计算机网络与通信

考试重点

- 协议栈：五层协议的名称、功能、协议、分组名称、各种设备处理的协议
- 基础知识：时延的类型与计算
- 应用层：HTTP协议报文、HTTP协议非持续连接与持续连接、SMTP协议与衍生协议的功能
- 运输层：可靠数据传输（StopWait、GBN、SN、TCP）、TCP协议的拥塞控制、CRC校验
- 网络层数据层面：子网划分、CIDR、NAT、IP协议报文
- 网络层控制层面：路由转发、链路状态选择算法、距离向量选择算法
- 链路层：MAC地址、ARP协议、以太网协议、争用期、交换机

补充

- 运输层：TCP建立连接的过程、TCP的流量控制、TCP Tahoe TCP Reno
- 网络层：DHCP、ICMP、路由器的三种交换结构、LS、OSPF、DV、RIP、BGP协议
- 网关路由器
- 链路层：码分多址

一、计算机网络与因特网

1.1 什么是因特网

1.1.1 具体构成

- 硬件层面
    - 主机/端系统：诸如桌面PC、$Linux$工作站、平板电脑等与因特网互联的设备

        > 端系统也称主机$(host)$，因为其能运行应用程序；主机又可以分为客户$(client)$和服务器$(server)$

    - 端系统通过**通信链路**和**分组交换机**连接到一起
    - 通信链路：由不同类型的物理媒体组成，包括同轴电缆、铜线、光纤和无线电频谱
    - 分组交换机：用于连接不同的通信链路传输数据，包括路由器和链路层交换机等
    - 端系统通过**因特网服务提供商**(ISP)接入因特网
- 软件层面
    - **协议**$(protocol)$控制着因特网中信息的接受和发送，其主要协议统称为TCP/IP
    - 协议定义了在两个或多个通信实体之间交换的报文的格式和顺序，以及报文发送和/或接收报文所采取的动作

1.2 网络边缘

1.2.1 接入网

- 基本概念
    - 接入网：将端系统物理连接到其**边缘路由器**的网络

        > 接入网的本质作用：通过各种方式使主机连接到路由器，以使得任意两个端系统间能够相互通信

    - 边缘路由器：端系统连接到其他任何远程端系统的路径上的**第一台**路由器
- 接入方式
    - 点对点方式

        - 数据用户线DSL：通过本地电话公司获得互联网接入，本地电话公司即为其ISP

          > 不对称数字用户线$ADSL$通过在不同的频段进行编码使得电话线能够同时承载数据信号与电话信号。

        - 光纤到户$FTTH$：将光纤从本地中心局直接连接到房间，通过光猫（光调制解调器）转换光电信号，使用双绞线连接电脑
        - 卫星：电脑连接卫星信号接收机，通过无线电波与卫星相连，卫星又通过无线电波与地面上另一接收机相连
    - 以太网方式
        - 有线以太网：端系统通过双绞线连到以太网，以太网连接至边缘路由器
        - $WiFi$：端系统连到$WiFi$，$WiFi$连接至边缘路由器
    - 广域无线接入方式
        - 由电信运营商提供，如$3G$、$4G$、$5G$

1.2.2 物理媒体

- 导引型媒体
    - 双绞铜线：由两根并行铜线组成，如传统电话线
    - 同轴电缆：由两根同心铜线组成，如电缆
    - 光纤：能够引导光脉冲，误码率低
- 非导引型媒体
    - 无线电波：通过电磁频谱传递信号

1.3 网络核心

1.3.1 电路交换

- 特点（在线交互程序）
    - 数据交换前需建立一条从发送端到接收端的物理通路
    - 在数据交换的全部时间，用户始终占用端到端的固定传输信道，且**只能为其所用**
    - 所有用户平分网络链路的传输容量，且**传输速率恒定**
    - 交换双方可以**实时**进行数据交换，不会存在延迟
    - 适合传送**大量数据**，传送分组时间远大于连接建立时间
- 复用方式
    - 频分复用FDM
        - 所有连接共用链路的频谱，每个连接专用一个频段，其宽度称为**带宽**
        - 每条连接连续地得到部分带宽
    - 时分复用TDM
        - 时间被划分为固定期间的帧，每个帧被划分为固定数量的时隙
        - 链路为每个连接在每个帧中指令一个时隙，这些时隙由每个连接单独使用
        - 每条连接周期性地得到全部带宽
    - 复用方式的对比

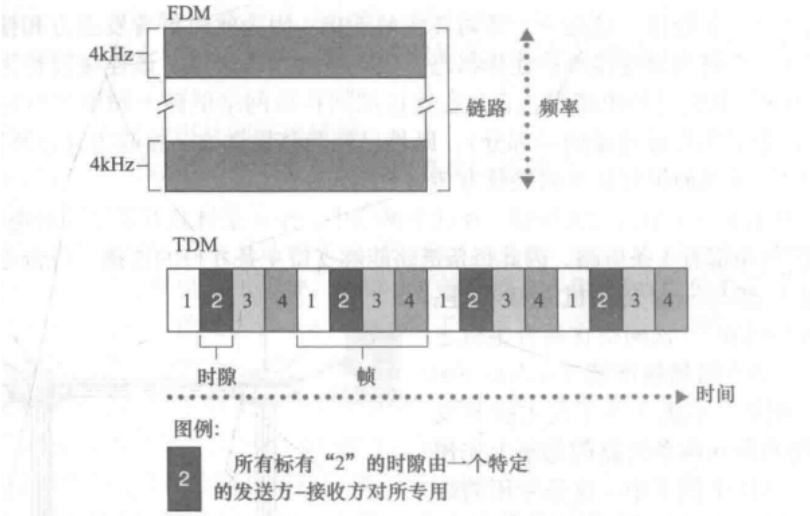


图 1-14　对于 FDM，每条电路连续地得到部分带宽。对于 TDM，每条电路在短时间间隔（即时隙）中周期性地得到所有带宽

- 举例说明：
  - 问：设所有链路速率皆为$1.536Mbps$，每条链路使用有24个时隙的TDM，建立端到端的电路需要500毫秒，则计算通过电路交换网络将一个640000比特长的文件从主机A传送到主机B需要多长时间
  - 答：$(640kb)/(1.536Mbps/24)+0.5=10.5s$
- 缺点：
  - 计算机之间的数据交换往往具有**突发性**和**间歇性**特征，而对电路交换而言，用户支付的费用则是按用户占用线路的时间来收费
  - 只要在通话双方建立的通路中任何一点出了故障，就必须重新拨号建立新连接，这对紧急和重要通信很不利。

1.3.2 分组交换

- 特点
  - 将要发送的报文分解成若干个小部分，称为分组

    > 每个分组都通过通信链路和分组交换机传送，以等于该链路最大传输速率地速度通过通信链路
    >
    > 通过某链路发送一个L比特的分组，链路传输速率为$R\,bit/s$，则传输分组的时间为L/R秒

  - 存储转发传输：交换机能够开始向输出链路传输该分组的第一个比特前，**必须接收到整个分组**
  - 每个分组传输的链路可能不同，且存在冗余路由

    > 在分组传输前**不必预先确定分组的传输路径**，而是在传输到某个分组交换机后根据转发表查找转发端口

  - 网络核心中每个交换结点均为共享结点
  - 适合传送**突发数据**
- 电路类型
  - 数据报网络
  - 虚电路网络（结合电路连接的优点）
    - 虚电路需要**建立连接**，即建立虚电路链路

        - **在建立连接时决定链路的路由**，在整个连接过程中保持不变
        - 在链路通过的每个节点，预留一定的资源
        - 每个分组携带一个标识（虚电路号），根据该标识知道该从哪个虚电路传输数据
        - 虚电路如果不再使用，需要**释放相关的资源**
- 与电路交换比较（在线交互程序）
    - 在相同条件下，分组交换能够比电路交换支持更多的用户

        > 设一条$1Mbps$的链路，每个用户需要$100kbps$，电路交换模式下仅支持10个用户；设分组交换下1个用户活跃的概率为0.1，在35个用户条件下，11个及以上用户同时活动的概率为0.0004，即10个及10个以内用户同时活跃的概率为0.9996，基本上与电路交换性能相当

    - 当用户数较少时，分组交换能够获得比电路交换更好的性能

        > 同一时刻仅有一个用户传输$1M$的数据，电路交换需要$10s$，分组交换需要$1s$

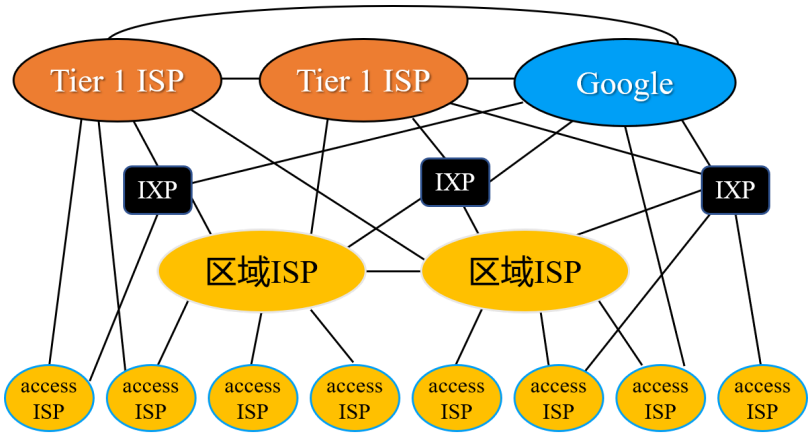
    - 在数据量大时，分组交换的传输时延比电路交换大

        > 通过由n条速率均为R的链路组成的路径，从源到目的地发送一个分组的时延为$d_{端到端} = N\frac{L}{R}$

- 缺点
    - 分组在各结点存储转发时因要排队**会造成一定的时延**，当网络通信量过大时，这种时延可能会很大。
    - 各分组必须携带一定的**控制信息（说明信息）**，从而带来额外开销。

1.3.3 ISP网络模型

- 各种ISP互相连在一起
- 低级ISP可以连入高级ISP进行互通
- 同级ISP之间通过IXP和对等链路进行**对等**链接



1.4 分组交换网络的衡量

1.4.1 时延

节点总时延主要包括节点处理时延、排队时延、传输时延、传播时延

- 节点处理时延(d_{proc})
    - 检查分组首部并决定该分组导向何处所需的时间
    - 检查比特级别的差错所需的时间

- **排队时延**(d_{queue})
    - 在队列中，分组在链路上等待传输的时间，时延长度取决于**先到达的正在排队的分组数量**
- 传输时延(d_{trans})
    - 将分组的所有比特推向输出链路所需的时间（在线交互程序）
    - 等于分组的长度除以**链路传输速率**，即L/R，其中R的单位为bps、$kbps$、$Mbps$等
- 传播时延(d_{prop})
    - 从输出链路的起点到目的地传播所需的时间，数据以链路的传播速率传播，取决于链路的物理媒体
    - 等于两台路由器间的距离除以**链路传播速率**，即d/s
- 关于传输时延和传播时延（在线交互程序，在线交互程序）

总传输时延$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

1.4.2 丢包

- 排队时延分析（在线交互程序）
    - 设链路传输速率为$R\,bps$，分组大小为$L\,bit/pkt$，分组到达队列的平均速率$a\,pkt/s$
    - 则比特到达队列的平均速率为$La\,bps$，流量强度为La/R
    - $La/R \sim 0$：平均排队时延很小，甚至为0

      $La/R < 1$：时延较小，且会随时间推延而变小

      $La/R = 1$：时延不会变化，具体数值取决于当时队列长度

      $La/R > 1$：比特到达队列的平均速率远超过从队列传输出去的速率，队列将无限增加，排队时延趋于∞
- 丢包
    - 当队列已满时，分组将会被路由器丢弃，分组即会丢失
    - 丢失的分组已经传输到网络核心，但是绝不会从网络发送到目的地
- 在$Windows$操作系统使用$tracert$命令、在$Linux$和Mac操作系统使用$traceroute$命令以测试端到端时延

1.4.3 吞吐量

- $P2P$系统
    - 服务器$\Longrightarrow$路由器$\Longrightarrow$客户端
        - 使用R_s表示服务器与路由器之间的链路速率，R_c表示路由器与客户端之间的链路速率
        - 服务器不能以快于R_s的速率向链路发送比特，路由器也不能以大于R_c的速率发送比特
        - 当$R_s > R_c$时，路由器端将会出现比特等待队列，当$R_s < R_c$时，数据将流畅地传输到客户端
        - 其吞吐量被定义为$min\{R_c, R_s\}$
    - 服务器$\Longrightarrow$路由器$1\Longrightarrow$路由器$2\cdots\Longrightarrow$路由器$N\Longrightarrow$客户端
        - 吞吐量为$minR_1, R_2, \cdots, R_N$
- 当网络核心的链路速率远大于接入网时，吞吐量的主要限制因素便为**接入网**
- 瓶颈链路：在端到端路径上限制了端到端平均吞吐量的链路（在线交互程序）

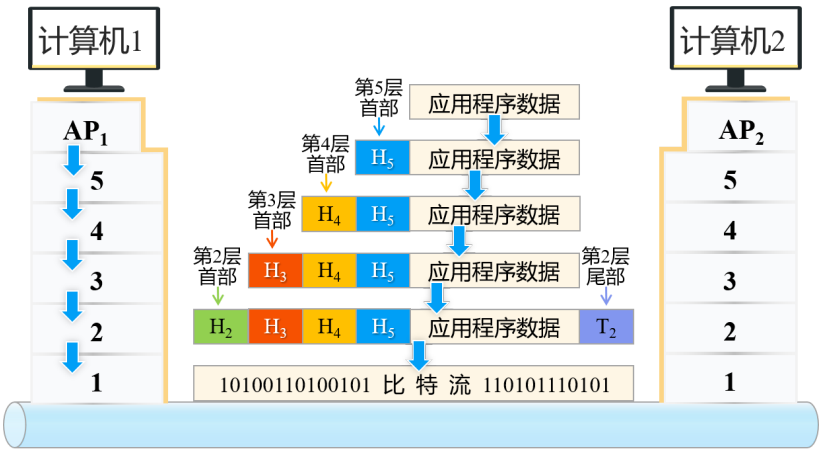
1.5 协议层次及其服务模型

1.5.1 协议分层

- 基本概念
    - **实体**是任何可以发送和接收信息的硬件和软件进程。通常是一个特定的软件模块
    - 不同机器上包含对应层的实体称为**对等体**，如客户端的运输层对应服务端的运输层
    - **服务**指为保证上层对等体之间能互相通信，下层向上层提供的功能
    - **接口**位于每对相邻层之间，定义了下层向上层提供的原语操作和服务
    - **协议数据单元**(PDU)是对等层次上传送数据的单位
    - **服务数据单元**(SDU)是层与层之间交换数据的单位
    - **网络体系结构**是层和协议的集合
    - **协议栈**指一个特定的系统所使用的一系列协议（每层一组协议）
- 因特网的五层结构（在线交互程序）

