计算机网络笔记

原作者陈鸿峥, 蔡怀广修订

2022.10*

目录

*Build 20221008

本课程使用的教材为 James F. Kurose 和 Keith W. Ross 的《计算机网络—自顶向下方法（第七版）》。

1 计算机网络概述

什么是计算机网络？连接多台计算机进行通信的系统

1.1 网络连接方式

1. 直接连接的网络（直连网）

   - 点对点 (point-to-point) 网络：包括专用介质 (dedicated medium)、节点/主机
     - 单向 (simplex)：如广播、电视
     - 半双工 (half duplex)：异步双向，如对讲机
     - 全双工 (full duplex)：同步双向，如电话
   - 多路访问 (multiple access) 网络：共享介质 (shared medium)，会产生碰撞 (collision)

  - 单播 (unicast)：一对一
  - 多播 (multicast)：一对多
  - 广播 (broadcast)：一对所有

2. 间接连接的网络：多个直连网的结合

1.2 因特网

用路由器或网关 (gateway) 连接起来构成的网络称为互连网络 (internetwork)。

什么是因特网？因特网/互联网 (Internet) 是一种互连网络，可以看作是把世界各地的广域网互连的网络，是世界上最大的特定计算机网络，采用**TCP/IP 协议簇**作为通信规则。

- 系统域网 (System Area Network, SAN)：主机和其周围设备
- 局域网 (Local Area Network, LAN)：某一区域内由多台计算机互联成的计算机组，一般是方圆几千米以内，如小型实验室；常用多路访问网络
- 城域网 (Metropolitan Area Network, MAN)：4G，5G，有线电视
- 广域网 (Wide Area Network, WAN)：因特网

因特网组成划分（方式 1）：

- 终端系统 (end system)：主机、运行网络应用程序，如手机、浏览器
- 通信链路 (communication link)：光纤、铜线、无线电、卫星等
- 路由器 (router)：用于连接多个网络形成更大的网络

因特网组成划分（方式 2）：

- 网络边界 (network edge)：主机及网络程序，终端设备可以通过本地 ISP 或区域 ISP 连接上互联网
- 接入网络/接入网 (access network)：有线或无线接入，连接订阅者和服务提供商，如 WiFi
- 网络核心/主干网 (core network)：顶层 ISP（中国电信、中国移动、中国网通），可以连接局部提供商

1.3 网络服务

通信服务类型：

- 可靠/不可靠：会不会丢包/收发是否完全相同，如文件传输[1]（可靠）/视频（不可靠）
- 面向连接/无连接：需不需要建立通信线路，如电话（连接，双方都要在）/寄信、因特网（无连接，对方可能不在）
- 有确认/无确认：需不需要确认对方是否收包，因特网不需要
- 请求响应/消息流服务：有请求才有响应，如浏览网页/一直发消息，如有线电视

因特网是数据报服务，无连接无确认（尽力服务）。

[1]例如：下载文件即可以利用应用层的 ftp 协议，也可以利用应用层的 http 协议

1.4 因特网体系结构

因特网体系结构[2]包括以下这**五层（简记:atnlp)**，而 ISO/OSI(open system interconnection) 参考模型包括七层：

- 应用层 (Application Layer)：提供对某些专门应用的支持，如FTP、SMTP、HTTP
- (OSI) 表示层 (Presentationn Layer)：提供数据转换服务, 如加密解密，压缩解压缩，数据格式变换
- (OSI) 会话层 (Session Layer):提供会话式的数据传送服务,如数据流的检查点设置和回滚，多数据流同步
- 传输层 (Transport Layer)：在**进程之间**数据传送（端到端），如TCP、UDP，用到了**端口号**
- 网络层 (Network Layer):**路由选择**,实现在互联网中的数据传送（主机到主机）,如IP 协议、路由协议
- 数据链路层 (Link Layer):在**物理网络（直连网）**中传送**包**（跳到跳[3]，节点到节点）,如PPP、Ethernet
- 物理层 (Phsical Layer)：线上的**比特**（传送原始比特流）

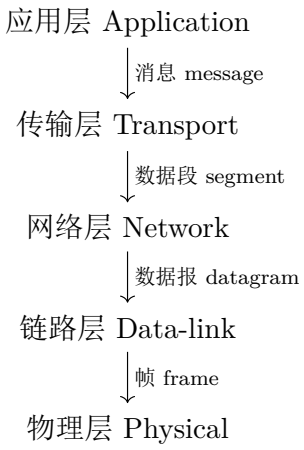
其中**网络层以下不可靠，以上可靠**；防止丢包的机制：**重发**。

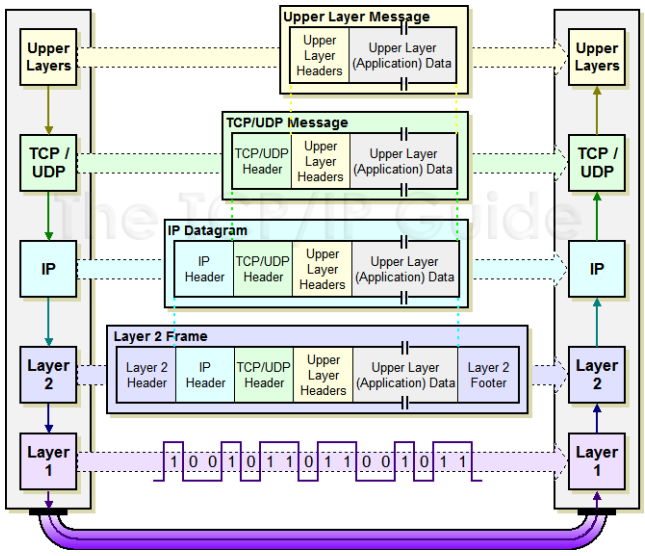
物理层和数据链路层又被称为物理网络，网络层和传输层被称为逻辑网络。

协议 (protocol)：在网络实体 (entities) 之间传送消息的规则，如消息的格式、收发消息的次序等。

TCP/IP 协议簇：将网络分为四层——将物理层和数据链路层合并起来变成物理网络层

每层传输的数据单元都称为**包**(packets)，都属于某个协议，又被称为**协议数据单元**(protocol data unit, PDU)，包括**头部/协议控制信息**(potocal control data, PCI) 和**服务数据单元**(service data unit, SDU) 两部分。



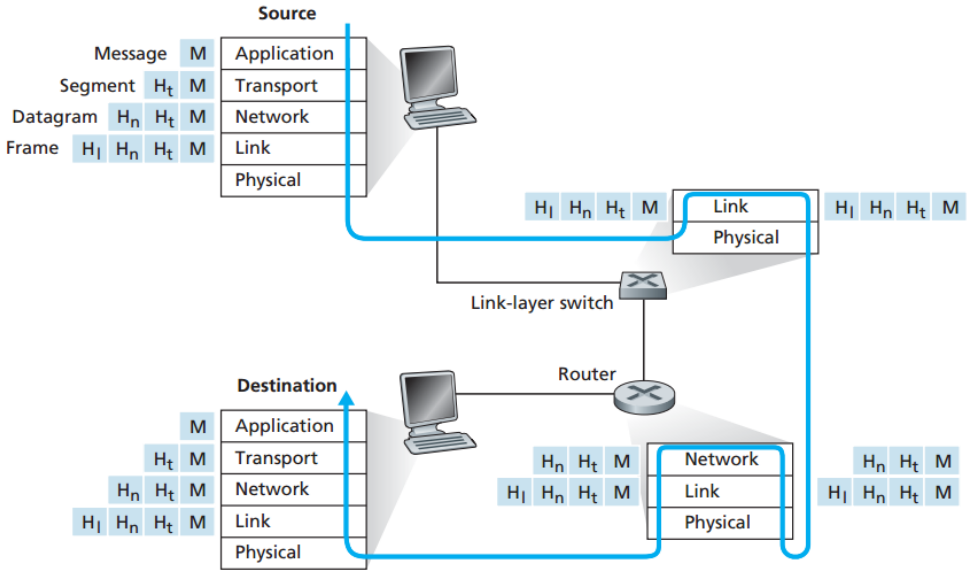


下层把上层通过服务访问点 (service access point, SAP) 传来的 SDU 用 PCI 封装为 PDU 后传给对等实体 (peer entity)，即实现相同协议的实体。同一个互连网络中网络层协议（及以上）需要相同，数据链路层和物理层协议可以不同。

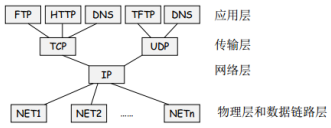
---

[2]按层划分的原因：简化网络设计；每个协议可以属于某一层或若干层，上层协议可以使用下层协议提供的服务。

[3]一跳 (hop)/节点为一个物理设备，即数据链路层只考虑直连网的情况

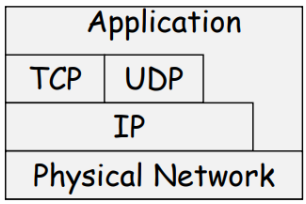


协议栈 (stack)：发送时封装 (encaptulation)，接收时拆封。



TCP/IP 协议簇 (protocol family)[a]

[a]IPV4 和 IPV6 不兼容，物理网络层的协议不属于 TCP/IP 协议簇



TCP/IP 协议簇分层[a]

[a]协议可以跨越分层

1.5 网络性能分析

当一个包到达时如果有空闲缓存则排队等待转发，产生延迟 (delay)[4]；如果没有空闲缓存，则丢弃该包，造成丢失 (loss)。

包交换网络中的延迟主要有以下四点：

- 处理 (processing) 延迟：检查比特错，确定输出链路
- 排队 (queueing) 延迟：依赖于路由器的拥塞程度
- 发送/传输 (transmission) 延迟：

$$\text{传输延迟} = \text{包长 (bits)}/\text{链路带宽 (bps, bit per second)}$$

指从发送第一个包到发送最后一个包的间隔

[4]为何不扩充容量来接受所有的包呢？这样延迟较大，重传更快

- 传播 (propagation) 延迟：指对于一个包来说从发送到接收所需的时间

$$\text{传播延迟} = \text{物理链路长度}/\text{信号传播速度}$$

接收延迟与传播延迟重合。故忽略掉处理、排队延迟，

$$\text{总延迟（从第一个包被发送到最后一个包被接收的时间）} = \text{传播延迟} + \text{发送延迟}$$

往返时间 (round trip time, RTT)：从源主机到目的主机再返回源主机所花的时间
带宽 (bandwidth)：一条链路或通道可达到的最大数据传输速率 (bps)
吞吐量 (thoughput)：一条链路或通路实际数据传输速率

例 1. 如果一个长度为 3000 字节的文件用一个数据包从源主机通过一段链路传给了一个交换机，然后再通过第二段链路到达目的主机。如果在包交换机的延迟为 $2ms$，两条链路上的传播延迟都是 $2 \times 10^8 m/s$，带宽都是 $1Mbps$，长度都是 $6000km$。采用以下三种方式，问这个文件在这两台主机之间的总延迟是多少？

1. 交换机采用存储转发方式
2. 将文件分成 10 个数据包，且存储转发
3. 收到一位转发一位

分析. 1. 因采用存储转发技术，先计算一段的延时，最后乘 2。

- 一段的传输延时：$3000B \times 8/10^6 bps = 24ms$
- 一段的传播延时：$6000km/(2 \times 10^8 m/s) = 30ms$
- 转发延时：$2ms$

总时长：$(24 + 30) \times 2 + 2 = 110ms$

2. 类似 1，但是总时长是一个包的传输传播转发延迟，加上剩余包的接收/传输延迟，见下表加粗部分

包 1	**传输**	**传播**	**接收**		
包 2		传输	传播	**接收**	
包 3			传输	传播	**接收**

- 一段的传输延时：$300B \times 8/10^6 bps = 2.4ms$
- 一段的传播延时：$30ms$
- 转发延时：$2ms$

总时长：$(2.4 + 30) \times 2 + 2 + 2.4 \times 9 = 88.4ms$

3. 同 1，但是只用计算一段传输延时，因为 1 位的转发延迟忽略。故总时长：$24 + 30 \times 2 = 84ms$

2 物理层

在直连网中传输原始比特流，不管包。需要做的事情：

$$\text{信息源} \to \text{调制/编码} \to \text{信道传输} \to \text{解调/解码} \to \text{目的地}$$

其中调制解调的对象为模拟信号，编码解码的对象为为数字信号。

信息能够被解释为数据 (data)，用符号 (sign) 记录，用信号 (signal)（光、电）传递 (transmit)，用熵 (entropy) 测量。

- 模拟信号：连续取值（连续波长而不是连续信息）
- 数字信号/跳变信号：离散取值

2.1 编码方式

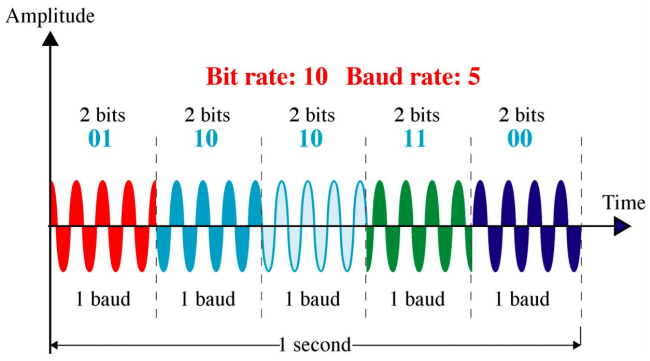
2.1.1 模拟信号

载波信号 (carrier) 一般采用正弦波信号：角频率 ω、频率 f、周期 T、振幅 A、相位 φ

- 频移键控 (frequency-shift keying, FSK)：通过不同频率表示不同信息
- 幅移键控 (amplitude-shift keying, ASK)：通过不同振幅表示不同信息
- 相移键控 (phase-shift keying, PSK)：通过不同相位表示不同信息
- 正交调幅 (quadrature amplitude modulation, QAM)：用不同的振幅和相位的组合表示不同的多位信息，如 $000 \sim 111$

信息传输速率的衡量方式：

- 波特率 (baud rate)：每秒码元传输的数目。码元：承载信息量的基本（最小）信号单位。
- 比特率 (bit rate)：每秒传输的信息量。比如下面单位时间间隔的波形是一个码元，每个码元的信号量为 2 比特。

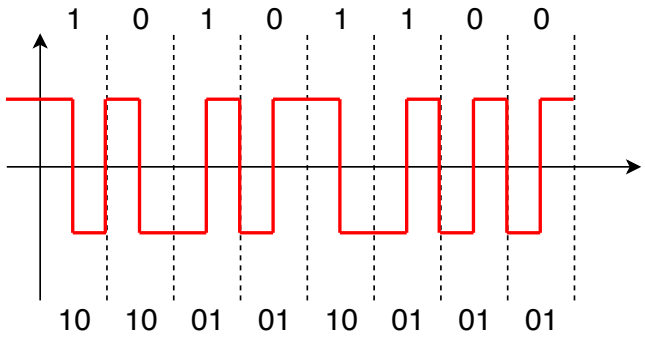


2.1.2 数字信号

1. 单极编码 (unipolar)：0V 即 0，$+E$V 为 1，但是会产生两种漂移 (这也是数字信号编码的基本问题)

- 时钟漂移：发送方和接收方采用**不同的时钟信号**，或长时间没有校正信号；一定要**有跳变**
- 基线漂移：线很长会有，长时间传输**相同电平信号**导致积累很多**同种电荷**，最后导致信号整体偏离基准线；一定要**有变化/正负**

2. 不归零编码/双极编码 (non-return-to-zero/bipolar, NRZ)：$-E$ 为 0，$+E$ 为 1。未能解决时钟漂移和基线漂移
3. 不归零反转编码 (Inverted, NRZI)：**差分**码波形，相邻码元（之前一个和当前的码元）的电位改变表示 1，而电位不改变表示 0；也可以反过来。该表示方法与码元本身电位或极性无关，而仅与相邻码元的电位变化有关。未能解决时钟漂移和基线漂移
4. 曼彻斯特 (Manchester) 编码：从相邻时刻的中间起降 $-E \sim +E$，$0 \to 10, 1 \to 01$，可克服时钟漂移和基线漂移；每一位都有跳变，这使得**电位无法一直不变**，**每一位中间时刻都可对时**。问题：频率高，传输有问题，对传输介质要求高
5. 差分曼彻斯特编码：在每一位开始时间如果跳变表示为 0，否则为 1，中间一定跳变，可克服时钟漂移和基线漂移



6. 4B/5B 编码：用 5 比特代表 4 比特，多一位冗余；每个编码没有多于 1 个前导零和多于 2 个末端零，即**最多 3 个 0**；如果结合 NRZI，就防止跳变过多，又可消除基线漂移和时钟漂移

4B	5B	4B	5B
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

2.2 物理介质

1. 有线介质

- 双绞线：
  - 非屏蔽双绞线 (unshielded twisted pair, UTP)：四对线（绿绿白、橙橙白、蓝蓝白、棕棕白），cat5/cat5e 百兆以太网，cat6 千兆以太网[5]
  - 屏蔽双绞线 (STP)：相较于非屏蔽双绞线，对线的外面多了一层 pair shield，四对线的外面多了一层 cable shield
- 同轴电缆 (coaxial cable)：从外到内：塑料护套，铜网屏蔽层（接地），内绝缘体，铜芯（信号传输）
- 光导纤维 (optical fiber)：利用光的全反射性质
  - 单模光纤 (single mode)：最大传输速率
  - 多模光纤：阶跃 (step-index) 光纤、渐变 (graded-index) 光纤

2. 无线介质：地面微波、WiFi、3G 网络、卫星

2.3 物理层传输方式

2.3.1 多路复用方式

- 时分多路复用 (time division multiplexing, TDM)：时间域被分成周期循环的一些小段，每段时间长度是相同的，每个时段用来传输一个子信道
- 频分多路复用 (frequency, FDM)：无线电台常用
- 波分多路复用 (wavelength, WDM)：利用多个激光器在单条光纤上同时发送多束不同波长激光的技术
- 码分多路复用 (code, CDM)：多路信号的编码方式是正交的，接收方用某个信号叠加在其之上就可解码出某路信号（其余信号被正交消掉）
- 统计多路复用 (static, SDM)：动态分配方法共享通信链路，比如 FIFO；对于多个可变速率的数据流，SDM 可以提高链路利用率

例 2. *如果有 8 个速率相同的数据流,且它们速率之和小于且接近一条链路的带宽,与用 8 个通道 (channel) 的 TDM 或 FDM 传送它们相比，采用统计多路复用技术的带宽利用率 (传送有效数据的比率) 怎么样？*

分析. *更差，都可以用完整个带宽，只是统计复用技术需要标识号（用于被接收方识别）或者目标地址，因此会差一点*

2.3.2 交换技术

- 电路交换技术：采用 FDM、TDM、WDM、CDM 技术；要先建立传输通道才能进行数据传输
- 包交换技术：采用的统计多路复用技术；

例 3. *在一个采用电路交换技术的网络中，将一个 640,000 bits 的文件从主机 A 发送到主机 B，链路 (link) 带宽为 1.536 Mbps，其建立要花费 500ms 建立，且使用 TDM——每分钟 24 个时间槽，求发送延迟？*

[5] 1KB(Kilobyte，千字节)，1MB(Megabyte，兆字节，简称 “兆”)，1GB(Gigabyte，吉字节，又称 “千兆”)

分析.

$$500\text{ms} = 640,000\text{bits}/(1.536\text{Mbps}/24) = 10.5\text{s}$$

3 数据链路层

数据链路层把**数据帧 (frame)**，从一个节点通过链路[6]（直连网络/**物理网络**中）传给**相邻**另一个节点（主机或路由器）。

数据链路层的功能如下：

(a) 成帧(framing)

(b) 差错检测(error detect)：比特错，纠错

(c) 差错控制(error control)：丢包、重复、错序、溢出等错误，实现流控制 (flow control)

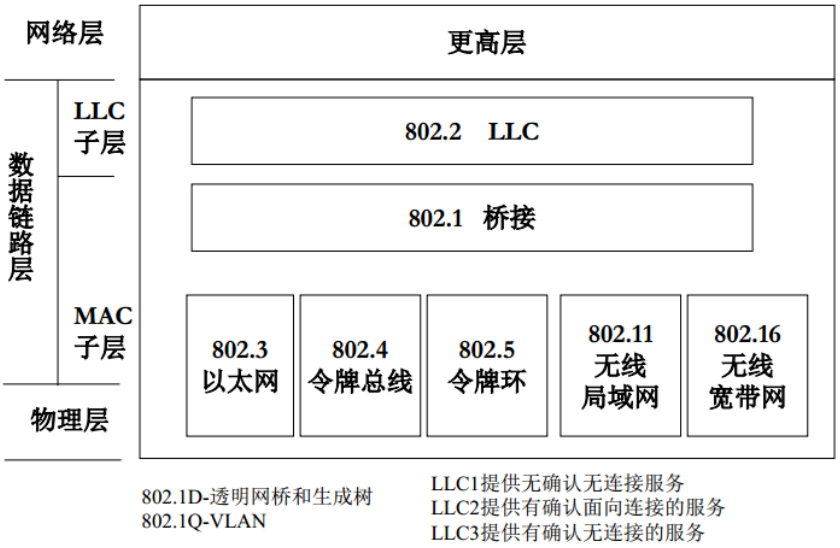
(d) 介质访问控制(medium access control)：多路访问，碰撞 (collision)

针对点对点和多路访问网络又分别制定了两个子层：

- 逻辑链路控制 (Logic Link Control, LLC) 子层：为上层协议提供服务
  - LLC1 提供**无确认无连接**服务
  - LLC2 提供**有确认面向连接**的服务，实现滑动窗口协议
  - LLC3 提供**有确认无连接**的服务
- 介质访问控制 (Media Access Control, MAC) 子层：控制和协调多路访问链路中节点对共享介质的访问，以避免或者减少冲突（点对点网络没有冲突就不用）

注意数据链路层、网络层错了就错了，不提供纠正服务，由上层纠正。链路层在**网络接口卡 (network interface card, NIC) 及其驱动程序**上实现，路由器在**接口模块**上实现。

[6]链路即连接相邻节点的通道，包括有线链路、无线链路、局域网等



IEEE802 系列标准

3.1 逻辑链路控制子层

3.1.1 差错检测

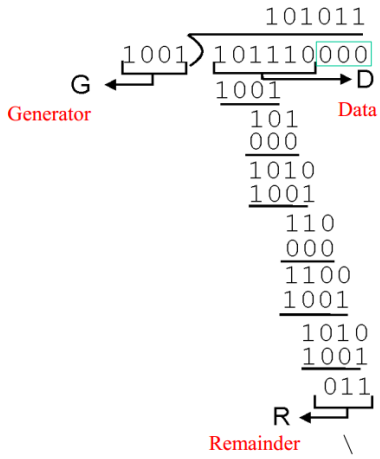
1. 奇 (偶) 校验：接收方收到奇 (偶) 数个 1 才正确
   - 一维偶校验：只能检错；最后补一位使得全部为偶数个 1，如 010 补为 010 | 1，而 101 补为 101 | 0
   - 二维偶校验：检错 + 纠错一位；横纵同时偶校验
2. 校验和 (checksum)：将所有数据加起来，每 16 位 1 组，最高位进位则末尾加 1，最后结果取反
   由于需要使用加法器，校验和一般不用于数据链路层，而用在更高层（网络层和传输层）

```
  10000110 10000111
+ 00000100 01000100
------------------
  10001010 11001011
+ 11000000 00000000
------------------
 1 01001010 11001011
+                  1
------------------
  01001010 11001100

反码：1011 010100110011
```

3. 循环冗余校验码 (Cyclic Redundancy Check, CRC)：补充 n 位 0 后除以一个 n+1 位的除数，模 2 除法（按位异或，做减法时没有借位）；如果传输过程中没有出现比特错，接收方用相同的除数去除 [数据和 CRC 校验码]，余数应该为 0。如下图，4 位除数补 3 个 0，最后的余数 011 即为校验码[7]

[7]模 2 除法时余数都是从 1 开始的，末尾处从被除数哪儿继承 1 位，如图



得到的余数就是 CRC 校验码[8]

数据链路层采用**CRC-32**校验码，因为检错率很高，且容易实现（触发器 + 异或门）

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

3.1.2 差错控制

流控制允许两个基站以不同的速率进行交互，主要有两大类方法：

1. 基于反馈 (feedback) 的流控制：发送方要收到接收方返回的确认才执行下一步操作
2. 基于速率 (rate) 的流控制：不需要接收方的确认，常被用于网络层和传输层

数据链路层的流控制是**基于反馈**的，确认帧的返回限制了这段时间内发送方能够发送的帧的数目。

每发送**一帧**都启动**一个超时定时器**，如果它的确认帧 (Acknowledgement frame, ACK) 在其超时时间内到达就删除该定时器；否则，自动重发请求 (Automatic Repeat reQuest, ARQ)，重传该帧并重启定时器

ARQ 协议是 OSI 模型中数据链路层和传输层的错误纠正协议之一。它通过使用确认和超时这两个机制，在不可靠服务的基础上实现了可靠的信息传输——发送方若未在**超时**时间内收到接收方传来的**确认帧**，那就**重传**。主要的 ARQ 协议包括停等协议 和滑动窗口协议。

以太网使用 CSMA/CD 协议进行来进行帧的发送，它没有确认帧，认为**没有冲突**就算发送成功，因此是不可靠的。和 PPP 协议一样，可靠性由上层保证

停等协议 (stop-and-wait)：发送方只有收到前一个数据帧的确认帧才可以发送下一个数据帧。效率/吞吐量十分低，信道空闲时间长；最少需要2 个序号[9]。

三种出错情况：

- 数据帧丢失 (loss)：发送到接收方的途中丢包；即发送方未在超时时间内收到确认帧，重发该帧即可

[8]当前余数就算把被除数的余下所有位都继承了，此时位数为 n，那么算法截至。比如数据为 1001，除数为 10011，那么 1001 0000 除以 10011 后余数为 1，就算继承了接下来的 000，也只有 4 位，因此，商就是 1，余数就是 1000

[9]停等协议至少需 2 个序号，0 和 1，轮流发送，如果重发则用同个序号，否则更换。接收方通过序号的更换就可以确认上一个确认帧发送方已经收到。

- 确认帧丢失：接收方回传时丢包；某数据帧的确认帧丢失后，超时重发此数据帧。接收方通过序号知道此数据帧重复接受（确认帧丢失或超时），丢弃然后再发一个确认帧。
- 确认帧超时[10]：若无序号，发送方无法区分目前数据帧和超时的之前数据帧的确认帧；两个序号应该可以解决？要考虑确认帧是否恰好就是发送方当前期望接受的，以及当前数据帧对应确认帧。

例 4. *把停等协议用于一个带宽为 20Mbps、长度为 3000 公里、传播速度为 200000 公里/秒的点到点链路，如果最长帧为 5000 字节，带宽的最大利用率（最大吞吐量/带宽）是多少？*

