计网期末复习

第四章 网络层

4.1 概念和概述

网络层的功能：

- **转发**：将分组从一个输入链路接口转移到适当输出链路结构的**路由器本地动作**
- **路由选择**：决定分组从源到目的地所采取的端到端路径，是**网络范围的过程**
- **连接建立**：在给定源到目的地连接中的网络层数据分组能够开始流动之前建立起连接状态

转发表（forwarding table）：每台路由器都有一张转发表，是由分组的首部字段值到输出链路的映射

网络服务模型：定义分组在发送与接收端系统之间的端到端运输特性

发送主机的网络层能提供的服务：

- **确保交付**
- **具有时延上界的确保交付**
- **有序分组交付**（以发送顺序到达目的地）
- **确保最小带宽**（只要发送主机以低于特定比特率的速率传输比特，则分组不会丢失，且每个分组会在预定的主机到主机时延内到达）
- **确保最大时延抖动**（确保位于发送方的两个相继分组之间的时间量等于在目的地接受到他们之间的时间量）
- **安全性服务**（使用仅由源和目的主机所知晓的一个秘密会话秘钥）

因特网的网络层只提供了单一的服务：

- **尽力而为服务best-effort service**

ATM网络体系结构提供的服务：

- 恒定比特率ATM网络服务（CBR）
- 可用比特率ATM网络服务（ABR）

4.2 虚电路和数据报网络

虚电路网络

在其他网络体系结构中（ATM/帧中继）使用虚电路网络，他们在网络层使用连接，这些网络层连接被称为虚电路。

$$\text{虚电路} = \begin{cases} \text{源和目的主机之间的路径（即一系列链路和路由器）} \\ VC\text{号（沿着该路径的每段链路的一个号码）} \\ \text{沿着该路径的每台路由器中的转发表的表项} \end{cases}$$

注：VC号是每段**链路**的号，有n个路由器就有n+1段链路。任意时刻一个分组只携带一个VC号，就是它所在的链路的VC号，每经过一个路由器，根据当前VC号和入接口号在表项中查询，得到新的VC号和出接口号，将VC号更新。转发表以微秒的时间尺度更新。

为什么不用相同的VC号：①减少VC字段长度②简化了虚电路的建立

连接状态信息：虚电路网络中，每个路由器需要为进行中的连接维持连接状态信息，即创建新连接时在转发表中新增一个新的连接项，释放一个连接时删除一个连接项。

虚电路的三个阶段

- 虚电路建立【信令报文】
  - 发送端运输层告诉网络层接收方地址
  - 网络层决定发送方与接收方之间的路径、为沿着该路径的每条链路**决定一个VC号**
  - 网络层在沿着路径的每台路由器的转发表中**增加一个表项**
- 数据传送
  - 数据沿着虚电路流动
- 虚电路拆除【信令报文】
  - 发送方/接收方通知网络层希望终止虚电路是进入这个阶段
  - 网络层通知另一侧端系统结束呼叫，并更新每台分组路由器中的转发表

虚电路建立与运输层连接建立的区别

运输层连接建立仅涉及两个端系统，两个端系统独自决定运输层连接参数，路由器不知情连接；虚电路沿着两个端系统之间路径上的路由器都要参与虚电路的建立，且每台路由器都完全知道经过它的所有虚电路。

数据报网络

发送端为分组加上目的端系统地址，每个路由器利用目的地址进行转发。

转发表：**目的地址前缀（prefix）--> 链路接口**的映射，1-5分钟左右更新一次转发表

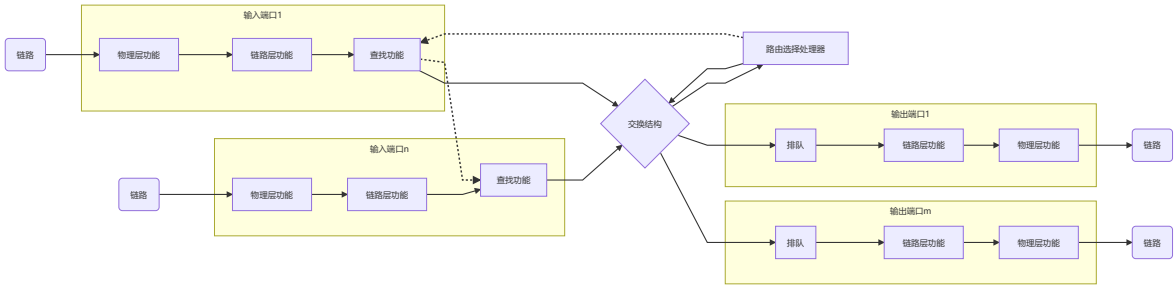
最长前缀匹配规则：一个目的地址可以匹配上转发表中的多条前缀时，选择最长的匹配项。

网络层和运输层的无连接服务和有连接服务的区别

- 网络层中：是网络层向运输层提供的主机到主机的服务 运输层中：是运输层向应用层提供的进程到进程的服务
- 网络层不同时提供**连接服务的虚电路网络**和**无连接服务的数据报网络**，不同计算机的网络不同。
- 运输层面向连接的服务只在端系统中提供，网络层提供的连接服务除了在端系统中，也在位于网络核心的路由器中实现

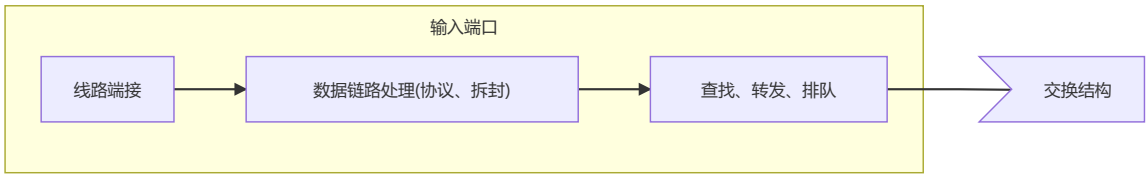
4.3路由器工作原理

$$路由器 = \begin{cases} 输入端口 \\ 交换结构 \\ 输出端口 \\ 路由选择处理器 \end{cases}$$



- **输入端口**（不止有一个输入端口）
  - 将一条**输入链路**与**路由器**相连接的物理层功能
  - 与位于**入链路远端的数据链路层交互**的数据链路层功能
  - 在输入端口完成**查找功能**，通过查询转发表决定路由器输出端口
- **交换结构**
  - 将输入端口与输出端口相连接，是路由器中的网络
- **输出端口**（不止有一个输出端口）
  - 存储从交换结构接受的分组，执行必要的链路层和物理层功能（同输入端口）
- **路由选择处理器**
  - 执行路由选择协议、维护路由选择表和连接的链路状态信息、计算转发表

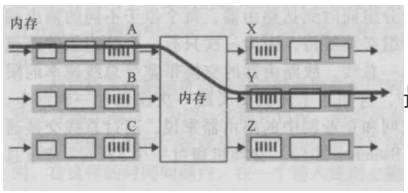
输入端口



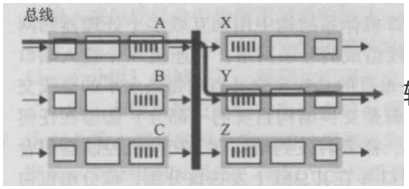
- 转发表的**影子副本**从路由选择处理器经由独立总线（总图中的虚线）复制到线路卡
  - 有影子副本可以避免集中式处理的瓶颈，每次查找不需要调用中央路由选择处理器
  - 用于查找的物理硬件：嵌入式片上DRAM，更快的SRAM作为缓存；三态内容可寻址存储器TCAM（可在常数时间内返回表项内容）
- 排队：查找结束后在进入交换结构时，可能来自另外一个链路的分组正在使用交换结构，这是需要排队
- 输入端口需要做的所有事情：
  - 查找（最重要）
  - 物理层和链路层处理
  - 检查分组的版本号、检验和以及寿命字段（并重写检验和和寿命字段）
  - 更新用于网络管理的计数器（IP数据报的数目）