层次	功能	协议	分组名称
应用层	直接为用户的网络应用程序提供服务	$HTTP$、$SMTP$、FTP、DNS	报文
表示层	**在**OSI**模型中**，统一表示数据的含义	—	—
会话层	**在**OSI**模型中**，数据交换的定界和同步	—	—
运输层	在不同主机的进程间数据传送	TCP、UDP	报文段
网络层	在不同主机间数据传送；选择合适的路由传输运输层分组	网际协议IP，路由协议	数据报
链路层	网络相邻结点间数据传送	PPP、以太网	帧
物理层	在线路上传输比特流	—	—

1.5.2 网络分层模型



二、应用层

2.1 应用层协议原理

2.1.1 进程通信

- 客户与服务器进程
  - 网络应用程序运行后，就变成了网络应用进程
  - 两个在不同端系统的进程，通过跨越计算机网络交换报文而相互通信
  - 网络应用程序由成对进程组成，在一对进程的通信会话场景中，发起通信的进程称为**客户**，等待联系的进程称为**服务器**
- 进程与计算机网络的接口
  - 进程通过一个称为**套接字**$(Socket)$的软件接口向网络发送报文和从网络接收报文
  - 套接字是同一台主机内应用层与运输层的接口，也成为应用程序和网络之间的**应用程序编程窗口**
- 进程寻址
  - 为了与目的主机上进程的通信，需要定义**目的主机的地址**和**目的主机中指定接收进程的标识符**
  - 目的主机地址由32位的IP地址标识
  - 目的进程地址由16位目的地端口号$Port\ Number$标识，如Web服务器的端口号为80，$SMTP$的端口号为25
  - 套接字长度为48位

2.1.2 运输服务简介

- TCP服务
  - **面向连接**：在报文流动前，TCP让客户和服务器互相交换运输层控制信息以为分组运输做好准备（握手），此时，一个TCP连接在两个进程的套接字直接建立
  - **可靠性**：无差错、无字节丢失与冗余、顺序地传输分组
  - **拥塞控制机制**：当网络拥塞时，抑制发送进程
  - 不提供加密机制
- UDP服务
  - 没有握手过程
  - 不可靠数据传送服务
  - 没有拥塞控制机制
  - 不提供加密机制
- SSL安全套接字层
  - 提供加密的TCP连接

- 提供数据完整性和端点鉴别

- 应用层协议与支撑的运输层协议

应用	应用层协议	支撑的运输层协议
电子邮件	$SMTP$	TCP
远程终端访问	$Telnet$	TCP
Web、流式多媒体	$HTTP$	TCP
文件传输	FTP	TCP
因特网电话	SIP、RTP	UDP或TCP

2.2 Web和$HTTP$

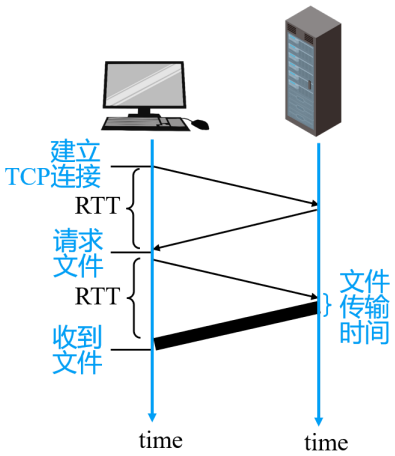
2.2.1 $HTTP$概述

- 相关概念
  - $HTTP$全称为超文本传输协议($HyperText\ Transfer\ Protocol$)
  - Web页面由**对象**组成，如$HTML$文件，$JPEG$图像
  - 通过URL地址引用对象，URL地址由**存放对象的服务器地址**和**对象的路径**组成

    > 如对于URL地址$http://cloudchewie.com/index.html$：
    > $cloudchewie.com$为主机名，$/index.html$为对象路径

  - $Web\ Browser$实现了$HTTP$的客户端，$Web\ Server$实现了$HTTP$的服务端，用于存储Web对象，每个对象由URL路径访问，因此Web是典型的C/S模式
  - $HTTP$负责定义客户向服务器请求Web页面的方式以及服务器向客户返回Web页面的方式，其传递的报文称为$HTTP$报文
  - 客户向其套接字接口发送$HTTP$请求报文并从其套接字接口接收$HTTP$响应报文；当客户发送报文后，该报文即脱离客户控制而进入TCP控制，从而使得应用层协议$HTTP$无需关心报文丢失和运输层的实现细节
  - $HTTP$服务器不保存关于客户的任何信息，是一个**无状态协议**
- 非持续连接
  - 每个TCP连接只传输一个请求报文和一个响应报文
  - 当客户接收$HTTP$响应报文后，TCP连接关闭
  - 如果需要继续发送请求，需要建立全新的TCP连接
  - 可以使用**并行的连接**改善缩短响应时间

    > TCP服务是建立在连接之上的，每次建立连接前，都需要进行三次握手过程，如下图所示：



1. 客户端向服务器发送一个小的TCP报文段
2. 服务器用一个小的TCP报文段进行确认和响应
3. 客户向服务器返回确认并发送$HTTP$请求报文
4. 在这之后，服务器开始发送文件到客户

其中，环节$1-2$占用了一个往返时间RTT，环节$3-4$占用了一个往返时间RTT并且耗费了传输文件的时间

因此，在非持续连接的$HTTP$下，每传送一个对象，就需要经受两个RTT的交付时延

- 持续连接
  - 服务器发送$HTTP$响应报文后保持TCP连接，使得客户的后续请求继续使用该连接进行传送
  - 当该连接经过一定时间间隔（可配置的超时间隔）仍未使用，其服务器就将关闭该连接

2.2.2 $HTTP$报文

- $HTTP$请求报文（在线交互程序）

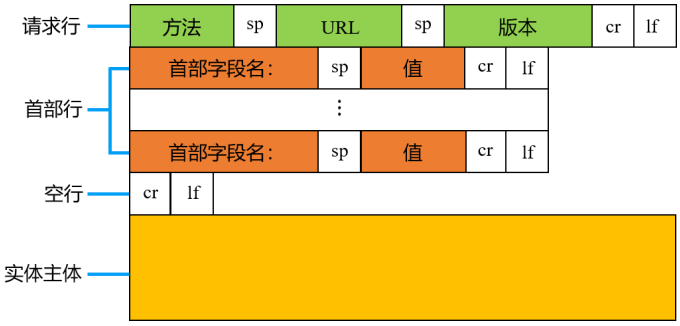
```
#典型的HTTP请求报文
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Mozilla/5.0\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

  - 请求报文的第一行称为**请求行**，包括方法字段、URL字段、$HTTP$版本字段

方法字段	主要作用
GET	向服务器请求指定URL的对象
$POST$	用于向服务器提交表单数据也可以同时请求一个Web页面
$DELETE$	返回响应报文，不包含请求的对象

方法字段	主要作用
PUT	上传的文件放在实体主体字段中，目标路径由URL字段标明
$HEAD$	删除URL字段中指定的文件

    - 其余行称为**首部行**
        - $Host$指明了对象所在的主机
        - $Connection$指明非持续连接$(Close)$和持续连接$(keep-alive)$
        - $Keep-Alive$指明持续连接的超时间隔
        - $User-agent$指明用户浏览器类型，有助于服务器根据不同的用户代理发送相同对象的不同版本
        - $Accept-*$指用户想要得到特定语言、编码格式、字符集的该对象
    - 结尾单独一行回车、换行表示报文首部结束
    - **实体体**，在首部行之后的请求体
        - 当发送GET请求时，实体体为空；
        - 当用户填写表单并发送$POST$请求时，实体体即为用户填写的表单；
        - 当用户填写表单时，也可以使用GET方法，如`/learning/search?key=banana&lang=zh`
    - $HTTP$请求报文的通用格式



- $HTTP$响应报文（在线交互程序）

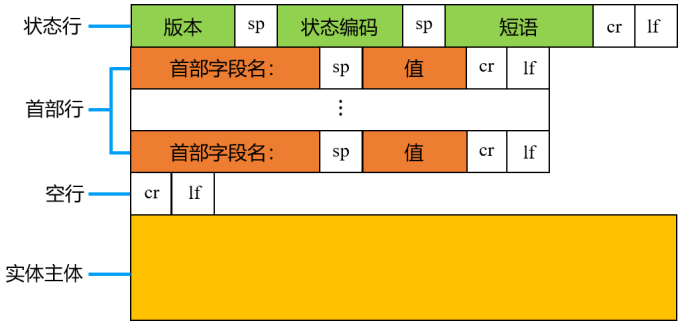
```
#典型的HTTP响应报文
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

    - 响应报文的第一行称为**初始状态行**，包括协议版本字段、状态码、状态信息

状态码	状态信息	说明

状态码	状态信息	说明
200	OK	请求成功
301	$Moved\ \ Permanently$	请求的对象被永久转移，新的URL定义在$Location$首部行
400	$Bad\ \ Request$	通用差错代码，请求不能被服务器理解
404	$Not\ \ Found$	被请求的对象不在服务器
500	$HTTP\ \ Version\ \ Not\ \ Supported$	服务器不支持请求报文中的$HTTP$协议版本

    - 其余行称为首部行
        - $Date$指示服务器发送响应报文的日期时间
        - $Server$指示服务器类型，类似于请求报文中的$User-agent$
        - $Last\ \ Modified$指示对象创建或最后修改的日期时间
        - $Content-Length$指示被发送对象的字节数
        - $Content-Type$指示实体体中对象为$HTML$文本
    - 实体体包含被请求的对象
    - $HTTP$响应报文的通用格式



2.2.3 $Cookie$

- $Cookie$的意义
    - 限制用户的访问
    - 将内容与用户身份相关联
    - 在无状态的$HTTP$上建立用户会话层
- $Cookie$的组成
    - $HTTP$响应报文中的$Cookie$首部行
    - $HTTP$请求报文中的$Cookie$首部行
    - 端系统中保留$Cookie$文件
    - Web服务器的$Cookie$数据库
- $Cookie$的使用
    - 客户向服务器发送普通请求报文
    - 服务器为客户创建ID如 `u202073245` 并放置在响应报文的首部行
    - 客户存储$Cookie$
    - 客户将$Cookie$放置在请求报文的首部行

- 服务器根据$Cookie$**采取指定动作**，并返回普通响应报文

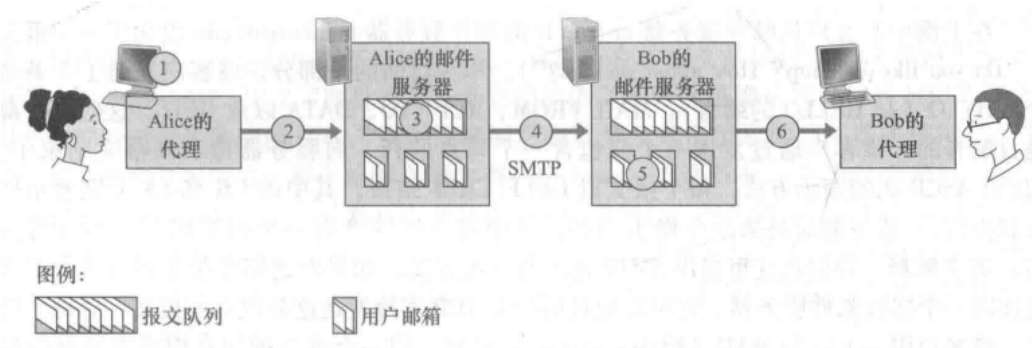
2.3 电子邮件

2.3.1 电子邮件系统的构成

- 用户代理
  - 用户可以撰写编辑邮件、查看邮件
  - 如网易邮箱大师、$FoxMail$
- 邮件服务器
  - 存储用户邮件的服务器
  - 在报文队列中维护需要发送的邮件报文
- 简单邮件传输协议$SMTP$
  - 将邮件从发送方的客户端发送到发送方的邮件服务器
  - 将邮件从发送方的邮件服务器发送到接收方的邮件服务器

2.3.2 $SMTP$协议

- 基本介绍（在线交互程序）
  - 使用TCP运输协议进行可靠的邮件传送，端口号为25
  - 不使用中间邮件服务器发送邮件，而是**直接在发送方服务器和接收方服务器间进行传输**
  - 使用持续连接
  - 要求报文（首部和信体）全部使用$7-bit\ ASCII$码
  - $SMTP$服务器用$CRLF.CRLF$表示邮件报文的结束



- 报文格式

```
From: alice@qq.com
To: Bob@google.com
Subject: Searching for the meaning of life.
```

- 与$HTTP$协议的对比

	$HTTP$	$SMTP$
协议类型	拉协议	推协议
报文编码格式	不限制	$7-bit\ ASCII$
多对象	每个对象分装在各自的响应报文中	多个对象在多分部的报文中

- 使用$telnet$指令访问QQ邮箱

```
telnet smtp.qq.com 25
auth login
base64-mail
base64-authentication-code
helo qq.com
mail from:<xxx@xx.xx>
rcpt to:<xxx@xx.xx>
data
Hello World!
Hello World!
.
quit
```

2.3.3 $POP3$协议

- 运行在端口110的邮件访问协议
- 运行方式
    - 特许阶段：用户代理发送用户名和口令以鉴别用户

      > 主要命令：$user$ $<$ $username$ $>$和$pass$ $<$ $password$ $>$
      >
      > 服务器的响应回答有$+OK$、$-ERR$

    - 事务处理阶段：用户代理取回报文，同时可以对报文做删除标记的更改，获取邮件统计信息

      > $list$——列出报文号码
      >
      > $retr$——用报文号码取回报文
      >
      > $dele$——用报文号码删除邮件

