

# 计算机通信网络

---

计算机通信网络

Chapter1 概述

Chapter2 协议分层原理

2.1 OSI/RM概述

2.2 层的相关概念

2.3 连接及其操作

2.4 数据传送

2.5 服务原语

Chapter3 数字通信基础

3.1 数据传输相关概念

3.1.1 数据、信号和码元

3.1.2 波特率和比特率

3.1.3 数据传输

3.1.4 信道及其参数

3.1.5 基带传输和宽带传输

3.1.6 异步通信和同步通信

3.1.7 串行通信和并行通信

3.2 信号编码技术

3.2.1 数字数据的模拟信号调制

3.2.2 数字数据的数字信号编码

3.2.3 模拟数据的数字信号编码

3.2.4 模拟数据的模拟信号调制

3.3 信道复用

3.3.1 频分复用FDM

3.3.2 时分复用TDM(同步)

3.3.3 统计时分复用STDM(异步)

3.3.4 波分复用WDM

3.3.5 码分复用CDM

3.3.6 正交频分复用OFDM

3.4 数据交换技术

3.4.1 电路交换(电话)

3.4.2 报文交换(电报)

3.4.3 分组交换

3.4.4 三种交换技术的比较

3.4.5 虚电路交换

Chapter4 物理层

4.1 物理层的基本概念

4.2 物理层下的传输媒介

4.2.1 导向传输媒体

4.2.2 非导向传输媒体

4.3 Internet的本地接入

4.3.1 拨号接入

4.3.2 ADSL接入(非同步)

4.3.3 基于社区电视系统

4.3.4 无线接入

Chapter5 数据链路层

## 5.1 数据链路层概述

- 5.1.1 数据链路层的定义
- 5.1.2 数据链路层的功能
- 5.1.3 数据链路层的服务类型

## 5.2 帧的构成

- 5.2.1 字符记数法
- 5.2.2 带字符填充的起始字符和终结字符法
- 5.2.3 带位填充的起始/终结标志法
- 5.2.4 物理层编码违例法

## 5.3 差错控制

- 5.3.1 差错检测和纠正
  - 5.3.1.1 汉明纠错码
  - 5.3.1.2 奇偶校验码
  - 5.3.1.3 校验和
  - 5.3.1.4 块校验码BCC
  - 5.3.1.5 循环冗余码CRC
- 5.3.2 反馈重发差错控制
  - 5.3.2.1 等待式ARQ
  - 5.3.2.2 回退N帧ARQ
  - 5.3.2.3 选择性重发ARQ
  - 5.3.2.4 通信效率

## 5.4 滑动窗口式流量控制

- 5.4.1 滑窗式流控的主要思想
- 5.4.2 滑动窗口协议主要工作原理

## 5.5 数据链路层协议实例

- 5.5.1 HDLC——高级数据链路控制
- 5.5.2 PPP——点对点协议

## 5.6 介质接入控制

- 5.6.1 广播信道的分配问题
  - 5.6.1.1 静态分配
  - 5.6.1.2 动态分配
- 5.6.2 查询技术
  - 5.6.2.1 roll-call查询
  - 5.6.2.2 hub查询
  - 5.6.2.3 探查法
- 5.6.3 ALOHA
  - 5.6.3.1 纯ALOHA
  - 5.6.3.2 时隙ALOHA
- 5.6.4 CSMA
  - 5.6.4.1 非持续型CSMA
  - 5.6.4.2 p-持续型CSMA
  - 5.6.4.3 1-持续型CSMA
- 5.6.5 CSMA/CD
- 5.6.6 令牌类接入控制技术

## 5.7 局域网标准

- 5.7.1 IEEE局域网标准
- 5.7.2 以太网
  - 5.7.2.1 经典以太网
  - 5.7.2.2 交换式以太网

- 5.7.2.3 快速以太网 (802.3u)
- 5.7.2.4 千兆以太网 (802.3z)
- 5.7.2.5 万兆以太网
- 5.7.3 无线局域网
  - 5.7.3.1 802.11无线局域网的组网模式
  - 5.7.3.2 802.11 MAC层协议
- 5.8 数据链路层交换
  - 5.8.1 局域网的扩展
  - 5.8.2 网桥
    - 5.8.2.1 透明网桥
    - 5.8.2.2 生成树网桥
    - 5.8.2.3 远程网桥
    - 5.8.2.4 源路由网桥
  - 5.8.3 交换机
  - 5.8.4 虚拟局域网VLAN
    - 5.8.4.1 VLAN的应用背景
    - 5.8.4.2 VLAN工作原理

## Chapter6 网络层

- 6.1 网络层概述
- 6.2 路由选择
  - 6.2.1 路由表
  - 6.2.2 路由选择算法
    - 6.2.2.1 随机路由选择
    - 6.2.2.2 洪泛式路由选择
    - 6.2.2.3 固定式路由选择(静态路由)
    - 6.2.2.4 自适应路由选择
- 6.3 网络拥塞控制
  - 6.3.1 网络拥塞的原因
  - 6.3.2 拥塞控制的基本原理
  - 6.3.3 拥塞预防策略
    - 6.3.3.1 虚电路子网中的拥塞控制
    - 6.3.3.2 数据报子网中的拥塞控制
- 6.4 网络流量控制
- 6.5 网络服务质量
  - 6.5.1 服务质量的需求
  - 6.5.2 流量整形
  - 6.5.3 包调度
  - 6.5.4 准入控制
  - 6.5.5 综合服务
  - 6.5.6 区分服务
  - 6.5.7 标签交换和MPLS
- 6.6 网络互联
  - 6.6.1 网络互联的概念
  - 6.6.2 网络互联的功能
  - 6.6.3 网络互联的要求
  - 6.6.4 网络互联的设备
    - 6.6.4.1 路由器
    - 6.6.5.2 三层交换机
- 6.7 因特网中的网络层

- 6.7.1 IP协议
  - 6.7.1.1 IP协议簇
  - 6.7.1.2 IP报文格式
  - 6.7.1.3 IP报文的分段与重组
  - 6.7.1.4 IP地址及其表示方法
  - 6.7.1.5 分类的IP地址
  - 6.7.1.6 划分子网
  - 6.7.1.7 无分类编址CIDR
  - 6.7.1.8 IP层转发报文的流程
  - 6.7.1.9 IPv6协议
- 6.7.2 ICMP、ARP、RARP协议
  - 6.7.2.1 ARP、RARP协议
  - 6.7.2.2 ICMP协议
- 6.7.3 VPN和NAT
  - 6.7.3.1 VPN
  - 6.7.3.2 NAT
- 6.7.4 路由选择协议
  - 6.7.4.1 路由信息协议RIP
  - 6.7.4.2 开放最短路径优先OSPF
  - 6.7.4.3 边界网关协议BGP

## Chapter7 传输层

- 7.1 传输层的功能
- 7.2 传输层的服务
- 7.3 传输层协议的要素
  - 7.3.1 寻址
  - 7.3.2 连接管理
    - 7.3.2.1 连接的作用
    - 7.3.2.2 连接的建立
    - 7.3.2.3 连接的释放
  - 7.3.3 流量控制和缓冲策略
  - 7.3.4 多路复用
  - 7.3.5 崩溃的恢复
- 7.4 因特网的传输层协议
  - 7.4.1 UDP
  - 7.4.2 TCP
    - 7.4.2.1 TCP概述
    - 7.4.2.2 TCP报文段的首部格式
    - 7.4.2.3 TCP的可靠传输-窗口机制
    - 7.4.2.4 TCP的可靠传输-超时重传机制
    - 7.4.2.5 TCP的可靠传输-选择确认SACK
    - 7.4.2.6 TCP的流量控制
    - 7.4.2.7 TCP的拥塞控制
    - 7.4.2.8 TCP的运输连接管理

## Chapter8 应用层

- 8.1 应用层概述
- 8.2 DHCP
- 8.3 DNS
  - 8.3.1 域名系统概述
  - 8.3.2 因特网的域名结构

8.3.3 域名服务器
8.4 E-mail
8.4.1 概述
8.4.2 简单邮件传送协议SMTP
8.4.3 电子邮件的信息格式
8.4.4 邮件读取协议
8.4.5 基于万维网的电子邮件
8.4.6 通用因特网邮件扩充MIME
8.5 万维网WWW
8.5.1 万维网概述
8.5.2 统一资源定位符URL
8.5.3 超文本传送协议HTTP
8.5.4 万维网的文档
8.6 文件传送协议FTP
8.6.1 FTP概述
8.6.2 FTP工作原理
8.6.3 FTP服务器的运行模式
8.7 TELNET
8.8 SNMP
8.8.1 网络管理的基本概念
8.8.2 管理信息结构SMI
8.8.3 管理信息库MIB
8.8.4 SNMP的协议数据单元和报文

## Chapter1 概述

- 分组交换网：以存储转发方式传输分组的通信子网
- 计算机通信网的形式化描述：
  - 计算机通信网 = **计算机主机, 通信子网, 协议** | 自治的主机按协议经通信子网互连
  - **三要素**：
    1. 必须有两台(或两台以上)自治的计算机互相连接起来才能构成网络
    2. 需要一条通道(或通信子系统)才能把两台或两台以上的计算机连接起来
    3. 计算机之间要交换信息，彼此之间就需要有某些规定和约定
- 计算机通信网的组成：
  - **通信子网**：负责数据的无差错和有序传递，其处理功能包括差错控制、流量控制、路由选择、网络互连等。
  - **资源子网**：是计算机通信的本地系统环境，包括主机、终端和应用程序等，主要功能是用户资源配置、数据的处理和管理、软件和硬件共享以及负载均衡等。
  - 计算机通信网就是一个由通信子网承载的、传输和共享资源子网的各类信息的系统。
- 计算机网络的分类
  - 按传输技术分：广播式网络、点到点网络
  - 按规模分：个域网(PAN)、局域网(LAN)、城域网(MAN)、广域网(WAN)、互联网(Internet)
  - 按传输介质分：有线网、无线网
  - 按拓扑结构分：总线、环形、网状(骨干网)、星形(中间节点)、不规则形、树形
  - 按使用范围分：专用网、公共网

- 通信协议

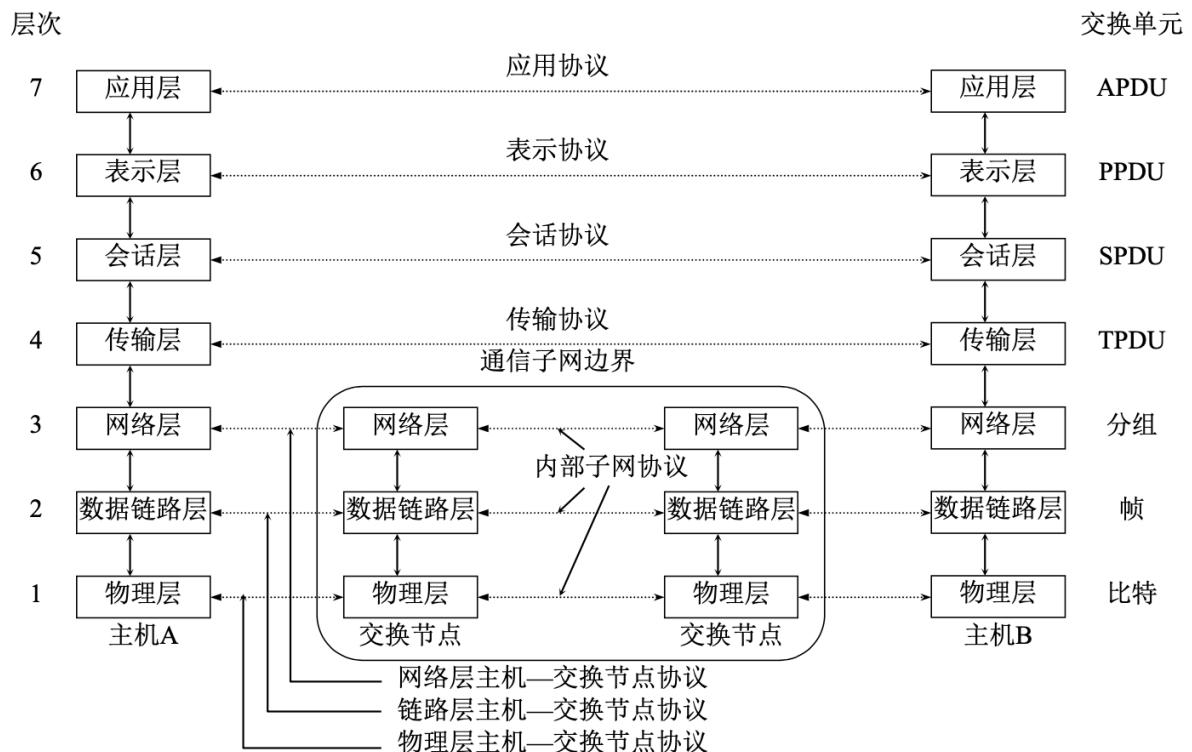
- 定义：相互通信的双方(或多方)对如何进行信息交换所必须遵守的一整套规则。
- 作用：完成计算机之间有序的信息交换。
- **三要素**

1. 语法：语法是用户数据与控制信息的结构与格式，以及数据出现顺序的意义。
2. 语义：用于解释比特流的每一部分的意义。
3. 时序：事件实现顺序的详细说明。

- 分层

- 基本思想：
  - 把整套的协议体系分成一些层，下一层对它的上一层提供服务
  - 每一层本身的功能与下层提供的服务叠加到一起，从而使最高层为用户提供一组完整的服务，以便实现通信或分布应用
- 基本原则：
  - 定义每一层向上一层提供的服务，以保证每层功能的相互独立，但不规定如何完成这些服务

- OSI/RM的体系结构



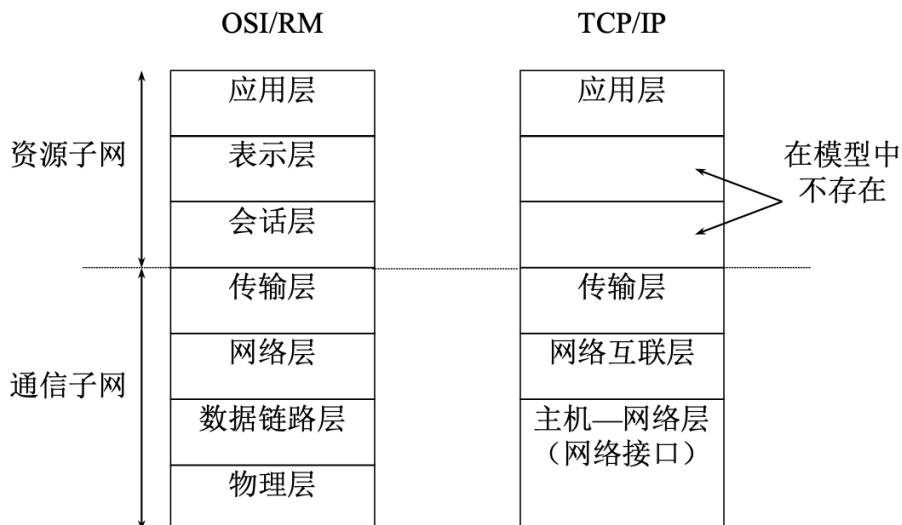
- **低三层属于通信子网：**涉及为用户间提供透明连接，操作主要以每条链路为基础，在节点间的各条数据链路上进行通信。由网络层来控制各条链路上的通信，但要依赖于其他节点的协调操作。
- **高三层属于资源子网：**主要涉及保证信息以正确可理解形式传送。
- **传输层是高三层和低三层之间的接口：**它是第一个端到端的层次，保证透明的端到端连接，满足用户的服务质量(QoS)要求，并向高三层提供合适的信息形式。

## OSI各层功能总结

层 次	功 能
7. 应用层	提供电子邮件、文件传输等用户服务
6. 表示层	转换数据格式，数据加密和解密
5. 会话层	通信同步，错误恢复和事务操作
4. 运输层	网络决策，实现分组和重新组装
3. 网络层	路由选择，计费信息管理
2. 数据链路层	错误检测和校正，组帧
1. 物理层	数据的物理传输

- TCP/IP协议模型

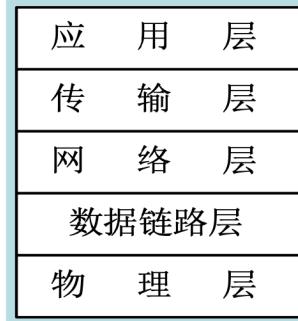
- 应用层、传输层、网际层、网络接口层
- TCP/IP模型与OSI/RM的对应
  - 都采用了层次结构的概念，在传输层中二者定义了相似的功能



- 各层的主要功能

TCP/IP模 型分层	主 要 功 能
主机-网 络层	定义了Internet与各种物理网络之间的网络接口
网络互联 层	负责相邻计算机之间（即点对点）通信，包括处理来自传输层的发送分组请求，检查并转发数据报，并处理与此相关的路径选择，流量控制及拥塞控制等问题。
传输层	提供可靠的点对点数据传输，确保源主机传送分组到达并正确到达目标主机。
应用层	提供各种网络服务，如SMTP, DNS, HTTP, SNMP等

- 5层的参考模型：



## Chapter2 协议分层原理

- 分层的原因：

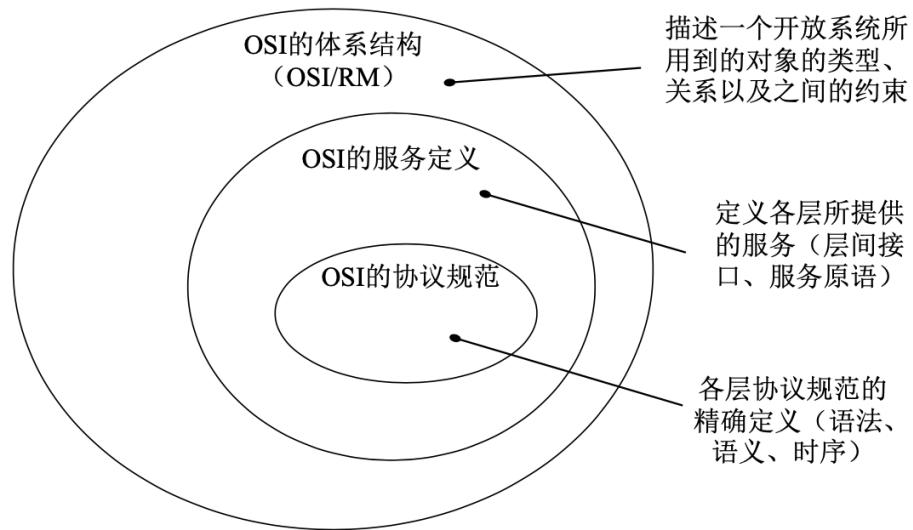
- 分而治之，每一层的目的都是向它的上一层提供一定的服务，而把如何实现这一服务的细节对上层加以屏蔽
- 协议分层能使问题简化，网络问题很多，将一个完整的问题分成许多较小的问题来解决
- 保证层次之间的独立

- 层次结构的特点：

- 具有一定的层次
- 层次之间呈单向依赖关系
- 上层起着隐藏下层细节和统一下层差异的作用

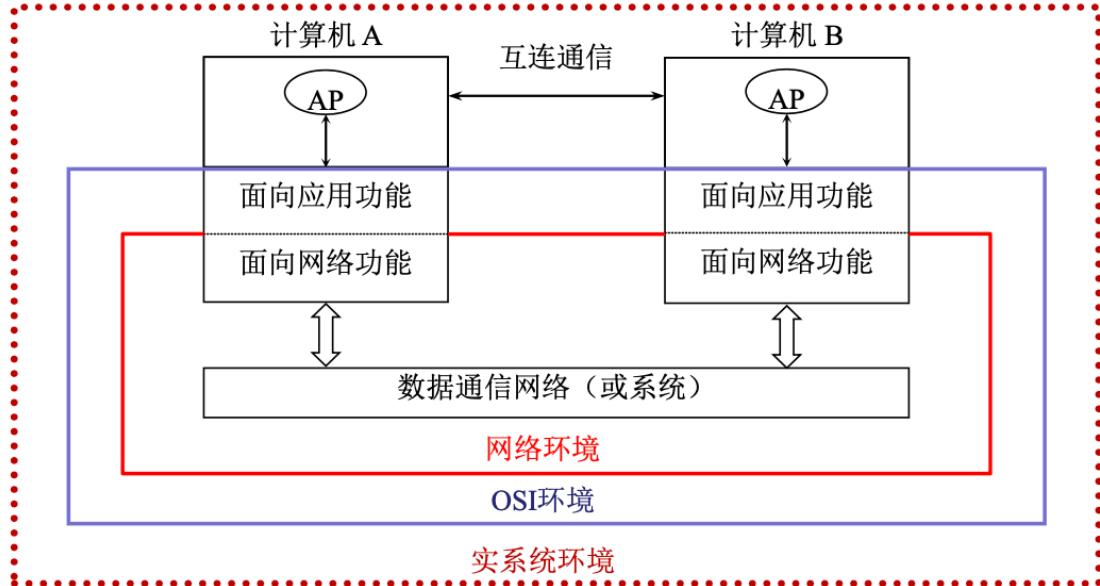
### 2.1 OSI/RM概述

- OSI/RM不规定实现它所描述的功能和和特定层的具体规范，不提供互连结构设施和协议的精确定义，不能作为评价和检测具体实现的一致性根据
- OSI/RM的三级抽象



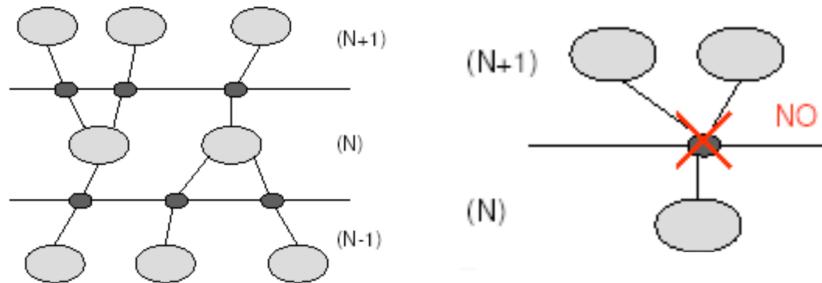
- 计算机的运行环境

- 实系统环境：建立在OSI环境之上，通信子网 + 资源子网



## 2.2 层的相关概念

- (N)层：用(N)层表示协议的某一层
  - (N+1)层表示相邻的高一层
  - (N-1)层表示其相邻的低一层
  - 对等的(N)子系统构成OSI/RM中的(N)层
- (N)实体：一个(N)子系统内部的一个活动要素(如一个TCP连接)
  - 一个(N)子系统可以看作由一个或多个(N)实体组成
  - 一层中的实体代表了该层在执行某功能时的分布处理能力，在一个单独的开放系统中，所有层的实体代表了该系统的互连能力，及能被其它开放系统所看到的处理能力
  - 位于不同子系统中的同一层实体称为对等实体
- (N)协议：对等的(N)实体间交互的规则与格式的集合，规定了(N)实体如何利用(N-1)服务并协同工作以完成(N)功能
- 接口：每一相邻层间有一个接口，该接口定义下层向上层提供的原语操作和服务——服务访问点SAP
- 虚拟通信(水平通信)：两个开放系统中的对等实体之间的信息交换——水平方向
- 实信息流(垂直通信)：对等实体的通信实际上依赖于本地系统中的(N-1)实体实现，实际的信息流是在同一系统的上下层之间发生——垂直方向
- (N)服务：定义为(N)层及其下面各层的综合能力，并在 (N)层和(N+1)层的分界面上提供给(N+1)实体
  - (N)实体实现的服务为(N+1)层所使用——服务提供者
  - (N)层可利用(N-1)服务来提供它自己的服务——服务用户
- 服务访问点：(N)SAP是(N+1)层可以访问(N)层服务的地方
  - 规则
    - 一个(N)SAP只能被一个N实体所使用，并且只能被一个(N+1)实体所使用
    - 一个(N)实体可以使用多个(N)SAP，而且一个(N+1)实体也可以使用多个(N)SAP



- **服务与协议的关系**

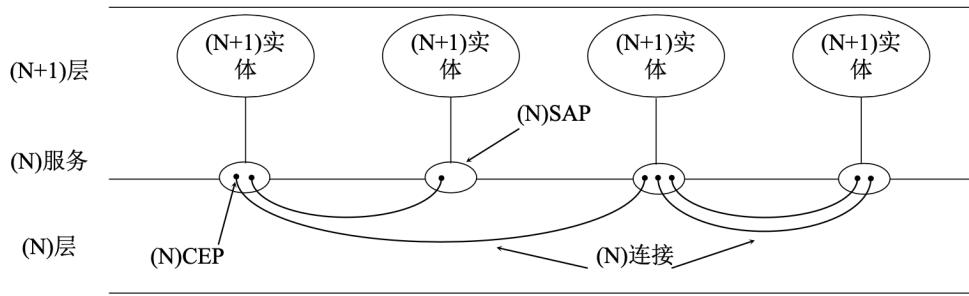
- 服务(上下关系)
  - 服务是各层向它的上层提供的一组原语(操作)
  - 服务定义了该层能代表它的用户完成的操作
  - 服务只与两层之间的接口有关
- 协议(水平关系)
  - 协议是一组规则
  - 决定同层对等实体交换帧、包和报文的格式和意义
  - 实体用协议来实现它们的服务定义
- 服务和协议是完全分离的

- 服务分类:

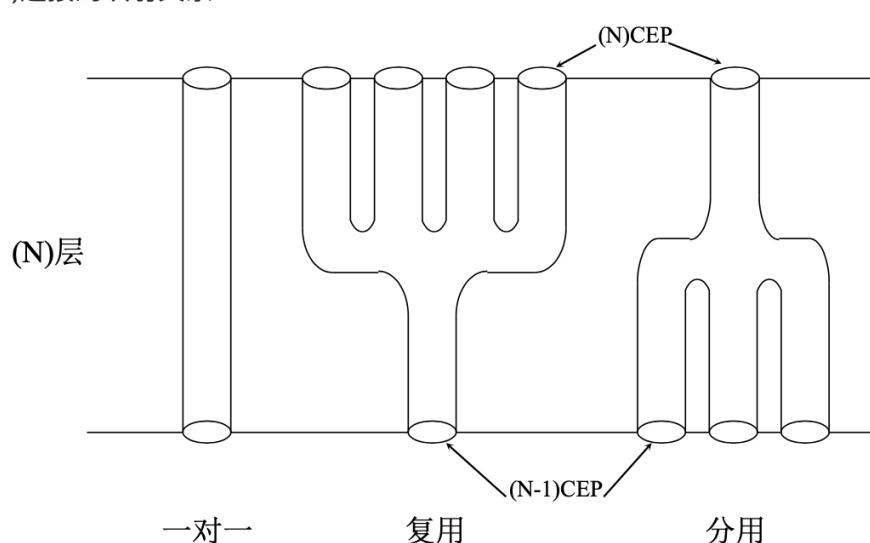
- **面向连接服务**
  - 具有连接建立、数据传输和连接释放这三个阶段
  - 在数据交换之前，必须先建立连接
  - 数据交换结束后，必须终止这个连接
  - 传送数据时，按顺序传送
  - 文件传输等服务比较适合于面向连接的服务
- **无连接服务**
  - 两个实体之间的通信不需要先建立好一个连接
  - 它不需要通信的两个实体同时是活跃的
  - 灵活方便，比较迅速
  - 不能防止报文的丢失、丢失或失序，不可靠服务
- 标识符:
  - 实体的局部名：实体在层内被唯一标识的名称
  - 实体的全局名则：实体在OSI环境内的唯一名称，它包括一个层名称域名和一个局部名两部分
  - (N)服务访问点的标识符称为**(N)地址**
    - 使用(N)地址可以唯一地标识与某个(N+1)实体相连的一个特定(N)SAP，当该(N+1)实体与这个(N)SAP脱离时，该(N)地址就不再提供对该(N+1)实体的访问
- **(N)地址映射：**
  - 负责建立(N)地址和与(N)实体有关联的(N-1)地址之间的映射 ((N)SAP和(N-1)SAP之间的对应关系)
  - 在一层内可存在两种(N)地址映射，即层次映射和表格映射

## 2.3 连接及其操作

- (N)连接：两个或多个对等(N+1)实体之间交换数据时，必须在(N)层内使用(N)协议在(N+1)实体之间建立联系
  - 为这些对等(N+1)实体提供服务的(N)实体名
  - 连接同一系统内建立服务关系的(N+1)实体与(N)实体间的(N)SAP
  - (N+1)实体要求(N)实体提供的(N)服务的服务质量
  - 与(N)服务有关的其它属性，如流量控制的窗口大小，用户数据长度等
- (N)连接形式
  - 点对点连接：实体之间仅有一个连接
  - 多端点连接：实体之间可有多个连接(如广播通信)
  - 同一对SAP之间也可以存在多条连接，以连接端点(CEP)标识区分



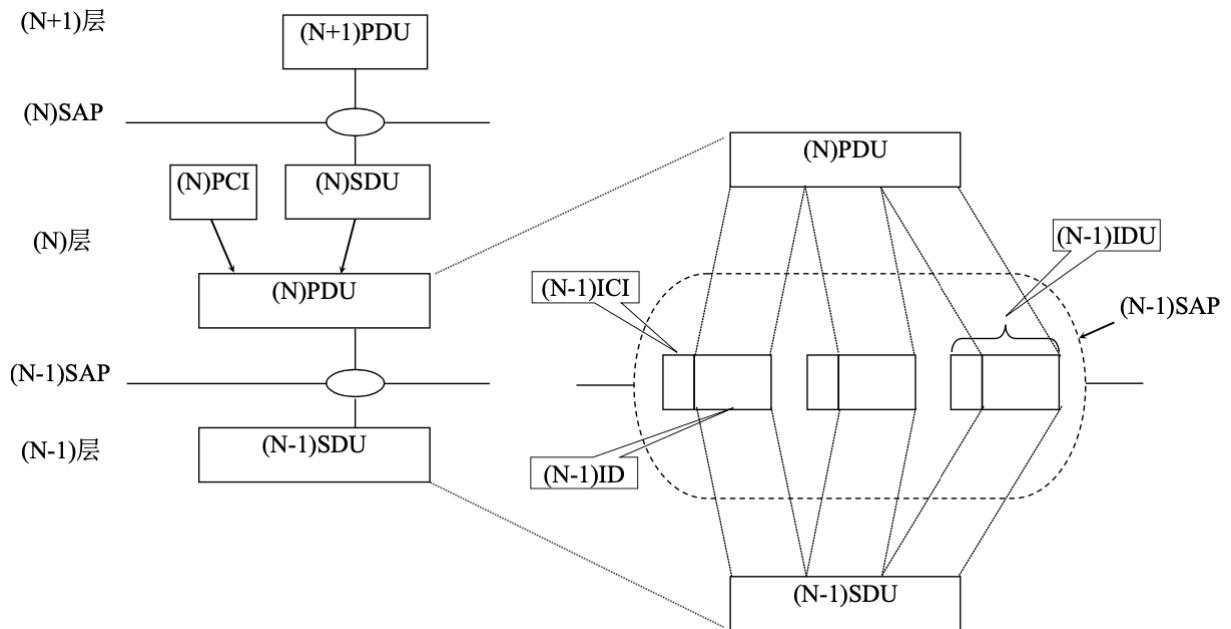
- (N)层对等实体间建立一条(N)连接的条件
  - 在提供支持的(N)实体之间有一条(N-1)连接
  - 两边(N)实体应处于能交换连接建立协议的状态
- 释放(N)连接的三种情况
  - 常规释放：对等(N+1)实体间完成数据传送后，任何一方(N+1)实体均可发起
  - 有序释放：持有令牌的(N+1)实体才有权发起释放(N)连接
  - 异常释放：当发生异常情况时，不能再进行(N)连接上的数据传输
    - 服务用户(N+1)实体发起异常释放
    - 服务提供者(N)实体发起异常释放
- 连接的复用与分用
  - (N)连接与(N-1)连接的映射关系



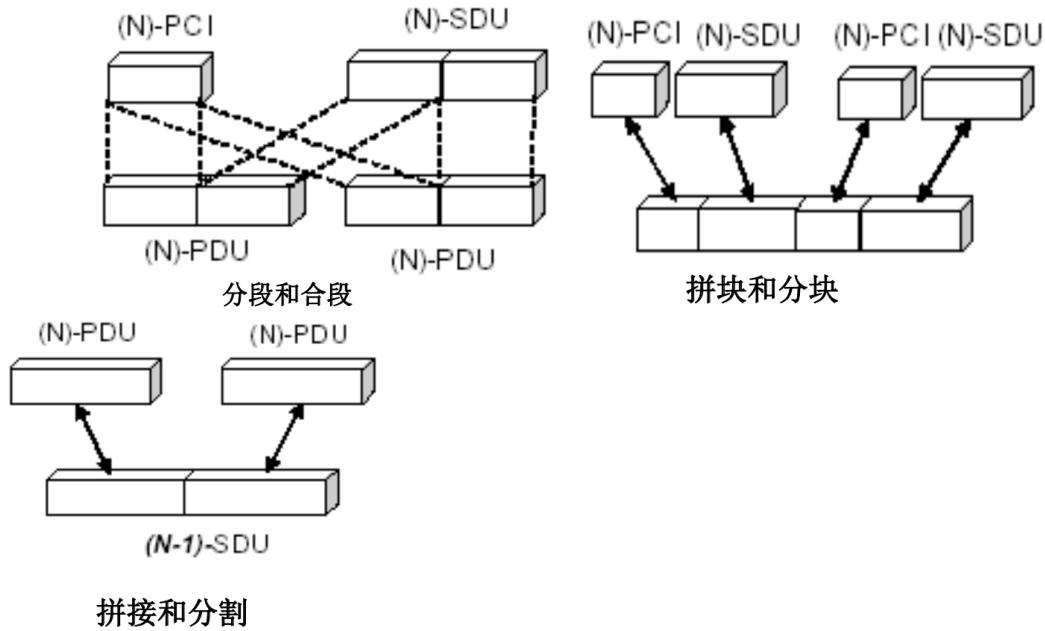
- 复用:
  - 提高使用效率，节省费用
  - 在只有一条(N-1)连接时，提供多条(N)连接
- 分用:
  - 在有多条(N-1)连接可用时，提高可靠性
  - 提高吞吐量和其他服务质量
  - 利用廉价的多条(N-1)连接提高经济效益

## 2.4 数据传送

- 信息传输的两种情况：
  - 水平传输：在已经建立连接的对等(N)实体之间传送(受(N)协议的控制)
  - 垂直传输：在同一开放系统相邻子系统的实体之间传送，即连接到同一个(N-1)SAP的(N-1)实体与(N)实体之间的传送
- 协议数据单元的组成：
  - (N)PCI：协议控制信息，协调对等实体间的协同工作，一般作为协议头加在用户数据之前，组成PDU
  - (N)PDU：跨过网络传给对等实体的信息，PCI+UD=PDU
  - (N-1)ICI：协调相邻层的(N)实体与(N-1)实体的联合操作，加在接口数据前面，组成IDU
  - (N-1)IDU：(N)层实体通过SAP传递给第(N-1)层实体的信息，ICI+ID=IDU
  - (N-1)SDU(UD)：(N)实体与(N-1)实体之间传送的信息，是(N-1)接口数据的总和
  - PDU、IDU和SDU三者的关系



- 数据单元的分与合
  - 不同层的数据单元长度受到该层协议的控制，不同层次数据的长度未必都一致，需要进行数据单元的分段和拼块

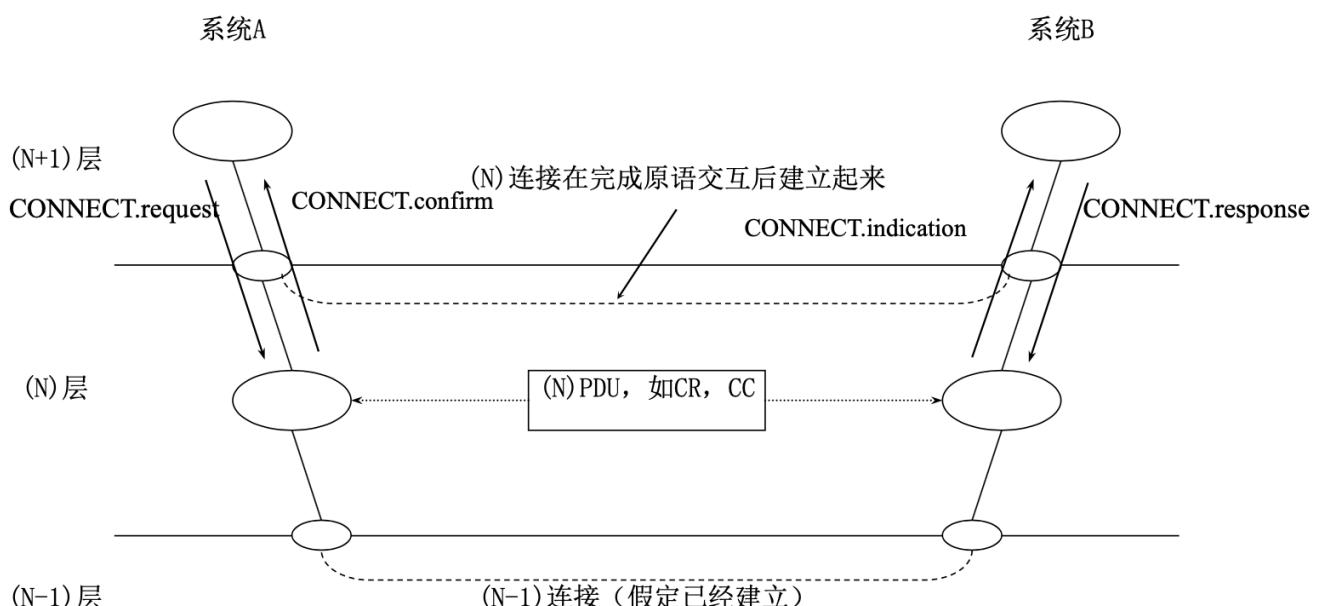


## 2.5 服务原语

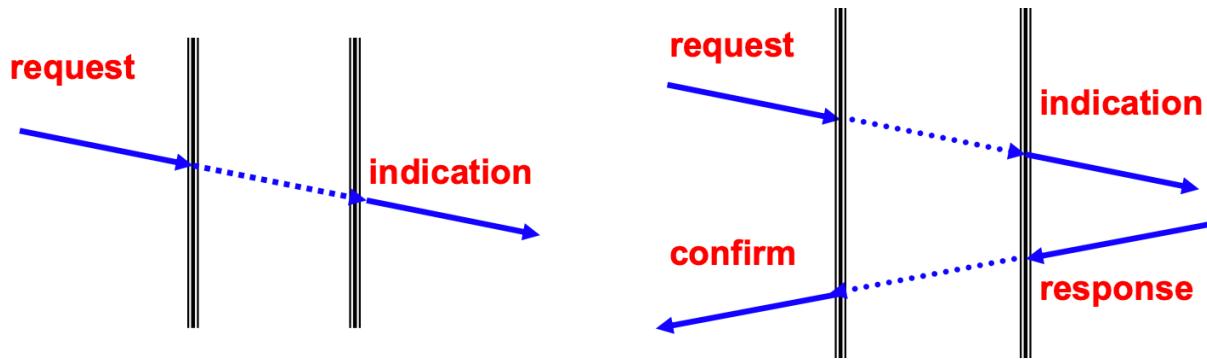
- 服务原语：是指服务用户与服务提供者之间进行交互时所要交换的一些必要信息
- **四种服务原语类型：**

原语类型	英文对照	含义
请求	request	一个实体希望得到完成某些操作的服务
指示	indication	通知一个实体，有某个事件发生
响应	response	一个实体对某个事件作出响应
证实	confirm	返回对先前请求的响应

- 建立(N)连接的原语交互



- 无确认服务与确认服务



## Chapter3 数字通信基础

### 3.1 数据传输相关概念

#### 3.1.1 数据、信号和码元

- 信号：数据的表示形式，使数据能以适当的形式在介质上传输。
- 码元：在时域的波形表示数字信号时，代表不同离散数值的基本波形。

#### 3.1.2 波特率和比特率

- 波特率：每秒钟信号变化的次数(每秒采样的次数)，也称为码元速率，单位是Hz。
- 比特率：每秒钟传输的数据的位数，即数据传输速率，单位是bps。
- 两者关系：如信号分为V级，则比特率 =  $\log_2 V \times$  波特率。

#### 3.1.3 数据传输

- 模拟传输：不关心内容，只关心尽量减少信号的衰减和噪声，长距离传输时，采用信号放大器放大被衰减的信号。
- 数字传输：关心信号的内容，长距离传输时，采用转发器(先恢复出数字数据，再生成外出信号)。
- 长距离传输时，通常采用的是数字传输。

#### 3.1.4 信道及其参数

- 传输介质与信道
  - 传输介质：传输信号的物理载体。
  - 信道：提供了传输某种信号所需的带宽，着重体现介质的逻辑特性。
  - 一根传输介质可能同时提供多个信道，一个信道也可能由多根传输介质级联而成。
- 信道的工作方式：
  - 单工通信——只能有一个方向的通信而没有反方向的交互，如广播、电视。
  - 半双工通信——通信的双方都可以发送信息，但不能同时发送。
  - 全双工通信——通信的双方可以同时发送和接收信息，需要两条信道。
- 数据的传输速率：
  - 码元传输的速率越高，或信号传输的距离越远，则失真越严重。
  - Nyquist定理**：在无噪声时，为了避免码间串扰，码元的传输速率的上限值
    - 在无噪声信道中，当带宽为W Hz时
      - 码元传输速率的上限值 =  $2W$  Baud
      - 数据传输速率 =  $2W \log_2 V$  b/s
  - 香农定理**：带宽受限且有高斯白噪声干扰的信道的极限、无差错的信息传输速率

- 在噪声信道中，当带宽为W Hz，信噪比为S/N（信号平均功率/噪声功率）

- 最大数据传输速率( $b/s$ ) =  $W \log_2(1 + S/N)$
- S/N：信噪比，若用分贝(dB)表示，则信噪比(dB) =  $10 \log_{10} S/N$

在噪声信道中，带宽为3500Hz，信噪比为30dB，则最大数据传输速率  
 $(b/s) = W \log_2(1 + S/N) = 3500 \log_2(1 + 1000) \approx 35000(b/s)$

最大数据传输速率为35k bps，这是传输速率极限，实际上不可能

### 3.1.5 基带传输和宽带传输

- 基带传输：信号源产生的原始电信号称为基带信号，即将数字数据0、1直接用两种不同的电压表示，然后送到线路上去传输。（距离短）
- 宽带传输：将基带信号进行调制后形成模拟信号，然后采用频分复用技术实现宽带传输。分为多个信道，模拟和数字数据可混合使用，需解决数据双向传输的问题。（距离长）

### 3.1.6 异步通信和同步通信

- 同步：要求通信的收发双方在时间基准上保持一致
  - 位同步：接收端必须对它所接收的数据流中每一位进行正确的采样
  - 位同步方式的分类：根据通信规程所定义的位同步方式，分为异步通信和同步通信
- 异步通信：发送方和接收方的采样时钟不是同一个
  - 以字符为单位的数据传输
- 每个字符：附加1位起始位和1位停止位作标志，以标记字符的开始和结束，还要附加1位奇偶校验位
- 同步通信：指发送方和接收方的采样时钟是同一个
  - 发送方在发送数据的编码中包含时钟，而接收方则从数据流中提取时钟用以采样
  - 面向字符的同步通信：数据块由字符组成，数据块前加一个或两个同步字符SYN用于数据块的同步，无需起始位和停止位
  - 面向bit流的同步通信：每个数据块的头部和尾部用一个或多个特殊的比特序列(如01111110)来标记数据块的开始和结束

### 3.1.7 串行通信和并行通信

- 串行通信：数据按位为单位，以时间为序。
- 并行通信：据按字符为单位，以时间为序。

## 3.2 信号编码技术

### 3.2.1 数字数据的模拟信号调制

- 数字数据在模拟信道上传输，将数字数据调制成模拟信号（调制解调器）进行传输，通常有四种调制方式：
  - 调幅ASK：用载波的两个不同的振幅来表示两个二进制值
  - 调频FSK：用载波附近的两个不同的频率来表示两个二进制值
  - 调相PSK：用载波的初始相位来表示两个二进制值
  - 正交调相QPSK(星座模型)

### 3.2.2 数字数据的数字信号编码

- 数字数据在数字信道上传输，最简单的方法是用两个不同的电压信号值来表示两个二进制的数字数据值0和1
- 数字信号编码方式：不归零编码、曼切斯特编码、差分曼切斯特编码、4B/5B编码。
  - 不归零编码NRZ：正电平表示1，零电平表示0，在表示完一个码元后，电平无需回到零。**不能携带时钟信号**，存在同步问题，**无法表示没有数据传输**。用不归零制编码时，一个时钟周期可表示一个bit，一次采样得到一个bit——效率最高的编码。
  - 曼切斯特编码：位中间有电平跳变，**能携带时钟信号**，且**可表示没有数据传输**。一个时钟周期只可表示一个bit，并且必须通过两次采样才能得到一个bit。
  - 差分曼切斯特编码：特性与曼切斯特编码相同，但抗干扰性能强。
  - 4B/5B编码：不归零制编码的一种变种，每4个bit成一个组合，并对应为5个bit的编码。为了让4B/5B编码后的码流中有足够多的跳变，需要编码后的码流中有尽量多的“1”和尽量少的“0”以便接收端提取出时钟信号。
- 四种编码方式的比较：
  - 不归零制编码：编码密度最高，接收端一次采样可得到一个bit，波特率等于比特率，但不能携带时钟。
  - (差分)曼切斯特编码：编码密度最低，接收端二次采样才可得到一个bit，波特率是比特率的两倍，但每个bit中都有信号跳变，即携带了时钟。
  - 4B/5B编码：编码密度略低于不归零制编码，但高于曼切斯特编码，波特率是比特率的1.25倍，然而在接收端能提取时钟。

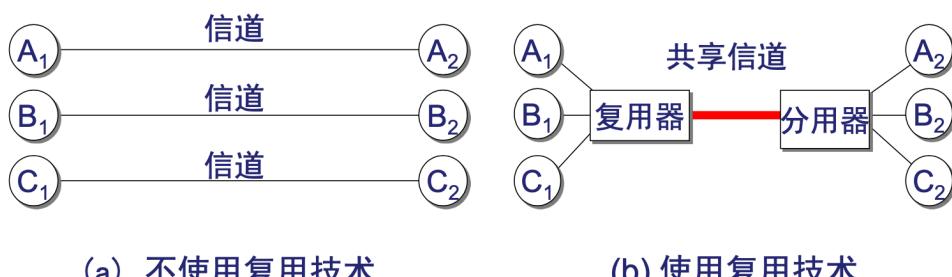
### 3.2.3 模拟数据的数字信号编码

- 模拟数据在数字信道上传输
  - 采用脉冲编码调制PCM技术，以采样定理为基础
  - **采样定理**：如果在规定的时间间隔内，以信号最高频率的二倍或二倍以上的速率对该信号进行采样，则这些采样值中包含了全部原始信号信息

### 3.2.4 模拟数据的模拟信号调制

- 模拟数据在模拟信道上传输
  - 调幅AM：载波的**振幅**随模拟数据的数值变化。
  - 调频FM：载波的**频率**随模拟数据的数值变化。
  - 调相PM：载波的**初始相位**随模拟数据的数值变化。

## 3.3 信道复用



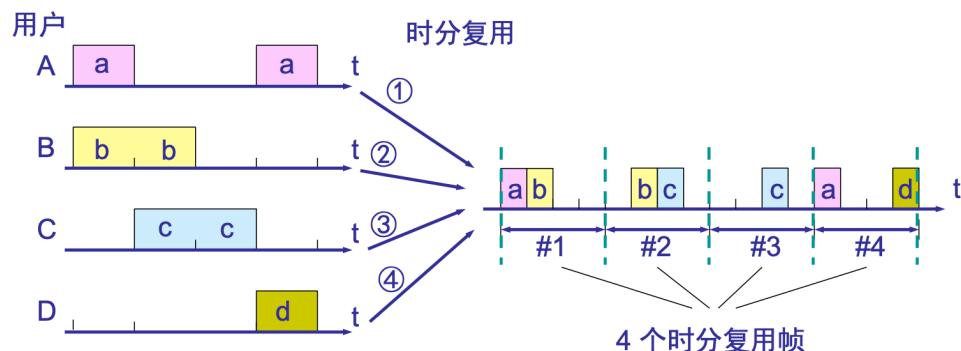
### 3.3.1 频分复用FDM

- 为每个用户分配固定的频带，在通信过程中自始至终都占用这个频带。

### 3.3.2 时分复用TDM(同步)

- 时分复用：将时间划分为一段段等长的**时分复用帧**(TDM帧)，每一个时分复用的用户在每一个TDM帧中占用固定序号的时隙。
- 每一个用户所占用的时隙是**周期性地出现**(其周期就是TDM帧的长度)。
- 时分复用的所有用户在不同的时间占用**同样的频带宽度**。

例：时分复用帧的形成过程

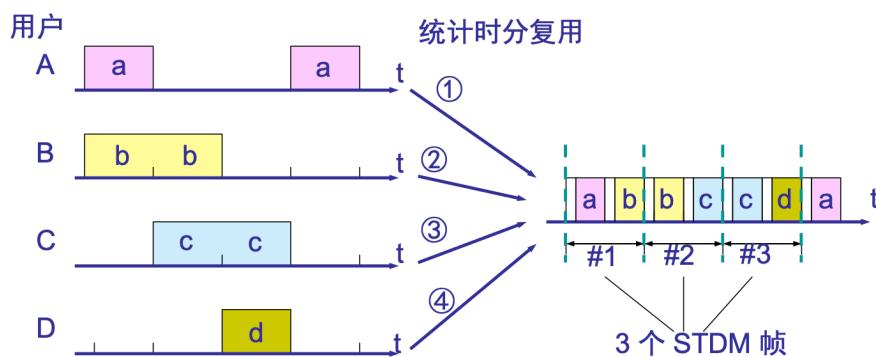


- 线路资源的浪费**：由于计算机数据的突发性质，用户对分配到的子信道的利用率一般不高。

### 3.3.3 统计时分复用STDM(异步)

- 集中器：实现信道的统计时分复用。
- 使用**STDM帧**传送复用的数据——每一帧的时隙数**小于**连接在集中器上的用户数
  - 各用户将数据发往集中器中的对应输入缓存。
  - 集中器按顺序依次扫描输入缓存，将缓存中的数据放入STDM帧。
  - STDM帧的数据放满，集中器将其发送出去。