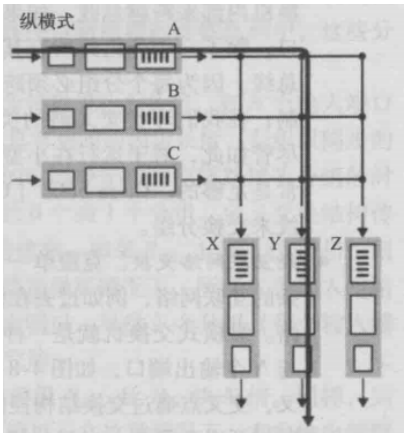
交换结构

有三种交换方式

- **经内存交换**
  - 

最简单、最早的路由器是传统的计算机

    - 输入与输出端口的功能就像在传统操作系统中的I/O设备一样
    - 过程：分组到达后，输入端口向路由选择处理器发送**中断信号**；然后该分组从输入端口处被**复制到处理器内存中**，路有选择处理器从其首部中**提取目的地址**，找到适当输出端口，将该分组**复制到输出端口的缓存中**
    - 若每秒可写进或可读出B个分组，总的吞吐量<B/2
    - 不能同时转发两个分组，因为共享系统总线一次只能执行一次内存读和写
- **经总线交换**
    - 

输入端口晶一根共享总线将分组直接传送到输出端口，不需要路由选择处理器的干预
    - 过程：输入端口为分组预先计划一个交换机**内部标签（首部）**，是分组在总线上传输到输出端口；该分组能由所有输出端口收到，但只有与该标签**匹配**的端口才能保存该分组，然后**标签被去除**
    - 同时多个分组在多个输入端口到达，除了一个分组，其他分组必须等待，因为一次只有一个分组能跨越总线。
- **经互联网络交换**
    - 

每个交叉点通过交换结构控制器，能够在任何时候开启和闭合。若来了一个分组从A出入到Y输出，A和Y交叉的那个交叉点闭合，分组被发送。若此时也有了一个分组从B发送到X，BX交叉的那个也可以闭合，可同时。

输出端口

与出入端口很类似，不过注意输出端口也有可能会排队。

排队

输出端口排队

- 设有N条输入链路，N条输出链路
- R_{line}分组到达速率（每秒R_{line}个分组）
- R_{switch}从输入端口到输出端口能够移动分组的速率（设$R_{switch} = NR_{line}$）
- 设分组以同步的方式到达输入端口，以同步的方式从输出端口输出

第一个时刻同时在N条输入链路上各到达一个分组，所有分组都发向同一个输出端口，那么在下一个时刻，N个分组都被堆积在那一条输出链路上，由于输出链路在一个单位时间内只能发出一个分组，会在输出链路上形成排队。

若此时输出端口的缓存大小不够，会造成丢包。

缓存大小B的设置：

$$B = RTT * C$$

RTT为平均往返时延，C为链路容量。

若有大量的TCP流（N）流过一条链路：

$$B = RTT * C/\sqrt{N}$$

N通常非常大。

分组调度程序：当输出端口出现排队时，根据分组调度程序来选择发送哪一个分组。

- FCFS：先来先服务
- 加权公平排队WFQ：在具有排队等待传输的分组的不同端到端连接之间公平的共享输出链路

如果缓存真不够了，需要丢弃分组：

- 丢弃到达的分组（弃尾drop-tail）
- 删除一个或多个已经排队的分组

主动队列管理算法AQM：在缓存被填满前就丢弃或者标记一个分组的策略。

随机早期检测RED：一种AQM算法

- 为输出队列长度维护一个加权平均值，当一个分组到达时：
  - 如果平均队列长度小于最小阈值min_{th}，将该分组加入队列
  - 如果平均队列长度大于最大阈值max_{th}，将该分组标记或丢弃
  - 否则在两者之间，则分组以某种概率被标记或丢弃，一般是长度、min_{th}、max_{th}的函数

输入端口排队

。。。

4.4网际协议：因特网中的转发与编址

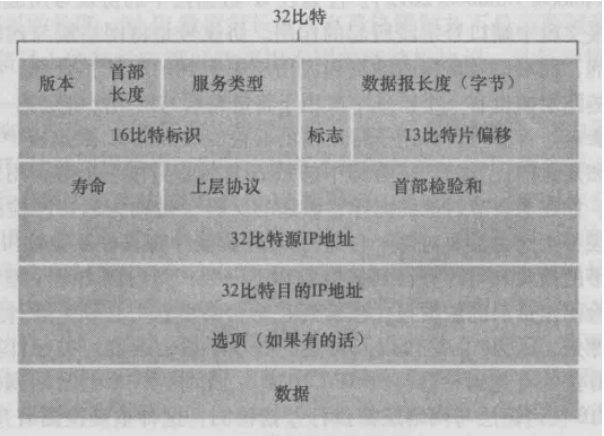
$$\text{因特网网络层} = \begin{cases} IP_{\text{协议}} \\ \text{路由选择部分（决定数据报从源到目的地所流经的路径）} \\ \text{报告数据报中的差错和对某些网络层信息请求进行响应的措施} \end{cases}$$

$IP_{\text{协议}}$目前有两个版本正在使用：$IPv4$（以下均是此版本的协议），$IPv6$

数据报格式

网络层分组被称为**数据报**。

$IPv4$数据报格式：



- **版本（号）**，规定IP协议版本，路由器可以根据IP版本确定如何解释数据报剩余部分
- **首部长度**，用首部长度标记IP数据报中数据部分实际从哪里开始，一般为20字节（无"选项"）
- **服务类型TOS**，区分类型不同的IP数据
- **数据报长度**，IP数据报总长度=首部+数据，单位字节，最长为65535字节
- **标识、标志、片偏移**，与IP分片有关
- **寿命TTL**，确保数据包不会永远在网络中循环，每当数据报由一台路由器处理，TTL-1
- **协议**，到达目的地时使用，指示IP数据报的数据部分应交给那个运输层协议（6给TCP，17给UDP）
- **首部检验和**，用于帮助路由器检测收到的IP数据报中的比特错误
    - 首部中的每2个字节当做一个数，用反码运算对他们求和，求和后的反码为检验和
    - 路由器对每个数据报计算首部检验和，若计算后不同，会丢弃
    - **为什么TCP/IP在运输层和网络层都执行差错检测?**
        - IP层支队IP首部计算了检验和，而TCP/UDP检验和是对整个TCP/UDP报文段进行的
        - TCP/UDP与IP不一定属于一个协议栈，TCP还可以运行在一个不同的协议上（ATM），而IP也可能携带不是TCP/UDP的数据
- **源和目的IP地址**
- **选项**，允许IP首部被扩展（IPv6以去掉）
- **数据（有效载荷）**，可以是TCP/UDP报文段，也可能是ICMP报文段

IP数据报分片

为什么要IP数据报分片?

因为并不是所有**链路层协议**都能承载相同长度的网络层分组（以太网帧可以承载不超过1500字节的数据，某些广域网链路的帧可承载不超过576字节的数据），在发送方与目的地路径上的每段链路可能使用不同的链路层协议，每种协议可能具有不同的MTU。这是如果IP数据报的长度比链路的MTU要大，需要把我的数据报分割成小的数据报，即分成**片（fragment）**。

最大传送单元MTU：一个链路层帧能承载的最大数据量

如何分片和重新组装?

