

# 计算机网络笔记

陈鸿峥

2019.07\*

## 目录

<b>1 计算机网络概述</b>	<b>2</b>
1.1 网络连接方式 . . . . .	2
1.2 因特网 . . . . .	3
1.3 网络服务 . . . . .	3
1.4 因特网体系结构 . . . . .	3
1.5 网络性能分析 . . . . .	5
<b>2 物理层</b>	<b>7</b>
2.1 编码方式 . . . . .	7
2.2 物理介质 . . . . .	8
<b>3 数据链路层</b>	<b>9</b>
3.1 逻辑链路控制子层 . . . . .	10
3.2 介质访问控制子层 . . . . .	14
3.3 透明网桥 . . . . .	17
3.4 生成树协议 . . . . .	19
3.5 虚拟局域网 . . . . .	20
3.6 物理设备 . . . . .	21
<b>4 网络层</b>	<b>23</b>
4.1 IP数据报 . . . . .	24
4.2 IP地址 . . . . .	28
4.3 IP数据报相关协议 . . . . .	30
4.4 路由协议 . . . . .	33
4.5 内部网关协议 . . . . .	37
4.6 外部网关协议 . . . . .	43
4.7 IP多播 . . . . .	45

---

\*Build 20190705

<b>5 传输层</b>	<b>47</b>
5.1 UDP协议	48
5.2 TCP协议	48
5.3 TCP与UDP比较	59
<b>6 应用层</b>	<b>60</b>
6.1 HTTP协议	60
6.2 FTP协议	62
6.3 Email协议	63
6.4 域名系统	63
<b>7 其他内容</b>	<b>65</b>
7.1 无线局域网	65
7.2 网络管理	65
7.3 广域网	66
7.4 软件定义网络	66
7.5 多媒体网络	66
<b>8 总结</b>	<b>67</b>
<b>9 参考资料</b>	<b>67</b>

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

计算机网络将终端设备连接起来并可以传输数据。

### 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: 涉及交换机、路由器

## 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)。

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): 电脑、鼠标、USB
  - 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)
  - 广域网(Wide Area Network, WAN): **因特网**
- 

因特网设备：

- 终端系统/主机(end system): 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
- 

因特网的组成：ISP(Internet Service Provider)

- 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（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/一直发消息，如电视

因特网是**数据报服务，无连接无确认（尽力服务）**。

## 1.4 因特网体系结构

因特网体系结构包括以下这五层，而ISO/OSI(open system interconnection)网络包括七层协议<sup>1</sup>：

- 应用层：提供对某些专门应用的支持，如FTP、SMTP、HTTP

---

<sup>1</sup>也有TCP/IP四层的说法，将物理层和数据链路层合并起来变成物理网络层

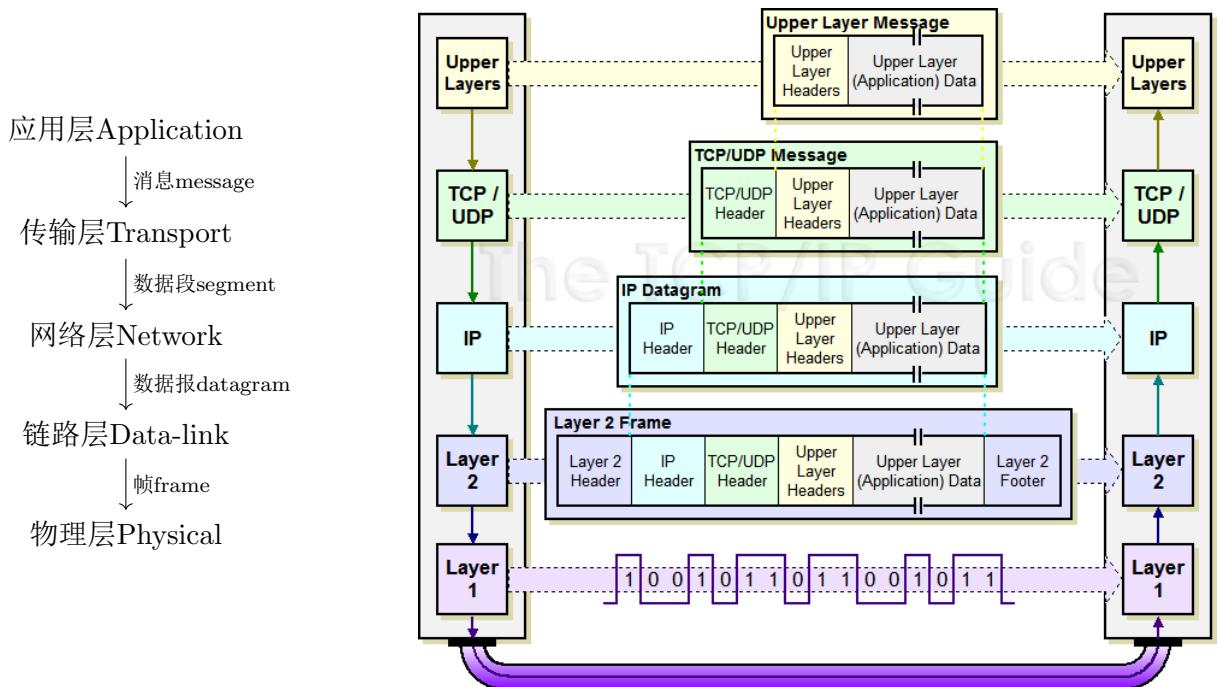
- (OSI)表示层(presentation): 提供数据转换服务, 如加密解密, 压缩解压缩, 数据格式变换
- (OSI)会话层(session): 简化会话实现机制, 如数据流的检查点设置和回滚, 多数据流同步
- 传输层: 将网络层获得的包在进程之间数据传送 (端到端), 如TCP、UDP
- 网络层: 路由选择, 实现在互联网中的数据传送 (主机到主机), 如IP协议、路由协议
- 数据链路层: 在物理网络中传送包 (跳到跳<sup>2</sup>, 节点到节点), 如PPP、Ethernet
- 物理层: 线上的比特 (传送原始比特流)

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。

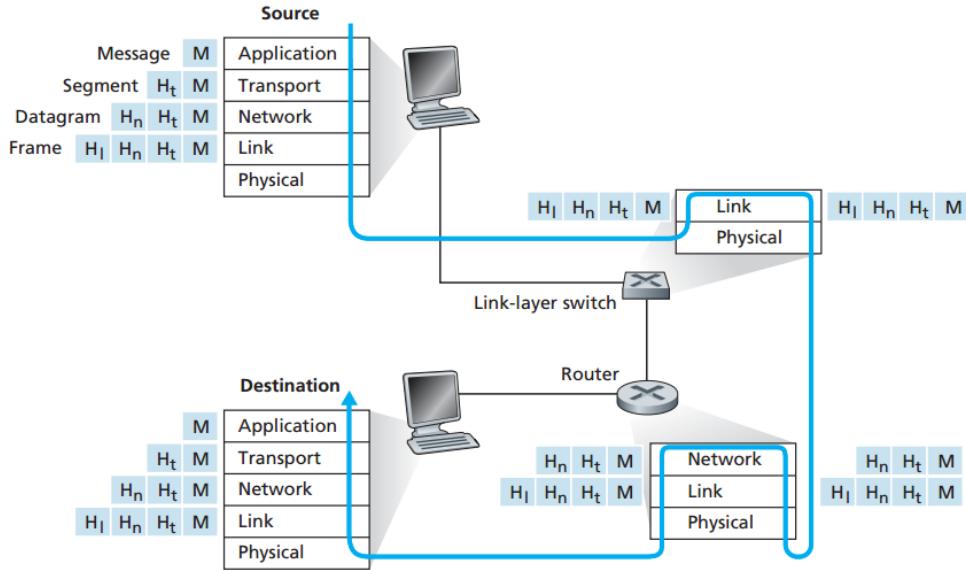
协议(protocol): 在网络实体(entities)之间传递消息的规则, 如消息的格式、收发消息的次序等。