例：STDM帧的形成过程



- STDM帧的特点：动态分配帧内时隙，每个时隙带有用户的**地址信息**。

### 3.3.4 波分复用WDM

- 波分复用就是光的频分复用 ( $\lambda f = C$ )

### 3.3.5 码分复用CDMA

- 码分多址CDMA，各用户在相同的时间使用相同的频带——各用户使用经过特殊挑选的不同码型，因此彼此不会造成干扰，有很强的抗干扰能力
- 每一个比特时间划分为  $m$  个短的间隔，称为码片
- 码片序列
  - 每个站被指派一个唯一的  $m$  bit 码片序列
    - 如发送比特 1，则发送自己的  $m$  bit 码片序列
    - 如发送比特 0，则发送该码片序列的二进制反码

例如，S 站的 8 bit 码片序列是 00011011

发送比特 1 时，就发送序列 00011011

发送比特 0 时，就发送序列 11100100

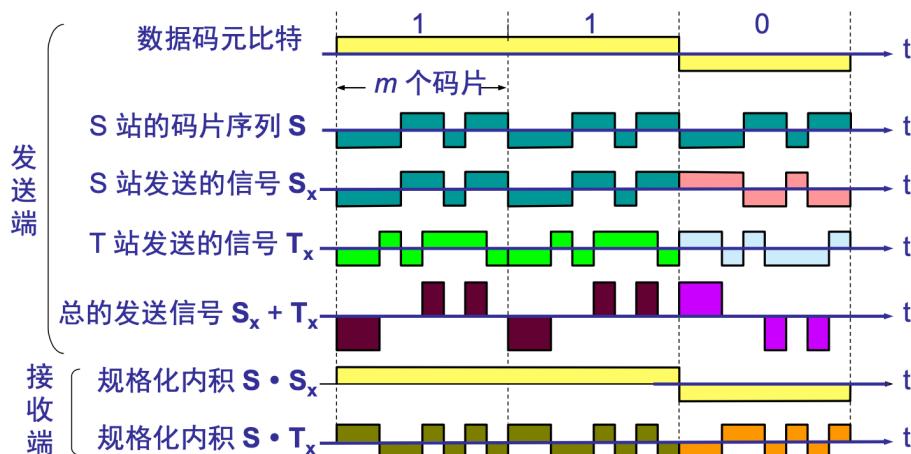
S 站的码片序列(码片向量):  $(-1 -1 -1 +1 +1 -1 +1 +1)$

- CDMA 的特点：每个站分配的码片序列各不相同，并且还必须互相正交
- 码片序列的正交关系：

- 令向量  $S$  表示站 S 的码片向量，令  $T$  表示其他任何站的码片向量。

$$S \cdot T = \frac{1}{m} \sum_{i=1}^m S_i T_i = 0$$

- 任何一个码片向量和自身的规格化内积是 1，和反码的规格化内积是 -1。
- CDMA 的工作原理：



### 3.3.6 正交频分复用OFDM

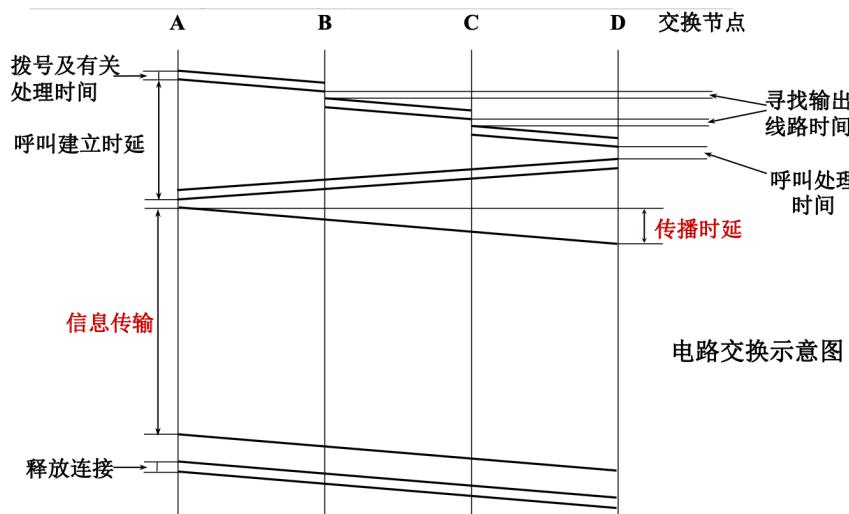
- 多载波调制方式，通过减小和消除码间串扰的影响来克服信道的频率选择性衰落
- 基本原理：将信号分割为  $N$  个子信号，然后用  $N$  个子信号分别调制  $N$  个相互正交的子载波。由于子载波的频谱相互重叠，因而可以得到较高的频谱效率

## 3.4 数据交换技术

- **交换**: 网络节点以某种转接方式来实现数据通路接续的技术。
- 通信网可能采用的交换方式:
  - 电路交换——**提供实时语音业务。**
  - 分组交换——**提供非实时数据业务。**

### 3.4.1 电路交换(电话)

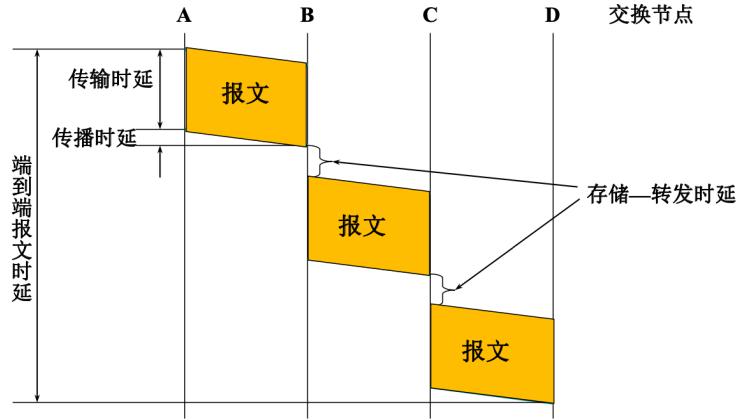
- 电路交换: 交换系统为通信的双方寻找并建立一条全程物理通路, 以供双方传输信息, 直到信息交换结束



- 电路交换的优缺点:
  - 优点: 数据传输时延小、对用户透明、吞吐量高
  - 缺点:
    - 必须有一个呼叫建立的过程
    - 占用固定的带宽, 信道带宽利用率低
    - 难以实施差错控制措施
    - 难以适应计算机和各种终端传输速率不一致等情况
    - **不适合具有突发性的计算机数据传输**
- 电路交换的实现结构:
  - **空分交换**: 指输入链路按空间分布来安排所需要的输出链路。
  - **时分交换**: 指输入链路按时间顺序来安排所需要的输出链路。

### 3.4.2 报文交换(电报)

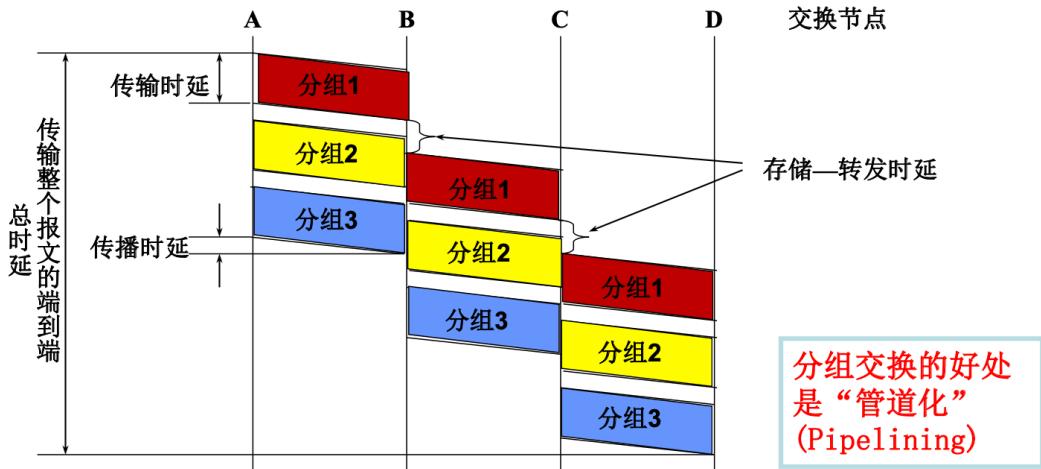
- 无论数据传输过程要跨越多少个交换局(通常是路由器), 只要下一站不忙, 该数据即送至下一站, 线路的利用率较高
- 数据的传输不需建立连接, 数据的传输是一站一站往下送, 所以数据中必须包含目的地址, 并采用**存储—转发机制**
- 在数据传输过程中, 除了信号传播的延时之外, 还有存储和转发的延时
- 每个中间站点都必须有**足够大的缓存**, 报文大小不一, 通常设置在硬盘中
- **不适合于实时性要求高的业务**
- **报文交换示意图:**



- 报文交换的端到端时延：传输时延、传播时延、存储—转发时延

### 3.4.3 分组交换

- 将报文分为若干个定长的分组，每个分组为一个子报文
- 每个分组中必须包含目的地址，并采用存储—转发机制，线路的利用率较高
- 每个中间站点必须有缓存，由于分组大小固定，通常在内存中设置
- 接收分组和发送分组的顺序可能不一致，可能还需要重组
- 分组交换示意图：



- 分组的组成：
  - 分组=分组头+信息包
  - 用户数据被切分成多个信息包，每个包加上分组头，形成分组
- 分组最佳长度：**
  - 设报文的总长度为  $M$ (bit)，令  $n_h$ (bit)为附加到每个分组的开销，用  $K_{max}$  表示包括附加开销在内的最大分组长度。
  - 以分组方式发送一个报文所必须传送的总比特数为

$$N_{bits} = M + \lceil M / (K_{max} - n_h) \rceil n_h$$

- 传送一个报文所需要的总时间  $T$  为第一个分组通过前面  $(j - 1)$  条链路的时间加上整个报文通过最后一条链路的时间 ( $E[\cdot]$  为均值)：

$$T = \frac{(j-1)K_{max} + M + \lceil M/(K_{max} - n_h) \rceil n_h}{R}$$

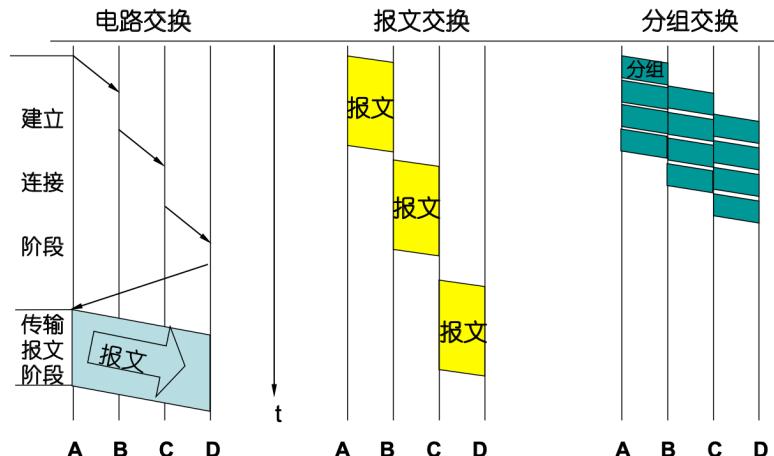
$$E[T] = \frac{(j-1)K_{max} + E[M] + E[\lceil M/(K_{max} - n_h) \rceil] n_h}{R}$$

- 将上式对  $K_{max}$  求导，并令导数等于零，于是得到分组最佳长度为：

$$K_{max}^{opt} \approx n_h + \sqrt{\frac{E[M]n_h}{j-1}}$$

- 目的节点收到分组后所作处理：
  - 卸掉分组头部和校验序列，得到分组数据，写入缓冲器
  - 根据分组编号检查构成一个完整报文的所有分组是否到齐
  - 若报文的所有分组到缓冲器，则进行报文重组；否则，等待其他分组到达
  - 报文组装完成后，通知用户接收此报文；同时向发送方返回一个确认信息
- 分组交换两种交换机制：
  - 数据报(无连接方式)：每个分组独立处理，独自选择在网络中传输的路由
  - 虚电路(面向连接方式)：首先要建立传输路径，所有分组均在这个连接上进行传送，帧顺序和路径都是确定的

#### 3.4.4 三种交换技术的比较



- 电路交换：
  - 在数据传输前，必须建立端到端的连接
  - 一旦某个节点故障，必须重新建立连接
  - 连接建立后，数据的传输没有额外的延时
  - 数据中不必包含地址域
  - 数据按序传输，但信道的使用率较低
  - 适合长时间传输大批量的数据，如流数据
- 报文交换：
  - 在数据传输前，不必建立端到端的连接；只要下一个节点空闲，即可传输；信道的使用率较高；数据中必须包含地址域；数据的传输采用存储转发，延时不可估计
- 分组交换
  - 在数据传输前，不必建立端到端的连接；只要下一个节点空闲，即可传输；信道的使用率高；数据中必须包含地址域；数据的传输采用存储转发，延时不可估计

- 时延大大低于报文交换
- 接收到的分组不一定按序，可能还需重组
- 适合传输文本型数据

### 3.4.5 虚电路交换

- 虚电路：将电路交换的概念引入到分组传输——**面向连接的分组交换，快速分组交换**
- 虚电路连接的建立：传输方发起连接请求，中间节点根据路径信息建立**交换表**，在交换表内，节点为连接分配一个**虚电路号**，并与输出端口号相关联，表示用户信息从该端口输入，立即从相关联的输出端口输出到下一节点
- 虚电路连接的传输：**分组中没有目的地址，只有虚电路号**，接收分组时只检查其头部，一旦得到其虚电路号，则立即查交换表，转发至适当的端口，**非存储—转发**

## Chapter4 物理层

### 4.1 物理层的基本概念

- 物理层的功能：与传输媒体的接口，完成传输媒体上的信号与二进制数据间的转换，并定义了与传输媒体的接口特性

### 4.2 物理层下的传输媒介

#### 4.2.1 导向传输媒体

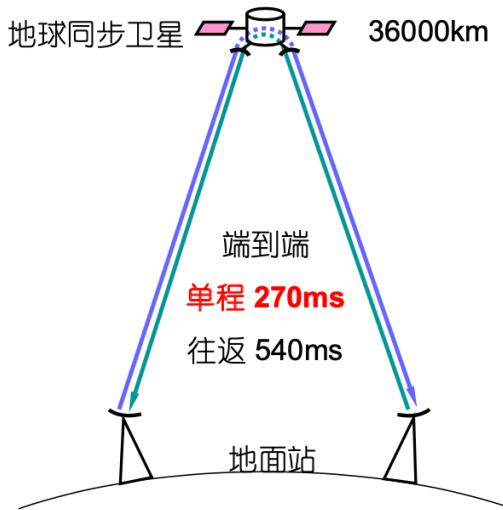
- 电磁波被导向沿着**固体媒体**(铜线或光纤)传播：双绞线、同轴电缆、光缆
- 双绞线：
  - 线间干扰较小、价格便宜、易于安装
  - 可传输模拟信号，也可传输数字信号
    - 在电话系统的最后一公里，用于传输模拟信号
- 同轴电缆：
  - $50\Omega$ 同轴电缆：用于数字信号传输，目前基本已被双绞线所替代
  - $75\Omega$ 同轴电缆：用于模拟信号传输，目前主要用于电视信号的传输
- 光缆：
  - 光传输系统：包括光发射器、光纤、光放大器、光检测器
  - 光纤类型：多模光纤(折射传输)、单模光纤(直线传输)
  - 光缆相对铜缆的特性：
    - 带宽高，距离远，损耗低，重量轻，无电磁干扰，防窃听
    - 端口设备价格高

#### 4.2.2 非导向传输媒体

- 无线传输：使用电磁波的频段很广，根据波长分成不同的波段，依次为无线电、微波、红外、可见光、紫外等
- 无线电传输（广播等）：
  - 全方向传播、主要靠电离层的反射
  - 多径效应——通信质量较差

- 微波传输

- 直线传播，能穿透电离层
- 地面微波接力通信：
  - 两个终端之间建立若干个中继站(天线)
- 卫星通信：
  - 以位于约36000公里高空的人造同步地球卫星作为中继器。



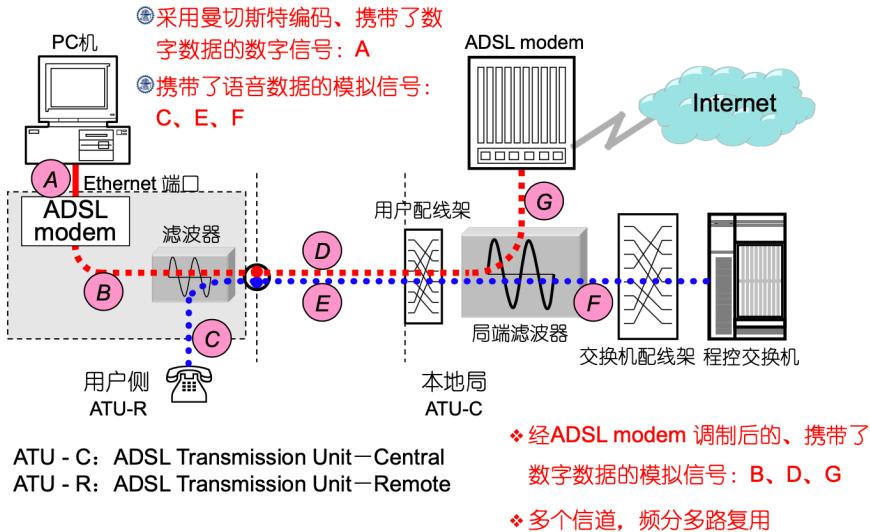
## 4.3 Internet的本地接入

### 4.3.1 拨号接入

- 通过电话线路访问远程服务器，使用调制解调器将数字信号转换成模拟信号
- 必须在某ISP注册成为合法用户
- 服务器端采用DHCP协议，为接入者分配一个临时的IP地址
- 数据传输需考虑的问题：
  - 任何传输介质在传输信号时都伴随着干扰(噪声)和衰减
  - 数字信号中包含着大量的高次谐波，所以基带信号不适合作长距离和高速的传输
  - 在电话系统中使用的是频分多路复用，人为地限制了每个信道的带宽为4kHz
- MODEM 调制解调器（猫）：
  - 调制方式：调幅、调频、调相
  - 使用调幅、调频、调相或其组合的调制技术，让载波携带数字数据

### 4.3.2 ADSL接入(非同步)

- DSL 数字用户线路：使用常规的电话线路，本地局端的滤波器将限制带宽为4 kHz，但本地回路通常采用的是3类UTP，其实际带宽远远大于4 kHz
- ADSL的接入模型：



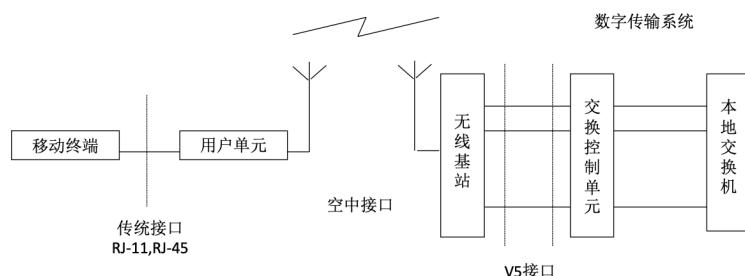
- DMT 离散多音调调制：将本地回路的可用带宽(约1.1MHz)分成256个4312.5Hz的独立信道

#### 4.3.3 基于社区电视系统

- 社区电视系统的特点：
  - 覆盖面很大，系统之间用光缆连接
  - 传播单向电视信号的一个共享系统
  - 用同轴电缆作为传输介质，带宽可达750 MHz
- 作为Internet 接入的可能性：
  - 数据的双向传输
  - 用户端的Cable MODEM

#### 4.3.4 无线接入

- 无线局域网：通常位于一个建筑物内
- 蜂窝移动网络：
  - 4G 蜂窝网络（有基站，用户接入为无线）
  - 无线接入系统的功能：用无线电波将用户设备与电话局交换机的用户接口连接起来——又称为无线本地环路系统



- 无线本地环路的结构：
  - 用户单元：包括收发信机，并提供一个面向基站的无线接口和面向用户的传统接口
  - 无线基站：由收发信设备与控制单元组成，提供一个面向交换机的标准网络接口和面向用户侧的空中接口
  - 无线交换控制单元：主要控制无线信道的分配，并提供与交换机接口，可连接并控制多个基站
- 多路传输方式——一个基站与若干用户连接：

- 频分多址：频带分为若干个频段，每个频段作为一路，将传输的信息调制在上面
- 时分多址：将载波频率分为若干时隙，为每个用户分配一个特定的时隙，来发送信号
- 码分多址：为每个用户分配一个码片序列，各码片序列之间具有正交性，允许所有用户在相同时间、相同频带内通信
- 卫星互联网

## Chapter5 数据链路层

### 5.1 数据链路层概述

#### 5.1.1 数据链路层的定义

- 物理链路：
  - 指有线或无线的传输通路。
  - 是一段无源的点到点的物理连接，中间没有任何交换节点。
- 数据链路：
  - 包括一条物理连接和为实现数据传输而在两端配置的硬件及其相关的通信协议。
  - 采用复用技术时，一条物理链路对应多条数据链路。
- 需要数据链路层的理由：
  - 传输数据的信道不可靠，可能产生差错。
  - 需要对数据的发送端进行流量控制。
  - 对于使用广播信道的通信，需要对信道进行分配，以保证通信的有序进行。

#### 5.1.2 数据链路层的功能

- 数据链路层的主要功能：在物理层提供的通信线路连接和比特流传输功能的基础上，负责在两个相邻节点间建立、维护和拆除链路，并通过差错控制、流量控制将不太可靠的物理链路改造成无差错的数据链路。
  - 在两个网络实体之间提供数据链路连接的建立、维持和释放。
  - 将需要传输的比特组装成数据链路层中的数据单元(帧)。
  - 控制帧在物理链路上的传输，主要是处理帧的同步、传输差错和调节帧的流速，以使发送速率与接收方的接收能力相匹配。

#### 5.1.3 数据链路层的服务类型

- 数据链路层借助于物理层为网络层提供服务。
- 数据链路层的服务类型：
  - 通过有无连接、有无确认来区分。
  - 无应答式无连接的服务：
    - 无应答：接收方在收到数据帧后，无需发回一个确认。无确认并非不可靠，其可靠性可由上层负责。
    - 无连接：在数据传输前后无需建立和释放连接。物理线路的连接并非意味着提供了有连接的服务。
    - 适用于：误码率很低的线路，错误恢复留给高层；实时业务；大部分局域网。
  - 应答式无连接的服务：
    - 应答式：每帧传输必须得到确认。在信号传播延时较大、线路状态不可靠的情况下是有效的。

- 适用于: **无线通信**
  - 如建立连接, 则信道使用率很低
  - 误码率相对较高, 确认是必要的
- 面向连接的服务:
  - 先建立连接, 然后使用, 最后释放连接。
  - 每帧的传输必须得到确认。
  - 适用于: **拨号电路**。

## 5.2 帧的构成

- 组帧方法:
  - 帧的定界: 识别一个**完整的帧**。
  - 帧的再同步: 一旦出现传输差错而导致前一个帧丢失时, 能识别后一个帧。

### 5.2.1 字符记数法

- 帧的长度用一个字节表示, 并作为帧的头部。
- 一旦帧长度计数被误读, 将无法再同步。



### 5.2.2 带字符填充的起始字符和终结字符法

- 用特殊的字符作为帧头和帧尾界符



由Flag标志的一个帧

(通常FLAG用ASCII字符7EH定义)

- **面向字符**的帧格式, 传输的数据都是字符, 帧中不允许出现帧界符标志。
- 帧的再同步: 接收方一旦丢失了一个FLAG, 只要继续搜索下一个FLAG, 就可重新确定帧边界。
- 字符填充法(透明传输): 类似于转义字符。

41 33 7E 5C 4B 0C	41 33 1B 7E 5C 4B 0C
41 33 1B 5C 4B 0C	41 33 1B 1B 5C 4B 0C
41 33 1B 7E 5C 4B 0C	41 33 1B 1B 1B 7E 5C 4B 0C
41 33 1B 1B 5C 4B 0C	41 33 1B 1B 1B 1B 5C 4B 0C

### 5.2.3 带位填充的起始/终结标志法

- 面向二进制位的帧格式，以特殊的位模式01111110作为帧标志，即一个帧的开始(前一个帧的结束)。
- 帧的再同步：如果一个帧标志没有被正确接收，则继续扫描，一旦扫描到01111110，即新的一帧从此开始。
- 位填充法(透明传输)：当帧中出现01111110，则在连续5个1后自动插入一个0，即变成01111101，接收方将自动删除第5个1后的0。

011011111111111110010
011011111011111011111010010
0110111111111111111110010

### 5.2.4 物理层编码违例法

- 采用冗余编码技术，如曼切斯特编码，对连续两个信号进行采样，可得到一个二进制位
  - 数据0：低-高电平对 
  - 数据1：高-低电平对 
- 高-高电平对和低-低电平对没有使用，如在编码中出现则称为编码违例，但这两种违例编码正好可用作帧界符，在令牌环网中使用编码违例格式。

## 5.3 差错控制

- 在传输时，数据中的一位或几位因噪声干扰而出错，通常接收方应能检错，甚至纠错。
- 可供选择的常用三种差错控制方法：
  - 前向差错控制（前向纠错(FEC)）：接收端检测到有错后，确定差错的具体位置，并自动加以纠正。
  - 反馈重发（自动重发请求(ARQ)）：接收端检测到有错后，通过反馈信号要求发送端重发原信息，直到接收端肯定确认为止。
  - 纠检混合的差错控制方式：接收端对少量的接收差错自动纠正，而超过纠正能力的差错则通过反馈重发的方法加以纠正。

### 5.3.1 差错检测和纠正

- 原理：通过信道编码，添加冗余度，从而检测或纠正传输中的错误。
- 码字的结构：
  - 分组码：将整个信息分成组，每一组的校验位由同一组的信息位进行线性变换得到，只对本组的信息位进行一致性校验。
  - 卷积码：在信息码中插入的每一校验位不仅实现本分组校验，还对前后分组中的信息位起校验作用。
- 纠错码和检错码
  - 纠错码：汉明码
  - 检错码：奇偶校验码、校验和、块校验码、循环冗余检错码CRC

### 5.3.1.1 汉明纠错码

- Hamming距离：

$$d(x, y) = \sum x[i] \oplus y[i]$$

- 编码的汉明距离：在一个码字集合中，任意两个码字之间汉明距离的最小值。
  - 编码的汉明距离越大，码组的抗干扰能力越强。
- 汉明码的格式：m个数据位外加r个纠错位，在 $2^i$  ( $i = 0, 1, 2, 3\dots$ ) 的位置放的是纠错位，m个数据位的次序不变。

如字符M的7位ASCII码为1101101，要加上4位纠错码0011(偶校验)，共11个bit。

11	10	9	8	7	6	5	4	3	2	1
A7	A6	A5	P4	A4	A3	A2	P3	A1	P2	P1
1	1	0	0	1	1	0	0	1	1	1

纠错位	纠错位的取值（再取偶/奇校验）
P4	=A7⊕A6⊕A5
P3	=A4⊕A3⊕A2
P2	=A7⊕A6⊕A4⊕A3⊕A1
P1	=A7⊕A5⊕A4⊕A2⊕A1

信息位	信息组位	组位展开	影响的纠错位
A7	11	=8+2+1	对P4、P2、P1有影响
A6	10	=8+2	对P4、P2有影响
A5	9	=8+1	对P4、P1有影响
A4	7	=4+2+1	对P3、P2、P1有影响
A3	6	=4+2	对P3、P2有影响
A2	5	=4+1	对P3、P1有影响
A1	3	=2+1	对P2、P1有影响

校验位	校验位计算表达式
S4	=A7⊕A6⊕A5⊕P4
S3	=A4⊕A3⊕A2⊕P3
S2	=A7⊕A6⊕A4⊕A3⊕A1⊕P2
S1	=A7⊕A5⊕A4⊕A2⊕A1⊕P1

- 计算结果S4 S3 S2 S1为0 0 0 0意味着接收正确
- 计算结果S4 S3 S2 S1为0 1 0 1意味着第5位出错

### 5.3.1.2 奇偶校验码

- 奇/偶检验：发现单个比特错
- 二维奇偶校验：检测奇数个独立、突发差错；如错码位于行列均出错的交叉点，可纠错

字符校验位	列校验位										块校验位
b8	0	1	0	1	1	1	1	0	0	0	0
b7	1	1	1	1	1	1	1	1	1	0	0
b6	0	1	0	1	1	1	0	1	1	1	1
b5	0	1	1	0	0	0	0	0	0	0	0
b4	0	0	0	1	0	1	0	1	1	0	0
b3	1	1	0	0	0	1	*1	0	0	1	1
b2	1	0	1	0	0	1	1	1	0	0	1
b1	1	1	1	0	1	0	1	0	1	1	1

### 5.3.1.3 校验和

- 算法简单，但检错强度较弱
- 发送端：将发送的数据看成是二进制整数序列，并划分成一段段规定的长度，计算它们的和，如计算和时有进位，则将进位加到最后的校验和中，将校验和与数据一起发送。
- 接收端：重新计算校验和，并与接收到的原校验和比较。

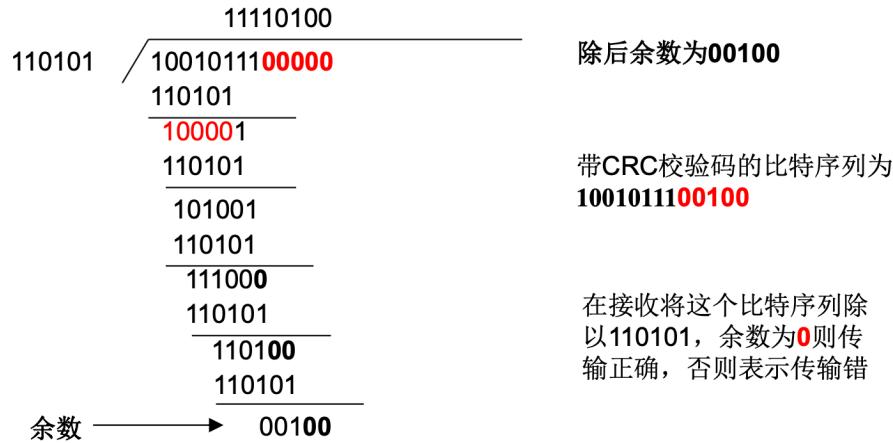
### 5.3.1.4 块校验码BCC

- 简单常用，但检错的强度较弱，如在同一列上有偶数位错，则不能检测
  - 如面向字符，每个字符进行奇偶校验，然后把所有的字符(连同奇偶位)进行异或运算，运算结果即为其块校验码。
  - 通常发送端在发送完数据区的结束标志后发送BCC，接收端一边接收数据一边计算BCC，最后与接收到的BCC比较。

### 5.3.1.5 循环冗余码CRC

- 分组码，用代数多项式来表示码字。
- 循环码的特性：
  - 任何两个码字按模2相加后，形成的新序列仍为一个有用码字——线性码。
  - 一个码字的循环移位也是另一个码字。
- 循环冗余码的原理：
  - 把k位的帧看成一个k-1次的多项式M(x)的系数列表
    - 1011001看成是多项式 $x^6 + x^4 + x^3 + x^0$ 的系数列表
  - 设定一个生成多项式G(x)，G(x)为n阶( $k > n$ )。
  - $x^n M(x)/G(x) = Q(x) + R(x)/G(x)$ ，其中Q(x)为商、R(x)为余数，R(x)即为M(x)的CRC码。
  - 将CRC码接在帧后一起发送，即发送数据为 $x^n M(x) + R(x)$ 。
  - 因为 $(x^n M(x) + R(x))$ 一定能被G(x)整除，接收方只要计算CRC，并所得余数为0即为正确。

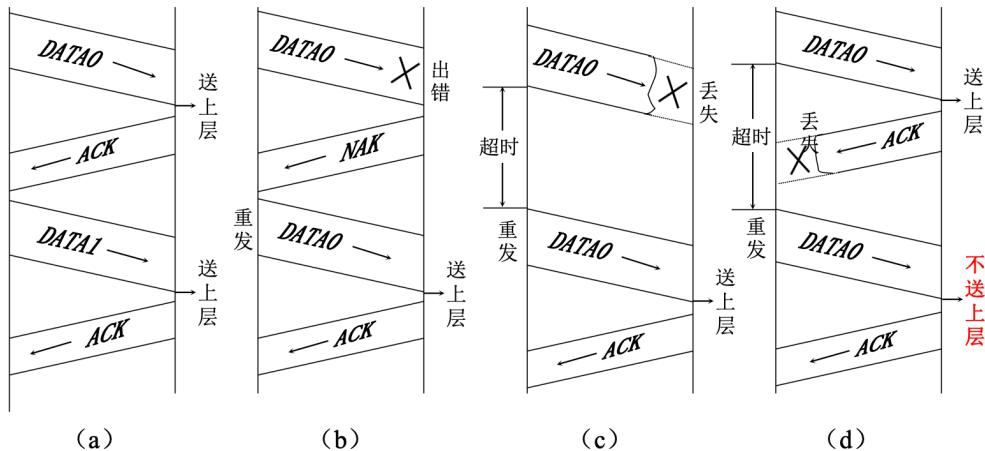
如数据比特序列为10010111, G(x)=110101



### 5.3.2 反馈重发差错控制

- 处理传输差错的通常办法：
  - 肯定确认**: 接收端对收到的帧校验后未发现错误，回送一个肯定确认信号(ACK)。
  - 否定确认**: 接收端收到一个帧后发现有错误，则回送一个否定确认信号(NAK)，发送端收到NAK后必须重发。
  - 超时重发**: 发送端在发出一个帧后开始计时，如果在规定时间内没有收到关于该帧的ACK或NAK，则认为发生帧丢失或确认信号丢失，必须重新发送。
- 三种ARQ方案: **等待式**、**回退N帧**、**选择性重发**。

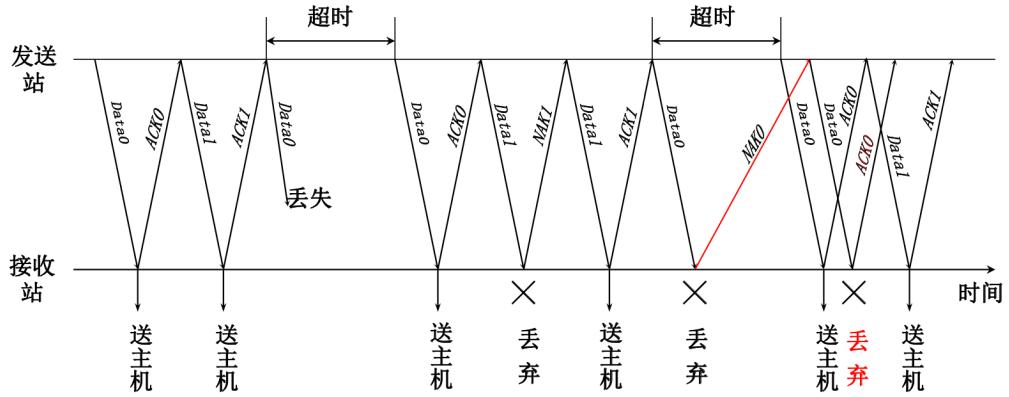
#### 5.3.2.1 等待式ARQ



等待式ARQ方案中数据帧在链路上的传输

(a) 正常情况 (b) 数据帧出错 (c) 数据帧丢失 (d) 应答帧丢失

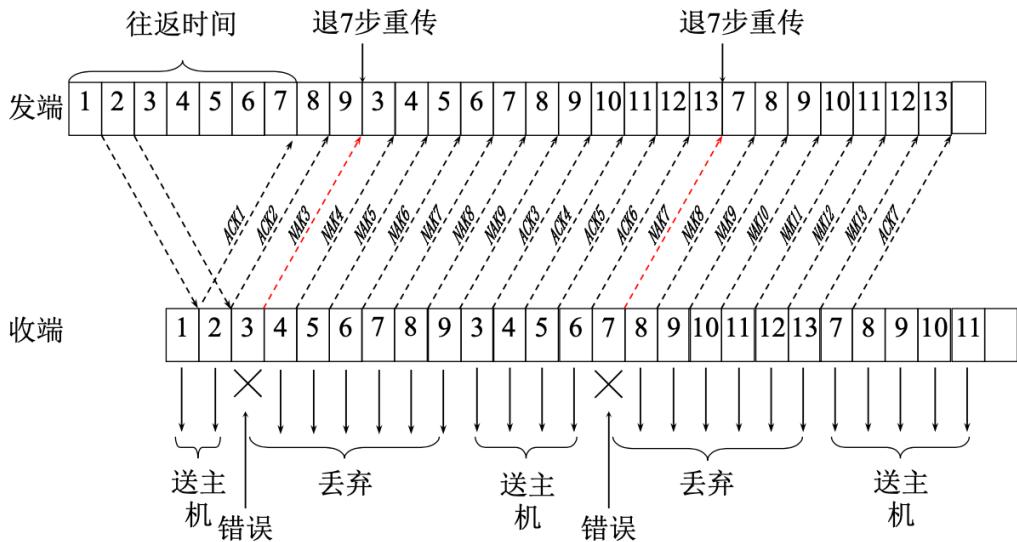
- 应答帧的编号问题：



等待式ARQ示例——1比特序号  
(发送缓冲区: 1, 接收缓冲区: 1)

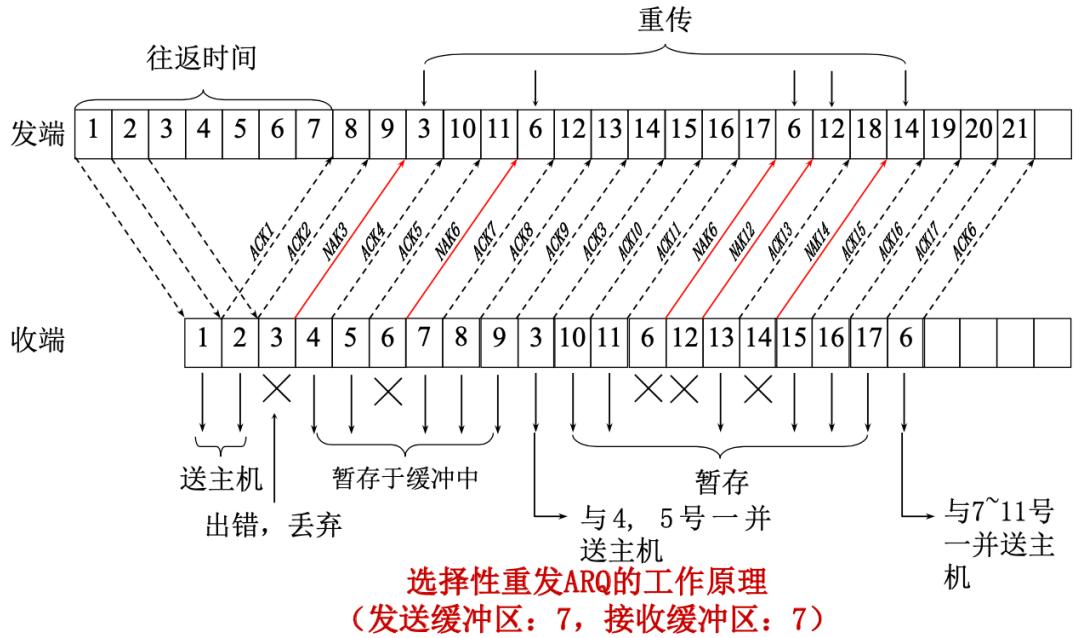
### 5.3.2.2 回退N帧ARQ

- 连续重发请求：发送端连续发送帧，不等前帧确认便发下一帧。
- 把等待时间利用起来，传输效率提高，然而需要更大的缓冲存储空间。
- 连续重发请求的两种基本方法：
  - 回退N帧ARQ
  - 选择性重发ARQ
- 回退N帧ARQ方案的示例：



回退N帧ARQ的工作原理  
(发送缓冲区: 7, 接收缓冲区: 1)

### 5.3.2.3 选择性重发ARQ



#### 5.3.2.4 通信效率

- 链路的有效效率  $R_e$ : 假设链路传输速率为  $R$

$$R_e = \frac{\text{正确接收的比特数}}{\text{传输这些比特的总时间}}$$

$$\text{通信效率} \eta = \frac{R_e}{R}$$

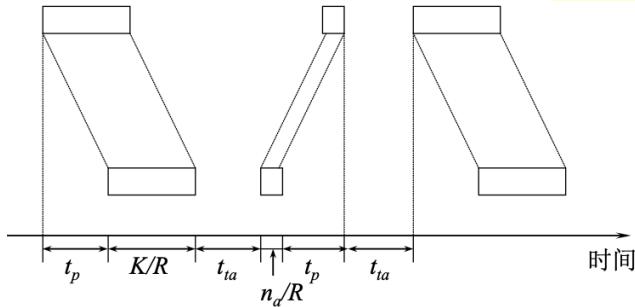
帧出错的概率 ( $p$  为出错概率) :  $p_e = 1 - (1 - p)^K$

$$P[\text{需 } j \text{ 次传输}] = p_e^{j-1}(1 - p_e)$$

$$\text{平均传输次数} \bar{N}_t = \sum_{j=1}^{\infty} j p_e^{j-1}(1 - p_e) = \frac{1}{1 - p_e}$$

- 传输一帧的时间  $T_s$ :

$$T_s = \frac{K + n_a}{R} + 2(t_p + t_{ta})$$



#### 数据链路的时延分析

- 等待式ARQ的通信效率

$$\text{传输时延} T = \frac{K}{R}$$

$$R_e = \frac{K - n_h}{N_T \cdot T_s} = \frac{K - n_h}{\frac{1}{1-p_e} \left[ \frac{K + n_a}{R} + 2(t_p + t_{ta}) \right]} = \frac{(1-p_e)(K - n_h)R}{K + n_a + 2(t_p + t_{ta})R}$$

$$\text{Re} \approx \frac{K}{\frac{1}{1-p_e}(T + 2t_p)} = \frac{K(1-p_e)}{(1+2a)T} \quad \longrightarrow \quad \eta = \frac{R_e}{R} = \frac{K(1-p_e)}{(T + 2t_p)R} = \frac{1-p_e}{1+2a} \quad a = \frac{t_p}{T}$$

- 等待式ARQ适用场合：

- 较适合于短链路和传输速率不高的场合。
- 不适合于卫星链路和地面高速率的数据链路。
- 为了可靠性牺牲了传输效率，过于浪费信道的有效带宽。

- 回退N帧ARQ的通信效率

$$\begin{aligned} \bar{N}_T &= 1 + Np_e(1-p_e) + 2Np_e^2(1-p_e) + \dots \\ &= 1 + \frac{Np_e}{1-p_e} = \frac{1 + (N-1)p_e}{1-p_e} \\ R_e &= \frac{K - n_h}{\frac{1 + (N-1)p_e}{1-p_e} \cdot \frac{K}{R}} = \frac{(1-p_e)(K - n_h)R}{[1 + (N-1)p_e]K} \quad \longrightarrow \quad \eta = \frac{(1-p_e)(K - n_h)}{[1 + (N-1)p_e]K} \\ N &= \left\lceil \frac{T_s}{K/R} \right\rceil \quad T_s = \frac{K + n_a}{R} + 2(t_p + t_{ta}) \end{aligned}$$

- 选择性重发ARQ的通信效率

$$\begin{aligned} R_e &= \frac{K - n_h}{\frac{1}{1-p_e} \cdot \frac{K}{R}} = \frac{(1-p_e)(K - n_h)R}{K} \\ \eta &= \frac{(1-p_e)(K - n_h)}{K} \end{aligned}$$

## 5.4 滑动窗口式流量控制

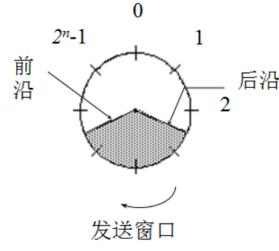
- 流量控制：协调发送端和接收端，避免发送过快，使得接收端来不及处理而丢失数据。
- 在接收方的缓冲区到达一定量时，及时通知发送方暂停发送，等候通知。

### 5.4.1 滑窗式流控的主要思想

- 允许连续发送多个帧而无需逐个地等待应答。
- 每个要发送的帧都包含一个序号(n位，取值为0 ~  $2^n - 1$ )，在传送过程中，循环重复使用已收到确认的那些帧的序号。
- 窗口机制：在发送端和接收端分别设置发送窗口和接收窗口。
  - 通信两端设置一定的缓冲区，用于保存已发送但尚未被确认/期望接收的数据帧。

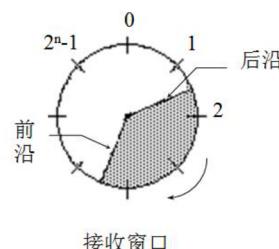
### 5.4.2 滑动窗口协议主要工作原理

- 发送端：



- 始终保持一个已发送但尚未确认的帧的序号表。
- 发送窗口：
  - 对应于在还没有收到对方确认的条件下最多允许发送的帧的序号范围。
  - **前沿**: 表示还能发送的帧的最大序号。
  - **后沿**: 表示还能发送的帧的最小序号(已发送、等待确认的帧的最大序号+1)。
  - 发送窗口大小 = 前沿-后沿+1, **大小可变**。
  - 发送端每发送一个帧, 序号取后沿值, 后沿加1; 每接收到一个正确响应帧, 前沿加1。

- 接收端:

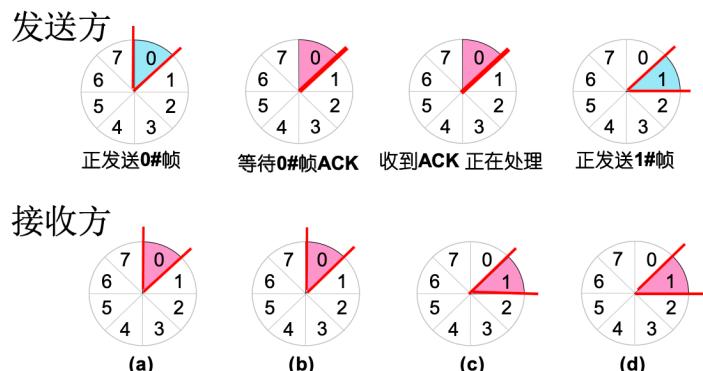


- 用接收窗口表示容纳**允许接收的信息帧**, 落在窗口外的帧均被丢弃。
- 接收窗口：
  - **大小固定, 但不一定与发送窗口相同**。
  - **前沿**: 表示允许接收的序号最大的帧。
  - **后沿**: 表示希望接收的序号最小帧。
  - 如果序号等于后沿的帧被正确接收, 并产生一个响应帧, 前沿、后沿都加1。

- 基于确认的窗口滑动:

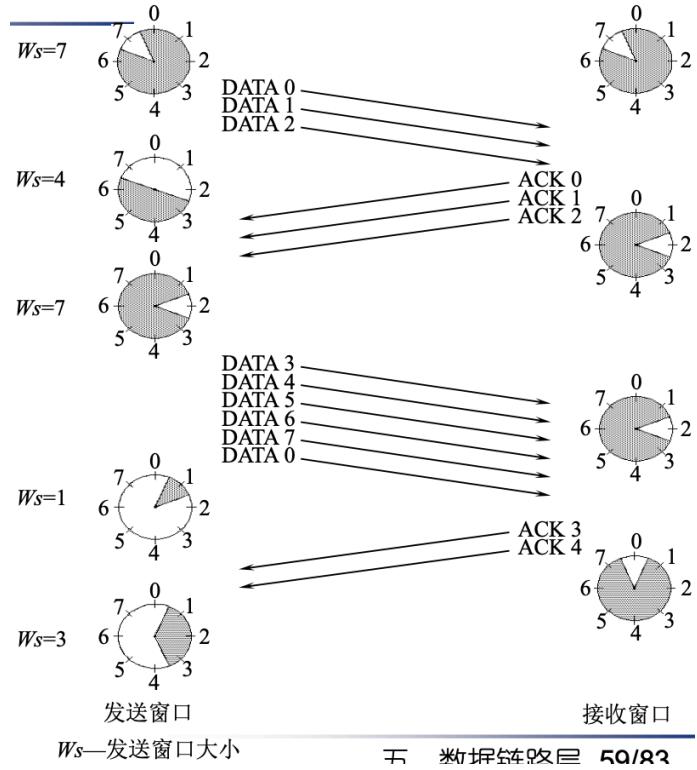
- 接收窗口不动时, 发送窗口不会前进。只有接收窗口前进后(发送确认), 发送窗口才能前进。
- 3种确认方式:

1. 接收端一收到正确帧, 就发送一个确认帧。
2. 累积确认: 在连续收到几个正确的数据帧以后, 对最后一个数据帧发回确认帧。
3. 指带确认: 在接收方向发送方发送数据时, 指带对前面收到的帧进行确认。