如果在传送过程中，一个数据报被分片了，那么它不会再下一个路由器中被重新组装，而是在端系统中被重新组装。因此，

目的端系统收到了数据报时，它需要判断：

- 此数据报是不是别的数据报的片（怎么判断？）
- 如果是，确定何时收到了最后一片，并且如何将这些收到的片拼接到一起（可以通过发送主机IP地址和标识号来确定该和那一个数据报拼在一起）
- 当且仅当一个数据报的有效载荷在IP层已被完全重构，才会被传递给目的地运输层，否则会被丢弃

发送主机在生成一个数据报时，为了让目的主机能够重新拼起来，需要：

- 为数据报贴上**标识号**（通常将它发送的每个数据报标识号+1）

路由器在分片的时候，需要：

- 更改**标志**，最后一个片的标志比特是0，其他所有片的标志比特被设为1（这样可以使目的主机确保自己收到了最后一片）
- 更改**偏移字段**，偏移字段指定改片应放在初始IP数据报的哪个位置（可使目的主机查看是否有片丢了）（偏移字段的值会被*8之后再被用来确定真正的偏移量）

分片的坏处?

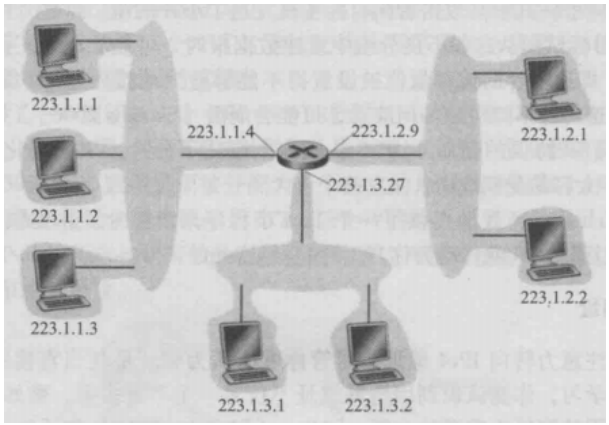
- 是路由器和端系统更为复杂
- 分片可能被用于生成致命的DoS攻击
    - 攻击者发送了一系列古怪的、无法预计的片
    - 攻击者发送交迭的IP片（偏移量的值被设置的不能够适当的排列）

IPv4编址

接口

主机与物理链路之间的边界叫接口；路由器与它的任意一条链路之间的边界也叫作接口。

每一个接口都有一个IP地址!



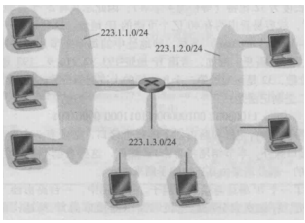
IP地址

长度：32bits（最多有2^32个地址，所以约有40亿个可能的IP地址）

点分十进制记法：地址中的每个字节用他的十进制形式书写，各字节间用点隔开

子网

上图中，223.1.1.1-223.1.1.4构成了一个**223.1.1.0/24**的子网，/24称为子网掩码，意为最左24位比特相同的，**223.1.1.0/24**子网由三台主机接口和一个路由器接口组成。



如何确定子网?

分开主机和路由器的每个接口，产生几个隔离的网络岛，使用接口端接这些网络端点。这些隔离的网络中的每一个都叫做一个子网。

重要观察：

- **子网之间通信不需要经过路由器，子网之间通信需要经过路由器。**
- **路由器每个端口连接一个子网。**

因特网地址分配策略

无类别域间路由选择CIDR

$$IP_{\text{地址}} = a.b.c.d/x$$

前缀：x最高比特构成的IP地址的网络部分

剩余32-x比特用于区分该组织内部设备

一个组织通常被分配一块连续的地址，即具有相同前缀的一段地址。

分类编址（CDIR之前的做法）

具有8、16、24比特的IP地址的网络部分被称为A、B、C类网络。一个子网只能是这三类中的一类

问题：

- C类子网只有$2^8 - 2 = 254$台主机（减去的2个用于特殊用途），太少，B类子网有65534主机，太多。B类太浪费，C类太少

广播地址

255.255.255.255

当一台主机中发出一个目的地址为255.255.255.255的数据报时，该报文会给同一个网络中的所有主机。

获取自己的地址

组织如何获得自己的地址

- 从ISP获取，ISP可能会将自己获得的地址块平均分成八块，分别给8个组织
- IP地址由**因特网名字和编号分配机构ICANN**管理（它还管DNS根服务器、分配域名和解决域名纷争）

主机如何获取自己的地址

动态主机配置协议DHCP（即插即用协议plug-and-play protocol）

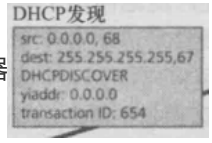
DHCP允许主机自动获取（被分配）一个IP地址，网络管理员能够配置DHCP，以使某给定主机每次与网络连接时能够得到一个相同的IP地址，或者某主机将被分配一个**临时的IP地址**（改地址每次与网络连接时也许是不同的）。

DHCP还允许主机得知它的**子网掩码**、第一跳路由器地址（**默认网关**）、**本地DNS服务器地址。**

DHCP是客户-服务器协议。新客户（主机）接入网络时，DHCP给它分配一个IP地址，客户退出网络时，DHCP将它的IP地址回收。**所以一个采用DHCP的组织需要的IP地址数量是最多同时在线人数。**

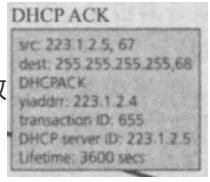
每个组织有一个DHCP服务器，或者DHCP中继代理。

获取一个IP地址的步骤：

- **DHCP服务器发现**
  - 新到的主机利用**DHCP发现报文**获取DHCP服务器 

  - 发现报文的源地址是0.0.0.0，表示本机
  - 目的地址是255.255.255.255，即广播给所有子网内的主机
  - 发到67端口，是UDP报文
- **DHCP服务器提供**
  - DHCP服务器收到发现报文，以提供报文作为相应 

  - DHCP服务器也发送一个广播报文，包含收到的**发现报文的事物ID**、**向客户推荐的IP地址**、**网络掩码**、**IP地址租用期（address lease time）**
  - 若组织中有多台DHCP服务器，每一台都会相应，这时就要由客户机去选择一个作为自己的IP地址
- **DHCP请求**
  - 新到达的客户从一个或多个服务器提供中选择一个，并向选中的服务器提供一个DHCP请求报文进行相应，回显配置参数 

  - 这是广播还是单播？？？为啥要是广播，DHCP ACK也不知道为啥要是广播
- **DHCP ACK**
  - 服务器用DHCP ACK报文对DHCP请求报文进行相应，证实所要求的参数 

DHCP不足之处

- 每当结点连接到一个新子网时，要从DHCP得到一个新的IP地址，当结点在移动式，无法维持TCP连接。

网络地址转换：NAT转换表

在家庭网络等网络中，如果仅采用DHCP，每个设备接入网时都要向DHCP服务器请求一个IP地址。但是子网的IP地址有限，可能的接入设备数量是无线的。为了解决这个问题，引入了NAT表。

以NAT-DHCP路由器为核心的网络中，所有接入此路由器的设备都由此路由器分配一个IP地址，此路由器对外界因特网有一个由DHCP服务器分配的IP地址。所有经由此路由器向外发送的数据报都先被打包成新的样子，返回的数据报也先查找NAT表，在由此路由器分发给个主机。