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(protocol control data, PCI)和服务数据单元(service data unit, SDU)两部分。

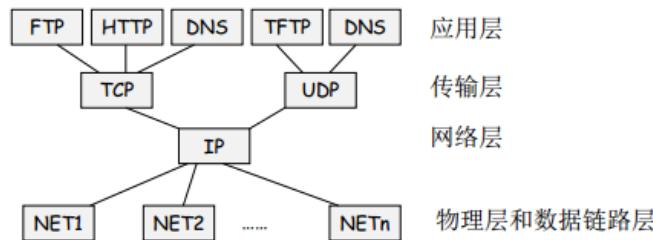


下层把上层通过服务访问点(service access point, SAP)传来的SDU用PCI封装为PDU后传给对等实体(peer entity), 即实现相同协议的实体。同一个互连网络中网络层协议需要相同, 链路层协议可以不同。

<sup>2</sup>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(encapsulation), 接收时拆封。



协议簇(protocol family)

## 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)；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：查路由，存储转发(store-and-forward)的延迟会很大
- 排队(queueing)延迟：依赖于路由器的拥塞程度
- 发送/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

- 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(throughput): 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

$$\text{总时长: } (24 + 30) \times 2 + 2 = 110ms$$

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	<b>传输</b>	<b>传播</b>	<b>接收</b>		
包2		传输	传播	<b>接收</b>	
包3			传输	传播	<b>接收</b>

- 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

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

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：

信息源 → 调制/编码 → 信道传输 → 解调/解码 → 目的地

其中调制解调为模拟信号，编码解码为数字信号。

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-传输：连续取值（连续波长而不是连续信息），放大器(amplifier)
- 数字信号/跳变信号-传输：离散取值，中继器(repeater)

### 2.1 编码方式

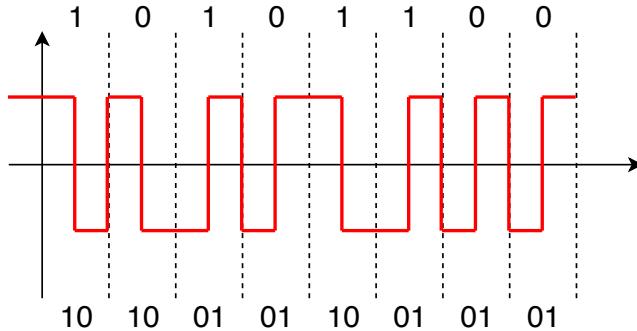
#### 2.1.1 模拟信号

载波信号(carrier)一般采用正弦波信号：角频率 $\omega$ 、频率 $f$ 、周期 $T$ 、振幅 $A$ 、相位 $\varphi$

- 频移键控(frequency-shift keying, FSK)：通过不同频率表示不同信息0/1
- 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 ~ 111

#### 2.1.2 数字信号

1. 单极编码(unipolar)：0V即0，+EV为1，但是会产生两种漂移
  - 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，解决基线漂移问题（平衡01）；全是0或全是1，还是没法区分
3. 不归零反转编码(Inverted, NRZI)：[差分](#)码波形，相邻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
4. 曼彻斯特(Manchester)编码：从相邻时刻的中间起降-E ~ +E，0 → 10, 1 → 01，[可克服时钟漂移和基线漂移](#)；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（当前编码与原数据不同）则为0，否则为1，且[中间也要跳变](#)



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**；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

### 2.2.1 分类

#### 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<sup>3</sup>
  - 屏蔽双绞线(STP)
- 同轴电缆(coaxial cable)
- 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、Wi-Fi、3G网络、卫星

<sup>3</sup>1KB(Kilobyte, 千字节), 1MB(Megabyte, 兆字节, 简称“兆”), 1GB(Gigabyte, 吉字节, 又称“千兆”)

### 2.2.2 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): 利用各路信号码型结构正交性而实现多路复用
- 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

## 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<sup>4</sup>（直连网络/物理网络）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、流控制(flow control)
- (d) 介质访问控制(medium access control): 多路访问，碰撞(collision)

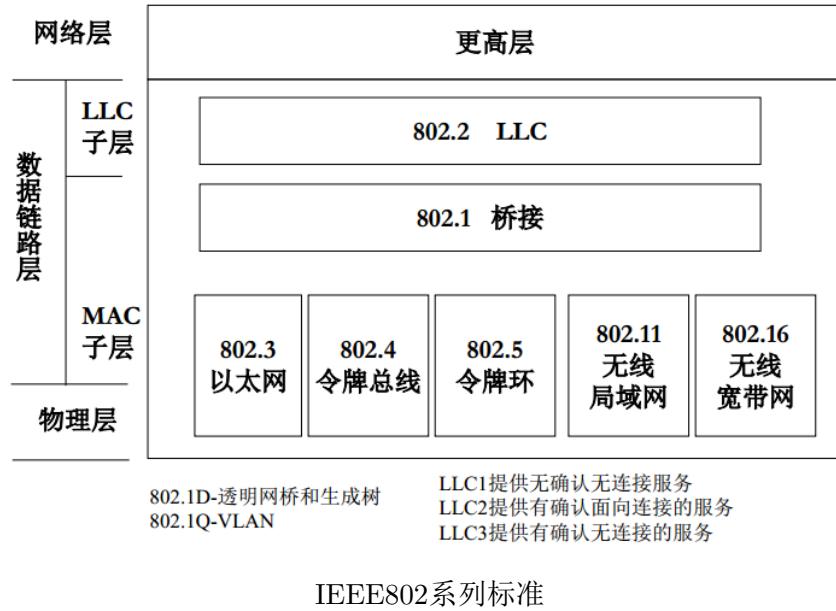
针对点对点和多路访问网络又分别制定了两个子层：

- 逻辑链路控制(Logical Link Control, LLC)子层：提供可靠数据传输
  - 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：专门用来处理多路访问网络中的冲突（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。

---

<sup>4</sup>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



### 3.1 逻辑链路控制子层

### 3.1.1 差错检测

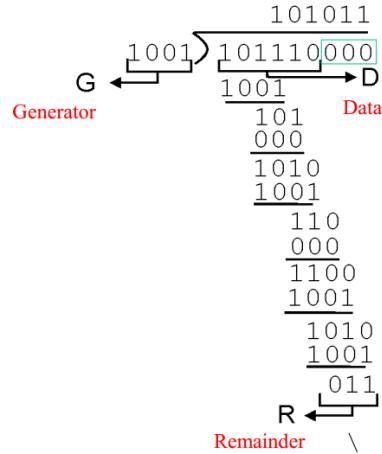
1. 奇偶校验: 若接收方收到奇数个1, 则有出错
    - 一维偶校验: 只能检错; 最后补一位使得全部为偶数个1, 如010补为010 | 1, 而101补为101 | 0
    - 二维偶校验: 检错+纠错一位; 横纵同时偶校验
  2. 校验和(checksum): 将所有数据加起来, 每16位1组, 最高位进位则末尾加1, 最后结果取反  
由于需要使用加法器, 校验和一般不用于数据链路层, 而用在更高层(网络层和传输层)

```

+ 00000100 01000100
-----
10001010 11001011
+ 11000000 00000000
-----
1 01001010 11001011
+           1
-----
01001010 11001100

```

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



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。

主要的ARQ协议包括停等协议和滑动窗口协议。

**停等协议(stop-and-wait):** 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<sup>5</sup>。三种出错情况：

- 数据帧丢失(loss): 正向传递时丢包
  - 确认帧丢失: 回传时丢包
  - 超时: 收到ACK表明接收方一定收到, 可以发送新的数据帧, 重传的也一定要发ACK