    - 更新阶段

      > $quit$——结束$POP3$会话，服务器删除被标记为删除的邮件

- 使用$telnet$指令访问QQ邮箱

```
telnet pop.qq.com 110
user QQ-ID
pass authentication-code
list
retr 1
dele 1
quit
```

2.3.4 $IMAP$协议

- 允许用户在服务器上组织自己的**邮件目录**
- 维护$IMAP$会话的用户信息：目录名以及报文ID与目录名之间的映射关系
- 使用$telnet$指令访问QQ邮箱

```
telnet imap.qq.com 143
a01 login QQ-ID authentication-code
a02 list "" *
a03 select inbox
a04 create folder
a05 delete folder
a06 rename oldfolder new folder
```

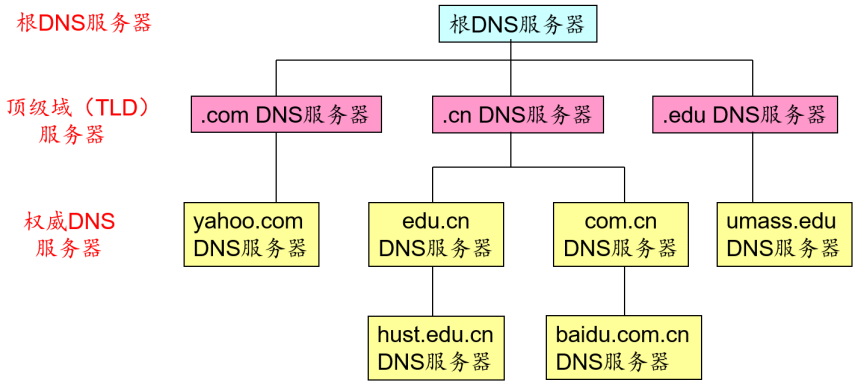
2.4 DNS

2.4.1 DNS服务

- DNS简况（在线交互程序）
  - DNS能够将主机名解析为主机的IP地址
  - DNS是一个分布式数据库，由很多DNS服务器按照层次结构组织
  - DNS运行在端到端系统上，且使用UDP协议（53号端口）进行报文传输
- DNS解析过程
  - 用户请求URL `http://cloudchewie.com/index.html`
  - 浏览器抽取出主机名 `cloudchewie.com` 并发送给DNS客户端
  - DNS客户端向DNS服务器发送查询请求报文
  - DNS服务器返回包含主机名对应IP地址的响应报文
  - DNS客户端将IP地址传送给浏览器
  - 浏览器向IP地址所在Web服务器发起TCP连接
- 其他服务
  - 主机别名：获取主机别名对应的主机规范名
  - 邮件服务器别名：获取邮件服务器主机别名对应的主机规范名
  - 负载分配：将一个IP地址**集合**与**同一个规范主机名**相联系

2.4.2 DNS工作机理

- 采用单台DNS服务器
  - 单点故障：一旦崩溃，因特网瘫痪
  - 通信容量：该台服务器不得不处理所有$HTTP$请求报文和电子邮件报文
  - 时延严重：集中式数据库将造成严重的拥塞与时延
  - 难以维护：不得不持续更新以适应数据更改
- 采用分布式层次化数据库



  - 根服务器提供TLD服务器的IP地址
  - TLD服务器提供权威服务器的IP地址，负责所有顶级域名和所有国家顶级域
  - 权威服务器提供域名到IP地址的映射服务

    - 本地DNS服务器（默认DNS服务器）
        - 当一台主机需要做一个域名查询的时候，查询请求首先被发送到本地域名服务器
- 递归查询与迭代查询（在线交互程序）
- DNS缓存（在线交互程序）
    - 一旦 (任何) 域名服务器得知了某个映射, 就将其缓存
    - 在一定的时间间隔后缓存的条目将会过期（自动消除）
    - TLD服务器的地址通常被缓存在本地DNS服务器中，以减少根服务器负载

2.4.3 DNS记录

格式为四元组$(Name, Value, Type, TTL)$，其中TTL表示记录的生存时间

$Type$	$Name$	$Value$
A	主机名	IP地址
$CNAME$	主机别名	规范主机名
NS	域	该域权威域名服务器的**主机名**
MX	邮件服务器的主机别名	邮件服务器的规范主机名

```
#使用nslookup进行DNS解析
#进入交互模式
nslookup
server www.net.cn
server dns.hust.edu.cn
set ty=A
cloudchewie.com
hust.edu.cn
set ty=ns
cloudchewie.com
set ty=cname
cloudchewie.com
set ty=mx
hust.edu.com
```

三、运输层

3.1 运输层服务

3.1.1 基本概念

- 运输层为不同主机上运行的应用进程提供逻辑通信信道$(logical\ \ communication)$
- 发送方把应用数据划分为**报文段**，交给网络层；接收方把报文段重组成应用数据，交付给应用层
- 网络层协议
    - 网际协议IP是**不可靠服务**，它将**尽力而为**地在不同通信主机间交付报文段
    - 但是它并不确保报文段的交付，不保证报文段按序交付，不保证报文段中数据完整性
- 运输层是**应用进程**之间的逻辑通信，网络层是**不同主机**之间的逻辑通信

- 运输层将**两个端系统间**IP的交付服务扩展为在端系统上的**两个进程之间**的交付服务

3.1.2 基本模型

- 假设家庭A的6个孩子要与家庭B的6个孩子相互写信
- 家庭A中的Amy负责收集所有信件并投递到邮局，并且负责将家庭B寄来的所有信件送到每个人手中
- 家庭B中的Bob负责收集所有信件并投递到邮局，并且负责将家庭A寄来的所有信件送到每个人手中
    - 进程=孩子们
    - 应用层报文=信封中的信笺
    - 主机=家庭
    - 运输层协议=Amy和Bob
    - 网络层协议=邮局提供的服务
- Amy和Bob只在各自家里进行收发工作
    - 运输层协议只工作在端系统中
- 假如Amy和Bob外出度假，分别替换成年龄较小的$Lisa$和$John$，他们可能会由于粗心大意丢失信件
    - 不同运输层协议提供的服务模型不一样，如UDP和TCP
- 邮局不承诺信件送达的时间
    - 运输层协议能够提供的服务**受到底层网络协议的服务模型的限制**
- 邮局不承诺信件一定安全可靠的送达，可能在路上丢失，但Amy和Bob可在较长时间内没有受到对方的回信时，再次誊写信件并寄出
    - 在网络层不提供某些服务的情况下，运输层自己提供

3.2 多路分解和多路复用

3.2.1 基本概念

- 多路分解：将运输层报文段中的数据交付到正确的套接字的工作
- 多路复用：在源主机从不同套接字收集数据块，并为每个数据块封装首部信息（用于多路分解）生成报文段，然后将报文段传送到网络层

3.2.2 要求

- 套接字有唯一标识符
- 每个报文段**有特殊字段指示该报文段要交付到的套接字**
    - 包括**源端口号字段**和**目的端口号字段**，各占$16\ bit$
    - 其中$0 \sim 1023$范围的端口号称为周知端口号，保留给诸如$HTTP$、$SMTP$等周知应用层协议

3.2.3 无连接的多路分解与多路复用

- 一个UDP套接字是**由一个<u>二元组</u>全面标识的**，具体为**（源端口号，目的端口号）**
- 如果两个报文段具有不同的IP地址或源端口号，但是具有相同的目的IP地址和目的端口号，那么这两个报文段将**通过相同的目的套接字被定向到相同的目的进程**
- 具体过程
    - 创建套接字：运输层自动为其分配一个端口号$(1024 \sim 65535)$，或者指定一个端口号

- 假设主机A中的一个进程具有UDP端口19157，欲发送一个报文给位于主机B中的另一进程，其具有UDP端口46428
- 主机A中的运输层封装报文，添加源端口号19157，目的端口号46428以及其他两个字段得到报文段
- 运输层将报文段传递到网络层，网络层封装IP数据包并尽力而为地传送到接收主机
- 如果报文段到达主机B，主机B中的运输层即检查目的端口号46428，并将报文段交付给端口号为46428的套接字
- 当主机B需要回发报文段给A时，即需要**将来自A的报文段中的源端口号作为目的端口号**发送报文段

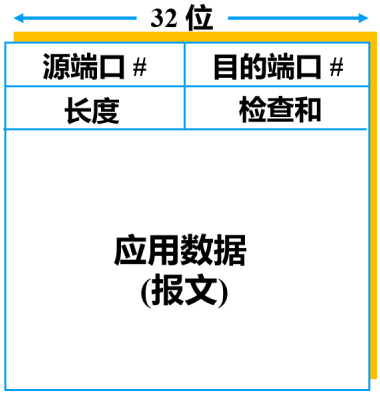
3.2.4 面向连接的多路分解与多路复用

- 一个TCP是**由一个四元组全面标识的**，具体为（**源IP地址，源端口号，目的IP地址，目的端口号**）
- 与UDP协议不同，如果两个报文段具有不同的IP地址或源端口号，而且具有相同的目的IP地址和目的端口号，那么这两个报文段将**通过不同的目的套接字被定向到不同的目的进程**
- 具体过程
    - 假设主机A中的一个进程具有TCP端口19157，欲发送一个报文给位于主机B中的另一进程，其具有TCP端口46428
    - 同时主机C中的一个进程也具有TCP端口19157，也要发送报文给主机B中具有TCP端口46428的进程
    - 主机B**依然能够正确分解具有两个相同源端口号和目的端口号的连接**，因为二者的IP地址不同，决定着其套接字不同
- Web服务器与TCP
    - 在Web服务器中，当客户发送请求时，所有报文段的目的端口号都将为80，这时服务器根据源IP地址和源端口号来区分来自不同客户的报文段
    - 在当今的高性能服务器中，通常只使用一个进程，而是通过为不同的套接字创建**新的线程（轻量级子进程）**提供服务

    > 非持久$HTTP$对每一个请求都建立不同的套接字，会影响性能

3.3 UDP协议

3.3.1 UDP报文段格式



- **长度字段**指示了包括首部在内的报文段中**字节**数
- **校验和**用于接收方校验报文段是否发生差错

3.3.2 UDP校验和

- 发送方
    - 把报文段看作是16**比特字的序列**
    - 对报文段的所有16比特字的和进行**反码**运算，加法有溢出时，需要将进位加到末尾
    - 将计算校验和的结果写入UDP校验和字段中
- 接收方
    - 将**包括校验和在内**的所有16比特字加在一起
    - 如果没有差错，结果将为$1111,1111,1111,1111$
- 无法纠正错误

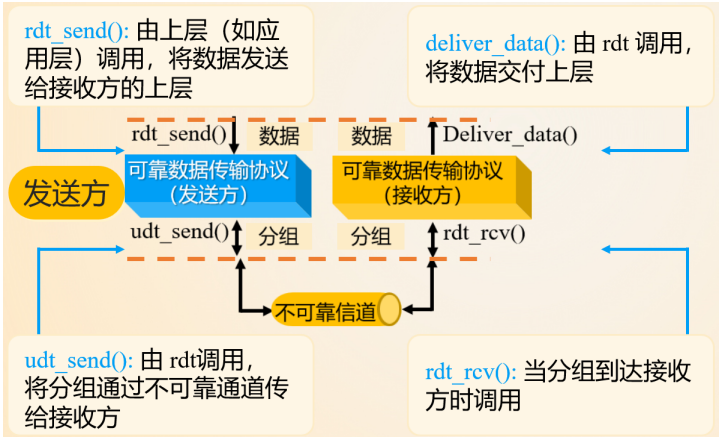
3.3.2 UDP与TCP对比

UDP	TCP	**说明**
无需建立连接	需要建立连接（握手）	建立连接会增加时延
无需维护连接状态	需要维护连接状态	
首部8 $Byte$	首部20 $Byte$	UDP段首部开销较小
无拥塞控制，可按需随时发送	有拥塞控制	大量使用UDP会导致路由器中堆积分组，挤占TCP会话
适用于DNS服务、因特网电话	适用于Web服务、电子邮件	

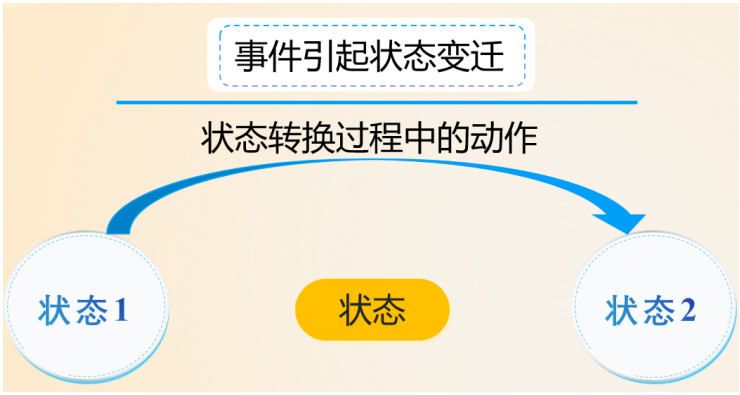
3.4 可靠数据传输

3.4.1 可靠数据传输概述

- 可靠数据传输：数据**不会丢失**且数据**不会出错**
- **网络层的IP协议是不可靠信道**，因此为了实现可靠数据传输，**需要在<u>运输层</u>做出保证可靠数据传输的工作**
- 可靠数据传输协议模型