打包和解析过程：

- 主机向NAT-DHCP发送了一个数据报，源IP地址是路由器给主机分配的地址，端口号是主机的端口号。路由器将数据报的源地址改成自己的IP地址，端口号新分配一个替换进去，然后再NAT表中加入**目的IP地址，目的端口号**到**主机IP地址，主机端口号**的映射
- 当数据报被发回时，路由器将收到的数据报的**源IP地址，源端口号**作为查询依据，得到**主机IP地址，主机端口号**，再发送给该主机。
- 由于端口号字段是16bits，NAT可支持超过60000个并行使用路由器广域网一侧的IP地址的连接

NAT的缺陷：

- 端口号不能用于主机编址
- 路由器通常仅应当处理高达第三层的分组
- NAT协议违反了端到端原则（主机彼此直接对话），路由器不应修改IP地址和端口号
- 应使用IPv6解决IP地址短缺的问题
- 妨碍P2P应用程序，比如P2P文件共享应用、P2PIP语音应用

因特网控制报文协议（网络层第三个组件）：ICMP协议

ICMP被主机和路由器用来彼此沟通网络层的信息。

ICMP最典型的用途：**差错报告**

ICMP报文作为IP的有效载荷。若主机收到一个指明上层协议为ICMP的IP数据报，它分解出该数据报内容给ICMP，就像给TCP和UDP一样。

ICMP报文字段：

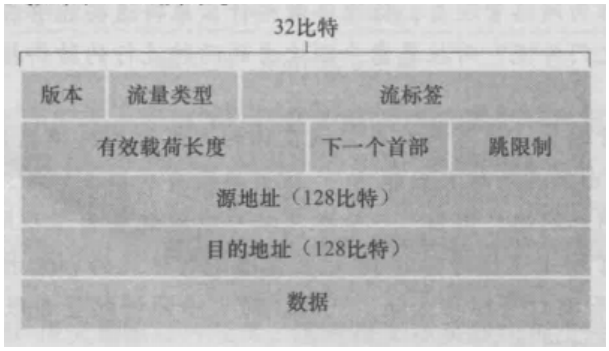
- **类型字段、编码字段**

ICMP 类型	编码	描述
0	0	回显回答（对 ping 的回答）
3	0	目的网络不可达
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制（拥塞控制）
8	0	回显请求
9	0	路由器通告
10	0	路由器发现
11	0	TTL过期
12	0	IP 首部损坏

- **引起ICMP报文首次生成的IP数据报的首部**和**前8字节的内容**
- eg. ping程序发送一个**ICMP类型8编码0**的报文到指定主机，看到该回显请求，目的主机发回一个**类型0编码0**的ICMP回显回答。

IPv6

IPv6数据报格式



变化：

- 扩大的地址容量：源和目的地址都变成了128比特（不会用尽）
  - 引入任播地址：可将数据报交付给一组主机的任何一个
- 40字节首部（简化高效）
- 流标签与优先级
  - 流：要求进行特殊处理的一系列报文之一，比如音视频、高优先级用户承载的流量
  - 流量类型：给出流中某些数据报的优先级，以便指明某些数据报比其他应用数据报有更高的优先权

IPv6首部字段：

- **版本**，IP版本号
- **流量类型**，约等于IPv4中的服务类型TOS
- **流标签**，表示一条数据报的流
- **有效载荷长度**，给出IPv6数据包中跟在定长40字节首部后面的字节流量
- **下一个首部**，表示数据报的内容需要交付给哪个协议（=IPv4首部中的协议字段）或选项
- **跳限制**，（等于IPv4中的寿命）
- **源地址和目的地址**
- **数据**

不存在的字段：

- **分片/重新组装**，IPv6不允许路由器分片与重组，只能在源、目的地上执行，若数据报太大不能发送，路由器只能扔掉
- **首部检验和**，运输层和数据链路层已经执行了检验操作，所以网络层无需再检验（检验耗时太长）
- **选项**，选项字段也能是“下一个首部”

IPv4迁移至IPv6

没看（

4.5路由选择算法

默认路由器/第一跳路由器：与主机直接相连的路由器

源路由器：源主机的默认路由器

目的路由器：目的主机的默认路由器

路由选择问题：给定一组路由器以及连接路由器的链路，路由选择算法要找到一条从源路由器到目的路由器的“好”路径。“好”表示费用/代价最低。

三种分类：

$$\text{路由选择算法} = \begin{cases} \text{全局式路由选择算法（用完整的、全局性的网络知识计算出最低费用路径）} \\ \text{分散式路由选择算法（以迭代、分布式的方式计算出最低费用路径）} \end{cases}$$

$$\text{路由选择算法} = \begin{cases} \text{静态路由选择算法（人工干预调整转发表）} \\ \text{动态路由选择算法（当网络流量负载或拓扑发生变化时改变路由选择路径）} \end{cases}$$

$$\text{路由选择算法} = \begin{cases} \text{负载敏感算法（链路费用动态变化以反映用塞水平）} \\ \text{负载迟钝算法（费用不反应用塞水平）} \end{cases}$$

全局式路由选择算法global routing algorithm

输入：以所有结点之间的连通性即所有链路的费用作为输入（要求在算法开始前可以获得这些信息）

具体算法：

链路状态算法(LS)

获取网络拓扑和所有链路费用的方法：让每个节点向网络中所有其他节点广播链路状态分组，包括链路的特征和费用，经常由**链路状态广播算法**来完成。这会使所有结点具有了该网络的等同的、完整的视图。

获取全局视图后，用**Dijkstra算法**计算最好路径。（复杂度$\mathcal{O}(n^2)$）

分散式路由选择算法decentralized routing algorithm

没有结点拥有关于所有网络链路费用的完整信息，每个结点仅有与其直接相连链路的费用知识即可开始工作，通过迭代计算过程并与相邻接点交换信息计算出最低费用路径。

算法为什么是分布式的?

因为每个结点都要从一个或多个直接相连的邻居接收信息，执行计算，然后把结果返还给邻居。

算法为什么是迭代的?

因为此过程一直要持续到邻居之间无更多信息要交换为止。

算法为什么是异步的?

因为不要求所有结点步伐一致地操作。

算法为什么是自我终止的?

因为没有计算应该停止的信号，算法就停止了。

具体算法：

距离向量算法(DV)

算法依赖的基础：$Bellman-Ford$方程

$$d_x(y) = min_v\{c(x,v) + d_v(y)\}$$

其中$d_x(y)$表示结点x到结点y的最短路径。v是x的所有邻点。

算法用一个距离向量$D_x = \{d_x(y) : y \in N\}$记录$x$到网络中所有结点的距离估计值。利用

$$D_x(y) = min_v\{c(x,v) + D_v(y)\}$$

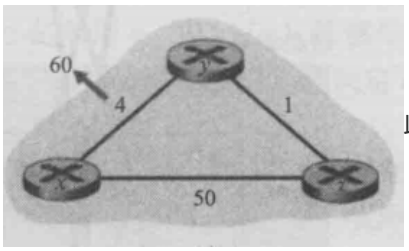
更新距离向量。若D_x改变了，x就向其所有邻居发送此向量。

只要所有结点以异步的形式计算距离向量，那么每个费用估计$D_x(y)$收敛到$d_x(y)$。

算法过程：

1. 每个路由器先初始化自己的距离向量，然后发送给自己的所有邻点。
2. 每个路由器接收到邻点发送的他们的距离向量，更新自己的距离向量。
3. 若自己的距离向量有所改变，将自己的距离向量再发送给自己所有的邻点。
4. 重复2-3，直到所有路由器的距离向量都不再变动。

链路费用改变：

1. **某一链路费用减少：**
    1. (u,v)边的费用减少，v检测到了，更新自己的向量表，发送给自己所有的邻点。
    2. 邻点更新自己的向量表，再传播。
    3. **好消息传播很快**
2. **某一链路费用增加：**（无穷计数问题）
    1. 