**例 2.** 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

<sup>5</sup>数据帧丢失，则重新发送该帧即可；确认帧丢失或确认帧延迟都是对方已经收到数据帧，因而为了避免重复收取同一数据帧，就需要标号。停等协议只需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接收方已经收到。

- 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 =  $1.25/20 \times 100\% = 6.25\%$

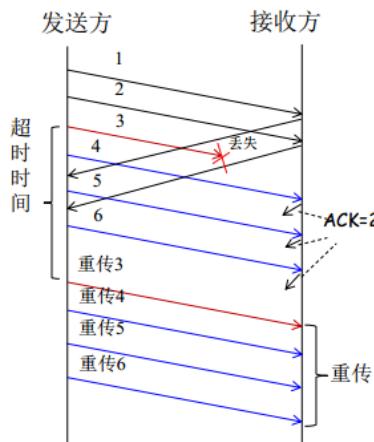
另, 改为滑动窗口协议, 窗口大小为8, 则最大利用率为  $8 \times 6.25\% = 50\%$

**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)<sup>6</sup>。

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:

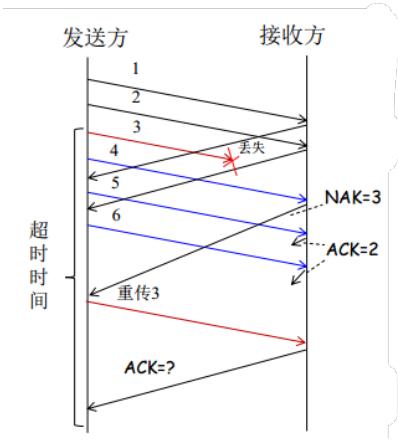
- 回退N协议(go back N): 某个ACK没收到则重传在此ACK之后的所有帧 (超时重传)



- 发送窗口需要缓存SWS个帧, 以便重传; 发送窗口中序号最小的为sendBase; 发送窗口之前的是已收到确认的, 窗口内的是已发送但仍未收到确认的
- 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
- 选择性重传(selective repeat): 通过发送否定性确认帧(negative acknowledgement, NAK)<sup>7</sup>要求重传该帧, 每个帧只发送一次NAK

<sup>6</sup>SWS即连续发送数据帧可用序号范围, 用于流控制: 控制发送速度, 否则会发生溢出(overflow), 后面覆盖前面的

<sup>7</sup>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。



- 接收窗口(receiving window size, RWS)表示接收缓冲区大小 ( $RWS \leq SWS$ , 最好是等于, 尽量减少重传帧; 但序号少的话导致重复; 错序到达的帧加上期待接收的帧最多SWS个), 用于确定应该保存哪些帧, 用序号范围表示, 会存放错序到达的帧; 其中最小序号帧是期待接受的帧(recvBase)
- 超时时间应略大于**2RTT** (否则会导致后续帧都重传<sup>8</sup>); 没有后续帧也会超时重传; 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错误, 吞吐量低; 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。

**例 3.** 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: 3467056701234

SendWin	ReceWin	操作
56	X6	发ACK=4
5670	X670	发ACK=4, 重传整个发送窗口
(5670)	(5670)	重传帧到达

选择性重传: 3467051234

SendWin	ReceWin	操作
56	X6	发NAK=5
5670	X670	因NAK还没到达, 发ACK=4
(5)670	670(5)	重传5, 要发回ACK=5, 否则发送窗口动不了
1234	1234	发送窗口右移

<sup>8</sup>如果重置所有超时定时器, 那超时时间可设为略大于1RTT

提高滑动窗口协议的效率：

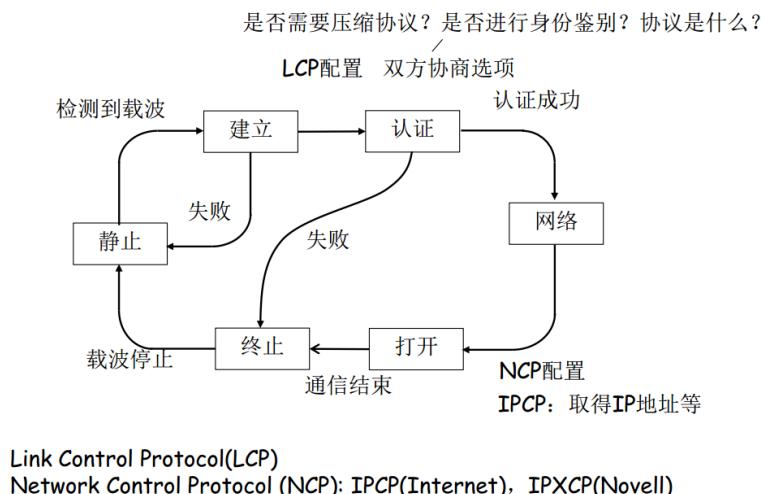
- 选择性确认(selective acknowledgement): 接受方把已收到的帧的序号告诉发送方，发送方要重传帧时，不会发送这些帧
- 搞带确认(piggybacking): 通信双方全双工方式工作，接收方在发数据给对方时顺便把确认号也告诉对方（两个滑动窗口，两边都要发数据），需要结合延迟确认一起使用
- 延迟确认(delayed acknowledgement): 接收方收到一帧后并不立即发送确认帧，而是等待一段时间再发送

## 3.2 介质访问控制子层

### 3.2.1 简介

#### 1. PPP协议(point-to-point protocol): 点到点网络

- 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
- ADSL的PPPoE和VPN中的PPTP协议都采用PPP协议进行封装



PPP协议数据帧格式

1B	1B	1B	1B-2B	$\leq 1500B$	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

#### 2. 以太网: 多路访问网络，采用共享介质连接所有站点

解决冲突问题: 随机访问协议(random access protocol)，注意不是滑动窗口**不用发确认帧**

- 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

IEEE 802.3规定以太网物理层标准:

- 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
  - 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
- 

其他几种以太网(IEEE 802.3), 主要在**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 $0.96\mu s$
  - 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
- 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)。采用载波监听CSMA/CD(with collision detection)协议 (1-坚持CSMA)。

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)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) $\tau$ 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 $\tau$ 或以上时间的两个发送数据的站点不会发生冲突。
- 时间片 $\tau$ 的长度为512b时间，10Mbps的以太网为 $51.2\mu s$
- 第 $i$ 次冲突从 $0, 1, \dots, 2^j - 1$ 个时间片随机选择一个， $i < 16$ ,  $j = \min(i, 10)$
- 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 4. 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，首先发送的站点都会引起后续所有站点冲突。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 <sup>9</sup>	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、多播或广播地址
- 源地址: 一般为发送者的单播地址
- 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

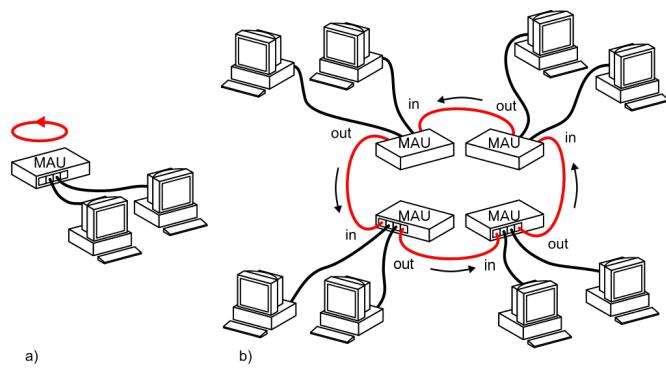
<sup>9</sup>这实际上决定了最小和最大传输单元(Maximum Transmission Unit, MTU)，为什么定这两个值可以见<https://community.cisco.com/t5/other-network-architecture/why-the-mtu-size-is-1500/td-p/105418>