一个大小为1 ( $W_T=1$ ,  $W_R=1$ )、有3位序列号的滑动窗口示例

- 接收端通过反馈确认帧的速度来调整或限制发送窗口的大小，达到流量控制的目的。



第5,6,7,0帧已接收，但仍在缓冲区中，尚未处理。

- 窗口大小的选取

- 对于等待式ARQ协议

$$X_S = 1, W_R = 1$$

- 对于回退N帧ARQ协议

$$1 < W_S \leq 2^n - 1, W_R = 1$$

- 对于选择性重发ARQ协议

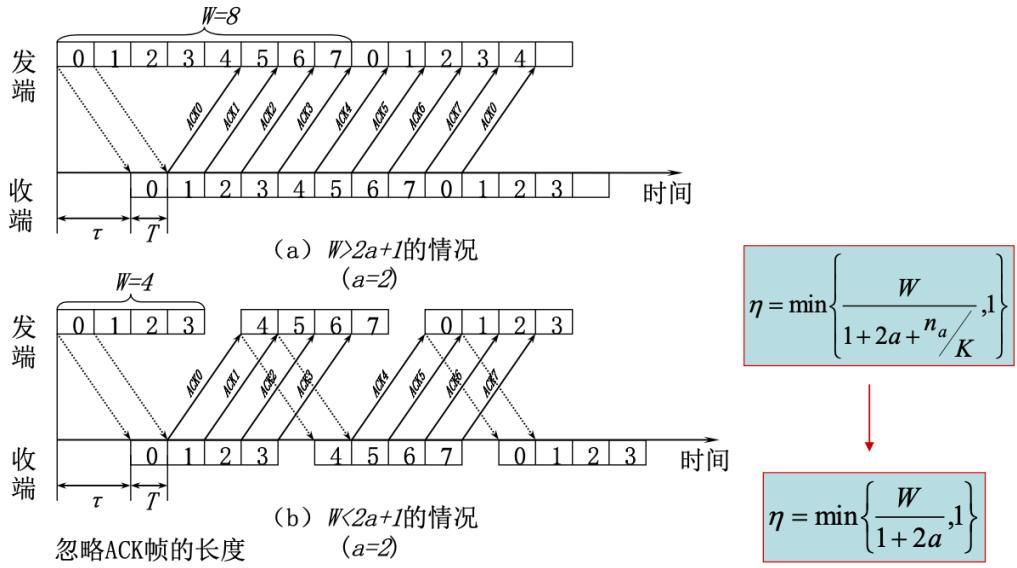
$$W_R > 1, W_S > 1, W_S \geq W_R$$

当  $W_S = W_R$  时,  $W_S = W_R \leq 2^{n-1}$

$$\quad \quad \quad W_S \leq 2^n - 1$$

$$\quad \quad \quad W_S + W_R \leq 2^n$$

- 滑窗流控的链路利用率



### 窗口大小对传输效率的影响

## 5.5 数据链路层协议实例

### 5.5.1 HDLC——高级数据链路控制

- HDLC: 面向bit的同步通信协议
- HDLC的特点
  - 不依赖于任何一种字符编码集，数据报文可透明传输，“0比特插入法”易于硬件实现。
  - 全双工通信，有较高的数据链路传输效率。
  - 采用CRC检验，对信息帧进行顺序编号，防止漏收或重复，传输可靠性高。
  - 传输控制功能与处理功能分离，具有较大灵活性。
- HDLC的配置方式：
  - 非平衡配置：一个主站及一个或多个从站组成，主站控制数据链路的工作过程并发出命令，从站接受命令，发出响应。
  - 平衡配置：链路两端的两个站都是组合站，同时具有主站和从站的功能。
- HDLC的帧类型(3种)：
  - 信息帧：承载要传递给站点的用户数据，也包含用于流控制和错误控制的控制信息。
  - 监控帧：实施流控制和错误控制，四种格式RR、RNR、REJ、SREJ。
  - 无序号帧：无序号帧用于控制链路本身，如呼叫、确认和断开连接等控制。
- HDLC的操作流程：初始化、数据传输、断开连接。

### 5.5.2 PPP——点对点协议

- PPP协议的功能：
  - 差错检测
  - 支持多种协议(IP、IPX、DECnet等)
  - 支持多种类型链路
  - 连接时允许协商IP地址
  - 允许身份认证
  - PPP协议不提供纠错、流量控制、序号、多点线路
- PPP协议的组成：
  - PPP协议提供了串行点对点链路上传输数据报的方法，包括以下三个部分：

- 成帧方法：将 IP 数据报封装到串行链路，支持异步链路、面向bit的同步链路。
  - 扩展的链路控制协议LCP：提供了建立、配置、维护和终止点对点链接的方法。
  - 网络控制协议簇NCP：支持各种网络层协议。
- PPP的帧格式：类似于HDLC，但面向字符。
- PPP的工作过程：
  - 发送端PPP首先发送LCP帧，以配置和测试数据链路。
  - 在LCP建立好数据链路并协调好所选设备之后，发送端PPP发送NCP帧，以选择和配置一个或多个网络协议(分配IP地址)。
  - 当所选的网络层协议配置好后，便可将各网络层协议的分组发送到数据链路上。
  - 配置好的链路将一直保持通信状态，直到LCP帧或NCP帧明确提示关闭链路，或有其它的外部事件发生。
- 一次使用PPP协议的过程：
  - a) 初始状态
  - b) 建立连接：建立成功到c)，否则到a)
  - c) 选项协商：协商成功到d)，否则到g)
  - d) 身份认证：认证成功到e)，否则到g)
  - e) 配置网络：网络配置完后到f)
  - f) 数据传输：数据传输完后到g)
  - g) 释放链路：回到a)

## 5.6 介质接入控制

### 5.6.1 广播信道的分配问题

- 广播信道：多路访问信道或随机访问信道，是公用信道
  - 存在冲突，需要访问机制解决信道的争用
- 介质接入控制MAC子层：
  - 用来决定广播信道中信道分配的功能及协议
  - 位于数据链路层的底层
- 信道分配方式：静态分配、动态分配

#### 5.6.1.1 静态分配

- 静态分配：将广播信道分为多个“子”信道，每个用户分配一个子信道
- 信道分配技术：频分复用、时分复用、波分复用、码分复用
- 缺点：信道利用率低
- 适用于卫星通信

#### 5.6.1.2 动态分配

- 假设：
  - 数据站模型：站点独立，以固定速率产生帧，生成一帧，就等待发送，直到成功发送。
  - 单信道假设：所有通信通过单一的信道来完成，对所用数据站都平等。
  - 关于冲突：如果两帧的发送在时间上出现重叠，则认为发生冲突，所有的站都能检测到冲突，受冲突的帧必须重传。

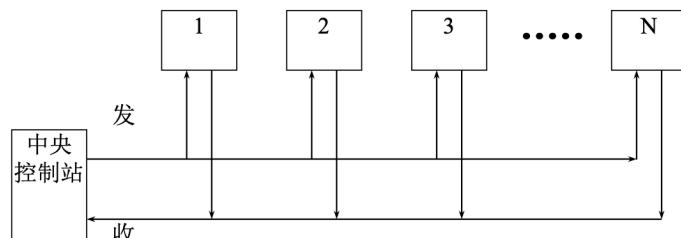
- 时间参考
  - 连续时间——帧在任何时刻可以发送。
  - 时隙——帧在时隙开始的时刻发送。
- 有载波侦听或无载波侦听
  - 载波监听——所有站在使用信道前都可以检测信道是否正在使用，如使用则等待至空闲。
  - 无载波监听——所有站使用信道前不检测信道，盲目发送帧，事后才能确定是否发送成功。
- 信道动态分配需要考虑的要素：
  - 在哪里分配？
    - 集中式：其他站点必须得到中央监控站的允许才能使用信道。
    - 分布式：所有的站点共同完成信道接入控制功能。
  - 如何分配？
    - 循环式：每个站轮流等待机会，适合于有多个站点要发送很多数据的情况。
    - 预约式：把信道的使用时间划分为时隙，站点在约定的时隙内发送数据。
    - 竞争式：各个站自由竞争发送机会。特点：简单，适合轻负载，负载重时，性能下降很快。

## 5.6.2 查询技术

- 常见的一种信道访问技术，需要一个中央控制站，中央控制站周期性地向各工作站发送查询帧
  - 各用户只有当被控制站查询时，才能发送信息。
  - 没有被查询时，各信息只能在本地站中排队等候。
  - 一般适用于集中式拓扑结构的网络，也可以用于回路式(loop)网络中。

### 5.6.2.1 roll-call查询

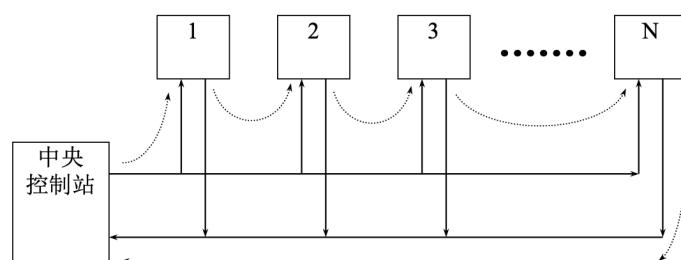
- roll-call查询：中央站为每个站产生查询帧，逐个查询各站点。



roll-call查询的工作原理

### 5.6.2.2 hub查询

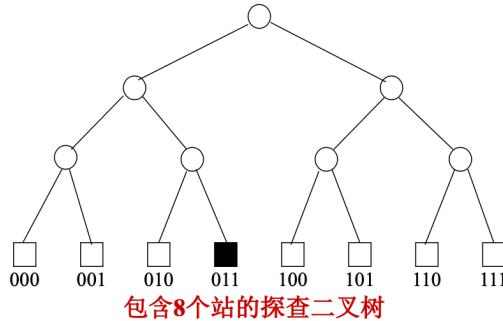
- hub查询：中央站产生一个查询帧，该查询帧在各站中按序传递。



Hub查询的工作原理

### 5.6.2.3 探查法

- 传统查询策略的缺点
  - 在轻负载的情况下，效率较低
  - 在重负载的情况下，将引入较长的时延
- 探查法的主要思想：先探查确定需要发送信息的站点 (**自适应查询**)
  1. 同时探查一组(两个或多个)站点。如果其中某个站点需发送数据，则它反馈以某种形式的信号。
  2. 将具有反馈信号的站点组分成两个子组，重复过程1。
  3. 对于最终的探查(即组内只有一个站点)，由控制站发送给该站点一个查询帧，使其可以发送数据。



总共需要7次询问就可以完成整个探查循环过程。

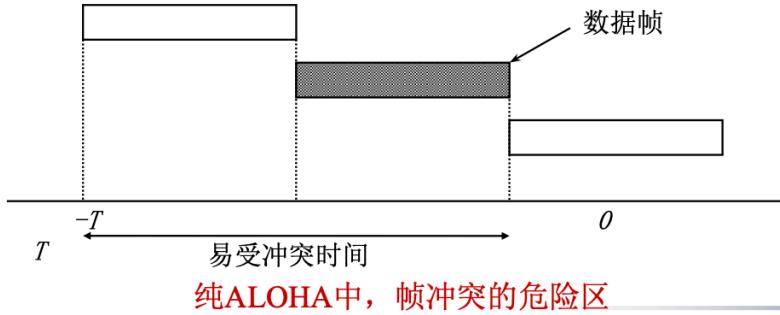
- 自适应查询的关键： $j$ (每组包含 $2^j$ 个站)的选取
  - 纯查询：
    - $j=0$ ，起始时每组只包括一个站。
    - 共需 $2^n$ 次查询，且与要送数据的站点个数无关。
  - 纯探查：
    - $j=n$ ，起始时只有一组。
    - 在只有一个站点希望发送数据时，需查询 $2n+1$ 次。
    - 如所有站点都要发送数据，查询次数为 $2^{n+1} - 1$ （遍历每个节点）。

### 5.6.3 ALOHA

- ALOHA协议：
  - **随机接入协议**，控制多个用户共用一条信道
  - 目的：解决信道的动态分配，可用于任何无协调关系的用户争用单一共享信道使用权的系统
  - 两种类型：纯ALOHA、时隙ALOHA

#### 5.6.3.1 纯ALOHA

- 工作原理：
  - 任何一个站都可以在帧生成后立即发送。
- 发送站通过信号的反馈，检测信道，以确定发送是否成功。
  - 如发送失败，发送站经**随机延时**后再发送。
- 通信量越大，碰撞的可能性越大。
- 纯ALOHA的易受冲突时间：**2T**(T为传输一帧时间)



- 纯ALOHA信道的效率
  - 帧时T：发送一个标准长度的帧所需的时间
  - 设：无限多个用户产生新帧的概率服从泊松分布，平均每个帧时产生S个新帧
  - 当S > 1时，肯定超过一个站点在发送帧，**每个帧都将冲突**，所以吞吐率应为  $0 < S < 1$
  - 除新帧外，凡冲突的帧也要重发
  - 设：帧发送的平均值为G帧/T， G中包括每个帧时内产生的新帧S和由于冲突而需重发的帧R，  
 $G=S+R$ 
    - 当轻负载( $S \ll 1 \rightarrow 0$ )时，几乎无冲突，则 $G \approx S$
    - 当重负载( $S \rightarrow 1$ )时，冲突频繁，则 $G > S$

- 纯ALOHA的吞吐率

设：在任一帧时内生成k帧（包括新旧帧）的概率服从泊

$$\text{松分布，为: } P_k = \frac{G^k}{k!} e^{-G}$$

则：生成0帧的概率为  $P_0 = e^{-G}$

两个帧时内产生的平均帧数为  $2G$ ，那么  $2T$  时间间隔内生成0

帧的概率为  $P_0 = e^{-2G}$ ， $2T$  时间间隔内至少产生一次冲突（即：在  $2T$  时间内至少有别的站发一帧）的概率为  $1 - e^{-2G}$

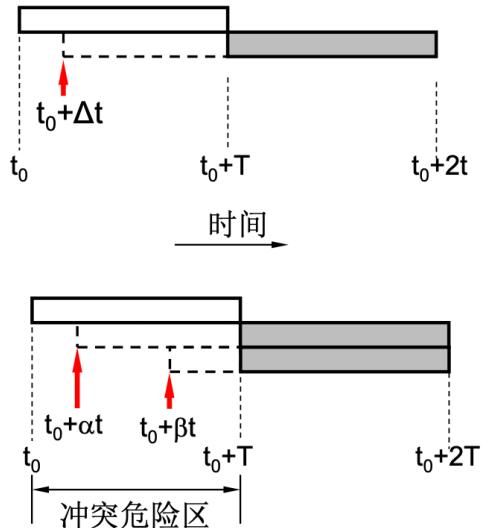
则： $T$  时间内需重发的平均帧数为  $R = G(1 - e^{-2G})$

$$G = S + R = S + G(1 - e^{-2G}) \rightarrow S = G e^{-2G}$$

在  $G=0.5$  时，吞吐量  $S$  达到最大值：  $S_{\max} = 1/(2e) \approx 0.184$

### 5.6.3.2 时隙ALOHA

- 时隙ALOHA的工作原理：
  - 将时间轴分成固定的时隙，各站点只能在一个时隙的起始时刻才能发送帧。
  - 如发送失败，发送站经**随机时隙的延迟**后再发送。
- 时隙ALOHA的冲突危险区
  - 在一个时隙内只产生一个新帧，新帧不允许立即发送，将在下一个时隙的开始处  $t_0 + T$  时发送，不会发生冲突。
  - 在一个时隙内产生一个以上新帧，下一个时隙的开始处  $t_0 + T$  时，一个以上的帧同时发送，将发生冲突，即**冲突危险区为T**。



- 时隙ALOHA中的时隙

- 时隙的长度对应一帧的传输时间，其起点由专门的信号来标志。
- 新帧的产生是随机的，但时隙ALOHA不允许随机发送，凡帧的发送必须在时隙的起点。
- 冲突主要发生在时隙的起点处，一旦发送成功，则不会出现冲突，即生成新帧并等待发送的这一帧时内，是冲突危险区，时间长度为T，是原来的一半。

- 时隙ALOHA的吞吐率

在一个时隙的起点没有其它帧发送的概率为：

$$P_0 = e^{-G}$$

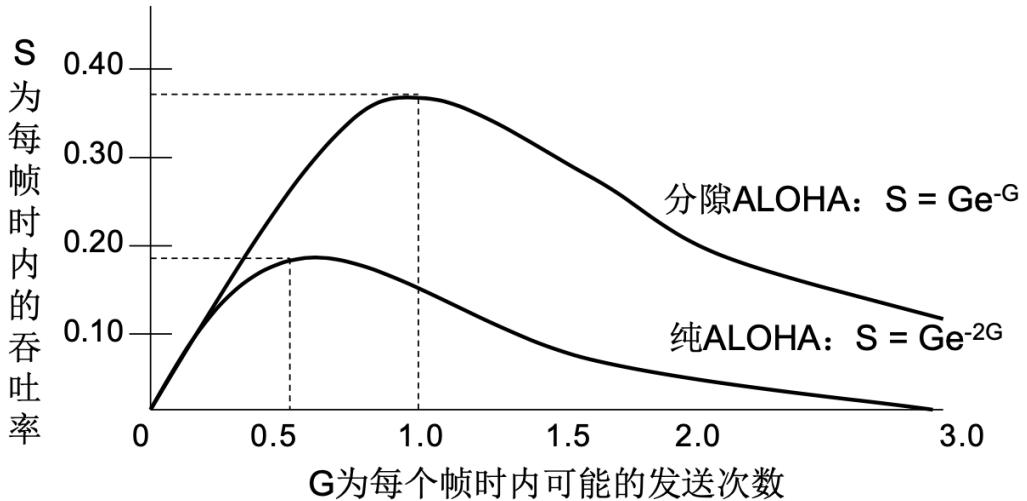
因此：  $S = GP_0 = Ge^{-G}$

当  $G = 1$  时，吞吐量  $S$  为最大，  $S_{max} \approx 0.368$

时隙ALOHA的最好结果是：37%的时隙为空，37%的时隙传送成功，26%的时隙产生冲突。

- 纯ALOHA和时隙ALOHA的比较

- 纯ALOHA中，一旦产生新帧，就立即发送，发生冲突的可能伴随着发送的整个过程。
- 时隙ALOHA中，发送行为必须在时隙的开始，一旦在发送开始时没有冲突，则该帧将成功发送。
  - 随着G的增大，空时隙数( $e^{-G}$ )会减少，而产生冲突的时隙数按指数增加。
  - 每帧传送次数  $E = e^G$ ，说明通信负载G稍微增加就会大大降低系统的吞吐性能。
  - 时隙ALOHA需要全网的同步机制，增加了复杂性。



ALOHA系统中吞吐率和帧产生率之间的关系

#### 5.6.4 CSMA

- CSMA载波监听多路访问
  - 在发送数据之前先侦听信道
    - 如果已经有站在发送信息，信道正忙，则本站就不发送，或者继续侦听信道，或者等待一段随机时间之后，再试图发送，并且在发送之前再侦听一下信道，看是否信道有空。
    - 如果侦听到信道是空闲的，则根据预定的控制策略来决定，是立即将自己的帧发送出去还是为慎重起见暂时不发送出去。
  - **减少站发送帧的盲目性**
- CSMA的三种方式：非持续型CSMA、p-持续型CSMA、1-持续型CSMA

##### 5.6.4.1 非持续型CSMA

- 工作原理：
  - 每个站在发送前先侦听信道，如果此时信道空闲，则该站就可以向公共的信道上发送帧。
  - 如信道正忙，它就停止侦听，而是**延时一随机时隙数后**，再侦听信道。
- 缺点：已经侦听到信道忙的站点在延迟时间没有结束之前，即使后来信道已经空闲，它们也不发送任何信息，限制了信道利用率的进一步提高。

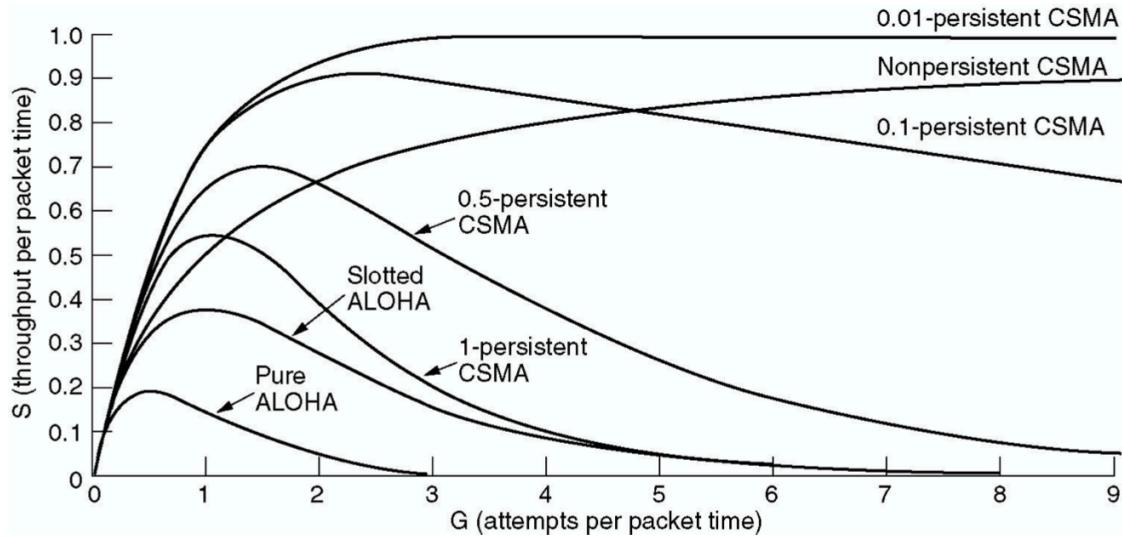
##### 5.6.4.2 p-持续型CSMA

- 工作原理：
  - 先侦听信道，如信道正忙，则**等到下一时隙**；如信道空闲，则以**概率p**发送，而以**概率(1-p)**把本次发送延至下一时隙，直至发送成功。
  - 既考虑到尽快使用已空闲的信道，又考虑到发送帧时的主动退避，在性能上比非持续型CSMA好。

##### 5.6.4.3 1-持续型CSMA

- 工作原理：
  - 每个站在发送前，先侦听信道，如信道正忙，则等待并**持续侦听**。一旦信道空闲，立即发送，即发送的概率为1。
  - 如冲突，则**延时一随机时隙数后**，重新发送。
- **p-持续CSMA的极端情况**，指只要发现信道空闲，就立即发送帧。

- 轻载时性能较好，但重载时性能急剧变坏。
- CSMA方式的性能分析



不同随机访问协议信道利用率的比较

S: 表示信道的吞吐量，在T时间内成功发送的平均帧数；

G: 表示网络负载， T时间内总共发送的平均帧数（成功+重发）

性能比较：

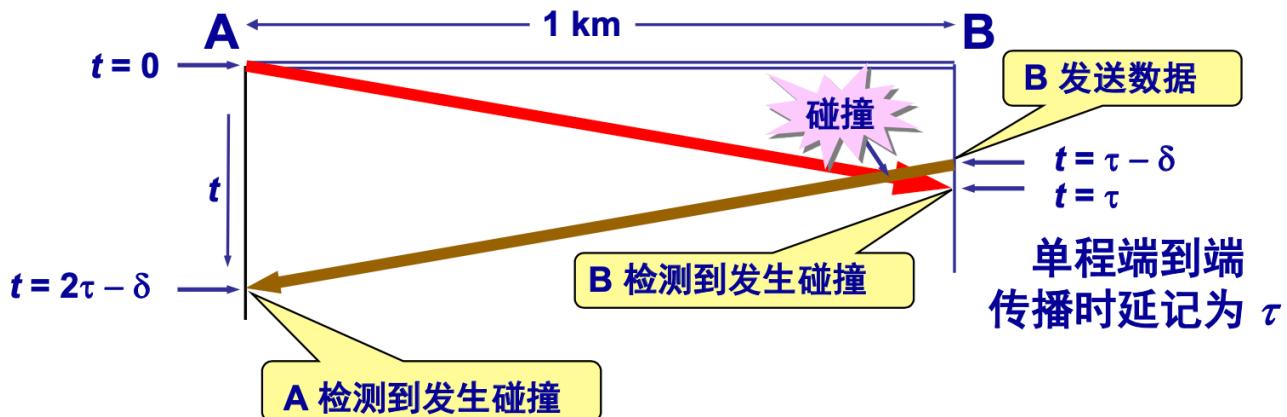
0.01-持续>0.1持续>0.5持续>1持续>slotted-ALOHA>pure-ALOHA

重载时倾向于politely退让

非持续在重载时性能表现良好

## 5.6.5 CSMA/CD

- CSMA并不能完全解决冲突问题：传播时延对载波监听的影响。

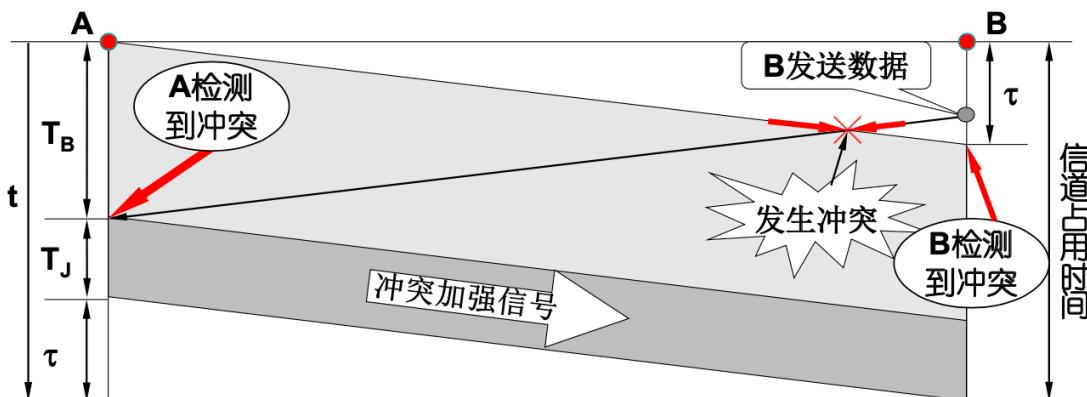


由于监听的传播时延，需要 $\tau$ 的时间，B才能监听到A已经发送了数据，即信道被占用。

如果有冲突，一定能在 $2\tau$ 时间内检测到。

- 检测冲突的方法：

- 信号电平法：基于基带传输，两个帧信号叠加后，电压大一倍。
- 过零点检测法：用曼切斯特编码时，零点在每比特的正中央，当有干扰时，则可能偏移。
- 自发自收法：在发送数据的同时也在接收，并逐个比特比较。
- 检测出冲突后，立即停止发送，可以节省时间和带宽。
- CSMA/CD(带冲突检测的载波侦听多路访问)
  - 不仅在发送前进行载波侦听，而且在发送过程中也侦听是否发生冲突 ( $2\tau$ 时间内)，并在监测到冲突后及时中断发送。
  - 可以减少信道工作时间，从而进一步提高系统吞吐性能和减小帧传输时延。
  - 在Ethernet总线式局域网中，采用CSMA/CD方式，有两个主要操作规程：**竞争发送和无冲突接收**。
  - 竞争发送方式：
    - 各站点在发送之前对总线进行侦听，只要总线空闲便开始发送，否则持续侦听总线，直至监测到总线空闲——**1持续型CSMA**。
    - 站点在发送出信息后，仍对总线进行侦听，进行**冲突检测** ( $2\tau$ 时间内)。若检测到冲突产生：
      - **停止数据发送**：放弃这次发送，延迟一段时间等待总线空闲后重传。
      - **对总线进行干预**：继续发送一小段时间(发送小于最小帧长的异常帧)，使总线上各站点都知道已发生了冲突而停止各自的发送。
  - CSMA/CD发生冲突时对信道占用时间的影响：(A检测到冲突后立即停止发送原数据，再发送一个冲突加强信号)



如一个站点发送并经 $2\tau$ 后，没有冲突，即发送成功。

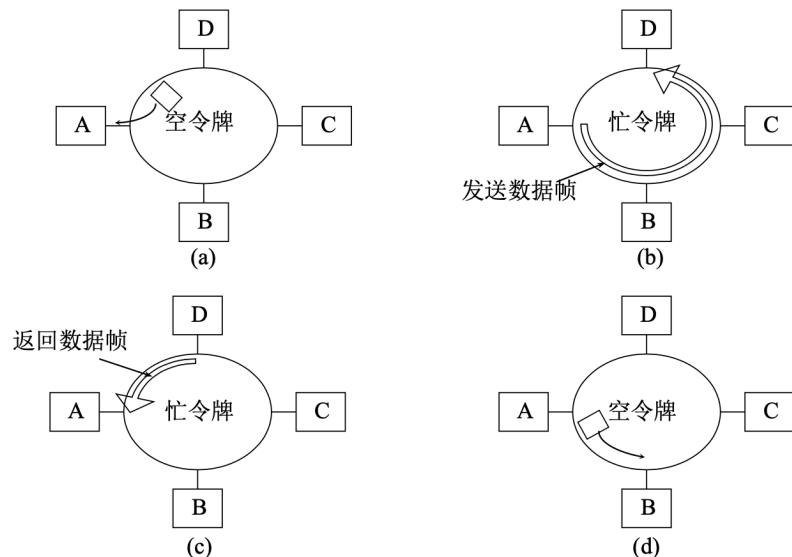
典型的，一公里长的同轴电缆， $\tau \approx 5\mu s$   $2\tau \approx 10 \mu s$ 。

- 二进制退避
  - 防止下一轮征用信道再次发生冲突
  - 让发生碰撞的站在停止发送数据后，不是立即再发送数据，而是**推迟一个随机时间**。（这样做是为了减小再次发生碰撞的概率）
    1. 确定基本退避时间，一般是取为争用期 $2\tau$ 。
    2. 定义参数 $k = \min(\text{重传次数}, 10)$ 。
    3. 从离散整数集合 $[0, 1, 2, \dots, (2^k - 1)]$ 中随机地取出一个数，记为 $r$ ，重传输所需的时间就是 $r$ 倍的基本退避时间。
    4. 当重传达16次仍不能成功时，则丢弃该帧，并向高层报告。
- 无冲突接收

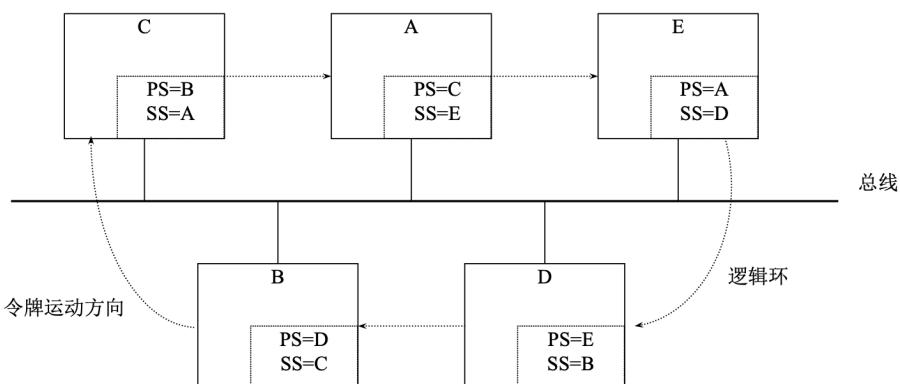
- 发送站点的干预方法是检测到冲突后，再继续发一个32位的不完全帧，它小于最短的信息帧长度（ $2\tau$ 对应的帧长）。
- 当接收站点收到这一不完全帧时，便知道发生了冲突，于是把已收到的信息作为错误信息处理，再进入另一次接收的准备。
  - 当接收站点收到异常帧时，知道发生了冲突，这是一种冲突加强的方式。

### 5.6.6 令牌类接入控制技术

- 令牌环工作原理



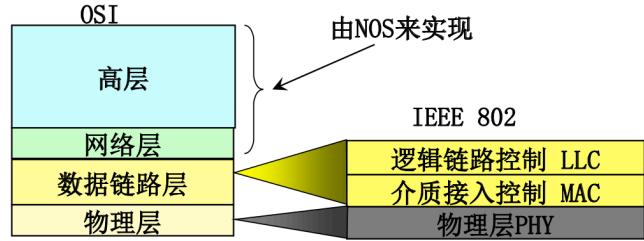
- 令牌总线的逻辑环



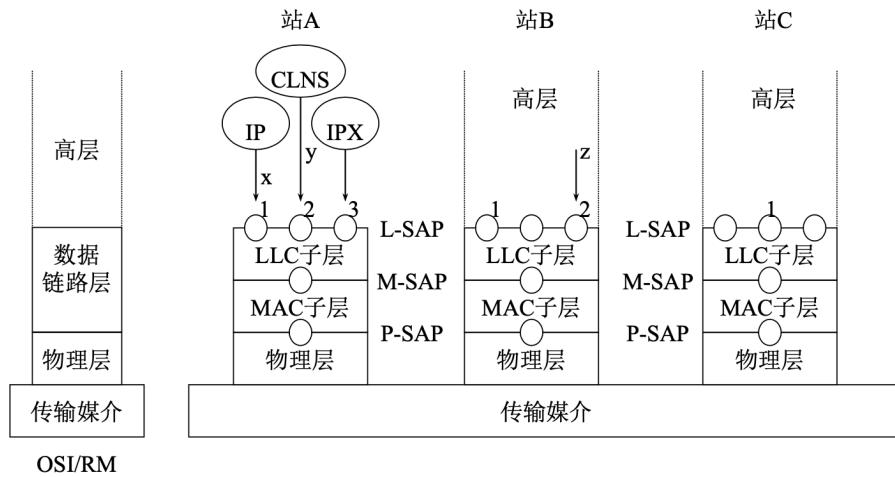
## 5.7 局域网标准

### 5.7.1 IEEE局域网标准

- IEEE局域网——IEEE 802标准，公用信道
  - 物理层：与OSI/RM的物理层相对应，具体协议标准与传输介质有关。
  - 介质接入控制(MAC)层：具体管理通信实体接入信道而建立数据链路的控制过程。
  - 逻辑链路控制(LLC)层：提供一个或多个服务访问点，以复用的形式建立多点—多点之间的数据通信连接，并包括寻址、差错控制、顺序控制和流量控制等功能。



- IEEE局域网参考模型



**IEEE局域网参考模型、服务访问点**

- 逻辑链路控制子层 LLC

- LLC子层提供**确认机制**和**流量控制**, MAC子层只提供尽力而为的数据报服务。
- LLC隐藏了不同MAC子层的差异, 为网络层提供单一的格式和接口。
- LLC提供三种服务选项:
  - 不可靠的数据报服务
  - 带确认的数据报服务
  - 可靠的面向连接服务
- LLC帧头基于HDLC协议

- 介质访问控制子层 MAC

- 数据封装 (发送和接收)
  - 成帧 (帧定界, 帧同步)
  - 寻址 (源和目的地址的处理)
  - 错误检测 (物理介质传输错误的检测)
- 介质接入管理
  - 介质分配 (冲突避免)
  - 争用解决方法 (冲突处理)

- 分成两个子层的原因

- 管理多点访问信道的逻辑不同于传统的数据链路控制。
- 对于同一个LLC, 可以提供多个MAC选择。
- LLC层与局域网形态和传输介质**无关**, MAC层与局域网形态和传输介质**直接相关**。

## 5.7.2 以太网

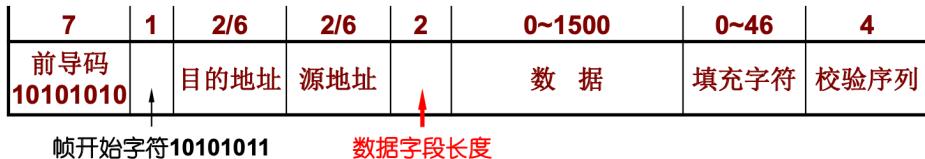
### 5.7.2.1 经典以太网

- IEEE 802.3，1-持续型CSMA/CD技术的802.3局域网标准，802.3标准与以太网协议略有差别
- 802.3的传输电缆分以下四种

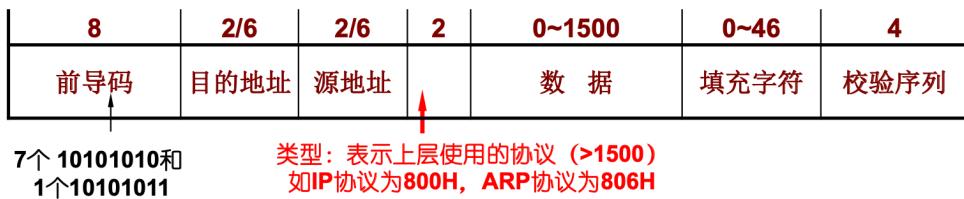
名称	电缆	最大区间长度	节点数/段	优点	接口
10Base5	粗缆	500m	100	用于主干 已基本淘汰	AUI
10Base2	细缆	185m	30	廉价 已基本淘汰	BNC
10Base-T	双绞线	100m	1024	易于维护	RJ-45
10Base-F	光纤	2km	1024	用于楼间	ST

- 双绞线以太网：仅用两对线，且全双工，采用曼切斯特编码。
- 集线器(HUB)
  - 使用了大规模集成电路芯片，可靠性很高。
  - 使用电子器件来模拟实际电缆线的工作，整个系统仍然像一个传统的以太网那样运行。
  - 各结点发送来的信号通过集线器集中，集线器将信号进行整形、放大后发送到其它所有结点。
  - 使用集线器的以太网在逻辑上仍是一个总线网，各工作站共享逻辑上的总线，使用CSMA/CD协议。
  - 集线器很像一个多接口的转发器，工作在物理层。
- 以太网MAC子层协议（可兼容）

#### 802.3的帧结构：



#### 以太网帧结构：



#### ● MAC地址：

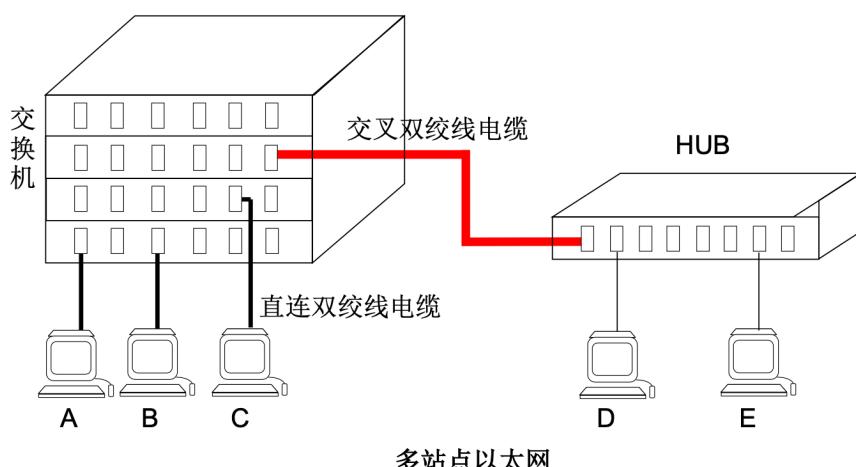
- 在6个字节(共48位)的地址中有46位用于地址的指定，即有 $2^{46} = 7.03687 \times 10^{13}$ 个地址
  - 最高位：I/G位，0-单个地址，1-组地址
  - 第2位：G/L位，0-全球管理，1-本地管理
- IEEE向厂家分配前24位，后24位由厂家自行指派，称为扩展标识符——地址唯一。
- 特殊MAC地址：
  - 目的地址最高位为0：普通地址
  - 目的地址最高位为1：多点发送

- 目的地址全1：广播发送
  - 网卡地址是一个全局地址
- 适配器检查 MAC 地址：适配器从网络上每收到一个MAC 帧就首先用硬件检查 MAC 帧中的 MAC 地址
  - 如果是发往本站的帧则收下，然后再进行其他的处理。
  - 否则就将此帧丢弃，不再进行其他的处理。
- “发往本站的帧”包括以下三种帧：
  - 单播帧(一对一)
  - 广播帧(一对全体)
  - 多播帧(一对多)
- 以混杂方式工作的以太网适配器只要“听到”有帧在以太网上传输就都接收下来。
- 校验序列：4个字节共32位的CRC码
- 填充字段：
  - 保证帧的最短长度为64个字节
  - 填充字节的长度为0 ~ 46字节 ( $64 - 6 - 2 - 4 = 46$ )
- 帧的最短长度为64个字节：
  - CSMA/CD中，如在 $2\tau$ 内没有冲突信号返回，则发送成功，如果发送端在 $2\tau$ 时间内帧已经发送结束，则即使冲突也无法检测，即最短帧长应与 $2\tau$ 相当。
  - 802.3局域网中：
    - 在极限条件下，802.3局域网中发送方和接收方间允许接有4个中继器，最大距离为2500 m，往返5000 m。
    - 传输速率为10M bps。
    - 往返的时间大约需要 $50\mu s$ ，再考虑一些安全余量以及2的整次幂的因素，所以通常取 $51.2\mu s$ 为争用时隙的时间长度( $51.2\mu s$ 即传输512 bit，即64字节所耗费的时间)
    - 网络速度提高，最短帧长应增大或者站点间的距离要减小。
- CSMA/CD中的二进制指数退避算法
  - 发送方在检测到冲突后，双方(或多方)都将延时一段时间。
  - 时隙的长度等于信号在介质上往返的传播时间(在以太网中，一个时隙即 $2\tau$ 为 $51.2\mu s$ )。
  - 一般地，经*i*次冲突后，发送站点需等待的时隙数将从 $0 \sim 2^i - 1$ 中随机选择。

在一个时隙的起始处，两个CSMA/CD站点同时发送一个帧，求前4次竞争都冲突的概率

- 第一次竞争冲突的概率为1;
- 第一次冲突后，A、B都将在等待0个或1个时隙之间选择，选择的组合有：00、01、10、11，共4种，其中00和11将再次冲突，所以第二次竞争时，冲突的概率为0.5
- 第二次冲突后：A、B都将在0、1、2、3之间选择，选择的组合有：00、01、02、03、10、11、12、13、20、21、22、23、30、31、32、33共16种，其中00、11、22、33将再次冲突，所以第三次竞争时，冲突的概率为0.25
- 第三次冲突后：A、B都将在0、1、2、3、4、5、6、7之间选择，选择的组合共有64种，其中00、11、……、77将再次冲突，所以第四次竞争时，冲突的概率为0.125（第*i*次冲突的概率： $1/2^{i-1}$ ）
- 前四次竞争都冲突的概率为： $1 \times 0.5 \times 0.25 \times 0.125 = 0.015625$

#### 5.7.2.2 交换式以太网



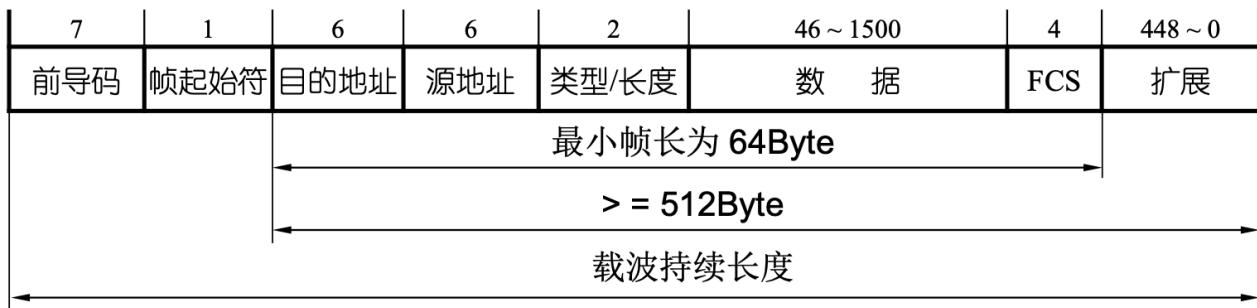
- 以太网交换机
  - 数据链路层上的设备，通过帧解析得到源地址与目的地址，在交换机下可实现定向转发（类似于点到点通信，A能直接发给C），而HUB（在物理层）仍是共享信道。
  - 每个模块实际上是一个规模较小的局域网，即一个模块就是一个共享域（以太网中，共享域即冲突域）。
  - 一个模块上任一时刻只能有一个站点发送，但分属不同模块上的端口可并行工作，这可理解为组交换：模块内共享，模块间交换——端口之间帧的传输不受CSMA/CD的限制。
  - 当每个模块都退化成只有一个端口时，即一个共享域中只有一个端口，则该交换机是全交换的。

#### 5.7.2.3 快速以太网（802.3u）

- 结构简单，兼容性好，价格相对低廉
  - 优选全双工操作，采用星型连接方式
  - 采用4B/5B的二进制编码

#### 5.7.2.4 千兆以太网 (802.3z)

- 千兆以太网支持两种工作模式
  - 全双工模式
    - 使用两根信道，不会产生冲突。
    - 传输距离取决于信号的衰减。
  - 半双工模式
    - 允许使用共享设备(如HUB)，采用CSMA/CD机制来实现信道的共享。
    - 传输距离必须考虑冲突检测，即必须考虑时隙问题。
- 802.3z的两个特性：802.3z允许链路中使用共享设备(如HUB)，为进行冲突检测，帧的传输时间必须大于最长距离的信号往返的传播时间( $2\tau$ )。
  - 载波扩展技术**：为保证与HUB的最大距离允许为100 m，802.3z把时隙定为 $4.096 \mu s$ (传输4096比特的时间)，为与802.3及802.3u兼容，不能增加最小帧的长度(512比特限制)，所以采用了载波扩展技术。
  - 短帧突发技术**：为了提高有效数据的传输率，还采用了短帧突发技术。
- 载波扩展：在帧传输完后，如尚未到达 $4.096 \mu s$ 时，则以载波信号充斥其余时间。
- 采用载波扩展技术的千兆以太网帧格式(千兆以太网的最短帧长与时隙不相关)：



对小于64Byte的以太帧，本来就必须填充，与千兆以太网格式无关

- 短帧突发
  - 载波扩展带来的问题：如长度为64Byte的短帧，都必须在512Byte的时隙内传输，将严重影响性能，其信道效率为：
 
$$\frac{\text{帧长}64 \text{ B}}{\text{扩展成}512 \text{ B} + \text{帧前导}8 \text{ B} + \text{帧间隙}12 \text{ B}} = \frac{64}{532} = 12\%$$
  - 64Byte的短帧扩展成512Byte，如冲突在64Byte之后（冲突在扩展的部分），帧的重发将是无意义的。
  - 当采用短帧突发技术时，让一个站发送多个帧，而只对第一个帧进行载波扩展，紧接着发送后面的帧，这些帧毋需载波扩展。



- 迟冲突不重发：发生在扩展位的冲突被认为是一次迟冲突，IEEE 802.3z规定发送方在检测到迟冲突之后不进行重传。

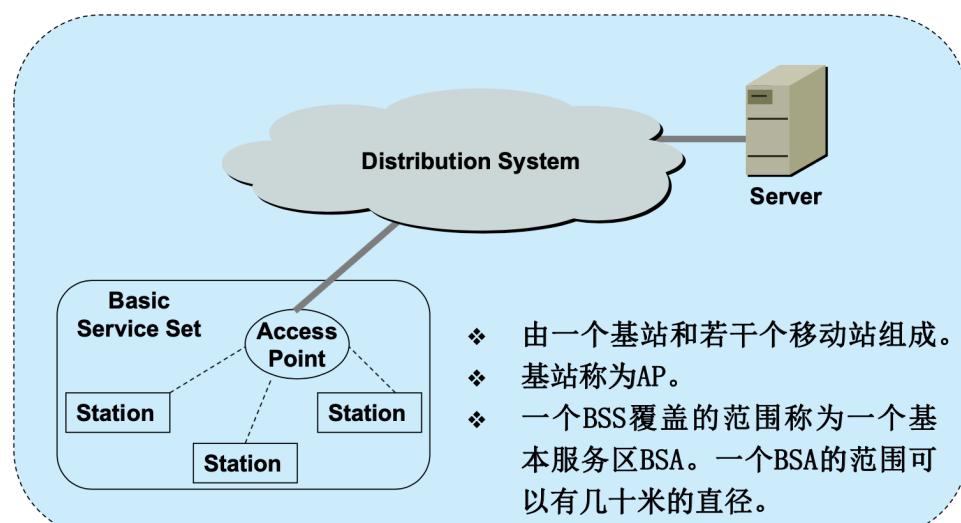
### 5.7.2.5 万兆以太网

- 万兆以太网
  - 只支持全双工
  - 不需要CSMA/CD

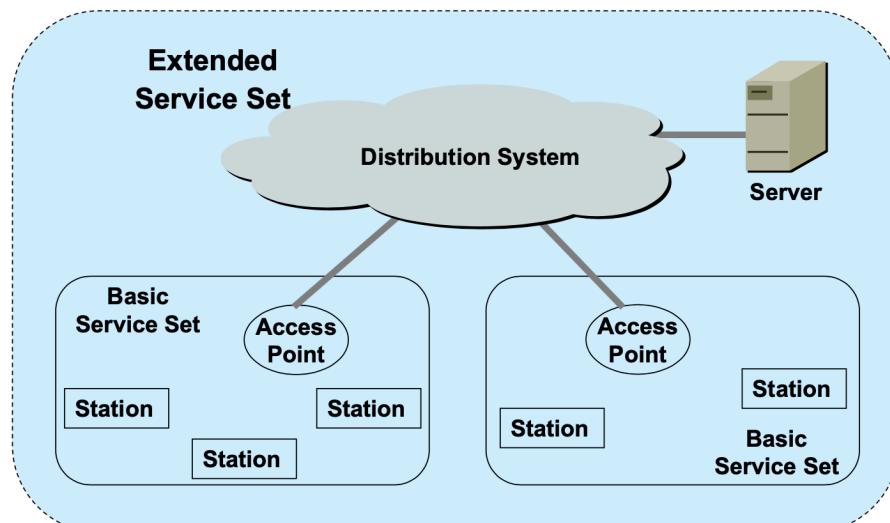
### 5.7.3 无线局域网

#### 5.7.3.1 802.11无线局域网的组网模式

- 802.11定义了两种组网模式
  - Ad-Hoc 模式：所有的移动站之间互相平等，每个节点既是主机又是路由器
  - Infrastructure 模式
    - 基本服务集合 BSS