此时y的向量表：

D_y	**x**	**y**	**z**
x	0	4	5
y	4	0	1
z	5	1	0

    2. (x,y)边的费用由4增加到60，y检测到了，更新$D_y(x)=min\{60+0,1+5\}=6$

D_y	**x**	**y**	**z**
x	0	4	5
y	6	0	1
z	5	1	0

    3. y把自己的向量表发给x和z
    4. z的向量表：

D_z	**x**	**y**	**z**
x	0	4	5
y	6	0	1
z	5	1	0

    z的向量表更新：$D_z(x)=min\{50+0,6+1\}=7$

D_z	x	y	z
x	0	4	5
y	6	0	1
z	7	1	0

5. z再把向量表发给y，他们俩互相发，直到z到x的距离大于51，共有44次迭代（z->y，y->z算两次，每次迭代+1）
6. **坏消息传播很慢**

LS与DV算法比较

LS与所有结点交谈，DV只与邻接结点交谈。两者具体区别：

1. 报文复杂性
    1. LS
        - LS要发送$\mathcal{O}(|N||E|)$个报文（N是路由器个数，E是边条数）
        - 某一链路费用变动时，需要向其他所有结点发送数据
    2. DV
        - 两个相邻结点交换报文
2. 收敛速度
    1. LS：$\mathcal{O}(|N|^2)$的算法
    2. DV：收敛很慢，有可能遇到无穷技术问题
3. 健壮性
    1. LS：路由计算在每台路由器上都进行，就算一台坏掉，其他也可以进行计算，比较健壮
    2. DV：若出现了一个不正确的费用，它可能会被传向全网络

路由层次选择

为什么需要把所有的路由器划分层次?

1. 规模：路由器数量越来越多，在每个路由器内部存储的数据和与其他路由器交换的数据越来越多
2. 管理自治：某些公司需要按自己的意愿运行路由器，或对外隐藏内部路由器的面貌

如何将路由器划分层次?

利用**自治系统AS。**

运行在同一AS下的路由器执行同一种路由选择算法，且拥有彼此的信息。

自治系统内部路由选择协议：一个自治系统**内部**运行的路由选择协议

它的任务：

- 从相邻AS中获得可达性信息
- 向该AS中所有路由器传播可达性信息

网关路由器：一个AS中，与其他AS相连的路由器。

热土豆路由选择：一个路由器到同一子网有不同路径，需要经过不同网关，热土豆路由选择方法选择具有最小的最低费用的网关加入转发表

4.6因特网中的路由选择

自治系统内部路由选择协议

RIP

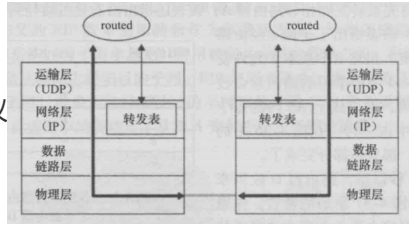
跳：源路由器到目的子网的最短路径经过的子网数量

RIP协议中：

- 每条链路的费用为1
- A路由器到b子网的总费用=A到b的最短路径的跳数
- 一条路径的最大条数=15，所以RIP被限制在网络直径不超过15跳的自治系统内

RIP响应报文/RIP通告（UDP）：

- 每对邻接的路由器间每30s交换一次，若180s还未收到某一邻居的报文，则不再视该邻居路由器为可达，则修改本地选择表，并向邻居通告。
- 包含该路由器或主机所在AS内多达25个子网列表，和路由器/主机到子网的距离
- RIP用运输层协议UDP实现网络层协议



路由选择表：

- 包含距离向量和转发表
- 每30s收到邻居的报文，更新本路由器的选择表

OSPF

核心：洪泛链路状态信息的链路状态协议+Dijkstra最低费用路径算法

如何找最短路径?

AS中的每一台路由器都构建一幅关于整个AS的完整拓扑图，在图中运行Dijkstra算法，确定一个以自身为根结点的到所有子网的最短路径树。

链路费用：由网络管理员定义

- 都为1：实现最小跳数
- 与链路容量成反比：不鼓励流量使用低带宽链路

如何构建拓扑图：

- 路由器向自治系统内**所有**其他路径广播路由选择信息
  - 链路状态（费用、连接状态）发生变化时，路由器会广播状态信息
  - 链路状态未发生变化，每30min广播一次

优点：

- 安全：只有受信任的路由器能参与一个AS的OSPF协议，可防止恶意入侵者（利用明文口令或秘钥）
- 多条相同费用的路径：多条路径费用相同时，OSPF允许多条路径，无需仅选择单一路径承载所有流量
- 对单播与多播路由选择的综合支持：多播：MOSPF提供对OSPF的扩展
- 支持在单个路由选择域内的层次结构

区域边界路由器：负责对流向该区域以外的分组提供路由选择

主干区域

- 为AS内其他区域之间的流量提供路由选择
- 包含所有区域边界路由器和一些非边界路由器

AS内区域间的路由选择：分组->区域边界路由器--经过主干区域-->区域边界路由器->目的地

自治系统间的路由选择协议

边界网关协议BGP

交换信息的途径：在179端口的半永久TCP连接

BGP对等方：TCP连接两端点的两台路由器

BGP会话：沿着TCP连接发送的所有BGP报文

- 外部BGP(eBGP)会话：跨越两个AS的BGP会话
- 内部BGP(iBGP)会话：同一个AS中的两台路由器之间的BGP会话

前缀聚合：若一个AS与138.16.65/24和138.16.64/24两个子网相连，那么这个AS向别的AS通告的时候会通告说我和138.16.65/22相连

如何知道别的AS可达哪些子网?

- 每个AS向相邻的AS通告与自己相连的子网前缀列表
- 每个AS由eBGP收到相邻的AS的通告，利用iBGP向本AS中的其他路由器发布前缀
- 一台路由器的值一个新前缀时，为该前缀在转发表中创建一个项

自治系统号ASN：标识一个自治系统

路由：包括一些BGP属性的前缀。（BGP对等方相互通告的其实是路由）

BGP属性：

- AS-PATH：包含前缀的通告已经通过的那些AS
  - 每传送到一个AS中，该AS就将他的ASN加入到AS-PATH中
  - 防止循环通告
- NEXT-HOP：开始某AS-PATH的路由器接口，其实是提供一个子网地址（不是和发送方相连的子网前缀，而是发送方的子网）。
  - 当该通告被传送到一个AS内部的某一路由器时，路由器查看NEXP-HOP的内容，得到目的子网地址，然后该路由器又从AS内部路由选择算法那里得到本路由器到该子网的最短路径上的第一个路由器I，将（前缀，I）加入转发表

如果一台路由器接收到了许多相同子网前缀的路由，如何选择?