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

注意：以太网没有确认机制，没有优先权。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）。

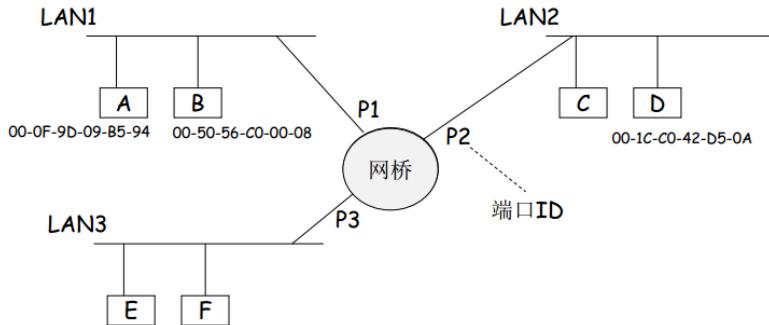
光纤分布式数据接口(Fiber Distributed Data Interface, FDDI)是另一种采用了令牌环的局域网，是一种100 Mbps的光纤局域网。

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)。



透明网桥的三个操作：

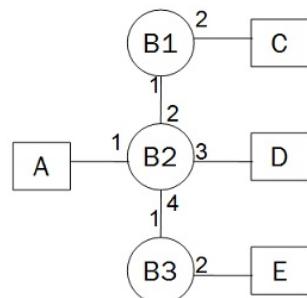
- 表里查到则转发(forward)
- 表里没有则扩散/泛洪(flood)
- 从某一条路发来则不能传回去，过滤/丢弃(filter)

透明网桥有自学习机制：利用源地址学习，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)<sup>10</sup>。如果收到的帧有错则直接丢弃，根本不会学习。如果源地址已经在表中，则更新记录，并重置超时计时器。

之所以称为透明，是因为插入网桥后无需改动硬件和软件，也无需设置地址开关、装入路由表或参数等，网桥就能工作（自学习）。

**例 5.** 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

<sup>10</sup>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）

<i>B1 MAC地址</i>	<i>端口</i>	<i>B2 MAC地址</i>	<i>端口</i>	<i>B3 MAC地址</i>	<i>端口</i>
<i>D</i>	1	<i>D</i>	3	<i>D</i>	1
<i>C</i>	2	<i>A</i>	1		
		<i>C</i>	2		

### 3.4 生成树协议

生成树协议(spanning tree protocol, STP)将所有的LAN和网桥都抽象为结点，避免冲突即构造一棵生成树（注意不是最小生成树）

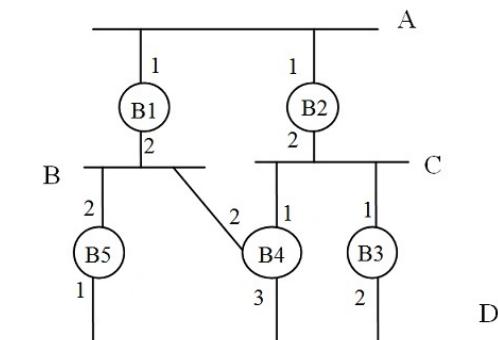
- IEEE 802.1D 生成树协议+透明网桥
- IEEE 802.1w RSTP(Rapid Spanning Tree Protocol)

工作流程如下

- 先确定根网桥，即**网桥ID(Bridge ID, BID)最小的**
- 每个网段(需要集线器)依赖于连通的网桥，每个网桥都把**自己到根的距离**发出去(竞选/配置消息)
- 网桥之间的开销为1，选一条**最短路径**
- 扩散自己BID，最后只剩下根网桥认为自己是根。取得优胜的，作为指定网桥(网段上离根最近的网桥)；相同距离时，BID小的优胜；相同BID，端口号小的优胜
- **网桥**上离根最近的端口为根端口，指定网桥上与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
- 只有**从根端口过来**的才扩散配置消息，其他端口来的不扩散，这样不会形成回路
- 断了/失效了则变成无穷大，其他网桥可成为指定网桥

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

例 6. 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

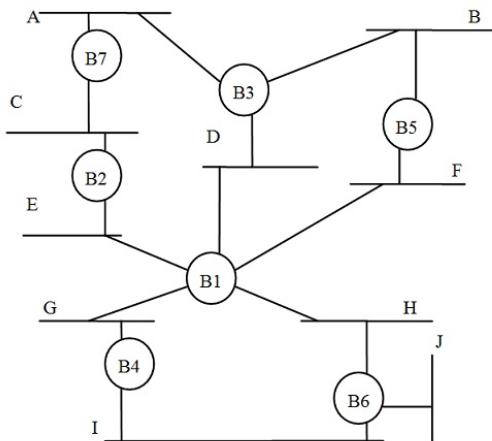
2. 网段*A-D*的指定网桥(*designated bridges*)分别是

	A	B	C	D
指定网桥	B1	B1	B2	B4

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

例 7. 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生成树消息(BPDU)，只生成1棵生成树，根为B2
  - 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

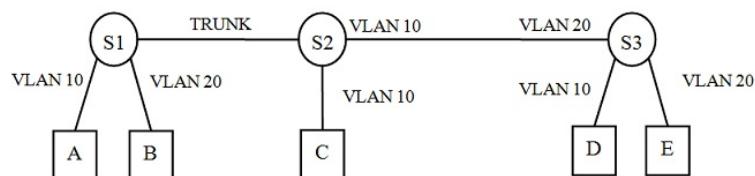
### 3.5 虚拟局域网

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，只在具有相同标号(VLAN ID)的端口间转发。

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。

每次扩散到帧内指定的端口或[干道端口](#)，同样查MAC地址表转发。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 8. 下图中哪些发送的帧将被目的主机收到



分析. 只有A到E或E到A可以成功发送信息，注意S2和S3的端口设错了（故意的）。如E到A，VLAN20经过S3转发到VLAN20，发到S2。S2误认为是从VLAN10发来的消息，故扩散到干道端口TRUNK加VLAN10，发到S1。S1接收到后转发至VLAN10。

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

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

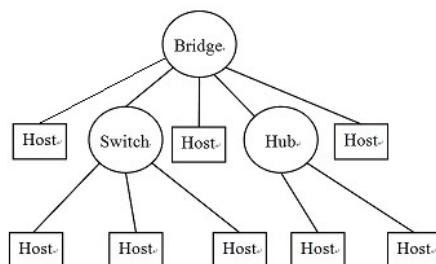
- 公共生成树(common spanning tree, CST)
- 多生成树(Multiple spanning tree protocol, MSTP)