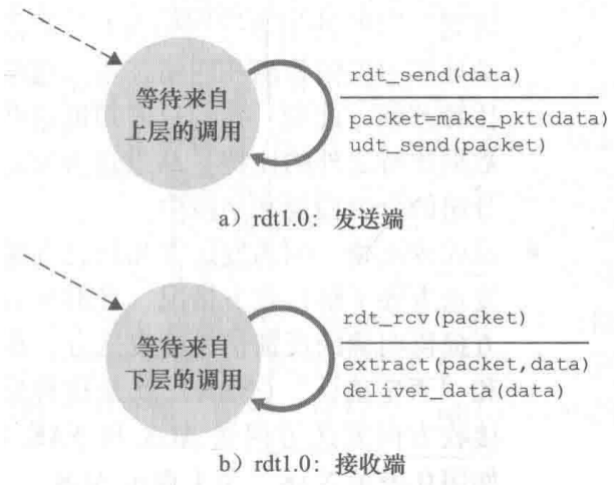
- 使用有限状态机FSM描述发送方和接收方



- 在模型中，存在多个状态
- 事件引起状态的变化
- 状态变化过程中又会存在一系列动作的发生，以完成状态的转换

3.4.2 可靠信道上的可靠传输

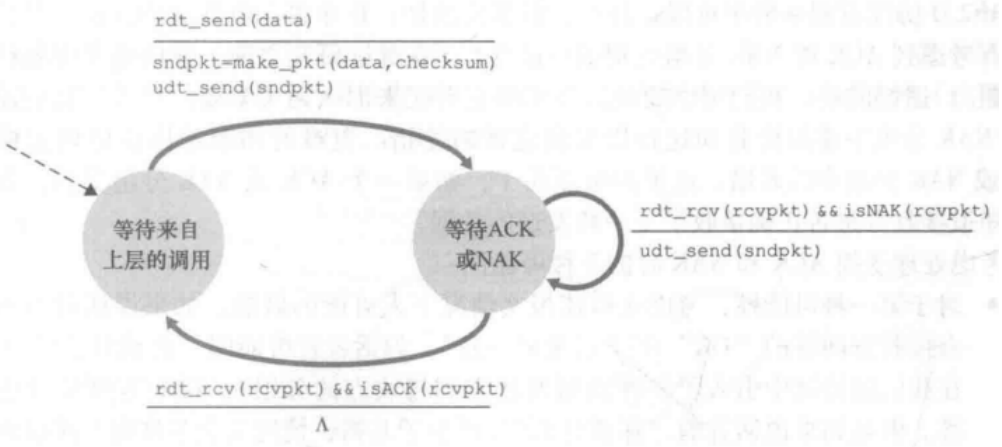
- 信道模型：不会发生比特传输错误，不会造成分组丢失
- 有限状态机FSM



  - 发送方——发送分组：发送报文段，继续等待上层调用发送分组
- 接收方——接收分组：分解得到数据

3.4.3 简单停等协议

- 信道模型：分组比特可能受损，但分组将按序接收，不会丢失
- 停止等待协议：每次发送分组后，发送方需要等待接收方的反馈
- 设想方案
  - 第一步：判断分组受损——**差错检测**
  - 第二步：通知发送方分组是否受损——**接收方反馈**$(ACK与NAK)$
  - 第三步：得知分组受损(NAK)后，发送方的处理手段——**出错重传**
- 有限状态机FSM



a）rdt2.0: 发送端

b）rdt2.0: 接收端

    - 发送方
        - 发送分组：生成校验码$checksum$，用于进行差错检测，发送报文段，跳转到等待状态
        - 反馈ACK：清除缓冲区中的报文段，跳转到等待上层调用发送分组的状态
        - 反馈NAK：重新发送报文段（保存在缓冲区中），继续等待反馈
    - 接收方
        - 接收到的分组出错$(corrupt)$：生成NAK报文段，发送给发送方
        - 接收到的分组正确$(notcorrupt)$：分解得到数据，生成ACK报文段，发送给发送方
- 新的问题：ACK和NAK分组也可能受损
    - 受损的分组视为ACK不合适
    - 受损的分组视为NAK可能导致接收方重复收到分组——**对分组进行编号，便于接收方识别是新分组还是旧分组**
- 改进后的有限状态机FSM

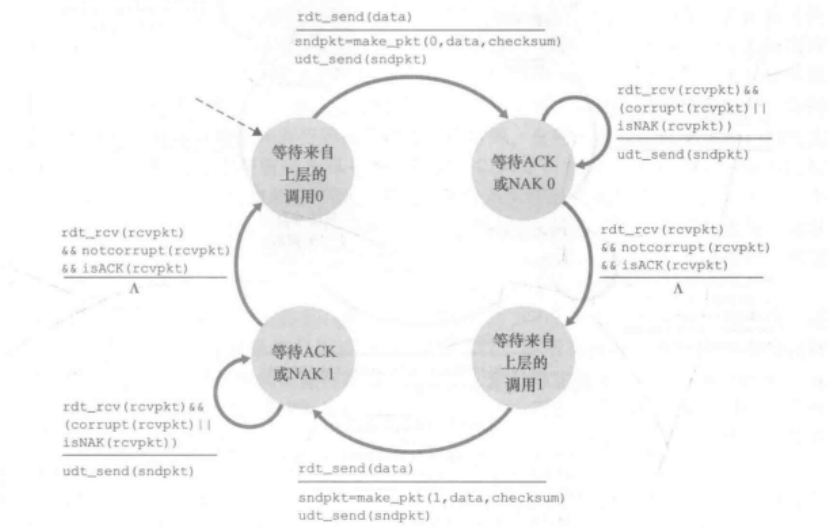


图 3-11　rdt2. 1 发送方

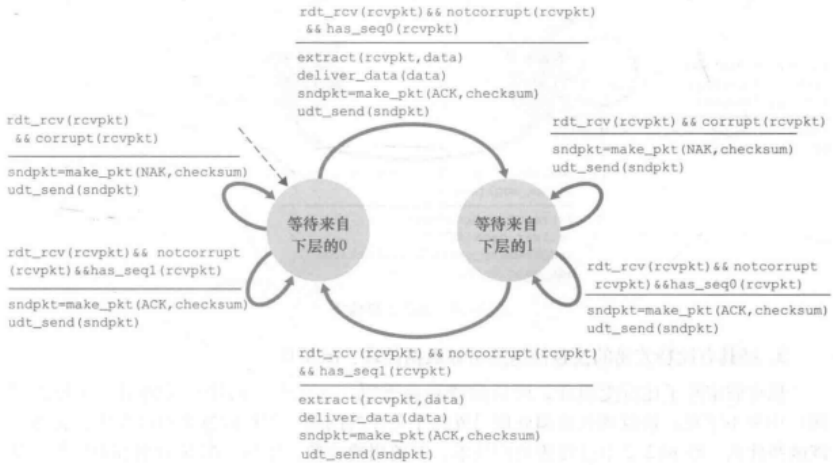


图 3-12　rdt2. 1 接收方

- 取消NAK
  - 取消NAK，接收方对最后一个正确收到的分组发送ACK
  - 同时，ACK中必须指出被确认分组的序号，以便于接收方标识正确收到了哪一个分组
  - $Double\ ACK = NAK$
- 取消NAK后的有限状态机FSM

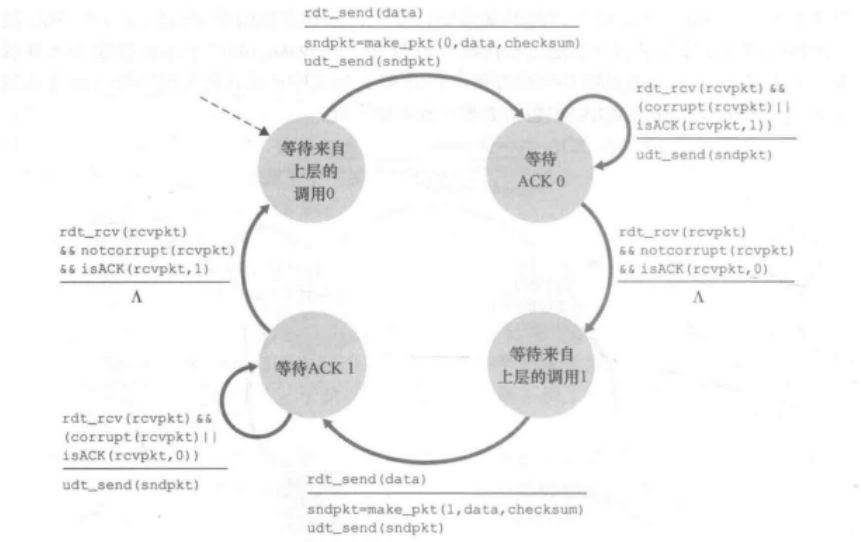


图 3-13　rdt2. 2 发送方

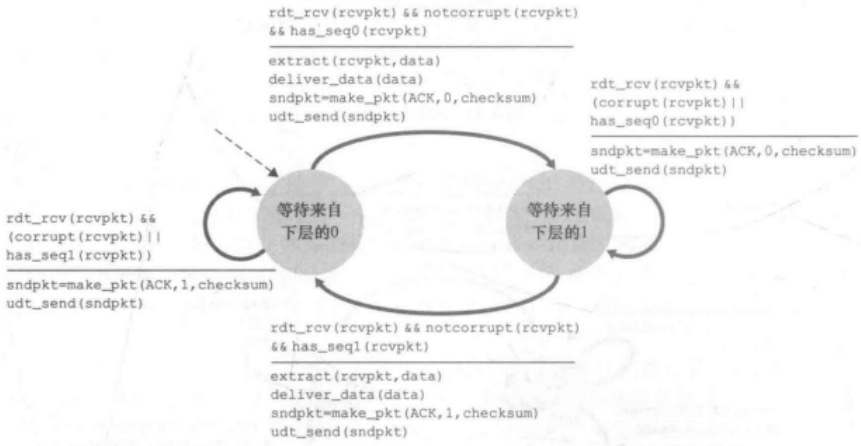


图 3-14　rdt2. 2 接收方

3.4.4 实用停等协议

- 信道模型：分组比特可能受损，还可能会丢包
- 如何检测丢包：耐心的等待——**超时重传**
- 有限状态机FSM

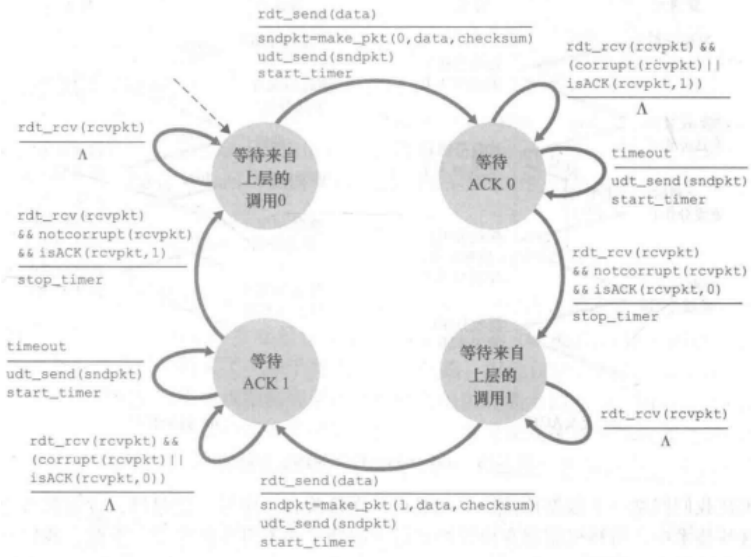


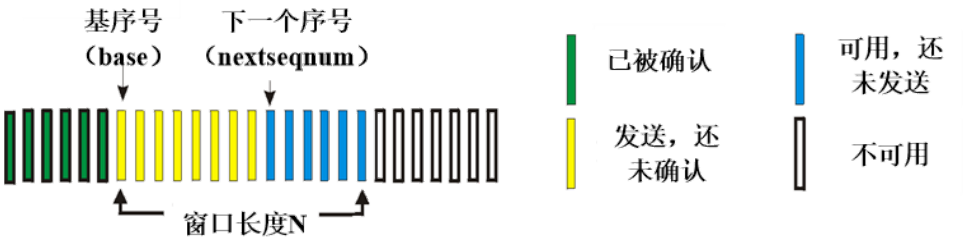
图 3-15　rdt3. 0 发送方

- 性能低下

- **信道利用率：传输时间/总时间（从发送方开始传输到发送方接收到反馈所用的时间）**
- 假设某条$1Gbps$的链路，存在$15ms$的端到端时延，则传送$1KB$大小的分组所用的时间为$T_{trans}+RTT=\frac{1KB}{1Gbps}+15ms\times 2=30.008ms$，也即链路速度被限制为$33KB/sec$的吞吐量
- 网络协议大大限制了物理资源的利用率，实际上用于传输分组的时间只占$\frac{0.008}{30.008}=0.00027$

3.4.5 滑动窗口协议

- 解决方案：允许发送方发送多个分组后再等待确认
  - 必须增大序号范围
  - 发送方和接收方需要对分组进行缓存
- 流水线技术工作原理
  - 用k位进行序号编码
  - 扩大发送方乃至接收方的缓冲区大小——**窗口**



  - 当丢失一个分组后，如何进行重传

	$GBN(Go\ Back\ N)$**协议**	**选择重传**(SR)**协议**
确认方式	累计确认	单个确认
定时器	对所有已发送但未确认的分组统一设置定时器	对所有已发送但未确认的分组分别设置定时器
超时(n)	重传分组n和窗口中所有序号大于n的分组	仅重传分组n
失序分组	丢弃（不缓存），接收方缓冲区大小为1个分组	缓存，接收方缓冲区大小与发送方一致
失序分组处理	重发按序到达的最高序号分组的ACK	对失序分组进行选择性确认
要求	$W_s\leq 2^k-1, W_r=1$	$W_s, W_r\leq 2^{k-1}$

- 分组序号长度与窗口大小的关系
  - 假设分组序号为k位，发送窗口大小为W_s，接收窗口大小为W_r，发送窗口当前序号为$i,\cdots,i+W_{s-1}$
  - 极端情况下，发送的所有分组均正常接收，但是ACK均丢失，则
    - 发送窗口当前序号为$i,\cdots,i+W_{s-1}$
    - 接收窗口当前序号为$i+W_s,\cdots,(i+W_s+W_{r-1})\ mod\ 2^k$
    - 两个窗口序号不重复，且均在$[0,2^k-1]$内