分析. *按照如下方法计算*

- *传播延迟 RTT*：$(3 \times 10^6 m)/(2 \times 10^8 m/s) \times 2 = 30ms$（注意是往返时间!）
- *传输延迟 L/R*：$(5000B \times 8)/(20 \times 10^6 bps) = 2ms$
- *吞吐量*：$L/(RTT + L/R) = (5000B \times 8)/32ms = 1.25Mbps$
- *带宽最大利用率*：*最大吞吐量/带宽* $=1.25/20 \times 100\% = 6.25\%$

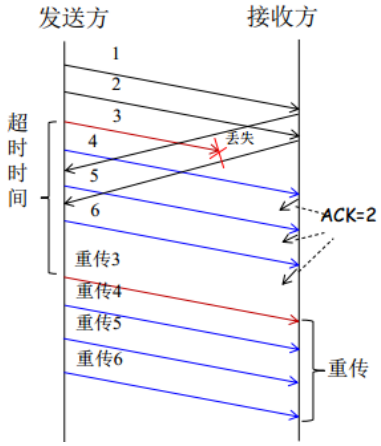
另，改为滑动窗口协议，窗口大小为 8，则最大利用率为 $8 \times 6.25\% = 50\%$

滑动窗口协议 (sliding window)：不需等待前面发送的帧的确认帧返回，就可以连续发送下一个，其个数不能超过发送窗口大小 (sending window size, SWS)[11]。

这里的确认帧是指在此之前的帧都已全部收到并已交给上层协议（直连网中间没有节点，后面收到前面一定收到；只要出错纠正不了直接丢弃），后面确认前面，提高可靠性。

滑动窗口协议又有以下两种：

- 回退 N 协议 (go back N)：某个 ACK 没收到则重传在此 ACK 之后的所有帧（超时[12]重传）；适用于出错率较低的情况



  - 发送窗口需要缓存 SWS 个帧，以便重传；接收窗口缓存 1 个帧就行；

[10]超时时间设为往返时延 (RTT,Round Trip Time)

[11]SWS 即连续发送数据帧可用序号范围，用于流控制

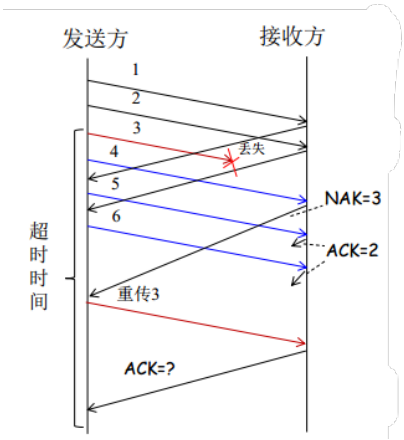
[12]超时时间略大于 1RTT

– 发送窗口之前的是已收到确认的，窗口内的可分为两部分：已发送但还未收到确认的，将要发送的；发送窗口中序号最小的的叫 sendBase；发送窗口每收到一个序号落在窗口内的确认帧就向后移动一下

– 回退 N 协议可能会**收到落在发送窗口之外的确认帧**：如果因确认帧迟到而出现超时重传，就可能收到一个帧的两个确认帧，第二个确认帧就会落在发送窗口之外

– 确认帧超时错误能否通过加长超时时间来避免？可以，但效率偏低（假设确认帧丢失了，发送方要过很久才能确认然后重传）

- 选择性重传 (selective repeat)：通过发送否定性确认帧 (negative acknowledgement, NAK)[13]要求重传该帧，NAK 也意味着这一帧之前的数据帧都已收到且交给上层协议，每个帧只发送一次 NAK



– 接收窗口 (receiving window size, RWS) 表示接收缓冲区大小（RWS≤SWS，最好是等于，尽量减少重传帧），用于确定应该保存哪些帧，帧可以错序到达（然后会给他们排序）；其中最小序号帧是期待接受的帧（recvBase，仅 1 个），当期待接受的帧抵达后，这个帧会和之后相连的已排序的帧一起传给上层协议（不一定是 RWS 大小），然后接收窗口移动到这些帧之后

– 帧的序号个数至少为 SWS+RWS；举个例子，假设帧的序号个数为 7，SWS=RWS=4，接收方发回的 1、2、3、4 确认帧都丢失了。此时接收窗口希望接受的帧的序号为 5、6、7、1，发送方超时后又发送 1、2、3、4 的帧，那么其中 1 号帧就被错误的接收了，因此序号至少为 8

– 超时时间应略大约**2RTT**（否则会导致丢失的确认帧对应的数据帧的后续帧都重传[14]）；无论窗口内窗口外收到都要发确认

– 选择性重传协议可能会**收到落在接收窗口之外的数据帧**：因确认帧丢失或超时到达而重传的数据帧都会落在接收窗口之外

– 选择性重传协议丢失了 NAK 并非致命错误，因为还有超时重传机制，保证该数据帧能够重新发送

ARQ 协议的超时时间不应设置得太长，否则会导致系统需要花很长的时间来纠正这些数据传输错误，吞吐量低；但也不能设太短，否则发送方会大量误认为帧丢失 而产生不必要重传。

[13]如果不采用 NAK，可以采用这样的方法：收到一个帧 3 个重复的确认帧后就重传该帧。这是网络层的实现方案，见后面的叙述。

[14]如果重置所有超时定时器，那超时时间可设为略大于 1RTT

例 5. 序号 *8* 个 *(0-7)*，*SWS=RWS=4*，按 *3456701234* 依次发送，*RTT* 大于 *4* 帧的发送时间。第一个 *5* 丢失，包含重传帧在内的其它帧均正确到达接收方，问接收方依次收到这些帧（含重传帧）的序号

分析. 回退 *N*：34 670 5670 1234

发送方先发 *3456*，然后等待确认帧；收到 *3*、*4* 确认帧后，发送窗口移动，接着发送 *70*；此时因为 *5* 丢失，接收方收到序号为 *6*、*7*、*0* 数据帧后发回的确认帧都是 *4*，发送窗口不移动直到超时；接着发送方重传 *5*、*6*、*7*、*0*，发送窗口每收到一个落在窗口内的确认帧就移动一下，直到 *1*、*2*、*3*、*4* 被接受。

选择性重传：34 670 5 1234

发送方先发 *3456*，然后等待确认帧；收到 *3*、*4* 确认帧后，发送窗口移动，接着发送 *70*；此时因为 *5* 丢失，接收方收到序号为 *6* 的数据帧，发 *NAK=5*；收到序号 *7*、*0* 数据帧后发回的确认帧都是 *4*，等待发送方重发 *5*；接收方收到序号为 *5* 的数据帧后，发送 *ACK=0*；发送窗口移动到 *0* 后，发送 *1*、*2*、*3*、*4*。

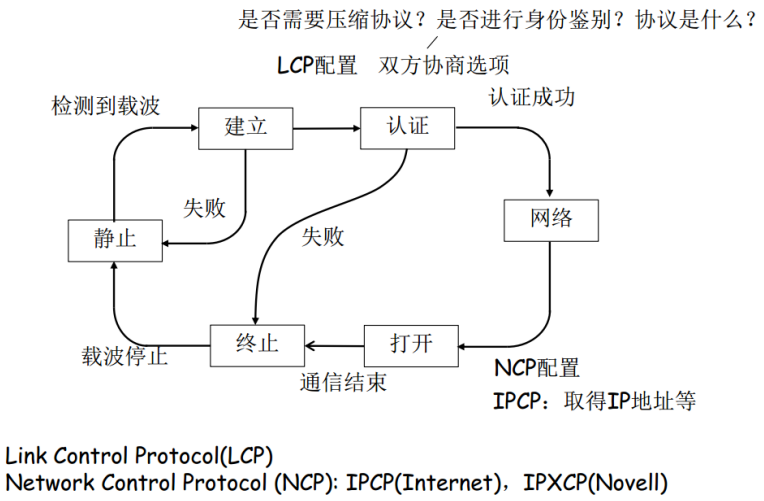
提高滑动窗口协议的效率：

- 选择性确认 (selective acknowledgement)：接受方把已收到的帧的序号告诉发送方，发送方要重传帧时，不会发送这些帧
- 捎带确认 (piggybacking)：通信双方全双工方式工作，接收方在发数据给对方时顺便把确认号也告诉对方（两个滑动窗口，两边都要发数据），需要结合延迟确认一起使用
- 延迟确认 (delayed acknowledgement)：接收方收到一帧后并不立即发送确认帧，而是等待一段时间再发送；这样可以减少确认帧的发送，节省带宽；相应的超时时间也延长；

3.1.3 PPP 协议 (point-to-point protocol)：点到点网络的数据链路层协议

- 根据 HDLC(high-level data link control) 协议进行设计，主要用于串行电缆、电话线 (MODEM) 等串行链路
- 提供连接认证、传输加密和压缩[15]功能，为网络层协议提供服务
- 采用字节填充法 (byte-stuffing) 替换掉保留字
- 没有纠错功能，也没有流控制和确保有序的功能，可靠性由上层保证
- ADSL 的 PPPoE 和 VPN 中的 PPTP 协议都采用 PPP 协议进行封装

[15]可以省略地址和控制字节 (头部压缩)。PPP 协议还可以进行 TCP 压缩和数据压缩



Link Control Protocol(LCP)
Network Control Protocol (NCP): IPCP(Internet)，IPXCP(Novell)

PPP 协议数据帧格式

1B	1B	1B	1B-2B	≤1500B	2B-4B	1B
标志	地址	控制	协议	数据	校验码	标志
0x7E	0xFF	0x03	0x0021 IP	···	CRC-16/CRC-32	0x7E

3.2 介质访问控制子层

3.2.1 简介

介质访问控制子层是针对多路访问链路（采用共享介质连接所有站点）的冲突问题的：随机访问协议(random access protocol)，以太网[16]；或者轮询 (take turns)，令牌环网

- 纯 ALOHA：想发送就发送，超时未收到确认则发生冲突
- 分槽 ALOHA：将时间分为长度相同的时槽，每个站点只在时槽开始时发送。
  信道空，立即以概率 p 发送，以概率 $1-p$ 延迟一个时间槽；信道忙，延迟一个时间槽。
- 载波监听 CSMA(Carrier Sense Multiple Access)：发送前先监听信道
  - 信道空，立即发送；信道忙，持续监听 (1-persistent CSMA，以太网)
  - 信道空，发送；信道忙，延迟一段随机长度时间 (non-persistent CSMA，较省电)
  - 信道空，立即以概率 p 发送，以概率 $1-p$ 延迟一个时间槽；信道忙，延迟一个时间槽 (p-persistent CSMA，**分槽 ALOHA**)

3.2.2 以太网物理层协议

以太网协议就是 IEEE 802.3 协议；IEEE 802.3 规定以太网[17]物理层标准：

- 传输方法：均使用异步传输，即信道空闲时以太网设备不任何发送信号（因此以太网才能成帧）

[16]以太网（英语：Ethernet）是为了实现局域网通信而设计的一种技术，它规定了包括物理层的连线、电子信号和介质访问层协议的内容。

[17]一开始以太网是 10Mbps 的传输速率，后来速率提高了，就给以太网加了修饰词

- 编码方法：曼彻斯特编码
- 命名规则：10BaseT 的 10 表示 10Mbps，Base 表示基带传输，T 表示双绞线；10Base2 的 2 表示最大距离 200m

以太网最短帧：

- 以太网 (10Mbps) 相距最远的两个站点之间的信号往返时间为 51.2 s(距离/信号在介质上的传播速度)
- 在“监听信道”的背景下，因为发送站点只在发送时检测冲突，因此冲突发生在离发送站点最远处后（也即冲突域最远处），返回的冲突信号回到发送站点后，要求此站点还在发送此数据，因此

$$\text{最短帧长度} = \text{数据传输速率} * \text{冲突域直径往返时间} = 10\text{Mbps} * 51.2\mu s = 512\text{b} = 64\text{B}$$

- 64B 也称为争用窗口 (contention window) 长度

其他几种以太网 (IEEE 802.3)，主要是在物理层不同。

- 快速以太网 (802.3u)：只是把传输速率提高到 100Mbps，其他均不变
  - MAC 子层的协议不变：CSMA/CD 协议不变，帧格式不变
  - 最大距离改为 100m (10base5 的最大距离为 2500m)，物理层改动
  - 帧间空隙隔依然为 96b，即 $0.96\mu s$(但对应的时间短了)
  - 100Base-TX、100Base-T4、100Base-FX
- 千兆以太网 (802.3ab)：除了把传输速率提高 1000Mbps，其它不变；相比于以太网，速度提高了 100 倍，因此半双工的冲突域变为 25m：提升冲突域距离的方法有两种
  - 载波延伸 (carrier extension)：通过附加填充字节，令帧长至少为 512B，网络半径可以延长到 200m（512B 是 64B 的 8 倍，因此 25 乘 8）。
  - 帧突发 (frame bursting)：一次传送多个短帧，第一帧小于 512B 时进行载波延伸（确保最短帧），后面的帧直接发出，不用加载波延伸。每一帧之间有一个小的间隔，填入延伸位。
- 万兆以太网
  - 保持帧格式不变
  - 光纤或双绞线，全双工
  - 无冲突，不使用 CSMA/CD 算法

3.2.3 以太网 MAC 层协议

以太网采用帧间空隙 (interframe space) 的成帧方法（每帧发送前要求信道空闲时间至少为 96bits，造成帧与帧之间有空隙）。采用带冲突检测的载波监听多路访问协议 CSMA/CD(carrier sense multiple access with collisi

1. 发送数据帧之前先监听信道。如果信道空闲，立即发送。如果信道忙，则持续监听，直到信道空闲，立即发送。
2. 边发送边检测冲突。如果发送完毕都没有检测到冲突，则发送成功。
3. 如果检测到冲突，则停止发送，并发送 32 位干扰位 (jamming signal，就是普通数据) 以加强冲突信号。采用二进制指数退避算法随机延迟一段时间后，转 (1)。

二进制指数退避算法 (binary exponential backoff)

- 规定最短帧是为了使发送站点可以监测到所有冲突。选择最短帧的发送时间作为其时间槽 (time slot)τ 的长度，其保证了首先发送的站点的信号可以到达最远的站点。如果先发送的只有一个站点，其他站点要不就检测到发送站点的信号而不能发送，要不就因为发送站点发送完毕而检测到信道空闲，总之不会产生过冲突。即任何间隔 τ 或以上时间的两个发送数据的站点不会发生冲突。
- 时间片 τ 的长度为 512b 时间，在 10Mbps 的以太网下为 51.2μs
- 第 i 次冲突从 $0, 1, \ldots, 2^j - 1$ 个时间片随机选择一个，$0 < i < 16$，$j = min(i, 10)$
- 前十次冲突后可选时间片数量每次加倍，后五次冲突后可选时间片数量不变，
  所以也称为截止式 (truncated) 二进制指数退避算法

例 6. *当一个以太网的信道忙时有五个站点都想发送一个最长帧（长度为 1520B），如果很长时间只有这五帧要发送，问最少经过几次冲突就可以全部发送成功？*

分析. *最长帧占用 $1520B \times 8/512b = 23.75$ 个时间槽，而在第 1、2、3、4 次冲突的延迟时间最多 16 个时间槽，这意味着从上一个冲突发生开始，信道忙的时间（最长帧发送）长于二进制指数退避算法延迟的时间，没有站点能延迟到信道空闲的时候。*

最好情况每次冲突后都让一个站点发送成功，所以最少 4 次冲突。详细来说，一开始信道空闲，5 个站点同时发送，第一次冲突，随机延迟 0 或 1 个时间片，假设 0 号立即发送，其他 4 个延后 1 个时间片，那么 0 号发送成功，其他 4 个检测到信道忙，不发送。到 0 号发完时，信道空闲，其他 4 个同时发送，第二次冲突，如此类推，每次成功发送一个。

802.3 的 MAC 帧格式

8B	6B	6B	2B	46B-1500B	4B
前导字符	目的地址	源地址	类型/长度	有效载荷/填充位[18]	帧校验序列

- 前导字符 (preamble)：同步字符 (7B) 和起始定界符 (start of frame delimiter)(1B)
- 目的地址：可以为接受者的单播、多播或广播 MAC[19]地址
- 源地址：一般为发送者的单播 MAC 地址

[18]这实际上决定了最小和最大传输单元 (Maximum Transmission Unit, MTU),为什么定这两个值可以见https://community.cisco.com/t5/other-network-architecture/why-the-mtu-size-is-1500/td-p/105418

[19]MAC 地址在一个直连网中有效，当包在直连网间传输时，MAC 地址要改变（通过 ARP 协议不断确定目标 MAC，修改源 MAC 地址）；IP 地址全球唯一，不用变（NAT 除外）

- 类型/长度：**0x0800 IP 数据报，0x0806 ARP 报文**，0x0835 RARP 报文
- 有效载荷 (payload)：用户数据，不足 46B 加入填充字节至 46B
- 帧校验序列 (frame check sequence)：对目的地址、源地址、类型/长度、有效载荷、填充位 进行**CRC-32**校验

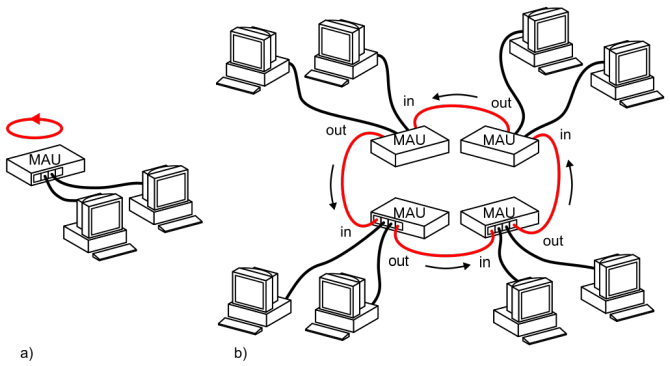
接收帧的方法

1. 以太网站点（网卡）会缓存所有的帧
2. 如果缓存的帧有错（长度错误，CRC 错等），则丢弃它。
3. 如果缓存的帧的目的地址为单播地址并且与接收该帧的网卡的 MAC 地址一致，则接收它。如果目的地址为多播地址并且为网卡预设的多播地址之一，或者为广播地址，也接收它。其它情况则丢弃它。
4. 如果把网卡设置为混杂模式 则会接收所有无错的帧。

3.2.4 令牌环网

令牌环网 (token ring, IEEE 802.5) 通过在站点之间传递令牌防止冲突并且具有优先权的星形**LAN**，为轮流协议(take turns protocol)，要与以太网的随机访问协议 区分。

多站点接入部件称为 MSAU(multistation access unit)



数据传送过程:

- 令牌 (帧) 绕环而行
- 只有截获令牌的站点才可以发送数据帧，各站点保有令牌帧的时间是相同的
- 发送的数据帧通过所有的活动站点
- 目的站点拷贝数据帧
- 只有发送方移除数据帧
- 当没有数据帧要发送或者持有时间到，当前的发送站点要释放令牌；被释放的令牌继续绕环而行

  令牌环网必然有特殊的站点（监控站点）产生令牌帧，选举出监控站点（MAC 地址最小）。
- 收到令牌的电脑坏了，那么令牌就消失了？监控站点具有超时机制，重发令牌

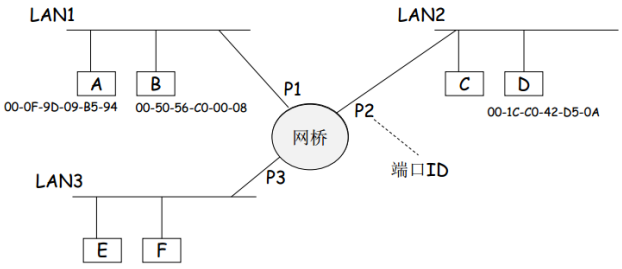
- 某站点发送完数据帧后坏了，没有站点回收此数据帧？监控站点为数据帧打标志，回收重复经过监控站点的数据帧

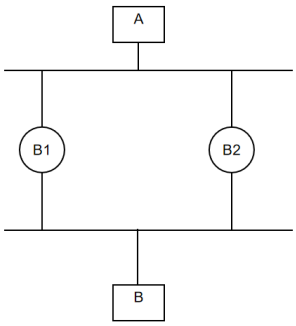
光纤分布式数据接口 (Fiber Distributed Data Interface, FDDI) 是另一种采用了令牌环的局域网，是一种 100 Mbps 的光纤局域网。

源路由桥接算法：由 IBM 开发的用于令牌环网的协议，将路径记到头部，下一次就不用查。为了兼容普通交换机，源路由网桥交换机也必须实现透明网桥的功能。

3.3 透明网桥

用网桥 (bridge) 连接若干局域网 (LAN) 可以建造一个更大的局域网, 称为桥接局域网 (bridged LAN) 或扩展局域网(extended LAN)。原来的局域网就成为该扩展局域网的一部分，称为该扩展局域网的一个网段 (segment)。





透明[20]网桥根据数据帧的目标地址转发到合适端口：

- 网桥收到单播帧：用该帧的目的地址查询 MAC 地址表。没有查到，则扩散[21]。若查到，看查到的端口是否为收到该帧的端口：是则丢弃（filter，过滤）；否则从此端口转发(forward)
- 网桥收到多播或者广播帧：扩散该帧

过滤操作可以处理部分单播帧的回路问题：

- 以上左图 A 发数据帧到 B 为例，对于多路访问链路中的节点，只有 B 会接受（因为此数据帧的目的地址和本机地址相同，其余节点不同则不会接受）；而网桥会先接受所有收到的数据帧。如果没有过滤操作，B 将会又收到一遍此数据帧——来自网桥的转发。加入过滤操作后，网桥发现目标地址 B 对应的转发端口就是收到此数据帧的接收端口，选择丢弃。
- 对于网桥连成的回路（上右图）：假设 B1、B2 都有 C 的 MAC 记录，当 A 发数据帧给 C 时，B1、B2 都会收到此帧。B1 收到后，转发到多路访问链路中，那么 B2、C 都会收到；同理，B2 收到后，转发到多路访问链路中，那么 B1、C 都会收到；如果没有过滤，那么数据帧会不断的在 B1、B2 间传递；当引入过滤操作后，对于 B1 转发给 B2、C 的帧，B2 会因为目标地址 C 对应的转发端口就是收到此数据帧的接收端口而拒收；因此不会形成循环。
- 对于网桥连成的回路（上右图）：假设 B1、B2 都没有 C 的 MAC 记录；或者 B1、B2 收到的是广播帧，此时会形成广播风暴，过滤操作无法解决。解决方案便是生成树协议，会使得数据链路层不

[20]对于站点透明：站点认为网桥不存在，认为目标站点也在当前局域网内

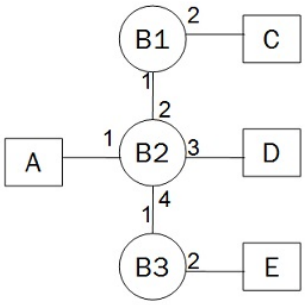
[21]flood，将该帧发送到除了接收端口外的所有其他端口

形成回路的。

透明网桥有自学习机制——就是形成一张 MAC 地址查询表：MAC 地址表（包含目的 MAC 地址，转发端口，TTL）初始为空；网桥接受所有帧，记下帧的**源地址和接收端口**。如信息从 A 主机-P1 端口来，则记为 A-P1，同时设好生存期 (Time to live, TTL)[22]。如果收到的帧有错则直接丢弃，根本不会学习。如果源地址不在表中，新建记录，启动定时器。如果源地址已经在表中，则更新转发接收端口[23]，并**重置超时计时器**。

例 7. *下面的扩展 LAN 包含三个透明网桥 B1、B2、B3 和四台主机 A、C、D、E。如果网桥的 MAC 地址表初始都是空的，在以下三次传输之后 MAC 地址表的内容是什么？*

1. *D 发送了一个帧给 E*
2. *A 发送了一个帧给 D*
3. *C 发送了一个帧给 A*



分析. *MAC 地址表如下*

B1 MAC 地址	*端口*	*B2 MAC 地址*	*端口*	*B3 MAC 地址*	*端口*
D	*1*	*D*	*3*	*D*	*1*
C	*2*	*A*	*1*		
		C	*2*		

3.4 生成树协议

生成树协议 (spanning tree protocol, STP) 构造一棵生成树（注意不是最小生成树）来防止交换机冗余[24]链路产生的环路，从而避免广播风暴

- IEEE 802.1D 生成树协议 + 透明网桥
- IEEE 802.1w RSTP(Rapid Spanning Tree Protocol)

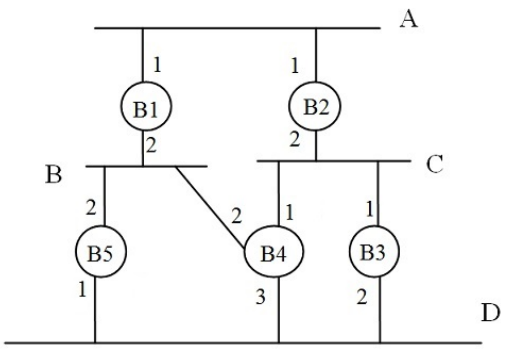
生成树的形成是通过所有网桥发配置消息 (BPDU, Bridge PDU) 给邻居达成共识实现的。生成树协议既能防止广播风暴，又能自动修复损害网桥（通过冗余方式），增加可靠性

- 根网桥，即**网桥 ID(Bridge ID, BID) 最小**的；根网桥没有根端口

[22]单位为秒，每次发送都会重置，对于不活跃的表项自动删掉（减少表的大小，查找速度更快）
[23]此机制解决了站点在网段间移动的问题
[24]交换机形成环路的优点在于提高可靠性——当某一条路断了，还存在另一条路可达

- 对于一个网段，直接相连的网桥中离根网桥最近的是指定网桥；网桥和相连的共享介质之间的距离为 1；相同距离时，BID 小的优胜；相同 BID，端口号小的优胜
- 网桥上离根网桥最近的端口为根端口，某网段的指定网桥上与该网段相连的端口为指定端口，网桥上非根端口又非指定端口的为阻塞端口
- 网桥只在根端口和指定端口之间转发数据帧，不可通过阻塞端口；这样便阻止了回路的生成

例 8. 下图显示了由五个透明网桥 *(B1-B5)* 形成的扩展 *LAN*，*A-D* 为网段。



分析. *1. B1* 是根网桥

2. 网段 *A-D* 的指定网桥 *(designated bridges)* 分别是

	A	*B*	*C*	*D*
指定网桥	*B1*	*B1*	*B2*	*B4*

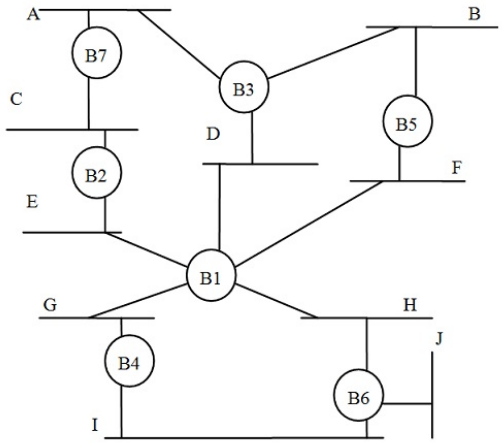
对于 *D* 网段，*B4*、*B5* 到根网桥距离都是 *1*，选 *BID* 小的