### 3.6 物理设备

集线器(hub)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)。

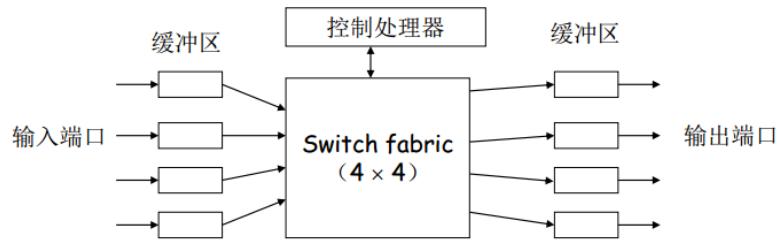
例 9. 下图的冲突域和广播域个数？



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。故8个冲突域，1个广播域。

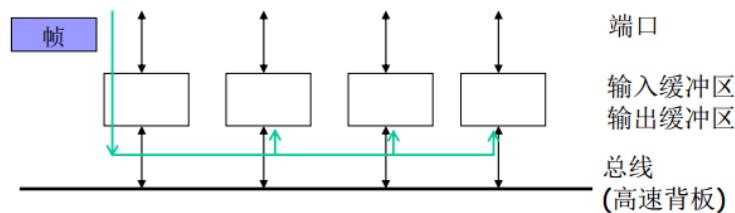
---

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。

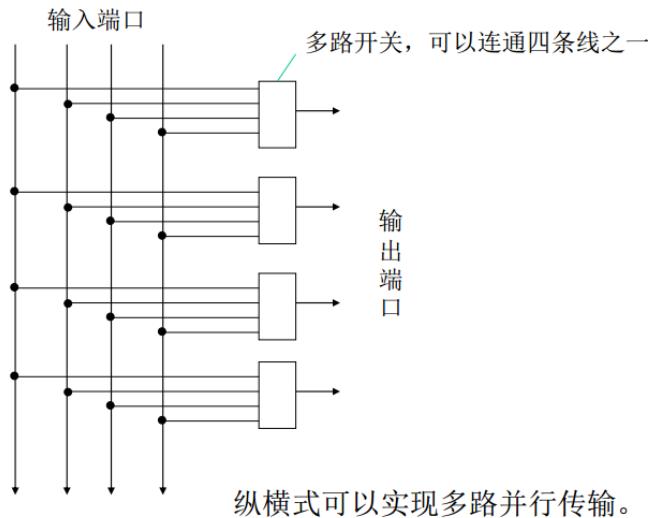


### 交换结构(fabrics)

- 共享总线式: 存在冲突问题



- 纵横式(crossbar): 可实现多路并行传输



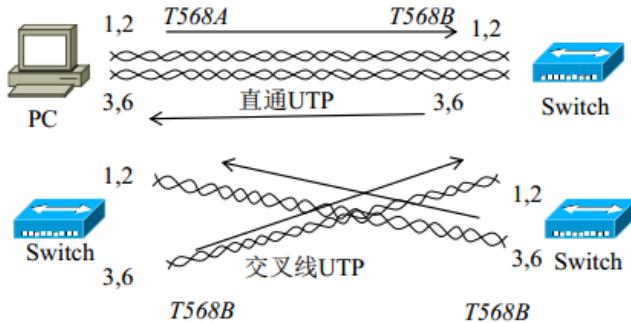
### 交换机转发方法

- 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, 转也要依照CSMA/CD来转发
- 直通(cut through): 收到帧的硬件地址后立即转发它, 出现碎片
- 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)**后, 都没冲突才开始转发

- 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<sup>11</sup>:

- 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。

<sup>11</sup>[http://www.qianjia.com/html/2017-08/09\\_274208.html](http://www.qianjia.com/html/2017-08/09_274208.html)

## 4.1 IP数据报

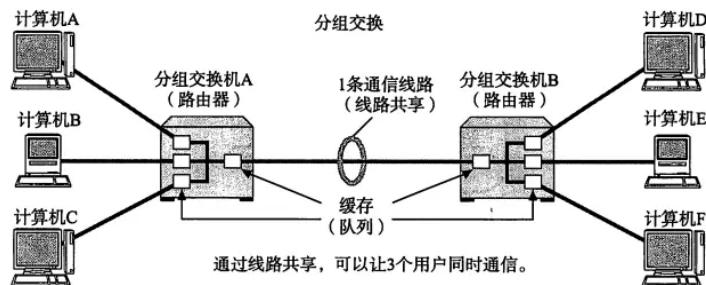
### 4.1.1 一般网络的服务模型

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

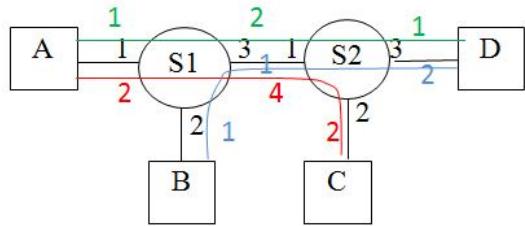
### 4.1.2 数据交换技术

- 电路交换(circuit switching): 实际接通一条物理线路, 如电话 (时分多路复用)、电视 (频分多路复用)。一直占用, 不管有无数据交互
- 包交换/分组交换<sup>12</sup>(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, 虚电路标识符(Virtual Circuits Identifier, VCI), 类似于电话
    - \* 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, 因特网, 不预留带宽



例 10. 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。

<sup>12</sup>所以IP数据报也被称为IP分组(packet)



分析. 交换机  $S_1$  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  $S_2$  的虚电路表

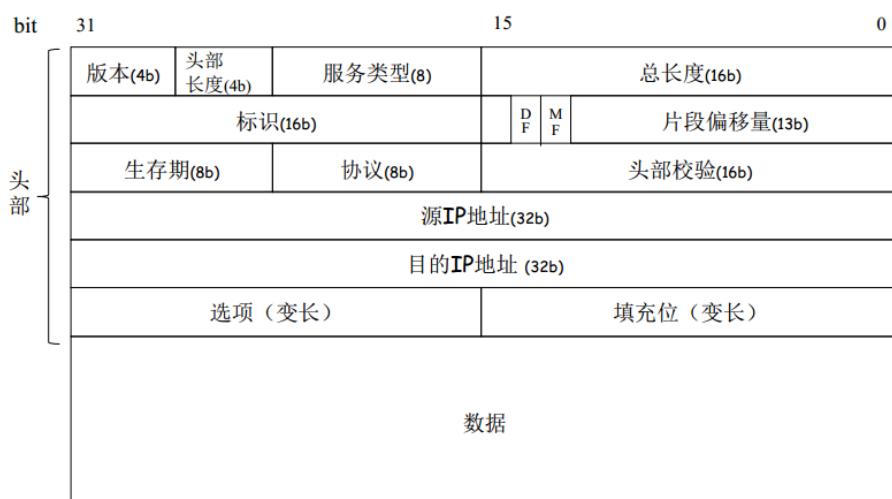
	输入接口	输入 VCI	输出接口	输出 VCI
红	1	4	2	2
绿	1	2	3	1
蓝	1	1	3	2

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 (三层)
- 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的两倍, Windows默认64, Unix默认为255
  - 长了就有捷径(cut-through), 因此发展到现在因特网的直径依然在32左右, 即TTL限制了因特网的直径
- 协议: TCP为6, UDP为17, ICMP为1, IGMP为2
- 头部校验: 将校验和字段置为0, 然后对IP包头进行校验和运算。如包头长度不是16b的倍数, 则用0填充。路由器会丢弃出错的数据报。
- 头部选项: 最后一定要对齐到边界。

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

- 头部长度只有4b, 以字为单位, 头部最多 $(2^4 - 1) \times 4 = 60\text{B}$ , 除去非选项部分 $4 \times 5 = 20\text{B}$ , 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, 通过物理层传输, 每次都要修改源和目的地址。在以太网帧的类型/长度字段填0x0800表明是IP数据报

#### 4.1.4 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)
- 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，其他不变
- IPv6中间不能分段

**例 11.** 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 $35 \times 8 = 280B$ ，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 12** (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

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
- 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，类似变长指令集，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)，其子网号就是最小一个的网络号。可以显著减少路由表中路由的数量，称为**路由聚合**(route aggregation)。
- 网络地址转换(network address translation, NAT)：**最节约地址的方法**，将内部地址映射为外部地址的技术（可以扩展6w多倍），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。**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