- 扩展服务集合 ESS

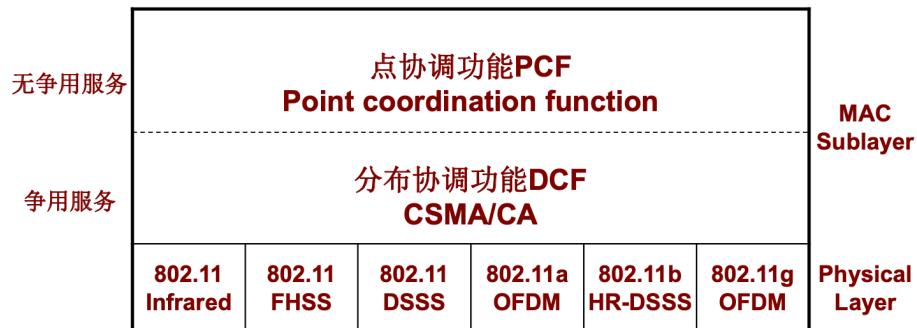


- ESS由多个BSS通过一个分布式系统DS互联而成，就像一个逻辑上的局域网。
- DS是一个有线主干局域网，通常表现为以太网。

- BSS之间的通信将通过DS实现，BSS在LLC子层上相统一，至此一个移动主机可以漫游在不同的BSS之间。

### 5.7.3.2 802.11 MAC层协议

- 802.11 MAC层协议



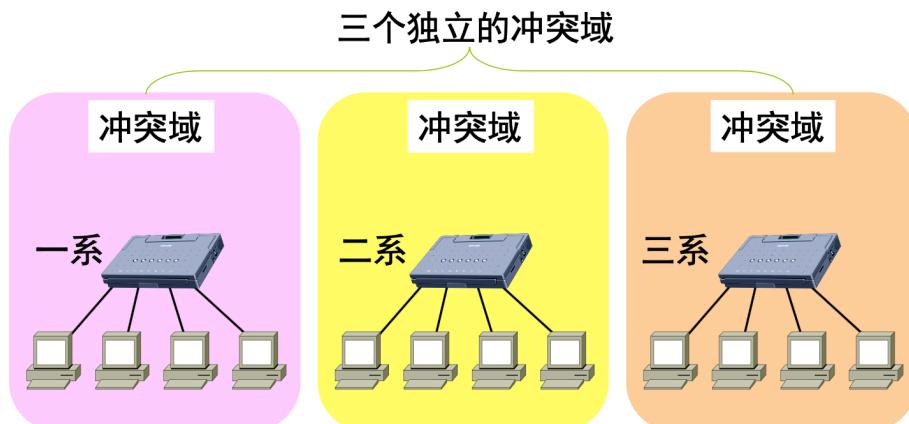
- 分布式协调功能DCF
  - DCF模型中每个站点都是相互独立的，没有主从关系，必须通过竞争获得信道。DCF采用CSMA/CA，即带冲突避让的载波多路监听CSMA/CA
- 无线局域网为什么不能使用CSMA/CD？
  - CSMA/CD要求每个站点在发送数据的同时还必须不间断地检测信道，而在无线局域网的设备中要实现这个功能花费过大。
  - 即使发送端能够实现碰撞检测，在接收端仍可能发生碰撞。
- CSMA/CA
  - 任何站在完成发送后必须等待一段很短的时间才能发送下一帧，这段时间称为帧间间隔IFS。间隔时间的长短取决于该站打算发送的帧类型。高优先级的帧等待时间短，低优先级的帧等待时间长。
  - 争用：当信道从忙转为空闲时，任何站在发送数据前，都要采用二进制退避算法减少发生冲突的概率。与以太网不一样的是第*i*次退避是从 $2^{2+i}$ 个时隙中选取一个。

## 5.8 数据链路层交换

### 5.8.1 局域网的扩展

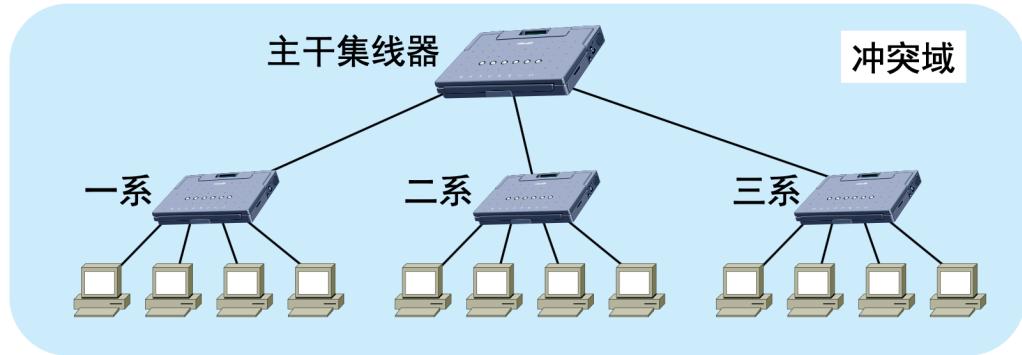
- 局域网的扩展即LAN互连
- 物理层上的局域网扩展——用多个集线器(HUB)可连成更大的局域网

某大学有三个系，各自有一个局域网



用集线器组成更大的局域网都在一个冲突域中，等价于所有机器共享信道。

## 一个更大的冲突域



- 优点

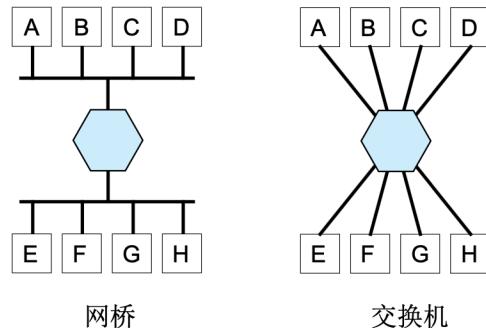
- 使原来属于不同冲突域的局域网上的计算机能够进行跨冲突域的通信。
- 扩大了局域网覆盖的地理范围。

- 缺点

- 冲突域增大了，但总的吞吐量并未提高。
- 如果不同的冲突域使用不同的数据率，那么就不能用HUB将它们互连起来。

- **数据链路层上的局域网扩展**——利用数据链路层上的交换设备，扩展局域网的规模

- 网桥
- 交换机



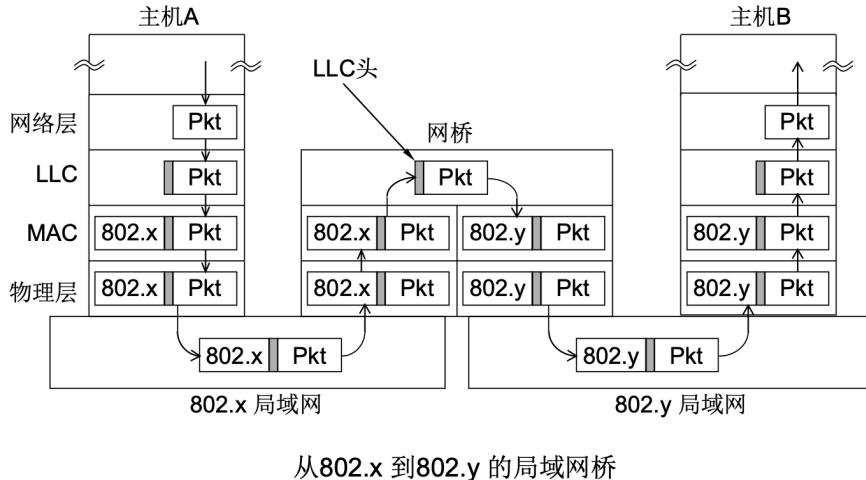
### 5.8.2 网桥

- 网桥

- 连接多个局域网，每个端口连接的所有主机称为一个**网段**。
- 所连接局域网数据链路层的MAC子层可不相同。

- 工作在**数据链路层**，能解析该层的帧，它根据 MAC 帧的目的地址对收到的帧进行存储和转发（不像HUB一样广播到其它点）

- **过滤帧**：当网桥收到一个帧时，并不是向所有的接口转发此帧，而是先检查此帧的目的MAC地址，然后再确定将该帧转发到哪一个接口。
- **转发依据**：转发表(MAC地址、接口)对。
- **MAC帧的转换**：当接收帧的网络类型与待发送到网络的类型不一致(不同协议)，进行MAC帧的转换。
- **MAC帧的传输速率调整**：当接收帧的网络与待发送网络的传输速率不一致，通过存储和转发过程，调整发送速率。



- 网桥的优点：

- 过滤通信量——使各网段成为隔离开的冲突域。
- 扩大了物理范围。
- 提高了可靠性。
- 可互连不同物理层、不同MAC子层和不同速率的局域网。

- 网桥的缺点：

- 存储转发增加了时延。
- 在MAC子层并没有流量控制功能。
- 若不同MAC子层的网段桥接在一起则时延更大。
- 网桥只适合于用户数不多和通信量不太大的局域网，否则可能因传播过多的广播信息而产生网络拥塞——广播风暴。

- 网桥与HUB的不同

- 集线器在转发帧时，不对传输媒体进行冲突检测——集线器所连的整个网络为一个冲突域。
- 网桥对接收帧进行过滤，在转发前必须执行CSMA/CD算法
  - 冲突域限制在所连的各个LAN内。
  - 若在转发过程中出现碰撞，就必须停止发送和进行退避。

- 网桥有以下类型：透明网桥、生成树网桥、远程网桥、源路由网桥

### 5.8.2.1 透明网桥

- 特点

- 透明：局域网上的站点并不知道所发送的帧将经过哪几个网桥，网桥对各站不可见。
- 即插即用设备——将网桥与相关的网络在物理上连接后，不需要做任何配置，即可实现网络互联。

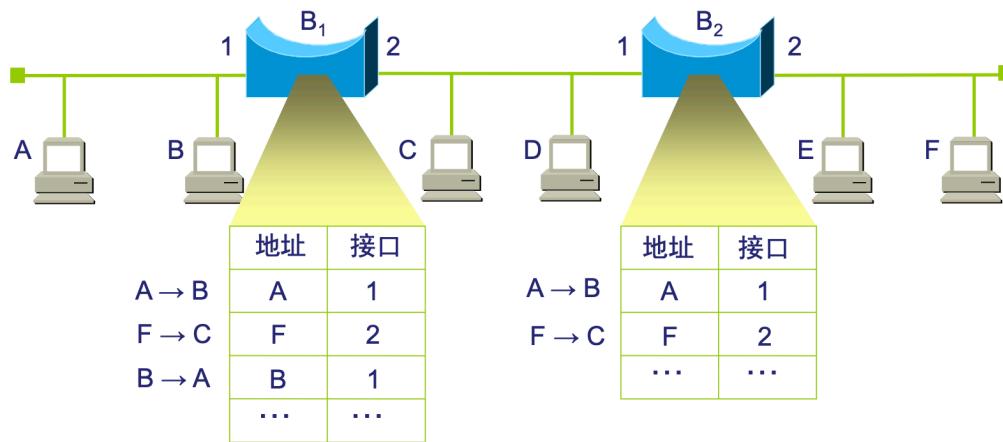
- 转发表的组成

- 转发表(MAC地址,接口,时间)
  - 以太网拓扑经常变化，站点可能更换适配器，站点并非总接通电源。
  - 把每个帧到达网桥的时间登记下来，在转发表中只保留网络拓扑的最新状态信息。

- 转发表的建立——自学算法

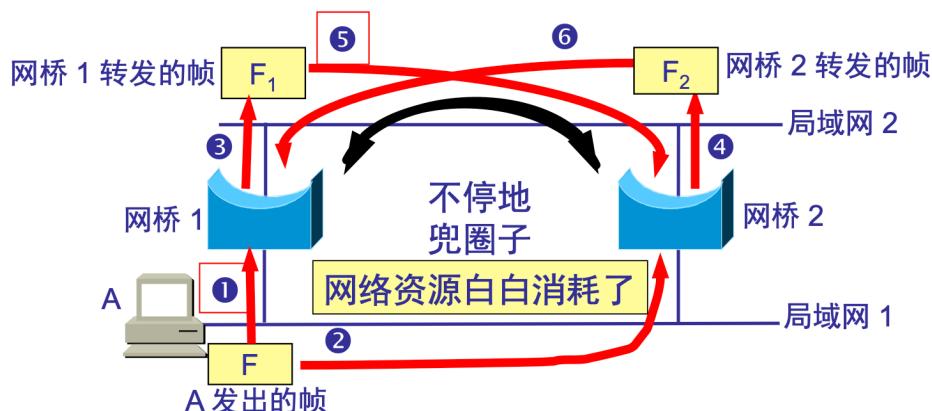
- 网桥每收到一个帧，检查源MAC地址：
  - 如不在转发表中，记下其源MAC地址、接收接口和时间，作为转发表中的一项。
  - 如在转发表中，更新时间。

- 接口管理软件周期性扫描转发表中的项目，删除在一定时间前登记的信息。
- 网桥接收一帧后的处理
  - 进行自学习以更新转发表，即在转发表中新增一项，或对原有项的时间进行更新。
  - 转发帧。查找转发表中与收到帧的目的地址有无相匹配的项：
    - 如没有，则通过所有其他接口转发该帧——扩散算法。
    - 如有，对比转发表对应项中的接口与该帧进入网桥的接口
      - 不同，按转发表中给出的接口进行转发。
      - 相同，应丢弃这个帧。



### 5.8.2.2 生成树网桥

- 若存在回路的拓扑结构，常规算法的透明网桥不能正常工作。



- 生成树算法

- 在物理上存在回路的拓扑中，生成一棵在逻辑上无回路的树，即生成树。
  - 根：最小序号或最高优先级的网桥。
  - 以最短路径为依据，为每个网段选定唯一网桥，使该网桥到根的路径最短。
  - 选定的网桥，指向根方向的端口为“根端口”，到其它网桥的端口为“指定端口”，这两个端口为转发状态，其它端口阻塞。
- 互连在一起的网桥进行彼此通信后，找出原网络拓扑的一个子集，整个连通的网络中不存在回路，即在任何两个站之间只有一条路径。
- 生成树必须能够反映网络拓扑的变化，根网桥每隔一段时间对生成树的拓扑进行更新。

### 5.8.2.3 远程网桥

- 远程网桥
  - 连接多个远距离的LAN。
  - 每对网桥间采用点到点线路连接(可看作是一个没有主机的LAN)。
  - 点到点线路上可选用PPP。
  - 网桥的2种帧转换：
    - 将MAC帧封装到点到点协议帧中，作为点到点协议帧的净载荷。
    - 源网桥处将MAC头和尾剥掉，剩下的部分放在点到点协议帧的净载荷中，目的网桥处重新生成一个新的MAC帧。

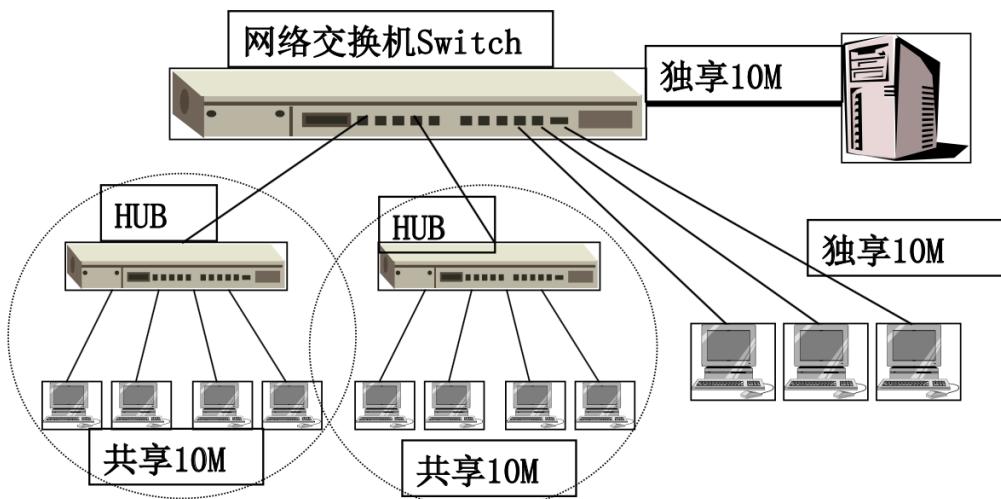
### 5.8.2.4 源路由网桥

- 发送帧的源站负责帧的路由选择，源路由网桥在发送帧时将路由信息放在帧首部中，依据该路由信息转发帧。
- 源站的路由学习
  - 单路由广播：单路由广播帧，转发时受生成树限制，在每一网段上仅出现一次。
  - 全路由广播：
    - 源站以广播方式向目的站发送一个全路由广播帧，每个发现帧都记录所经过的路由。
    - 发现帧到达目的站时就沿各自的路由返回源站。
    - 源站在得知这些路由后，从所有可能的路由中选择出一个最佳路由。
    - 从该源站向该目的站发送的帧，其首部都携带源站所确定的路由信息。
- 源路由网桥的帧转发过程
  - 源路由网桥无须保存路由表，只需记住自己的地址标识符和它所连接的LAN标识符。
  - MAC帧中包含一条路由：LANID、网桥ID的序列。
  - 仅当路由信息包含 $LAN_i-B_x-LAN_j$ 时，网桥才转发帧
    - $LAN_i$ : 帧到达的LAN
    - $B_x$ : 本网桥ID
    - $LAN_j$ : 本网桥连接的另一个LAN

### 5.8.3 交换机

- 以太网交换机实质上是一个多接口的网桥。
- 每个接口都直接与一个单个主机或集线器相连，工作在全双工方式。当主机需要通信时，交换机能同时连通多对接口，使每一对相互通信的主机都能像独占通信媒体一样，无碰撞地传输数据。
- 每个接口都有自己的冲突域，交换机永远不会由于冲突而丢弃帧。
- 即插即用设备，内部转发表也是通过自学习算法自动逐渐建立起来的。
- 使用专用的交换结构芯片，其交换速率高。
- 交换机的帧转发方式
  - 直接交换方式：只要接收并检测到目的地址字段就立即将该帧转发出去，而不管这一帧数据是否出错。(无法检错和调整速率)
  - 存储转发方式：交换机先完整接收数据帧进行差错检测再转发。(可检错和调整速率)

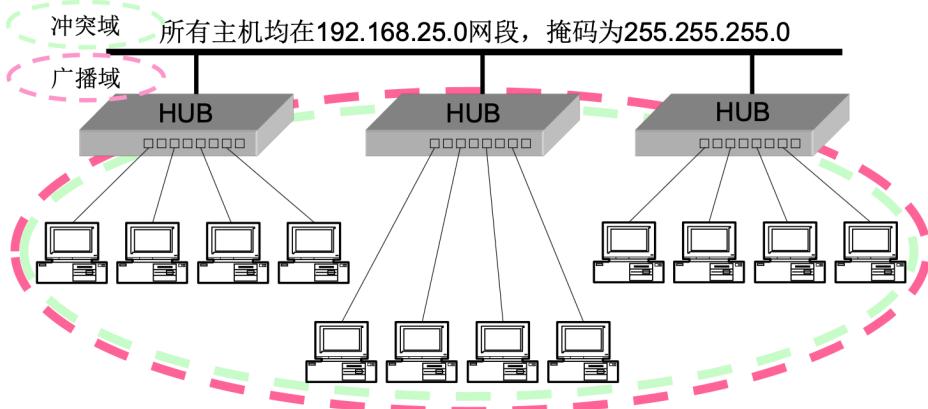
- 改进的直接交换方式：接收数据帧的前64字节，判断字头段正确再转发。
- 独占传输媒体的带宽
  - 对于普通 10 Mb/s 的共享式以太网(HUB)，若共有 N 个用户，则每个用户占有的平均带宽只有总带宽(10 Mb/s)的N分之一。
  - 以太网交换机时，虽然在每个接口到主机的带宽还是 10 Mb/s，但由于一个用户在通信时是独占而不是和其他网络用户共享传输媒体的带宽，因此对于拥有 N 对接口的交换机的总容量为 Nx10 Mb/s。（每对接口之间不是共享信道）
- 使用交换机后，可建立多个并发的通信。
  - 例如：8个端口可建立4个并发通信，总带宽 =  $(8/2)*10\text{Mbps} = 40\text{ Mbps}$
- 交换机的两种用法
  - 端口下接站点：站点独占10Mbps带宽
  - 端口下接网段(通过HUB)：网段中所有站点共享10Mbps带宽



## 5.8.4 虚拟局域网VLAN

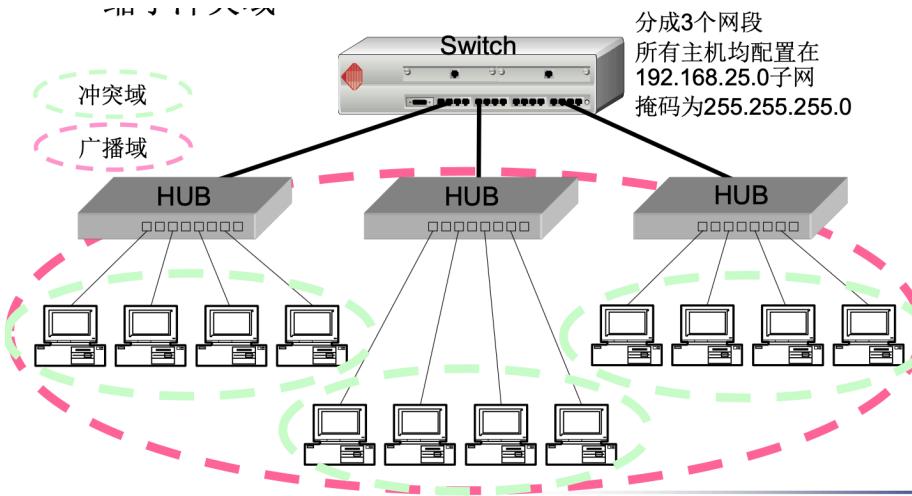
### 5.8.4.1 VLAN的应用背景

- 局域网的广播域、冲突域
  - 冲突域是共享信道的。



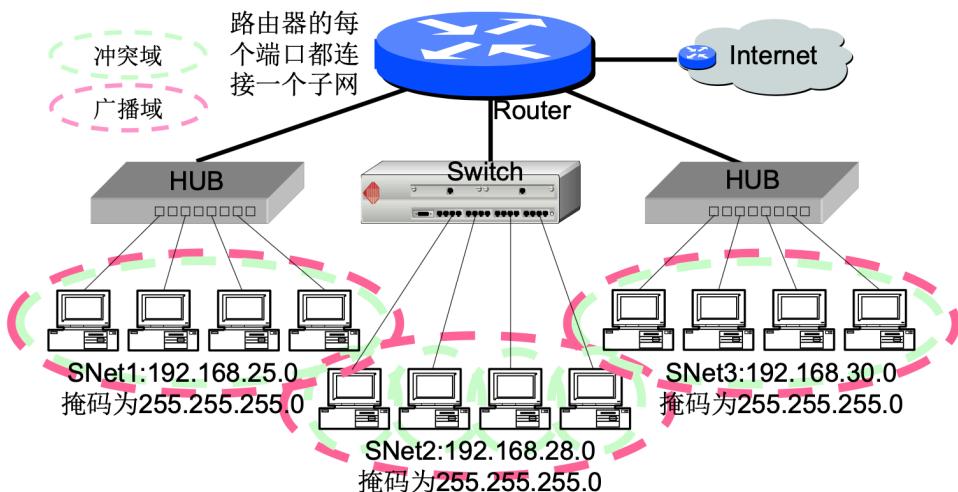
- 局域网的网段分隔：
  - 局域网中使用网桥和交换机来分隔网段。
  - 缩小冲突域。

- 网桥（交换机）不过滤广播帧。



- 局域网的子网划分：使用**路由器**来划分子网，并与Internet连接。

- 路由器阻断广播帧、广播IP报文。



- 网段和子网的区别：

- 网段(一个网段即一个冲突域)

- 将若干个网段通过网桥连接，以增加网络内的主机数并扩大覆盖范围，网桥的每个端口连接一个网段，**网段是链路层概念**。
- 一个网段内的主机为一个冲突域，所有的主机是一个广播域，不同网段的主机间的通信则由网桥负责转发或扩散。

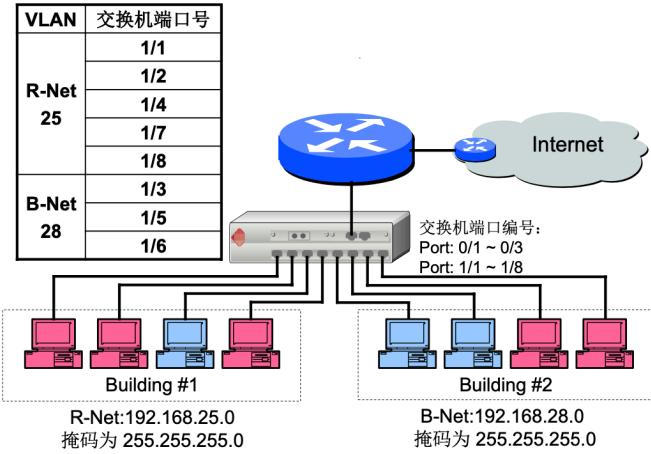
- 子网(一个子网即一个广播域)

- 将若干个子网通过路由器连接，以实现网络的互联并组成一个更大的网络，路由器的每个端口连接一个子网，**子网是网络层概念**。
- 一个子网就是一个广播域，子网间的通信必须由路由器控制。

#### 5.8.4.2 VLAN工作原理

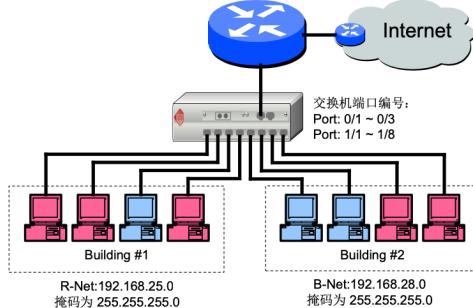
- VLAN采用交换机技术(**3层交换机**，之前所述为2层交换机)，将原有网络分成若干个逻辑上的**子网**，**逻辑子网**也具有物理子网相同的网络性能。
- 一个VLAN是一个广播域。
- VLAN基于交换技术

- 交换机将维护一张交换表
  - 交换表是交换端口的编号、MAC地址和所连接主机的MAC地址的关联表。
  - 交换表可以静态配置，也可以动态维护。
- 按端口划分VLAN(静态)



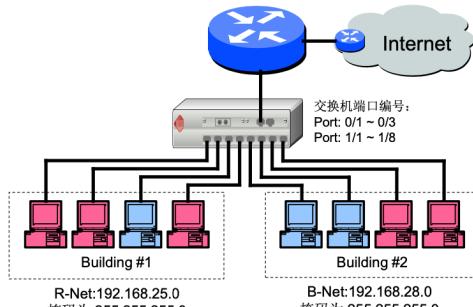
- 按MAC地址划分VLAN(动态)
  - 为每个注册用户(MAC)划分VLAN

VLAN	主机的MAC
R-Net 25	121235415121
	238923483756
	304537848445
	546372896745
	424872064821
B-Net 28	246879015426
	987461200455
	357878840212



- 按主机的IP地址划分VLAN(动态)
  - 为每个注册用户(IP)划分VLAN

VLAN	主机的IP地址
R-Net 25	192.168.25.41
	192.168.25.42
	192.168.25.43
	192.168.25.44
	192.168.25.45
B-Net 28	192.168.28.65
	192.168.28.66
	192.168.28.67



## 局域网VLAN划分举例

1. 每个交换端口为一个VLAN
2. 多个交换端口同属一个VLAN
3. 分属不同交换机的交换端口同属一个VLAN

- VLAN的作用
  - 提高网络的可管理性
    - 不同物理网络中的主机可定义在同一个VLAN内
    - 同一物理网络内的主机可定义在不同的VLAN中
  - 提高网络的安全性
    - 不在一个VLAN中的主机，无法监听
  - 有效地避免广播风暴，以提高信道利用率，并降低路由器的投资成本
    - 在一个物理网络(一个路由器端口连接的网络)内可划分多个逻辑子网
  - 减少主机因网络逻辑拓扑改变而在物理上的移动

## Chapter6 网络层

### 6.1 网络层概述

- 网络层的定义：
  - 在计算机通信系统之间提供建立、保持和终止网络连接的手段，并为传输层实体通过网络连接交换网络服务数据单元提供功能和规程方面的手段。
  - 通信子网的最高层。
- 网络层的功能：
  - 网络互联，屏蔽各种不同类型网络之间的差异
  - 路由选择，了解通信子网的拓扑结构，实现报文的网络传输
  - 流量控制和拥塞控制
  - 为上层提供服务，服务的设计目标为：
    - 与路由器技术无关，其数量、类型、拓扑结构对于传输层不可见。
    - 传输层获得的网络地址采用统一的编址方式，并允许跨越多个LAN和WAN。
- 网络层服务的类型
  - 面向连接的服务：将复杂的功能放在网络层，包括连接建立、数据传送、连接释放。
  - 无连接的服务：将复杂的功能放在传输层，不在端系统之间建立连接，所需要的仅是通信实体之间的关联。
- 通信子网提供的服务(面向连接或无连接)与通信子网结构(虚电路或数据报)没有必然联系。
- 服务原语：层服务用户和服务提供者之间的一种抽象且与实现无关的交互，不必直接与协议要素相关。
- 网络层的内部结构
  - 虚电路
    - 在发送前要建立连接，等待时间为RTT。
    - 连接请求包含目的地地址，但每个数据报文只携带很小的虚电路标识，开销很小。
    - 如果连接的链路或交换机故障，连接将被中断，必须建立新的连接。
    - 为预留资源提供了支持。
  - 数据报
    - 无需建立连接
    - 每个报文包含完整的目的地址
    - 每个报文独立地被转发
    - 转发表由路由协议动态生成

- 都属于分组交换，采用存储转发机制

- 虚电路子网/数据报子网的比较

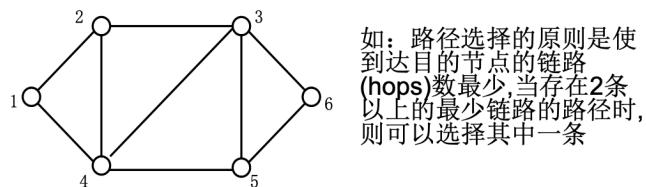
比较项目	数据报子网	虚电路子网
初始建立连接	不需要	需要
地址	每个分组都含有完整的源地址和目的地址	每个分组含有一个短的虚电路号，目的站地址仅在连接建立阶段使用
状态信息	子网不存储状态信息	已建立的虚电路占用子网路由表空间
路由选择	每个分组独立选择	虚电路一建立，路由就已确定，所有分组都经过此路径
分组的顺序	到达目的站时可能不按发送顺序	总是按发送顺序到达目的站
节点故障影响	除节点崩溃时会丢失分组外，无其他影响	所有经过失效设备的虚电路都会被终止
差错处理	由主机承担	对主机透明（由通信子网承担）
拥塞控制	由主机负责，较难实现	由通信子网负责，若每条虚电路分配有足够的缓冲区，则容易控制
功能复杂性部分	在传输层	在网络层
适用方式	面向连接和无连接服务	面向连接服务

## 6.2 路由选择

- 路由选择的定义：
  - 在分布式分组交换网中每个节点具有自动选择传送分组到达目的地的最佳路径的能力。
- 路由选择的要求：
  - 能正确、迅速、合理地传送报文信息。
  - 能适应网络内节点或链路故障而引起的拓扑变化，使报文在故障条件下一般仍能到达终点。
  - 能适应流量变化，使各通路的流量均匀，整个网络的通信设备负荷平衡。
  - 算法尽量简单，减少网络开销。

### 6.2.1 路由表

- 路由表：对每个目的节点指出分组应该发向的下一个节点。
- 数据报子网的路由表



节点1上的路由表						节点4上的路由表					
目的节点	2	3	4	5	6	目的节点	1	2	3	5	6
下一个节点	2	2	4	4	2	下一个节点	1	2	3	5	5

- 虚电路使用的路由表

- (节点1与节点6之间的三条双向虚电路：1-2-3-6、1-4-3-6、1-4-5-6)
- 路由表中使用的是虚电路号而不是目的节点

节点1上的路由表						
入虚电路	(-, -)	(-, -)	(-, -)	(2, 1)	(4, 2)	(4, 4)
出虚电路	(2, 1)	(4, 2)	(4, 4)	(-, -)	(-, -)	(-, -)
节点4上的路由表						
入虚电路	(1, 2)	(1, 4)	(3, 5)	(5, 4)		
出虚电路	(5, 4)	(3, 5)	(1, 4)	(1, 2)		

- 输入虚电路,(X,n)表示前一个节点X和虚电路号;
- 输出虚电路,(X,n)表示后一个节点X和虚电路号;
- (-,-)表示虚电路在这个节点上起始和或终止.

## 6.2.2 路由选择算法

- 路由选择算法的功能：源/宿对之间的路径选择，以及选定路由之后将报文传送到它们的目的地。
- 路由选择算法的要求：正确性、简单性、自适应性、稳定性、公平性、最优性。

### 6.2.2.1 随机路由选择

- 路由转发：随机选择一个出口转发。
- 缺点：效率低，时延大，可能造成某些分组长期中转，到达不了目的节点。
- 优点：简单，较好的健壮性，与网络拓扑结构无关，到达目的地的可能性大。

### 6.2.2.2 洪泛式路由选择

- 路由选择
  - 完全扩散：将收到的每一个分组，向除了该分组到来的线路外的所有输出线路发送。
  - 选择扩散：向着目的方向选择几条链路发送分组，依据是必须满足某些事先确定的条件。
- 优点：可靠性高，路径最短，常用于军事。
- 缺点：重复数据包多，浪费带宽。
- 解决方法：
  - 在包头设一计数器，每经过一个节点减1，计数值为0时则丢弃。
  - 在每个节点上建立登记表，数据包再次经过时丢弃。

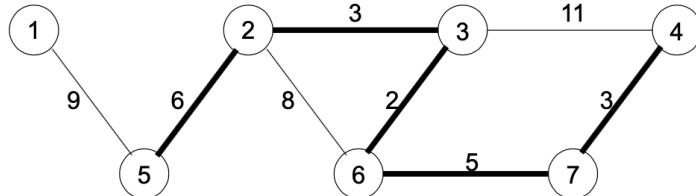
### 6.2.2.3 固定式路由选择(静态路由)

- 固定式单路由算法(绝对固定式路由选择)：
  - 各节点都有一张人工计算得到的**固定路由表**，给出了子网中各节点作为目的节点时，分组对应的转发出口。
  - 优点：简单、实现方便，可选择正常情况下的最佳路由。
  - 缺点：路由表不能联机修改，不能适应网络的业务量变化和拓扑变化。
- 固定式多路由算法(迂回式路由选择)：
  - 任何一对节点之间有多条可选路由，一旦最佳路由不通或负荷过大，就可以选择第二路由。
  - 实现方法：每个节点装有一张路由表，对应每个目的节点，给出最佳、次佳、再次佳……的后继节点和权数。
  - 缺点：路由表不能联机修改。
- **最短路径法**
  - 性质：

- 最短路径(A, B)上的某一段(Na, Nb)也是最短路径。
- 实例: Ford-Fulkerson算法、Dijkstra算法、floyd算法
- Dijkstra算法
  - 用边的权值作为距离的度量来计算最短路径。

如下图:

5和4之间边数最少的路径是5234, 但最短路径是523674



从5出发到各个节点的最短路径

S	R1	D1	R2	D2	R3	D3	R4	D4	R6	D6	R7	D7	u
{1,2,3,4,6,7}	5	9	5	6	0	$\infty$	0	$\infty$	0	$\infty$	0	$\infty$	2
{1,3,4,6,7}	5	9	5	6	2	9	0	$\infty$	2	14	0	$\infty$	1
{3,4,6,7}	5	9	5	6	2	9	0	$\infty$	2	14	0	$\infty$	3
{4,6,7}	5	9	5	6	2	9	3	20	3	11	0	$\infty$	6
{4,7}	5	9	5	6	2	9	3	20	3	11	6	16	7
{4}	5	9	5	6	2	9	7	19	3	11	6	16	4

#### 6.2.2.4 自适应路由选择

- 孤立的自适应路由算法

- 路由表的更新: 依据本节点自身的当前运行状态(流量、排队)信息来决定路由, 与其他各节点的状态无关。
- 特点: 算法简单, 但是适应能力有限, 不能适应网络各节点或链路状态的变化。
- 节点必须具备:
  - 一张预置于本节点的固定式路由表。
  - 各输出链路的目前状态(通或断)。
  - 等待发送的各链路排队长度。
  - 选路程序。
- 实例
  - 最短排队等待法: “热土豆”法, 仅考虑节点的排队状态。
  - 最短排队加偏法: 考虑了本节点排队状态和网络拓扑(如链路数)两种因素。

- 分布式自适应路由选择

- 路由表的更新: 在网络相邻节点之间, 每经过一定时间相互交换一次状态信息, 各节点根据相邻节点送来的状态信息来修改自己的路由表。
  - 经过n次修改即可反映出第n级相邻节点的状态改变。
- 特点: 可适应整个网络的状态变化, 附近节点处的状态较快得到反映, 远处节点的状态较慢得到反映。
- 实例: 距离矢量算法(D-V)、链路状态算法(L-S)

## ● 距离矢量算法(D-V)

- 工作原理：

- 各路由器用两个向量 $D_i$ 和 $S_i$ 来表示该点到网上所有节点的最短路径距离及其下一个节点。
- 相邻路由器之间交换路径信息。
- 各节点根据相邻路由器传递来的路径信息更新路由表。

$d_{i1}$ : 从节点*i*到节点1的最短路径

$d_{i2}$ : 从节点*i*到节点2的最短路径

$$D_i = \begin{bmatrix} d_{i1} \\ d_{i2} \\ d_{i3} \\ \dots \\ d_{in} \end{bmatrix} \quad S_i = \begin{bmatrix} s_{i1} \\ s_{i2} \\ s_{i3} \\ \dots \\ s_{in} \end{bmatrix}$$

$s_{i1}$ : 从节点*i*到节点1的一条最短路径上的下一个节点

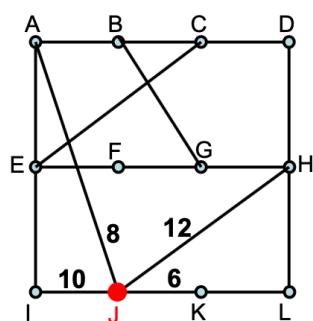
$s_{i2}$ : 从节点*i*到节点2的一条最短路径上的下一个节点

- 路由表的更新

$$d_{ij} = \min(d_{ix} + d_{xj}) (x \in A)$$

$$s_{ij} = x$$

- $A$ —与*i*相邻的所有节点的集合
- $d_{ij}$ —*i*到*j*的最短距离
- $d_{ix}$ —*i*到*x*的最短距离
- $d_{xj}$ —*x*到*j*的最短距离



注意： $A \rightarrow I$ 为21； $I \rightarrow A$ 为24

因为：往和返的信道流量不一定相同，节点A和I也并非在同一时刻测得，且线路状态是动态变化的

所谓节点即路由器  
当前节点为J

J重新估计的最短距离

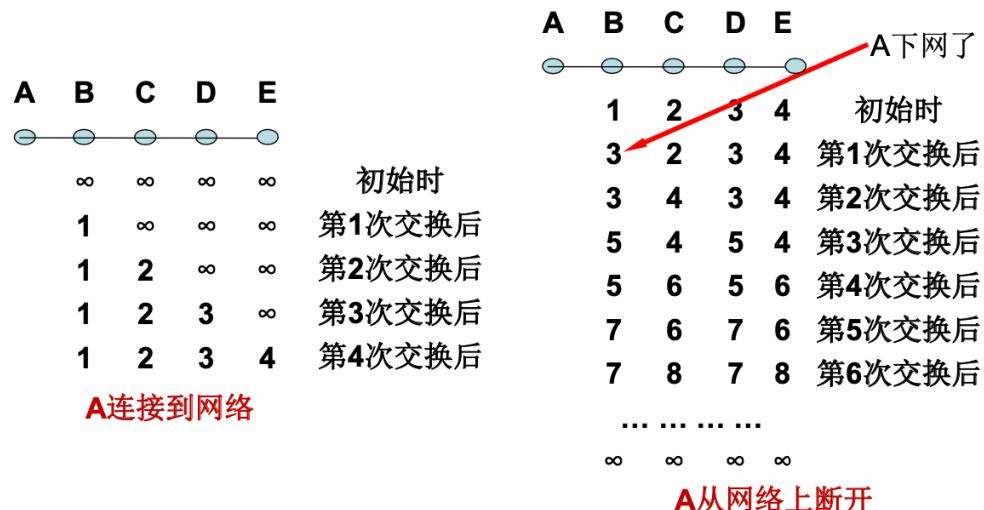
To	通过A	通过I	通过H	通过K	↓	线路
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	J到A 最短 距离 为8	J到I 最短 距离 为10	J到H 最短 距离 为12	J到K 最短 距离 为6	节点J的新路由表	

D-V算法的路由表更新

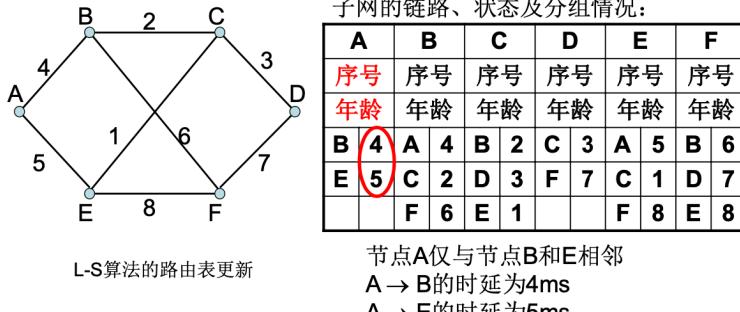
- D-V算法的缺点

- 交换的路径信息量大
- 路径信息不一致

- 不适合大型网络
- 收敛速度慢(坏消息)
- 无穷计算问题: 好消息传播得快, 坏消息传播得慢



- 克服收敛速度慢的方法
  - 水平分裂——防止路由器向邻居返回一个从该邻居获得的最佳路径
    - 对下方点通知真正的距离, 对上方点, 给出无穷大。
    - 如上图中的C点, 它向D通知到A的是真正距离, 而向B通知到A的距离是无穷大。
  - 抑制算法
    - 当发现与相邻节点不通时, 不重新选路径, 而是把它设成无穷大。
- 链路状态算法(L-S)
  - 基本思想:
    - 发现它的邻接节点, 并得到其网络地址
    - 测量它到各邻接节点的延迟或开销
    - 组装一个分组以告知它刚知道的所有信息
    - 将这个分组发给所有其他路由器
    - 计算到每个其它路由器的最短路径
  - 发现邻接节点
    - 路由器启动后, 向每个点到点线路发送HELLO分组(携带自己的网络地址), 另一端的路由器返回答来说明它是谁, 即通报其网络地址。
  - 测量线路开销
    - 发送一个ECHO分组要求对方立即响应, 测量来回时间除以2, 发送方得到一个延迟估计值 (可重复取平均)。
  - 构造链路状态分组
    - 节点到其邻节点的线路开销测量值(即延时, 假设以ms计)。



- 分发链路状态分组

- 用扩散法(向邻接的节点)发布链路状态分组。
- 以B为例, B的邻接点有A、C、F。
- 第三分组: 源节点E的链路状态分组经A和F到节点B, B将E的链路状态分组转送到C, 并向A和F发ACK。

源	序号	年龄	发送标志			ACK标志			数据
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

路由器B 中的状态分组缓冲区

- 计算新路由

- 用Dijkstra算法, 生成一棵以该路由器为根节点的生成树。
- 得到该路由器节点到其它每个路由器节点的最短路径, 即路由。

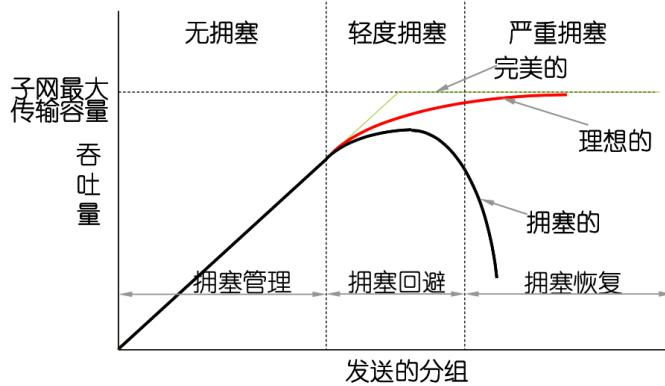
- L-S路由算法的优缺点

- 优点
  - 路由信息的一致性好, 坏消息也一样传播得快。
  - 状态分组的长度较短, 仅包含到邻接点的距离、序号和年龄等, 与网络规模关系不大, 传输所耗用的网络带宽不大。
  - 由于年龄参数的设定, 状态分组不会无限制扩散, 可适用于大型网络。
- 缺点
  - 路由器需要较大的存储空间, 用以存储收到的每一个节点的链路状态分组。
  - 计算工作量大, 每次都必须计算最短路径。

## 6.3 网络拥塞控制

### 6.3.1 网络拥塞的原因

- 网络拥塞: 通信子网中有太多的分组, 导致其性能降低。
- 网络拥塞的原因
  - 存储空间不足, 但单纯增加存储容量, 因会导致超时重发, 拥塞会更加严重。
  - 带宽容量不足。
  - 处理器处理速度慢。
- 吞吐量增大将发生拥塞



### 6.3.2 拥塞控制的基本原理

- 开环控制
  - 通过良好的设计来避免拥塞问题的出现，确保问题在一开始就不会出现。
  - 开环控制的方法
    - 什么时候接受新的数据报
    - 什么时候开始丢弃分组，丢弃哪些分组
    - 制定网络中各个节点的调度策略
  - 所有这些决定与网络中的当前状态无关
- 闭环控制
  - 基于反馈的拥塞控制
  - 分为三个阶段：
    1. 由监视系统检测何时何地发生了拥塞
    2. 将拥塞信息传送到拥塞控制点
    3. 调整系统操作以更正系统
  - 闭环控制的反馈方式
    - 显式反馈：如某一节点发现拥塞，它发一个回答帧给相关节点，通知它们网络拥塞了。
    - 隐式反馈：如通过定时器方法，发送端每发一个帧就启动一个定时器，若在规定的时间内没有收到相应的ACK，则认为该帧丢失，若丢失率相当高，则认为网络发生拥塞。
  - 一般不适合于高速网，因为等反馈的控制信号到达，早已时过境迁。

### 6.3.3 拥塞预防策略

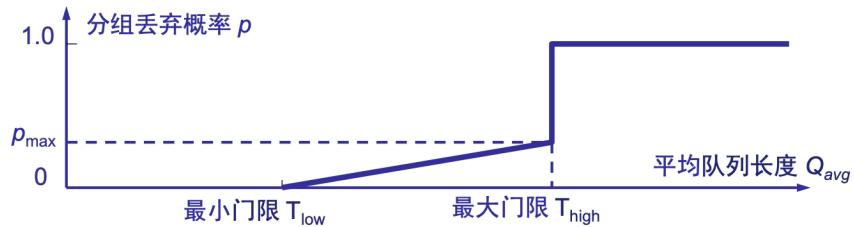
- 传输层、网络层、数据链路层的传输策略都会对拥塞产生影响。

#### 6.3.3.1 虚电路子网中的拥塞控制

- 通过准入控制：一旦拥塞出现，不允许建立新的虚电路。
- 如果允许建立新的虚电路，则在路由选择时要绕过拥塞地段。
- 虚电路建立时，在主机和子网之间协商一个协议，该协议指出了新建流的量、特征、服务质量和其他参数，子网将为该数据流预留资源。

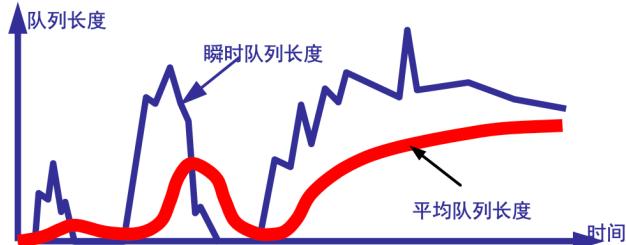
### 6.3.3.2 数据报子网中的拥塞控制

- 路由器的缓冲队列管理
  - 被动队列管理
    - 拥塞发生后采取措施：尾丢弃、随机丢弃、头丢弃。
  - 主动队列管理
    - 在拥塞发生之前采取措施。
    - 随机早期检测RED：使用平均队列长度来度量网络拥塞的程度，并反馈给端系统。
- 随机早期检测RED
  - 路由器的队列维持两个参数：队列长度最小门限 $T_{low}$ 和最大门限 $T_{high}$ 。
  - RED对每一个到达的数据报，都先计算平均队列长度 $Q_{avg}$ 
    - 若平均队列长度小于最小门限 $T_{low}$ ，则将新到达的数据报放入队列进行排队。
    - 若平均队列长度超过最大门限 $T_{high}$ ，则将新到达的数据报丢弃。
    - 若平均队列长度在最小门限 $T_{low}$ 和最大门限 $T_{high}$ 之间，则按照某一概率 $p$ 将新到达的数据报丢弃。



- 平均队列长度

$$\text{新}Q_{avg} = (1 - W_q) \times (\text{旧}Q_{avg}) + W_q \times (\text{当前队列长度样本})$$



- 路由器的拥塞通知
  - 警告位方法：当发生拥塞时，在分组的头部设置一位警告位，当该分组到达目的主机时，传输实体将警告位拷贝到ACK上，源主机能得知拥塞的发生，以调整它的发送速率。
  - 抑制分组：拥塞的路由器直接发抑制分组给源主机。
  - Hop-by-Hop阻塞包：直接发抑制分组给源主机可能反应太慢，取而代之的是发给它的前一站路由器，通知它放慢速率，前一站再发给它的前一站，一直到源主机。

## 6.4 网络流量控制

- 流量控制的功能
  - 防止由于网络和用户过载而产生的吞吐量降低及响应时间增长(防止网络拥塞)
  - 避免死锁：储存-转发死锁、分组重装死锁
  - 在用户之间合理分配资源
  - 网络及其用户之间的速率匹配