3. 网桥 *B1-B5* 的根端口、指定端口和阻塞端口分别是

	B1	*B2*	*B3*	*B4*	*B5*
根端口	无	*1*	*1*	*2*	*2*
指定端口	*1,2*	*2*	无	*3*	无
阻塞端口	无	无	*2*	*1*	*1*

指定端口的确认应该是从网段出发，先确定此网段的指定网桥，再确定指定端口；比如 *B* 的指定网桥是 *B1*，因此指定端口是 *B1* 中的 *2*；*A* 的指定网桥是 *B1*，因此指定端口是 *B1* 中的 *1*。这样 *B1* 的两个端口都是指定端口了

例 9. 下图是一个扩展 *LAN:*



分析. *a. 如果 B1 没有启动生成树算法但是转发生成树消息 (BPDU)，只生成 1 棵生成树，根为 B2；从物理的角度看，就是短路*

b. 如果 B1 没有启动生成树算法而且丢弃所有收到的生成树消息 (BPDU)，生成 2 棵生成树，根分别为 B2 和 B4；从物理的角度看，就是断路

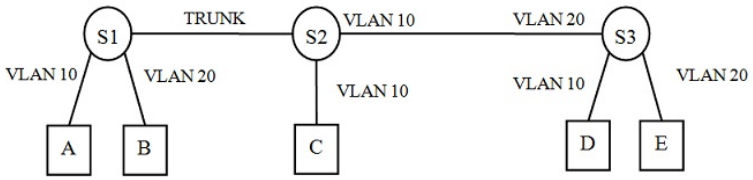
3.5 虚拟局域网

虚拟局域网 (Virtual LAN, VLAN, IEEE 802.1Q) 将原来的局域网分割成多个相互隔离的局域网，帧只在具有相同标号 (VLAN ID) 的端口[25]间转发。虚拟局域网转发的算法也是透明网桥，只不过为每种 VLAN ID 建立了单独的 MAC 地址表

如果所有交换机都是连通的，并且交换机连至交换机的接口都配置为干道 (trunk) 接口，交换机连至主机的接口都配置为 VLAN 接口（主机接口），则所有连至相同的 VLAN 接口的主机都位于**同一个广播域**，连至不同 VLAN 接口的主机位于不同的广播域。

网桥转发帧时，只有发往干道端口的帧才需要**加上 VLAN ID**。如果从干道收到的帧没有 VLAN ID，则认为是本征 (native)VLAN，默认为 VLAN 1。

例 10. 下图中哪些发送的帧将被目的主机收到



分析. *A、C、E 之间可以互发（可以实验验证），注意 S2 和 S3 的端口设错了（故意的）。如 E 到 A，S3 收到 E 发来的帧后，只在相同 VLAN ID 的端口间转发帧，因此发给另一个 VLAN 20 的接口（**未带上 VALN ID**）S2 从 VLAN ID 为 10 的端口收到帧，可以转发给 C（如果目标地址是 C 的话），于是扩散*

[25]VLAN ID 只存在于网桥内部，从终端发往网桥的帧是不带 VLAN ID 的

到干道端口 *TRUNK*并加 ***VLAN10***，发到 *S1*。*S1* 接收到后转发至 *A*。
而 *D* 到 *B* 没有办法，因为从 *S3* 就转发不出去，没有干道端口。

虚拟局域网帧格式也可以看作以太网帧格式，这时将优先权到类型这 4B 看作“数据”，相应的，数据的最小长度仍为 46B

虚拟局域网帧格式

6B	6B	2B	2B	2B	46B-1500B	4B
目的地址	源地址	协议 (0x8100)	3b 优先权 1b 标准格式指示位 12b VLAN ID	类型	数据	CRC-32

多生成树协议：管理员规定哪些 VLAN 为一组，构成多生成树，其余的用公共生成树

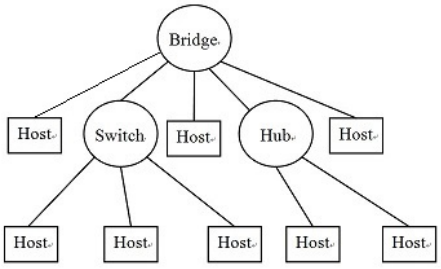
- 公共生成树 (common spanning tree, CST)：所有 VLAN 共享一棵公共生成树
- 多生成树 (Multiple spanning tree protocol, MSTP)

3.6 物理设备

集线器（hub，中继器）属于物理层的器件，采用电子线路方法模拟总线方式的以太网，若两台主机同时发送会产生冲突，所以是半双工工作。如果通过两个接口同时发送数据会产生冲突，则这两个接口属于同一个冲突域(collision domain)。

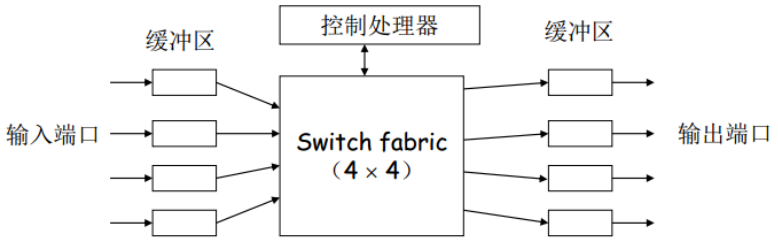
一个广播帧可以到达的所有接口属于同一个广播域。属于同一个冲突域的以太网部分称为网段(segment)。

例 11. 下图的冲突域和广播域个数？



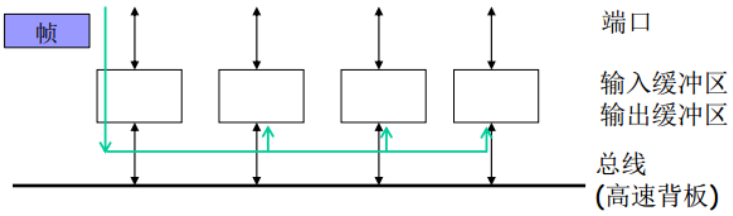
分析. 交换机 *(switch)*/网桥 *(bridge)* 的每个端口处于一个冲突域，集线器 *(hub)* 的所有端口处于一个冲突域。图中 *1* 个 *hub*，两个 *switch*，故 *8* 个冲突域，*1* 个广播域。

交换机 (switch) 是一个把多个网段连接起来的设备，也称为多端口网桥。

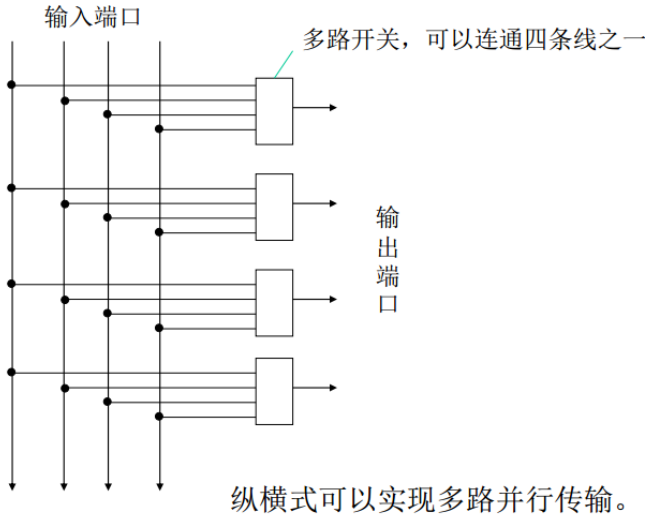


交换结构 (fabrics)

- 共享总线式：存在冲突问题；高速背板的带宽非常大，大于所有端口之和



- 纵横式 (crossbar)：可实现多路并行传输



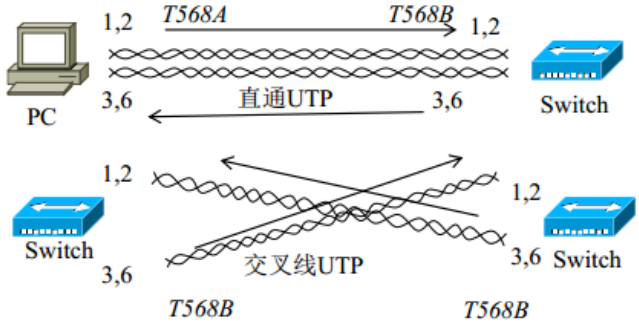
交换机转发方法

- 存储转发 (store and forward)：交换机收到整个帧后转发，大多数采用该模式，也是依照 CSMA/CD 来转发
- 直通 (cut through)：收到帧的硬件地址后立即转发它；如果输出 (outgoing port) 忙，则会转为存储转发。问题是容易出现碎片（转发了一部分数据后，发现冲突）
- 无碎片 (fragment free)：交换机不用收到整个帧而是收到**64B**（冲突窗口，最小帧保证，这里可参考之前 3.2.2 以太网最短帧）后立即转发（意味着之后也不会出现冲突了）
- 适应性交换 (adaptive switching)：自动在上面三种方式中选择

交换机的工作模式

- 全双工模式：因为没有冲突，CSMA/CD 算法可以被关闭

- 自动翻转 (auto-MDIX)：大部分交换机可以自动选择连接方式，交叉线或直通线



- 自适应 (autonegotiation)：两个站点周期性使用快速链路脉冲 (fast link pulse,FLP)，选择 10M/100M/1000Mbps 自适应

集线器、交换机、路由器的区别如下

- 集线器 (hub)：物理层/一层，广播，排队，冲突，共享型设备（一个端口往另一个端口发数据，其他端口就处于等待状态，全部端口属于一个冲突域），半双工，监听，响应
- 交换机 (switch)：数据链路层/二层，MAC 地址，建立连接，独享信道，全双工，增加冲突域数量，减少冲突范围大小
- 路由器 (router)：网络层/三层，建立路由表，IP 地址，路由选择

路由器能连接不同类型的网络，如以太网、ATM 网、FDDI 网、令牌环网等，并实现帧类型的转换，但集线器和交换机一般只用于连接以太网。

小范围的局域网，如我们的校园网大多采用交换机，路由器少。

注意交换机相当于透明网桥，故路由器不会知道，路由器只知道下一跳。

4 网络层

每个数据链路层协议只涉及一个直连网，而网络层协议涉及整个网络。

网络层协议负责确定把收到的包从哪条路径转发 (forwarding) 出去，即路由选择(routing) 功能。具体的传送则由数据链路层和物理层负责。

4.1 IP 数据报

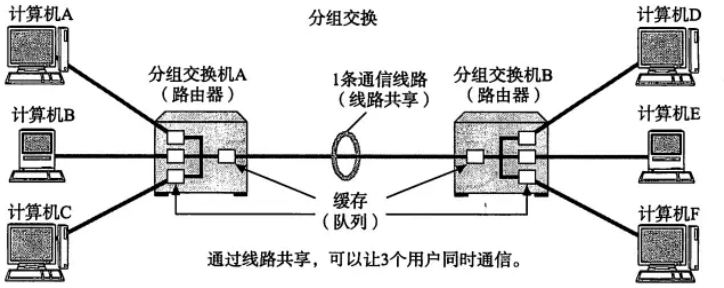
4.1.1 一般网络的服务模型

异步传输模式 (Asynchronous Transfer Mode, ATM)

网络结构	服务模型	带宽	不丢包	有序	及时	拥塞反馈
ATM	恒定位速率	固定速率	是	是	是	无拥塞
ATM	可变位速率	确保速率	是	是	是	无拥塞
ATM	可用位速率	最小保证	否	是	否	是
ATM	未指定位速率	无	否	是	否	否
因特网	尽力服务	无	否	否	否	否

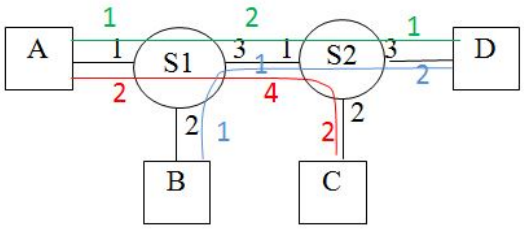
4.1.2 数据交换技术

- 电路交换 (circuit switching)：连接多条物理电路（可以是链路或者链路通过 FDM、TDM 形成的通道）形成一条通路，如电话（时分多路复用）、电视（频分多路复用）。要先建立传输通道，一直占用，不管有无数据交互，直到通道拆除
- 包交换/分组交换[26](packet switching)：统计多路复用，按需分配；可能引起网络拥塞，适合发送突发数据
  - 虚电路 (virtual circuit)：需建立连接才可以传输数据（仿照电话系统，恒变位速率 ATM，因特网之前），好处在于保留带宽
    - 交换式：要传数据时才建立连接，传完则释放；建立虚电路 (VC) 表（输入/输出端口，输入/输出 VCI），虚电路标识符 (Virtual Circuits Identifier, VCI）处于数据帧的内部，数据帧从输入 VCI 到输出 VCI 的过程中内部 VCI 也要变
    - 永久式：建立后一直保持，由管理员维护
  - 数据报 (datagram)：不需建立连接，不可靠，因特网使用数据包交换技术，不预留带宽



例 12. 下图存在 3 条虚电路（红绿蓝），它们都是从 A 或者 B 出发的虚电路，请填写它们的虚电路表。接口编号用黑色字表示。

[26]所以 IP 数据报也被称为 IP 分组 (packet)



分析. 交换机 *S1* 的虚电路表

	输入接口	输入 *VCI*	输出接口	输出 *VCI*
红	*1*	*2*	*3*	*4*
绿	*1*	*1*	*3*	*2*
蓝	*2*	*1*	*3*	*1*

交换机 *S2* 的虚电路表

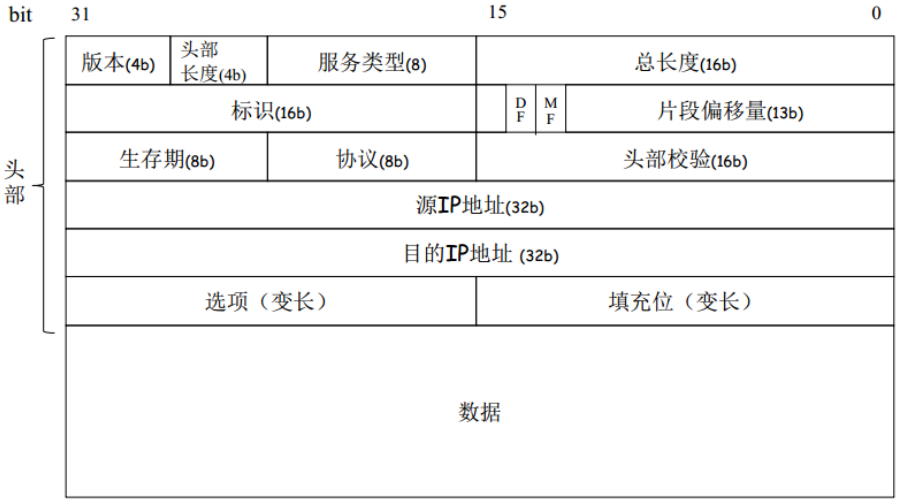
	输入接口	输入 *VCI*	输出接口	输出 *VCI*
红	*1*	*4*	*2*	*2*
绿	*1*	*2*	*3*	*1*
蓝	*1*	*1*	*3*	*2*

IP 协议是因特网的网络层协议

- 可路由的(routable)：全局地址，按层分配
- 尽力服务(best effort)：不可靠的无连接无确认的数据报服务
- IP 协议可以运行在任何物理网络上，不仅仅是因特网

注意：IP 协议不具有拥塞控制机制

4.1.3 数据报格式



- 版本：共两个版本，IPv4 为 4，IPv6 为 6
- 头部长度：以**字 (32b)**为单位
- 服务类型 (Type of Service, ToS)：现在重新定义为**区分服务**
- 总长度：整个数据报的长度，以**字节**为单位
- 标识 (identification)、标志 (DF/Dont fragment、MF/More fragment)、偏移量：用于划分片段
- 生存期 (Time-to-live, TTL)：
  - 防止数据报长时间滞留在因特网上，实际限制为经过的路由器数目，即跳数 (hop count)，每经过一个路由器减 1，超过则自动清除，防止兜圈，同时发送 ICMP 包告知源主机
  - TTL 初值默认设置为**网络直径（所有点对的最短距离中最大值）的两倍**，Windows 默认 64，Unix 默认为 255
- 协议：定义数据部分的协议TCP 为 6，UDP 为 17，ICMP 为 1，IGMP 为 2
- 头部校验：先将**校验和的字段**置为 0，然后对 IP 包头进行**校验和**而非 CRC 运算。路由器会丢弃出错的数据报。
- 选项和填充位：两项均可变长，两项之和最高 40 个字节，填充位用于 32 位对齐

选项的格式：1B 代码 + 1B 总长度 + nB 数据

代码	名称	描述
0	选项列表结束	一个字节：0x00，用于最后选项 4 字节对齐
1	无操作	一个字节：0x01，用于中间选项 4 字节对齐
7	记录路由	记录下每个转发路由器的 IP 地址
131	松散源路由	指明一系列必须经过的路由器
137	严格源路由	指明一系列必须且只能经过的路由器
50	记录时间戳	每个转发的路由器都记录下自己的 IP 地址和当时的时间

  - 头部长度只有 4b，以字为单位，头部最多 $(2^4-1)\times 4 = 60$B，除去非选项部分 $4\times 5 = 20$B，IPv4 选项最多 40B，太少了
  - 记录路由：记录下每个转发路由器的 IP 地址，代码为 7。代码和长度后面跟 1B 指针，然后每个 IP 地址 4B；指针指向下一个 IP 地址的位置，4 为空，40 为满，最多记录 9 个；每经过一个路由器就会记录转出接口的 IP 地址
- IP 数据报一定要**封装成帧**，在不同物理网络（直连网）间传送时，每次都要修改源 MAC 和目的 MAC 地址，但 IP 地址不要变。在**以太网帧的类型/长度字段填 *0x0800***表明是 IP 数据报

4.1.4 IP 数据报的分段和重组

- 一个物理网络的最大传输单元 (maximum transmission unit, MTU) 是该网络可以运载的最大有效载荷，即**数据帧 数据部分**的最大长度
  如：以太网 (DIXv2) 的 MTU 为 1500, FDDI 和令牌环的 MTU 分别为 4353 和 4482

- 如果一个数据报的大小大于要承载它的网络的 MTU，路由器需要先对该数据报进行分段 (fragment)（将数据部分拆开，每个子数据部分加上头部）
- 源主机每次发送 IP 数据报时都会把标识字段加 1。分段时标识的值保持不变，用偏移量字段 (offset) 指出该片段的数据部分相对原来数据报数据部分的偏移量（以8 字节为单位）
- 当目的主机收到该数据报的所有片段时，它会重组 (reassemble) 为原来的数据报
- 第一个片段到达目的主机时目的主机会启动一个重组定时器（默认超时值为 15 秒）。如果该定时器到期时没有收集到所有片段，目的主机会放弃本次重组并丢弃该数据报的所有片段。
- 分段后MF、偏移量、头部校验（检验和）和总长度 会变。不论分不分段，只要接收到数据包，TTL 就会变，接着头部校验（检验和）也会变
  - 当没有 MF 时，假设分段中的最后一段丢失了，重组时能发现吗？不行，中间段丢失可以通过 OFFSET 察觉，但是最后一段丢了就没办法了，因此引入 MF；对于一个数据包多次分段，也仅会有一个子数据包的 MF 等于 0
  - 假设有些主机无法完成重组的任务，怎么办？引入 DF
- IPv6 中间不能分段

例 13. *一个没有选项的 IP 数据报的总长度为 3000 字节，标识是 10034，DF=0，OFFSET=0，要转发到 MTU 为 800 的一个物理网络上。如果前面的片段尽量大，如何划分片段？如果第二个片段在后面的路由器上要转发到 MTU=300 的物理网络上，要继续划分片段，则应该如何划分？*

分析. *要减去 IP 数据报头部长度，* $\lfloor(800-20)/8\rfloor = 97$，*实际载荷* $97\times 8 = 776B$，*故分为 4 段，* $3000 = 776+776+776+672$。由数据报总长度来确定数据部分边界。

	标识	偏移量	MF
片段 1	10034	0	1
片段 2	10034	97	1
片段 3	10034	194	1
片段 4	10034	291	0

第二个片段长度 776，偏移从 97 开始， $\lfloor(300-20)/8\rfloor = 35$，*实际载荷* $35*8 = 280B$，*故分为 3 段，* $776 = 280+280+216$

	标识	偏移量	MF
片段 1	10034	97	1
片段 2	10034	132	1
片段 3	10034	167	1

例 14 (路径 MTU 发现/Path MTU Discovery)**.** *当一台主机要向远方的另一台主机发送很多数据报。如果它希望这些数据报中途不要分段以节约路由器的时间，这就要找到路径上最小的 MTU，有何方法？假设这段时间该路径不会改变。*

分析. *Ping 远端主机，每个数据报的 DF 均设置为 1（即不允许分段），使 ICMP 有效载荷的字节数从大到小变化，直到得到响应。最终直到 MTU 足够小以至于走完整条路径都不需要分段，此时获得的 MTU 就是路径最小 MTU。*

4.2 IP 地址

48 位的 MAC 地址和 32 位的 IP 地址都是全局的（全球分配），但是 IP 地址空间**分层**（即划分为几个部分），是**可路由的**（分层避免路由表太大）。IP 地址属于**接口/网卡**(Network interface card, NIC)。主机或路由器的每个接口可以配置一个或多个 IP 地址。

4.2.1 有类网

IPv4 地址占 32 位，以点分十进制方式表示，每 8 位占一格，如 192.168.1.1。
IP 地址可划分为两个部分：

- 网络号/网络前缀 (prefix)/网络标识 (ID)：确定拥有该 IP 地址的主机位于哪个网络
- 主机号(host identifier)：确定属于该网络的哪台主机

网络类别	IP 地址范围	网络号开头标识	说明
A 类	0 ~ 127	0	主机号占 3B，单播
B 类	128 ~ 191	10	主机号占 2B，单播
C 类	192 ~ 223	110	主机号占 1B，单播
D 类	224 ~ 239	1110	多播地址
E 类	240 ~ 255	1111	保留

注意：ABC 类网网络号全 0 或全 1 不可用，故可用网络地址数要减 2，如一个 C 类网可用的 IP 地址只有 $2^8 - 2 = 254$ 个

4.2.2 无类网

IPv4 地址不够用的解决方案

- 将一个有类网可以划分为多个相同大小的子网 (subnet)
  - 用子网掩码 (subnet mask) 划分边界：主机号全 0，剩下的部分（网络号和子网号）全是 1
  - 子网掩码与 IP 地址**相与**，若相等则在同个子网中
  - **主机号全 1 或全 0**的地址被保留，不能使用；子网号为全 0 或全 1 的子网现在都可以使用（以前规定不能使用）
- 变长子网掩码 (Variable-length subnet mask, VLSM)：允许把一个有类网划分为多个不同大小的子网，用长度来表示子网掩码，如/26 代表 255.255.255.192
- 无类域间路由选择协议 (classless inter-domain routing, CIDR)：将多个有类网合并为一个更大的网络，称为超网 (supernet)，其子网号就是所有有类网的网络号中最长的公共部分（通过所有网络号相与）；可以显著减少路由表中路由的数量，称为路由聚合(route aggregation)。
  - 例如，把有类网 192.24.8.0 192.24.15.0 合并为网络号为 192.24.8.0、子网掩码为 255.255.248.0 的超网：从 1111 1000 到 1111 1111，公共部分为 1111 1，因此子网掩码为

$$255 - 7$$

- 网络地址转换 (network address translation, NAT)：将内部地址映射为外部地址的技术，从而缓解 IPV4 地址耗尽的问题，例如将私有地址映射为全局地址；NAPT/PAT/过载 NAT 将端口号也加入 NAT 的映射中。当数据包从内部网络发向外部网络时，出口路由器修改源地址，将内部地址转换为外部地址；当数据包从外部网络发向内部网络时，出口路由器修改目的地址，将外部地址转换为内部地址；出口路由器在内网数据报发往外网时自动把内网地址映射为外网地址的方法称为动态 NAT。动态 NAT 自动转换，但每个动态映射都关联一个 TTL，若没使用会被出口路由器删除；静态 NAT 直接由管理员加入映射，不会被自动删除

  - NAT 省 IP 地址吗？可以，但是是建立在不是所有的内部网络的节点同时和外部网络通讯的基础上的；当所有的内部网络节点同时和外部网络通讯，那么每一个内部网络的节点的 IP 地址就会和一个外部网络 IP 地址相关联，这样并没有节省地址
  - NAPT 省多少地址？6W。省地址的原因在于端口号的加入，出口路由器分配的一个外部网络的地址的端口号是自由选择的（未被占用），端口号是 2B，因此相当于省了 $2^{16}-1=65526$ 个 IP 地址
  - 随着 IPV6 的使用，NAT 用处不大了（因为 IPV6 地址 128 位，远远高于 IPV4 的 32 位，地址耗尽的问题解决了）

例 15. *一个 C 类网 192.1.2.0 划分为 6 个子网，它们分别需要配置 2、2、2、2、50、50 个接口的 IP 地址。如果要求消耗最少的 IP 地址，请采用点分十进制 (dotted decimal) 格式 (a.b.c.d) 写出它们的子网号和子网掩码*

分析. *由于主机号全 1 或全 0 的地址被保留，故 2 个接口的子网也要配到 4 个*

子网号	末字节	子网掩码	IP 地址数
192.1.2.0	0000	255.255.255.252	4
192.1.2.4	0100	255.255.255.252	4
192.1.2.8	1000	255.255.255.252	4
192.1.2.12	1100	255.255.255.252	4
192.1.2.64	01000000	255.255.255.192	64
192.1.2.128	10000000	255.255.255.192	64

非军事化区 (Demilitarized Zone, DMZ) 是位于内部网络和外部网络之间并为双方提供因特网服务的区域。

- 内网主机可以访问内网主机、DMZ 和因特网。
- 内网主机可以使用内部地址或全局地址 访问 DMZ 的服务器。
- 外部主机只能通过全局地址 访问 DMZ 的服务器，不能访问内网主机