**例 13.** 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。

#### 4.2.4 IPv6

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

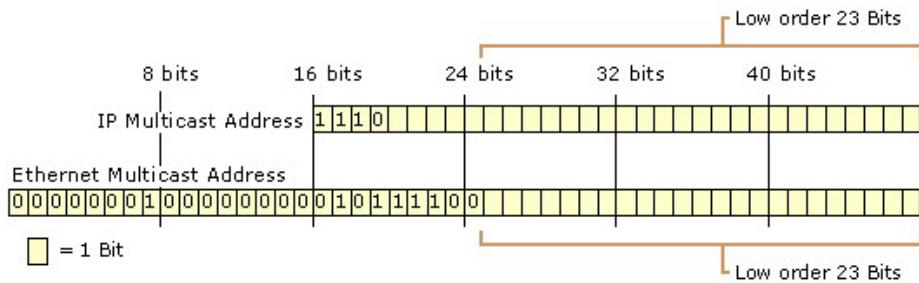
多种IPv4过渡IPv6方式

- 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
- 在IPV4和IPV6之间进行NAT转换

### 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



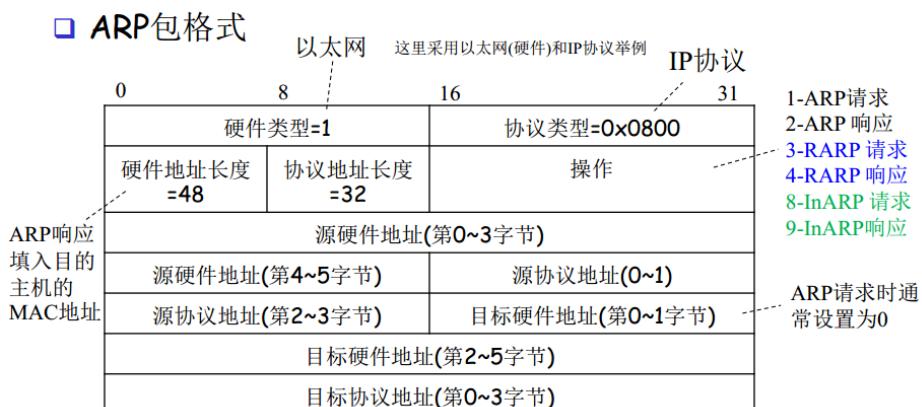
如224.0.1.5写为16进制是0xE0-00-01-05，取低23位，得到MAC完整地址为0x01-00-5E-00-01-05

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

- ARP请求广播帧（谁的IP地址是XXX），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

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

- 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
- 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



带有ARP包的以太网帧：

Preamble	Dest. Addr.	Src. Addr.	type=0x0806	ARP分组	CRC
----------	-------------	------------	-------------	-------	-----

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)**用于主机在加入网络时动态租用IP地址，基于UDP，四个步骤如下：

- 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)

### 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）：超时是回来的路找不到；而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）：用目的地址未查到匹配的路由项、需要分段但不可分段(DF=1)、网络/主机/协议/端口不可达，数据部分填原IP头部+原IP数据部分的头64b
- 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 14.** 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 15.** 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
- 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, 因为更详细
- 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, 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**, 封装成帧发送, 要遵守**以太网协议(CSMA/CD)**
  - 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), 封装成帧发送

- 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 16. 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

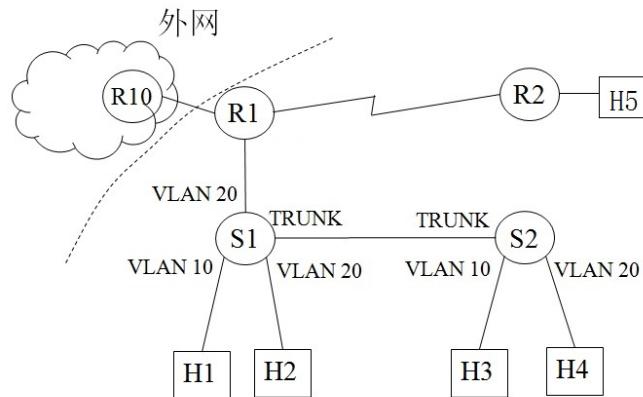
例 17 (实战分析). “在链路上”意味着是**直连网**，“网关”相当于**下一跳**，“接口”直接**用IP地址标注**

IPv4 路由表					
活动路由:	网络目标	网络掩码	网关	接口	跃点数
	0.0.0.0	0.0.0.0	172.19.127.254	172.19.108.182	35
	127.0.0.0	255.0.0.0	在链路上	127.0.0.1	331
	127.0.0.1	255.255.255.255	在链路上	127.0.0.1	331
	127.255.255.255	255.255.255.255	在链路上	127.0.0.1	331
	172.19.64.0	255.255.192.0	在链路上	172.19.108.182	291
	172.19.108.182	255.255.255.255	在链路上	172.19.108.182	291
	172.19.127.255	255.255.255.255	在链路上	172.19.108.182	291
	192.168.56.0	255.255.255.0	在链路上	192.168.56.1	281
	192.168.56.1	255.255.255.255	在链路上	192.168.56.1	281
	192.168.56.255	255.255.255.255	在链路上	192.168.56.1	281
	224.0.0.0	240.0.0.0	在链路上	127.0.0.1	331
	224.0.0.0	240.0.0.0	在链路上	192.168.56.1	281
	224.0.0.0	240.0.0.0	在链路上	172.19.108.182	291
	255.255.255.255	255.255.255.255	在链路上	127.0.0.1	331
	255.255.255.255	255.255.255.255	在链路上	192.168.56.1	281
	255.255.255.255	255.255.255.255	在链路上	172.19.108.182	291

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.  $172.19.64.0/18$ : 校园网内部网络, 通过默认网关发出
5.  $192.168.56.1/32$ : VirtualBox 虚拟机网卡接口 IP
6.  $224.0.0.0/4$ : 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 (接口不一样, 因此要在路由器里写两项, 一项内部一项外部)
7.  $255.255.255.255/32$ : 在无线网络中广播

**例 18 (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，逐层拆封，取出IP分组，向上传递。

所以针对以下几种情况，可以得到对应使用的协议

- $H1 \text{ ping } H3 \rightarrow H1:9653 \text{ S1:3123 S2:3213 H3:8}$   
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。
- $H1 \text{ ping } H4 \rightarrow H1:9643 \text{ S1:3123 S2:328653 H4:8}$   
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
- $H1 \text{ ping } H5 \rightarrow H1:9643 \text{ S1:3123 S2:3286423 S1:3213 R1:867 R2:8653 H5:8}$   
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。
- $H1 \text{ ping 外网} \rightarrow H1:9643 \text{ S1:3123 S2:3286A423 S1:3213 R1:86AB43 R10:386A\dots$   
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。

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。所建立的路由分别称为静态路由和动态路由。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。

- 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，EGP协议和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。

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): Dijkstra → 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。

- 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
- 如果路由已经存在且下一跳就是邻居，则必须进行更新

3. 如果路由存在，就要重置失效定时器(invalid timer)，更新TTL(Time-To-Live)

**例 19.** 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	-

记得要更新下一跳的内容！

---

### RIP协议存在的问题

- 慢收敛：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的距离改为16（无穷大）之后，R2将路由表发给R1

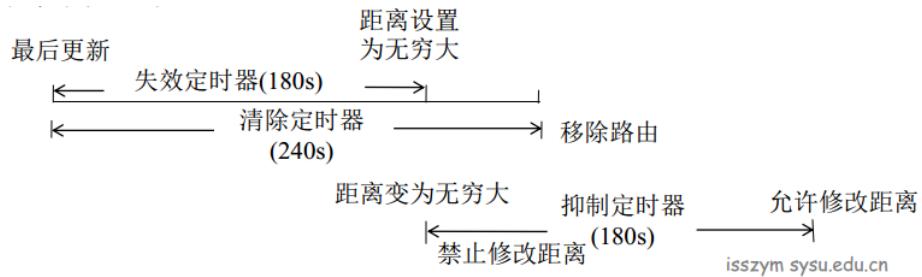
解决方案：

- 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，三角形R1断了，R1先发，R2后发
- 毒性反转(poison reverse)技术：当一条路由变为无效之后，路由器并不立即将它从路由表中删除，而是将其距离改为用16后广播给邻居，使邻居所拥有的该路由立即失效，而不是等待TTL到期后删除，以迅速消除路由环路，这种方法称为毒性反转，距离为16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

- 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)：到期时被标记为无效路由（距离改为16）。路由被更新时其失效定时器会被重置，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。