- 流量控制技术

- 集中式

- 指定网络控制节点收集当前报文流信息，并计算各节点的报文流量分配
    - 各节点依据报文流量分配，限制报文的输入率

- 分布式

- 由多个或每个网络节点来控制从外部新进入的业务量

- **流量控制与拥塞控制的区别和联系**

- 区别

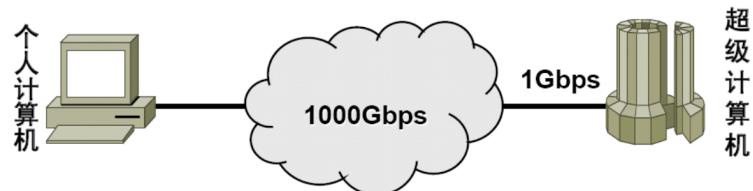
- 流量控制

- 只在一对给定的发送方和接收方之间。
      - 调节某发送点和接收点之间的通信量。

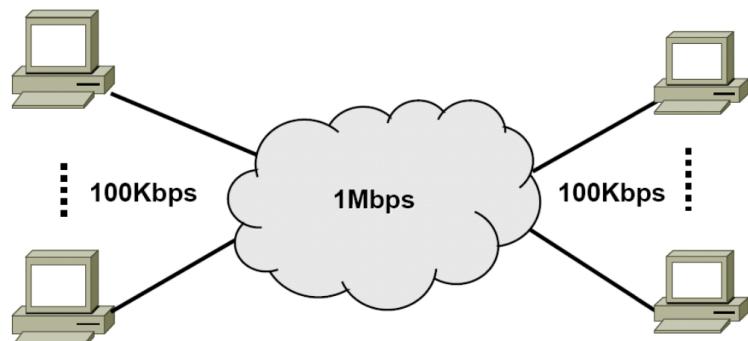
- 拥塞控制

- 全局性过程，涉及到网络中所有主机、路由器的行为。
      - 必须使得通信子网能够传送所有待传送数据。

- 联系：流量控制限制了进入网络的信息总量，可以在一定程度上缓解网络拥塞。

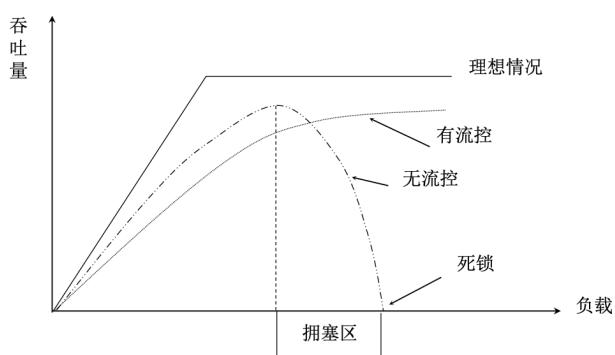


**必须进行流量控制**



**无需流量控制，但要拥塞控制**

- 流量控制对拥塞的减缓作用



## 6.5 网络服务质量

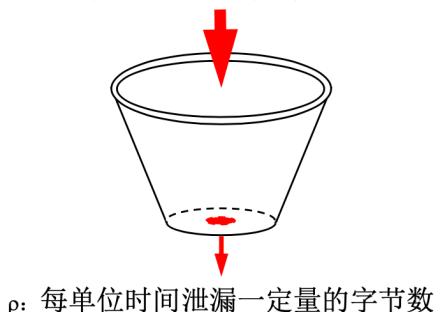
### 6.5.1 服务质量的需求

- 流：具有同样的源 IP 地址、源端口号、目的 IP 地址、目的端口号、协议标识符以及服务质量需求的一连串分组。
- 流的服务质量有四个指标：可靠性、延迟、抖动、所需带宽。
- 提供服务质量的方法：
  - 提供充足的资源。
  - 提供均衡路由：当源站点到目的站点有多条路径时，通常的做法是选一条最佳路径，均衡路由是把流量分散到多条路径上。

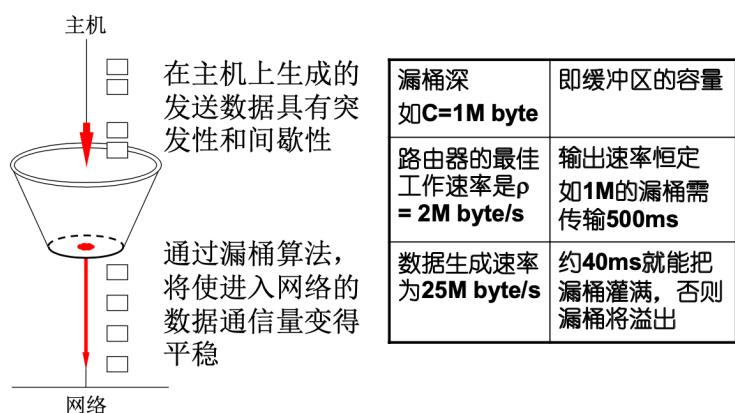
### 6.5.2 流量整形

- 流量整形：调节进入网络的数据流的平均速率和突发性。
  - 允许应用程序可发送各种各样的流量，包括带有某种程度的突发。
  - 可减少拥塞，有助于服务提供者兑现服务承诺。
- 漏桶
  - 平滑输入流量，控制进入网络的流量。

(漏桶对应于一个容量有限的内部队列)



- 漏桶算法中的主机与网络



- 漏桶算法突发时间长度的计算

- 设 $C$ 为漏桶容量(Byte, cell数)
- $\rho$ 为漏桶的输出速率(Byte/s)
- 则突发速率 $r$ 与允许突发时间的长度 $S$ 的关系为：

$$S = \frac{C}{r - \rho}$$

<b>漏桶深 <math>C = 1M\text{ byte}</math></b>	<b><math>C = 1M</math></b>
<b>路由器的最佳工作速率是 <math>\rho = 2M\text{ byte/s}</math></b>	<b><math>\rho = 2M</math></b>
<b>数据生成速率为<math>25M\text{ byte/s}</math> <math>r = 25M/s</math></b>	允许有 <b>43.5ms</b> 的突发数据

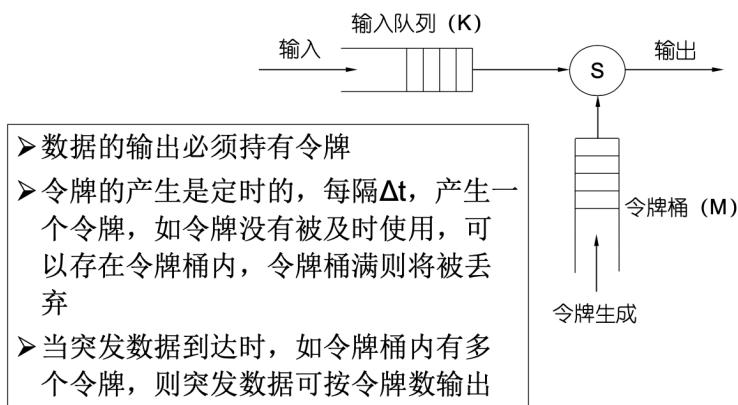
允许突发时间的长度  $S = C/(r-\rho) = 1/(25-2) = 43.5(\text{ms})$

- 缺点：

- 漏桶满时，造成数据丢失。
- 强迫输出保持一个固定的平均速率，不能体现通信量的突发。

- 令牌桶

- 当大的突发流量到来时，输出也能有适当响应



- 令牌桶突发时间长度的计算

- $V$ 为令牌桶的容量， $\rho$ 为令牌到达速率， $M$ 为最大的输出速率(数据到达速率  $> M$ )
- 则最大的突发时间长度 $S$ 为： $V + \rho S = MS$ ，即  $S = V/(M - \rho)$

### 6.5.3 包调度

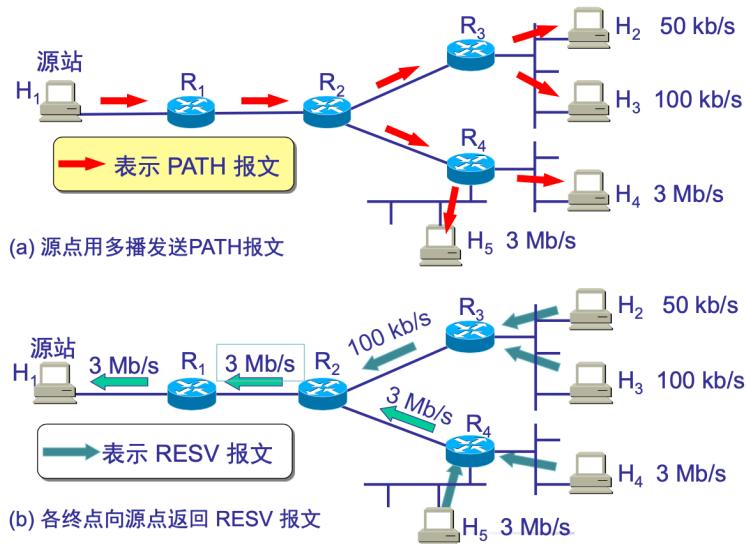
- 包调度算法：假定一个流的数据遵循同样的路径，在同一个流的数据包之间以及竞争流之间分配路由器资源
- 类似于虚电路方式
- 包调度算法的分类
  - 先进先出/先来先服务
    - 实现简单，当存在多个流时，一个流会影响其它流的性能
  - 公平队列
    - 路由器为每个流设置单独的队列，按数据包/字节为发送单位，路由器循环扫描各队列
  - 加权公平队列
    - 为每个流分配一个权值，依据流的权值分配每个流占用的出口带宽
  - 优先级队列
    - 每个数据包标记一个优先级别，高优先级数据包先于低优先级数据包发送

#### 6.5.4 准入控制

- 准入控制：网络根据流量特征及所需的资源决定是否接受该数据流。
- 流规范：一组参数，用来描述流的特征，网络根据这些特征确定所需的资源。
- 流参数的协商：发送方生成一个流规范，当这组参数沿着路径传播时，沿途的路由器都会修改这组参数（只能降低，不能提高），当到达接收方就建立起流参数。

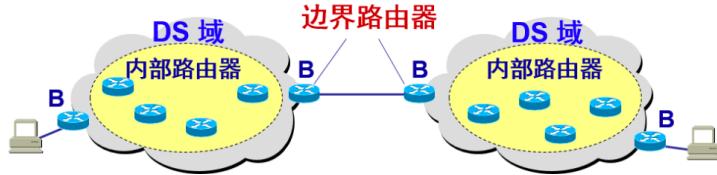
#### 6.5.5 综合服务

- 综合服务(IntServ)可对单个的应用会话提供服务质量的保证，其主要特点有：
  - 资源预留：路由器需要知道不断出现的会话已预留了多少资源。
  - 呼叫建立：需要服务质量保证的会话必须首先在源站到目的站的路径上的每个路由器预留足够的资源，以保证其端到端的服务质量要求。
- IntServ定义了两类服务：
  - 有保证的服务：可保证一个分组在通过路由器时的排队时延有一个严格的上限。
  - 受控负载的服务：可以使应用程序得到比通常的“尽最大努力”更加可靠的服务。
- IntServ的四个组成部分：资源预留协议 RSVP、接纳控制、分类器、调度器。
- 资源预留协议RSVP：
  - 将在沿途的路由器上预留一定的资源，包括带宽、缓冲区、表空间等。
  - RSVP是一种基于接收端，并由接收端发起的资源预留协议
    - 接收方沿着组播路由生成树，给组播发送方发送一条资源预留消息。
    - 在沿途每一个路由器依据该预留消息，预留带宽。
    - 如没有足够带宽，路由器就返回报告失败。



#### 6.5.6 区分服务

- 区分服务DiffServ
  - 区分服务可由一组路由器提供，这组路由器形成一个管理域。
  - 每个域中定义一组服务级别，即一组转发规则。
  - 如果一个客户已订购了区分服务，那么进入到该管理域的客户数据包会被标记上服务类型字段，用来表示所需的服务级别。



- 区分服务的特点
  - 不需要为每一流进行端到端的协商和资源预留
    - DiffServ不是为网络中的每一个流维持供转发时使用的状态信息，而是将若干个流根据其DS值聚合成少量的流。
  - 区分服务仅为单跳行为，由该管理域内的路由器提供服务，而不是整个网络的服务保证。
  - 路由器对相同DS值的流都按相同的优先级进行转发，简化了网络内部路由器的转发机制。

### 6.5.7 标签交换和MPLS

- 标签交换：
  - 在传统的IP网络中添加了面向连接的特性，是在尽力而为的传统网络中将数据流按照不同分类选择不同的路径。
  - 类似于虚电路方式，在数据包头上加一个标签，一般为转发表中的索引。
  - 在中间路由器上根据标签而不是根据目的地址作路由，这样可加快转发速度。
- MPLS：
  - 独立于网络层和链路层，将无连接的数据报转化为面向连接的虚电路交换——2.5层。
- MPLS网络的组成
  - 标记交换路由器(LSR)
  - 标记边缘路由器(LER)
  - 标记交换路径(LSP)
- MPLS的工作过程
  - 建立连接：形成标记交换路径的过程——建立路由表
    - 逐跳路由：每个节点独立为每个转发等价类(FEC)选择下一跳节点。
    - 显式路由：出/入口LER决定LSP的LSR序列。
  - 数据传输：入口LER给IP分组加上标记(将其划分到一组FEC)，沿途LSR按标记进行转发，出口LER去掉标签。
  - 拆除连接：释放LSP。

## 6.6 网络互联

### 6.6.1 网络互联的概念

- 网络互联：两个以上的计算机网络，通过一定的方式用一种或多种通信处理设备相互连接起来，以构成更大的网络系统。

### 6.6.2 网络互联的功能

- 扩大网络通信范围与限定信息通信范围
  - 小网互连实现共享资源
  - 大网分成互连的小网减少全网的通信量
- 不同网络之间的连接

- 提高网络系统性能和系统可靠性
  - 隔离问题，提高网络的各种性能

### 6.6.3 网络互联的要求

- 提供网络之间的通信设施
- 提供不同网络上用户之间的路由选择
- 提供计费服务
- 在不改变互联网络的体系结构情况下提供各种必需的服务

### 6.6.4 网络互联的设备

- 网络互相连接起来要使用中间设备
  - 物理层中继系统：转发器、HUB
    - 对弱信号进行放大或再生，以便延长传输距离。
  - 数据链路层中继系统：网桥、交换机
    - 在局域网之间存储转发帧，可以改变帧格式。
  - 网络层中继系统：路由器
  - 网络层以上的中继系统：网关
    - 实现高层协议的转换。
- 网络互联使用路由器
  - 当中继系统是转发器或网桥时，一般并不称之为网络互联，这仅是将一个网络扩大了，仍然是一个网络。
  - 互联网都是指用路由器进行互连的网络。

#### 6.6.4.1 路由器

- 路由器的原理
  - 用于连接多个逻辑上分开的网络(逻辑网络代表一个单独的网络或者一个子网)。
  - 方便地连接不同类型的网络，只要网络层运行的是IP协议，就可通过路由器互联起来。
  - 利用网络地址(IP地址)来区别不同的网络，实现网络的互联和隔离，保持各网络的独立性。
- 路由器的工作
  - 为所经过的每个IP报文寻找一条最佳传输路径，并将该数据有效地传送到目的地。
  - 核心：路由选择算法和路由选择协议。
- 分组在互联网中的传送
  - 当主机A要向另一个主机B发送数据报时，先要检查目的主机B是否与源主机A连接在同一个网络上。
    - 如果是，就将数据报直接交给目的主机B而不需要通过路由器。
    - 如果不是，则应将数据报发送给本网络上的某个路由器，由该路由器按照转发表指出的路由将数据报转发给下一个路由器——间接交付。

### 6.6.5.2 三层交换机

- 三层交换机：集成了位于数据链路层的**交换机**和位于网络层的**路由器**两者的功能
  - 二层交换模块：负责MAC层的寻址和数据帧的传递
  - 三层交换模块：负责IP层的寻址
- 硬件转发表：**(目的IP地址、VLAN ID、端口、下一跳MAC地址)**
- 三层交换机的工作原理
  - 对于待转发的报文：
    - 若硬件转发表中有相应的转发项，按转发项转发——**二层快速转发**
    - 如硬件转发表中没有转发项，利用三层交换模块功能获得下一跳路由器IP地址，利用ARP广播请求包，获得下一跳路由器的MAC地址，建立起硬件转发表中转发项——**三层交换模块**
- 三层报文交换技术（类似于数据报，前后报文无关）
  - 每个报文都要经历第三层处理
  - 数据帧到达系统物理接口后，在数据链路层，搜索硬件转发表：
    - 如果有与目的IP地址相匹配的项，第二层转发。
    - 如果没有，进入**三层交换模块**，确定路由及进行其它服务，**再进入二层进行转发**。
- 三层流交换技术（类似于虚电路）
  - 流中仅第一个报文需要三层交换模块处理：识别流、确定路由
  - 其后，流中第一个报文及后继报文，依据选定的路由，在第二层转发
  - **直通路由技术，即路由一次，交换多次**

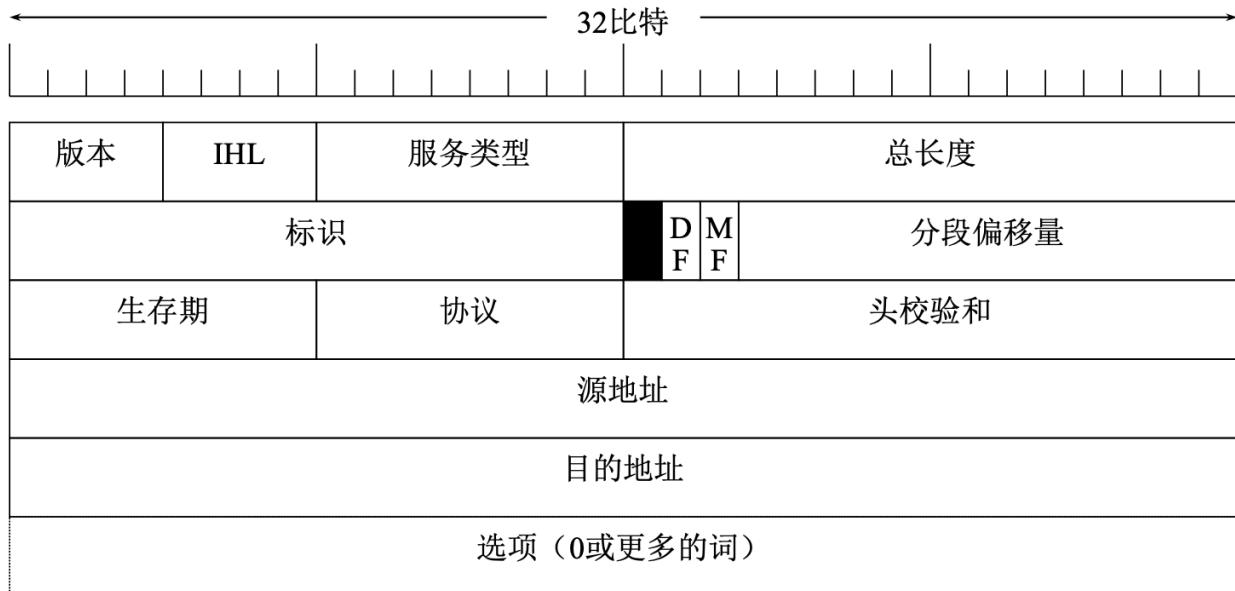
## 6.7 因特网中的网络层

### 6.7.1 IP协议

#### 6.7.1.1 IP协议簇

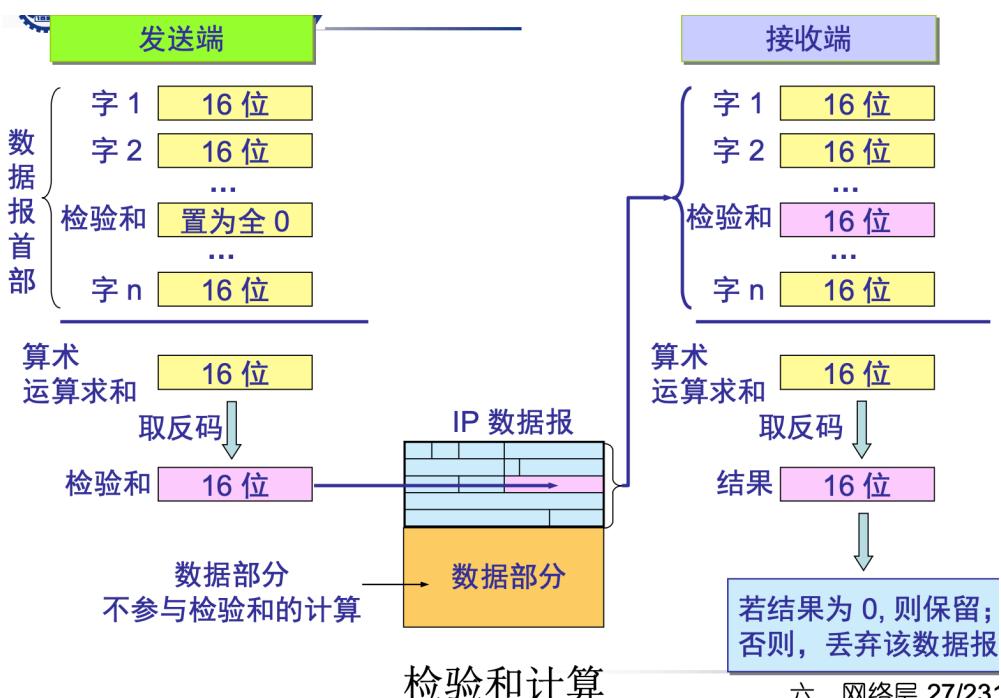
- 因特网的网络层协议：网际协议IP、地址解析协议 ARP、逆地址解析协议 RARP、网际控制报文协议 ICMP、网际组管理协议 IGMP
- IP协议**提供数据报服务**
  - 灵活性：支持面向连接和无连接服务的多种网络。
  - 健壮性：数据报独立选择路由，对拥塞反映迅速。
  - 对无连接应用的支持：与TCP配合，提供对无连接应用的可靠传输和流量控制。

#### 6.7.1.2 IP报文格式



## IP头部的组成

- IP头部：包括20个字节的固定部分和变长(最长40字节)的可选部分。
- IHL：IP报文头长度，最小为5，最大为15，单位为32 bit。
- 总长度域：包括头部。
- DF：不能分段，所有机器必须能够接收小于等于576字节的报文。
- MF：更多分段，除最后一个分片外的所有分片都要置MF位。
- 生存期(TTL)：IP报文每经过一个路由器TTL减1，为0则丢弃，并给源主机发送一个告警包。
- 协议：上层为哪种传输协议，TCP、UDP等
- 头校验和：只对IP报头做校验
  - 算法：每16位循环相加(进位加在末尾)，和再求反。



- 可选项：变长，长度为4字节的倍数，不够则填充，最长为40字节。

### 6.7.1.3 IP报文的分段与重组

- MTU：最大信息传送长度，整个IP报文的长度(IP头+IP数据)。
- 按MTU的大小及数据包的实际负载长度计算所需段数并划分，分段应满足两个条件：
  - 各段在不大于MTU的前提下，尽可能地大
  - (数据)段的长度为8字节的整数倍
- 原数据包的包头作为每段的数据包包头，并修改其中的某些字段，指明：
  - 属原来的哪个报文
  - 属原来段中的第几个分段
  - 哪一个段尾
- 通过三个参数实现
  - 标识**：16位
    - 发送方每发送一个报文编号加一
    - 各分段的标识相同
    - 源地址加标识来区分各个属于不同报文的分段
  - 标志**：3位(一位保留)
    - DF = 0：本报文允许分段
    - DF = 1：本报文必须完整到达，途中不允许分段
    - MF = 0：本段为本分组的最后一个段
    - MF = 1：本段后面还有更多的分段
  - 段偏移**：13位
    - 实际偏移量 = 段偏移值 × 8 Byte

0	1	2
保留	DF	MF

- DF = 0：本报文允许分段
- DF = 1：本报文必须完整到达，途中不允许分段
- MF = 0：本段为本分组的最后一个段
- MF = 1：本段后面还有更多的分段
- 段偏移**：13位
  - 实际偏移量 = 段偏移值 × 8 Byte

一个物理网络的MTU为1500B，现要传输一个IP分组(其IP分组头为20B，数据区长度为1400B)到MTU为620B的另一个物理网络，其分段情况为：

原IP报头	600	600	200
分段1报头	600		
分段2报头	600		
分段3报头	200		

每个分段的头部其基本部分即为原IP分组的头部，与分段相关的域则应重新生成

		原头部	分段1头部	分段2头部	分段3头部
ID	标识	30303	30303	30303	30303
M	标志	0	1	1	0
OS	段偏移	0	0	75	150
TL	总长	1420	620	620	220

$$\text{段未结束} \quad 75 \times 8 = 600$$

- 段偏移量：指明报文在分段后，某段在原报文中的相对位置。也就是说，相对于用户数据字段的起点，该段从何处开始。除最后一个段外的所有段的长度必须是8字节(基本段长)的倍数。

### Example

- # 4000 byte datagram
- # MTU = 1500 bytes

1480 bytes in data field

$$\text{offset} = \frac{1480}{8}$$

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

One large datagram becomes several smaller datagrams

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

- 分段的重组：

- 途中的任意一台路由器都无法重组，重组必须在各分段都到达目的地之后才能进行
  - 减少路由器中保存的信息量及路由器的工作量。
- 互联网层尽力而为来传送IP报文，也存在力不从心的时候，此时只能丢弃。
- IP协议采用非透明重组，主机将遵循：要么重组成功，要么全部丢弃的原则。

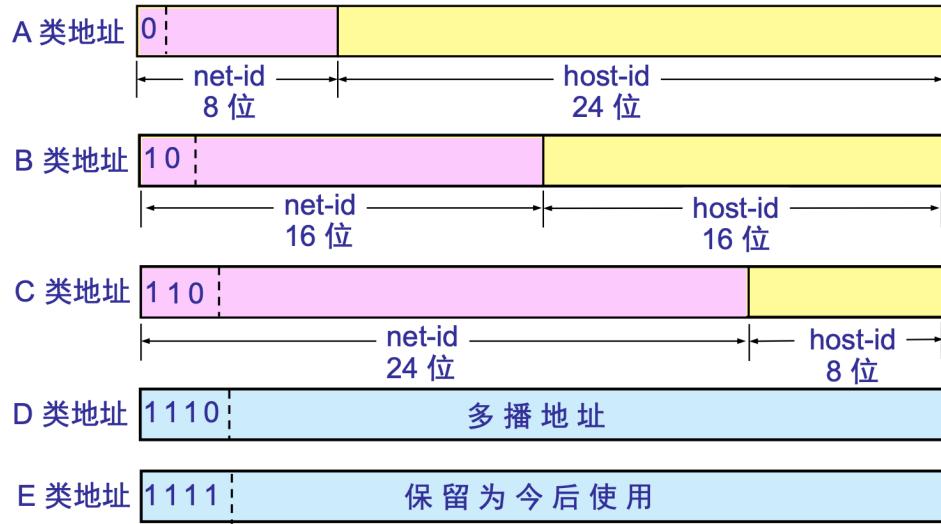
#### 6.7.1.4 IP地址及其表示方法

- IP地址就是给每个连接在因特网上的主机(或路由器)分配一个在全世界范围是唯一的32位的标识符
- IP地址的编址方法：
  - 分类的IP地址：最基本的编址方法。
  - 划分子网：对最基本的编址方法的改进。
  - 无分类编址CIDR：构成超网。

#### 6.7.1.5 分类的IP地址

- 每一类地址都由两个固定长度的字段组成：
  - 网络号net-id：标志主机(或路由器)所连接到的网络
  - 主机号host-id：标志该主机(或路由器)
- 两级的IP地址可以记为：

IP 地址 ::= { <网络号>, <主机号> }



- 常用三种类别IP地址的范围：
  - A类网络去掉全0和127，主机去掉全0、全1
  - B类网络去掉10000000 00000000，主机去掉全0、全1
  - C类网络去掉11000000 00000000 00000000，主机去掉全0、全1

网络类别	最大网络数	第一个可用的网络号	最后一个可用的网络号	每个网络中最大的主机数
A	$126 (2^7 - 2)$	1	126	$2^{24}-2$
B	$16,383(2^{14} - 1)$	128.1	191.255	$2^{16}-2$
C	$2,097,151 (2^{21} - 1)$	192.0.1	223.255.255	$2^8-2$

- 特殊IP地址

前缀	后缀	地址类型	用途
全0	全0	本机	启动时使用
全0	主机ID	主机	标识本网络中的主机
网络ID	全1	直接广播	在指定网上广播
全1	全1	受限广播	在本地网上广播 无盘工作站启动时尚不知道自己所处的网络ID，所以用32为全1地址广播，请求回答
127	任意	回送	测试（127.0.0.1）

- 私有网络的IP地址：保留给私有网络的IP地址段

地址类别	地    址
<b>A类</b>	<b>10.0.0.0 - 10.255.255.255</b>
<b>B类</b>	<b>172.16.0.0 - 172.31.255.255</b>
<b>C类</b>	<b>192.168.0.0 - 192.168.255.255</b>

- 分类IP地址的特点
  - IP地址是一种分等级的地址结构，分两个等级的好处：
    - IP地址管理机构只分配网络号，而主机号则由得到该网络号的单位自行分配，方便管理。
    - 路由器仅根据目的主机所连接的网络号来转发分组，使路由表中的项目数大幅度减少。
  - IP地址是标志一个主机(或路由器)和一条链路的接口
    - 当一个主机同时连接到两个网络上时，该主机就必须同时具有两个相应的IP地址，其网络号net-id必须是不同的，这种主机称为多归属主机。
    - 一个路由器至少应当连接到两个网络，这样它才能将IP数据报从一个网络转发到另一个网络，因此一个路由器至少应当有两个不同的IP地址。
  - 用转发器或网桥连接起来的若干个局域网仍为一个网络，具有同样的网络号net-id。
  - 所有分配到网络号net-id的网络，都是平等的。

#### 6.7.1.6 划分子网

- 两级IP地址的设计不够合理：
  - IP地址空间的利用率比较低，不够灵活
- 三级的IP地址：增加“子网号字段”，使两级的IP地址变成为三级的IP地址。
- 划分子网的基本思路
  - 划分子网属于一个单位内部的事情，单位对外仍然表现为没有划分子网的网络——划分成的所有子网对外仍像一个网络。
  - 从主机号借用若干位作为子网号subnet-id，而主机号host-id也就相应减少了若干位。
  - IP地址 ::= {<网络号>, <子网号>, <主机号>}
  - 只是把IP地址的主机号进行再划分，而不改变IP地址原来的网络号。
  - 子网之间的数据传输，必须通过路由器。
- 划分子网的IP数据报交付
  - 凡是从其他网络发送给本单位某主机的IP数据报，仍然是根据IP数据报的目的网络号net-id，先找到连接在本单位网络上的路由器。
  - 此路由器在收到IP数据报后，再按目的网络号和子网号找到目的子网。
  - 最后将IP数据报直接交付目的主机。
- 子网掩码
  - 从一个IP数据报的首部无法判断源主机或目的主机所连接的网络是否进行了子网划分。
  - 使用子网掩码可以找出IP地址中的子网部分。
  - 子网掩码为32位，由一串1和跟随的一串0组成
    - 一串1对应IP地址中net-id+subnet-id
    - 一串0对应host-id
- IP地址的各字段和子网掩码



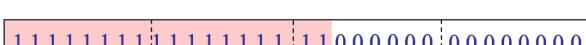
- 默认子网掩码

<b>A类地址</b>	网络地址	net-id	host-id 为全 0
	默认子网掩码 255.0.0.0	11111111 00000000000000000000000000000000	
<b>B类地址</b>	网络地址	net-id	host-id 为全 0
	默认子网掩码 255.255.0.0	1111111111111111 0000000000000000000000000000	
<b>C类地址</b>	网络地址	net-id	host-id 为全 0
	默认子网掩码 255.255.255.0	111111111111111111111111 00000000	

- 子网掩码是一个网络或一个子网的重要属性

- 路由器在和相邻路由器交换路由信息时，必须把自己所在网络(或子网)的子网掩码告知相邻路由器
- 路由器的路由表中的每一个项目，除了要给出目的网络地址外，还必须同时给出该网络的子网掩码
- 若一个路由器连接在两个子网上，就拥有两个网络地址和两个子网掩码

已知IP地址是 141.14.104.24，子网掩码是 255.255.192.0，试求网络地址。

- 点分十进制表示的 IP 地址 
- IP 地址的第 3 字节是二进制 
- 子网掩码是 255.255.192.0 
- IP 地址与子网掩码逐位相与 
- 网络地址（点分十进制表示） 

### 6.7.1.7 无分类编址CIDR

- 无分类域间路由选择CIDR
  - CIDR消除了传统的A类、B类和C类地址以及划分子网的概念，可以更加有效地分配地址
  - 使用各种长度的“**网络前缀**”来代替分类地址中的网络号和子网号，用来指明网络
  - IP地址从三级编址又回到了两级编址

- 无分类的两级编址

- 无分类的两级编址格式：

IP地址 ::= {<网络前缀>, <主机号>}

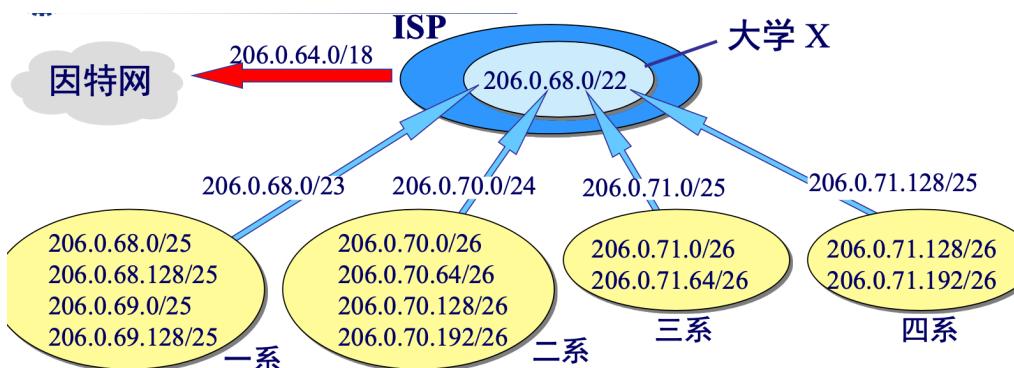
- CIDR使用“**斜线记法**”，在IP地址后面加上一个斜线“/”，写上网络前缀所占的位数
- 网络前缀都相同的连续IP地址组成“**CIDR 地址块**”
- 由主机数目分配网络前缀

- 路由聚合

- 路由聚合：一个CIDR地址块可以表示很多地址，这种地址的聚合称为**路由聚合(构成超网)**，它使得路由表中的一个项目可以表示很多个(例如上千个)原来传统分类地址的路由。
- CIDR仍然使用“掩码”这一名词(不叫子网掩码)。
  - 对于/20 地址块，它的掩码是20个连续的1。

- 构成超网

- 前缀长度不超过23位的CIDR地址块包含了多个C类地址，这些C类地址合起来构成了超网。
- CIDR地址块中的地址数一定是2的整数次幂。
- 网络前缀越短，其地址块所包含的地址数就越多。而在三级结构的IP地址中，划分子网是使网络前缀变长。



单位	地址块	二进制表示	地址数
ISP	206.0.64.0/18	11001110.00000000.01*	16384
大学	206.0.68.0/22	11001110.00000000.010001*	1024
一系	206.0.68.0/23	11001110.00000000.0100010*	512
二系	206.0.70.0/24	11001110.00000000.01000110.*	256
三系	206.0.71.0/25	11001110.00000000.01000111.0*	128
四系	206.0.71.128/25	11001110.00000000.01000111.1*	128

这个ISP共有64个C类网络。如果不采用CIDR技术，则在与该ISP的路由器交换路由信息的每一个路由器的路由表中，需要有64个项目。采用地址聚合后，只需用路由聚合后的1个项目206.0.64.0/18就能找到该ISP。

- 最长前缀匹配

- 使用CIDR时，路由表中的每个项目由“网络前缀”和“下一跳地址”组成。在查找路由表时可能会得到不止一个匹配结果，**网络前缀可能重叠**。
- 应当从匹配结果中选择具有最长网络前缀的路由(**最长前缀匹配**)。
- 网络前缀越长，其地址块就越小，因而路由就越具体。

收到的分组的目的地址  $D = 206.0.71.128$

路由表中的项目：206.0.68.0/22 (ISP)  
206.0.71.128/25 (四系)

查找路由表中的第 1 个项目

第 1 个项目 206.0.68.0/22 的掩码  $M$  有 22 个连续的 1。

$$M = 11111111 11111111 11111100 00000000$$

因此只需把  $D$  的第 3 个字节转换成二进制。

$$\begin{array}{r} M = 11111111 \quad 11111111 \quad 11111100 \quad 00000000 \\ \text{AND } D = \underline{206.} \quad 0. \quad 01000100. \quad 0 \\ \hline 206. \quad 0. \quad 01000100. \quad 0 \end{array}$$

与 206.0.68.0/22 匹配

再查找路由表中的第 2 个项目

第 2 个项目 206.0.71.128/25 的掩码  $M$  有 25 个连续的 1。

$$M = 11111111 11111111 11111111 10000000$$

因此只需把  $D$  的第 4 个字节转换成二进制。

$$\begin{array}{r} M = 11111111 \quad 11111111 \quad 11111111 \quad 10000000 \\ \text{AND } D = \underline{206.} \quad 0. \quad 71. \quad 10000000 \\ \hline 206. \quad 0. \quad 71. \quad 10000000 \end{array}$$

与 206.0.71.128/25 匹配

$$D \text{ AND } (11111111 \quad 11111111 \quad 11111100 \quad 00000000) = 206.0.68.0/22 \text{ 匹配}$$

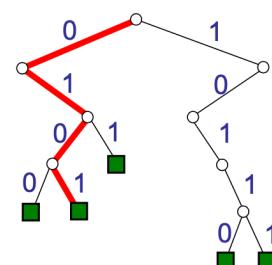
$$D \text{ AND } (11111111 \quad 11111111 \quad 11111111 \quad 10000000) = 206.0.71.128/25 \text{ 匹配}$$

选择两个匹配的地址中更具体的一个，即选择**最长前缀**的地址。

### ● 使用二叉线索查找路由表

- 为了提高路由表查找效率，将无分类编址的路由表存放在一种层次的数据结构中，然后自上而下地按层次进行查找(**二叉线索**)
- IP地址中从左到右的比特值决定了从根结点逐层向下层延伸的路径
- 二叉线索中的各个路径就代表路由表中存放的各个地址

32 位的 IP 地址	唯一前缀
01000110 00000000 00000000 00000000	0100
01010110 00000000 00000000 00000000	0101
01100001 00000000 00000000 00000000	011
10110000 00000010 00000000 00000000	10110
10111011 00001010 00000000 00000000	10111



### ● 使用二叉线索的路由查找

- 依据目的地址(从左到右)，查找二叉线索中是否有对应的叶子节点

- 如没有，则无路由选项，丢弃该分组
- 如有，将目的地址与叶子节点的掩码逐位“AND”运算
  - 结果与叶子节点的网络前缀相匹配，按叶子节点对应的路由项转发
  - 结果与叶子节点的网络前缀不匹配，丢弃该分组

#### 6.7.1.8 IP层转发报文的流程

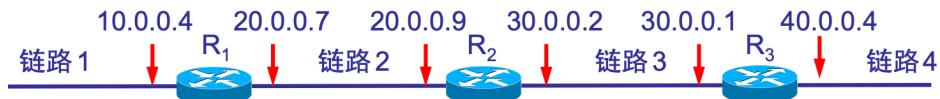
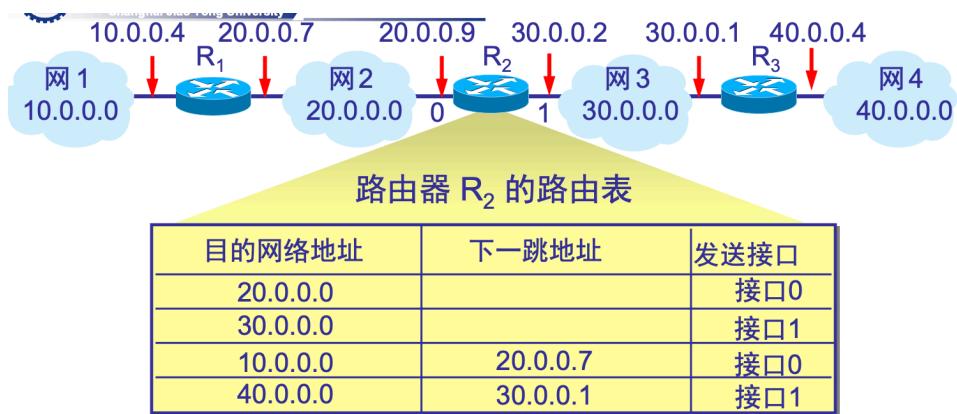
- 路由表的组成
  - 使用分类IP地址网络的路由表

目的网络地址	下一跳地址	接口
20.0.0.0		接口 0
30.0.0.0		接口 1
10.0.0.0	20.0.0.7	接口 0
40.0.0.0	30.0.0.1	接口 0

- 使用子网、CIDR编址网络的路由表

目的网络地址	(子网)掩码	下一跳地址	接口
128.30.33.0	255.255.255.128		接口 0
128.30.33.128	255.255.255.128		接口 1
128.30.36.0	255.255.255.0	128.30.33.129	接口 1

- 默认路由
  - 路由表最后一项是默认路由，当目的地址与路由表中其它行都不匹配时，就按默认路由转发。
  - 默认路由的好处：
    - 减少路由表所占用的空间和搜索路由表所用的时间。
    - 如果一个主机连接在一个小网络上，而这个网络只用一个路由器和因特网连接，使用默认路由则非常合适。
- 分类IP地址的分组转发过程
  1. 从数据报的首部提取目的主机的IP地址D，得出目的网络地址为N
  2. 若网络N与此路由器直接相连，则把数据报直接交付目的主机D；否则是间接交付，执行3
  3. 若路由表中有目的地址为D的特定主机路由，则把数据报传送给路由表中所指明的下一跳路由器；否则，执行4
  4. 若路由表中有到达网络N的路由，则把数据报传送给路由表指明的下一跳路由器；否则，执行5
  5. 若路由表中有一个默认路由，则把数据报传送给路由表中所指明的默认路由器；否则，执行6
  6. 报告转发分组出错



- 使用掩码的分组转发过程

- 从收到分组的首部提取目的IP地址D
- 先用路由器所连各网络的(子网)掩码和D逐位相“与”，看是否与相应的网络地址匹配。若匹配，则将分组直接交付；否则就是间接交付，执行3
- 若路由表中有目的地址为D的特定主机路由，则将分组传送给指明的下一跳路由器；否则执行4
- 对路由表中每一行的子网掩码和D逐位相“与”，若其结果与该行的目的网络地址匹配，则将分组传送给该行指明的下一跳路由器；否则执行5
- 若路由表中有一个默认路由，则将分组传送给路由表中所指明的默认路由器；否则执行6
- 报告转发分组出错

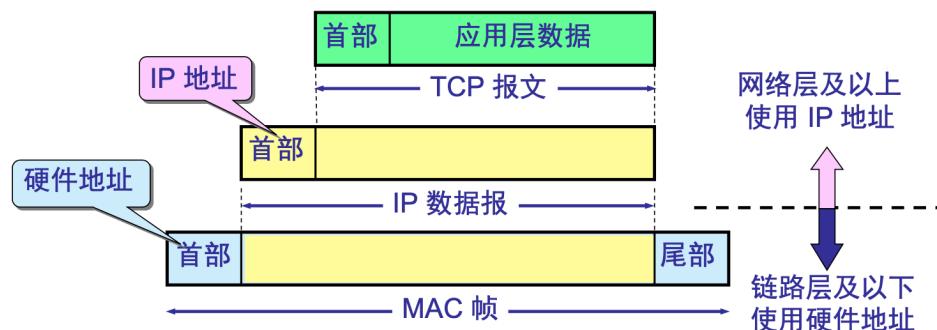
### 6.7.1.9 IPv6协议

- IPv4的不足
  - 地址基本耗尽
  - 路由表越来越大
  - 功能不足，缺少对多媒体信息传输、高速传输、安全、主机漫游的支持
- IPv6的主要改进
  - 更大的地址空间：16字节，128位
  - 首部的简化：只有7个固定域，撤消了有关分段的域和校验和域
  - 更好地支持选项：选项是有次序的，以便路由器可简单地跳过与它无关的选项，加快分组的处理速度
  - 增强了安全性：认证和隐私是关键特征
  - 更加关注服务质量：以支持Internet上日益增长的多媒体应用
- IPv4到IPv6的过渡
  - 双协议栈：过渡时期，站点必须同时支持IPv4和IPv6
  - 隧道技术：IPv6主机之间通信必须使用IPv4的隧道
  - 首部转换：用于发送方使用IPv6，而接收方使用IPv4

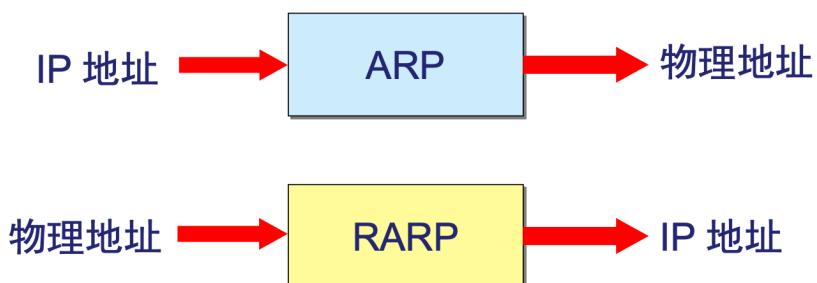
## 6.7.2 ICMP、ARP、RARP协议

### 6.7.2.1 ARP、RARP协议

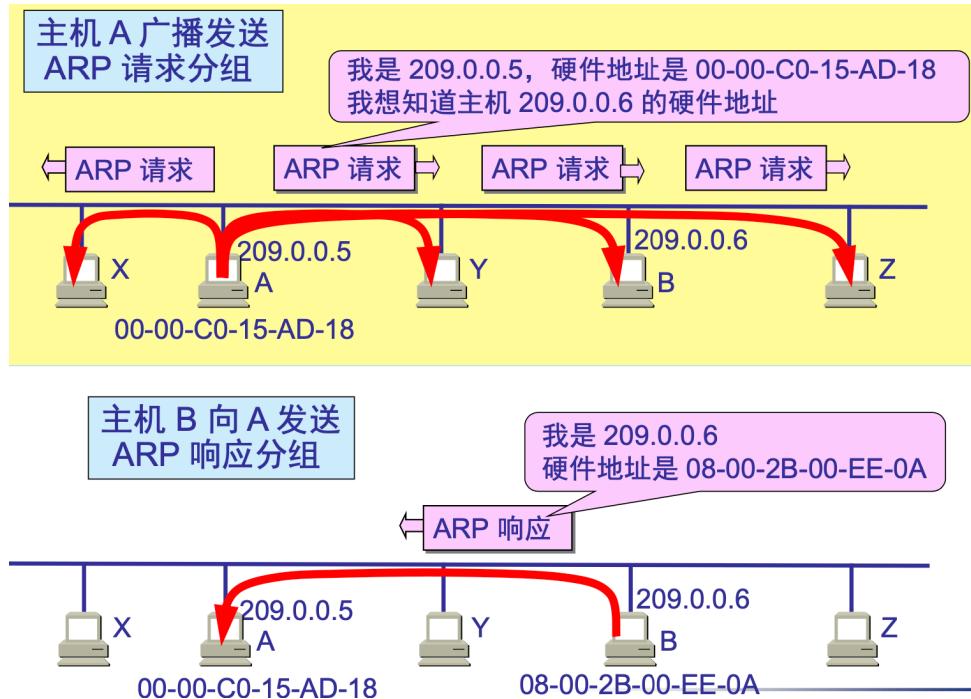
- IP协议不直接使用硬件地址，而是使用IP地址进行通信
  - 全世界存在着各式各样的网络，它们使用不同的硬件地址。要使这些异构网络能够互相通信就必须进行非常复杂的硬件地址转换工作，几乎不可能。
  - 连接到因特网的主机都拥有统一的IP地址，它们之间的通信就像连接在同一个网络上那样简单方便，因为调用ARP来寻找某个路由器或主机的硬件地址都是由计算机软件自动进行的，对用户来说是看不见这种调用过程的。
- IP地址和硬件地址的使用



- 路由器只根据目的站IP地址的网络号进行路由选择。
- 在具体物理网络的链路层，只能看见MAC帧而看不见IP数据报
- IP层抽象的互联网屏蔽了下层很复杂的细节，在抽象的网络层上讨论问题，就能够使用统一抽象的IP地址研究主机和主机或主机和路由器之间的通信。
- IP地址与硬件地址间的转换



- 地址解析协议ARP的工作过程
  - ARP是解决**同一个局域网内**的主机或路由器的IP地址和硬件地址的映射问题。
  - 直接交付时**，ARP解析的是目的主机IP地址与MAC地址的映射——ARP广播帧。
  - 间接交付时**，ARP解析的是下一跳路由器IP地址与MAC地址的映射。
  - 解析是自动进行的，主机的用户不知道解析过程。
  - ARP响应非广播。

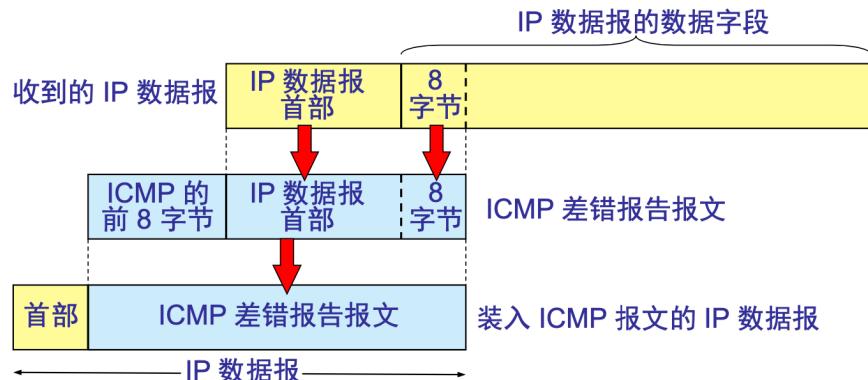


- ARP高速缓存
  - 减少地址解析次数。
  - 高速缓存中每项都设置生存时间，超过生存时间的项目就从高速缓存中删除掉。
  - 高速缓存项目的建立过程：
    - 主机A向B发送其ARP请求分组时，就将自己的IP地址到硬件地址的映射写入ARP请求分组。
    - 当主机B收到A的ARP请求分组时，就将主机A的这一地址映射写入自己的ARP高速缓存中。
    - 当主机A接收到B的ARP响应时，将主机B的地址映射写入自己的ARP高速缓存中。
- RARP协议
  - 逆地址解析协议RARP使只知道自己硬件地址的主机能够知道其IP地址。
  - 获取IP地址的协议
    - RARP协议
    - BOOTP协议
    - DHCP协议（接入网络自动分配IP地址）