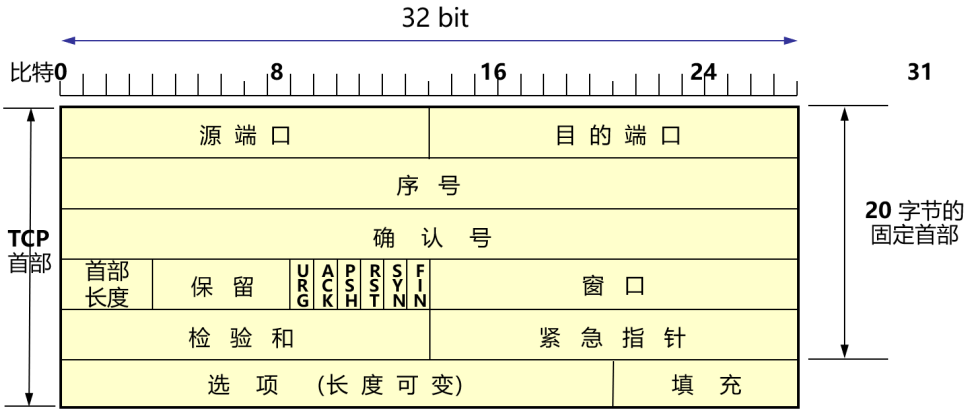
- 要求$W_s + W_r \leq 2^k$

3.5 TCP协议

3.5.1 TCP工作流程

- 建立连接——三次握手
  - 客户发送一个特殊的TCP报文段
  - 服务器也发送一个特殊的TCP报文段作为响应
  - 客户再发送一个特殊的报文段作为响应，可以承载有效负荷
- 发送数据
  - 客户进程从套接字传递出数据流，经过该门后即进入TCP控制
  - TCP将数据流引导至该连接的**发送缓存**中
  - TCP每次取出一块数据放入报文段中进行运输，每次取出的数据数量受限于**最大报文段长度**MSS，而该值受限于**最大链路层帧长度**MTU（能从源到目的地的所有链路上发送的最大链路层帧）
  - MSS指报文段中应用层数据的最大长度，而不包含TCP首部
- TCP是**全双工**的

3.5.2 TCP报文



- 序号：**报文段的首个字节在整个字节流中的的序号**，是离散型编号

> 假设数据流包含500000字节，MSS为1000字节，那么将构建500个报文段，其序号依次为0、1000、2000等等

- 确认号（期待号）：期待得到的下一个字节的序号，当ACK字段为1时，确认号才有效

> 如确认号为$n+1$时表示接收方接收到第n号，期望得到$n+1$号分组，这称为**累计确认**

- 首部长度：TCP最长60个字节，以4个字节为单位进行递进，即与图示行数相同，取值为$[5,15]$
- 窗口：用于TCP流量控制
- 检验和：差错检测
- 紧急指针：当URG字段为1时，表示需要传送紧急数据；紧急指针指示了紧急数据的结尾

> 当PSH字段为1时，整个数据部分都为紧急数据

3.5.3 TCP机制

- 快速重传
    - 原因：超时周期往往太长，增加重发丢失分组的延时
    - 通过重复的ACK检测丢失报文段：如果发送收到一个数据的3个**冗余**ACK，即确认数据之后的报文段丢失，以在超时到来之前重传报文段
    - 每次TCP重传都会将下一次超时间隔设置为先前值的两倍
- 流量控制
    - 可变滑动窗口：接收方将缓冲区的空闲空间大小写入报文段返回给发送方，发送方根据空闲空间大小调整发送窗口大小
    - 当发送方接收到空闲空间大小为0时，为避免进入假死状态（发送方不发送报文段，接收方不发送反馈），发送方会持续发送小的报文段试探接收方
    - 当接收方重新获得空间空间后，小报文段将会被接收方响应处理并反馈空闲空间大小给发送方
- 拥塞控制
    - 拥塞：包括**丢包**(路由器缓冲区溢出）和**时延长**(在路由器缓冲区排队）
        - 当路由器缓存无限大且不会重传时，随着发送方速率的增大，接收方速率也会增大；但当接收方速率饱和时，吞吐量也随之固定。而此时，也就意味着在路由器缓冲区排队的分组越来越多，造成越来越大的排队时延
        - 当路由器缓存有限且会对丢失的分组重传时，对延迟到达（而非丢失）的分组的重传使得发送方速率比理想情况下更大于接收方速率；发送方在遇到大时延时所进行的不必要重传会引起路由器转发不必要的分组拷贝而占用其链路带宽
        - 当路由器缓存有限且会进行超时重传时，由于超时重传，当分组被丢弃时，该分组曾用到的所有**上游**传输容量被浪费
    - 端到端拥塞控制
        - 每个发送方自动感知网络拥塞的程度，发送方根据感知的结果限制外发的流量
        - 如何限制外发流量：控制拥塞窗口长度
        - 如何感知拥塞程度：超时或者3个冗余ACK
        - 如何调节发送速率：加性增，乘性减（出现丢包事件后将当前$CongWin$大小减半，可以大大减少注入到网络中的分组数；当没有丢包事件发生，每个RTT之后将$CongWin$增大1个MSS，使拥塞窗口缓慢增大，防止网络过早拥塞）
    - $Reno$算法
        - 慢启动：$CongWin = 1MSS$，小于阈值$ssthresh$时指数增加
        - 拥塞避免：达到$ssthresh$后，加性增
        - 收到3个冗余ACK：$ssthresh = CongWin/2$，$CongWin = ssthresh + 3MSS$，加性增
        - 超时：$ssthresh = CongWin/2$，$CongWin = 1MSS$，指数增加到阈值后加性增

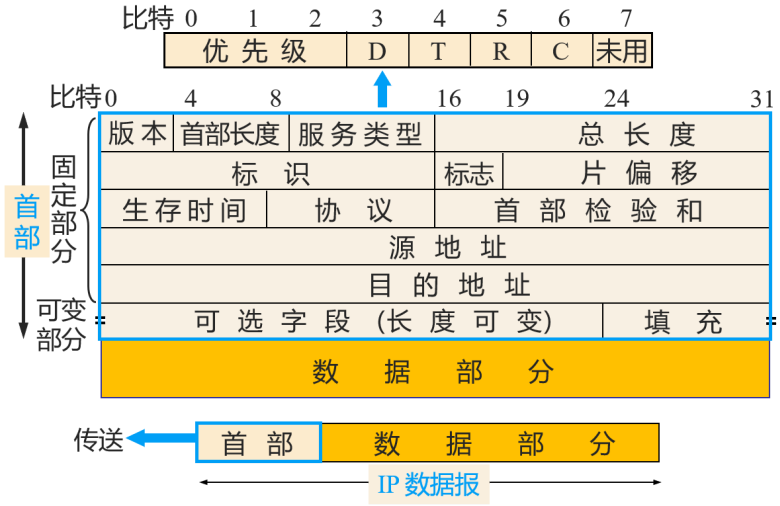
四、网络层

4.1 网络层概述

- 网络层：实现**主机和主机**之间的通信
    - 数据平面：如何转发
        - **转发**是指将分组从一个输入链路接口转移到适当的输出链路接口的路由器本地动作
        - 转发的时间尺度很短(ns)，因此采用硬件实现
        - 假设某个驾驶员从武昌到汉口经过许多立交桥，**转发类比于经过某个立交桥的过程**
        - 分组交换机上的网络层根据转发表以及分组首部信息，将分组向适当链路进行转发
    - 控制平面：如何选路
        - **路由选择**是指确定分组从源到目的地所采取的端到端路径的网络范围处理过程
        - 路由选择的时间尺度较长(s)，因此采用软件实现
        - 假设某个驾驶员要从武昌到汉口，**路由选择类比于规划从武昌到汉口的行程**
        - 在全局范围为主机之间的通信进行选路，选路结果反映为分组交换机上的转发表

4.2 IP协议

4.2.1 $IPv4$数据报格式



- 版本：$4\,bit$，规定数据报的IP协议版本
- 首部长度：由于首部中含有可选字段，因此需要使用首部长度确定载荷实际开始的地方
- 服务类型：$8\,bit$，代表报文处理方式，每一位分别代表最小延时、最大吞吐量、最高可靠性、最小成本，**只选一个**
- 数据报长度：**首部加上载荷的总长度**，以字节计，理论上数据报最大长度为65535字节，但以太网链路报文只允许1500字节
- 标识、标志、片偏移：与IP分片有关，但$IPv6$不允许在路由器上分组分片
    - 链路层帧大小有限，超过时应该进行分片处理
    - 将一个大数据报拆分为几个小数据报，传输后在目的主机进行重组
    - 标识：$16\,bit$，网络层服务的上层传输层的同一次报文（可能超过1500字节），使用相同标记

    - 标志：$3bit$，1位保留，2位表示是否能分片，3位（MF）表示分片是否结束（1为未结束，0为结束）
    - 片偏移：每个分片在整个报文（分组）中的位置（**以8字节为度量单位**）
- 生存时间：确保数据报不会永远在网络中循环，每当一台路由器处理数据包时，该字段值减1，减为0时必须丢弃该数据报
- 协议：当数据报到达最终目的地时才有用，指示了载荷要交付给哪个特定的运输层协议，如6标识TCP，17标识UDP

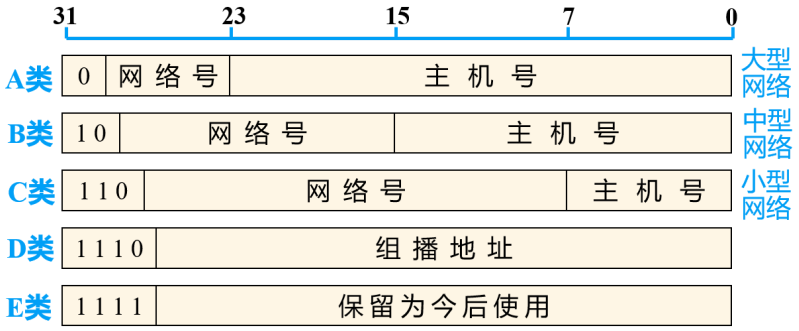
> 数据报中的协议号将网络层与运输层绑定在一起，报文段中的端口号将运输层和应用层绑定在一起

- 首部校验和：按**首部**的每两个字节进行校验和计算，每台路由器都要重新计算首部校验和并放置在首部校验和位中
- 源和目的地址
- 如果不包含可选字段，IP数据段的首部将包含20个字节

> 首部长度不定（$20-60$字节），中间节点（路由器）需要消耗相当资源用于分组处理

4.2.2 IP地址

- IP地址的定义
    - 一台主机通常只有一条链路连接到网络；当主机中的IP想发送数据报时，通过该链路发送
    - 主机与物理链路之间的边界叫做**接口**
    - 路由器负责从链路上接收数据报并从其他链路转发出去，其与任意一条与其连接的链路之间的边界也叫做**接口**
    - 每条主机和路由器都能发送和接收数据报，因此IP协议要求每台主机和路由器接口都有自己的IP地址
    - **一个IP地址和一个接口相关联，而不是与包含该接口的主机或路由器相关联**
- $IPv4$编址
    - 每个IP地址长度为32比特，因此共有2^{32}个可能的IP地址
    - 通常按照点分十进制记法书写，即地址中的每个字节用十进制表示，各字节间以句点隔开，如$193.32.216.9$
    - 地址分类

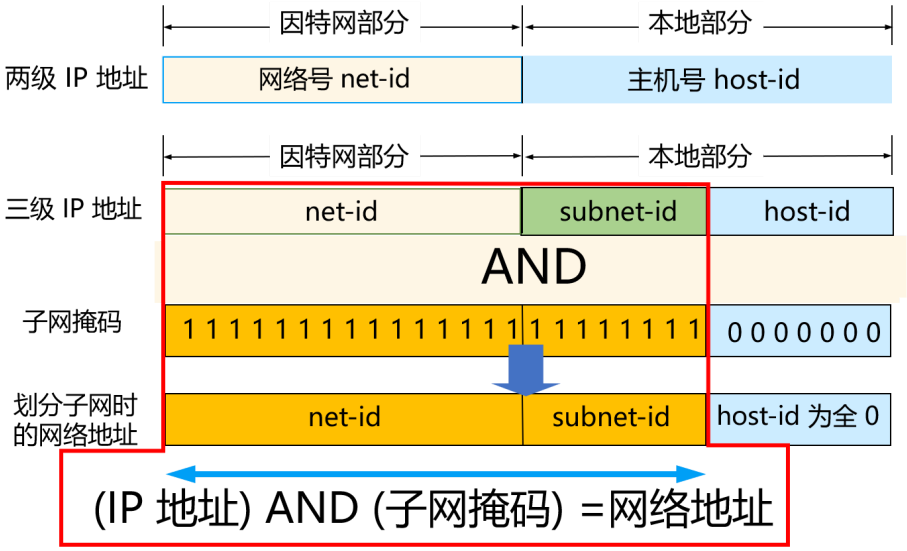


- 大约有40亿$IPv4$地址，需要分类以便于寻址和层次化构造网络
- 在同一个局域网上的主机或路由器的IP地址中的网络号必须一样
- 路由器具有两个及以上的IP地址，其每一个接口都有一个不同网络号的IP地址
- A类地址以$0-127$开头，B类地址以$128-191$开头，C类地址以$192-224$开头
- 特殊IP地址

- 全0主机号的IP地址表示网络本身，如$129.152.0.0$是网络号为129.152的B类网络
- 全1主机号的IP地址表示广播地址，如$129.152.255.255$是网络号为129.152的B类网络的广播地址
- 十进制127开头的地址是回环地址，用于测试自身TCP或IP软件是否正常

4.2.3 子网划分

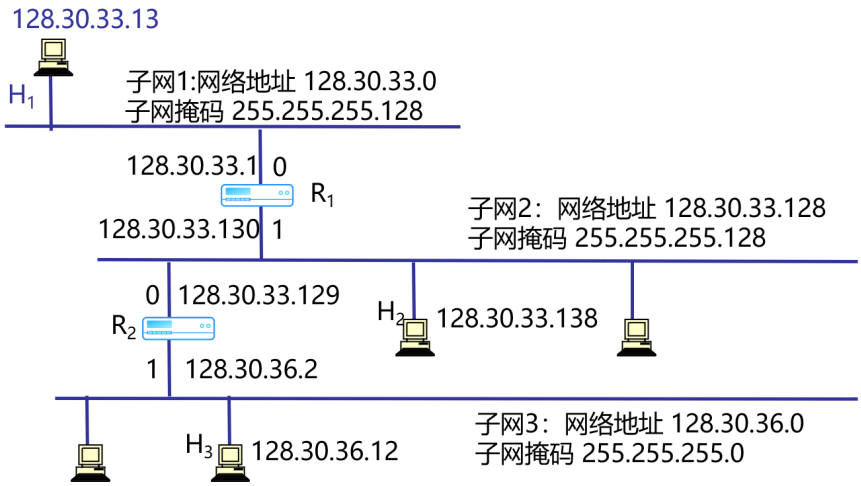
- 子网的定义
  - 划分子网以方便地址划分、分块管理，需要**从主机号中借用一部分比特作为子网号**
  - **网络号**唯一指定主机所在网络，该网络包含若干子网
  - **子网号**唯一指定主机所在子网
  - **主机号**唯一指定子网内某台主机
- 子网掩码