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

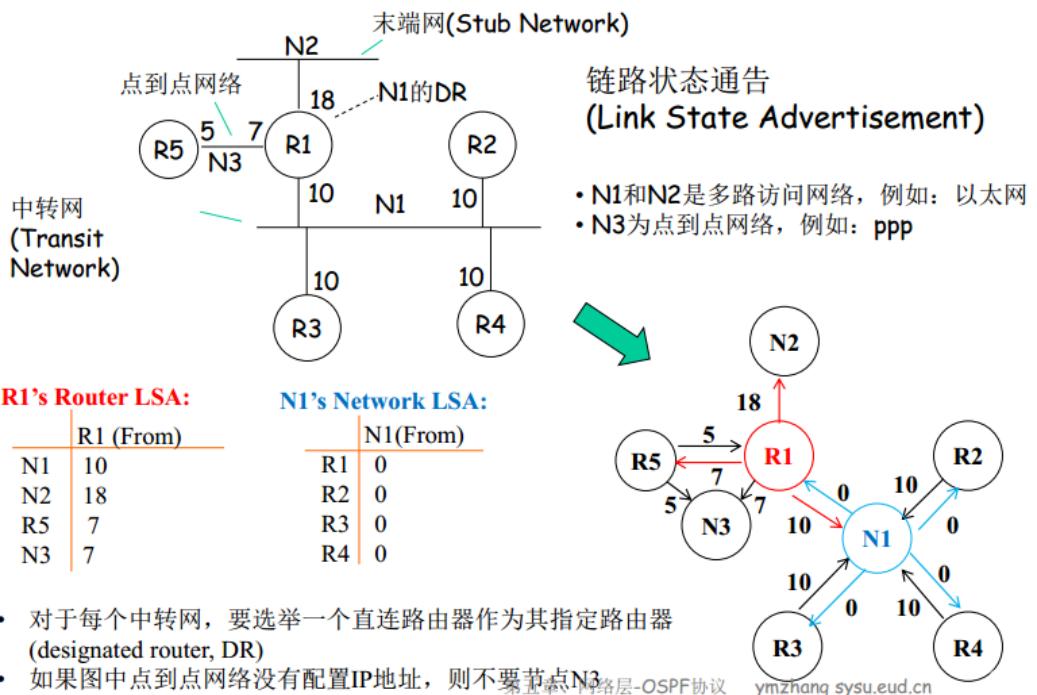
#### 4.5.2 OSPF协议

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。

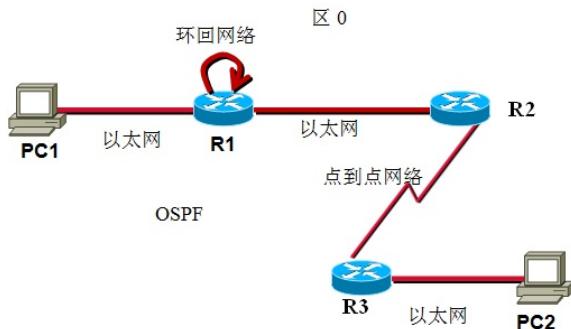
- 周期性地收集链路状态，并扩散给AS中的所有路由器
- 用收到的链路状态建立整个AS的拓扑结构图
- 利用Dijkstra算法计算到AS中所有网络的最短路径
- 利用这些路径上的下一跳建立路由表

OSPF第一步需要将整个网络(AS)转化为AS的拓扑结构图。

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

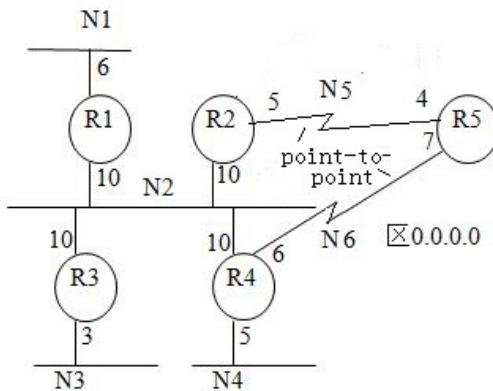


详细过程如下

- 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 21. 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!

<i>R1 LSA</i>	开销	链路类型	<i>R2 LSA</i>	开销	链路类型	<i>N2 LSA</i>	开销
<i>N1</i>	6	末端网	<i>R5</i>	5	点到点	<i>R1</i>	<b>0</b>
<i>N2</i>	10	中转网	<i>N5</i>	5	<b>末端网</b>	<i>R2</i>	0
			<i>N2</i>	10	中转网	<i>R3</i>	0
						<i>R4</i>	0

*R5*的路由表，中转网只算一次开销！总开销直接目测！

目的	开销	下一跳	目的	开销	下一跳
<i>N1</i>	20	<i>R2</i>	<i>N4</i>	12	<i>R4</i>
<i>N2</i>	14	<i>R2</i>	<i>N5</i>	4	-
<i>N3</i>	17	<i>R2</i>	<i>N6</i>	7	-

OSPF协议采用路由器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, Dijkstra)	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 <sup>13</sup>	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。

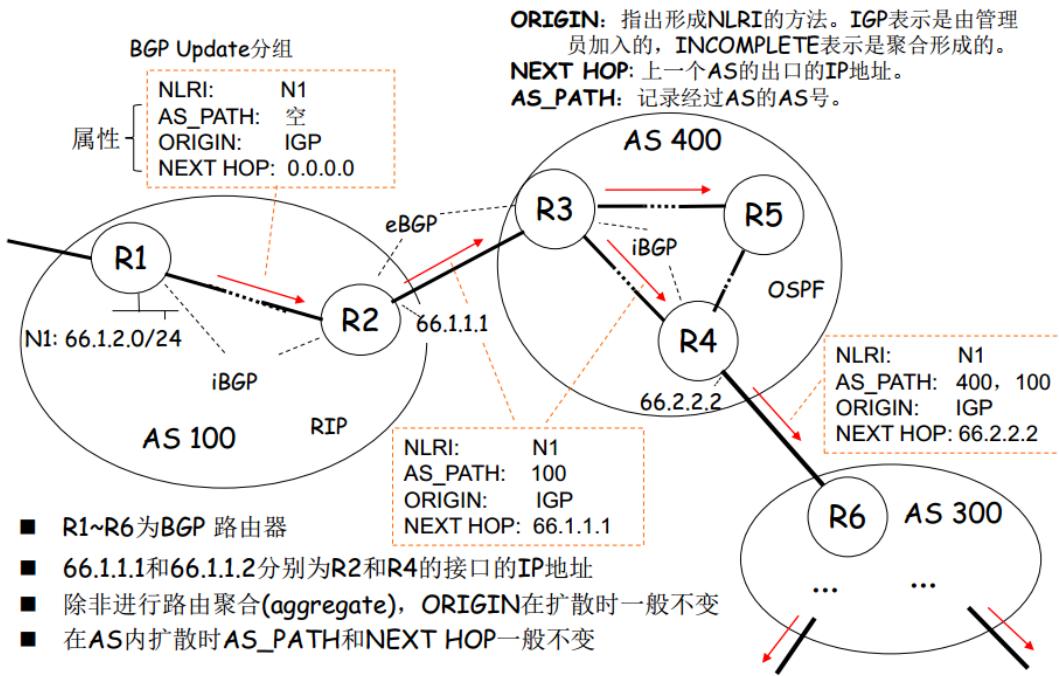
---

#### 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
- 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 Reachability 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

---

<sup>13</sup> 健壮性指路由器失效时会出现的情况



- 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，或者通过隧道技术连接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
- 内部路由如何发到外部？
- 查next hop，BGP注入IGP路由，但非BGP路由承受不住；或将路径上所有路由都变为BGP路由
  - 在IP层运用了虚电路（等于挖了个隧道），不用记中间路由器，现在都用这种方式