按照以下规则选择，直到只剩一个路由：

- 每个路由具有一个本地偏好值的属性，先选择本路由器偏好的路由
- 余下的路由中（偏好值相同），最短的AS-PATH留下
- 余下的路由中（偏好值和AS-PATH长度都相同），选择具有最靠近（具有最低费用路径）NEXP-HOP路由器的路由【热土豆】
- 余下的路由中，使用BGP标识符选择

第五章 链路层

5.1链路层概述

结点node：云心链路层的任何设备都成为结点，包括主机、交换机、Wifi接入点

链路link：沿着通信路径连接相邻接点的通信信道

链路层帧：传输结点将数据报封装在链路层帧中

链路层提供的服务：

- 成帧：将网络层数据报用链路层帧封装起来
- 链路接入：媒体访问控制（MAC）协议规定了帧在链路上传输的规则
- 可靠交付：保证无差错的经链路层移动每个网络层数据报（通过确认和重传）
- 差错检测和纠正：检测比特差错（让发送结点在帧中包括差错检测比特，接受结点检查，硬件实现）

链路层在何处实现：

链路层主体部分在**网络适配器(network adapter，也称网络接口卡NIC)**中实现。核心是链路层控制器，是一个实现了许多链路层服务的专用芯片。

5.2差错检测和纠正技术

差错检测和纠正比特EDC：保护网络层来的数据报和链路帧首部信息，和链路层数据比特一起被发送给接收方。（注意传输过程中数据比特和EDC都有可能改变）

三种检测差错的技术：

奇偶校验

使用单个**奇偶校验位(parity bit)**，设数据比特有d个

- **发送方：**
  - 偶校验方案：选择校验比特的值，使得【数据比特+校验比特】这d+1个比特中的1的总数为偶数
  - 奇校验方案：.....奇数
- **接收方：**先计算这d+1个比特中1的数目
  - 偶校验方案：有奇数个1，说明出现了奇数个差错；有偶数个1，可能无差错，可能出现偶数个差错
  - 奇校验方案：类似

使用**二维奇偶校验**

- **发送方：**
  - 将d个比特信息和划分成i行j列（d=i*j）
  - 对每一行和每一列就进行单奇偶校验，那么帧中就多包含了i+j+1个比特
- **接收方：**
  - 若数据比特中的某一位发生了改变，可以通过行列校验位进行定位和纠正。

前向纠错FEC：接收方监测和纠正差错的能力。

检验和方法

将d比特数据视为k比特整数的序列。

因特网检验和：将d比特数据视为16比特整数的序列，将所有k比特整数加起来的结果的反码。

接收方：对接收的数据（包括检验和）视为16比特整数序列，加起来的和取反，若全为1，则无错。

循环冗余检测CRC

之后再看！

5.3多路访问链路和协议

点对点链路：链路一端的单个发送方和链路另一端的单个接受方组成。

广播链路：让多个发送和接收结点都连接到相同的、单一的、共享的广播信道上。

多路访问问题：如何协调**多个发送和接收结点**对**一个**共享广播信道的访问。

多路访问协议/多址接入协议MAC：结点通过这些协议来规范他们在共享的广播信道上的传输行为。

碰撞：一结点同时接到多个帧，那么在该结点处发生碰撞。

MAC协议比较

信道划分MAC协议：
- 重负载下高效：没有冲突，节点公平使用信道
- 轻负载下低效：即使只有一个活跃节点也只能使用1/N的带宽

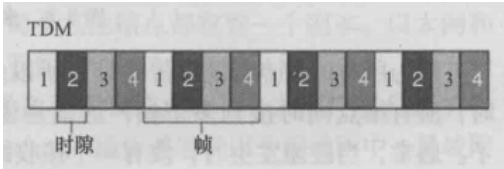
随机接入MAC协议：
- 轻负载时高效：单个活跃节点可以使用整个信道
- 重负载时低效：频繁发生冲突，信道使用效率低

轮流协议（试图权衡以上两者）：
- 按需使用信道（避免轻负载下固定分配信道的低效）
- 消除竞争（避免重负载下的发送冲突）

信道划分协议

时分多路复用TDM

帧和时隙：TDM将时间划成时间帧，把每一帧划分成为多个时隙。



把每个时隙分配给N个节点中的一个，他在被指派的时隙中才发送分组。

优点：

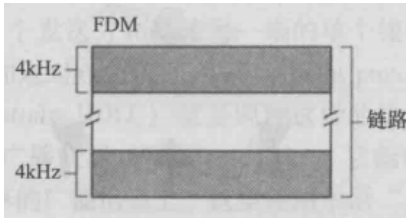
- 消除了碰撞
- 公平：每个结点都有R/N bps的带宽

缺点：

- 每个结点被限制在了R/N bps带宽内
- 每个结点需要等待自己的时隙到来才能发送

频分多路复用FDM

FDM把R bps的信道划分成了N个不同的频段，每个频段具有R/N bps的带宽，分配给每个结点。



优缺点与TDM类似。

码分多址CDMA

CDMA对每个结点分配一种不同的编码，每个结点用他唯一的编码来对它发送的数据进行编码。接收方如果知道发送方的编码，那么如果所有结点同时传输，接收方可以辨别哪个出发送方发送的数据。

随机接入协议

一个结点总是以信道的全部速率进行发送。有碰撞时，涉及碰撞的结点反复重发它的帧，直到该帧无碰撞地通过。

若出现碰撞，等待一个随机时延，在时延经过后再重发。

注：

ALOHA和CSMA都能保证如果只有一个结点活跃，它可以使用R的带宽；但不能保证如果有M个结点活跃，每个活跃结点的吞吐量接近R/M。轮流协议可以保证第二点。

时隙ALOHA

假设：所有帧都是L比特；时间被划分成L/R秒的时隙；结点只在时隙起点传输；结点是同步的，每个结点都知道时隙何时开始；所有结点都能在时隙结束前检测到此时隙内的碰撞事件。

时隙ALOHA的执行过程：

- 结点有新的帧发送时，在下一个时隙开始时发送
- 没有碰撞，不考虑重传
- 发生碰撞，在时隙结束前检测到碰撞事件，那么以概率p在后续的每个时隙中重传此帧，直到该帧无碰撞地传过去。

优点：

- 若只有一个结点在传输，他可以获得R的速率
- 高度分散，每个结点独立地决定什么时候重传

效率：

N个结点，时隙ALOHA的效率是$Np(1-p)^{N-1}$。

大量结点，最大效率为$1/e = 0.37$

ALOHA

如果一个传输的帧与一个或多个传输经历了碰撞，这个结点在传输完整个碰撞帧后立即以概率p重传；否则等待一个帧传输的时间，再以p的概率重传或1-p的概率等待（一个帧传输时间）。

效率：

大量结点，最大效率为$1/2e = 0.37$

载波侦听多路访问CSMA

- **载波侦听**：传输前先听信道，若信道被占用，等待直到检测到一小段时间没有传输，再传输

用了CSMA为什么还会发生碰撞?

因为数据在信道中的传输有时间，A在发，但是传到B需要时间（**信道传输时延**），B在接受到之前都认为信道是空闲的。

具有碰撞检测的载波侦听多路访问CSMA/CD

- **碰撞检测**：一个结点在传输时一直侦听此信道，若检测到另一结点在干扰帧，就停止传输，等待一个随机时间，再进入"侦听-空闲时传输"的循环

仅仅的CSMA不能碰撞检测，即如果发生了碰撞，它还是会继续发送碰撞帧，有了碰撞检测，可以立即避免碰撞。

以太网采用此协议

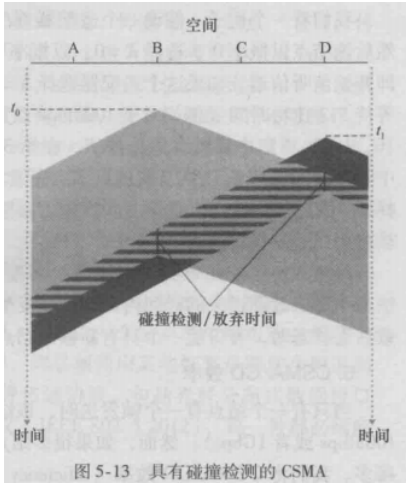


图 5-13　具有碰撞检测的 CSMA

如何随机地选等待时间?