4.2.3 特殊 IP 地址

- 0.0.0.0：未知或秘密 IP 地址，只用作源地址

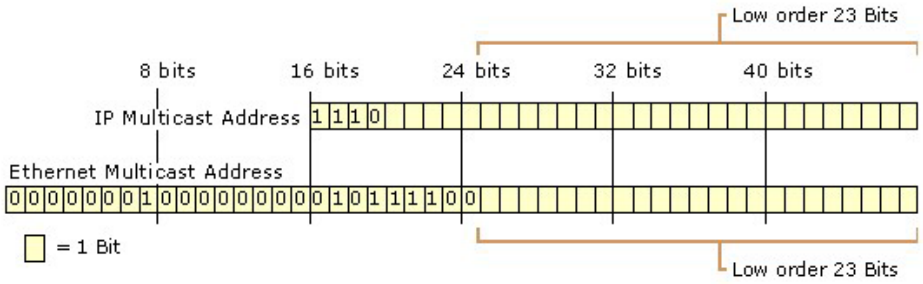
- 00···00+Host：同一子网的主机，只用作源地址
- 255.255.255.255：有限广播，对于一个**直连物理网络的广播**
- Network+11···11：对于一个**远程网络的广播**（如 192.168.1.255）
- Network+00···00：用 32b 表示的网络号（含子网号）
- 127.X.X.X：**环回 (loopback)/本机地址**，127.0.0.1 为**本地地址**(localhost)
- 224.0.0.0 - 239.255.255.255：IPv4 的**多播地址空间**（D 类网）
- 私有 IP 地址：无需 IANA 分配，任何人都可使用
  * A 类：10.0.0.0 - 10.255.255.255
  * B 类：172.16.0.0 - 172.31.255.255
  * C 类：192.168.0.0 - 192.168.255.255

  私有地址只能用于内部网络。主干网上的路由器会过滤掉目的地址为私有地址的 IP 数据报。因此，离开内部网络的 IP 数据报必须使用由 IANA 分配的全局地址作为目的地址。

4.3 IP 数据报相关协议

4.3.1 地址解析协议

- IP 单播地址：通过 ARP 协议获得 MAC 单播/网卡/烧录地址（全球唯一，每个网卡/接口一个）
- IP 广播地址 (4B 全 1)：转为 6B 全 1 的 MAC 广播地址
- IP 多播地址 (1110 开头)：MAC 的高 25 位为`0x01-00-5E`，低 23 位为 IP 地址的低 23 位



如 224.0.1.5 写为 16 进制是`0xE0-00-01-05`,取低 23 位,得到 MAC 完整地址为`0x01-00-5E-00-01-05`

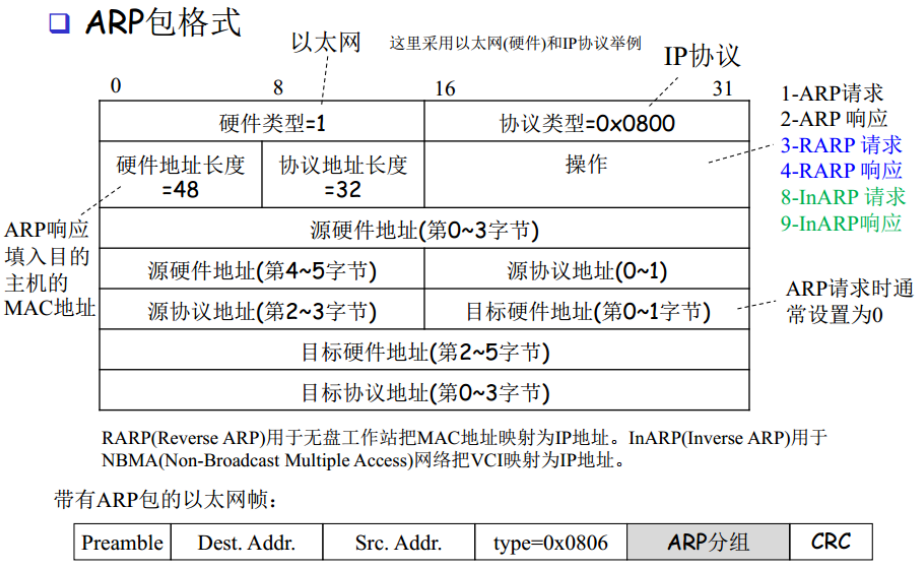
地址解析协议 (address resolution protocol, ARP) 可以将 IP 地址映射为 MAC 地址

- 在一个直连网中，源主机发 ARP 请求**广播帧**（谁的 IP 地址是 XXX），目的主机回复 ARP 响应**单播帧**（返回 MAC 地址），IP 地址与 MAC 地址的端口号相同
- **没有超时重传机制**，超时没有收到响应则丢弃引发 ARP 查询的 IP 分组
- 源主机获得的映射结果缓存在 ARP 表中，TTL 一般为 2 到 20 分钟；缓存的形式：

$$\langle \text{IP address}, \text{MAC address}, \text{TTL} \rangle$$

当收到 ARP 请求，目的主机会缓存源主机的映射，其他监听到的主机如果已缓存该映射，则会重置 TTL

- 也可直接将映射加入 ARP 缓存，称为静态 ARP 映射，不会因超时而删除
- ARP 请求包中源硬件地址和协议地址、目标协议地址都知道，但目的硬件地址 (MAC)不知
- 任何物理网络都要使用 ARP 协议来获得 MAC 地址吗？不对。PPP 协议针对点对点网络，不需要 MAC 地址；广播也不用启动 ARP 协议



可以用 ARP 请求来确定一个 IP 地址在以太网中是否被使用，如果没有响应则说明没有使用。

4.3.2 动态主机配置协议

DHCP 协议 (Dynamic Host Configuration Protocol) 用于主机（可以是手机或电脑）在加入网络时动态租用IP 地址，基于 UDP，四个步骤（四种 DHCP 数据报）如下：

- DHCP 发现 (discover)：
- DHCP 提供 (offer)：还可以指出 DHCP 中转服务器，使多个网络可以共享一个 DHCP 服务器（通过 DHCP discover 的域名选项进行区分）
- DHCP 请求 (request)：从多个响应的服务器中选择一个，并通告其它服务器已拒绝了它们的 offer
- DHCP 确认 (ACK)

因为客户端还没有 IP 地址，因此在以太网中上面四个步骤都是广播帧

4.3.3 因特网控制消息协议

因特网控制消息协议 (Internet Control Message Protocol, ICMP) 用于主机或路由器发布网络级别的控制消息，主要是出错/丢包后将信息发回给源主机，如

- 回响请求和答复消息/因特网包探索器 (Packet Internet Grouper, ping)（类型 8）：主机不可达 是去的路找不到，中途原路返回

- 不可达消息（类型 3）：源路由错误（IP 源路由选项出错）、需要分段但不可分段 (DF=1)、网络（目的地址为私有地址，路由表出错等）/主机（不能找到到目的网络的路由）/协议（上层协议不存在或者未运行）/端口（UDP 端口号没有绑定进程）不可达
- 源端抑制（类型 4）：控制源主机发送速度
- 重定位消息（类型 5）
- 时间超时消息（类型 11）：TTL 减到 0、数据报重组超时
- 参数问题（类型 12）：坏的 IP 头部、缺少必要选项

例 16. 主机和路由器通过三个以太网连接：*[H1]-N1-[R1]-N2-[R2]-N3-[H2]*。主机和路由器的每个接口的 *IP* 地址都配置正确。如果除了一种配置其他配置都是正确的，问导致以下问题的原因，并给出 *ping* 返回的结果？可选项：

A. R1 没有配置 *N3* 的静态路由指向 *R2*

B. R2 没有配置 *N1* 的静态路由指向 *R1*

C. H1 没有配置默认路由指向 *R1*

D. H2 没有配置默认路由指向 *R2*

分析. 连上网后邻近的路由表端口就会被自动添加入本机的路由表，而且处于“在链路上”/直连的状态，故在路由表中查到 *R1* 左侧端口的路由表项，发现是直连网，会直接封装成帧发送出去，到达后路由器通过同样的物理网络发送 *ICMP* 包回来，故 *H1 ping R1* 左侧端口必然可以 *ping* 通（除非本机的 *IP* 协议出现故障）。

1. *H1* 可以 *ping* 通 *R1* 左边接口的 *IP* 地址但是 *ping* 不通 *R1* 右边接口的 *IP* 地址：*C*，*ping* 返回主机不可达。由于 *H1* 没有配置默认路由，故 *ping R1* 右侧端口将直接在本机路由表项中找不到，进而丢弃，在主机出口处就直接返回不可达消息。
2. *H1* 可以 *ping* 通 *R1* 右边接口的 *IP* 地址但是 *ping* 不通 *R2* 左边接口的 *IP* 地址：*B*，*ping* 返回超时。*ping* 通 *R1* 右侧接口说明 *H1* 配了默认路由，*ping R1* 右侧接口时匹配上默认路由，故转发给 *R1*，*R1* 又通过查路由表项，发现是自己右端端口，直连网到达，然后返回 *ICMP* 包。正常来讲，在 *R1* 路由表项中会有后续网络的路由表项，*ping R2* 左侧时匹配上并发送给 *R2*。但由于 *R2* 没有配置 *N1* 的静态路由，导致 *ICMP* 数据报无法返回，在 *R2* 路由表中查不到对应路由表项，进而被丢弃，因为没有返回 *H1*，故是超时。
3. *H1* 可以 *ping* 通 *R2* 左边接口的 *IP* 地址但是 *ping* 不通 *R2* 右边接口的 *IP* 地址：*A*，*ping* 返回主机不可达。与 *(1)* 类似的道理。
4. *H1* 可以 *ping* 通 *R2* 右边接口的 *IP* 地址但是 *ping* 不通 *H2* 的 *IP* 地址：*D*，*ping* 返回超时。与 *(2)* 类似的道理

例 17. *ping* 可以在子网中产生一个广播帧，请给出并解释方法。

分析. *ping* 本网一个不存在的 *IP* 地址，因为 *ARP* 映射表中肯定没有，所以会发送 *ARP* 请求。*ARP* 请求就是广播帧。或者直接 *ping* 对本网的广播，例如：*ping 192.168.1.255*，这个 *ICMP* 消息会用广播帧封装。

4.4 路由协议

4.4.1 有类网的路由选择算法

利用数据包中的目的地址得到目的网络号，然后查询路由表(routing table)/转发表 (forwarding table)

- 如果查询的结果为直连网，则下一跳 (next hop) 为空，直接把数据包从查出的接口转发到目的主机
- 否则，如果查询得到下一跳(路由器)，则把数据包转发给下一跳（转发的方法：ARP 协议，修改 MAC 地址，有数据帧的拆开重新封装的过程）
- 如果没有查到任何匹配项，则把数据包转发给默认路由器（也算查到）
- 如果没有设置默认路由，则丢弃该数据包

4.4.2 无类网的路由选择算法

无类网的路由表里有子网掩码

- 匹配方法：目的 IP 地址 & 子网掩码 == 子网号
- 最长匹配原则 (The longest match rule): 当有多条路由都匹配时选择子网掩码最长（1 的长度）的路由
- 从 IP 数据报中获取目的地址，利用目的地址查路由表（同有类网）
  - 没有匹配项：丢弃该分组
  - 有匹配项，没有下一跳，匹配项接口为以太网：直接取**IP 数据报中的目的 IP 地址查询 MAC 地址**（已经到达了目标主机所在的直连网），封装成帧后从查出的接口处发送
  - 有匹配项，下一跳接口为以太网：从路由表中查出下一跳的 IP 地址，通过**ARP 协议**获得**目的 MAC 地址**，封装成帧后从查出的接口处发送，要遵守**以太网协议 (CSMA/CD)**
  - 有匹配项，下一跳接口为直连网 (PPP)（或没有下一跳，匹配项接口为 PPP）：将数据报直接封装成帧后从查出的接口处发送，不需要目的地址（放到物理网络中自然会到达）
- 每到一个路由器都将帧拆出来，再重新封装，目的和源 **MAC 地址**全要发生变化（ARP 协议取下一跳/目的地址的 IP 地址填入，来获取下一跳的 MAC 地址进行封装）。但注意路由器只是将帧拆出来，里层的**IP 数据报并不会改变**。

注意：路由表中每一行的下一跳和接口必然处于同一物理网络中

例 18. 给定目的地址求下一跳

网络号 (Network Destination)	子网掩码 (Subnet mask)	下一跳点 (Next Hop)	接口 (Interface)	开销 (Metric)
128.96.39.0	255.255.255.128	128.96.39.1	128.96.39.1	1
128.96.39.128	255.255.255.128	128.96.39.131	128.96.39.131	1
128.96.40.0	255.255.255.128	199.1.3.1	199.1.3.2	4
199.1.3.0	255.255.255.240	199.1.3.2	199.1.3.2	1
192.4.153.0	255.255.255.192	128.96.39.212	128.96.39.131	5
0.0.0.0	0.0.0.0	128.96.39.198	128.96.39.131	1

分析. 结果如下

目的地址	下一跳
192.4.153.17	*128.96.39.212*
128.96.39.10	*128.96.39.1*
128.96.40.12	*199.1.3.1*
128.96.40.151	*128.96.39.198*
192.4.153.90	*128.96.39.198*

其中最后两项匹配未成功，因此下一跳是默认路由。例如 192.4.153.90&255.255.255.192 = 192.4.153.64 ≠ 192.4.153.0

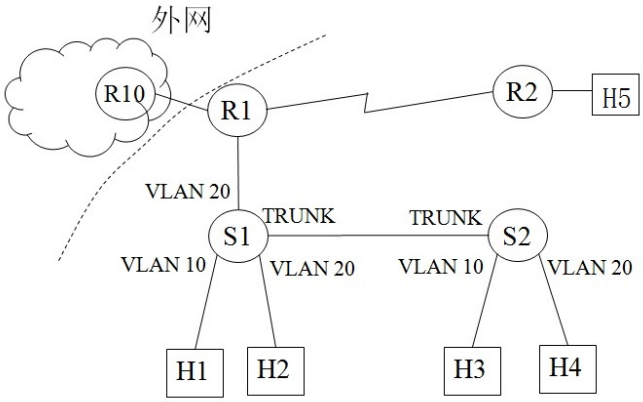
例 19 (实战分析). “在链路上”意味着是直连网，“网关”相当于下一跳，“接口”直接用 ***IP*** 地址标注

分析. 网卡接上网络后，将会自动添加路由表项（直连网），一般也会将默认网关设好，即下一跳指向最近的路由器。

1. *0.0.0.0/0*：默认路由，即其他项匹配不上的都会被送至 *172.19.108.182* 的接口，然后发送到 *172.19.127.254*（默认网关，校园网私有地址，这是 *WiFi* 协议）
2. *127.0.0.0/8*：环回网络，发给自己
3. *127.0.0.1/32*：环回网络，本地地址（内部地址，其他是外部地址都需要经过网络层防火墙）

4. *127.255.255.255/32*：对 *127* 这个网络（也就是自己）广播
5. *172.19.64.0/18*：校园网内部网络，通过默认网关发出
6. *192.168.56.1/32*：*VirtualBox* 虚拟机网卡接口 *IP*
7. *224.0.0.0/4*：从跃点数较小 *(281)* 的端口 *192.168.56.1* 发出去，在无线网络中多播
8. *255.255.255.255/32*：在无线网络中广播；这里接口的 *IP* 地址可以为 *127.0.0.1*（表示数据包直接进入主机，未过防火墙）或者 *192.168.56.1*（这是主机自身的 *ip* 地址，可通过 *ipconfig* 查看，经过防火墙进入主机）

例 20 (综合题). 在下图中，*R1* 和 *R2* 为路由器，*S1* 为二层交换机，*S2* 为三层交换机并配置了 *VLAN 10* 和 *VLAN 20* 的虚接口。*R1* 到 *R2* 为一个配置了 *IP* 地址并使用 *PPP* 协议的点到点网络，其它四个 *(VLAN 10, VLAN 20, R1-R10, R2-H5)* 子网都是以太网。如果所有主机、三层交换机和路由器都正确配置了接口的 *IP* 地址，三层交换机和路由器都启动了 *OSPF* 协议，*R1* 的默认路由指向 *R10* 并被发布到 *OSPF* 协议，问：*H1 ping H3*、*H1 ping H4*、*H1 ping H5* 时依次经过了哪些设备（主机和路由器）以及它们分别使用了以下哪种协议？



可选项：

1. *802.1* 透明网桥算法（带 *VLAN*）
2. *802.1Q* 协议（*trunk*）
3. 以太网协议
4. *ARP* 协议（*IP* 地址为下一跳）
5. *ARP* 协议（*IP* 地址为 *IP* 分组的目的地址）
6. 查询路由表
7. *PPP* 协议
8. 从收到的帧中取出 *IP* 分组
9. 网络层从上层收到数据并封装为 *IP* 分组

A. 匹配了默认路由

B. NAT

分析. 针对不同的物理设备，有以下流程

- **主机端**都是从上层收到数据，然后查询路由表，网络层封装为 *IP* 数据报，下去数据链路层封装成帧（需要用 *ARP* 协议 查出对应的 *MAC* 地址），然后遵循以太网协议 通过物理层传输。
- **交换机**遵循以太网协议 获取数据帧，然后依据透明网桥算法 扩散或转发。由于涉及虚拟局域网，发到干道上时要遵循 *VLAN* 干道协议，添加 *VLAN ID*。同样遵循以太网协议 继续通过物理层传输。
- **路由器**遵循以太网协议 获取数据帧，向上传递，取出 *IP* 分组，查询路由表，如果没有下一跳，则用 *ARP* 协议 查 *IP* 分组目的地址的 *MAC* 地址；有下一跳则用 *ARP* 协议 查下一跳的 *MAC* 地址，然后重新封装成帧遵循以太网协议 传输。如果是直连网，则查完路由表可直接依照 *PPP* 协议 发送。
- 最终数据帧被**接收端**接收，以太网协议 获取数据帧，取出 *IP* 分组，向上传递。
- 转发数据包或 *ARP* 协议时，只要发送出去，都要封装成帧。如果转出去的接口是以太网，则要封装成 *802.3* 的帧，并用 *CSMA/CD* 从物理层发送出去。

所以针对以下几种情况，可以得到对应使用的协议

- *H1 ping H3 → H1:9653 S1:3123 S2:3213 H3:38*
  由于在同一 *VLAN*，且不涉及路由，**故 *ARP* 协议取的 *MAC* 地址直接是目的主机的 *MAC* 地址**。
- *H1 ping H4 → H1:9643 S1:3123 S2:328653 H4:38*
  跨两个 *VLAN* 需要三层交换机虚接口的协助，故 *S2* 的作用等同于路由器
- *H1 ping H5 → H1:9643 S1:3123 S2:3286423 S1:3213 R1:3867 R2:78653 H5:38*
  由于 *S1* 是交换机只遵循透明网桥算法，故收到 *VLAN 10* 发来的帧，查不到目的 *MAC* 地址，只会扩散，通过干道端口传输到 *S2*。而 *S2* 作为三层交换机/路由器，重新将其拆封打包贴上 *VLAN 20* 的标签，发回给 *S1*。这时 *S1* 才可以将数据帧转发给 *R1*，然后再执行后续的操作。
- *H1 ping* 外网 *→ H1:9643 S1:3123 S2:3286A423 S1:3213 R1:386AB43 R10:386A...*
  由于内部路由不会有外部地址的路由项，故都会匹配上默认路由 *(R1)*，通过转发数据帧给 *R1*，再由 *R1* 进行 *NAT* 转换，实现内部地址与外部地址的通信。

4.4.3 路由表的建立

路由器在收到一个数据报之后用其目的地址查找路由表 (routing tables) 得到下一跳，再把该数据报转发给下一跳[27]

路由表可以由管理员手工建立，也可以由路由/路由选择协议 (routing protocols) 自动建立，建路由表是即记**最短路径**的下一跳。所建立的路由分别称为**静态路由**和**动态路由**。

路由协议即自动建立路由表，包括网络号、子网号、下一跳、接口、开销等。默认路由和直连路由都是静态路由。

整个因特网实际上由很多机构进行管理。每个机构管理自己的网络，它们有权决定采用什么协议和网络控制策略。这样在**同一个机构**管理下的**网络**称为一个**自治系统**(autonomous systems, AS)。因特网实际上是由很多自治系统构成的。

[27] 虽然同样是转发算法，但和运行在数据链路层透明网桥不同（只要看到局域网内部），路由算法在网络层，要看到整个局域网连接起来的网络

- 用于在 AS内部(Intra-AS) 建立动态路由的路由协议称为内部网关协议(Interior Gateway Protocols, IGP)。例如，RIP 协议 和OSPF 协议。一个 AS 通常运行单一 IGP。
- 用于在 AS之间(Inter-AS) 建立动态路由的路由协议称为外部网关协议(Exterior Gateway Protocol, EGP)。例如，BGP 协议。
- 运行同一个 IGP 协议的连通区域也称为路由选择域 (routing domain)。一个 AS 可以运行多个 IGP 协议，形成多个路由选择域。因此，一个 AS 内的每个路由器都要选择开启哪种 IGP，而不能只有一个路由器开启（没有扩散 IGP 协议这种操作）

路由算法 (Routing algorithm)：路由协议里用的算法，由于两个路由器之间都有开销，可以建立一个图，找最短路径。有以下两种算法：

- 距离向量 (distance vector, DV)：BellmanFord → RIP
- 链路状态 (link state, LS)：Dijstra → OSPF

4.5 内部网关协议

4.5.1 RIP 协议

路由信息协议 (Route Information Protocol, RIP)：距离向量算法 (Distance vector algorithm) 的路由协议（问路），工作原理是采用邻居的路由表构造自己的路由表。

距离向量算法的思想是：每个节点周期性地将自己的距离向量估计发送给邻居，节点 x 收到所有邻居的距离向量估计生成自身的距离向量估计 $D_x(y) = \min_v\{c(x,v) + D_v(y)\}, \forall v \in Neighbor(x)$，表示节点 x 到节点 y 的距离估计。算法会收敛于实际的最短路径开销

RIP 协议用的算法和原始的距离向量算法有细微差别，节点不是一次收到全部邻居的距离向量估计，因此更新方法为每次将自身距离向量估计和收到的取最小值（排除自身距离向量估计不准的情况）

- 每 30 秒RIP 路由器把它的整个路由表发送给邻居。具体实现时每个邻居会错开发送，30 秒的时间也会随机变化一点。
- 初始时每个 RIP 路由器只有到直连网的路由，它们的距离为 1。
- 到目的网络的距离以跳为单位。最大距离为 15。距离 16 表示无穷大，即目的网络不可达。

具体算法:当收到邻居发来的路由表 (update packet),路由器将更新它的路由表,⟨目的网络, 开销, 下一跳⟩

1. 收到路由的距离全部加 1（即一跳的距离）
2. 利用上述路由修改路由表:
   - 把路由表中不存在的路由 加入路由表
   - 如果比路由表中的路由的距离更小，则 (a) 更新该路由的距离为新距离，(b) 把下一跳改为邻居
   - 如果路由已经存在（指目的网络相同）且原本路由的下一跳就是发送路由的邻居，则必须进行更新[28]

[28]即使变大了，因为如果本节点到该目的网络的路由下一跳是邻居，那么当邻居到目的网络的距离变了后，说明本节点到该目的网络的路由现在是错误的，必须更新

例 21. 路由器 *A-G* 运行 *RIP* 协议，每跳的距离为 *1*。*B* 和 *C* 是邻居。如果 *B* 和 *C* 此时的路由表如下所示：

路由器 *B* 目的网络	距离	下一跳	路由器 *C* 目的网络	距离	下一跳
N1	*5*	*A*	*N1*	*3*	*F*
N3	*3*	*C*	*N3*	*6*	*F*
N6	*2*	*E*	*N6*	*3*	*B*
N9	*5*	*D*	*N7*	*3*	*G*
N10	*1*	-	*N9*	*5*	*G*
			N10	*2*	*B*

当路由器 *B* 接收到来自 *C* 的路由表之后对路由表进行自己的更新，请写出更新之后 *B* 的路由表

分析. 路由器 *B* 路由表更新后的结果为

目的网络	距离	下一跳
N1	*4*	*C*
N3	*7*	*C*
N6	*2*	*E*
N7	*4*	*C*
N9	*5*	*D*
N10	*1*	-

其中第二行：路由器 *B* 的目的网络为 *N3* 时下一跳为 *C*，又收到 *C* 的关于 *N3* 的路由，因此强制更新 *N3* 这一项

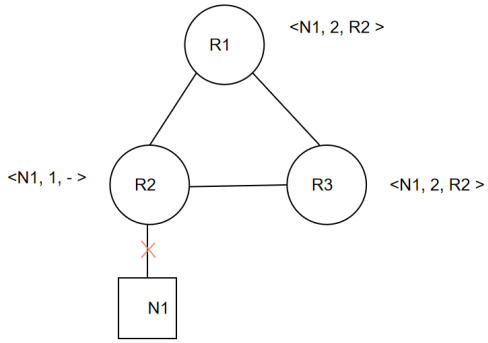
记得路由项更新后下一跳为 *C*

RIP 协议存在的问题

- 慢收敛：假设 m 个路由器连成线，这条线的一端新增了一个网络，那么关于这个网络的路由要传递到另一端速度很慢。最短时间接近 0（看更新时刻），最长时间 $30(m-1)$，平均时间 $15(m-1)$
- 计数到无穷：N1–R1–N2–R2–N3。在 R1 因 N1 失效而把 N1 的路由的距离改为 16（无穷大）（路由项为 $<N1,16,->$）之后，R2 将路由表（$<N1,2,R1>$）发给 R1。R1 就会根据到 N1 的最小距离修改路由项（$<N1,3,R2>$），再发给 R2；R2 收到原本路由项的下一跳就是发送方的路由后，修改为（$<N1,4,R1>$）；这样 R1、R2 到 N1 都是通过对方，因此不断循环直至距离均为 16；这个结果是合理的，但是收敛速度漫且浪费带宽

解决方案：

- 水平分割(split horizon) 技术：从一个接口学来的路由不会从该接口发回去；依然会计数到无穷，解决方案是抑制技术（部分解决了）和触发更新（更高程度，但无法完全解决）



无法解决的示意图如上：

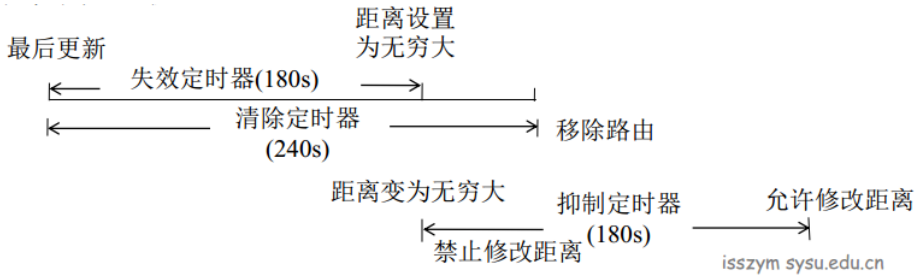
1. N1 失效，R2 修改路由项（$< N1, 16, - >$）
2. 由于水平分割，R1、R3 不会将路由项发给 R2；之后 R2 将路由项发给 R1，R1 修改（$< N1, 16, R2 >$）
3. R1、R3 间没有水平分割的约束（因为都是从 R2 学过来的）可以互发；于是 R3 发给 R1 后 R1 修改（$< N1, 3, R3 >$）
4. 接下来就可以 $R1 \to R2, R2 \to R3, R3 \to R1$ 循环了，满足水平分割协议的同时计数到无穷