### 6.7.2.2 ICMP协议

- ICMP报文功能：
  - 主机或路由器报告差错情况和异常情况。
  - 查询主机是否可达、报文传递时间等。
- ICMP不是高层协议，而是IP层的协议。
- ICMP报文的种类
  - ICMP差错报告报文
    - 终点不可达
    - 源点抑制
    - 时间超过
    - 参数问题
    - 改变路由(重定向)

- ICMP询问报文
  - 回送请求和回答报文：测试目的主机是否可达。
  - 时间戳请求和回答报文：进行时钟同步和测量时间。
- ICMP差错报告报文中数据字段的内容



- ICMP的应用举例
  - PING
    - 用来测试两个主机之间的连通性。
    - 使用了ICMP回送请求与回送回答报文。
    - 应用层直接使用网络层ICMP的例子，没有通过传输层的TCP或UDP。
  - Traceroute
    - Traceroute通过ICMP数据报超时报文来得到一张途经路由器端口列表。
    - 源主机向目的主机发一个IP数据报(端口不可到达的UDP报文)，并置TTL为1，到达第一个路由器时，TTL减1，为0，则该路由器回发一个ICMP数据报超时报文，源主机取出路由器的IP地址即为途经的第一个路由端口地址。
    - 接着源主机再向目的主机发第二个IP报文，并置TTL为2，然后再发第三个、第四个IP数据报，.....直至到达目的主机(ICMP终点不可达报文)。
    - 互联网运行环境状态是动态的，每次路径的选择有可能不一致，只有在相对较稳定的互联网中，Traceroute才有意义。
  - 得到路径中最小的MTU
    - 源主机发送一系列的探测IP数据报，并置DF = 1，即不允许分段，如途经某个网络的MTU较小，则路由器将丢弃该数据报并发回一个ICMP数据报参数错，要求分段，源主机则逐步减小数据报长度，并仍置DF = 1，直至某个探测报文成功到达目的主机，即得到路径中的最小MTU。

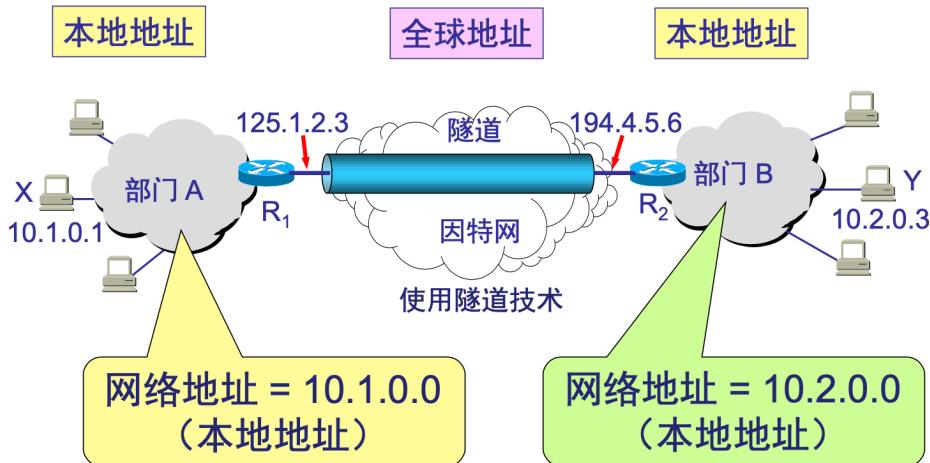
### 6.7.3 VPN和NAT

- **本地地址**：仅在机构内部使用的IP地址，可以由本机构自行分配，也称为**专用地址**
  - 只能用于一个机构的内部通信，而不能用于与因特网上的主机通信。在因特网中的所有路由器对目的地址是专用地址的数据报一律不进行转发。

#### 6.7.3.1 VPN

- 专用互联网
  - 采用专用IP地址的互连网络称为专用互联网，简称为专用网。
  - 位于不同地点的专用网间的通信
    - 租用电信公司的通信线路。

- 利用公用的因特网作为专用网间的通信载体——**虚拟专用网VPN**。
- 用隧道技术实现虚拟专用网（使用私有地址通信）



- VPN类型
  - 内联网 intranet VPN: 一个机构的内部网络所构成的虚拟专用网VPN。
  - 外联网 extranet VPN: 一个机构和某些**外部机构**共同建立的虚拟专用网VPN。
  - 远程接入VPN: 在外地工作的员工拨号接入因特网, 而驻留在员工PC机中的VPN软件可在员工的PC机和公司的主机之间建立VPN隧道, 因而外地员工与公司通信的内容是保密的, 员工们感到好像就是使用公司内部的本地网络。

#### 6.7.3.2 NAT

- NAT的作用
  - 解决公用IP地址紧缺的问题: 一个局域网可只申请一个合法IP地址。
  - 屏蔽内部网络的拓扑结构: 内部网络对外不可见。
- NAT的工作原理
  - 需要在专用网连接到因特网的路由器上安装NAT软件。
  - 装有NAT软件的路由器叫做**NAT路由器**, 它至少有一个有效的外部全球地址IP<sub>G</sub>。
  - 所有使用本地地址的主机在和外界通信时都要在NAT路由器上将其本地地址转换成IP<sub>G</sub>才能和因特网连接。
  - **网络访问请求通常由内部网络主机发起。**
- NAT的工作过程
  - 内部主机X用本地地址IP<sub>X</sub>和因特网上主机Y通信, 所发送的数据报必须经过NAT路由器。
  - 根据**NAT转换表**, NAT路由器将数据报的源地址IP<sub>X</sub>转换成全球地址IP<sub>G</sub>, 但目的地址IP<sub>Y</sub>保持不变, 然后发送到因特网。
  - NAT路由器收到主机Y发回的数据报时, 知道数据报中的源地址是IP<sub>Y</sub>而目的地址是IP<sub>G</sub>。
  - 根据NAT转换表, NAT路由器将目的地址IP<sub>G</sub>转换为IP<sub>X</sub>, 转发给最终的内部主机X。
- NAT的类型
  - 静态NAT
    - 内部网络中的每个主机都被永久映射成某个合法的IP地址。
  - 动态地址NAT
    - 申请了一系列的合法IP地址, 采用动态分配的方法映射到内部网络。
  - 网络地址端口转换NAPT

- 把内部地址映射到一个合法IP地址的**不同端口**上。
- NAT转换表
  - 静态NAT和动态地址NAT的转发表（有几个IP地址即支持几台主机同时访问，很少用）

方向	字段	旧的IP地址	新的IP地址
出	源IP地址	192.168.0.3	172.38.1.5
入	目的IP地址	172.38.1.5	192.168.0.3
出	源IP地址	192.168.0.7	172.38.1.5
入	目的IP地址	172.38.1.5	192.168.0.7

- 网络地址端口转换NAPT的转发表（不同主机映射到不同端口号）

方向	字段	旧的IP地址和端口号	新的IP地址和端口号
出	源IP地址:TCP源端口	192.168.0.3:30000	172.38.1.5:40001
出	源IP地址:TCP源端口	192.168.0.4:30000	172.38.1.5:40002
入	目的IP地址:TCP目的端口	172.38.1.5:40001	192.168.0.3:30000
入	目的IP地址:TCP目的端口	172.38.1.5:40002	192.168.0.4:30000

#### 6.7.4 路由选择协议

- 分层次的路由选择协议
  - 因特网采用分层次的路由选择协议。
  - 因特网的规模非常大，如果让所有的路由器知道所有的网络应怎样到达，则这种路由表将非常大，处理起来也太花时间。而所有这些路由器之间交换路由信息所需的带宽就会使因特网的通信链路饱和。
  - 许多单位不愿意外界了解自己单位网络的布局细节和本部门所采用的路由选择协议，但同时还希望连接到因特网上。
- 自治系统AS
  - 经典定义
    - 由一个组织管理的一整套路由器和网络。
    - 使用一种**AS内部的路由选择协议**和共同的度量以确定分组在该AS内的路由。
    - 使用一种**AS之间的路由选择协议**用以确定分组在AS之间的路由。
  - 目前
    - 尽管一个AS使用了多种内部路由选择协议和度量，但对其他AS表现出的是一个**单一的**和**一致的**路由选择策略。
- 因特网的两大类路由选择协议
  - 内部网关协议IGP：
    - 即在一个AS内部使用的路由选择协议，如RIP和OSPF协议。
    - **域内路由选择**。
  - 外部网关协议EGP：

- 当源主机和目的主机处在不同的AS中，在数据报到达AS的边界时，使用外部网关协议EGP将路由选择信息传递到另一个自治系统中，如BGP-4。
- 域间路由选择。

- 自治系统和内部网关协议、外部网关协议示意图



#### 6.7.4.1 路由信息协议RIP

- 工作原理
  - RIP：一种基于距离向量的路由选择协议。
  - RIP协议要求网络中的每一个路由器都要维护从它自己到自治系统内其他每一个目的网络的距离和下一跳路由器地址。
- “距离”的定义——跳数
  - 从一路由器到直接连接的网络距离定义为1。
  - 从一个路由器到非直接连接网络的距离定义为所经过的路由器数加1。
  - RIP认为一个好的路由就是它通过的路由器的数目少，即“距离短”。
  - RIP允许一条路径最多只能包含15个路由器，“距离”的最大值为16时即相当于不可达。
  - RIP只适用于小型互联网。
  - RIP不能在两个网络之间同时使用多条路由，仅选择一个具有最少路由器的路由(即最短路由)，哪怕还存在另一条高速(低时延)但路由器较多的路由。
- RIP协议的三个要点
  - 仅和相邻路由器交换信息。
  - 交换的信息是当前本路由器所知道的全部信息，即自己的路由表。
  - 按固定的时间间隔交换路由信息。
- 路由表的组成

目的网络	距离	下一跳路由器
Net1	3	R1
Net2	4	R2
Net3	1	直接交付
...	...	...

- 路由表的建立
  - 路由器在刚开始工作时，只知道到直接连接网络的距离(此距离定义为1)。
  - 每一个路由器和相邻路由器交换并更新路由信息——距离向量算法。
  - 经过若干次更新后，所有的路由器最终都会知道到达本自治系统中任何一个网络的最短距离和下一跳路由器的地址。
  - RIP协议的收敛过程较快，即在自治系统中所有的结点都能较快地得到正确的路由选择信息。

举例

(路由器R4、路由器R6相邻)

当前路由器R6的路由表为：

目的网络	距离	下一跳路由器
Net2	3	R4
Net3	4	R5

路由器R4发来的路由更新信息为：

目的网络	距离	下一跳路由器
Net1	3	R1
Net2	4	R2
Net3	1	直接交付

途径路由器R4的候选路由表项：

目的网络	距离	下一跳路由器
Net1	4	R4
Net2	5	R4
Net3	2	R4

更新后的路由器R6路由表：

目的网络	距离	下一跳路由器
Net1	4	R4
Net2	5	R4
Net3	2	R4

- RIP协议的优缺点

- 实现简单，开销较小。
- 限制了网络的规模，它能使用的最大距离为15。
- 路由器之间交换的路由信息是路由器中的完整路由表，随着网络规模的扩大，开销也就增加。
- 当网络出现故障时，要经过比较长的时间才能将此信息传送到所有的路由器。

#### 6.7.4.2 开放最短路径优先OSPF

- 一种链路状态协议

- OSPF协议的三个要点

- 向本自治系统中所有路由器发送信息——洪泛法。
- 发送的信息是与本路由器相邻的所有路由器的链路状态：与哪些路由器相邻，以及该链路的度量，但这只是路由器所知道的部分信息。
- 只有当链路状态发生变化时，路由器才向所有路由器用洪泛法发送此消息。

- 链路状态数据库

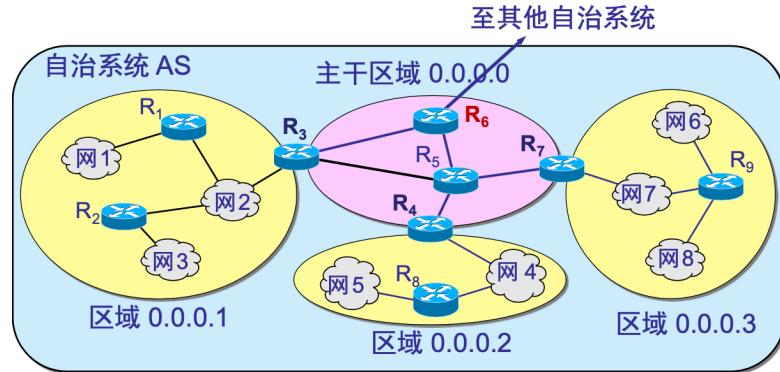
- 由于各路由器之间频繁地交换链路状态信息，因此所有的路由器最终都能建立一个链路状态数据库。
- 实质为全网的拓扑结构图，它在全网范围内一致。
- OSPF的更新过程收敛快是其重要优点。
- 每个路由器使用链路状态数据库中的数据，构造路由表(如：使用Dijkstra最短路径路由算法)。

- OSPF的分层管理方法

- 将一个AS分成若干个区域，每个区域都有一个32位的区域标识符。
- 主干区域(0.0.0.0)：连通其他在下层的区域
  - 主干路由器。
  - 自治系统边界路由器：专门与该自治系统外的其他自治系统交换信息。

- 下层的区域

- 一个区域内的路由器之间交换所有的信息，而对于同一AS内的其他区域则隐藏其详细拓扑结构。
- 通过**区域边界路由器**与主干区域交换信息。



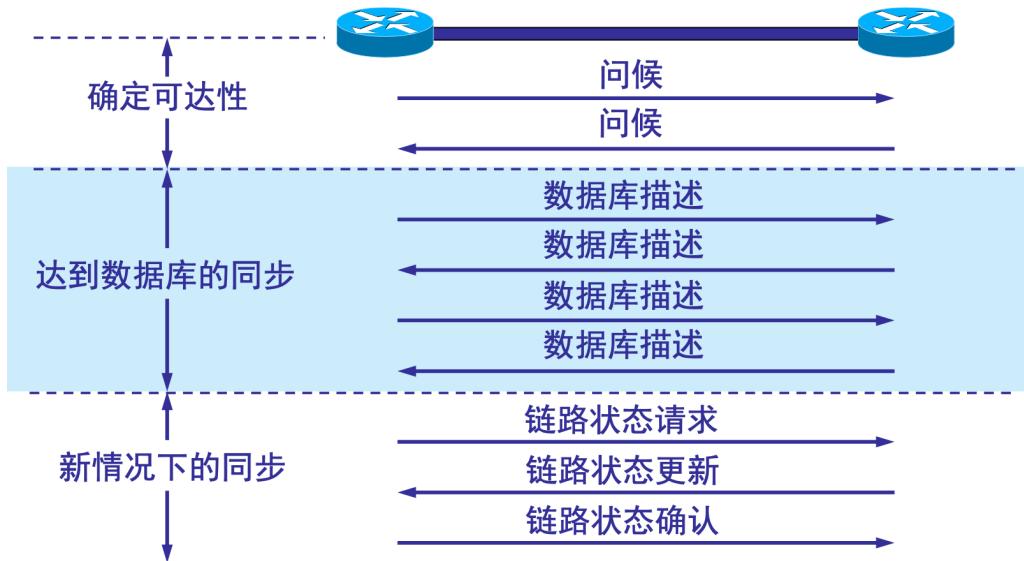
- OSPF协议的报文格式

- OSPF不用UDP而是**直接用IP数据报传送**。
- OSPF构成的数据报很短，可减少路由信息的通信量。
- 数据报很短的另一好处是数据报不必分片传送：分片传送的数据报只要丢失一个，就无法组装成原来的数据报，而整个数据报就必须重传。

- OSPF的五种分组类型

- 类型1，问候(Hello)分组。
- 类型2，数据库描述分组。
- 类型3，链路状态请求分组。
- 类型4，链路状态更新分组，用洪泛法对全网更新链路状态。
- 类型5，链路状态确认分组。

- OSPF的工作过程



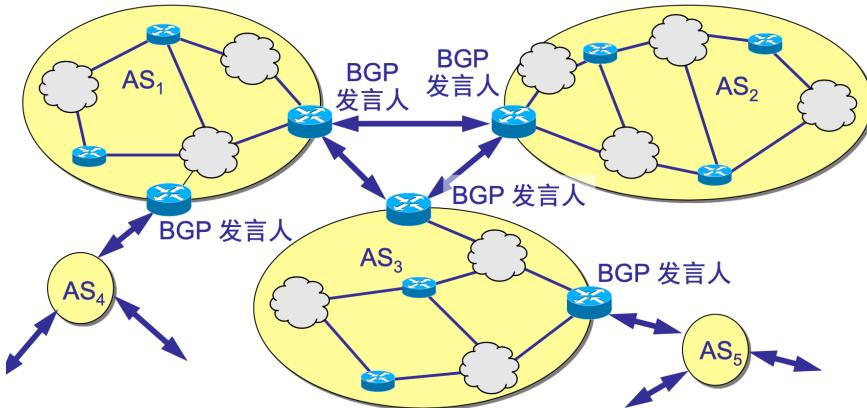
- OSPF协议的特点

- OSPF对不同的链路可根据IP分组的不同服务类型而设置成不同的代价，对于不同类型的业务可计算出**不同的路由**。
- 如果到同一个目的网络有多条相同代价的路径，那么可以将通信量分配给这几条路径——**多路径间的负载平衡**。

- 所有在OSPF路由器之间交换的分组都具有鉴别功能。
- 支持可变长度的子网划分和无分类编址CIDR。
- OSPF没有“坏消息传播得慢”的问题。

#### 6.7.4.3 边界网关协议BGP

- 边界网关协议的概念：不同自治系统的路由器之间交换路由信息的协议。
- BGP协议的设计思路
  - 因特网规模大，自治系统之间路由选择时，要寻找最佳路由不现实，比较合理的方法是在AS之间交换“可达性”信息。
  - 自治系统之间的路由选择必须考虑有关策略。
  - 力求寻找一条能够到达目的网络且比较好的路由(不能兜圈子)，而并非要寻找一条最佳路由。
  - 采用路径向量路由选择协议。
- 路由策略的分类
  - 控制本AS到其他AS的路径：如禁止本AS发出的数据经过某一个中间AS。
  - 控制本AS是否为某一相邻的AS传递过境数据：如出于经济原因。
  - AS内部的协调：决定出境数据的最佳路径。
- BGP发言人
  - 每一个自治系统的管理员要选择至少一个路由器作为该自治系统的“BGP发言人”。
  - 两个BGP发言人一般通过一个共享网络连接在一起，而BGP发言人往往就是BGP边界路由器，但也可能不是BGP边界路由器。
- BGP发言人和自治系统 AS 的关系



- BGP发言人交换路由信息
  - 先建立TCP连接，然后在此连接上交换BGP报文以建立BGP会话，利用BGP会话交换路由信息
    - 使用TCP连接能提供可靠的服务，简化了路由选择协议。
    - 使用TCP连接交换路由信息的两个BGP发言人，彼此成为对方的邻站或对等站。
  - BGP所交换的网络可达性的信息是路径向量，即要到达某个网络所要经过的一系列AS。
  - 当BGP发言人互相交换了网络可达性的信息后，各BGP发言人就根据所采用的策略，从收到的路由信息中找出到达各AS的较好路由。
- BGP协议的特点
  - BGP协议交换路由信息的结点数量级是自治系统数的量级，比这些自治系统中的网络数少很多。
  - 每一个自治系统中BGP发言人的数目很少，自治系统之间的路由选择不致过分复杂。
  - BGP支持CIDR，路由表项包含目的网络前缀、下一跳路由器，以及到达该目的网络所要经过的各个自治系统序列。

- 在BGP刚运行时，BGP的邻站交换整个的BGP路由表，但以后只需要在发生变化时更新有变化的部分。

## Chapter7 传输层

### 7.1 传输层的功能

- 传输层在OSI模型中的位置



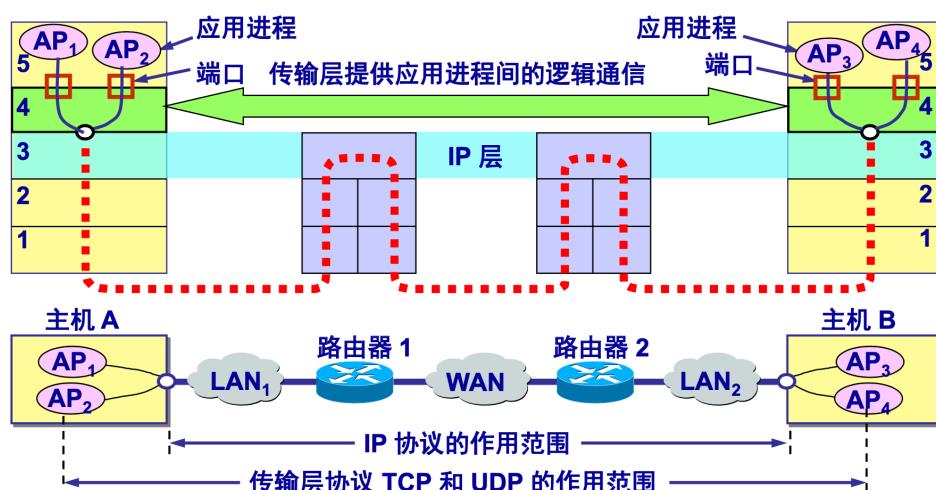
- 传输层的功能

- 为用户提供端到端的通信。
- 介于通信子网和资源子网之间，对高层用户屏蔽了通信的细节。
- 弥补了通信子网所提供的服务的差异和不足，提供端到端之间的无差错保证。
- 传输层工作的繁简取决于通信子网提供服务的类型。

- 端到端的通信

- 两个主机进行通信实际上就是两个主机中的应用进程互相通信。
- IP协议把分组送到目的主机，但是这个分组还停留在主机的网络层而没有交付给主机中的应用进程。
- 由于通信的两个端点是源主机和目的主机中的应用进程，因此应用进程之间的通信又称为端到端的通信。

- 传输层为相互通信的应用进程提供了逻辑通信

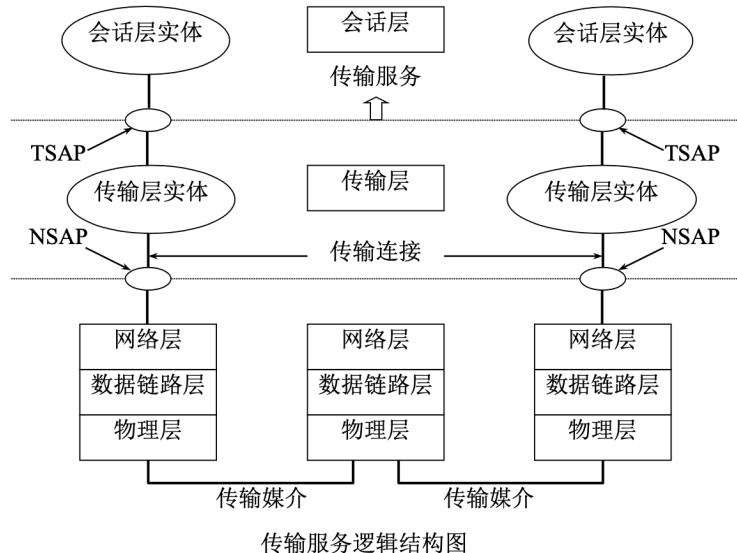


- 传输层与上下层之间的关系

- 传输层向高层用户屏蔽了下面网络核心的细节，使高层用户看见的好像就在两个传输层实体之间有一条端到端的、可靠的、全双工的通信通道。

## 7.2 传输层的服务

- 传输服务提供者也称为传输实体，是提供有效、可靠的传输服务的硬件和软件。
- 在同一系统中，传输实体经过**传输服务访问点(TSAP)**交换**传输服务原语**与上层的**传输服务用户**进行交互，经过**网络服务访问点(NSAP)**交换**网络服务原语**与下层的**网络服务提供者**进行交互。

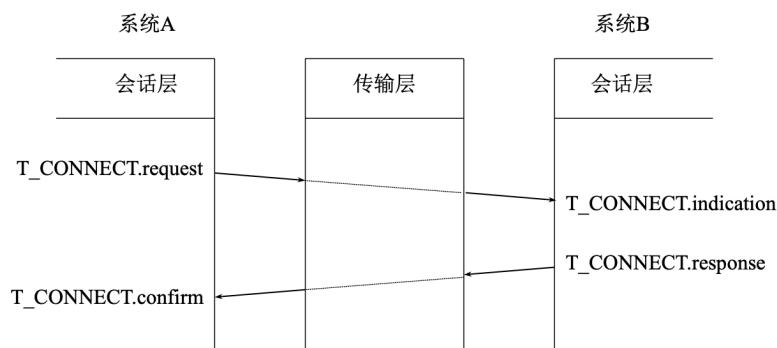


- 传输层依据应用层(会话层)的要求提供服务
  - 会话层实体向传输层实体提出进程间通信所需要的传输条件方面的请求，这种请求用**传输连接**建立**服务原语**中的**服务质量(QoS)**参数来传递。
  - 传输层需要事先知道**网络连接的服务质量**，并根据不同的**网络服务质量**来定义相匹配的**传输服务和协议等级**。
  - 通信子网提供的服务越多，传输协议就可以设计得越简单。
- 网络层提供的服务类型(按服务质量分类)
  - A型网络服务
    - 提供的服务本质上很完善，如局域网的服务。
    - 仅需要尽可能简单的**传输层协议**。
  - B型网络服务
    - 具有可接受的残留差错率和不可接受的被告知的故障率。
  - C型网络服务（大多数）
    - 相当不可靠，具有不可接受的残留差错率和不可接受的被告知的故障率。
  - **传输层依据会话层的QoS要求、网络层提供的网络服务类型，选取合适的协议类型。**
- OSI传输层的协议类型
  - 传输协议类0(TP0)：支持A型网络，实现**分段和重组**功能。
  - 传输协议类1(TP1)：支持B型网络，执行**分段和重组**和**差错恢复**功能。
  - 传输协议类2(TP2)：支持A型网络，实现**分段和重组**，以及**单一虚拟电路上的数据流多路复用技术**和**解除复用技术**等功能。
  - 传输协议类3(TP3)：支持B型网络，提供**差错恢复**、**分段和重组**、以及**单一虚拟电路上的数据流复用技术**和**解除复用技术**等功能，支持**协议数据单元排序操作**。
  - 传输协议类4(TP4)：支持C类网络，提供**差错恢复**功能，实现**分段和重组**处理，并支持**单一虚拟电路上的数据流复用技术**和**解除复用技术**，支持**协议数据单元排序操作**。是OSI传输协议中使用最为普遍的，它类似于TCP/IP协议集中的**传输控制协议(TCP)**。

- OSI传输协议类别与网络服务类型之间的匹配关系
  - 协议复杂性递增。
  - TP0、TP1、TP2、TP3只适用于面向连接通信。
  - TP4既可以用于面向连接通信也可以用于无连接通信。

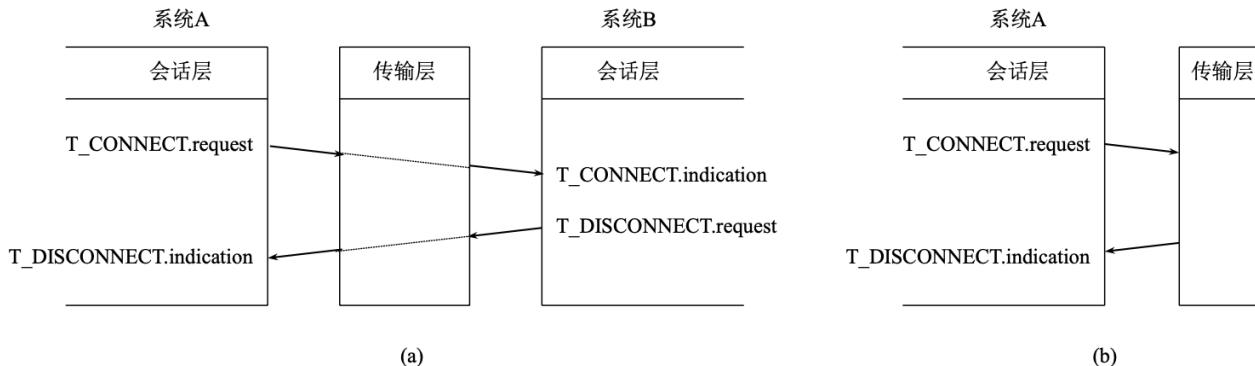
传输协议类别	网络服务类型	基本功能
TP <sub>0</sub>	A	建立连接
TP <sub>1</sub>	B	基本差错修复
TP <sub>2</sub>	A	复用
TP <sub>3</sub>	B	差错修复与复用
TP <sub>4</sub>	C	差错检测与修复，复用

- 传输服务的分类
  - 面向连接的服务：通信可靠，对数据有重发。如应用层协议FTP、Telnet等。
  - 面向非连接的服务：对数据无重发，通信速率高。如应用层协议SNMP、DNS等。
- 传输服务的功能
  - 在传输服务用户之间为传输TPDU(传输层协议封装的分组)而建立一条或多条传输连接。
  - 传输连接建立时，使用QoS参数对服务质量进行协商。
  - 透明地传送TPDU，传输服务提供者对TPDU内容不加限制和改变。
  - 数据传输速率受接收方传输服务用户的控制，即流量控制功能。
  - 经传输服务用户事先一致同意的情况下，可不受流量控制地独立传送加速TPDU。
  - 提供无条件释放传输连接的手段。
- 传输服务质量(QoS)
  - QoS参数是由传输连接用户根据需要提出，在连接建立过程中协商决定的，一般给出希望的值和最小可接受的值，并作为传输连接提供者选择协议的基础。
- 服务原语
  - 服务在形式上是一组原语来描述的。原语被用来通知服务提供者采取某些行动，或报告某同层实体已经采取的行动。
  - 原语可以携带参数。
- 传输层中建立连接的时序图(会话层调用传输层原语)



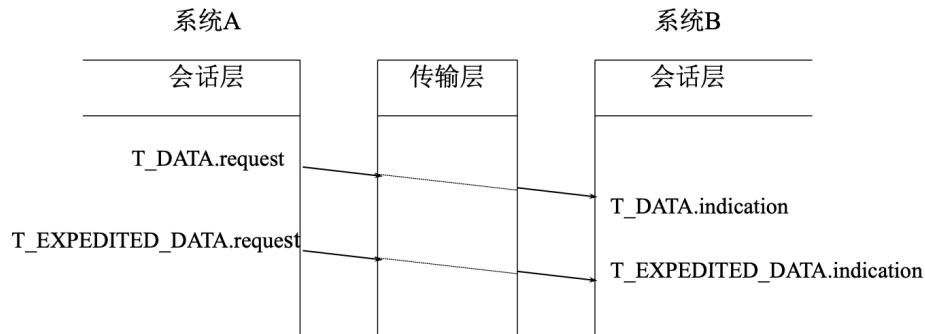
- 传输服务中**连接建立拒绝**的两种情况

1. 是目的主机传输服务用户拒绝接受该连接。
2. 是传输层服务提供者拒绝建立连接。

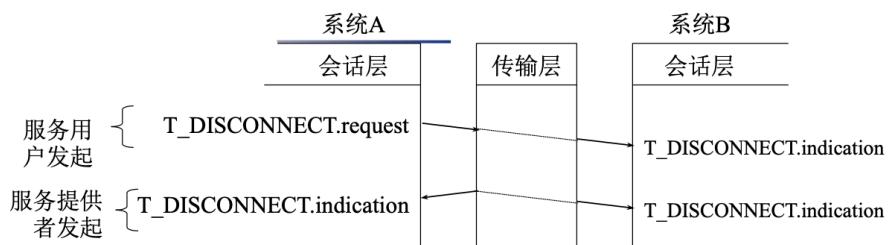


- 数据传送阶段的服务原语

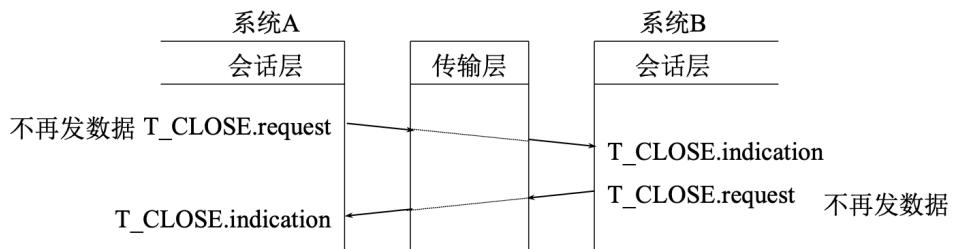
- 数据传送时，不需要从接收数据的传输用户得到响应和证实服务原语，**传输协议负责数据的正确交付**。



- OSI中传输连接释放的两种情况

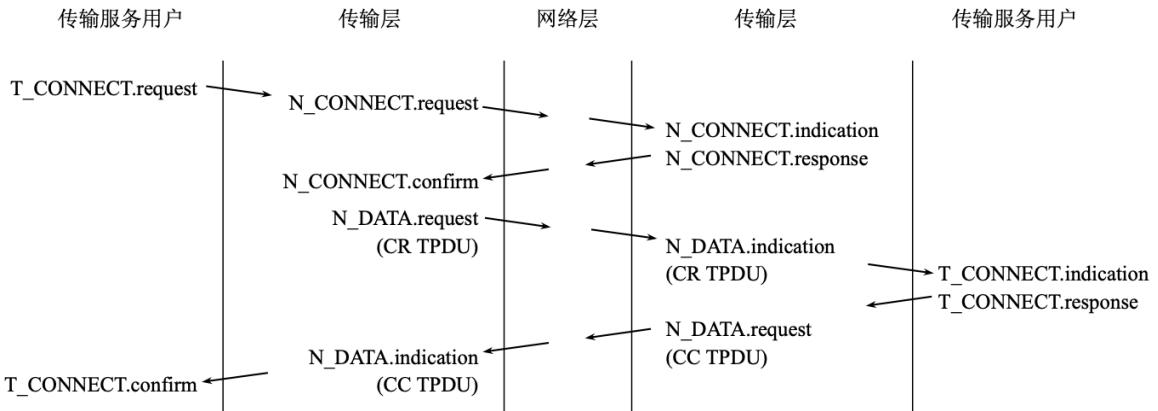


- 传输服务中连接的有序释放

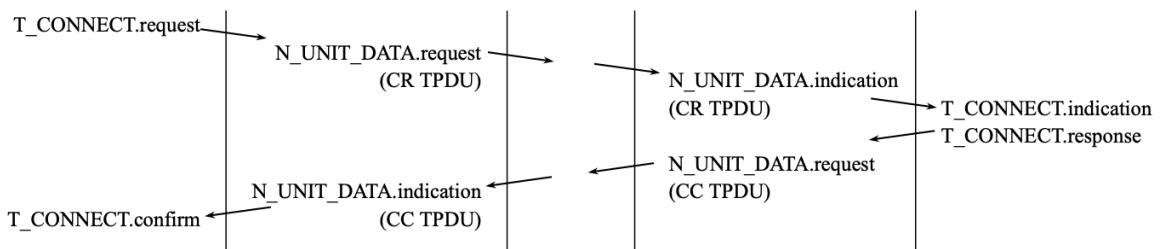


- 使用网络服务建立传输连接

- 面向连接的网络服务



- 无连接的网络服务

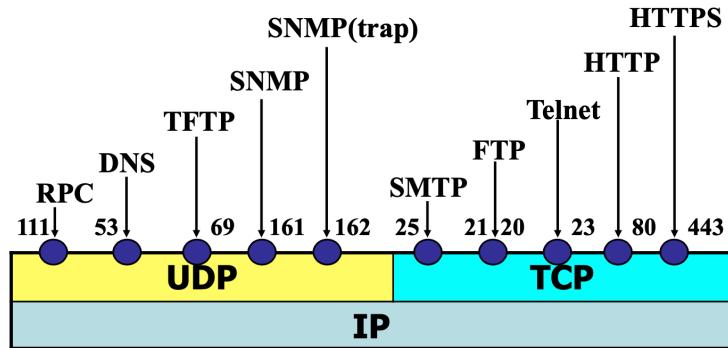


## 7.3 传输层协议的要素

### 7.3.1 寻址

- 传输服务访问点TSAP:
  - 两个程序要建立连接时，必须指明对方是哪一个应用程序，这个标记称为**传输层地址**，也称为**传输服务访问点(TSAP)**。
  - 在TCP协议中传输层地址即TCP的**端口号**。
  - 网络层地址称为**网络服务访问点NSAP**，NSAP在IP协议中即IP地址。
- 通信进程的标识
  - 在单个计算机内，进程用由操作系统分配的“**进程标识符**”来标志。
  - 进行网络通信的进程，用网络上**统一的标识方法**进行标识，即为**协议端口号**。
- 端口的作用：
  - 屏蔽不同操作系统间进程标识符格式的不同。
  - 端口实际上与**应用功能**相对应。
- 端口的表示：
  - 端口用一个**16位端口号**(65536个)进行标志。
  - 端口号只具有**本地意义**，即端口号只是为了标志本计算机应用层中的各进程。在因特网中不同计算机的相同端口号没有联系。
  - 客户端端口号唯一，服务器端端口号唯一和公开(某一类型应用固定端口号)。
  - 两个计算机间的进程要互相通信，需知道对方的IP地址(为了找到对方的计算机)和对方的端口号(为了找到对方计算机中的应用进程)。
- 端口号的分类
  - 服务器端使用的端口号
    - 熟知端口号或系统端口号**，数值为0~1023，由IANA指派给TCP/IP最重要的一些应用程序。
    - 登记端口号**，数值为1024~49151，为没有熟知端口号的应用程序使用。使用这个范围的端口号必须在IANA登记，以防止重复。

- 常用的熟知端口号



- 客户端使用的端口号
  - 称为**客户端口号**或**短暂端口号**, 数值为49152~65535, 留给客户进程选择暂时使用(由操作系统分配)。
  - 当服务器进程收到客户进程的报文时, 就知道了客户进程所使用的端口号。通信结束后, 这个端口号可供其他客户进程以后使用。
- 获取对方TSAP(端口号)的方法
  - well-known TSAP**
    - 每个服务都有自己固定的TSAP, 所有网络用户都知道。
  - 采用**名字服务器或目录服务器**
    - 用户与名字服务器建立连接, 向服务器发送一个报文, 指明服务的名称, 服务器将该服务对应的TSAP返回给用户, 类似于114查号。
  - 服务器方将分配的TSAP通知主机

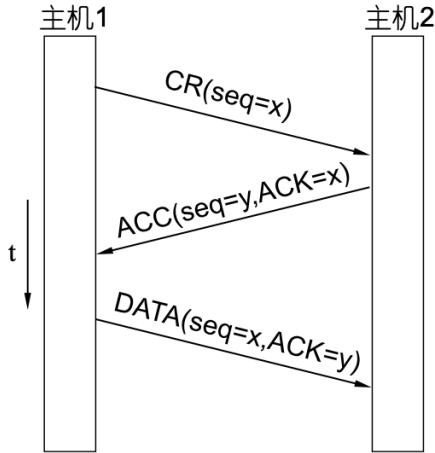
### 7.3.2 连接管理

#### 7.3.2.1 连接的作用

- 通信子网的不可靠性。
- 通信子网中存在着延时和分组的丢失, 以及由此带来的重复分组。
- 由于通信子网的尽力而为的传输原则, 一个早已超时的分组最终还是到达了目的端, 所以有必要将分组的生命周期限制在一个适当的范围内。
- 连接建立时, 如何处理过期分组, 保证连接的唯一性是连接建立过程中首要考虑的问题。
- 常用的方法是: 三次握手法。

#### 7.3.2.2 连接的建立

- 两次握手方式建立连接的问题示例
  - 两次握手: 第一次连接使用阶段中发送的分组由于时延出现在了第二次连接中, 被当作一个新的分组而接收下来, 而真正的新分组却被当作重复分组丢弃掉。
- 正常连接的三次握手过程**
  - 主机1发出的连接请求序号为 $x(\text{seq}=x)$ , 主机2应答接受主机1的连接请求, 并声明自己的序列号为 $y(\text{seq}=y, \text{ACK}=x)$ , 主机1收到确认后发送第一个数据TPDU并确认主机2的序列号( $\text{seq}=x, \text{ACK}=y$ ), 至此, 整个连接建立过程正常结束, 数据传输已正式开始。
  - CR: Connection Request
  - ACC: Connection Accepted

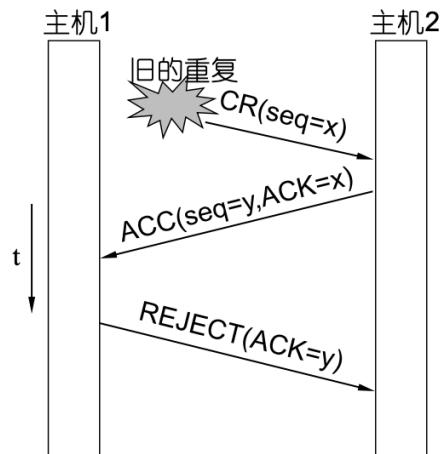


采用三次握手方法建立的正常情况

- 非正常的连接建立过程

- 出现延迟的重复TPDU时三次握手的工作过程

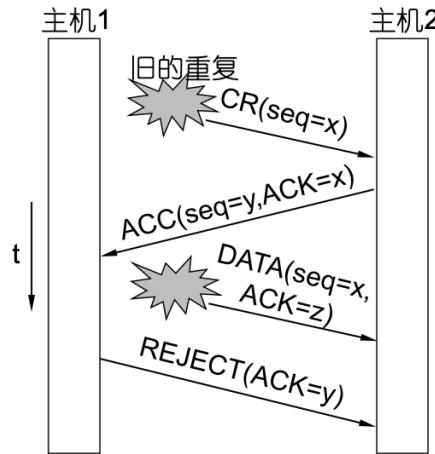
- 来自一个已经释放连接的主机1的延时重复的连接请求，该TPDU在主机1毫不知情的情况下到达主机2，主机2通过向主机1发送一个接受连接请求的TPDU来响应该TPDU，并声明自己的序号为y(seq=y,ACK=x)，主机1收到这个确认后感到莫名其妙并当即拒绝，主机2收到了主机1的拒绝才意识到自己受到了延时的重复TPDU的欺骗并放弃该连接，据此，延时的重复请求将不会产生不良后果。



采用三次握手建立的第二种情况——  
重复的CR突然出现

- 子网中同时有作废的CR和ACC的情况

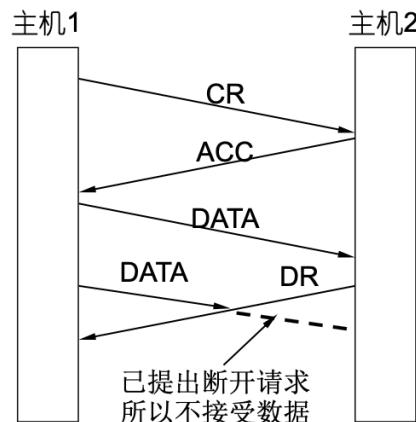
- 与上例一样，主机2收到了一个延时的CR并做了确认应答，在这里，关键是要认识到主机2已经声明使用y作为从主机2到主机1进行数据传输的初始序号，因此主机2十分清楚在正常情况下，主机1的数据传输应捎带对y确认的TPDU，于是，当第二个延时的TPDU到达主机2时，主机2根据它确认的是序号z而不是y知道这也是一个过时的重复TPDU，因此也不会无故建立无人要求的连接。



采用三次握手建立的第三种情况——  
重复的CR和重复的ACK

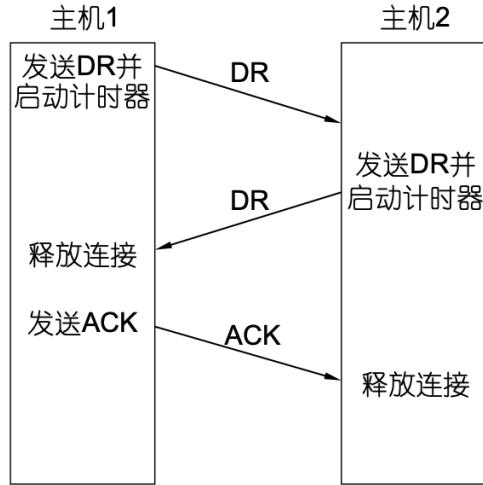
### 7.3.2.3 连接的释放

- 非对称释放——一方中止连接，则连接即告中断
  - 缺陷：可能导致数据丢失。
  - 当连接建立后，主机1发送了一个数据TPDU并正确抵达主机2，接着主机1发送了第二个数据TPDU，然而主机2在收到第二个TPDU之前先突然发出了释放连接请求DISCONNECT，结果是连接立即被释放，数据被丢失。



突然释放连接将造成数据丢失

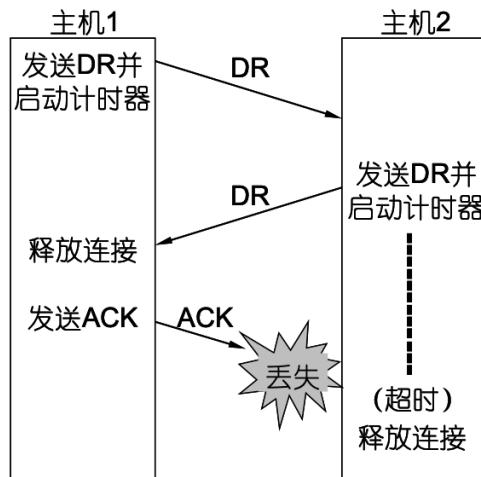
- 对称释放
  - A提出中止请求，B同意即中止
    - 对称释放方式适用于每个用户进程有固定数量的数据需要发送，而且清楚地知道何时发送完毕的情况。
  - 三次握手的正常情况
    - 主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，主机1在计时器没有超时前收到主机2的DR，便正式释放连接并发送ACK，主机2也在计时器没有超时前收到主机1的ACK，于是也释放了连接，至此整个数据传输过程，包括建立连接、传输数据和释放连接的过程正常结束。



三次握手的连接正常释放情况

- 最后的确认TPDU丢失

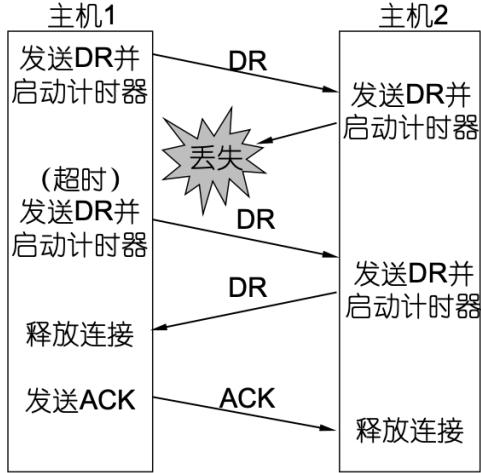
- 主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，主机1在计时器没有超时前收到主机2的DR，便正式释放连接并发送ACK，然而主机2在计时器超时后还未收到主机1的ACK，但是由于已经超时，于是也释放了连接。



最后的确认TPDU丢失的情况

- 应答丢失

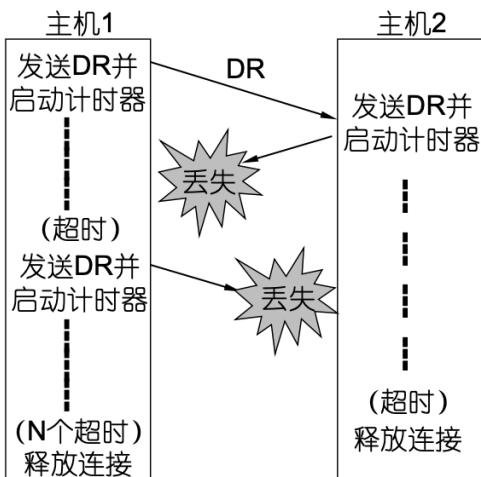
- 主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，然而，主机1在计时器超时后还未收到主机2的DR，于是又重新发送DR并启动计时器，下面便是一个正常的三次握手，并最后正常释放连接，即整个数据传输过程正常结束。



应答丢失的情况

- 应答丢失以及后续的DR丢失

- 主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，然而，紧接着的一段时间内，线路遇到了灾难性的干扰，无论是哪一方的超时重发的TPDU都不能到达对方，最终，接收方计时器的超时而也释放连接，发送方经过n次重发和超时后只能无奈地放弃努力并释放连接。



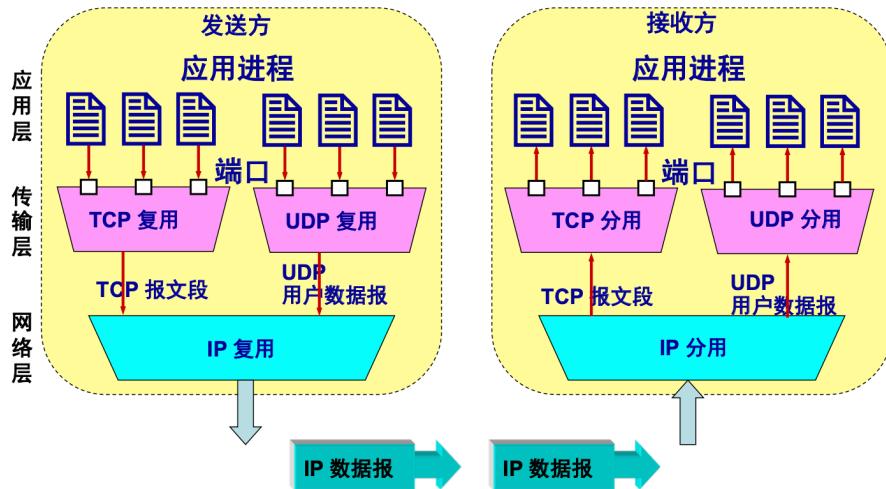
应答丢失以及后续的DR丢失

### 7.3.3 流量控制和缓冲策略

- 发送方和接收方之间的传输速率上的匹配，为使没有得到确认的PDU在超时后的重发，通常必须在缓冲区中暂存。
- 在数据链路层，实现的是**点对点的通信**，双方缓冲区的大小根据滑动窗口协议而定。
- 传输层实现的是**端到端的通信**，某一时刻，一台主机可能同时与多台主机建立了连接，多个连接必须有多组缓冲区，所以缓冲区的动态分配和管理策略要复杂得多。

