头部被拆掉，并将剩余的数据送至上一层
- 在传输层，把TCP头部拆掉，将真实的数据传送至应用层



6.3.3 真实网络环境

- 发送方和接收方中间可能会有多个硬件中转设备
- 中间可能会增加交换机和路由器
- 数据在传输过程中不断地进行封装和解封装的过程，每层设备只能处理哪一层的数据
    - 交换机属于数据链路层
    - 路由器属于网络层

7. 附录

7.1 不同层中的称谓：

- 数据帧（Frame）：是一种信息单位，它的起始点和目的点都是 **数据链路层**。
- 数据包（Packet）：也是一种信息单位，它的起始和目的地是 **网络层**。
- 段（Segment）：通常是指起始点和目的地都是 **传输层** 的信息单元。
- 消息（message）：是指起始点和目的地都在网络层以上（经常在 **应用层**）的信息单元。

7.2 IP头服务类型

- IP首部中的服务类型（TOS）
- TOS包括共8位，包括3 bit的优先权字段（取值可以从000-111所有值），4 bit的TOS子字段和1 bit未用位但必须置0。
- 3bit的8个优先级的定义如下：
    - 111--Network Control(网络控制）一般保留给网络控制数据使用，如路由。
    - 110--Internetwork Control(网间控制)
    - 101--Critic(关键)语音数据使用。
    - 100--Flash Override(疾速)视频会议和视频流使用。
    - 011--Flash(闪速)语音控制数据使用。
    - 010--Immediate(快速)数据业务使用
    - 001--Priority(优先)数据业务使用
    - 000--Routine(普通)默认标记值。
- 4 bit的TOS分别代表：最小时延、最大吞吐量、最高可靠性和最小费用。4 bit中只能置其中1 bit。如果所有4 bit均为0，那么就意味着是一般服务。
- Telnet、Rlogin这两个交互应用要求最小的传输时延，FTP文件传输要求最大吞吐量，最高可靠性是指网络管理（SNMP）和路由选择协议。用户网络新闻要求最小费用