- 毒性反转(poison reverse) 技术：从一个接口学来的路由会把距离改为无穷大后从该接口发回去，这叫做带毒性反转[29]的水平分割 (split horizon with poison reverse)。距离为无穷大的路由称为毒化路由 (poisoned route)。
- 抑制技术(hold down)：距离被改为无穷大的路由在一段短时间 (180 秒) 内其距离不允许被修改。
- 触发更新(triggered update)：一旦出现路由变化将立即把变化的路由发送给邻居。原有的 30 秒发一次完整的路由表依然不变（因触发更新可能丢失；路由表也可能出错；要用 TTL 清除无效路由）

RIP 协议的定时器：每个路由器具有更新定时器，路由器的每条路由具有失效、清除、抑制定时器

- 更新定时器 (Update Timer)：控制一个路由器定期把路由表发送给邻居的时间，默认为 30 秒。
- 失效定时器 (Invalid Timer)：加入路由启动；路由更新，就要重置；到期时被标记为无效路由（距离改为 16）。TTL 默认为 180 秒。
- 清除定时器 (Flush Timer)：到期时该路由将从路由表中删除。路由被更新时其清除定时器会被重置，默认为 240 秒。
- 抑制定时器 (Hold-down Timer：在路由的距离变为无穷大（包括收到毒化路由）时启动。在其到期之前不允许修改该路由的距离，默认为 180 秒。

[29]N1–...–A–B，B 根据 A 发来的到目的网络 N1 的路由修改自身路由项后，向 A 发送一个不可达 N1 的消息（B 到达 N1 开销设置为 16），这样 A 就不会试图经过 B 到达 N1，因此就可以避免路由环路的产生。



注意的是，抑制定时器和清除定时器之间是独立的。路由的距离变为无穷大可能是失效定时器到时，也可能是邻居发来的

RIP 数据包格式：RIP 数据包用 UDP 数据报封装 (端口号为 520)

- RIPv1 采用广播方式发送给邻居。RIPv1 只能发布有类网。
- RIPv2 数据包可以采用广播方式或多播方式 (224.0.0.9, 所有 RIPv2 路由器) 发送给邻居。RIPv2 支持无类网（因此多了子网掩码，以及下一跳；RIPv1 中对应位置为 0）。

RIP 协议简单、容易实现。特点如下：

- 网络的直径不能超过 16 跳
- 不允许把一个大网络分成多个区
- 每 30 秒发送完整路由表会消耗大量的带宽
- 存在慢收敛问题和计数到无穷问题
- 实际运行的 RIP 协议具有如下特性：
  - 可以保存多达 6 个等距离的路由在路由表中，默认为 4 个
  - 直连网的管理距离[30]为 0，RIP 协议的距离为 1

4.5.2 OSPF 协议

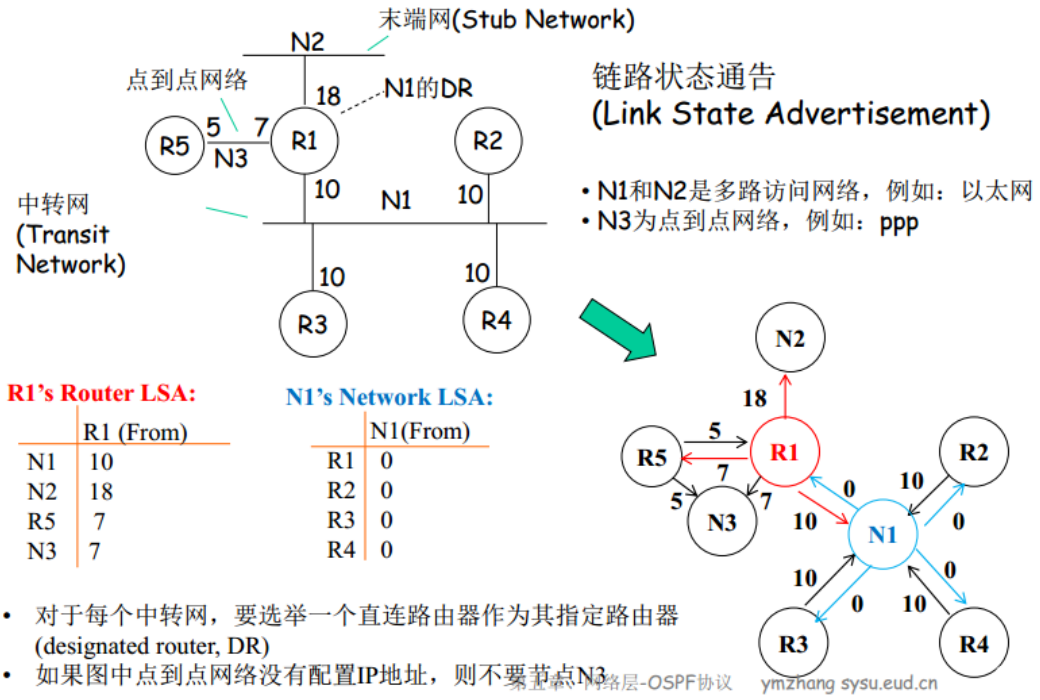
开放最短路径优先协议 (Open Shortest Path First, OSPF) 采用链路状态路由算法，可能是在大型企业中使用最广泛的内部网关协议。

- 周期性地收集链路状态，并扩散给 AS 中的所有路由器
- 用收到的链路状态建立整个 AS 的拓扑结构图
- 利用 Dijkstra 算法计算到 AS 中所有网络的最短路径
- 利用这些路径上的下一跳建立路由表（到了下一跳再查路由表确定下下一跳）

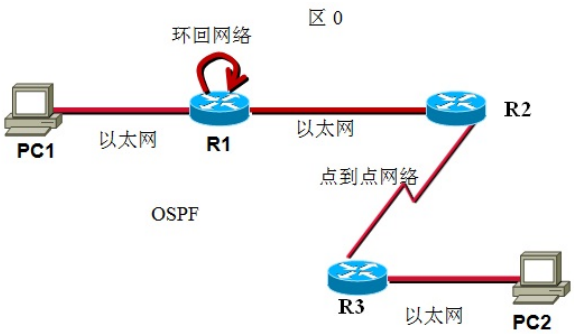
OSPF 第一步需要将整个网络 (AS) 转化为 AS 的拓扑结构图。

[30]管理距离是指路由协议的路由可信度。目的是统一路径开销的衡量标准。RIP 协议是根据路径传递的跳数来决定路径长短也就是传输距离，而像 EIGRP 协议是根据路径传输中的带宽和延迟来决定路径开销从而体现传输距离的。管理距离越小，它的优先级就越高，也就是可信度越高。

- 每一个路由器和每一个网段（多路访问网络/点到点网络）都作为一个结点；如果点到点网络没有配置 IP 地址，则该结点可去除
- 每个中转网(transit network)，要选举一个直连路由器作为其指定路由器 (designated router, DR)
- 中转网只有入边有权，出边都没有
- 末端网(stub network) 即不再连其他路由器的网络，管理员设的
- 每一个路由/中转网都有自己的链路状态通告(Link State Advertisement, LSA)（2 种 LSA）



例 22. 如下图，有 *3* 个 *router LSA*，而只有 *1* 个 *network LSA*（环回网络是末端网）

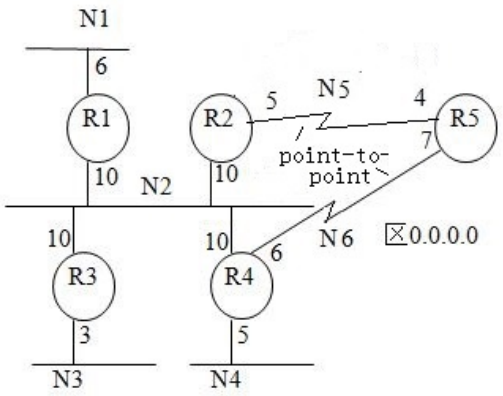


详细过程如下

- 发现邻居：每 10 秒向邻居发送 Hello 分组，如果**40 秒 (dead interval)**都收不到邻居发来的 Hello 分组，则把到邻居的链路标记为失效。多路访问网络采用多播 (224.0.0.5, all OSPF routers) 发送 Hello 分组。一个 Hello 分组包含优先权、已知的邻居（收到过 Hello）、DR 和 BDR。

- 完全相邻：在发现邻居之后，OSPF 路由器将与邻居交换链路状态数据库中的 LSA，请求得到更新的或者没有的 LSA。在与邻居的链路状态数据库变得完全一样时，它们就处于完全相邻状态(fully adjacency)。
- 生成 LSA：每**30 分钟**或**链路变化**时，每个 OSPF 路由器会生成 router LSA，**中转网的 DR**会生成 network LSA
- 扩散 LSA:产生的 LSA 立即封装为 Update 分组，被**可靠**地扩散出去（需要确认）。每次产生的 LSA 的序号会加 1，序列号越大表示越新。若收到多个 LSA，由发出此 LSA 的路由器 ID（发通告路由器）、链路状态 和序列号 唯一确定。第二次收到来自相同的三元组将丢弃它以防止扩散形成回路。
- 收集 LSA：路由器收集到 LSA 之后，用新 LSA 替换链路状态数据库中旧 LSA。如果一个 LSA 在**60 分钟 (max age)**没有被更新，它将从链路状态数据库移除。
- 计算最短路径：当链路状态数据库被改变时，OSPF 路由器将利用 Dijkstra 算法计算到所有网络的最短路径。
- 建立路由表：利用得到的最短路径产生路由表。

例 23. *写出对应的网络和路由 LSA*



分析. *举例如下，中转网 LSA 到其他路由开销为 0!*

R1 LSA	开销	链路类型	*R2 LSA*	开销	链路类型	*N2 LSA*	开销
N1	*6*	末端网	*R5*	*5*	点到点	*R1*	***0***
N2	*10*	中转网	*N5*	*5*	**末端网**	*R2*	*0*
			N2	*10*	中转网	*R3*	*0*
						R4	*0*

下面是 R5 的路由表，中转网只算一次开销（比如 R5 到 N1：4+10+0+6，0 是离开中转网)！总开销直接目测！

目的	开销	下一跳	目的	开销	下一跳
N1	*20*	*R2*	*N4*	*12*	*R4*
N2	*14*	*R2*	*N5*	*4*	-
N3	*17*	*R2*	*N6*	*7*	-

OSPF 协议采用路由器 ID(Router ID, RID) 标识每一个路由器。路由器 ID 由以下方法得到：

- 使用直接配置的 RID(`(conf)#router-id id`)
- 所有活动环回接口中最大的 IP 地址
- 所有活动物理接口中最大的 IP 地址

除非路由器重启、所选接口故障或关闭或 IP 地址改变、重新执行了 router-id 命令，RID 都将保持不变。

指定路由器的选举方法：

- 当多路访问网络重启时，就开始选举 DR。在等待时间结束 (Wait Time/Dead Interval, 40s) 时，带有**最高和次高优先权**的路由器分别成为 DR 和 BDR(Backup DR)。如果优先权相同，RID 更大的成为 DR，次大的成为 BDR。
- 如果路由器不希望参与选举，则应该把**优先权设置为 0**。如果优先权相同，具有**更高 RID**的路由器成为 DR。如果收到的 Hello 列出了 DR（RID 不是 0.0.0.0），路由器成为 DR。
- 如果一个新的路由器在选举之后加入或者有路由器修改为更高的优先权，它也不可能抢占现存的 DR/BDR 和变为 DR/BDR。
- 当 DR 失效时，BDR 成为 DR，将开始一个新的选举过程来选出 BDR。
- 一个多路访问网络中的 OSPF 路由器只与 DR 和 BDR 建立相邻关系。
- 收到一个 LSA 后，一个多路访问网络中的 OSPF 路由器将把它首先**多播**(224.0.0.6) 给 DR 和 BDR，然后 DR 再把它多播 (224.0.0.5) 给所有 OSPF 路由器

LSA 具有多个定时器

- 每**10s**(Hello Interval) 向邻居发送一次 Hello，4 倍的 Hello interval(Dead Interval，40s) 没有收到邻居的 Hello 就认为邻居失效。
- 每**30 分钟**会产生新的 LSA，最小间隔时间为 5s。
- 每个 LSA 都有年龄字段 (age)，发给邻居时被设置为 0，在链路状态数据库中 age 会不断增长，增长到 max age（默认为 60 分钟）时 LSA 被标记为失效。失效的 LSA 会被扩散到整个 AS，令 AS 的所有路由器把该 LSA 从链路状态数据库中移除。
- 存储在链路状态数据库中的 LSA 每 10 分钟会被计算校验和，如果有错将被删除。
- 接收来自邻居的 LSA 的最小间隔时间为 1s。
- 计算最短路径的最小间隔时间为 10s。

OSPF 特点：

- 所有的 OSPF 消息都要认证（防止恶意入侵）。
- 路由表中允许多个**相同开销**的路径存在（RIP 只允许一条路径），可以实现负载均衡。
- 对于每条链路，允许同时有多个 (TOS) 开销。

- 多播 OSPF(MOSPF) 使用与 OSPF 相同的链路状态数据库（思科路由器不支持）。
- 在大型路由选择域中 OSPF 可以分区。
- 比 RIP收敛快而且更安静。
- 实现起来更复杂，需要更多的计算开销。

4.5.3 两种算法比较

	链路状态 (LS, Dijstra)	距离向量 (DV, BellmanFord)
消息复杂性	n 个节点, E 条链路, 要发送 $O(nE)$ 条消息	只在邻居之间交换消息
收敛速度	$O(n^2)$ 算法需要 $O(nE)$ 条消息 可能会震荡	收敛时间变化 可能出现路由循环 计数到无穷问题
健壮性[31]	可能通告不正确的链路开销 每个节点只计算自己的路由表	可能通告不正确的路径开销 每个节点的路由表被其它节点所用 错误会通过网络传播开

4.6 外部网关协议

4.6.1 EGP 协议

外部网关协议 (Exterior Gateway Protocol, EGP) 既是分类也是协议，只能对树型，还要判回路

4.6.2 BGP 协议

边界网关协议 (Border Gateway Protocol, BGP) 可以用于图，随便连

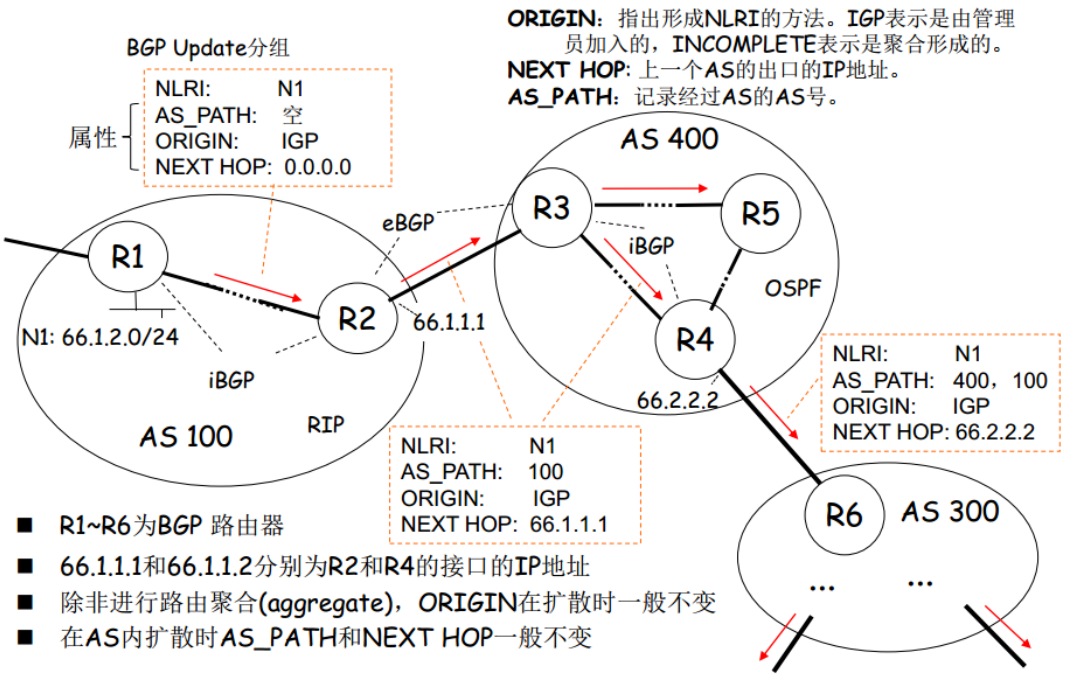
- 采用可靠扩散 (reliable flooding) 的方法把 AS 内的网络的信息传遍整个因特网
- 每个 AS 需要分配一个号码。与 IP 地址分配相同，全局 AS 号 (1 - 64511) 由 ICANN 的下属机构进行统一分配。64512 - 65535 为私有 AS 号。

工作原理

- 运行了 BGP 协议的路由器被称为 BGP 路由器，所运行的 BGP 协议称为 BGP 发言人 (speaker)，其他路由器为内部路由器。
- 在 BGP 路由器之间可以通过**TCP 连接**（端口号为 179）建立相邻关系，由 AS 管理员指定，也可以采用自动产生（重发布）和聚合产生。这些指定网络只有发布路由表中存在才会被改路由器扩散出去。如果它们失效，则会把撤销路由的消息扩散出去。
- AS 内的两个 BGP 路由器之间建立的相邻关系称为iBGP(interior BGP) 相邻关系，而位于不同 AS 的两个 BGP 路由器之间建立的相邻关系称为eBGP(exterior BGP) 相邻关系

[31] 健壮性指路由器失效时会出现的情况

- BGP 协议所扩散的网络前缀/网络号称为网络层可达信息 (Network Layer Reachablility Information, NLRI)，BGP 路由器可以把 NLRI 连同它们的属性一起通过相邻关系扩散给邻居，进而扩散到因特网中所有的 BGP 路由器
- BGP 路由器在 NLRI 引入 BGP 协议时扩散一次，并**不定期扩散**



NLRI 分组发送

- BGP 路由器可以聚合若干 NLRI 网络形成一个新的 NLRI
- 如果从多条路径收到同一个 NLRI，在默认情况下选择**AS-PATH 中 AS 数最少的路径**
- BGP 路由器根据 NLRI 的属性 NEXT HOP 查询 IGP 路由表得到 NEXT HOP，就可以使用该路由。如果没有查询到匹配项，则丢弃该 NLRI
- 如果设置了 IGP 同步并且 NLRI 在 IGP 路由表中没有匹配项，该 NLRI 不能转发给 eBGP 邻居
- BGP 路由器可以把多个路由聚合 (aggregate) 为一个路由，其 NLRI 的`ORIGIN`属性要改为`IMCOMPLETE`
- 如果 iBGP 邻居之间的路由要经过内部路由器，那么就要给 IGP 路由表中**注入 AS 外的路由**（但很可能非 BGP 路由承受不住），或者通过**隧道技术**（虚电路）连接 iBGP 邻居
- 内部路由用**默认路由**转到 BGP 路由，然后连到世界上任一台路由器；在路径上的路由不可设默认路由，否则只能转到一个方向

防止出现回路的方法：

- 内部：从 iBGP 邻居收到的 NLRI Update 分组不能再发给邻居
- 之间：利用 AS Path 防止回路

BGP 同步：BGP 路由器不应该使用或向 EBGP 邻居通告从 IBGP 邻居那里学习到的 BGP 路由信息，除非该路由是本地的或者该路由存在于 IGP 数据库，即该路由也能从 IGP 学习到。

同步规则要求 BGP 库与 IGP 库同步。这么做是为了使 AS 内的所有路由器都能达到同步，确保能够转发其它 AS 因使用该 AS 通告的路由而发送到该 AS 的那些数据。BGP 同步规则也保证了整个 AS 中路由信息的一致性，避免了 AS 内的路由“黑洞”。

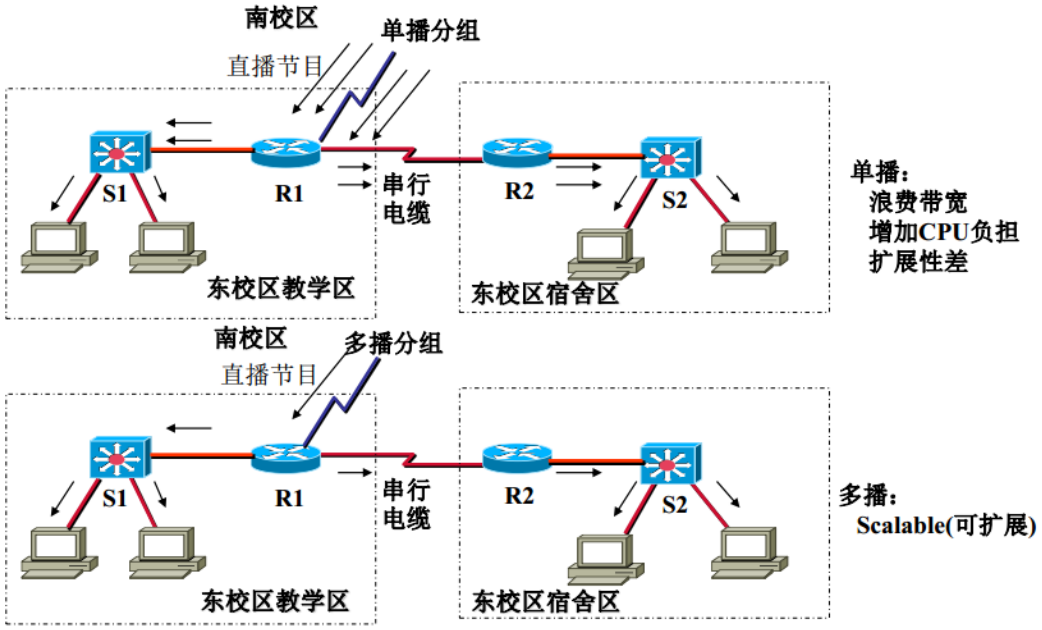
- 同步开启，BGP 和 IGP 必须同步，且有可达的下一跳（运行了 IGP 的情况下，一般来说就能保证下一跳可达）。
- 同步关闭，必须有可达的下一跳。

4.7 IP 多播

4.7.1 概述

访问同个网站，只发一次请求并回收；但单播访问同一个网站也要多次发送

- 单播：浪费带宽，增加 CPU 负担，扩展性差
- 多播：可扩展（视频直播非常有优势）

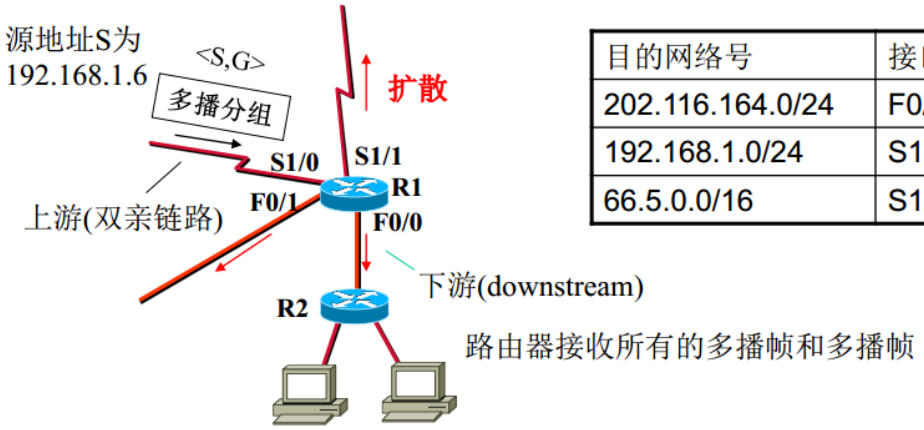


4.7.2 多播 IP 地址

多播地址范围	用法
224.0.0.0 - 239.255.255.255	IPv4 的多播地址空间
224.0.0.0 - 224.0.0.255	由 IANA 分配的永久地址。路由器不转发目的地址为这些地址的 IP 数据包
224.0.1.0 - 224.0.1.255	由 IANA 分配的永久地址。路由器会转发目的地址为这些地址的 IP 数据包
232.0.0.0 - 232.255.255.255	用于指定源的多播应用
233.0.0.0 - 233.255.255.255	由 AS 分配的全局多播地址
239.0.0.0 - 239.255.255.255	私有多播地址
其它地址	临时多播地址 (transient address)

4.7.3 多种多播协议

- 逆向路径广播：指定源的树
  利用源地址查路由表，当一个路由器收到一个源地址为 S 发往组 G 的多播分组 <S,G> 时, 当且仅当该分组到来的接口在从该路由器到 S 的最短路径 (Parent Link) 上时, 该路由器才在它的其它接口广播 (flooding) 该分组。（类 DFS）



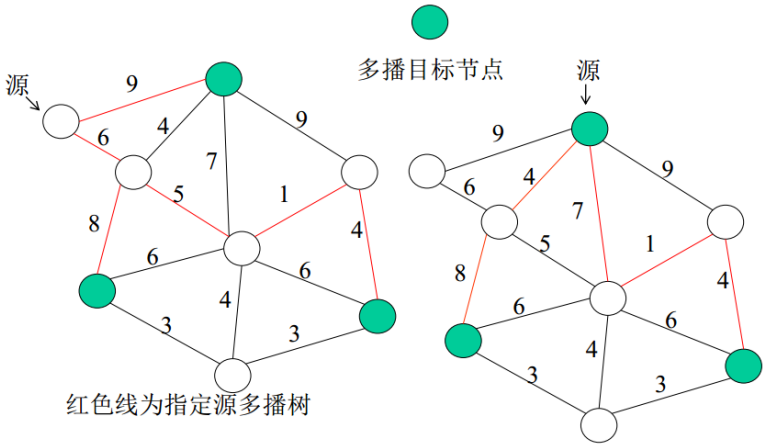
目的网络号	接口
202.116.164.0/24	F0/1
192.168.1.0/24	S1/0
66.5.0.0/16	S1/1

  建路由表时源地址路径上一定是最短路径，每一次都往远的地方走，故一定没有回路

- 逆向路径多播：对于每个多播地址 G，每个源地址形成一棵树，即源特定组播树 (source-specific multicast tree)，然后剪枝
  对于基于一个源地址的组播流，如果路由器的所有下游接口均无该组成员或已被剪枝，则它通过其双亲链路向上发送剪枝消息 (Prune Message)。路由器不会把多播分组从剪枝口转发出去
  如果有新主机加入多播组，则要通过嫁接消息 (Graft Message) 逐级向上通知直到某个未被剪枝的接口或者根路由器。