二进制指数后退。（碰撞次数越多，等待时间越长）

经历n次碰撞，在[0,...,2^n-1]中选一个值为K，延迟K*512比特时间。

CSMA/CD效率：

- 只有一个帧，该帧能以全速率传输
- 大量结点传输时有近似式：

$$\text{效率} = \frac{1}{1 + \frac{5d_{prop}}{d_{trans}}}$$

结论：应控制以太网的规模

轮流协议

轮询协议polling protocol

蓝牙利用此协议

选择一个主结点，主节点轮询每个结点：告诉第一个结点它能够传输的帧的最多数量，第一个结点发送完毕后，主节点再访问下一个结点，重复。

优点：

- 清楚了随机接入协议的碰撞和空时隙，提高了效率

缺点：

- 引入了轮询时延
- 主结点也要轮询非活跃的结点，降低了效率

令牌传递协议token-passing protocol

无主结点，**令牌**（token，小的特殊帧）在结点之间交换。

一个结点收到了令牌，若它要传输帧，那么它持有并发送最大数目的帧，否则他传输给下一个。

优点：

- 令牌传递是分散的，效率很高

缺点：

- 一个结点故障会使信道崩溃
- 一个结点偶然忘记释放令牌，需要调用恢复步骤

5.4交换局域网

由于交换机只运行在链路层，它不识别网络层的IP地址，所以交换机构建的网络利用链路层地址来转发链路层帧通过交换机网络。

5.4.1链路层寻址和ARP

MAC地址

链路层地址=MAC地址=LAN地址=物理地址

一个网络适配器具有一个MAC地址，该MAC地址不会改变。没有两块适配器有相同地址。MAC地址被IEEE管理。

MAC地址长度**6Btyes**，共有2^{48}个可能的MAC地址。通常用16进制表示法：1A-23-F9-CD-06-9B

发送适配器：

- 将目的适配器的MAC地址插入到该帧中，并将该帧放到局域网上

接收适配器：

- 一个适配器接收到一个帧时，先检查MAC地址，若MAC地址和自己匹配，才拆包；否则丢弃。

MAC广播地址：发送方想让局域网上所有其他适配器来接受并处理它打算发送的帧时，便在目的地址字段加入特殊的MAC广播地址FF-FF-FF-FF-FF-FF在哪加?

有了IP地址为什么还需要MAC地址?

- 不是只有IP和因特网，还有其他的网络层协议，如果没有MAC，对于其他网络层协议将不能支持
- 如果只用网络层地址的话，它被存在适配器的RAM中，每次启动或移动是都要重新配置。

- 如果不使用MAC地址，每次到网络层去看地址的话，会导致主机将被局域网上发送的每个帧中断

子网内发送帧：地址解析协议ARP

地址解析协议用于将**网络层地址**和**链路层地址**之间转换。

ARP的用途：发送主机的ARP模块可以将**相同局域网上的IP地址**作为输入，返回该IP地址对应的MAC地址，若输入不是同一子网上的IP地址，返回错误。

ARP表在哪里：每台主机或路由器的内存中。

ARP表包含：IP地址到MAC地址的映射关系，每一个IP地址的寿命TTL（表示从表中删除每个映射的的时间，过期时间通常20min）。

地址解析协议（数据报发送过程）：

- ARP模块到ARP表中查找该目的IP地址是否有对应的MAC地址
  - 若有且不过期，把该MAC地址加入链路层帧中，发送数据报
  - 若没有
    - 发送方构造一个ARP查询分组（属于ARP分组，包括发送和接收的IP地址和MAC地址）
    - 发送方适配器用MAC广播地址发送此分组，发送到子网中（发送方地址段=自己的MAC地址和IP地址，目标字段=B的IP地址）
    - 子网中每个适配器都接收到，并将返祖向上传递给ARP模块，每个ARP模块都检查自己主机的IP地址是否与该目的地址相匹配，若匹配，返回一个ARP响应分组。
    - 发送方收到响应分组，更新ARP表，发送数据报。

ARP报文格式：真正的帧还要在ARP外封一层首部的，这相当于有效数据

- 硬件类型：硬件接口类型。对于以太网，该值为“1”。
- 协议类型：高层协议地址类型。对于IP地址，该值为 0800_{16}。
- 操作：ARP请求为1，ARP响应为2
- 在以太网上，ARP报文封装在以太帧中传输

注意：

1. ARP查询分组是广播帧，ARP响应分组是标准帧
2. ARP表是自动建立的，无需管理员配置

发送数据报到子网外

子网1的A主机想将一个数据报发送给子网2的B主机：

A主机的适配器需要将MAC地址设置成为第一跳路由器的与此子网相连的适配器地址，此地址可以在ARP表中被找到，然后当该帧到达第一跳路由器时，路由器拿到网络层的数据报，读取目的IP地址，从转发表中读到这个数据报该被送往那个接口，被送过去之后，接口外还有一个适配器，该适配器再从路由器的ARP表中，找到下一个MAC地址，包装，发送，直到数据报达到B主机。

5.4.2以太网

最流行的有线局域网技术。

以太网帧结构

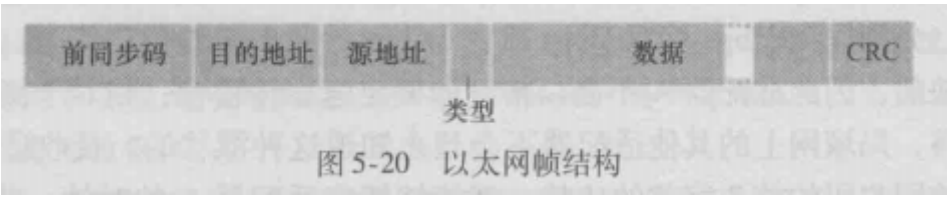


图 5-20　以太网帧结构

- 数据字段（46-1500Bytes），承载IP数据报
  - 数据报<46字节，要填充套46
  - 数据报>1500字节，要分片
- 目的地址（6Bytes）：目的MAC地址
- 源地址（6Bytes）：发送方适配器MAC地址
- 类型字段（2Bytes）：允许以太网复用多种网络层协议
  - 分解的时候查看类型字段，确定用哪种网络层协议解读。
  - ARP协议有自己的类型编号
- CRC（4Bytes）
- 前同步码（8Bytes）：前7字节都是10101010，用于唤醒适配器和是目的适配器与发送适配器的时钟同步；最后一个字节是10101011，此字节的后两比特警告目的适配器：重要内容来了。

以太网提供的是**无连接（无握手）、不可靠（不确认也不否定确认）服务**，如果网络层使用UDP，可能会看到”间隙“。

为什么有最小帧长的要求?

为确保结点在发送结束前检测到冲突，帧的发送时间必须足够长

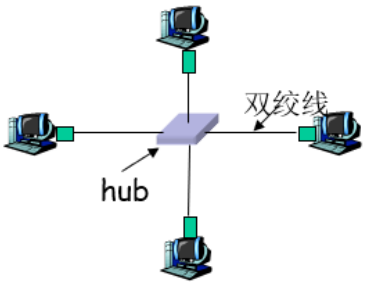
1. 结点检测冲突需要时间
2. 假设信号在相距最远的两个适配器之间的往返延迟为2τ，则帧的发送时间不应小于2τ，即帧的最小长度≥链路速率×2τ

为什么最小帧长为64字节（不包括前导码）?