  - 将网络号和子网号相应的位置全置1，主机号相应的位置全置0，得到子网掩码
  - 将IP地址和子网掩码相与得到网络地址
  - A、B、C类地址具有默认子网掩码

地址分类	默认掩码	网络号比特	主机号比特	子网可容纳主机数
A	255.0.0.0	8	24	2^24-2
B	255.255.0.0	16	16	2^16-2
C	255.255.255.0	24	8	2^8-2

- 子网寻址：先检查目的IP地址的网络号，然后检查目的IP地址的子网号



从主机H_1发送分组到主机H_2

- 当主机1访问主机2时，先检查主机2是否在本网络上：将目的主机地址$128.30.33.138$与本网络子网掩码$255.255.255.128$相与得到$128.30.33.128$不等于子网1的网络地址$128.30.33.0$，说明主机2不在主机1的子网中
- 依次查询路由器R_1的路由转发表

R_1 的路由表（未给出默认路由器）

目的网络地址	子网掩码	下一跳
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	R_2

  - 对于第一条记录，目的主机地址与子网掩码相与得到$128.30.33.128$不等于$128.30.33.0$
  - 对于第二条记录，目的主机地址与子网掩码相与得到$128.30.33.128$等于该记录的目的网络地址，因此要访问目的主机时需要访问目的网络地址，也即经过路由器R_1的接口1进行转发

- 子网划分

1. 确定子网个数
2. 确定每个子网内的最大主机数
3. 确定从主机号字段中借用的比特数，用于创建子网号字段
4. 确定主机号字段中保留的比特数
5. 确定原始网络号字段和主机号字段的比特数
6. 检查主机号字段长度大于第3步和第4步中确定的比特数
7. 设置子网号字段的最佳长度
8. 创建子网掩码
9. 确定有效的子网号
10. 确定每个子网的IP地址范围

4.2.4 无类域间路由$CIDR$

- 编码格式
  - IP地址表示为{网络前缀，主机号}
  - 斜线记法：$192.168.0.1/25$表示左边25位为网络前缀，对应子网掩码表示中的$192.168.0.1/255.255.255.128$
  - 简写记法：$10.0.0.0/10$简写为$10/10$
- 示例

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

    - 假设某个地址块为$XX.XX.XX.XX/n$，即该地址块的左边n位为网络前缀，则剩余$32-n$位用来具体表示主机号，其地址数为2^{32-n}
    - 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，就需要有64个项目。但采用地址聚合后，只需用路由聚合后的1个项目$206.0.64.0/18$就能找到该ISP。
- 寻址方式
    - 使用$CIDR$编址后，路由转发表变为如下格式

网络前缀	下一跳
128.30.33.0/25	接口 **0**
128.30.33.128/25	接口 **1**
128.30.36.0/24	**R_2**

    - 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，此时从匹配结果中选择具有最长网络前缀的路由
        - 网络前缀越长，其地址块就越小，因而路由就越具体

        > 如对于实例中，假设目的地址为$206.0.71.142$，其匹配大学的地址块前缀$206.0.68.0/22$，也匹配四系的地址块前缀$206.0.71.128/25$，根据最长前缀匹配规则，需要匹配四系的地址块

    - 示例（B、A、E、F、C、D）

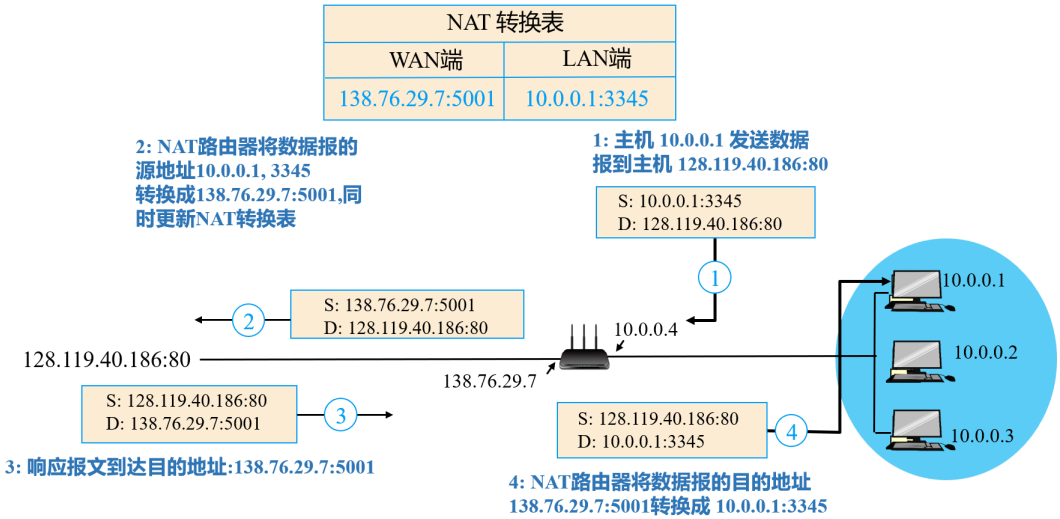
右表为一个使用CIDR的路由表，请说明下列地址的下一跳各是什么？

(a) C4.5E.13.87
(b) C4.5E.22.09
(c) C3.41.80.02
(d) 5E.43.91.12
(e) C4.6D.31.2E
(f) C4.6B.31.2E

网络/前缀长度	下一跳点
C4.50.0.0/12	A
C4.5E.10.0/20	B
C4.60.0.0/12	C
C4.68.0.0/14	D
80.0.0.0/1	E
40.0.0.0/2	F
0.0.0.0/2	G

4.2.5 网络地址转换NAT

- 实现原理



- 发送数据报：将每个向外发送报文的源IP地址与端口号**映射**为$NAT\ IP$地址以及新端口号；
- 同时远程客户机或者服务器将以$NAT\ IP$地址以及新端口号做为目的地址进行响应
- NAT路由器将每一个地址转换对记录在NAT转换表中

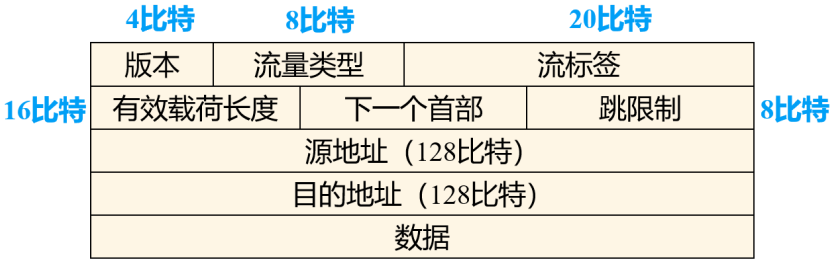
  > 转换表的表项为：(源IP地址,端口号)→(NAT的IP地址,新端口号)

- 接收数据报：根据NAT转换表将每个向内发送报文的$NAT\ IP$地址和端口号替换为相应的源IP地址以及端口号
- 内网主机对外网不可见；ISP变更后内网地址无需变化

- 使用NAT的变化

不使用NAT	**使用NAT**
主机=IP	主机=IP+端口
统一全球地址	全球地址+本地地址
主机访问双向公平	外网主机无法主动访问内网主机

4.2.6 $IPv6$数据报格式



- 由于$IPv4$地址池即将用尽，因此为了适应对大IP地址空间的需求，开发了新的IP协议——$IPv6$协议
- IP地址长度被扩大为128比特，首部长度为固定为40字节
- 版本：标识IP版本号，该字段值为6时即表示$IPv6$协议

  > 注意：该字段值置为4并不能创建合法的$IPv4$数据报

- 流量类型：同$IPv4$中的TOS字段

- 流标签：用于标识一条数据报的流
- 有效载荷长度：无符号整数，指示数据的字节数
- 下一个首部：标识需要交给哪个运输层协议，如UDP或TCP
- 跳限制：同$IPv4$中的生存时间
- 不存在分片与重组：中间结点不再负责分片和重组，由端结点负责

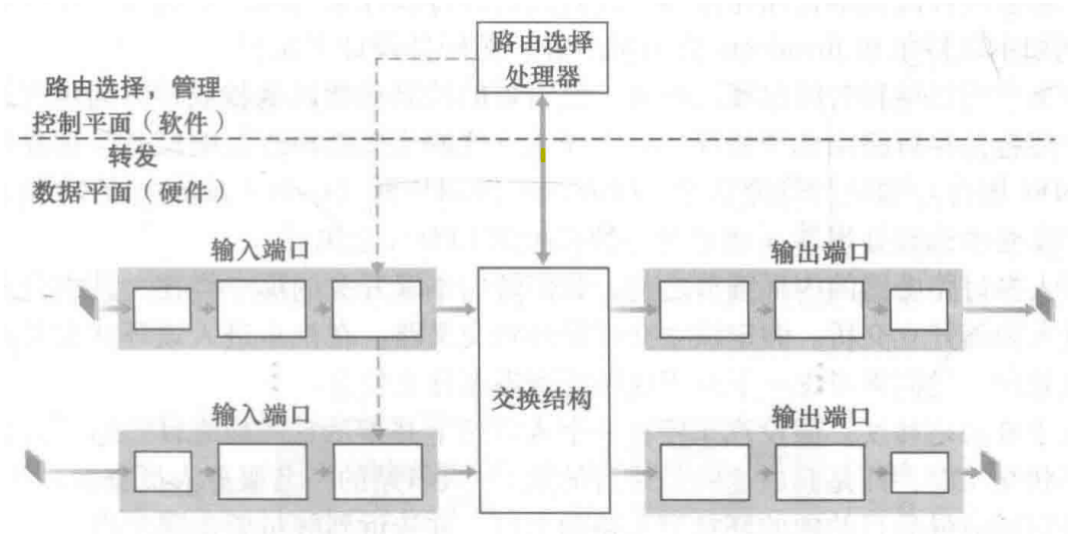
> 即不允许在中间路由器上进行分片与重组，只能在源和目的地执行；当路由器接收到的$IPv6$数据报太大时，会丢弃数据报并发回分组太大的$ICMP$差错报文

- 不存在首部校验和：中间节点无需计算
- 不存在选项字段：首部长度固定，加速中间节点转发速度
- 从$IPv4$到$IPv6$
    - 双栈技术
        - 新加入的设备支持$IPv4/IPv6$双协议栈
        - 一段链路上，如果源和目标均支持$IPv6$，则使用$IPv6$进行通信
        - 如果任一方不支持$IPv6$，则使用$IPv4$进行通信
        - 转换开销较大，可能会出现信息的丢失
    - 隧道技术
        - 将$IPv6$的数据报封装在$IPv4$的数据报中，即建立隧道传输$IPv6$数据报

4.3 路由器的工作原理

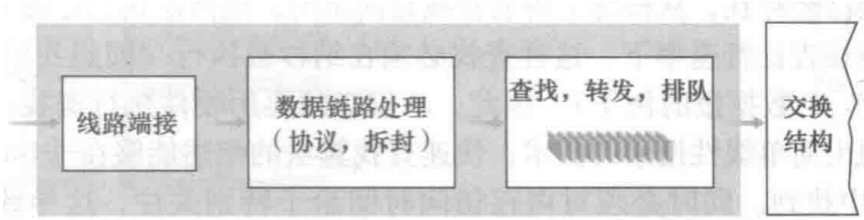
4.3.1 路由器结构

- 路由器基本结构



    - 输入端口：执行物理层的线路连接功能；执行数据链路层功能；执行网络层查找功能决定输出端口
    - 交换结构：根据输入端口查找得到的输出端口，将数据转发到对应的输出端口
    - 输出端口：存储从交换结构接收的分组，并执行对应的链路层和物理层功能
    - 路由选择处理器：执行**控制平面**功能
- 转发方式：如何选择输出端口
    - 基于目的地转发：根据输入分组的最终目的地转发，类比于在立交桥中根据目的地决定出口
    - 通用转发：除了目的地，还根据其他因素进行转发

4.3.2 输入端口

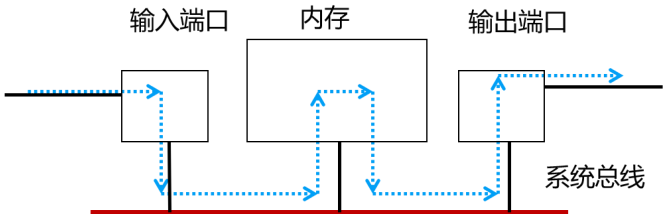


- 基于目的地转发
    - 对于32位的IP地址，在转发表中可以维护每个目的地址的表项，但需要维护的表项数十分庞大
    - 实际上，可以通过将目的地址进行分组管理，通过**前缀匹配**的方式进行转发
    - 当有多个匹配项时，采用**最长前缀匹配**规则：寻找表中最长的匹配项

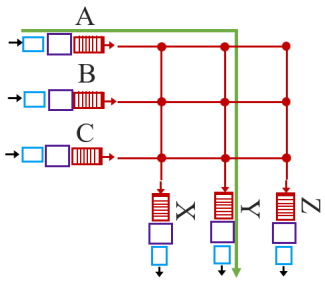
> 例如有如下仅包含4个表项的转发表
>
> | 前缀匹配 | 输出接口 |
> | --- | --- |
> | $1100,1000,0001,0111,0001,0$ | 0 |
> | $1100,1000,0001,0111,0001,1000$ | 1 |
> | $1100,1000,0001,0111,0001,1$ | 2 |
> | 其他 | 3 |
>
> 对于目的地址是$1100,1000,0001,0111,0001,0110,1010,0001$的分组，其匹配转发表中的第一项，因此将被转发到输出接口0；对于目的地址是$1100,1000,0001,0111,0001,1000,1010,0001$的分组，其匹配转发表中的第二项和第三项，但根据最长前缀匹配规则，将被转发到输出接口1

4.3.3 交换结构

- 内存交换



    - 输入和输出端口间的交换是在**路由处理器**的直接控制下完成
    - 分组被拷贝到系统内存中，在CPU的控制下转发至输出端口
    - 转发速度受限于内存带宽（每个分组走两次总线）
- 总线交换
    - 输入报文经共享总线将分组直接转发到输出端口
    - 总线交换速度受限于总线带宽
- 内联网络交换