### 7.3.4 多路复用

- 基于端口的**复用**和**分用**功能



### 7.3.5 崩溃的恢复

- 网络崩溃的恢复
  - 数据报子网：如果传输层对丢失的TPDU留有副本，可以通过重发来解决。
  - 虚电路子网：必须重新建立连接，并询问远端的传输实体哪些TPDU已经收到，没有收到的则必须重发。
- 主机崩溃的恢复
  - 客户端主机崩溃，恢复较为简单。
  - 服务器崩溃，如能及时重新启动，则重新连接后，客户端可能处于两种状态之一：S1(有一个未被确认的TPDU)、S0(没有未被确认的TPDU)
    - 在一般情况下，远端服务器的传输实体在接收到一个客户端的TPDU后，先发送一个确认，再对应用进程执行一个写操作。发送一个确认和执行一个写操作是两个不同的而又不可分的事件，但不能同时进行。
    - 先发送确认，然后再执行写操作，中间发生崩溃
      - 此时客户端将收到这个确认，当崩溃恢复声明到达时它处于状态S0，客户端将因此不再重发，因为它错以为那个TPDU已经到达服务器端，客户端的这种决定会导致丢失一个TPDU。
    - 先执行写操作，然后再发送确认，中间发生崩溃
      - 此时客户端将处于状态S1并因此重传数据，从而会导致在服务器应用进程的输出流上出现一个无法检测的重复的TPDU。
    - 总是存在协议不能正确地从故障中恢复的情况，传输层无法彻底解决该问题，将由高一层协议处理。

## 7.4 因特网的传输层协议

- TCP/IP的传输层有两个主要协议
  - 用户数据报协议UDP
  - 传输控制协议TCP

### 7.4.1 UDP

- UDP的功能：

- UDP只在IP的数据报服务之上增加了很少的功能，即**端口的功能和差错检测**的功能。
- UDP用户数据报只能提供**不可靠的交付**，传输的可靠性要靠应用程序来解决。
- UDP的协议开销小、效率高。

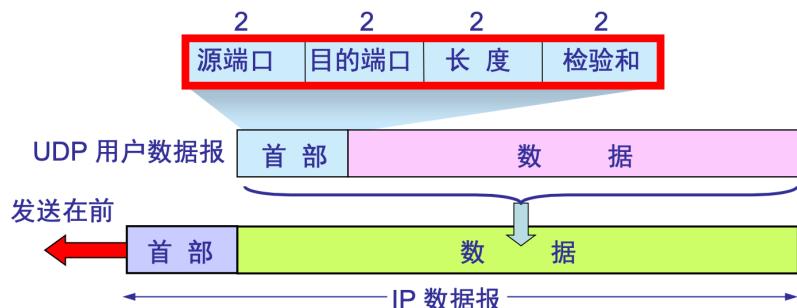
- UDP的主要特点：

- UDP是**无连接**的，即发送数据之前不需要建立连接。
- UDP使用尽最大努力交付，即不保证可靠交付。
- UDP是**面向报文**的。
- UDP没有拥塞控制。
- UDP支持一对一、一对多、多对一、多对多交互通信。
- UDP的首部只有**8个字节**。

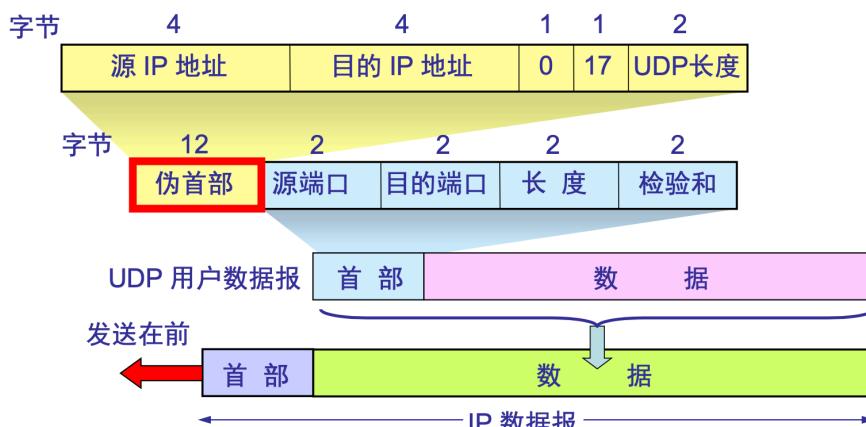
- UDP的面向报文特性

- 发送方UDP对应用程序交下来的报文，在添加首部后就向下交付IP层。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。
- 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。
- 接收方UDP对IP层交上来的UDP用户数据报，在去除首部后就原封不动地交付上层的应用进程，一次交付一个完整的报文。
- 应用程序必须选择合适大小的报文。

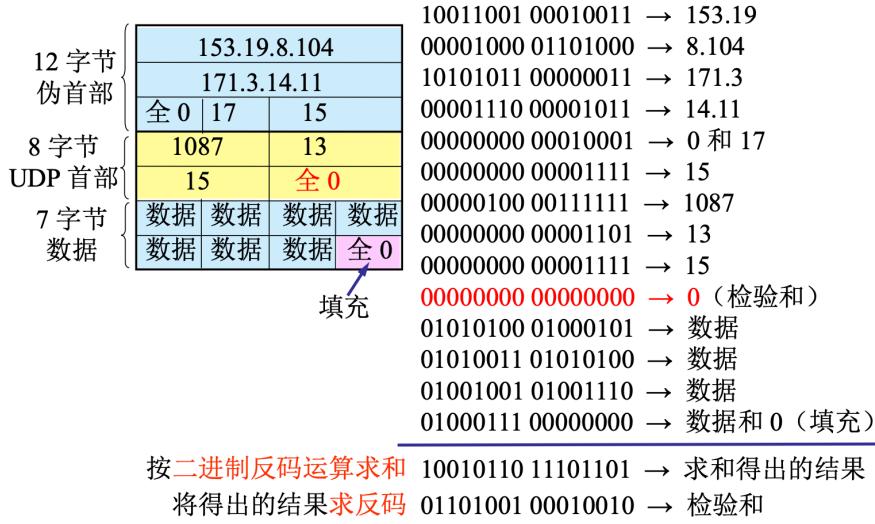
- UDP的首部格式



- 在计算检验和时，临时把“伪首部”和UDP用户数据报连接在一起，“伪首部”仅仅是为了计算检验和。



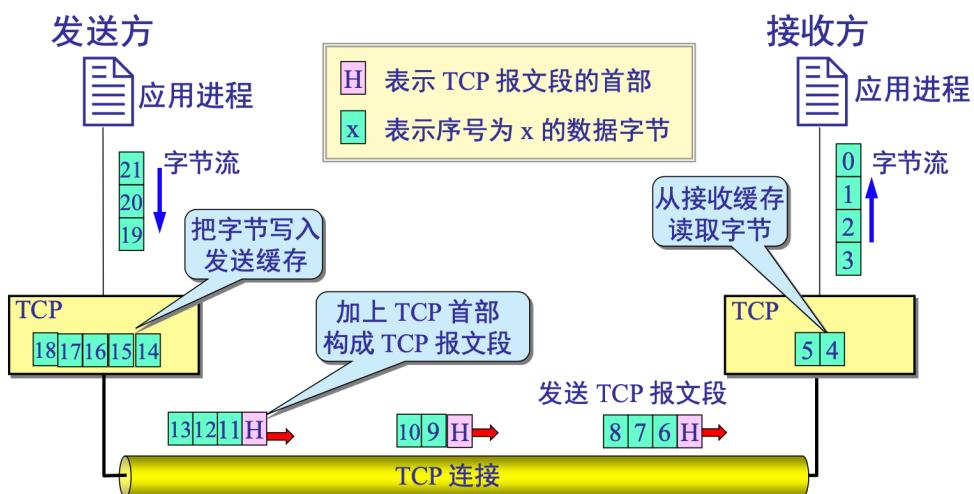
- 计算UDP检验和的例子



## 7.4.2 TCP

### 7.4.2.1 TCP概述

- TCP的功能
  - TCP是面向连接的传输层协议。
  - 每一条TCP连接只能有两个端点，每一条TCP连接只能是点对点的。
  - TCP提供可靠交付的服务。
  - TCP提供全双工通信。
  - 面向字节流。
- TCP面向流的工作过程
  - TCP不关心应用进程一次将多长的报文发送到TCP缓存。
  - TCP对连续的字节流进行分段，形成TCP报文段。



- 注意点
  - TCP连接是一条虚连接而不是一条真正的物理连接。
  - TCP不关心应用进程一次将多长的报文发送到TCP的缓存中。
  - TCP根据对方给出的窗口值和当前网络拥塞的程度，来决定一个报文段应包含多少个字节，而UDP发送的报文长度是应用进程给出的。
  - TCP可将太长的数据块划分为短的数据块再传送，也可等待积累有足够的字节后再构成报文段发送出去。
- TCP连接

- TCP把连接作为**最基本的抽象**。

- 每一条TCP连接有两个端点。

- TCP连接的端点叫做**套接字**

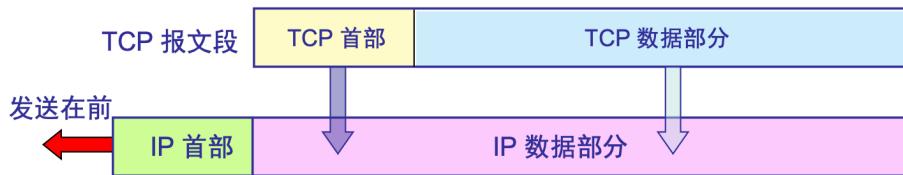
套接字 socket = (IP地址: 端口号)

- 每一条TCP连接唯一地被通信两端的两个端点(即两个套接字)所确定

TCP 连接 ::= {socket1, socket2}  
= {(IP1: port1), (IP2: port2)}

#### 7.4.2.2 TCP报文段的头部格式

- TCP报文段的组成



- TCP报文段的头部格式



- 最短**20字节**, 最长**60字节**。
- 源端口和目的端口字段: 各占2字节。端口是传输层与应用层的服务接口。传输层的复用和分用功能都要通过端口才能实现。
- 序号字段: 占4字节。TCP连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值指的是本报文段所发送数据的第一个字节的序号。
- 确认号字段: 占4字节。是期望收到对方的下一个报文段数据的第一个字节的序号。
- 数据偏移(即首部长度): 占4位。它指出TCP报文段的数据起始处距离TCP报文段的起始处有多远。“数据偏移”的单位是32位字。
- 保留字段: 占6位。保留为今后使用, 目前置为0。
- 紧急URG: 当URG=1时, 它告诉系统此报文段中有紧急数据, 应尽快传送。
- 确认ACK: 只有当ACK=1时确认号字段才有效。
- 推送PSH: 接收TCP收到PSH=1的报文段, 就尽快地交付接收应用进程, 而不再等到整个缓存都填满了后再向上交付。

- 复位RST：当 RST=1时，表明 TCP 连接中出现严重差错，必须释放连接，然后再重新建立运输连接。
- 同步SYN：同步SYN=1表示这是一个连接请求或连接接受报文。
- 终止FIN：FIN=1表明此报文段的发送端的数据已发送完毕，并要求释放传输连接。
- 窗口字段：占2字节，为自己的接收窗口大小，用来让对方设置发送窗口的依据，单位为字节。
- 检验和：占2字节。检验和字段检验的范围包括首部和数据这两部分。在计算检验和时，要在TCP报文段的前面加上12字节的伪首部。
- 紧急指针字段：占16位，指出在本报文段中紧急数据共有多少个字节(紧急数据放在本报文段数据的最前面)。
- 选项字段：长度可变。TCP最初只规定了最大报文段长度MSS。MSS告诉对方TCP：“我的缓存所能接收报文段的数据字段的最大长度是MSS个字节。”数据字段加上TCP首部才等于整个的TCP报文。
  - MSS的选取
    - MSS与接收窗口值没有关系。
    - 若选择较小的MSS长度，网络的利用率就降低。
    - 若选择较大的MSS，那么在IP层传输时就有可能要分解成多个短数据报片。
    - MSS应尽可能大些，只要在IP层传输时不需要再分片就行。
  - 其它选项：窗口扩大选项、时间戳选项、选择确认选项。
- 填充字段：这是为了使整个首部长度是4字节的整数倍。

#### 7.4.2.3 TCP的可靠传输-窗口机制

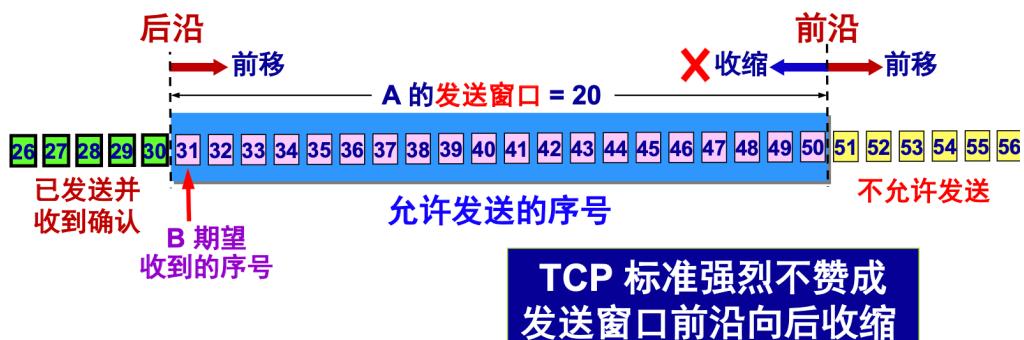
- 窗口机制
  - TCP连接的每一端都必须设有两个窗口：发送窗口、接收窗口。
  - TCP两端的四个窗口经常处于动态变化之中。
  - TCP的可靠传输机制利用字节的序号进行控制，TCP所有的确认都是基于序号而不是基于报文段。

根据B给出的窗口值，A构造出自己的发送窗口。

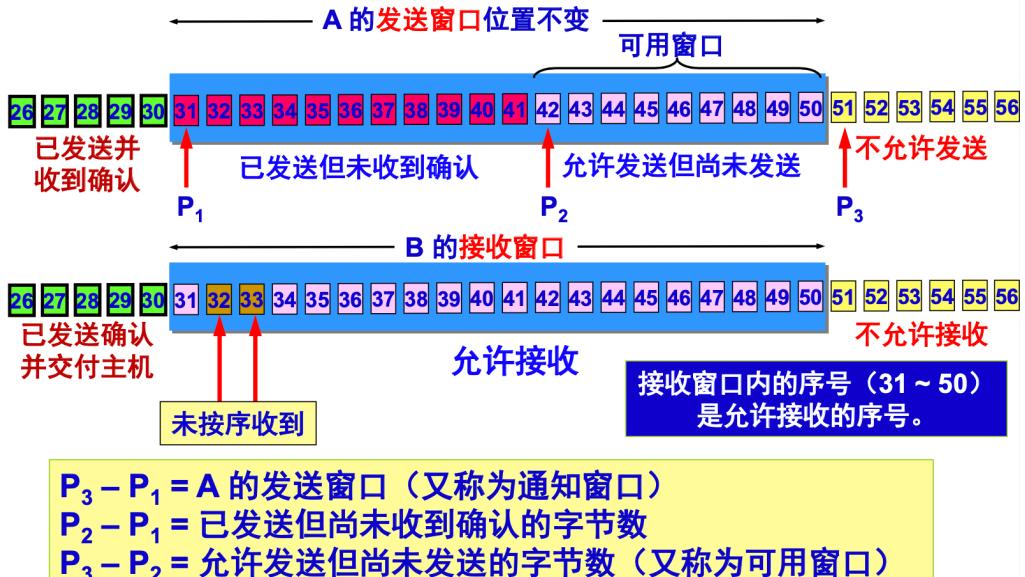
发送窗口表示：在没有收到B的确认的情况下，A可以连续把窗口内的数据都发送出去。

发送窗口里面的序号表示允许发送的序号。

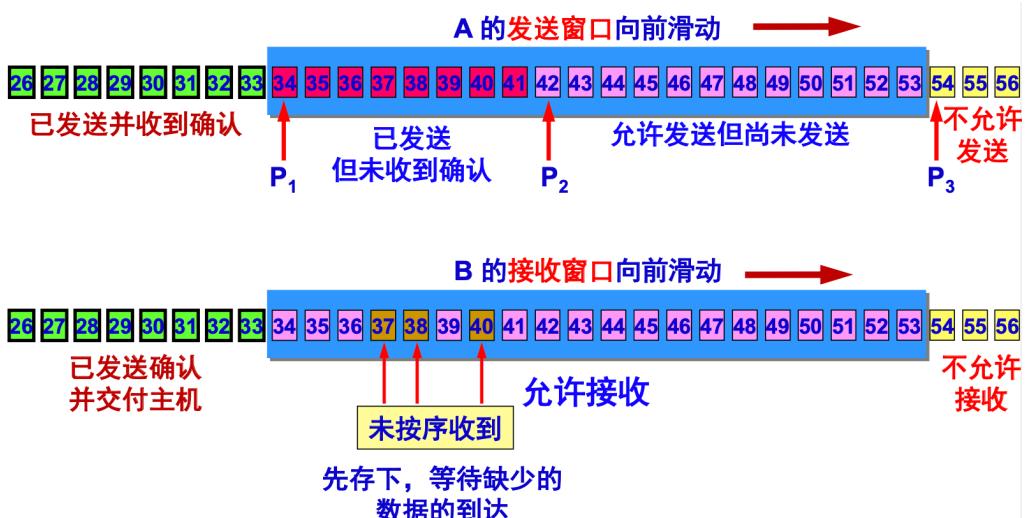
窗口越大，发送方就可以在收到对方确认之前连续发送更多数据，因而获得更高的传输效率。



A发送了11个字节的数据



A收到新的确认号，发送窗口向前滑动

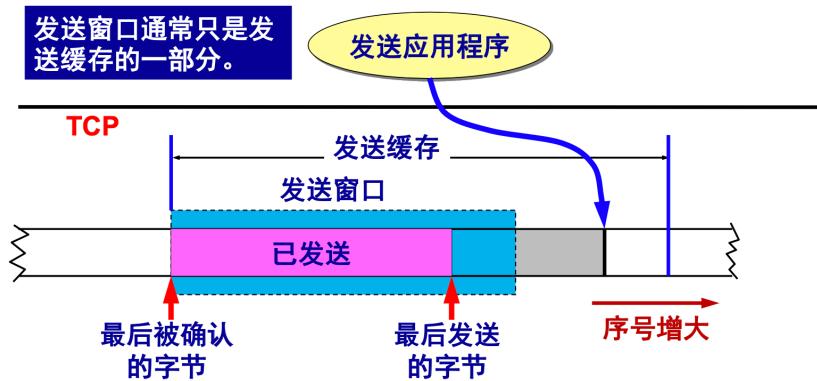


A的发送窗口内的序号都已用完，但还没有再收到确认，必须停止发送



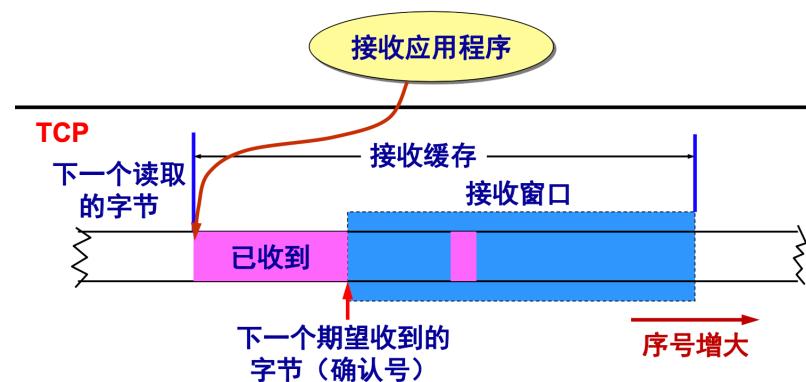
发送缓存

### 发送方的应用进程把字节流写入 TCP 的发送缓存。



接收缓存

### 接收方的应用进程从 TCP 的接收缓存中读取字节流。



- 注意点
  - A的发送窗口并不总是和B的接收窗口一样大(有一定的时间滞后)。
  - TCP标准没有规定对不按序到达的数据应如何处理，通常是先临时存放在接收窗口中，等到字节流中所缺少的数据收到后，再按序交付上层的应用进程。
  - TCP要求接收方必须有累积确认的功能，这样可以减小传输开销。
- TCP使用与数据链路层不同的窗口协议来实现流量控制
  - 数据链路层：在数据链路层的滑动窗口协议中，发送窗口的滑动依赖于数据的发送(后沿滑动)和确认的到达(前沿滑动)。
  - TCP层：TCP的发送窗口大小由接收方通告的接收窗口大小决定，接收到接收方的确认报文后，窗口后沿向前滑动，并滑动窗口前沿，使得：**前沿-后沿=调整后的发送窗口大小**。

#### 7.4.2.4 TCP的可靠传输-超时重传机制

- 超时重传机制
  - TCP每发送一个报文段，就对这个报文段设置一次**超时计时器**。只要超时计时器设置的重传时间到了，但还没有收到确认，就要重传这一报文段。
  - 超时计时器时间的设置应与报文段的**往返时间RTT**相关。
- 加权平均往返时间RTT<sub>S</sub>
  - 第一次测量到RTT样本时，RTT<sub>S</sub>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RTT<sub>S</sub>：

$$\text{新}RTT_s = (1 - \alpha) \times (\text{旧}RTT_s) + \alpha \times (\text{新}RTT\text{样本})$$

- 超时重传时间RTO

$$RTO = RTT_s + 4 \times RTT_D$$

- $RTT_D$ 是RTT偏差的加权平均值

- 第一次测量时， $RTT_D$ 值取为测量到的RTT样本值的一半。
- 在以后的测量中，则使用下式计算加权平均的  $RTT_D$ ：

$$\text{新}RTT_D = (1 - \beta) \times (\text{旧}RTT_D) + \beta \times |RTT_s - \text{新}RTT\text{样本}|$$

- 报文段重传对RTO计算的影响

- Karn算法

- 在计算平均往返时间RTT时，只要报文段重传了，就不采用其往返时间样本。
- 这样得出的加权平均平均往返时间 $RTT_S$ 和超时重传时间RTO就较准确。

- 修正的Karn算法

- 报文段每重传一次，就把RTO增大一些：

$$\text{新}RTO = \lambda \times (\text{旧}RTO)$$

- 当不再发生报文段的重传时，才根据报文段的往返时间，更新平均往返时间 $RTT_S$ 和超时重传时间RTO的数值。

#### 7.4.2.5 TCP的可靠传输-选择确认SACK

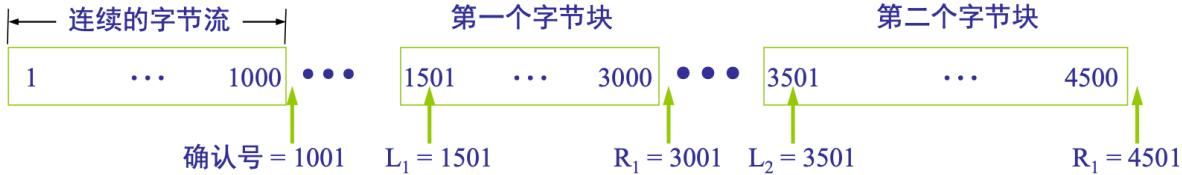
- 工作原理

- 接收方收到了和前面的字节流不连续的字节块
- 如果这些字节的序号都在接收窗口之内，那么接收方就先收下这些数据，但要把这些信息准确地告诉发送方，使发送方不要再重复发送这些已收到的数据。

- 接收到的两个不连续的字节块

- 和前后字节不连续的每一个字节块都有两个边界：左边界和右边界
- 左边界指出字节块的第一个字节的序号，但右边界减1才是字节块中的最后一个序号
- 图中用四个指针标记这些边界：

- 第一个字节块的左边界 $L_1=1501$ ，右边界 $R_1=3001$
- 第二个字节块的左边界 $L_2=3501$ ，右边界 $R_2=4501$



- 选择确认SACK的实现

- 如果要使用选择确认，那么在建立TCP连接时，就要在TCP首部的选项中加上“允许 SACK”的选项，而双方必须都事先商定好。
- 如果使用选择确认，那么原来首部中的“确认号字段”的用法仍然不变，只是以后在TCP报文段的首部中都增加了SACK选项，以便报告收到的不连续的字节块的边界。
- 由于首部选项的长度最多只有40字节，而指明一个边界就要用掉4字节，因此在选项中最多只能指明

4个字节块的边界信息。

#### 7.4.2.6 TCP的流量控制

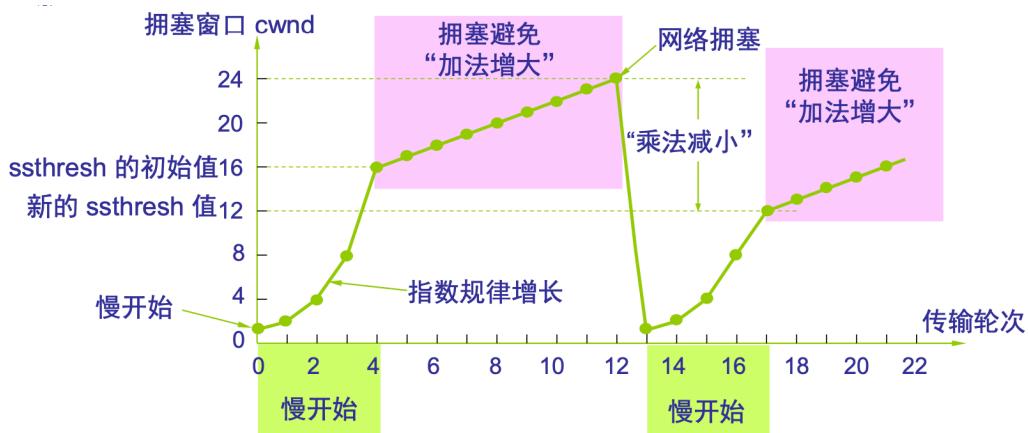
- 流量控制：让发送方的发送速率不要太快，既要让接收方来得及接收，也不要使网络发生拥塞。
  - 利用滑动窗口机制可以方便地在TCP连接上实现流量控制
    - 发送方依据发送窗口来发送报文段
      - 未收到新的确认报文，发送窗口后沿、前沿都不移动
      - 收到新的确认报文段，发送窗口后沿前移，前沿依据确认报文段中的接收窗口调整
    - 接收方依据接收缓存的使用情况，动态调整接收窗口的大小，通过确认报文段告知发送方，从而使发送方的发送窗口进行调整——发送窗口受接收窗口控制。
  - 预防死锁僵局
    - TCP为每一个连接设有一个持续计时器，只要TCP连接的一方收到对方的零窗口通知，就启动持续计时器。
    - 若持续计时器设置的时间到期，就发送一个零窗口探测报文段，而对方就在确认这个探测报文段时给出现在的窗口值。
      - 若窗口仍然是零，则收到这个报文段的一方就重新设置持续计时器。
      - 若窗口不是零，则死锁的僵局打破。
  - 提高TCP的传输效率
    - TCP报文段发送时机的控制机制
      - 第一种机制：TCP维持一个变量，它等于最大报文段长度MSS，只要缓存中存放的数据达到MSS字节时，就组装成一个TCP报文段发送出去。
      - 第二种机制：由发送方的应用进程指明要求发送报文段，即TCP支持的推送操作。
      - 第三种机制：发送方的一个计时器期限到了，这时就把当前已有的缓存数据装入报文段(长度不能超过MSS)发送出去。
    - Nagle算法
      - 将从应用程序收到的第一块数据，立即发送出去。
      - 在发送第一个报文段后，在发送缓存中积累并等待，若数据已积累到可以装成一个最大的报文段或积累数据达到发送窗口的一半时，发送下一个报文段，或者接收到确认报文段，将发送缓存中的数据组装成下一个报文段，发送出去。
    - 解决糊涂窗口综合症
      - Clark算法
        - 只要有数据到达就发送确认，但告知的窗口为0，直至接收缓存空间已能放入具有最大长度的报文段或者接收缓存空间的一半已经空闲。
        - 延迟确认：当一个报文段达到时并不立即发送确认，一直等待直至接收缓存有足够的空间。

#### 7.4.2.7 TCP的拥塞控制

- 拥塞控制策略：
  - 策略一：开环控制
    - 重在预防，希望通过完美的设计来避免拥塞的发生。
  - 策略二：闭环控制
    - 重在解决，在拥塞发生后设法控制和缓解拥塞。

- 属于闭环控制的有以下几种措施：
  - 监测网络系统，以便检测到拥塞在何时、何处发生。
  - 将拥塞发生的信息传送到可采取行动的地方。
  - 调整网络系统的运行以解决出现的问题。
- 拥塞控制算法的分类
  - 在端系统上使用的源算法——慢开始、拥塞避免、乘法减小、快重传和快恢复。
  - 在网络设备上使用的链路算法——RED(随机早期检测)。
  - TCP拥塞控制采用的是基于窗口的端到端闭环控制方式。
- 发送窗口的上限值
  - 发送方维持拥塞窗口 $cwnd$ 的状态变量，拥塞窗口的大小取决于网络的拥塞程度，并且动态地在变化。
  - 发送方的发送窗口的上限值应当取为接收方窗口 $rwnd$ 和拥塞窗口 $cwnd$ 这两个变量中较小的一个，即发送窗口的上限值 =  $\min[rwnd, cwnd]$
- 发送方控制拥塞窗口的原则：
  - 只要网络没有出现拥塞，拥塞窗口就再增大一些，以便把更多的分组发送出去
  - 只要网络出现拥塞，拥塞窗口就减小一些，以减少注入到网络中的分组数。
- 慢开始算法
  - 慢开始算法的原理：
    - 在主机刚刚开始发送报文段时可先设置拥塞窗口 $cwnd = 1$ ，即设置为一个最大报文段MSS的数值。
    - 在每收到一个对新的报文段的确认(重传的不算在内)后，将拥塞窗口加1。
    - 用这样的方法逐步增大发送端的拥塞窗口 $cwnd$ ，可以使分组注入到网络的速率更加合理。
  - 传输轮次
    - 把拥塞窗口 $cwnd$ 所允许发送的报文段都连续发送出去，并收到了对已发送的最后一个字节的确认。
    - 使用慢开始算法后，每经过一个传输轮次，拥塞窗口 $cwnd$ 就加倍。
    - 一个传输轮次所经历的时间  $\approx$  往返时间RTT
      - 例如，拥塞窗口 $cwnd = 4$ ，这时的往返时间RTT就是发送方连续发送4个报文段，并收到这4个报文段的确认，总共经历的时间。
  - 慢开始门限状态变量 $ssthresh$ 
    - 慢开始门限 $ssthresh$ 的用法如下：
      - 当  $cwnd < ssthresh$  时，使用慢开始算法。
      - 当  $cwnd > ssthresh$  时，停止使用慢开始算法而改用拥塞避免算法。
      - 当  $cwnd = ssthresh$  时，既可使用慢开始算法，也可使用拥塞避免算法。
- 拥塞避免算法(加法增大算法)
  - 让拥塞窗口 $cwnd$ 缓慢地增大，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，使拥塞窗口 $cwnd$ 按线性规律缓慢增长。
- 乘法减小算法
  - 网络出现拥塞时的措施
    - 无论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(其根据是没有按时收到确认)
      - 将慢开始门限 $ssthresh$ 设置为出现拥塞时的拥塞窗口值的一半(但不能小于2)

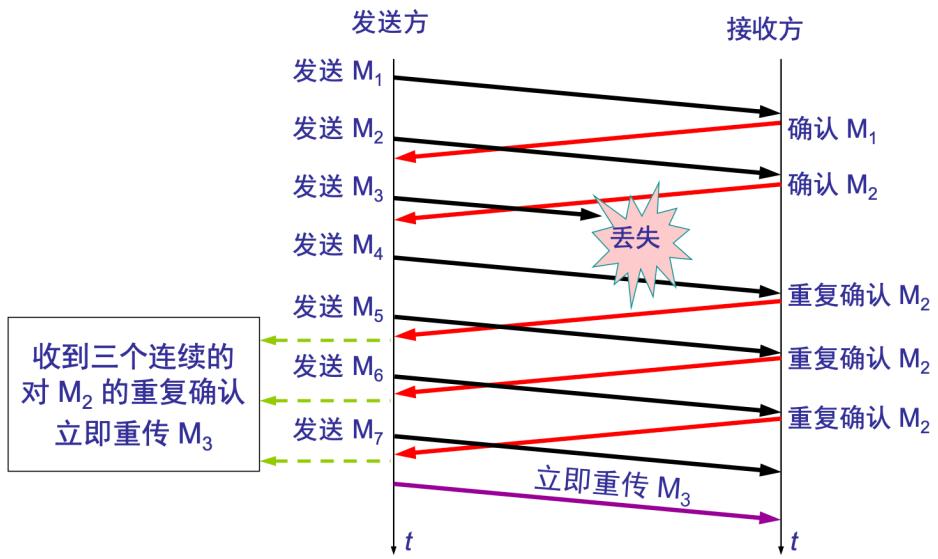
- 将拥塞窗口 $cwnd$ 重新设置为1，执行慢开始算法。
- 目的：迅速减少主机发送到网络中的分组数，使得发生拥塞的路由器有足够时间把队列中积压的分组处理完毕。



1. 当TCP连接进行初始化时，将拥塞窗口置为1。图中的窗口单位不使用字节而使用报文段。慢开始门限的初始值设置为16个报文段，即 $ssthresh = 16$ 。
2. 发送端的发送窗口不能超过拥塞窗口 $cwnd$ 和接收端窗口 $rwnd$ 中的最小值，假定接收端窗口足够大，因此现在发送窗口的数值等于拥塞窗口的数值。
3. 在执行慢开始算法时，拥塞窗口 $cwnd$ 的初始值为1，发送第一个报文段 $M_0$ 。
4. 发送端每收到一个确认，就把 $cwnd$ 加1。于是发送端可以接着发送 $M_1$ 和 $M_2$ 两个报文段。
5. 接收端共发回两个确认。现在 $cwnd$ 从2增大到4，并可接着发送后面的4个报文段。
6. 拥塞窗口 $cwnd$ 随着传输轮次按指数规律增长。
7. 当拥塞窗口 $cwnd$ 增长到慢开始门限值 $ssthresh$ 时，就改为执行拥塞避免算法，拥塞窗口按线性规律增长。
8. 假定拥塞窗口的数值增长到24时，网络出现超时，表明出现网络拥塞。
9. 更新后的 $ssthresh$ 值变为12(即拥塞窗口数值24的一半)，拥塞窗口再重新设置为1，并执行慢开始算法。
10. 当 $cwnd = 12$ 时改为执行拥塞避免算法，拥塞窗口按线性规律增长，每经过一个往返时延就增加一个MSS的大小。

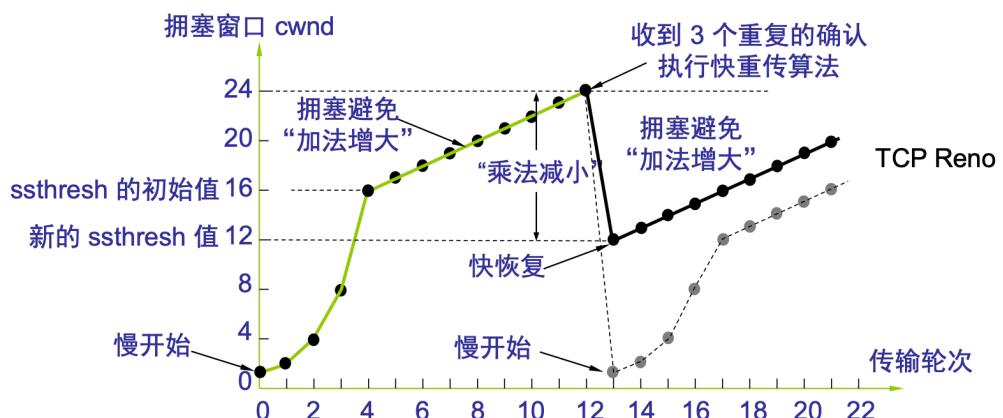
### ● 快重传算法

- 要求接收方每收到一个失序的报文段后就立即发出重复确认，以让发送方尽早知道有报文段没有到达接收方。
- 发送方只要一连收到三个重复确认就应当立即重传对方尚未收到的报文段。
- 快重传并非取消重传计时器，而是在某些情况下可更早地重传丢失的报文段。



- 快恢复算法

- 当发送端收到连续三个重复的确认时，就将慢开始门限ssthresh设置为当前拥塞窗口的一半。
- 由于发送方现在认为网络很可能没有发生拥塞，因此现在不执行慢开始算法，即拥塞窗口cwnd现在不设置为1，而是设置为减半后的慢开始门限ssthresh值，然后开始执行拥塞避免算法，使拥塞窗口缓慢地线性增大。



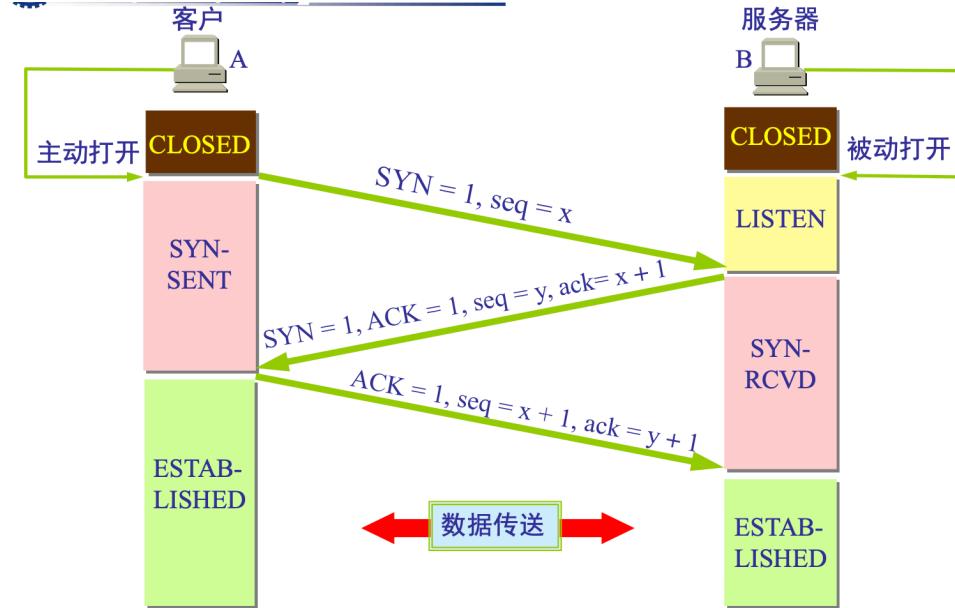
#### 7.4.2.8 TCP的运输连接管理

- 运输连接的概念

- 运输连接有三个阶段：连接建立、数据传送、连接释放。
- 运输连接的管理就是使运输连接的建立和释放都能正常进行。
- 连接建立过程中要解决以下三个问题：
  - 要使每一方能够确知对方的存在。
  - 要允许双方协商一些参数。
  - 能够对运输实体资源进行分配。
- 客户服务器方式
  - TCP连接的建立都是采用客户服务器方式。
  - 主动发起连接建立的应用进程叫做客户。
  - 被动等待连接建立的应用进程叫做服务器。

- TCP的连接建立

- 三次握手

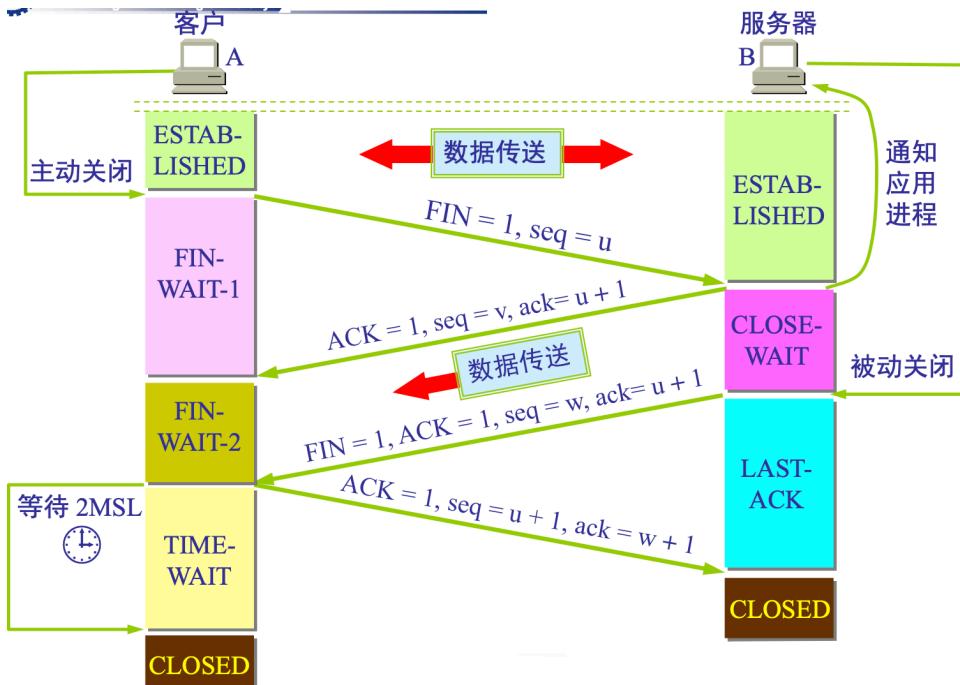


- A的TCP向B发出连接请求报文段，其首部中的同步位 $SYN = 1$ ，并选择序号 $seq = x$ ，表明传送数据时的第一个数据字节的序号是x。
- B的TCP收到连接请求报文段后，如同意，则发回确认。
- B在确认报文段中应使 $SYN = 1$ ， $ACK = 1$ ，其确认号 $ack = x + 1$ ，自己选择的序号 $seq = y$ 。
- A收到此报文段后向B给出确认，其 $ACK = 1$ ，确认号 $ack = y + 1$ 。
- A的TCP通知上层应用进程，连接已经建立。
- B的TCP收到主机A的确认后，也通知其上层应用进程：TCP连接已经建立。

- TCP 的连接释放

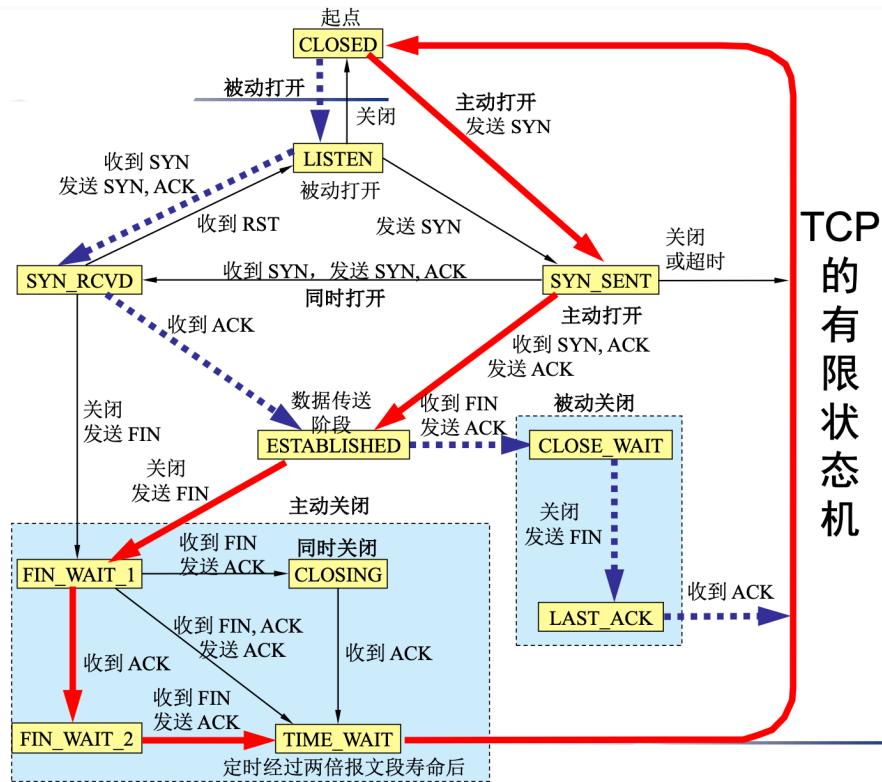
- 四次挥手释放

- 



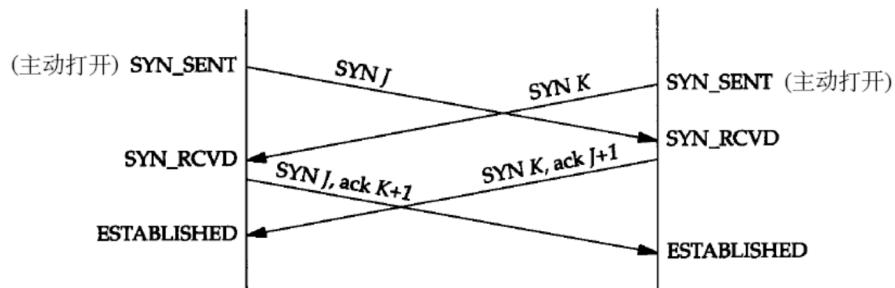
- 数据传输结束后，通信的双方都可释放连接。现在A的应用进程先向其TCP发出连接释放报文段，并停止再发送数据，主动关闭TCP连接。
  - A把连接释放报文段首部FIN置1，其序号seq = u，等待B的确认。
  - B发出确认，确认号ack = u + 1，而这个报文段自己的序号seq = v。
  - TCP服务器进程通知高层应用进程。
  - 从A到B这个方向的连接就释放了，TCP连接处于半关闭状态。B若发送数据，A仍要接收。
  - 若B已经没有要向A发送的数据，其应用进程就通知TCP释放连接。
  - A收到连接释放报文段后，必须发出确认。
  - 在确认报文段中ACK = 1，确认号ack = w + 1，自己的序号seq = u + 1。
  - TCP连接必须经过时间2MSL后才真正释放掉。
  - A必须等待2MSL的时间
    - 为了保证A发送的最后一个ACK报文段能够到达B。
    - 防止“已失效的连接请求报文段”出现在本连接中。
      - A在发送完最后一个ACK报文段后，再经过时间2MSL，就可以使本连接持续的时间内所产生的所有报文段都从网络中消失。
      - 使下一个新的连接中不会出现这种旧的连接请求报文段。
- 保活计时器
    - 服务器每收到一次客户的数据，就重新设置保活计时器(2h)。
    - 若保活计时器时间内都没收到客户的数据，服务器就发送探测报文段，以后每隔75分钟发送一次。
    - 若一连发送10个探测报文段后仍无客户的响应，则关闭连接。
  - TCP的有限状态机
    - TCP有限状态机的图中每一个方框都是TCP可能具有的状态，每个方框中的大写英文字串是TCP标准所使用的TCP连接状态名。
    - 状态之间的箭头表示可能发生的状态变迁。
    - 箭头旁边的字，表明引起这种变迁的原因，或表明发生状态变迁后又出现什么动作。
    - 图中有三种不同的箭头
      - 粗实线箭头(红色)：表示对客户进程的正常变迁
      - 粗虚线箭头(蓝色)：表示对服务器进程的正常变迁
      - 细线箭头(黑色)：表示异常变迁

## TCP的有限状态机



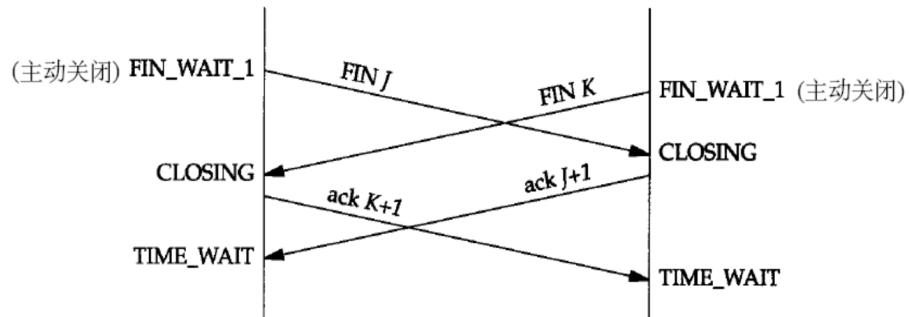
- 同时打开

- 两端同时执行主动打开，交换四个报文段。
- TCP仅建立一条连接，每一端既是客户又是服务器。



- 同时关闭

- 双方都执行主动关闭，交换四个报文段。



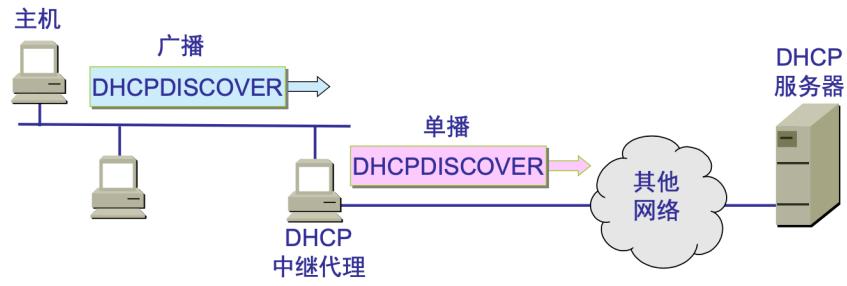
# Chapter8 应用层

## 8.1 应用层概述

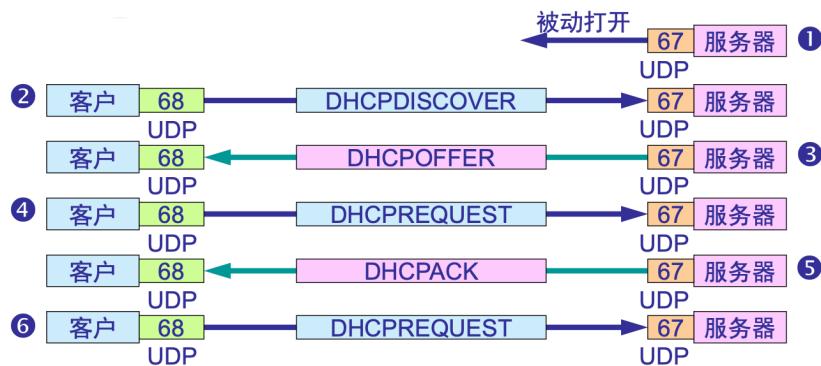
- 应用层主要提供应用进程与通信进程之间的接口，规定应用进程在通信时所遵循的协议。
- 应用层的许多协议都是基于客户/服务器方式
  - 客户和服务器都是指通信中所涉及的两个应用进程。
  - 客户/服务器方式所描述的是进程之间服务和被服务的关系：
    - 客户是服务请求方。
    - 服务器是服务提供方。