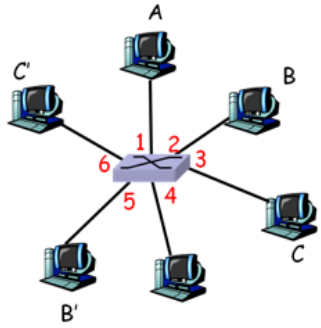
根据早期以太网的最大直径（2500米）和数据速率（10Mbps）计算得到

讨论：共享式以太网和交换式以太网

- 共享式以太网：
  - 集线器的所有端口位于同一个冲突域
  - 任一时刻最多只允许一个主机发送
  - 网络规模（节点数量）与网络性能的矛盾无法解决



- 交换式以太网：
  - 交换机的每个端口为一个冲突域
  - 多对端口可以同时通信
  - 网络的集合带宽=各个端口的带宽之和
  - 从根本上解决了网络规模与网络性能的矛盾



5.4.3链路层交换机

转发和过滤

过滤：决定一个帧该转发到某个接口还是应当将其丢弃的交换机功能

转发：决定一个帧应该被导向哪个接口，并把帧移动到那些接口

交换机表：包含某局域网上某些主机和路由器但不必是全部的表项

- MAC地址-通向该MAC地址的接口-表项放置在表中的时间

转发过程：

- 一个帧从交换机的x接口到达，交换机取得目的MAC地址，在交换机表中索引
- 若表中没有对应表项，交换机广播该帧，即向所有（除去x）的接口转发该帧
- 若表中有对应表项，且表项是目的地址-x接口（即对应接口是到达接口），那么将帧过滤，无需转发
- 若表中有对应表项，且表项是目的地址-y接口，y≠x，将帧转发至y接口

如何配置交换机表：自学习

---

1. 初始表为空
2. 存储每个入帧
   - 源地址的MAC地址
   - 该帧到达的接口
   - 当前时间
3. 一段时间后若交换机没有接收到以该地址作为源地址的帧，将该表项删除

交换机是**即插即用设备**，不需要网络管理员或用户干预。

链路层交换机的性质

- 消除碰撞：没有因碰撞而浪费的帧，不用CSMA/CD
- 异质的电路：将链路彼此隔离，不同链路能够以不同速率运行
- 管理：安全性，易于网络管理

交换机和路由器的比较

---

1. 交换机用MAC地址索引，路由器用IP地址索引
2. 交换机是第二层的分组交换机，路由器是第三层的
3. 交换机不能连接异构链路（即MAC协议不同的网络），因为交换机只是按原样转发帧；路由器可以连接异构链路，因为路由器需重新封装链路层帧
4. 交换机不能阻断广播帧的传播；路由器可以阻断广播帧的传播

交换机：

- 优点：
    1. 即插即用
    2. 高分组过滤和转发率
- 缺点
    1. 为防止广播帧的循环，交换网络的活跃拓扑限制成为一个生成树（不能有环）
    2. 大型交换网络中共有大的ARP流量和处理量
    3. 交换机对于广播风暴不提供任何保护措施

路由器：

- 优点：
    1. 网络寻址是分层次的，当网络中存在冗余路径是，分组不会通过路由器循环，所以分布不会被限制在一个生成树上
    2. 允许以丰富的拓扑结构构件因特网
    3. 对第二层的广播风暴提供防火墙的保护
- 缺点：
    1. 不是即插即用的，需要人为配置IP地址
    2. 路由器对每个分组处理的时间长

什么时候用交换机，什么时候用路由器?

几百台主机组成的小网络：交换机

几千台主机组成的更大网络中：交换机+路由器

5.4.4虚拟局域网

虚拟局域网（VLAN）：支持VLAN的交换机允许经一个单一的物理局域网基础设施定义多个虚拟局域网。在同一个VLAN中的主机彼此通信，仿佛它们是以交换机连接的。

网络管理员把交换机的接口划分为组，每个组构成一个VLAN。各组之间的帧相互隔离。

交换机如何在VLAN间转发帧?

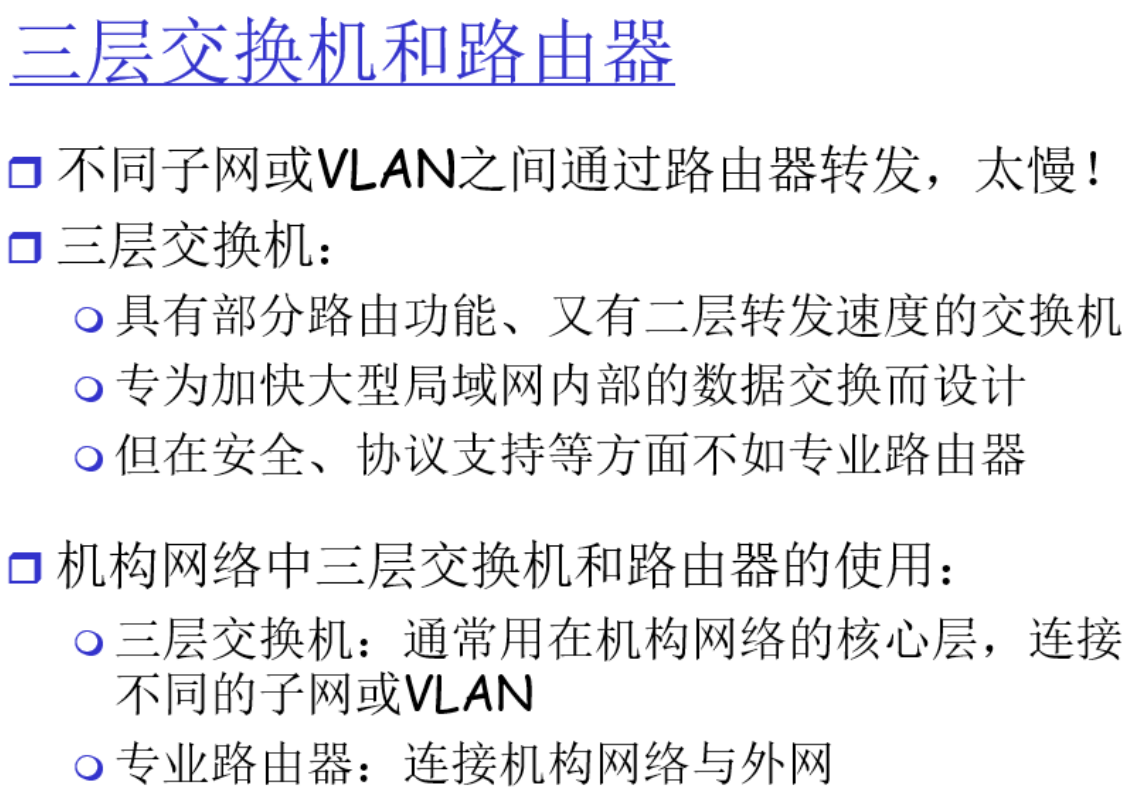
当一个帧到达时，交换机判断该帧属于哪个VLAN，查找配置表得到该VLAN对应的端口，在该VLAN对应的所有端口上转发帧。

两个隔离的分组如何相互通信?

将VLAN交换机的一个关口与一台外部路由器相连，并将该端口配置为属于两个隔离的分组，即使两个分组共享相同的物理交换机。帧的路径：分组1中的主机->分组1的VLAN->路由器->分组2的VLAN->分组2的主机

VLAN干线连接：每台交换机上的一个特殊端口被配置成为干线端口，两台交换机可以通过干线端口互联。通过检查跨越干线的帧中的首部**VLAN标签**（新的以太网帧中有）可以知道该帧时发往哪个VLAN

三层交换机为什么快



三层交换机为什么快？

- 路由器转发IP包的过程：
  1. 用目的IP地址查找转发表，获得下一跳IP地址及端口
  2. 利用ARP获得下一跳MAC地址
  3. 用下一跳MAC地址构造链路层帧，发送
- 三层交换机转发IP包的过程：
  1. 将以上第1、第2步的结果缓存到本地三层转发表中
  2. 用目的IP地址查找三层转发表：
     1）若命中，直接用下一跳MAC地址构造链路层帧，发送
     2）若未命中，执行以上第1、2、3步
- 三层交换机转发速度快的原因：
  - 一次选路，多次转发