内联网络

- 克服总线带宽限制

4.3.4 输出端口

- 排队现象的产生
  - 设输入线路和输出线路的传输速率相同，均为$R_{line}\ pkt/s$，有N个输入输出端口，交换结构传送速率为R_{switch}
  - 当$R_{switch} \gg R_{line}$时，可以使得输入的分组**无时延**地通过交换结构
- 输入排队
  - 当R_{switch}不满足$R_{switch} \gg R_{line}$时，将使得输入端口出现分组排队
  - 线头阻塞：输入队列中排队分组被位于线头的另一个分组阻塞，须等待交换结构发送
    - 当两个不同输入端口的分组均要发往同一个输出端口时，其中一个分组必须等待交换结构转发完毕另一个分组
- 输出排队
  - 当R_{switch}超过R_{line}时，需要对分组进行缓存
  - 输出端口缓冲区溢出会导致分组的排队和丢失
  - 缓冲区大小：对于有N条TCP连接经过的链路而言，缓存数量为$B = RTT\frac{R}{\sqrt{N}}$

> 其中R为链路容量，RTT为平均往返延迟

※4.4.7 $ICMP$协议

- 用途
- 分类
- 报文格式
- 应用
  - $Ping$命令
  - $Tracert$命令

4.5 路由协议

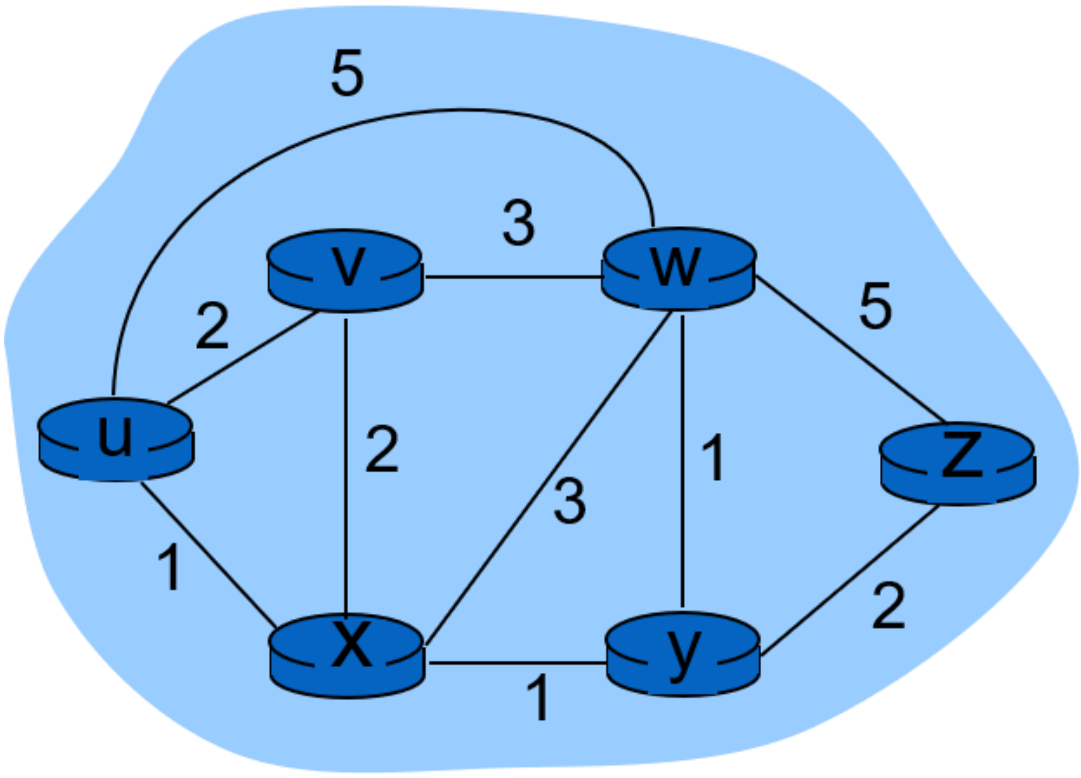
4.5.1 概述

- 什么是路由
  - 路由是从源主机到目的主机的路径
  - 路由是在路由器上执行的过程，包括接收路由协议、对路由进行选路
  - 路由是在两个路由器间传递消息的路由协议
  - 路由是在路由协议的处理下得到的路由表
- 什么是好的路由

    - 好的路径：无环路，可收敛，费用低
    - 好的路由协议：开销低（内存、带宽占用），安全性高
    - 好的路由表：保存局部路由，共同构成完成路由
- 什么是路由器
    - 默认路由器：一台主机连接到的路由器
    - 源路由器：源主机的默认路由器
    - 目的路由器：目标主机的默认路由器
    - 给定一组路由器以及连接路由器的链路，从中找到一条从源路由器到目标路由器的好路径

        > 例外：A和B之间的路径费用很低，但是二者处于两个组织之间，而这两个组织作出的路由策略不允许相互通行

- 路由抽象模型



    - 该模型表示为图$G(N,M)$
    - 其中路由器集$N=\{u,v,w,x,y,z\}$，链路集
      $M=\{(u,v),(u,x),(v,x),(v,w),(x,w),(x,y),(w,y),(w,z),(y,z)\}$
    - $c(x,x')$为节点X和X'间边的费用，例如图中$c(w,z)=5$

        > 费用可以是经济的，但也有可能和链路的带宽以及链路拥塞状况有关

    - 路径费用$(x_1,x_2,x_3,\cdots,x_p)=c(x_1,x_2)+c(x_2,x_3)+\cdots+c(x_{p-1},x_p)$
    - 选路算路就是选取路径费用最低的路径
- 选路算法分类
    - 按照性能目标分类
        - 链路状态算法——基于路径成本最优
            - 所有路由器都知道整个网络拓扑图以及链路的费用信息
        - 距离向量算法——基于路径距离最短

        - 每个路由器仅有与其相连链路的费用信息，通过迭代计算过程与相邻节点交换信息
        - 路由信息可以更快地变化，可以响应拓扑或链路费用的变化
    - 按照负载是否敏感分类
        - 负载敏感算法：链路费用会动态地变化以反映出链路的当前状况
        - 负载迟钝算法：链路费用不明显地反映链路的当前状况

4.5.2 链路状态选择算法LS与$OSPF$

- 迪杰斯特拉算法
    - 算法概述
        - 给定带权图$G =< V, E, W >$（所有边的权重为正值）和源路由器s，找到源路由器s到所有其他路由器t的最小成本$\delta(s, t)$和最小成本路径$< s, \ldots, t >$
        - 算法从结点集$V - S$中选择当前最小成本路径估计最小的路由器u，将u从Q中删除，并加入到S中，$u.d$就是源路由器s到u的最短路径的长度。这里Q是一个最小优先队列，保存结点集$V - S$
        - 以每个结点为源路由器，由上述算法得到的最小生成树即可得到**源路由器到其他路由器的转发表**
    - 前提：所有节点都知道网络拓扑和链路费用
        - 所有节点具有该网络的同一个完整的视图
        - 通过链路状态广播获得信息
    - 目标：产生某节点的转发表
        - 计算从某节点到网络中所有其它节点的最低费用，并产生转发表
    - 算法复杂度为$O(n^2)$
    - 算法的问题
        - 当模型采用负载流量作为路径成本，即模型为负载敏感型网络时，会产生振荡问题
        - 解决方案
            - 不以负载流量作为成本——无法解决高拥塞问题
            - 所有的路由器不同时运行LS算法
- $OSPF$协议
    - 即$Open\ Shortest\ Path\ First$协议，公开发表的最短路径优先协议
    - 协议交互范围：工作在本自治系统域内，采用**泛洪法**发送消息
    - 协议交互消息内容：**与本路由器相邻的所有路由器的链路状态**
    - 协议交互时机：**仅当链路状态发生变化时**，采用泛洪法向所有路由器发送信息
    - 当链路状态发生变化时，每个路由器都向本AS中的所有路由器发送与本路由器相邻的所有路由器的链路状态，信息发送完毕后，所有路由器上都将有全网一致的拓扑结构图

        > 即使链路状态未发生变化，每30分钟广播一次链路状态
        >
        > 链路状态以$OSPF$通告的形式封装在报文中，由IP分组承载（协议号：89）
        >
        > $OSPF$路由器之间的交换经过$MD5$鉴别，以确认$OSPF$通告的真实性，防止伪造和篡改

4.5.3 距离向量选择算法DV与RIP

- 距离向量选择：$d_x(y) = min_v c(x,v) + d_v(y)$
- RIP路由表更新算法
    - 路由器X得到相邻路由器Y的路由表，从而得知Y到网络Z的最短距离为N
    - 如果路由器X没有到网络Z的路由条目，则添加一条经由路由器Y到网络Z距离$N+1$的路由条目
    - 如果路由器X已有到网络Z的路由条目，其距离为M，如果$M > N+1$，则更新该条目为经由路由器Y到网络Z距离$N+1$，否则不更新
- 当链路状态改变时
    - 在$t0$时刻，y检测到链路费用变化，更新距离向量，同时将这个变化通知给它的邻居
    - 在$t1$时刻，z收到来自y的更新报文并更新距离向量表，计算出到x的新的最低费用，并向邻居发送它的新距离向量
    - 在$t2$时刻，y收到自z的更新并更新其距离向量表，Y的最低费用未变，因此y不发送任何报文给z
- 协议参数
    - 链路费用：相邻两点链路费用为1跳，最大费用限制为15
    - 通告周期：选路更新通告周期为30秒
    - 邻居离线：邻居离线判定周期为180秒
    - 协议端口：基于UDP，端口为520

4.5.5 域间$BGP-4$协议

- 层次路由
    - 因特网规模过大，导致路由器无法存储每台主机的选路信息，路由表更新的报文广播将导致无剩余带宽供发送数据使用
    - 将路由器聚合到一个区域，即自治系统AS，在不同AS内的路由器可以运行不同的自治系统内部选路协议
- 转发选路算法
    - 转发表是由AS内部选路算法和AS间选路算法共同决定的
    - AS内部选路算法为内部目的地址设置转发表信息
    - AS内部选路算法和AS间选路算法共同为外部目的地址设置转发表信息
    - 假设从源到目标仅有一条路可选
        - 假设$AS1$从AS间选路协议知道子网x经过网关路由器$1c$至$AS3$可达，但是通过$AS2$不可达
        - $AS1$向它的所有路由器广播该可达信息
        - 路由器$1d$知道，它的接口I在到路由器$1c$的最低费用路径上
        - 从而路由器$1d$将表项(x, I)放入其转发表
    - 假设从源到目标有多条路径可选
        - 现在假设$AS1$知道子网x可以通过$AS3$和$AS2$到达
        - 为了配置转发表, 路由器$1d$必须决定通过哪个网关路由器转发报文（$1b$或$1c$)
        - 热土豆选路: 将报文发送到最近的路由器
- $BGP-4$发言人
    - 每一个AS要选择一个路由器作为该AS的BGP **发言人**
    - 两个BGP发言人通过一个共享网络连接在一起

- BGP发言人通过BGP通告广播该自治系统AS能够到达哪些网络
    - 通过将多个路由前缀聚合为单一前缀并转发之达到路由通告的目的
    - 发言人得知某些通告信息后，向该AS内的路由器广播，路由器为之创建新的表项
    - 路由通告中包含前缀（能够到达的网络前缀）、路径（到达前缀地址经过的路径）、下一跳（到达前缀地址需要经过的下一跳地址）

五、链路层

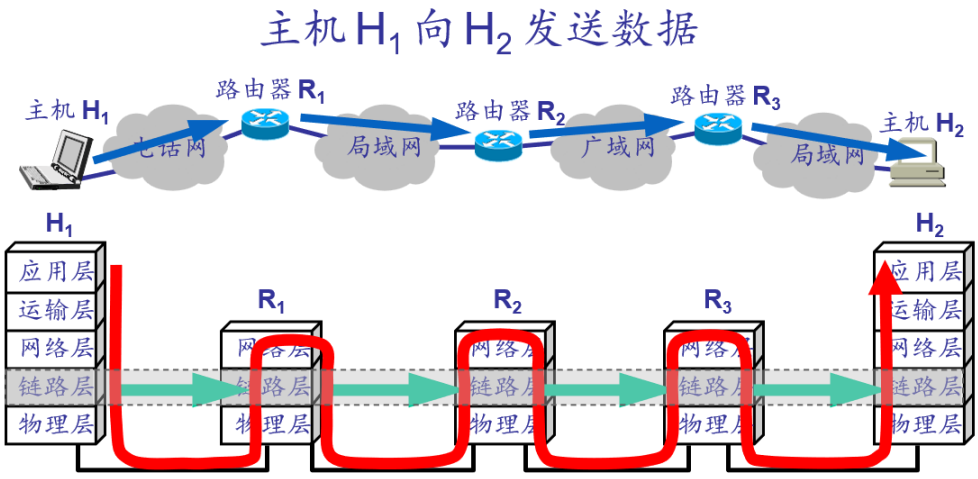
5.1 概述

5.1.1 术语

- 主要讨论传播时延：从输出链路的起点到目的地传播所需的时间
- 节点：主机和路由器
- 链路：沿着通信路径连接相邻节点的通信信道
- 帧：数据链路层的分组单元

> 链路层负责将数据报封装成**帧**通过链路从一个节点传输到**物理上相邻**的下一个节点

5.1.2 链路层基本模型



- 数据报在不同链路上可能**由不同的链路层协议**进行处理

> 第一段链路上由PPP处理，最后一段链路上由以太网处理，中间链路上由广域链路层协议处理

- 不同的链路层协议可能提供不同的服务
- 链路层提供的服务
    - 成帧、链路访问：将数据加上头部和尾部，封装成数据帧，其中帧头部用MAC地址标识源和目的地
    - 可靠传递：用于误码率高的链路，如无线链路
    - 流量控制：在相邻的收发节点间限制流量
    - 差错检测：接收方检测到错误存在后，给发送方发送信号要求重传或丢弃该数据帧
    - 差错纠正：接收方检测和纠正帧中错误，不用重传
    - 半双工和全双工：半双工时，链路两端的节点都能传输分组，但不能同时传输
- 链路层的实现
    - 链路层在“适配器”（网卡NIC）或者芯片上实现