为了防止嫁接消息丢失引起转发受阻，路由器定期取消所有剪枝。

- 距离向量多播路由协议 (Distanse Vector Multicast Routing Protocol, DVMRP)：在距离向量算法的基础上使用逆向路径多播算法实现的多播路由算法
- 协议无关多播-稠密模式 (Protocol Independent Multicast - Dense Mode, PIM-DM) 协议：逆向路径多播算法，路由器只需要知道到源主机的最短路径的接口，与使用什么内部网关协议无关
- MOSPF 协议是另一种用于组成员稠密方式下的多播协议：如果使用 OSPF 协议，路由器可以通过 Group Membership LSA 把自己的哪些直连网有组成员的信息传遍整个 AS，最后，所有 AS 的路由器都在原拓扑结构图上标志哪些（网络）节点有组成员。这样, 每个节点就可以计算出源节点到组成员的最短路径多点播送生成树。MOSPF 路由器在收到多播分组时为每个源和多播组建造一颗生成树。由于计算量很大，MOSFP 不适用于大型网络。Steiner 树是总代价最小的分布树。求 Steiner 树是 NP 问题。



- 协议无关多播-稀疏模式协议 (Protocol Independent Multicast - Sparse Mode, PIM-SM) 用实现于组成员稀疏情形下的多播

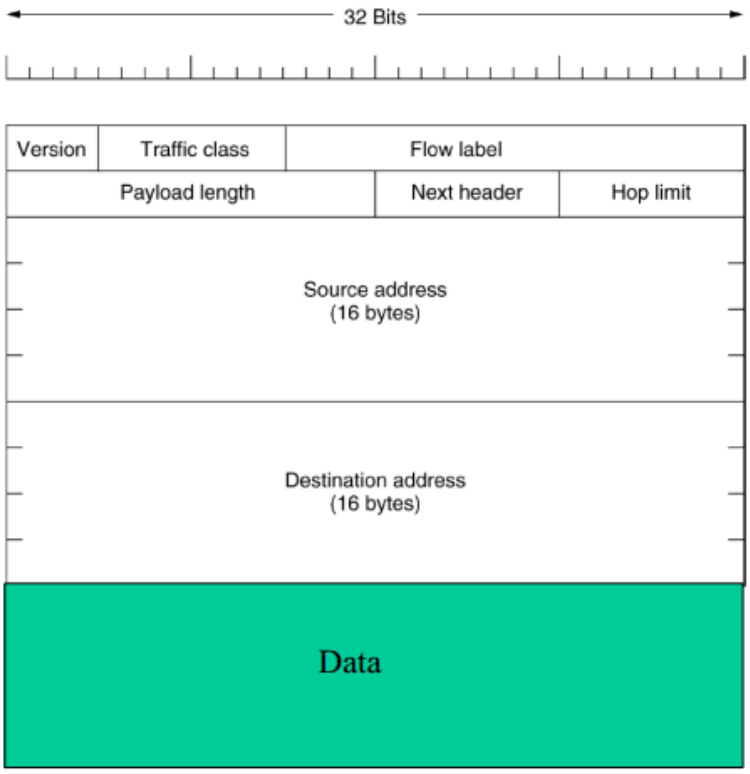
因特网组管理协议 (Internet Group Management Protocol, IGMP) 用于路由器查询与它直连的网络上是否存在组成员

- IGMPv1 协议只能对某个接口查询所有组，如果三次查询在十秒内都没有收到响应报告，则认为该接口没有任何组成员。
- IGMPv2 协议可以直接针对某个组进行查询，而且主机加入组和离开组都要发通告。

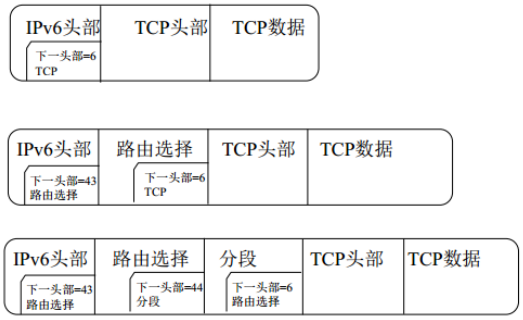
4.8 IPv6

4.8.1 IP 数据报

IPv4 的缺点有很多，如服务类型 (ToS) 没有用、选项长度太少、分段给路由器 CPU 造成很大的压力等。因此 IPv6 解决了 IPv4 存在的问题



- 版本号：取值为 6，表示 IPv6。
- 流量类别：数据流的类别或优先级，即 IPv4 的服务类型。
- 流标签：用于区分相同信源和信宿之间的不同数据流。取值 0 时，表示未用该标签。
- 有效载荷长度：包括扩展头和上层 PDU，最多 65535 字节。超过这一字节数的负载，该字段值置为“0”，使用扩展头逐个跳段 (Hop-by-Hop) 选项中的巨量负载 (Jumbo Payload) 选项。
- 下一包头：指出紧随 IPv6 头部之后的包头 (Header) 类型，如扩展头（有的话）或某个传输层协议头（如 TCP/UDP/ICMPv6），对应 IPv4 的选项。
  - 逐跳选项 (Hop-by-Hop)/0：要求每个路由器都必须检查选项部分，必须作为第一个选项，而且只允许出现一次。如用于传送超大分组（有效载荷的字节数用 32b 表示）的 Jumbo Payload 选项。
  - 目的地选项 (Destination)/60：用于为目的地传送信息，只在目的地检查该选项。
  - 路由选择选项 (Routing)/43：类似于 IPv4 的松散源路由。
  - 分段选项 (Fragmentation)/44：提供分段和重组服务，**IPv6 只能在源主机进行分段**。
  - 认证 (Authentication)/51：提供数据源认证、数据完整性检查和反重播保护。
  - ESP 选项 (Encrypted security payload)/50：提供加密服务。



- 跳数限制：类似于 IPv4 的 TTL。每次转发减 1，减到 0 时如果还没有到达目的地则被丢弃。
- 源地址/目的地址：IPv6 将 IP 地址扩充至**128 位**（不是 64!），确保物联网的地址足够用
- IPv6 只在源端分段，中间不能分段。

4.8.2 地址书写与简化

IPv6 地址由 128 位二进制数组成，分割成 8 段 16 位组，然后把每段 16 位组换算成 4 个十六进制数来表示，每段十六进制数值的范围是 0000-FFFF，每段之间使用冒号来分隔。

简化规则如下：

- 每一个段中开头的 0 可以省略不写，但末尾的 0 不能省略
  原始 IPv6 地址：`3ffe:1944:0100:000a:0000:00bc:2500:0d0b`
  简化后 IPv6 地址：`3ffe:1944:100:a:0:bc:2500:d0b`
- 如果某段或连续几段全是 0，则可以使用一个:来代替
  原始 IPv6 地址：`ff02:0000:0000:0000:0000:0000:0000:0005`
  简化后 IPv6 地址：`ff02::5`
- 如果 128 位全部为 0 的地址，则可以使用一个::来表示
  原始 IPv6 地址: `0000:0000:0000:0000:0000:0000:0000:0000`
  简化后 IPv6 地址：`::`
- IPv4 地址的表示方法：`0:0:0:0:0:0:d.d.d.d`
  IPv4 地址: `170.1.2.3`
  用 IPv6 地址表示：`0:0:0:0:0:0: 170.1.2.3` 或 `::170.1.2.3`

注意：在 IPv6 地址中，只能使用一次双冒号。如`2001:0d02:0000:0000:0014:0000:0000:0095` 以下两种缩写方式都是正确的：

```
2001:d02::14:0:0:95
2001:d02:0:0:14::95
```

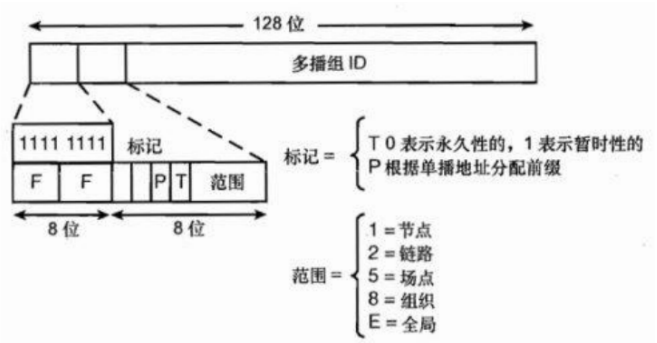
但下面这种缩写方式是错误的：

```
2001:d02::14::95
```

4.8.3 地址类型

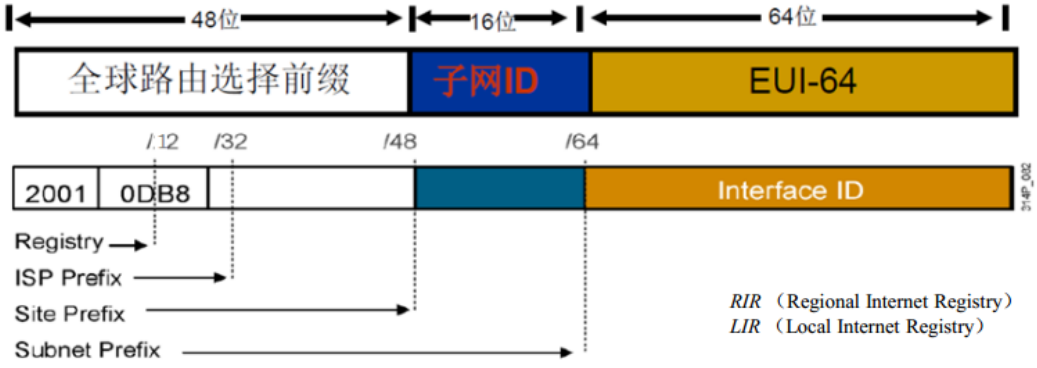
IPv6 单个接口可以分配多个任意类型的地址，包括以下三种

- 单播地址：标识单个节点
  - Global 全局：从2000::/3开始，由 IANA 分配
  - Reserved 保留：被 IETF 使用
  - Private 私有：局部链接（从FE80::/10开始）
  - Loopback 环回 (::1)
  - Unspecified 未指明 (::)
- 组播地址：标识一组节点（前 8 个 bit 为 1111 1111）

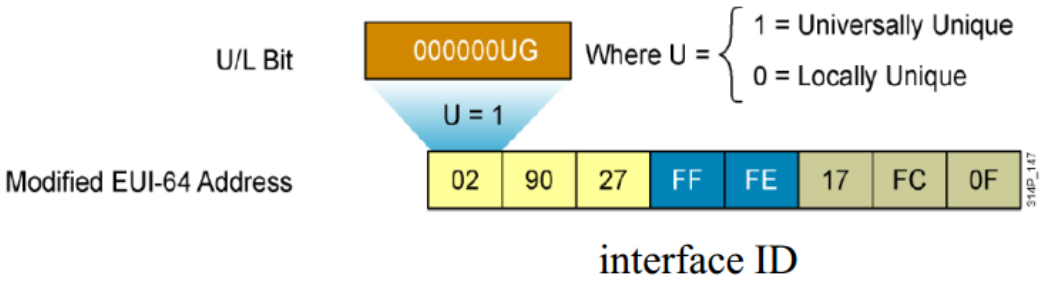


- 任意播地址：标识一组节点中任何一个成员，即源节点的数据流被转发到组里最近的节点
  - 任意播地址直接采用全局单播地址
  - 多台设备设置相同的任意播地址
  - 任意播地址的广泛使用会带来的混乱和危险

4.8.4 地址格式



全球网络前缀是由几个层次来构成的，根据地址分配空间来决定的。



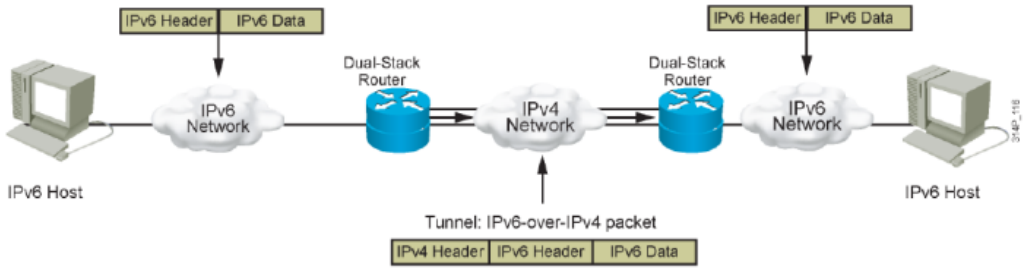
EUI-64 是通过在48 位 MAC 地址插入“FFFE”作为中间 16 位形成的。所选的 MAC 地址要求是全球唯一，U/L 位要设置为 1(0 为本地范围)。

4.8.5 兼容性

IPv6 和 IPv4 不兼容，故各种路由协议都要为 IPv6 单独设计。

多种 IPv4 过渡 IPv6 方式

- 双栈：即在接口上同时配置 IPv4 和 IPv6
- 使用 IPv4隧道来传输 IPv6 数据（加 IPv4 的头部）



- 在 IPV4 和 IPv6 之间进行 NAT 转换

4.8.6 优点

- 提供更大的地址空间
  - 改善全球的可达性和灵活性
  - 通过严格的地址分配和聚类 (Aggregation) 大大减小路由表的长度
  - 自动配置链路层地址，从而实现即插即用功能：无需配置 NAT 即可实现端到端的通信，简化了重新编址和修改地址的机制
- 支持移动性和安全性
  - IPv6 有加密与鉴别选项

  - IPv6 移动性内置

- 报头更简单
  - 路由选择效率更高，提高性能转发速率
  - 没有广播，不会出现广播风暴
  - 不需要处理检验和
  - 报头扩展机制更简单
  - 流标签无需查看传输层信息即能识别各种流

- 多种 IPv4 过渡 IPv6 方式

5 传输层

传输层协议称为**端到端**或**进程到进程**的协议。因特网的传输层可以为两个进程在**不可靠的网络层**上建立一条**可靠的逻辑链路**，提供**字节流**传输服务，并且可以进行**流控制**和**拥塞控制**。

因特网的传输层有两个协议：

- UDP 协议提供无连接的不可靠尽力服务（IP 数据报协议字段**UDP 为 17**）
- TCP 协议提供面向连接的可靠字节流服务（IP 数据报协议字段**TCP 为 6**）

我们把传输层的数据单元（报文）称为数据段 (segment)。

每个 TCP/UDP 连接可以由一个四元组唯一标识:源 IP 地址、源端口号、目的 IP 地址、目的端口号，通过**端口号**区分不同进程。

而一对IP 地址和端口号 就构成一个通信端点，也被称为套接字 (socket)。

- 知名端口：0-1023，为提供知名网络服务的系统进程所用。如:

80	HTTP	25	SMTP
20	FTP 数据	53	DNS
21	FTP 控制	110	POP3
23	Telnet		

- 注册端口：1024-49151。在 IANA 注册的专用端口号，为企业软件所用。
- 动态端口：49152-65535，没有规定用途的端口号，一般用户可以随意使用。也称为私用或暂用端口号。

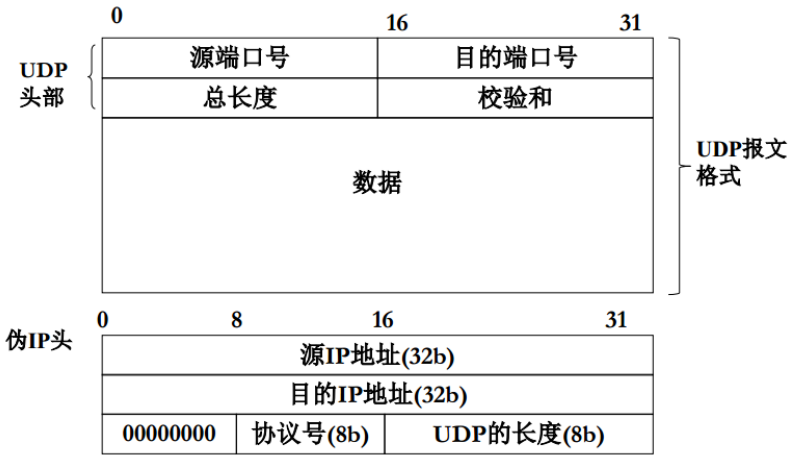
5.1 UDP 协议

5.1.1 概述

用户数据报协议 (User Datagram Protocol, UDP) 只提供**无连接的不可靠的尽力服务**。发送给接收进程的数据有可能**丢失**，也有可能**错序**。可以说 UDP 协议是 IP 协议的简单扩展，只是在 IP 协议上增加了**端口号**，把进程关联起来了。

- UDP 是面向报文的传输方式，即应用层交给 UDP 多长的报文，UDP 就照样发送，每一次都发送一个报文。既不合并，也不拆分，而是保留这些报文的边界。
- 接收进程每次接收一个完整的数据报，如果进程设置的接收缓冲区不够大，收到的数据报将被截断。

5.1.2 数据段格式

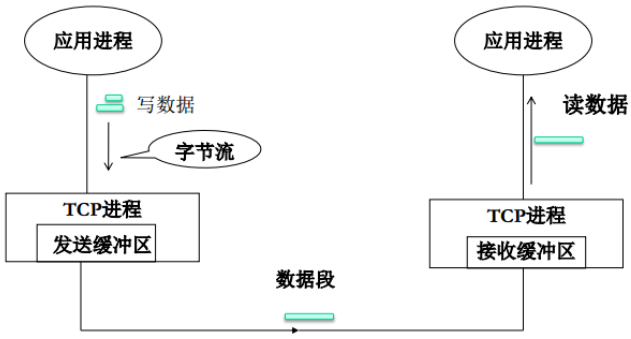


- 总长度：整个 UDP 报文长度
- 源端口号和目的端口号 (2B)：用于关联发送进程和接收进程
- 校验和：由伪 IP 头（只用来算校验和，不进行传递，实际上不存在）、UDP 头（校验和用 0 填充）和UDP 数据 形成。其中，伪 IP 头的协议号为 17。如果发送方把校验和设置为 0，接收方会忽略校验和。UDP 长度就是 UDP 头部填写的总长度。

5.2 TCP 协议

5.2.1 概述

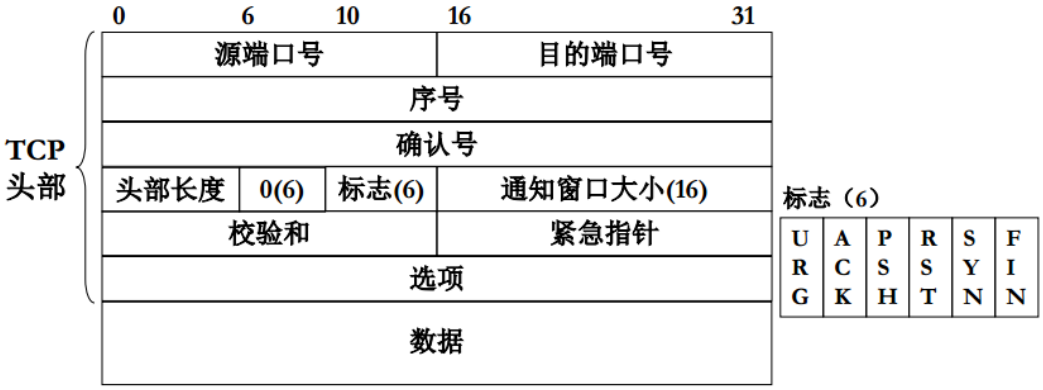
传输控制协议 (Transmission Control Protocol, TCP) 为进程之间提供面向连接的可靠的数据传送服务（通过滑动窗口协议实现）。TCP 为全双工协议。TCP 提供流控制机制，即控制发送方的发送速度，使发送的数据不会淹没接收方。作为因特网的主要数据发源地，TCP 还提供拥塞控制功能。



- TCP 连接只能在两个进程间建立

- 每个 TCP 连接可以由四元组唯一标识：源 IP 地址, 源端口号, 目的 IP 地址, 目的端口号。
- TCP 是**字节流**的服务，多次发送的数据可以放在一个数据段中传送且**不标识边界**。TCP 有一个缓冲，当应用程序传送的数据块太长，TCP 就可以把它划分短一些再传送。如果应用程序一次只发送一个字节，TCP 也可以等待积累有足够多的字节后再构成报文段发送出去。（注意上图，是将两条写数据合并在一起变为一个数据段才发送）
- TCP 连接提供**可靠的无比特错的数据传送**，但经过因特网可能出现**丢失和错序（同 UDP）**。
- 每个**数据段数据部分**的**最大长度 (字节)**不能超过 MSS(Maximum Segment Size)[32]。
- 客户端通过查路由表知道**IP 地址**，**端口号**自动选一个未用的

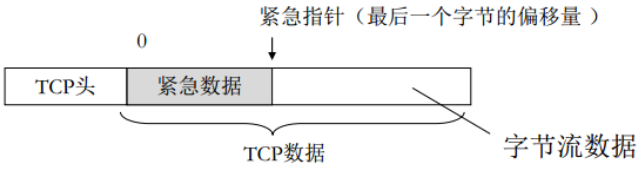
5.2.2 数据段格式



- 序号：字节流中的**每个字节**都会被编号。初始序号(initial sequence number, ISN) 采用基于时间的方案，一般采用**随机数**。数据部分的第一个字节的编号为**ISN+1**。每个**数据段的序号**采用其数据部分第一个字节的编号。
- 确认号：确认号为期待接收的下一个数据段的序号。只有设置了确认标志，确认号才有效
- 头部长度：以**字 (32b)**为单位，同 IP 数据报。
- 标志：标记为 1 时有效
  - SYN：同步序号标志，用来发起一个 TCP 连接
  - FIN：表示关闭连接，**不再发送数据**，但是可以接收数据，也可以发送**数据段**（不包含数据）
  - ACK：表示确认号有效
  - PSH：告知接收方发送方执行了推送 (Push) 操作，接收方需要尽快将所有缓存的数据交给接收进程
  - RST：发现连接可能出了问题，连接重置
  - URG：紧急指针标志位

[32]一般来讲，TCP 协议会以最大传输单元 (MTU，帧的数据部分) 减去 IP 头和 TCP 头作为 MSS。如果是以太网，则是 $1500 - 20 - 20 = 1460$。

- 通知窗口 (advertised window, advWin) 大小：接收方告知发送方还可以发送的字节的序号范围，发送方据此做出调整
- 校验和：由伪 IP 头、TCP 头和 TCP 数据部分 形成，其形成方法与 UDP 协议类似。伪 IP 头的协议字段值为 6。
- 紧急指针：用于指出紧急/带外数据(out-of-band, OOB)[33]的边界，即紧急数据的最后一个字节的偏移量。标志 URG 为 1 时有效。

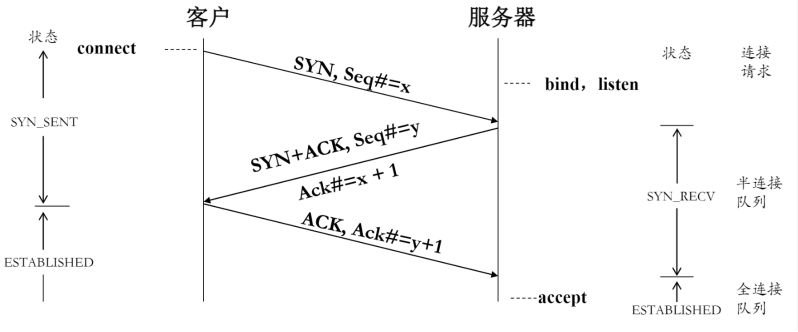


- 选项：建立连接时有 MSS、窗口比例 (Scale)、是否使用选择性确认 (SACK-Permitted) 等。数据传送时有选择性确认的序号范围 (Seletive ACK, SACK)、时间戳等。

5.2.3 工作过程

建立连接 ⟶ 传送数据 ⟶ 释放连接

- 建立连接：非对称活动，服务器一直在等，客户向服务器呼叫
- 传送数据：全双工方式
- 释放连接：对称活动，可由任何一方发起



三次握手建立连接

- 客户端需要知道服务器的 IP 地址和端口号。服务器收到客户端发来的连接请求（SYN 报文，含服务器端口）后

  1. 查看是否有进程监听该端口。若有，则将此连接请求传给该进程；否则，服务器发 RST 拒绝它。

[33]带外数据不属于字节流

2. 如果该进程接受连接请求，则发回 SYN+ACK 报文。

- 每一步均采用超时重传，多次重发后将放弃。重发次数与间隔时间依系统不同而不同（Linux 超时重传默认次数为 5）。
- x 和 y 为初始序号（随机数），分别用于两个方向的数据传送。两个方向的下一个数据段的序号分别为 x+1 和 y+1。
- 注意这里确认号含义与数据链路层不同，传输层的确认号是期待接收下一个字节的序号

例 24. *为什么需要三次握手建立连接？*

分析. *只有一次握手的话，也就是说客户端只要发送了连接请求就认为 TCP 已连接，也许服务器根本就不存在或者没打开。如果继续发送数据的话，浪费带宽。再说客户端也需要服务器传来的初始序号和很多选项，这个都做不到。*

如果采用两次握手，则客户端可以通过发送大量伪造的源地址连接请求，经过两次握手后服务器误以为连接已经建立，最终耗尽所有资源，使得合法的请求连接都被拒绝，无法提供正常的服务，此即 DoS 攻击的原理。

即使是三次握手也无法避免 DoS 攻击和 DDoS 攻击，因为依然可以实现大量客户端同时向服务器发送连接请求，然后接收下服务器发来的确认数据包后，不再向服务器发送确认数据包（即客户端主动不进行第三次握手），那么服务器在短时间内会维护这样的半连接队列，等待队列中的客户端确认。但由于每个半连接都会耗费服务器的资源，故最终会导致资源不够用，拒绝正常的连接请求，使得正常服务无法提供。

预防 DDoS 的方法：限制同时打开 SYN 半连接的数目，缩短 SYN 半连接超时时间，关闭不必要的服务。

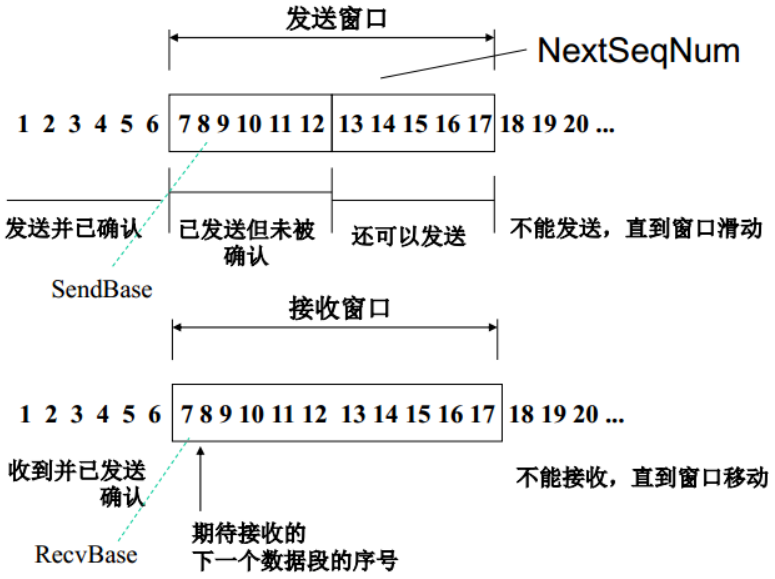
半连接队列和全连接队列

- 服务器发送 SYN+ACK 之后会把该连接请求放入该进程的半连接队列 (SYN 队列)。在收到客户端 ACK 后连接请求将从半连接队列转移到全连接请求队列 (accept 队列)。
- 每次执行 accept 时都会从全连接队列取出一个连接请求。
- 半连接队列和全连接队列的长度由执行 listen 函数时的参数 backlog 和几个系统参数确定。这些系统参数确定都可以修改。
- 如果半连接请求队列和全连接请求队列满，后面到来的连接请求将被直接丢弃。客户端此时已认为连接建立了，并不断发送包含 ACK 的数据段。服务器收到这些客户端发送的 ACK 后，一般会丢弃它。客户端第一个数据段重传几次后也会中止该连接。
- 在连接请求队列满丢弃新的连接请求时，还可以发送一个 RST 包给源主机（非默认设置）。