## 8.2 DHCP

- 软件协议的参数化
  - 软件协议运行前，需对协议的参数赋值，即协议配置
  - 需配置的参数取决于协议栈
- 因特网主机的协议配置项
  - IP地址
  - 子网掩码
  - 默认路由器的IP地址
  - 域名服务器的IP地址
  - 这些信息通常存储在配置文件中，计算机在引导过程中读取这个文件
- 动态主机配置协议DHCP
  - 提供一种即插即用连网机制。
  - 这种机制允许一台计算机加入新的网络和获取IP地址，不用手工参与。
  - DHCP报文只是UDP用户数据报中的数据。
- DHCP的工作方式
  - C/S。
  - 需要IP地址的主机在启动时，向DHCP服务器广播发送发现报文，这时该主机就成为DHCP客户。
  - 本地网络上所有主机都能收到此广播报文，但只有DHCP服务器才回答此广播报文。
  - DHCP服务器先在其数据库中查找该计算机的配置信息
    - 若找到，则返回找到的信息。
    - 若找不到，则从服务器的IP地址池中取一个地址分配给该计算机。
  - DHCP服务器回答报文为提供报文。
- DHCP中继代理
  - 每一个网络至少有一个DHCP中继代理，配置DHCP服务器的IP地址信息。
  - DHCP中继代理的工作步骤：
    - 接收主机发送的发现报文。
    - 以单播方式向DHCP服务器转发此报文，并等待其回答。
    - 转发DHCP服务器回答的提供报文给主机。



- 租用期：
  - DHCP服务器所分配IP地址有使用时间的限制，称这段时间为**租用期**。
  - 租用期的数值由DHCP服务器决定。
  - DHCP客户可在自己发送的报文中提出对租用期的要求。
- DHCP协议的工作过程



1. DHCP服务器被动打开UDP端口67，等待客户端发来的报文。
2. DHCP客户从UDP端口68发送DHCP发现报文。
3. 凡收到DHCP发现报文的DHCP服务器都发出DHCP提供报文，DHCP客户可能收到多个DHCP提供报文。
4. DHCP客户选择其中的一个，并广播DHCP请求报文，告知选定的DHCP服务器。
5. 被选择的DHCP服务器发送确认报文DHCPACK，DHCP客户进入已绑定状态，并可开始使用得到的临时IP地址。DHCP客户根据服务器提供的租用期T，设置两个计时器T<sub>1</sub>和T<sub>2</sub>，它们的超时时间分别是0.5T和0.875T。当超时时间到就要请求更新租用期。
6. 租用期过了一半(T<sub>1</sub>时间到)，DHCP客户发送请求报文DHCPREQUEST，要求更新租用期。
7. DHCP服务器若同意，则发回确认报文DHCPACK。DHCP客户得到新的租用期，重新设置计时器。
8. DHCP服务器若不同意，则发回否认报文DHCPNACK。这时DHCP客户必须立即停止使用原来的IP地址，并重新申请IP地址(回到步骤2)。若DHCP服务器不响应步骤6的请求报文DHCPREQUEST，则在租用期过了87.5%时，DHCP客户必须重新发送请求报文DHCPREQUEST(重复步骤6)，然后又继续后面的步骤。
9. DHCP客户可随时提前终止服务器所提供的租用期，只需向DHCP服务器发送释放报文DHCPRELEASE。

## 8.3 DNS

### 8.3.1 域名系统概述

- 网络主机的标识
  - 因特网采用**层次结构**的命名树作为主机的名字，并使用分布式的域名系统DNS。
- DNS功能：提供域名与IP地址间的映射关系，实现域名到IP地址的解析。
  - 由若干个域名服务器程序完成，域名服务器程序在专设的结点上运行。

- 运行该程序的机器称为**域名服务器**。
- 通信过程中的目标地址转换

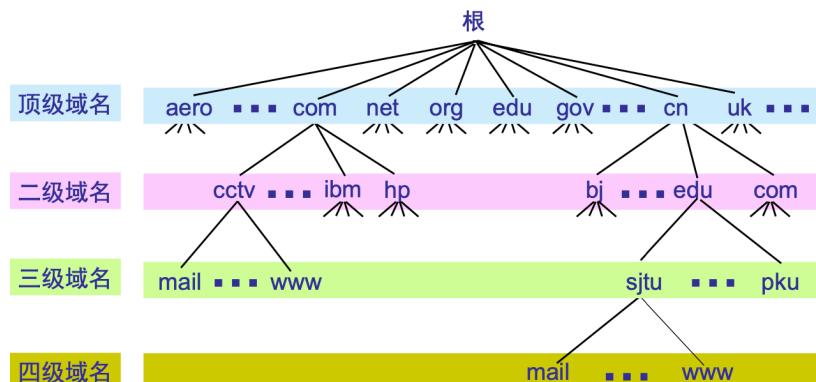


### 8.3.2 因特网的域名结构

- 因特网采用**层次树状结构**的命名方法。
- 任何一个连接在因特网上的服务器，都有一个**唯一的**层次结构的名字，即**域名**。
  - 域名的结构由**标号序列**组成，各标号之间用“.”隔开，各标号分别代表不同级别的**域名**

... . 三级域名 . 二级域名 . 顶级域名

- 因特网域名的分层树状结构



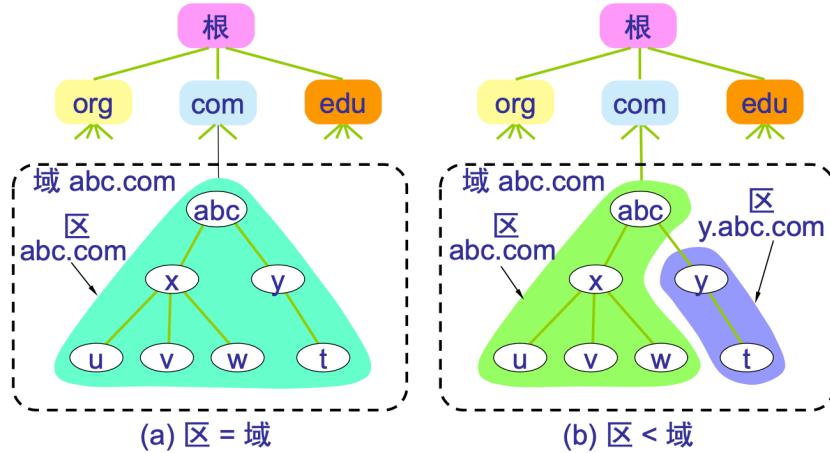
域名：从叶到根的路径，用点分开  
例 mail.sjtu.edu.cn

- 顶级域名TLD
  - 国家顶级域名nTLD，如.cn表示中国
  - 通用顶级域名gTLD，如.com表示公司和企业，.edu表示教育机构
  - 基础结构域名
    - 这种顶级域名只有一个，即arpa，用于反向域名解析。
    - 反向域名格式如：170.103.30.218.in-addr.arpa
    - 实例：邮件服务器采用反向域名解析功能，拒绝接收所有没有注册域名的地址发来的消息。
- 域名的特点
  - 大小写不敏感。
  - 新建一个域，必须征得**所属域**的同意。
  - 命名遵循**组织界限**，而非物理网络。
  - 域名是逻辑概念，便于人的使用和记忆。
  - 域名中标号长度受限，各级标号组成的完整域名总长度受限。

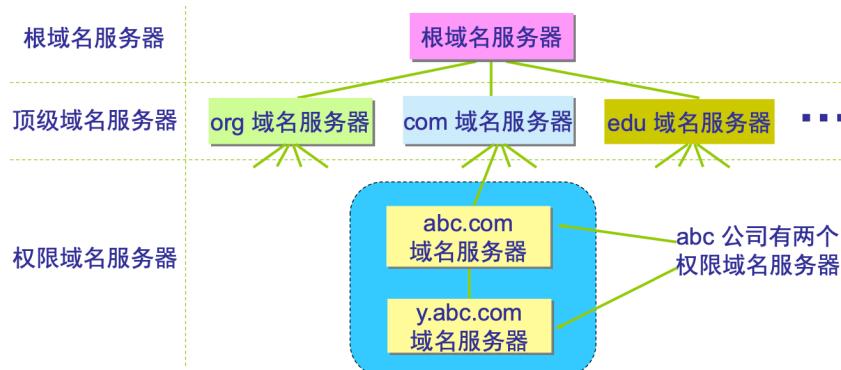
### 8.3.3 域名服务器

- 域名服务器的功能：提供域名解析。
- 域名服务器的管辖范围：不是以“域”为单位，而是以“区”为单位
  - 一个服务器所负责管辖的范围叫做区。
  - 各单位根据具体情况来划分自己管辖范围的区，但在一个区中的所有节点必须是能够连通的。
  - 每一个区设置相应的权限域名服务器，用来保存该区中的所有主机的域名到IP地址的映射。

区的不同划分方法举例



- 域名服务器的层次结构

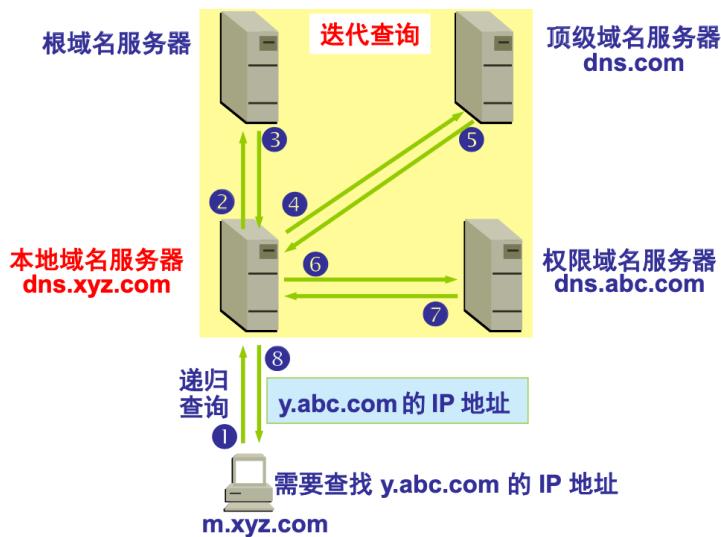


- 域名服务器的四种类型：根域名服务器、顶级域名服务器、权限域名服务器、本地域名服务器
- 本地域名服务器(默认域名服务器)
  - 当一个主机发出DNS查询请求时，这个查询请求报文就发送给本地域名服务器。
  - 每一个因特网服务提供者ISP，或一个大学，甚至一个大学里的系，都可以拥有一个本地域名服务器。
- 根域名服务器
  - 最重要的域名服务器，所有的根域名服务器都知道所有的**顶级域名服务器的域名和IP地址**。
  - 不管是哪一个本地域名服务器，若要对因特网上任何一个域名进行解析，只要自己无法解析，就**首先求助于根域名服务器**。
  - 在因特网上**共有13套装置**的逻辑根域名服务器。
  - 根域名服务器并不直接把域名直接转换成IP地址。
  - 在使用迭代查询时，根域名服务器把下一步应当找的顶级域名服务器的IP地址告诉本地域名服务器。
- 顶级域名服务器

- 负责管理在该顶级域名服务器注册的所有二级域名。
- 当收到DNS查询请求时，就给出相应的回答(可能是最后的结果，也可能是下一步应当找的域名服务器的IP地址)。
- 权限域名服务器
  - 负责一个区的域名解析。
  - 当一个权限域名服务器还不能给出最后的查询回答时，就会告诉发出查询请求的 DNS 客户，下一步应当找哪一个权限域名服务器。
- 提高域名服务器的可靠性
  - DNS域名服务器都把数据复制到几个域名服务器来保存，其中的一个是主域名服务器，其他的是辅助域名服务器。
  - 当主域名服务器出故障时，辅助域名服务器可以保证DNS的查询工作不会中断。
  - 主域名服务器定期把数据复制到辅助域名服务器中，而更改数据只能在主域名服务器中进行，以保证数据的一致性。

### ● 域名的解析过程

- 主机向本地域名服务器的查询——递归查询
  - 如果本地域名服务器不知道被查询域名的IP地址，那么本地域名服务器就以DNS客户的身份，向其他根域名服务器继续发出查询请求报文。
- 本地域名服务器向根域名服务器的查询——迭代查询
  - 当根域名服务器收到本地域名服务器的迭代查询请求报文时，要么给出所要查询的IP地址，要么告诉本地域名服务器下一个查询的域名服务器，由本地域名服务器进行后续的查询。



### ● 名字的高速缓存

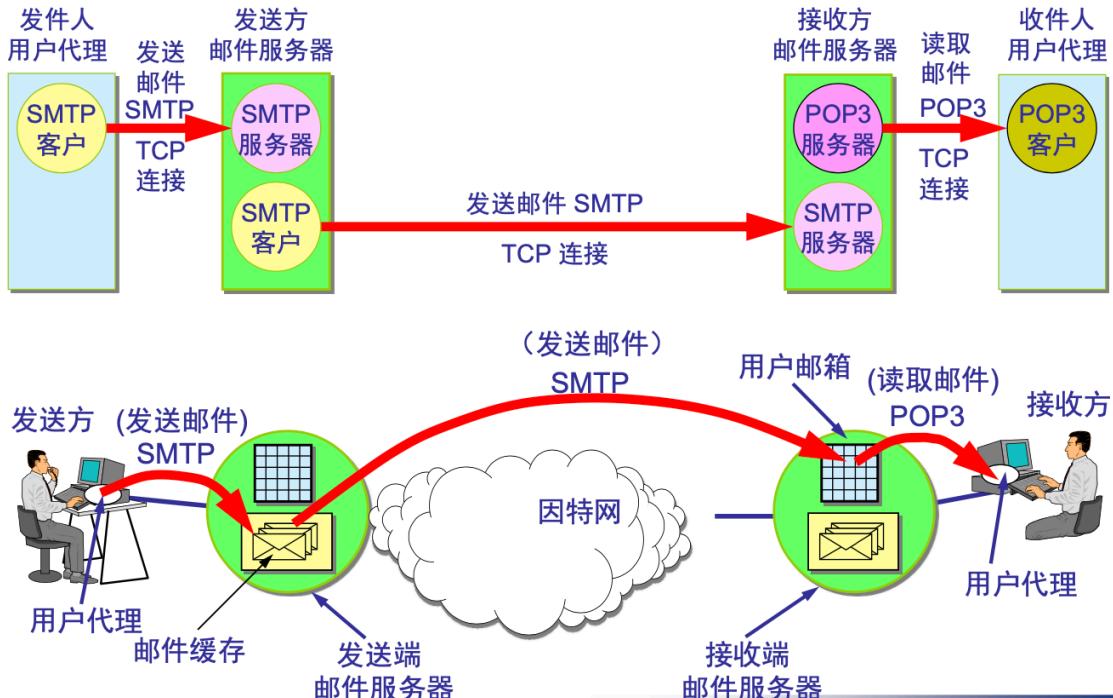
- 每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从何处获得名字映射信息的记录。
- 为保持高速缓存中的内容正确，域名服务器应为每项内容设置计时器，并处理超过合理时间的项。

## 8.4 E-mail

### 8.4.1 概述

- 电子邮件：将邮件发送到收件人使用的邮件服务器，并放在其中的收件人邮箱中，收件人可随时上网到自己使用的邮件服务器进行读取。
- 电子邮件的标准
  - 发送邮件的协议：SMTP

- 读取邮件的协议：POP3和IMAP
- MIME(多功能Internet邮件扩充服务)：在其邮件首部中说明了邮件的数据类型，使用MIME可在邮件中同时传送多种类型的数据。
- 电子邮件的主要组成构件



- 用户代理UA
  - 用户与电子邮件系统的接口，是**电子邮件客户端软件**。
  - 用户代理的功能：撰写、显示、处理和通信。
- 邮件服务器
  - 功能：发送和接收邮件，同时还要向发信人报告邮件传送的情况。
  - 按照**C/S方式**工作，邮件服务器需要使用发送和读取两个不同的协议。
  - 邮件服务器既可以作为客户(SMTP客户)，也可以作为服务器(SMTP服务器)。
- 电子邮件的发送和接收步骤
  1. 发件人调用PC机中的用户代理，撰写和编辑要发送的邮件。
  2. 发件人的用户代理将邮件用**SMTP协议**发给发送方邮件服务器。
  3. SMTP服务器将邮件临时存放在**邮件缓存队列**中，等待发送。
  4. 发送方邮件服务器的**SMTP客户**与接收方邮件服务器的**SMTP服务器**建立**TCP连接**，然后将邮件缓存队列中的邮件依次发送出去。
  5. 运行在接收方邮件服务器中的**SMTP服务器**进程收到邮件后，把邮件放入收件人的**用户邮箱**中，等待收件人进行读取。
  6. 收件人收信时，运行PC机中的用户代理，使用**POP3(或IMAP)**协议读取发送给自己的邮件。

注意：**POP3服务器和POP3客户之间的通信由POP3客户发起**。

- 电子邮件地址的格式：收件人邮箱名@邮箱所在主机的域名

#### 8.4.2 简单邮件传送协议SMTP

- SMTP的功能：在两个相互通信的SMTP进程之间应如何交换信息。
- SMTP使用客户服务器方式。

#### 8.4.3 电子邮件的信息格式

- 电子邮件分为**信封**和**内容**两部分。
- 邮件内容的**首部**格式已规定，邮件的**主体**由用户自由撰写。
- 用户写好首部后，邮件系统**自动**将信封所需的信息提取出来并写在信封上。
- 邮件内容的首部
  - “To:”后面填入一个或多个收件人的电子邮件地址。
  - “Subject:”是邮件的主题，它反映了邮件的主要内容，便于用户查找邮件。
  - 抄送“Cc:”表示应给某些人发送一个邮件副本。
  - “From”和“Date”表示发信人的电子邮件地址和发信日期。
  - “Reply-To”是对方回信所用的地址。

#### 8.4.4 邮件读取协议

- **邮局协议POP(POP3)**
  - 简单、功能有限的邮件读取协议(POP3)
  - 客户服务器工作方式
    - 接收邮件的用户PC机：运行POP客户程序。
    - 邮件服务器：运行POP服务器程序。
  - 用户从POP服务器读取邮件后，服务器就删除该邮件。
  - **POP3功能扩充**：设置邮件读取后仍在服务器存放一段时间。
- **IMAP协议(IMAP4)**
  - 客户服务器工作方式。
  - IMAP是联机协议，用户在PC机上就可操纵邮件服务器的邮箱
    - 创建邮箱文件夹。
    - 在邮箱文件夹中移动邮件。
    - 当用户打开服务器上邮箱时，就可看到邮件首部。若用户需要打开某个邮件，则该邮件才传到用户的计算机上。
  - IMAP协议特点
    - 在用户未发出删除邮件命令之前，IMAP服务器邮箱中的邮件一直保存。
    - 允许收件人只读取邮件中的某一部分。

#### 8.4.5 基于万维网的电子邮件

- 邮件发送方与发送方邮件服务器间：使用HTTP协议。
- 两个邮件服务器间：使用SMTP协议。
- 邮件接收方与接收方邮件服务器间：使用HTTP协议。



#### 8.4.6 通用因特网邮件扩充MIME

- SMTP缺点
  - SMTP限于传送7位的ASCII码。
  - SMTP服务器会拒绝超过一定长度的邮件。
- MIME的特点
  - MIME并没有改动SMTP或取代它。
  - MIME增加了邮件主体的结构，并定义了传送非ASCII码的编码规则。
- MIME和SMTP的关系



### 8.5 万维网WWW

#### 8.5.1 万维网概述

- 万维网WWW
  - 不是传统意义上的物理网络，而是在超文本、超媒体基础上形成的信息网，即信息意义上的网络。
  - 运用链接的方法，从因特网上一个站点访问另一个站点，主动、按需地获取丰富的信息。
- 超媒体与超文本
  - 万维网是分布式超媒体系统，它是超文本系统的扩充。
  - 超文本：
    - 文档仅包含文本信息，由多个信息源链接成，利用一个链接可找到另一个文档。
    - 链接到的文档可以位于世界上任何一个接在因特网上的超文本系统中。
  - 超媒体：
    - 文档包含文本、图形、图像、声音、动画、视频图像信息。
- 万维网的工作方式
  - 以客户服务器方式工作。
  - 浏览器就是在用户计算机上的万维网客户程序。
  - 万维网文档所驻留的计算机则运行服务器程序，该计算机也称为万维网服务器。
  - 客户程序向服务器程序发出请求，服务器程序向客户程序送回客户所要的万维网文档。
  - 在一个客户程序主窗口上显示出的万维网文档称为页面。
- 万维网需解决的问题
  - 怎样标志分布在整个因特网上的万维网文档

- 统一资源定位符URL。
- 每一个文档在整个因特网的范围内具有唯一的标识符URL。
- 用何协议实现万维网上各种超链的链接
  - 万维网客户程序与服务器程序之间，使用超文本传送协议HTTP进行信息交互。
  - HTTP是应用层协议，使用TCP连接进行可靠的传送。
- 怎样使各种万维网文档都能在因特网上的各种计算机上显示出来，同时使用户清楚地知道在什么地方存在着超链
  - 超文本标记语言HTML：用超链，可从本页面的某处链接到因特网上的任何一个万维网页面，并且能够在计算机屏幕上将这些页面显示出来。
  - 可扩展超文本标识语言XHTML。
- 怎样使用户能够很方便地找到所需的信息
  - 各种的搜索工具，即搜索引擎。

### 8.5.2 统一资源定位符URL

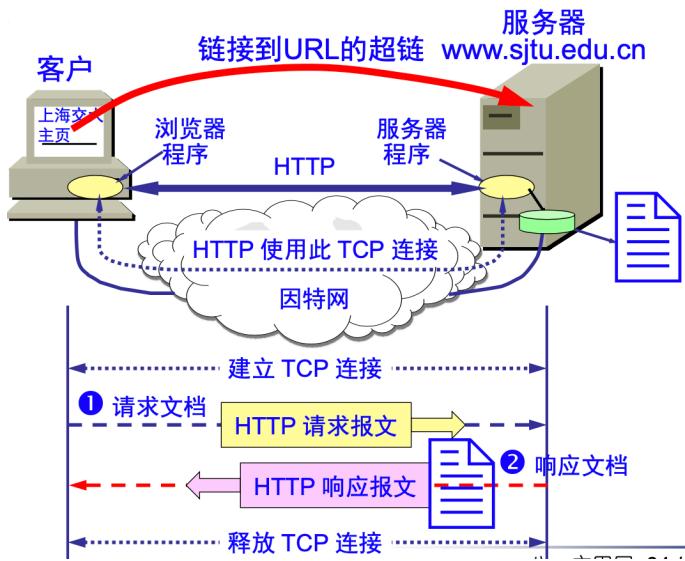
- URL的作用：
  - 表示可从因特网上得到的资源的位置和访问方法。
  - 利用URL，进行资源定位，然后系统就可以对资源进行各种操作，如存取、更新、替换和查找其属性。
  - URL相当于一个文件名在网络范围的扩展，因此URL是与因特网相连的机器上的任何可访问对象的一个指针。
- URL的一般形式
  - 由以冒号隔开的两大部分组成，字符对大小写没有要求。

**<协议>://<主机>:<端口>/<路径>**

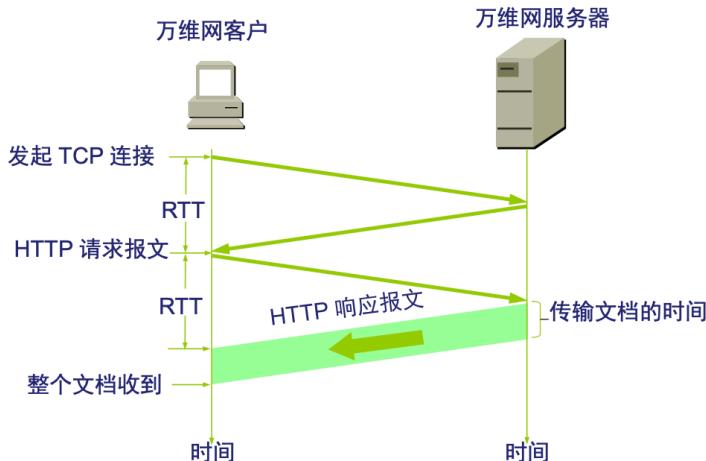
- <协议>：ftp—文件传送协议FTP、http—超文本传送协议HTTP、https—基于TLS/SSL的HTTP。
- <主机>：存放资源的主机在因特网中的**域名**。
- <端口>/<路径>：有时可省略
  - HTTP的默认端口号是80，通常可省略。
  - 若省略文件的<路径>项，则URL就指到因特网上的某个**主页**。

### 8.5.3 超文本传送协议HTTP

- HTTP是**面向事务**的应用层协议。
- **C/S方式**，使得万维网上能够可靠地交换文件。
- 超文本间的链接通过HTTP协议来传送信息。
- HTTP协议本身是**无连接**的，虽然使用面向连接的**TCP**向上提供的服务。
- HTTP协议是**无状态**的。
- 网页浏览
  - 方法一：在浏览器的地址窗口，输入所要找页面的URL。
  - 方法二：在某一页面中，用鼠标点击超链接，由浏览器找到所要链接的页面。
- 万维网的工作过程

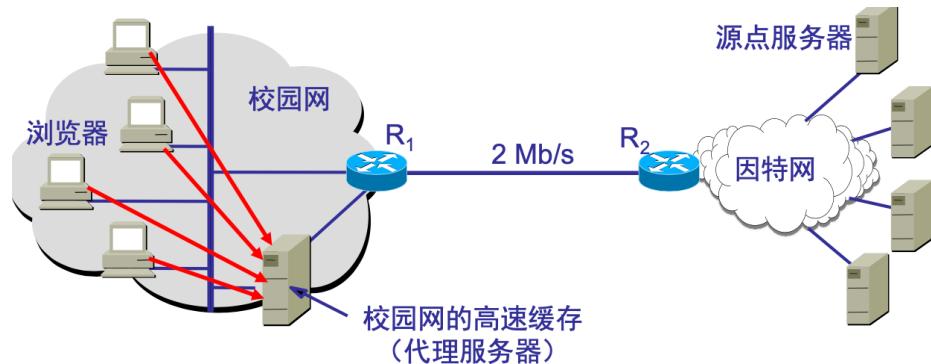


1. 浏览器分析超链指向页面的URL。
  2. 浏览器向DNS请求解析[www.sjtu.edu.cn](http://www.sjtu.edu.cn)的IP地址。
  3. 域名系统DNS解析出上海交通大学服务器的IP地址。
  4. 浏览器与服务器建立TCP连接。
  5. 浏览器发出取文件命令: GET/index.htm。
  6. 服务器给出响应, 把文件index.htm发给浏览器。
  7. TCP连接释放。
  8. 浏览器显示“上海交通大学”主页文件index.htm中的所有文本。
- 请求万维网文档所需的时间



- HTTP/1.1
  - 保活连接
    - 万维网服务器在发送响应后, 仍然在一段时间内保持这条连接, 使同一个客户(浏览器)和该服务器可以继续在这条连接上传送后续的HTTP请求报文和响应报文。
    - 不局限于传送同一个页面上链接的文档, 而是只要这些文档都在同一个服务器上就行。
  - 串行工作方式: 客户在收到前一个响应后, 才能发出下一个请求。
- HTTP/2
  - 引入了HTTP流的概念。
  - 通过抽象HTTP实现, 将不同的HTTP交换并发地复用到同一个TCP连接上, 浏览器可以更有效地重用TCP连接。
  - 并发工作方式: 同一连接同时传输多个请求/响应。
- HTTP/3 QUIC

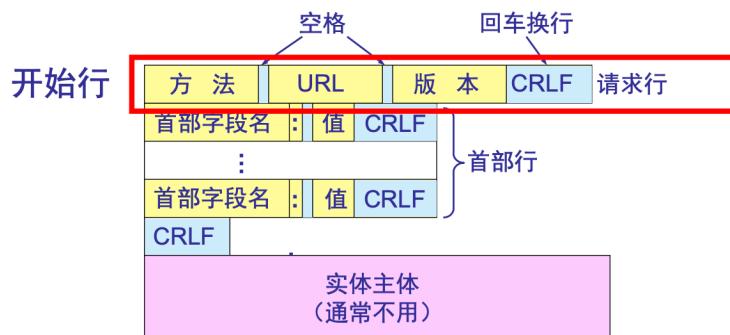
- 基于UDP的低时延互联网传输层协议。
- 同一条QUIC连接上可以创建多个stream，来发送多个HTTP请求，但是QUIC是基于UDP的，一个连接上的多个stream之间没有依赖。
- 代理服务器
  - 代理服务器：代表浏览器发出HTTP请求。
  - 万维网高速缓存把最近的一些请求和响应暂存在本地磁盘中。
  - 当与暂时存放的请求相同的新请求到达时，就将暂存的响应发送出去，而不需要按URL的地址再去因特网访问该资源。
  - 工作过程



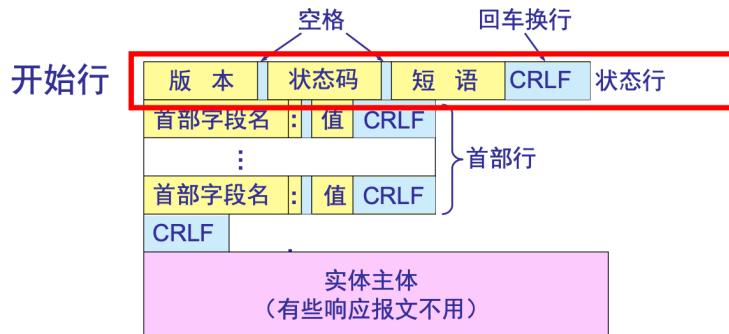
1. 浏览器访问因特网的服务器时，要先与校园网的代理服务器建立TCP连接，并向代理服务器发出HTTP请求报文。
2. 若代理服务器已经存放了所请求的对象，则将此对象放入HTTP响应报文中返回给浏览器。
3. 否则，代理服务器就代表发出请求的用户浏览器，与因特网上的源点服务器建立TCP连接，并发送HTTP请求报文。
4. 源点服务器将所请求的对象放在HTTP响应报文中返回给校园网的代理服务器。
5. 代理服务器收到此对象后，先复制在其本地存储器中，然后再将该对象放在HTTP响应报文中，通过已建立的TCP连接，返回给请求该对象的浏览器。

- HTTP的报文结构

- HTTP有两类报文
  - 请求报文



- 报文由三个部分组成，即开始行、首部行和实体主体。在请求报文中，开始行就是请求行。
- “方法”：对所请求的对象进行的操作，实际上就是一些命令。请求报文的类型由它所采用的方法决定。
- “URL”：所请求资源的URL。
- “版本”：HTTP的版本。
- 响应报文

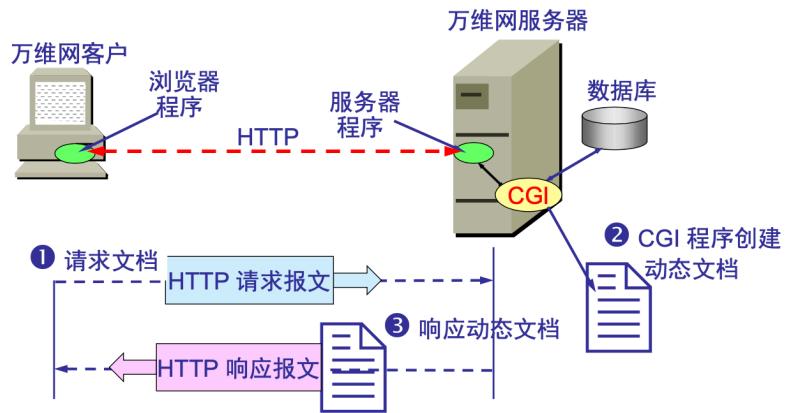


- 开始行是状态行，包括三项内容，即HTTP的版本、状态码，以及解释状态码的简单短语。
  - 由于HTTP面向文本，因此报文中每一个字段都是一些ASCII码串，每个字段的长度都是不确定的。
- Cookie
  - 万维网站点使用Cookie来跟踪用户。
  - Cookie表示在HTTP服务器和客户之间传递的状态信息。
  - 使用Cookie的网站服务器为用户产生一个唯一的识别码，利用此识别码，网站就能够跟踪该用户在该网站的活动。

#### 8.5.4 万维网的文档

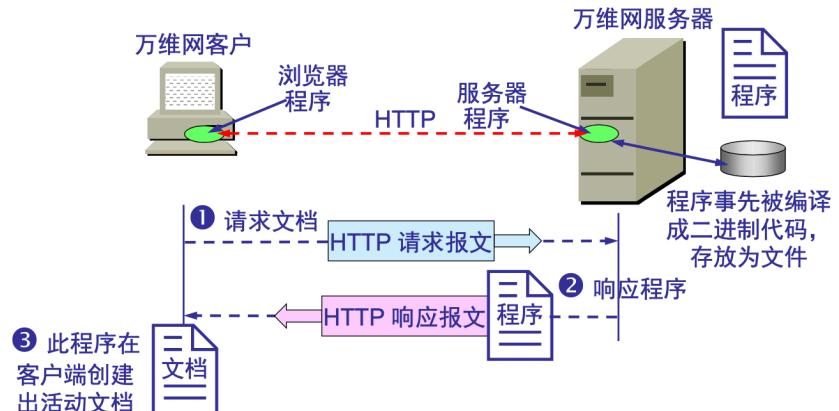
- 超文本标记语言HTML
  - HTML定义了许多用于排版的命令(即标签)。
  - HTML把各种标签嵌入到万维网的页面中，构成HTML文档。
  - HTML文档是一种可以用任何文本编辑器创建的ASCII码文件。
- HTML文档
  - 仅当HTML文档是以.html或.htm为后缀时，浏览器才对此文档的各种标签进行解释。
  - 当浏览器从服务器读取HTML文档后，就按照HTML文档中的各种标签，根据浏览器所使用的显示器的尺寸和分辨率大小，重新进行排版并恢复出所读取的页面。
- 动态万维网文档
  - 静态文档：指该文档创作完毕后就存放在万维网服务器中，在被用户浏览的过程中，内容不会改变。
  - 动态文档：指文档的内容是在浏览器访问万维网服务器时，才由应用程序动态创建。
  - 两者主要差别体现在服务器一端，即文档内容的生成方法不同。
  - 从浏览器的角度看，这两种文档并没有区别。
- 万维网服务器功能扩充
  - 应增加另一个应用程序：处理浏览器发来的数据，并创建动态文档。
  - 应增加一种机制
    - 万维网服务器把浏览器发来的数据传送给这个应用程序。
    - 万维网服务器解释这个应用程序的输出，并向浏览器返回HTML文档。
- 通用网关接口CGI
  - CGI是一种标准，它定义了：
    - 动态文档应如何创建。
    - 输入数据应如何提供给应用程序。
    - 输出结果应如何使用。
  - 遵循CGI标准的动态文档生成程序称为CGI程序——依据输入数据，动态生成文档。

- CGI程序的正式名字是CGI脚本。



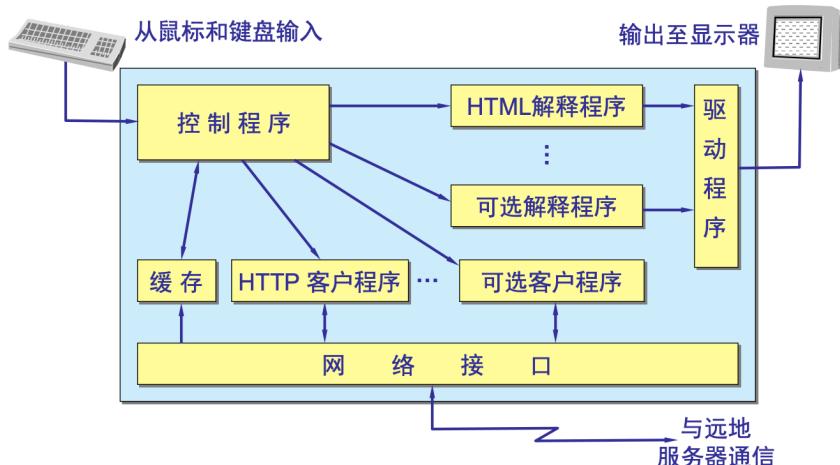
- 活动万维网文档

- **活动文档技术**把所有的工作都转移给浏览器端
  - 每当浏览器请求一个活动文档时，服务器就返回一段程序副本在浏览器端运行。
  - **活动文档程序**可与用户直接交互，并可连续地改变屏幕的显示。
- 活动文档技术不需要服务器的连续更新传送，对网络带宽的要求不高。



- 活动文档的创建技术
  - Java语言可用于创建和运行活动文档。
  - Java 技术中，使用“**小应用程序**”来描述活动文档程序。

- 浏览器的结构



- 浏览器的主要组成部分

- 浏览器有一组**客户程序**、一组**解释程序**，以及管理这些客户程序和解释程序的**控制程序**。

- 控制程序：核心部件，它解释鼠标的点击和键盘的输入，并调用有关的组件来执行用户指定的操作。
  - 当用户用鼠标点击一个超链的起点时，控制程序就调用一个客户程序从远地服务器上取回该文档，并调用解释程序向用户显示该文档。
- 客户程序
  - HTTP客户程序：网页浏览。
  - FTP客户程序：用来获取文件传送服务。
  - SMTP客户程序：使浏览器能够发送和接收电子邮件。
- 解释程序
  - HTML解释程序：将HTML规格转换为适合用户显示硬件的命令，来处理版面的细节。
  - 其他的解释程序则是可选的，如JAVA。

## 8.6 文件传送协议FTP

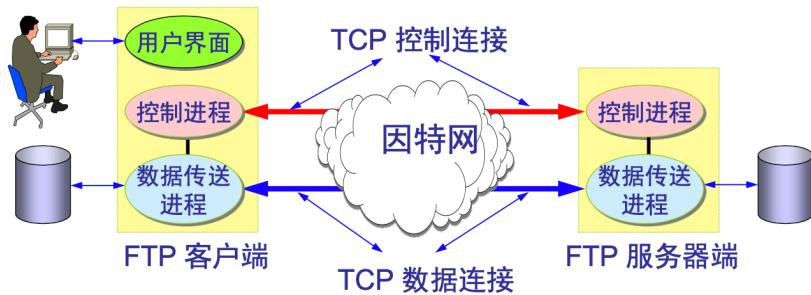
### 8.6.1 FTP概述

- FTP提供交互式的访问，允许客户指明文件的类型与格式，并允许文件具有存取权限。
- FTP屏蔽了各计算机系统的细节，因而适合于在异构网络中任意计算机之间传送文件。

### 8.6.2 FTP工作原理

- 网络环境下复制文件的复杂性：
  - 计算机存储数据的格式不同。
  - 文件的目录结构和文件命名的规定不同。
  - 对于相同的文件存取功能，操作系统使用的命令不同。
  - 访问控制方法不同。
- 文件传送协议FTP：使用TCP可靠的运输服务，提供文件传送的一些基本的服务，减少或消除在不同操作系统下处理文件的不兼容性。
- FTP使用客户/服务器方式，一个FTP服务器进程可同时为多个客户进程提供服务。
- FTP的服务器进程由两大部分组成：
  - 一个主进程，负责接受新的请求。
  - 若干个从属进程，负责处理单个请求。
- FTP服务器的主进程
  - 主进程的工作步骤
    - 被动打开熟知端口(端口号为21)，使客户进程能够连接上。
    - 等待客户进程发出连接请求。
    - 启动从属进程来处理客户进程发来的请求
      - 从属进程处理客户进程的请求，完毕后即终止。
      - 从属进程在运行期间，可根据需要创建其他一些子进程。
    - 回到等待状态，继续接受其他客户进程发来的请求。
  - 主进程与从属进程的处理并发进行。
- FTP服务器的从属进程
  - 控制进程

- FTP服务器的主进程，每接收到一个客户连接请求，就创建从属进程——控制进程，并建立一条控制连接。
- 数据传送进程
  - FTP服务端的控制进程，在接收到FTP客户发送来的文件传输请求后，就创建从属进程——数据传送进程，并建立一条数据连接，来连接客户端和服务器端的数据传送进程。
  - 数据传送进程负责文件传送，在文件传送完毕后，关闭数据连接。
- 客户与服务器间的连接
  - FTP的客户端与服务器建立两个TCP连接
    - 控制连接
      - 在整个会话期间一直保持打开。
      - 控制连接始终等待客户端和服务器之间的通信，并且将相关命令从客户端传送给服务器，同时将服务器的应答传送给客户端。
      - 客户端发出的文件传送请求通过控制连接发送给服务器端的控制进程，由控制进程创建数据传送进程和数据连接。
    - 数据连接
      - 服务器执行主动打开数据连接，通常也执行主动关闭数据连接，但是当客户端向服务器发送流形式的文件时，则需要客户端关闭数据连接。
      - FTP中传输方式是流方式，并且文件结尾以关闭数据连接为标志，对每一个文件传输或目录列表来说，都要建立一个全新的数据连接。
- FTP使用两个TCP连接



- 端口号使用
  - 当客户进程向服务器进程发出建立连接请求时，连接服务器进程的熟知端口21，同时还要告诉服务器进程自己的另一个端口号，用于建立数据连接。
  - 服务器进程用自己传送数据的熟知端口20与客户进程所提供的端口号建立数据连接。
  - FTP使用了两个不同的端口号，数据连接与控制连接不会发生混乱。

### 8.6.3 FTP服务器的运行模式

- Standard(即PORT方式，主动方式)
  - 客户端首先和服务器的TCP21端口建立控制连接，通过该连接发送命令。
  - 客户端需要发送数据时，利用该控制连接发送PORT命令，PORT命令包含了客户端用什么端口接收数据。
  - 在传送数据的时候，服务器端通过自己的TCP20端口，与客户端的指定端口建立数据连接，通过该连接发送数据。
- Passive(即PASV方式，被动方式)
  - 客户端首先和服务器的TCP21端口建立控制连接，通过该连接发送命令。
  - 客户端需要传送数据时，利用该控制连接发送Pasv命令。FTP服务器收到Pasv命令后，随机打开一

一个高端端口(端口号大于1024), 并通过PORT命令通知客户端在这个端口上传送数据的请求。

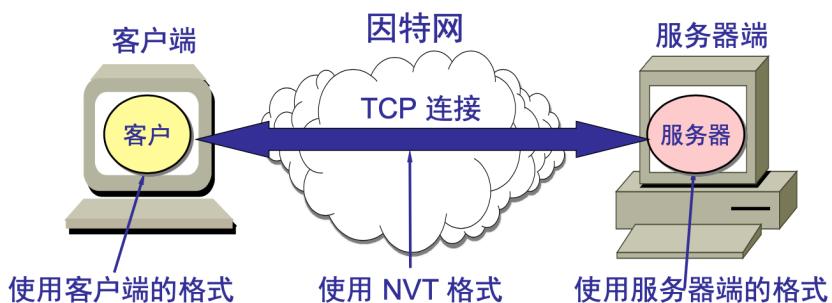
- 客户端连接FTP服务器的此端口, 然后FTP服务器将通过这个端口进行数据的传送。

- 优缺点

- 主动方式对FTP服务器的管理有利, 但对客户端的管理不利。
- 被动方式对FTP客户端的管理有利, 但对服务器端的管理不利。

## 8.7 TELNET

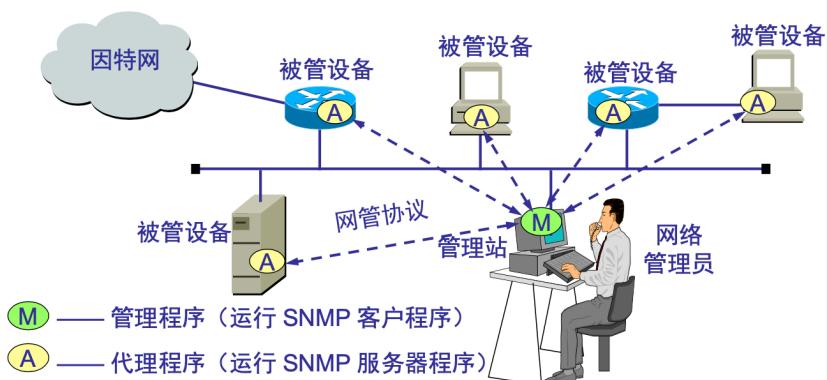
- 用户用TELNET就可在其所在地, 通过TCP连接注册(即登录)到远地的另一个主机上(使用主机名或IP地址)。
- TELNET能将用户的击键传到远地主机, 同时也能将远地主机的输出通过TCP连接返回到用户屏幕。
- 这种服务是透明的, 因为用户感觉到好像键盘和显示器是直接连在远地主机上。
- 客户服务器方式
  - 服务器中的主进程等待新的请求, 并产生从属进程来处理每一个连接。
- 网络虚拟终端NVT格式
  - 客户软件把用户的击键和命令转换成NVT格式, 并递交服务器, 服务器软件把收到的数据和命令, 从NVT格式转换成远地系统所需的格式。
  - 向用户返回数据时, 服务器把远地系统的格式转换为NVT格式, 本地客户再从NVT格式转换到本地系统所需的格式。



## 8.8 SNMP

### 8.8.1 网络管理的基本概念

- 网络管理:
  - 对网络资源进行监视、测试、配置、分析、评价和控制。
  - 常简称为网管。
- 网络管理的一般模型



- 网络管理模型中的主要构件

- 管理站：网络运行中心NOC，是网络管理系统的核心。
  - 管理程序：运行在管理站上的程序，在运行时就成为管理进程。
  - 管理者：
    - 管理站(硬件)或管理程序(软件)。
    - 不是指人而是指机器或软件。
    - 大型网络往往实行多级管理，因而有多个管理者，而一个管理者一般只管理本地网络的设备。
  - 网络管理员指的是人。
  - 被管对象：
    - 网络的每一个被管设备中可能有多个被管对象。
    - 被管设备有时可称为网络元素或网元。
    - 在被管设备中也会有一些不能被管的对象。
  - 网络管理代理程序：
    - 在被管设备上，与管理站中的管理程序进行通信的程序，简称为代理。
    - 在管理程序的命令和控制下，在被管设备上采取本地的行动。
  - 网络管理协议：
    - 管理程序和代理程序之间进行通信的规则。
    - 网络管理员利用网管协议，通过管理站对网络中的被管设备进行管理。
- SNMP工作方式
    - 管理程序和代理程序按客户服务器方式工作：
      - 管理程序运行SNMP客户程序，向某个代理程序发出请求(或命令)。
      - 代理程序运行SNMP服务器程序，返回响应(或执行某个动作)。
    - 在网管系统中往往是一个(或少数几个)客户程序与很多的服务器程序进行交互。
  - SNMP的指导思想
    - SNMP最重要的指导思想：尽可能简单。
    - SNMP的基本功能：包括监视网络性能、检测分析网络差错和配置网络设备等。
      - 在网络正常工作时，SNMP可实现统计、配置、测试等功能。
      - 当网络出故障时，可实现各种差错检测和恢复功能。
    - 虽然SNMP是在TCP/IP基础上的网络管理协议，也可扩展到其他类型的网络设备上。
  - SNMP的管理站和委托代理
    - 整个系统必须有一个管理站，管理进程和代理进程利用SNMP报文(UDP)进行通信。
    - 若网络元素使用另外的网络管理协议，可使用委托代理
    - 提供如协议转换和过滤操作等功能，对被管对象进行管理。
  - SNMP的组成：管理信息结构SMI、管理信息库MIB、SNMP协议。

### 8.8.2 管理信息结构SMI

- SMI的功能：
  - 被管对象的命名方式。
  - 用来存储被管对象的数据类型。
  - 在网络上传送的管理数据的编码。
- SMI规定所有被管对象必须在命名树上。
- 被管对象的数据类型

- SMI使用ASN.1来定义数据类型。
- SMI把数据类型分为两大类：简单类型、结构化类型。
- 编码方法
  - 基本编码规则BER
    - 发送端用BER编码，将用ASN.1所表述的报文转换成唯一的比特序列。
    - 接收端用BER解码，得到该比特序列所表示的ASN.1报文。
  - 数据元素：表示为T-L-V三个字段组成的序列。

### 8.8.3 管理信息库MIB

- 管理信息库MIB：被管对象维持的可供管理程序读写的若干控制和状态信息。
- 管理程序使用MIB中这些信息的值对网络进行管理。

### 8.8.4 SNMP的协议数据单元和报文

- SNMP协议
  - 定义了管理站和代理之间所交换的分组格式，所交换分组包含各代理中的对象(变量)名及其状态(值)。
  - 负责读取和改变这些数值。
- SNMP的管理功能：
  - “读”操作，用get报文来检测各被管对象的状况。
  - “写”操作，用set报文来改变各被管对象的状况。
- SNMP操作方式
  - 探询
    - SNMP管理进程定时向被管理设备周期性地发送探询信息。
    - 优点：
      - 可使系统相对简单。
      - 能限制通过网络所产生的管理信息的通信量。
    - 缺点：
      - 探询管理协议不够灵活。
      - 所能管理的设备数目不能太多。
      - 探询系统的开销较大。
  - 自陷
    - 指被管对象不经过询问就能发送的信息。
    - 当被管对象的代理检测到有事件发生时，就检查其门限值。代理只向管理进程报告达到某些门限值的事件(即过滤)
      - 仅在严重事件发生时才发送自陷
      - 自陷信息很简单且所需字节数很少。
- SNMP工作方式
  - SNMP使用无连接的UDP，客户服务器方式。
  - 服务器端：
    - 被管设备上，运行代理程序。
    - 用熟知端口161来接收探询报文(get或set报文)和发送响应报文。

- 客户端：
  - 管理站上，运行管理程序。
  - 使用熟知端口162来接收来自各代理的trap报文。