5.1.3 差错检测和纠正

- 单比特奇偶校验：检测单个比特错误
- 二维奇偶校验：检测和纠正单个比特错误
- 因特网检查和：用于TCP、UDP和$IPv4$协议中
- CRC冗余校验：广泛应用于以太网、802.11 $WiFi$、ATM

5.2 多路访问链路和协议

5.2.1 链路概述

- 链路类型
    - 点到点链路：PPP/以太网交换机和主机之间的点到点链路
    - 广播链路(共享线路或介质)：传统以太网/802.11无线LAN
- 链路特点
    - 单个共享广播信道
    - 两个或多个节点同时传输时，会相互干扰

        > 碰撞：一个节点同时收到两个或多个信号

5.2.2 信道划分协议

- 协议概述
    - 信道划分协议：将信道划分成小的“片”（时隙、频率、编码）分配给节点使用
- $TDMA(Time\ Division\ Multiple\ Access)$
    - 循环访问信道
    - 每个节点在每次循环中得到固定长度的时隙（时隙长度 = 传输单个分组时间）
    - 没有数据发送的时隙空闲
- $FDMA(Frequence\ Division\ Multiple\ Access)$
    - 信道按频谱分成若干频段
    - 每个节点分配固定频段
    - 在频段不用时该部分信道被闲置和浪费
- $CDMA(Code\ Division\ Multiple\ Access)$
    - 每个用户使用自己的码片序列对数据编码
    - 当需要发送比特1时，发送$mbit$码片序列
    - 当需要发送比特0时，发送$mbit$码片序列的二进制反码

5.2.3 随机访问协议

- 协议概述
    - 当节点有数据发送时，以信道全部速率R传输，没有主节点起协调作用，因此两个或多个节点发送时会发生**碰撞**
    - 如何检测碰撞
    - 如何从碰撞中恢复，如延时后重传
- $ALOHA(Additive\ Link\ On-Line\ HAwaii\ system)$
- 时隙$ALOHA$
- 载波监听$CSMA$
- 带冲突检测的载波侦听$CSMA/CD$

5.2.4 轮流协议

- 协议概述
    - 信道划分协议在**低负荷**时效率低——即使只有一个活动节点，也只能分配到$\frac{1}{N}$的带宽
    - 随机访问协议在**高负荷**时效率低——碰撞的开销增加
- 轮询协议
- 令牌传递协议

5.3 交换局域网

5.3.1 MAC地址

- 又称为LAN地址、物理地址
- 48比特，前24比特由$IEEE$分配管理——OUI号，后24比特由厂商自行分配
- 通常采用**十六进制**表示法，如$5C-66-AB-90-75-B1$
- 在数据链路层标识**每块网络适配器**，使得能够在广播信道上寻址目标节点
- MAC地址烧入网络适配器的ROM中，**不可更改**
- MAC地址类似于身份证号，不会随着网络迁移而改变；IP类似于邮件通信地址，需要根据网络配置策略更改
- 和网络层地址类似，主机和路由器上的每个接口（适配器）也都有链路层地址，但是链路层交换机的接口没有链路层地址
- 链路层交换机的作用是在主机和路由器之间承载数据报，并**透明**地执行该任务

5.3.2 地址解析协议ARP

- 协议概述
    - 根据目标的IP地址获取其MAC地址
    - 每台主机或路由器上存在ARP表，包含从IP地址到MAC地址的映射关系，具体存储为$<IP, MAC, TTL>$
    - ARP协议工作在网络层和链路层之间
- 同一局域网内工作流程
    - 建立ARP请求包

MAC报头		IP报头		ARP请求报文
目的	源	目的	源	你的MAC
FF-FF-FF-FF-FF-FF	02-60-8C-01-02-03	197.15.22.126	197.15.22.33	地址是什么?

    - 广播ARP请求包
    - 建立ARP应答包

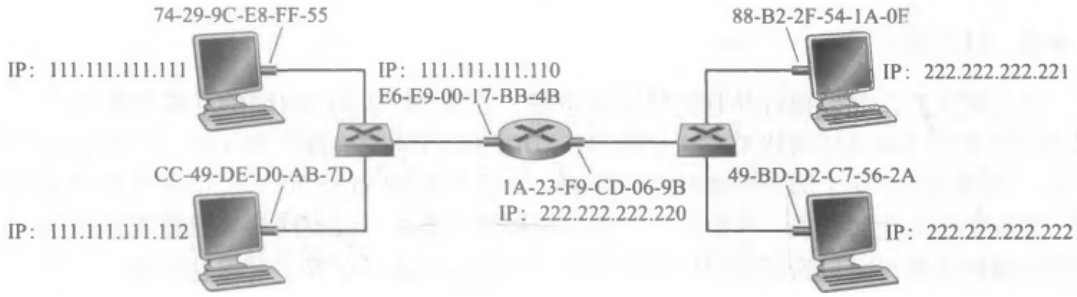
MAC报头		IP报头		ARP应答报文
目的	源	目的	源	我的MAC地
02-60-8C-01-02-03	08-00-02-89-90-80	197.15.22.33	197.15.22.126	址是……

        - 局域网内的所有适配器都把帧中的ARP分组向上传递给ARP模块
        - 与目的IP地址匹配的适配器构建应答包

```
1 - 向源发送应答包
```

```
- 源更新$ARP$表
```

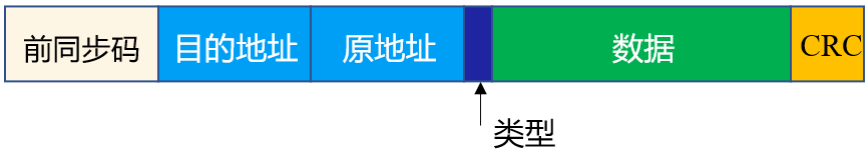
- 局域网间工作流程



  - 发送方主机首先获取两个局域网间路由器的端口的MAC地址（ARP）
  - 发送方向路由器发送帧
  - 路由器根据路由转发表转发数据报到输出接口
  - 输出接口将数据报发送给其适配器
  - 适配器封装称新的帧，发送至另一个子网，此时的MAC地址即为目的主机地址

5.4 以太网

5.4.1 以太网帧结构



- 数据最长1500个字节
- 前同步码：总共8字节
  - 前7字节用于唤醒接收适配器，并同步时钟
  - 前7字节为10101010，最后一个字节为10101011
- 地址：6字节，若适配器收到以太网帧，目的地址为自己的MAC地址或广播地址（如ARP包），就将帧中的数据传给网络层，否则丢弃该帧
- 类型：上层协议类型（大多为IP协议，也支持其它协议，如$AppleTalk$）
- CRC：由接收方检查,若检测到错误，就将该帧丢弃
- 以太网提供的服务
  - 无连接的服务：在发送适配器和接收适配器之间不需要握手
  - 不可靠的服务：接收适配器**不发送确认帧或否认帧**给发送方

5.4.2 $CSMA/CD$

- 特点
  - 没有时隙
  - 当适配器侦听到其它适配器在传输，则它不传输帧，即**载波侦听**
  - 正在传输的适配器若检测到其它适配器也在传输，则它中止自己的传输，即**碰撞检测**
  - 在重新传输之前，适配器要等待一段随机时间，即**随机回退**
- 术语
  - 拥塞信号：长度为48比特，用来确保所有传输者都能检测到碰撞而传输的信号

    - 比特时间：传输1比特所需时间

      > 在$10Mbps$的以太网中，当$K=1023$时，等待时间大约为$50ms$

- 算法
    - 适配器收到来自网络层的数据报，创建帧
    - 若适配器检测到信道空闲，则开始传输帧；若检测到信道忙，就开始等待，直到信道空闲再开始传输该帧
    - 若适配器传输了整个帧而没有检测到其它适配器的传输，则该适配器完成该帧的传输
    - 若适配器在传输时检测到其它适配器也在传输，则停止传输，发送拥塞信号
    - 中止传输后，适配器进入指数回退阶段，在经历第m次碰撞后，适配器随机从$\{0,1,2,\cdots,2^m-1\}$中选择K值。适配器在等待$K*512$比特时间后，返回第2步
- **指数回退算法**
    - 目的：适配器重传时试图估计正确的负载
        - 重载：随机等待的时间可能会更长
    - 第一次碰撞后：从$0,1$中选择K；延迟是$K*512$比特传输时间
    - 第二次碰撞后：从$0,1,2,3$中选择K
    - 第十次碰撞后：从$0,1,2,3,4,\ldots,1023$中选择K
- **只能进行半双工通信**

5.4.3 争用期

- 什么是争用期
    - A向B发送数据，τ后到达B（端到端传播时延，记为τ）
    - 若B在A的数据到达之前，发送自己的数据（这时B检测信道是空闲的，因为它没收到任何数据），则必然会与A的数据在信道中发生碰撞
    - A在发送后多久才能知道发生碰撞？要等到B的数据到达A，而A还未结束发送
    - 假设一个极端情况，当图中所示的时间差δ（δ为B发送数据时刻与A数据到达B的时刻的时间差）趋向于0，则A检测到发生碰撞的时间$2\tau-\delta=2\tau$
    - 当超过这个时间都未检测到碰撞，则A发送出的数据就一定不会产生碰撞了。
    - 以太网中端到端的往返时间2τ称为争用期，也叫碰撞时间
    - 只有经过争用期这段时间还未检测到碰撞，才能肯定这次发送不会发生碰撞，这时就可以放心把这一帧发送完毕。
- 传统以太网（$10Mbps$）规定争用期为$51.2\mu s$，最短有效帧长为64字节
- 最短有效帧长=2τ*链路传输速率
- 当传送前64个字节内没有发生碰撞时，就一直占用信道直到传输完毕所有字节（此时有其他数据来到时必须等待）
- 如果发生碰撞，则一定是在发送的前64字节之内
- 任何小于64字节的帧都是由于冲突而异常中止的无效帧

5.4.4 以太网交换机

- 链路层设备，负责存储转发以太网帧
- 主机不知道交换机的存在
- 检查帧头部，根据目的MAC地址转发
- 交换机的工作原理
    - 交换机不转发同一网段内通信的帧

    - 当收到帧的目的地MAC地址属于另一个网段，则通过交换表决定向何端口转发
    - 类比于物流中转站，从不同的物流点接受包裹（帧），当物流中转站发现包裹没有问题（帧无差错）时，保留包裹（缓存帧），否则丢弃包裹但不会要求商家重发包裹（**不会要求帧重发**），而由运输层处理丢包问题。物流中转站根据包裹发往的地址决定转到什么物流（交换机转发），并且不会修改包裹的寄件人地址（交换机不修改帧的源地址）
- 交换机转发和过滤
    - 过滤：决定一个帧应该转发到某个接口还是丢弃帧
    - 转发：决定一个帧应该被导向到哪个接口，并将其移动到那些接口
    - 转发和过滤借助**交换机表**完成，其表项为<MAC地址，通向该地址的交换机接口，表项放在表中的时间>
- 交换机自学习
    - 交换机表初始为空
    - 对于在每个接口接收到的每个入帧，存储一个表项
    - 根据表项存储时间和老化期清除表项
- 与路由器对比

	路由器	以太网交换机	集线器
类型	网络层设备	链路层设备	链路层设备
维护	维护路由表	维护交换表	
算法	路由算法	MAC地址过滤、学习算法	
是否需配置	需要配置	即插即用	即插即用

六、无线网络

6.1 概述

- 特性
    - 无线特性：基于无线链路
    - 移动特性：用户的网络接入点变化
- 包含固定基础设施的网络组成
    - 无线主机——手机
    - 无线链路——大气层
        - 信号强度递减
        - 会受到来自其他源的干扰
        - 多径传播
        - 比特差错率高于有线网络
            - 采用CRC进行校验
            - 采用ARQ协议重传
    - 基站——连接无线网络，负责转发覆盖范围内的主机的分组，起到链路层中继作用
        - 关联：主机在某个基站的覆盖范围内
        - 切换：主机从某个基站切换到另一个基站
    - 基础设施——预先建立的固定基站

- $Ad\ hoc$网络
    - 不包含固定基础设施的自组网络——无基站
    - 每个移动主机兼具主机和基站的作用
    - 节点(移动主机)仅仅能够在其覆盖范围内向其他节点传送数据
    - 节点之间相互通信组成的临时网络：在它们内部进行选路和地址分配
- 无线链路的质量
    - 信噪比$SNR(SIGNAL-NOISE\ RATIO)$：信号强度与噪声强度的比值
    - 比特差错率$BER(Bit\ Error\ Rate)$
    - 调制方案相同，SNR越高，BER越低；SNR相同，比特传输率高的调制方案的BER高

6.2 $WiFi$概述

- 执行802.11协议的无线LAN
- 802.11协议是一个协议簇，使用$CSMA/CA$协议进行多路访问
    - $802.11a$的频率范围为$5.1\sim5.8GHz$
    - $802.11ac$的频率范围为$5.1\sim5.8GHz$，支持单流和多流通信
    - $802.11b$的频率范围为$2.4\sim2.485GHz$，不需要许可证
    - $802.11g$的频率范围为$2.4\sim2.485GHz$
    - $802.11n$的频率范围为$2.4\sim2.485GHz$和$5.1\sim5.8GHz$，支持单流和多流通信
- $802.11b$的信道划分
    - 将$85MHz$划分为11个部分重叠的信道，仅当两个信道间隔4个及以上的信道时没有重叠，可以同时工作，如1、6、11
    - 每个无线访问接入点AP周期性发送信标帧，包含自己的$SSID$和MAC
    - 主机扫描11个信道获取所有可用的AP的信标帧
    - 主机连接到某个AP，加入其子网，并通过$dhcp$获取IP地址（需要身份鉴别）
- 发送流程
    - 监听到信道闲置$DIFS$秒后才开始传输帧，并且不进行冲突检测
    - 监听到信道忙后，则定时避退，定时到且信道闲置就发送数据
    - 接收方收到帧后，等待$SIFS$秒发送ACK
    - 发送方收到确认后，继续发送数据；没有收到确认则重新发送
- 冲突避免
    - 发送方在发送帧之前，使用$CSMA$协议发送短的请求RTS帧给AP（RTS也可能冲突）预约信道
    - AP回应允许发送CTS帧表示预约成功
    - 其他发送方也能接收到RTS帧，收到后推迟自己的发送
    - 如果RTS发生冲突，则两个发送方进行随即回退，总有一方先发送第二个RTS帧