TCP 协议传输数据采用选择性重传协议（滑动窗口），不使用 NAK。只有一个超时定时器[34]。采用字节流方式，每个数据段使用其第一个字节的编号作为序号，按字节编号。

[34]工作机制：上一个数据段收到 ACK 后超时定时器指向接下来一个定时器；好处：可靠性，没有数据段被遗漏；缓解拥塞，虽然当连续多个包未收到 ACK 时重传导致发送速率变慢（Linux 重传时间会随着重传次数变多而变大），但是这恰恰缓解了网络拥塞的情况——连续丢包说明网络拥塞，应该降低发包速率

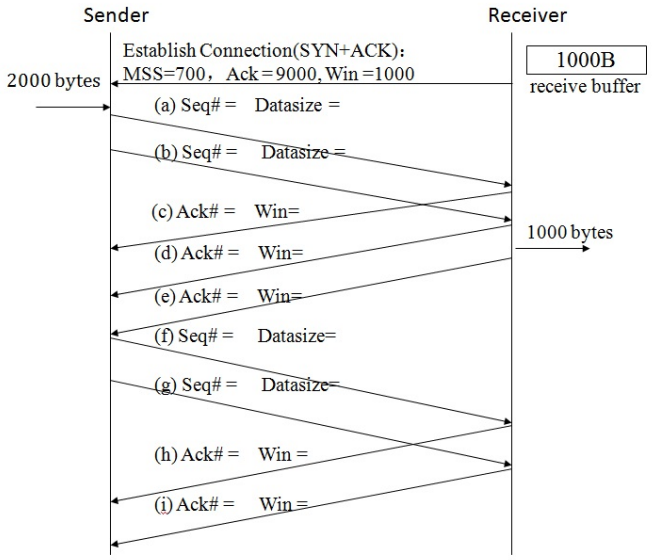
注意 TCP 协议中没有说明如何处理错序到达的数据段，要取决于具体实现。



* advWin为接收窗口的大小，即空闲块的大小（包含错序到达的数据段）

- 接收方先传 MSS，x 和接收/通知窗口大小 (advWin)
- 发送方做发送窗口，序号为 x+1，大小等同 advWin（接收窗口和发送窗口大小都会改变，流控制）
- 若接收缓冲区已满，则要等接收方进程将缓冲区取空，发送方才能继续发，否则发送窗口大小始终为 0
- 超时定时器会自动移动

例 25. 下图为一个普通 *TCP* 连接的数据传送图（不使用 *Nagle Algorithm,Delayed ACK,Fast Retransmission,Slow start* 等），请填空



分析. 注意每次发送数据最多就只有 *MSS*，但发送窗口大小可以大于 *MSS*。

(a) *9000 700*，取初始序号作为 *Seq* 初值，*advWin* 作为 *sendWin* 初值。发送数据段大小为 *MSS*，因为 *MSS* 小于滑动窗口大小。

(b) *9700 300*，继续将发送窗口内的字节发光，此时 *sendWin* 还是 *1000*，只是里面的字节都已发送但未确认而已

(c) *9700 300*，期待接收序号为 *9700*，*advWin* 减为 $1000 - 700 = 300$

(d) *10000 0*，期待收 *10000*，*advWin* 减为 *0*，进而 *sendWin* 减为 *0*

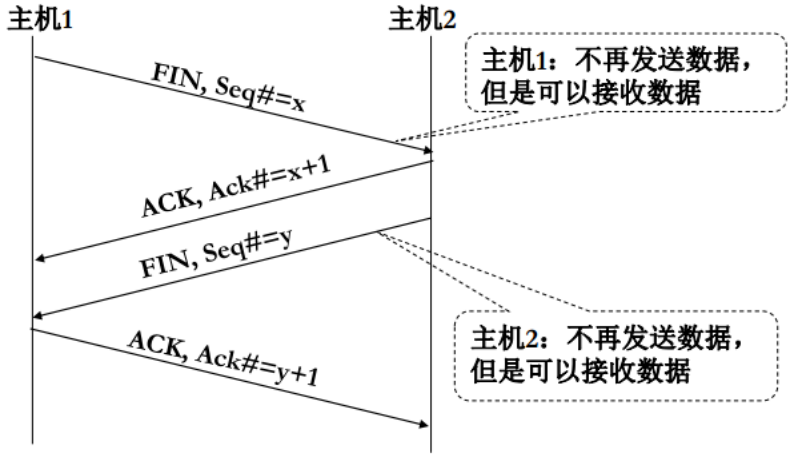
(e) *10000 1000*，取走 *1000*[35]，故 *advWin* 恢复，发 *ACK*

(f) *10000 700*，重复 *(a)(b)(c)(d)* 过程

(g) *10700 300*

(h) *10700 300*

(i) *11000 0*



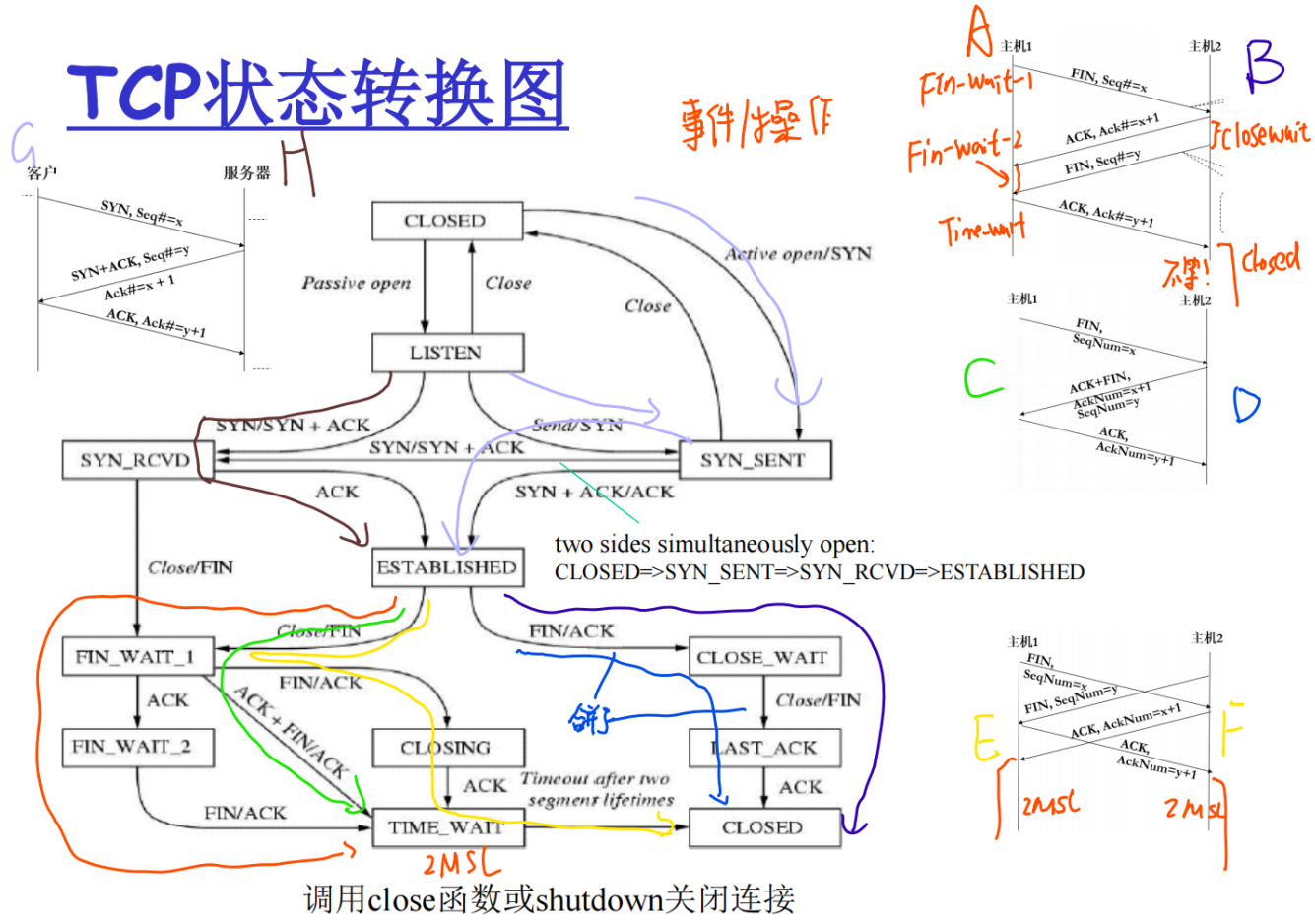
四次握手关闭连接

- FIN 报文采用超时自动重发方式。在若干次重发后依然没有收到确认，则发送 RST 报文给对方后强行关闭连接。不同的系统重发方法不同。x 和 y 都是上一个收到的数据段的确认号。
- 先发送 FIN 报文的一方在 ACK 发送完毕后需要等待**2MSL**(Maximum Segment Lifetime) 的时间才完全关闭连接（占用端口号），用于等待该连接的数据在因特网中消失[36]。TCP 标准中 MSL 采用 60 秒，Unix 采用 30 秒。如果两方同时发了 FIN（即上面的图中主机 2 的三角形上移至和主机 1 的三角形重合），那么两方都要等 2MSL
- 可以合并中间两次握手（ACK 和 FIN）或两方同时发出 ACK。

[35]有机会造成死锁：这个包是用来改发送窗口的；如果丢失了，发送方 SWS=0，发不了；接收方缓存为空，回不了信息；死锁了。解决方法：两个角度。接收方：定期发送确认，直至收到新的数据段（问题在于当发送方已经发完全部数据段，这种定期确认就是浪费资源）；发送方: 发一个探针（probe，一个字节的数据），如果接收方取走了，advWin 移动；如果接收方没取走，就可确认不是丢包的情况了，继续等待即可

[36]防止之后新建的连接收到这些包，导致接下来这个连接真正的包传送失败；使用与时间相关的随机初始序号也是为了避免这一点

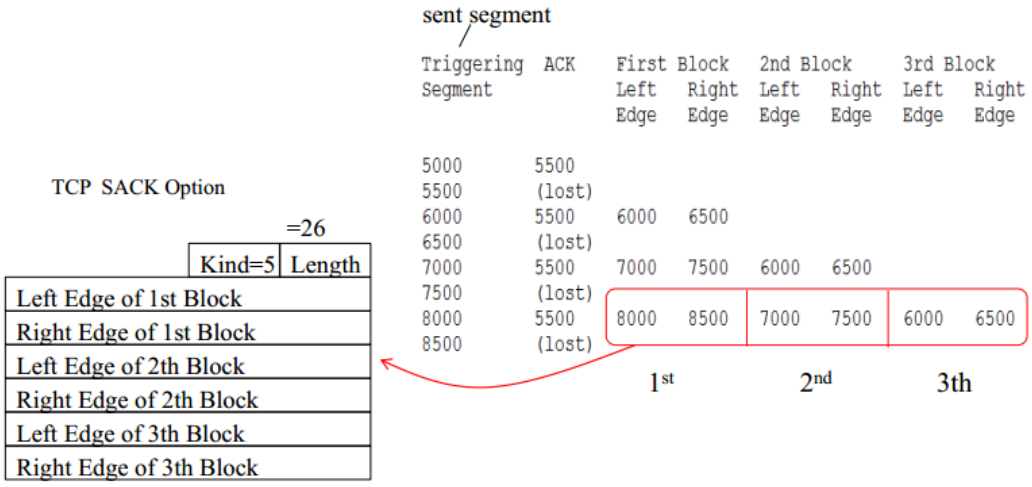
- 在进程异常退出时内核会直接发送 RST 报文而不是四次挥手来关闭连接，是一个暴力关闭连接的方式



5.2.4 重传机制

数据链路层每个帧都有一个超时定时器，而传输层只有**一个超时定时器**。

- 快速重传(fast retransmit)：如果发送方收到一个数据段的**3 次重复的 ACK**（包括第一次则一共 4 次），它就认为其后的数据段（由确认号指出）已经丢失，在超时之前会重传该数据段。缺点是丢包时间很长，优点是可以减缓网络压力。
- 延迟确认(delayed ACK)：接收方并不在收到数据段立即进行确认，而是延迟一段时间再确认。如果这个期间收到多个数据段，则只需要发送一个确认。如果在这个期间接收方有数据帧要发往发送方，还可以使用捎带确认 (piggybacking)。大部分的系统 (Windows/Unix) 的延迟确认时间为 200 毫秒。TCP 标准要求延迟确认不大于 500 毫秒。
- 选择性确认：接收方把收到的数据块通过数据段的选项告知发送方，使发送方不会重传这些数据块



5.2.5 传输层与数据链路层比较

传输层滑动窗口协议和数据链路层的区别

	传输层	数据链路层
序号	随机初始序号 +1，按字节计数	每一个帧一个序号
确认号	期待接收的序号	收到哪一个就确认哪一个 代表当前与之前的全收到了
窗口大小	会改变	不会改变
超时定时器	只有一个，只要有未确认的数据段 就会启动（针对未确认序号最小的）； 超时没收到确认，就会重传	每个帧都有一个

5.2.6 超时计算

数据链路层的回退 N 协议由于没有中间节点，故可以用固定的超时时间 (1RTT)。而传输层涉及多个节点，故需要实时变化，进行估计。

原始公式/指数加权移动平均方法 (Exponentially Weighted Moving Average, EWMA)

$$\text{EstimatedRTT} = (1-\alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$$

$$\text{RTO} = 2 \times \text{EstimatedRTT}$$

RTO(retransmission timeout) 为超时时间，$0 < \alpha < 1$，α 越小过去样本的影响越大。一般取值 $\alpha = 0.9$，这会使过去影响指数减少。

TCP 超时计算最常用的算法—**Jacobson 算法**

$$\text{EstimatedRTT} = (1-\alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$$

$$\text{DevRTT} = (1-\beta) \times \text{DevRTT} + \beta \times |\text{SampleRTT} - \text{EstimatedRTT}|$$

$$\text{RTO} = \text{EstimatedRTT} + 4 \times \text{DevRTT}$$

在 RTO 计算加上一个合适的安全边际 (safety margin)，使得在样本变化较大时 RTO 会很快变得更大（先估计偏移 RTT）。取 $\alpha = 1/8$，$\beta = 1/4$。如果发送窗口为 12MSS，则每 12 个段取样一次（采样频率）。

Karn 算法：在收到重传段确认时不要计算 EstimatedRTT，直接计算 RTO。

$$\text{RTO} = \gamma \times \text{RTO}$$

而是在每次重传时直接把 **RTO 加倍**（$\gamma = 2$）直到数据段首次得到确认，并把这个 RTO 作为后续段的 RTO。在 12 次重传后 TCP 协议发送 RST 并关闭连接。

例 26. *如果只有最后三个连续发送的数据段被丢失，其它数据段全部收到且 RTT 一直保持 20ms, 从这三个数据段的第一个数据段发送开始计时，还需要多少时间 (ms) 才可以全部收到这三个数据段的确认?*

分析. *一个 TCP 连接的发送方只有一个超时定时器，只针对未收到确认的第一个数据段启动。*
采用 Karn 算法，第一个数据段 RTO 20 + 重传时间 RTT 20 = 40ms
收到重传数据段确认，超时定时器移动，同时 RTO 加倍为 40 + 重传时间 RTT 20 = 60ms
收到重传数据段确认，超时定时器移动，RTO 加倍为 80 + 重传时间 RTT 20 = 100ms
总数为 $40 + 60 + 100 = 200ms$

5.2.7 拥塞控制

- 流控制：单一发送方和接收方，控制发送的速度，防止太多数据涌向接收方
- 拥塞控制：防止太多数据涌向网络，表现为丢包（路由器上缓冲区溢出）和长延迟（在路由器缓冲区中排队）

拥塞控制的两大类方法：

1. 端到端的拥塞控制:
   - 没有来自网络的明确的反馈
   - 终端系统通过丢包和延迟推导的拥塞
   - TCP 协议的方法
2. 网络辅助的拥塞控制:
   - 路由器反馈给终端系统
   - 用一个比特指出拥塞发生 (SNA, DECbit, TCP/IP ECN, ATM)
   - 向发送方给一个明确的发送速率

* 不主张发送 ICMP 包告知终端发生了丢包，因为会加重拥塞；应靠 TCP 自己发现拥塞。

TCP 的拥塞控制属于端到端的拥塞控制

- 超时 或收到 3 个重复 ACK 就认为丢包了（同快速重传），看作拥塞发生

- TCP 协议通过降低发送速率来控制拥塞。发送速率与发送窗口大小有关：

$$\text{发送速率 (rate)} = \text{SWS(Sending Window Size)} / \text{RTT}$$

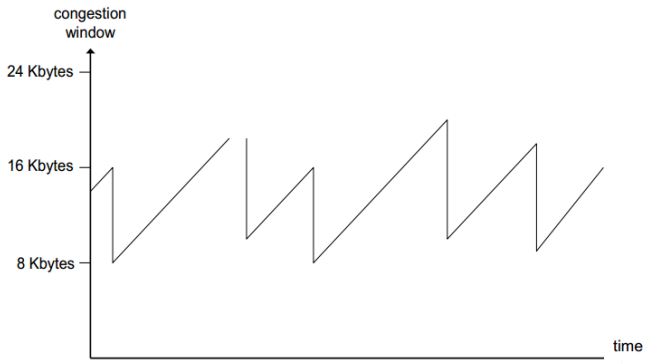
- 通过引入**拥塞窗口变量 CongWin**来限制 SWS。

$$\text{SWS} = \min\{\text{CongWin}, \text{AdvWin}\}$$

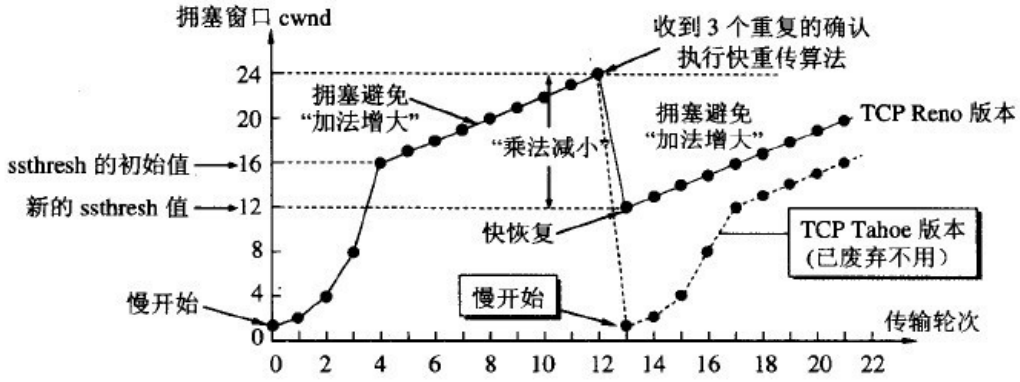
CongWin 为发送窗口的流量控制，AdvWin 为接收方要求的流量控制

TCP 协议改变 CongWin 的几种机制：

- 加性增乘性减 (AIMD)：发生拥塞**立即减半**，然后**线性增长**（每过 1 个 RTT 时间 CongWin 增加 1MSS）



- 慢启动 (slow start)：Tahoe 算法



假设算法开始时通知窗口大小 AdvWin=65535（用系统参数`TCP_MaxWin`（一般为 65535）限制 CongWin 的大小），SegSize 为被确认的数据段的大小。超时或收到 3 个重复 ACK 就会触发快速重传及 CongWin 的变化

1. 初始时，CongWin 设为 1MSS, 阈值 (threshold) 设为 65535, 发送一个数据段。

2. 在当前窗口所有数据段的确认都收到之后, CongWin 加倍。实际上，每收到一个确认，CongWin 增加一个 MSS。把它称为慢启动 (slow start) 是因为这个方法比立即采用 advWin 更慢。
3. 当拥塞发生时，把**当前 CongWin(或 SWS) 的一半**保存为阈值，然后 CongWin 又从 1MSS 开始慢启动（时刻 0 就是 1MSS）。
4. 当 CongWin 增长到等于或大于阈值时，在当前窗口所有数据段的确认都收到之后, CongWin 增加一个 MSS（拥塞避免）。实际上，每收到一个 ACK，CongWin 增加 SegSize/CongWin。如果发生拥塞，转 (3)。

- 快速恢复 (fast recovery)：Reno 算法
  从原来 CongWin 的一半开始线性增长（时刻 0 就是一半 CongWin）

5.2.8 存在的问题

问题零—关闭连接后立即又生成新的连接导致错乱

随机初始序号和**2MSL**都用于阻止重建 TCP 连接相同 4 元组，不会收到上一次连接遗留的数据的干扰

问题一—长肥管道：带宽大

$$\text{未确认的数据量 capacity(b)} = \text{bandwidth(b/s)} \times \text{RTT(s)}$$

- **序号回绕**问题（因为速度太快）：使用一般数据段的选项`timestamp`(TS)。只用于区分回绕的序号是不同的，不用于确定先后次序。也可以用 Window scaling 的方法，在 SYN 数据段选项设置 WinScale（取值 0-14, 默认值为 0）

$$\text{sendWin} = \text{advWin} * 2^{\text{WinScale}}$$

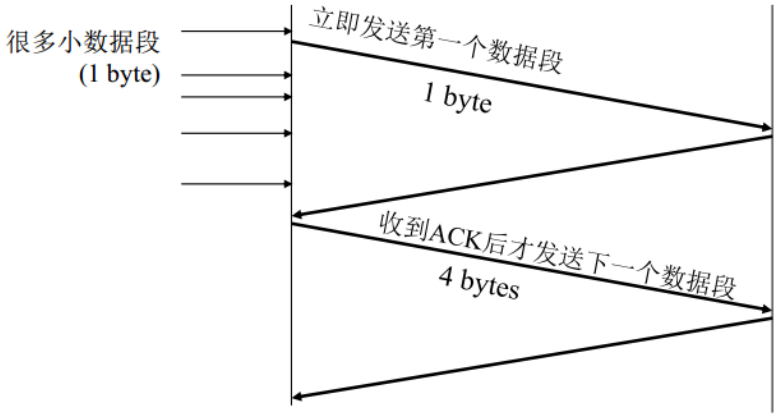
- **发送窗口太小**，满足不了需求：当丢包发生时，由于 SWS 的限制，管道将会被清空。解决方法：快速重传、快速恢复。（数据链路不会，因为直连网）

问题二—死锁现象：sendWin 为 0，接收方取空，再发不为空的 ACK 丢失了。缓冲区不变化，不会发确认回来。解决方案：

- 接收方可以启动一个超时定时器，可以是可以，但如果发送方没有数据传了，那接收方还是会继续发。（无谓发送数据）
- 一般从发送方解决：在发送窗口为 0 之后，并且发送方有数据要发送，则启动坚持定时器 (Persist Timer)，定期从要发送的数据中取一个字节发送出去 (Window Probe)，直到收到不为 0 的通知窗口为止

问题三—傻瓜窗口症候 (silly window syndrome)

- 发送进程有很多小批量数据要发送，如用 telnet 作为远程终端
  Nagle 算法—启发式算法（类似于停等协议）



  1. 立即发送一个数据段，即使发送缓冲区只有一个字节
  2. 只有收到上一个数据段的确认（此时缓冲区已经积累一定数据）或者
     发送缓冲区中数据超过 MSS，才可以发送下一个数据段
  3. 对于即时性要求高的地方，如 Window 方式的鼠标操作，要关闭Nagle 算法

- 接收进程频繁取走小批量数据
  Clark 算法：要等到接收缓冲区的空闲块大小为接收缓冲区大小的一半
  或达到 MSS 时才发送确认。

5.2.9 定时器

- 每个连接只针对第一个未确认数据段启动重传定时器(retransmission timer)。所有数据段都已确认，则关闭它。超时重传或发送窗口移动时要重启该定时器。
- 坚持定时器(persist timer) 用于保持信息流动，即使另一端关闭了接收窗口。
- 保活定时器(keep-alive timer) 在长时间没有交换数据段之后，用于检测连接的另一端是否出了问题。（如微信）隔 2 个小时，发 10 个数据段，如果没有 ACK 则关闭连接。（由应用层程序来做而不是 TCP）

5.3 TCP 与 UDP 比较

	TCP	UDP
连接	面向连接	无连接
传输可靠性	可靠的	不可靠的
消息	面向字节流	面向报文
通信模式	只支持点对点	支持一对一、多对多、多对一、多对多
拥塞控制	有	无
系统资源/开销	大	小

当数据传输的性能必须让位于数据传输的完整性、可控制性和可靠性时，TCP 是更好的选择；反之，当强调传输性能而不是传输的完整性时，如：音频和多媒体应用，UDP 是最好的选择。

6 应用层

应用层协议一般都是采用客户-服务器结构

应用	应用层协议	传输层协议	端口号
Email	SMTP [RFC 2821]	TCP	25
远程终端访问	Telnet [RFC 854]	TCP	-
Web	HTTP [RFC 2616]	TCP	-
文件传输	FTP [RFC 959]	TCP	21
域名解析服务	DNS	UDP	53
简单网络管理协议	SNMP	UDP	161

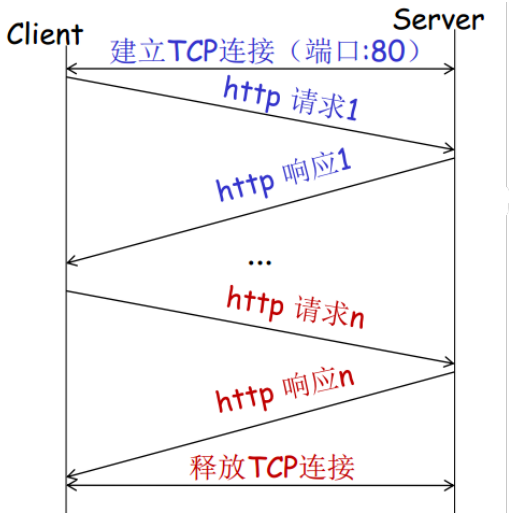
6.1 HTTP 协议

6.1.1 概述

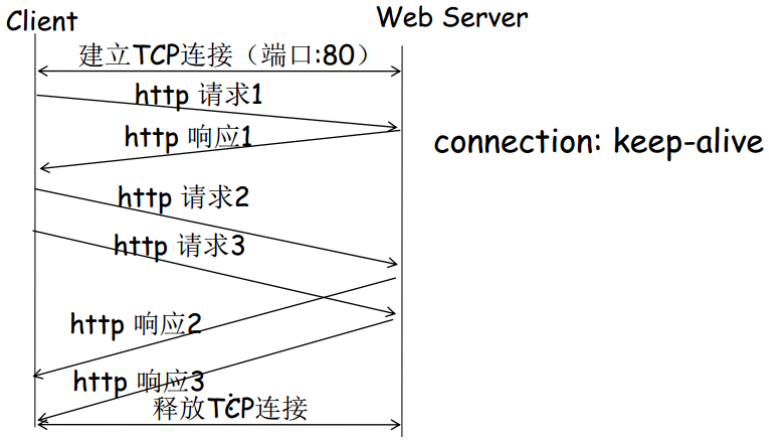
- 网页 (Web page) 是由对象 (objects) 构成的。这些对象可以是 HTML 文件、JPEG 图像文件、MP4 视频文件等。
- 网页的 HTML 文件中指出了所需的其他对象。
- 每个对象采用 URL 指明存放地址。URL 由主机名 和路径名 构成。

HTTP: 超文本传送协议 (hypertext transfer protocol)，是无状态的 (stateless)，server 不保留过往客户端的任何信息

- 非持续连接的 HTTP：每个时刻最多请求一个 Web 对象，每建立一个连接最多只能传送一个 Web 对象。`connection: close`
- 非流水式持续连接 HTTP：每个时刻可以请求多个 Web 对象，每建立一个连接可以传送多个 Web 对象。`connection: keep-alive`

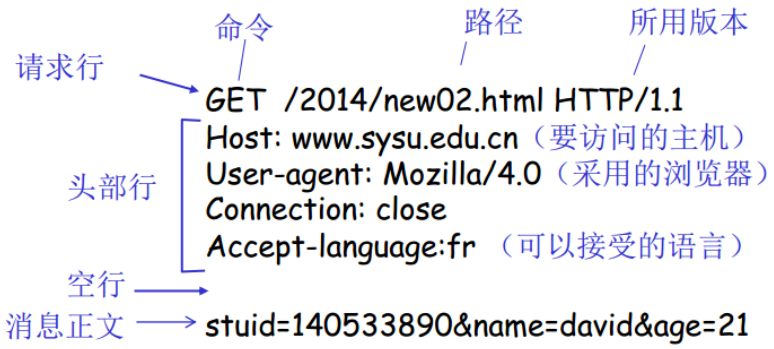


- 流水式持久连接 HTTP：可以连续请求多个 Web 对象，每建立一个 TCP 连接可以传送多个 Web 对象。HTTP/1.1 默认使用持续连接。`connection: keep-alive`



6.1.2 消息

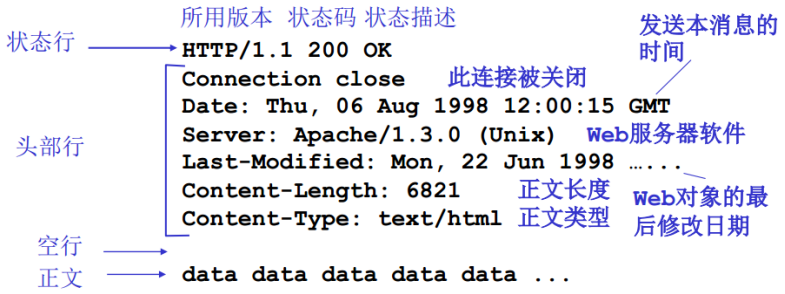
HTTP 请求消息 (message)



- 只有 POST 命令才有消息正文

- 每一行均以回车换行结束（\r\n），这一点很重要，否则收不到回复！

HTTP 响应信息



用 telnet 建立 TCP 连接，然后发送 GET 请求，如

```
telnet www.sysu.edu.cn 80
GET /2012/cn/index.htm HTTP/1.1
Connection: close
Host: www.sysu.edu.cn
```

6.1.3 HTTP 版本更新

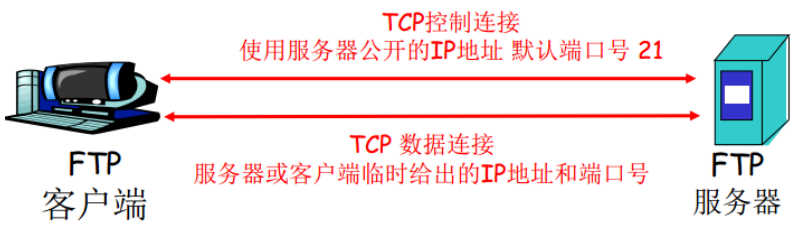
HTTP1.1 对 HTTP1.0 的改进：

1. 一个 TCP 连接可以传送多个 HTTP 请求和响应。
2. 可以采用流水线方式，即多个请求和响应过程可以重叠。
3. 增加了更多的请求头和响应头。

HTTP2.0

1. 采用二进制格式，而非文本格式
2. 完全多路复用，而非有序并阻塞的
3. 使用报头压缩来降低开销
4. 让服务器可以将响应主动推送到客户端缓存中

6.2 FTP 协议



- 使用 FTP 协议首先建立控制连接，然后建立数据连接，客户端再通过控制连接发出命令，通过数据连接得到服务器返回的结果或上传数据给服务器。
- 控制连接为带外数据 (out of band)
- FTP 服务器会保留状态: 当前目录、已做的认证

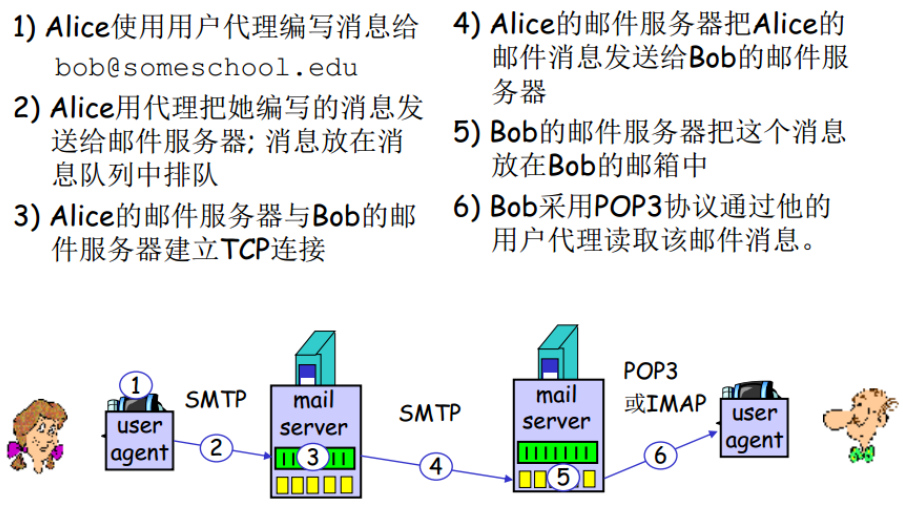
如下例，得到 IP 地址 (202.116.86.101) 后，计算端口号 (12*256+26=3098)，进而建立数据连接telnet 202.116.86.101 3098

```
telnet 202.116.86.101 21
220 Microsoft FTP Service
user net
331 Password required for user.
pass 123456
230 User user logged in.
pasv
227 Entering Passive Mode (202,116,86,101,12,26).
list
125 Data connection already open; Transfer starting.
226 Transfer complete.
quit
221
```

6.3 Email 协议

Email 有三个主要组件：

- 用户代理 (user agents)
- 邮件服务器 (mail servers)
- 简单邮件传送协议 (simple mail transfer protocol, SMTP)：TCP 端口号 25



6.4 域名系统

6.4.1 概述

域名系统 (Domain Name System, DNS) 提供的服务

- 主机名到 IP 地址的转换
- 为主机取别名：权威名 (Canonical name)、别名 (alias names)
- 为邮件服务器取别名
- 负载分配 (load distribution)：重复的 Web 服务器: 一个权威名对应一组 IP 地址

不采用集中式 DNS 的原因：不易扩大规模

- 单点失效
- 流量过于集中
- 可能距离数据库太远
- 维护问题

6.4.2 DNS 服务器

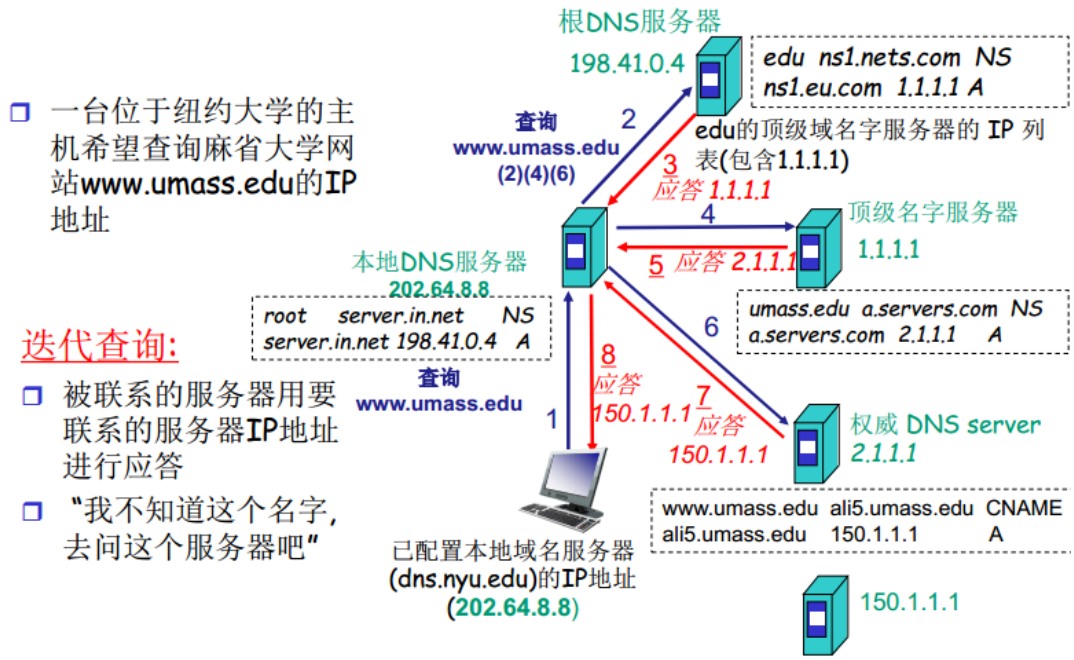
根名字服务器

- 公开的 IP 地址，不需要解析名字，由本地名字服务器直接联系
- 根名字服务器：如果不知道名字映射，把该名字所在的权威名字服务器的 IP 地址返回给本地名字服务器
- 根服务器主要用来管理互联网的主目录，全世界只有 13 台：1 个为主根服务器，放置在美国。其余 12 个均为辅根服务器，其中 9 个放置在美国，欧洲 2 个，位于英国和瑞典，亚洲 1 个，位于日本。

权威服务器

- 顶级域名提供了权威 DNS 服务器的 IP 地址，包括 com, org, net, edu, uk, fr, ca, jp 等
- 权威 DNS 服务器：
  - 每个组织机构的公开可访问主机都必须提供公共可访问的 DNS 记录，这些记录保存在权威 DNS 服务器上，并把这些主机的主机名映射到 IP 地址上 (e.g., Web, mail).
  - 权威服务器由大型组织或服务提供商维护

DNS 解析有两种方法：迭代查询和递归查询，下图是迭代查询



6.4.3 资源记录

采用分布数据库保存资源记录 (resource records, RR)

PR 格式: (name, value, type, class,ttl)

- Type=A (Address RR)
  - name 是主机名
  - value 是 IPv4 地址
- Type=CNAME
  - name 是别名
  - 值是规范名 (canonical name) 或真名 (the real name)：例如，www.jazz.com 是主机 bp2.jazz.com 的别名

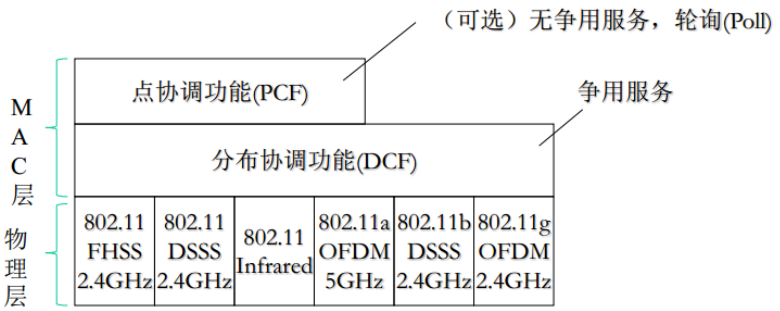
7 其他内容

7.1 无线局域网

7.1.1 基本特性

无线局域网 (WiFi)IEEE 802.11

- 不同标准 MAC 层一样，改进的是物理层（发、编码、怎么收）
- 无线信道特点：信号衰减快、抗干扰能力差、发送速率慢、共享信道有限带宽



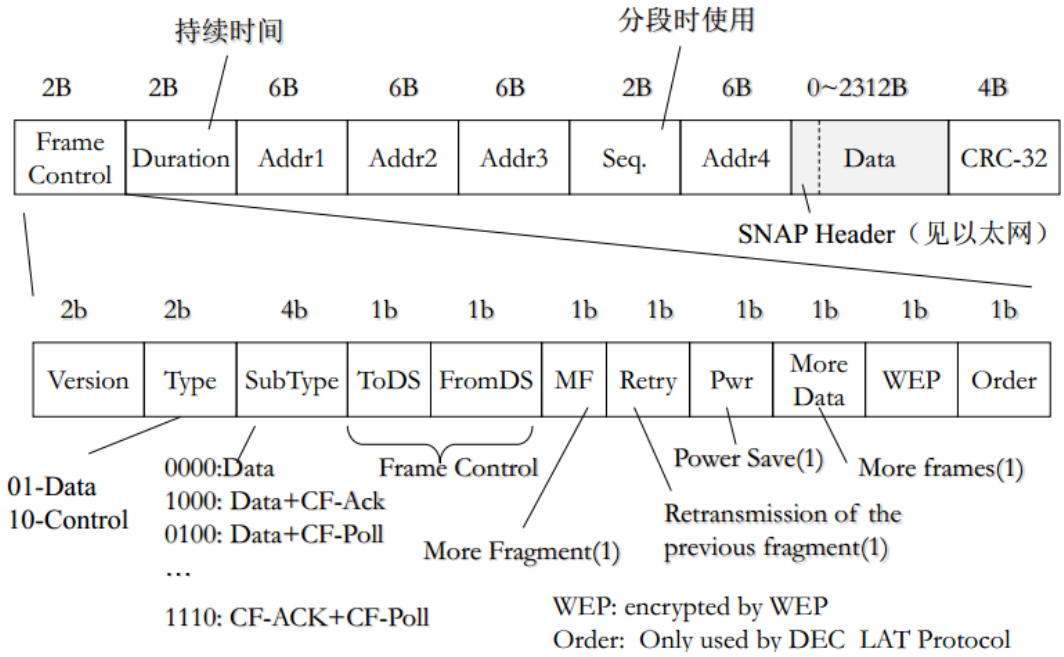
MAC 层一般用分布协调功能 (DCF)

- 原子操作：发完数据等待 28μs 后才发 ACK
- CSMA/CA(Carrier Avoidance) 算法：尽可能少冲突，尽可能少重传；即使空闲也要随机等一段时间

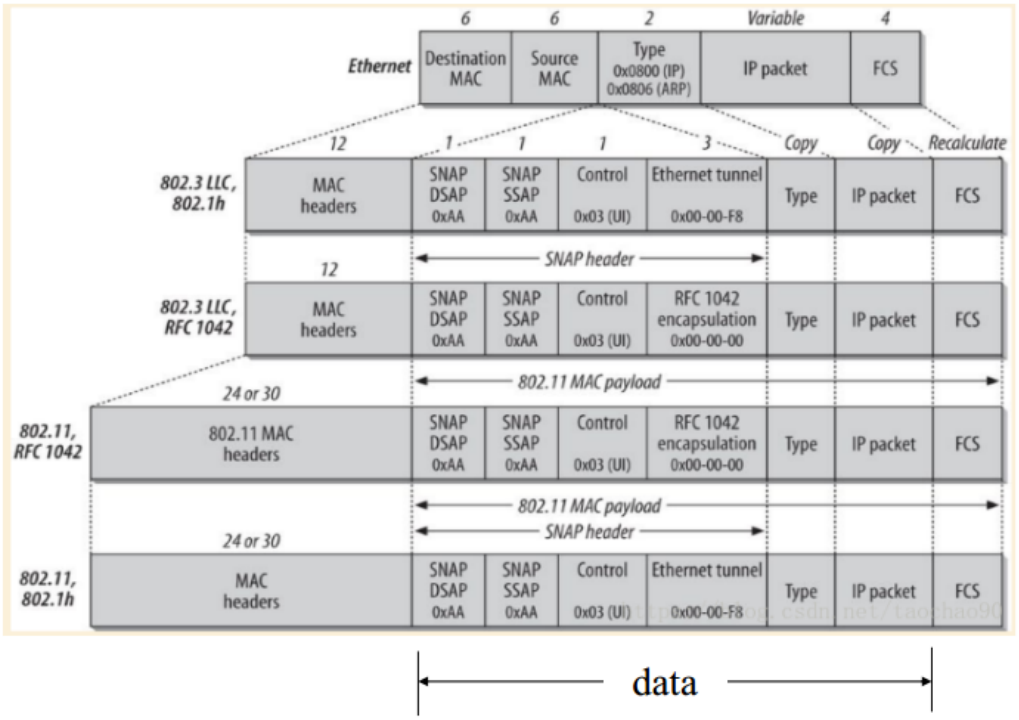
7.1.2 设施

- 无固定设施（自组织 IBSS）：独立基本服务集 (Basic Service Set, BSS)，不需要路由器，可以几台 PC 直接连
- 有固定设施：有个接入点 (Access Point, AP)，有路由器，连成基本服务集；扩展的服务集 (ESS)，完全无缝连接，自动迁移到下一个 AP

7.1.3 帧格式



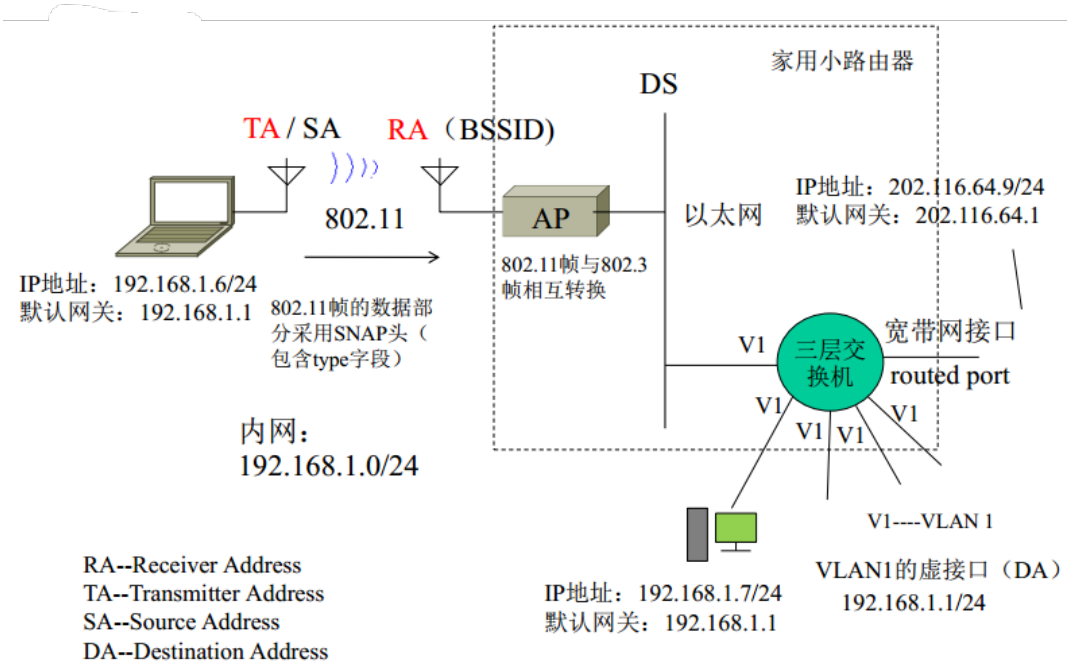
802.11 的帧一共有**4 个地址**，并没有指出上层协议的部分，所以数据部分 (payload) 加了**SNAP**的头部，里面包含了类型指明上层协议。



Function	ToDS	FromDS	Address1 (receiver)	Address2 (transmitter)	Address3	Address4
IBSS	0	0	DA	SA	BSSID	Not Used
To AP (infBSS)	1	0	BSSID	SA	DA	Not Used
From AP (infBSS)	0	1	DA	BSSID	SA	Not Used
WDS (bridge)	1	1	RA	TA	DA	SA

DA - Destination Address
SA - Source Address
TA - Transmitter Address
RA - Receiver Address
BSSID - AP MAC address
WDS - Wireless Distributed System

7.1.4 家用小路由器



- 很多家用小路由器的内网 IP 都设成 192.168.1.0/24
- 默认网关配成虚接口的地址 (192.168.1.1)
- 三层交换机的路由表：两个直线网 (192.168.1.0/24、202.116.64.9)、默认路由 (202.116.64.1)，访问因特网就会匹配上默认路由（有线方式）
- 无线方式：发出来有三个地址
  - TA（左侧电脑无线网卡的 MAC 地址）作为 SA（源地址）
  - DA（虚接口的 MAC 地址）作为目的地址
  - RA（AP 的 MAC 地址/BSSID），因为有很多个 AP，所以要指定 RA

  AP 收下帧后，把 802.11 的帧转化为以太网的帧（有源地址和目的地址了，同时 SNAP 中也给出了类型），然后用以太网协议发送，三层交换机收到（虚接口地址），再查路由表转发给对应主机或访问外网
- 当然 DA 也可以直接写目的主机的 IP 地址，这样的话交换机就作为透明网桥使用

7.1.5 其他技术

- 翻译网桥：一种帧转换为另一种帧
- 不同频率的 WiFi：5GHz WiFi 干扰小（不是说现在的 5G）
- 跟以太网不同，WiFi 在物理层是有格式的，远距离会换编码，降速进行传输
- MIMO 技术：多发射天线和多接收天线形成多个空间数据流

7.2 网络管理

SNMP 协议：Simple Network Management Protocol

7.3 广域网

ADSL 不是用语音传，是用数据传的，现在都不用电话线连了。

7.4 软件定义网络

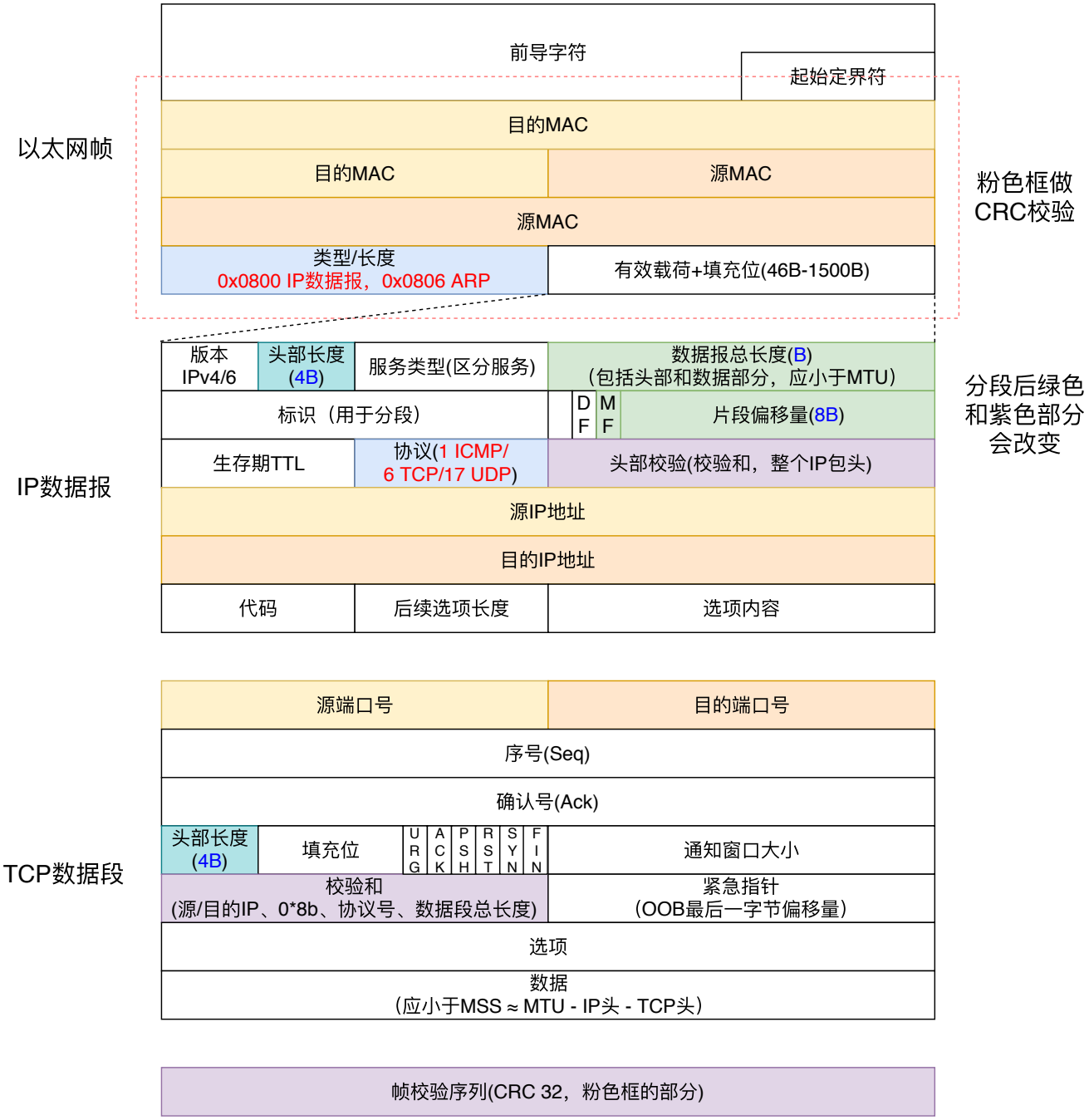
软件定义网络 (Software Defined Network, SDN)：原来路由器交换机都是固定电路，但现在可编程，是未来网络发展方向，最大的好处在于虚拟化。

控制器相当于一个电脑，用来修改下面（几十台）转发器的路由表。

7.5 多媒体网络

- 插值解决丢包问题，因为是连续变化的
- 传语音数据是用 RTP 协议传的，基于 UDP 协议
- SIP 协议用来做网络会议
- H.232 协议则更加完整
- 令牌桶：没传的时候放令牌，有传的时候用光令牌
- MPLS 加标签实现 VPN，沿着同一条路径走过去

8 总结



9 参考资料

1. 计算机网络总结，https://blog.csdn.net/n1neding/article/list/7?t=1&