防止出现回路的方法：

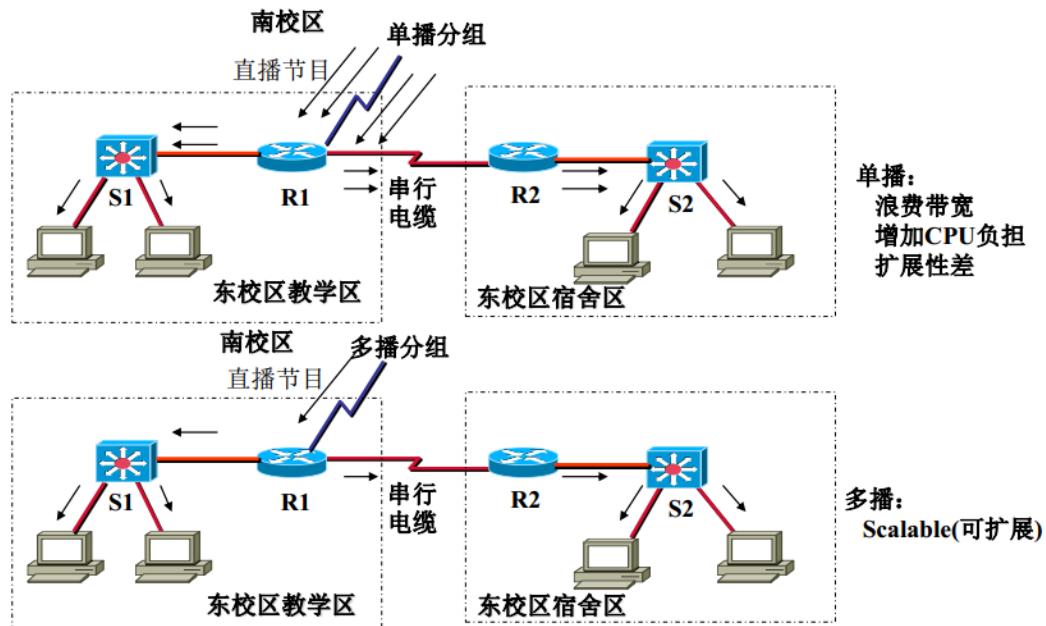
- 内部：从iBGP邻居收到的NLRI Update分组不能再发给邻居
- 之间：利用AS Path防止回路

## 4.7 IP多播

### 4.7.1 概述

访问同一个网站，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）



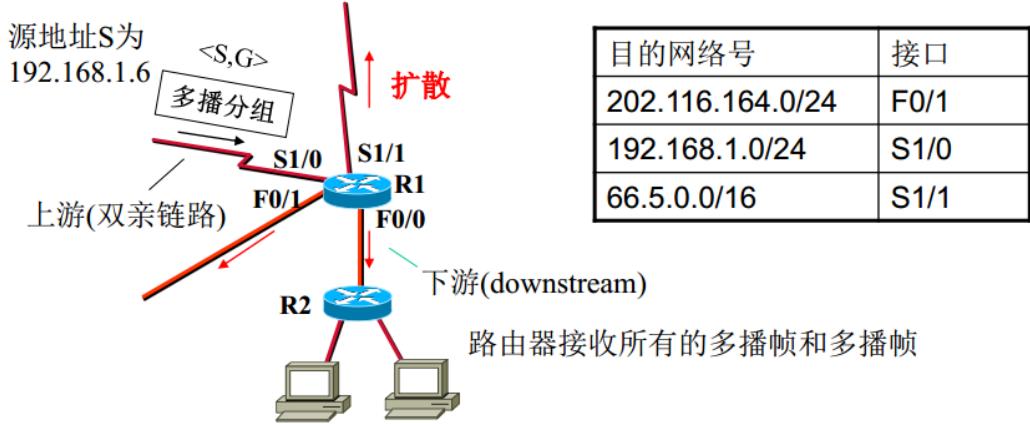
### 4.7.2 多播IP地址

多播地址范围	用法
<b>224.0.0.0 - 239.255.255.255</b>	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)



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

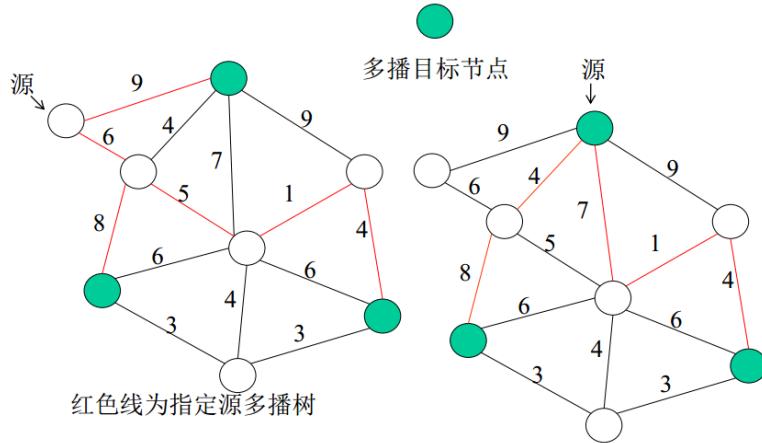
- 逆向路径多播：对于每个多播地址G，每个源地址形成一棵树，即源特定组播树(source-specific multicast tree)，然后剪枝

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，MOSPF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。

## 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。

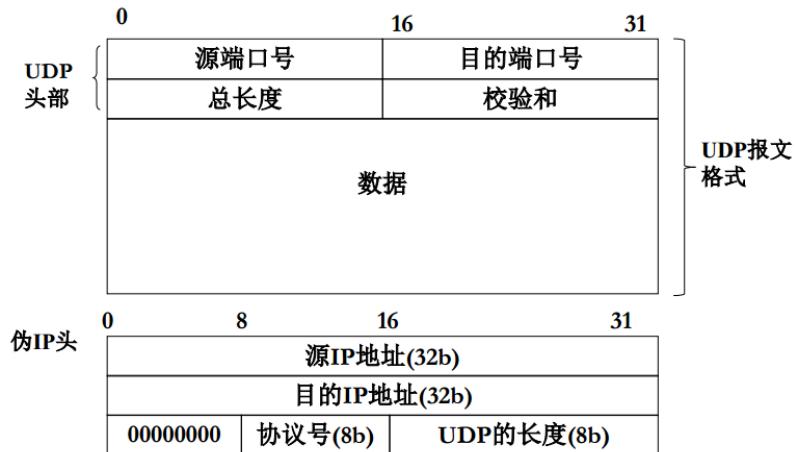
## 5.1 UDP协议

### 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



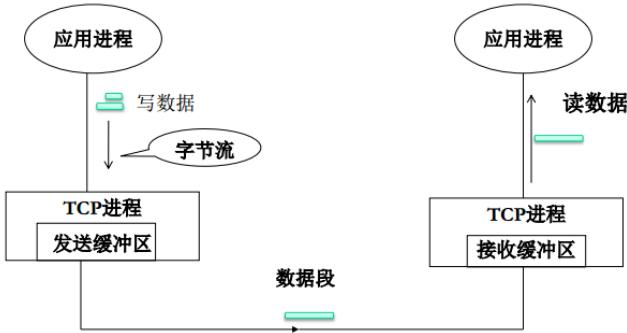
- 总长度：整个UDP报文长度
- 源端口号和目的端口号(2B)：用于关联发送进程和接收进程
- 校验和：由**伪IP头**（只用来算校验和，不进行传递）、**UDP头**（校验和用0填充）和**UDP数据**形成。其中，伪IP头的协议号为17。如果发送方把校验和设置为0，接收方会忽略校验和。UDP长度就是UDP头部填写的总长度。

## 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
- TCP是面向字节流的传输方式，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)<sup>14</sup>。
- 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

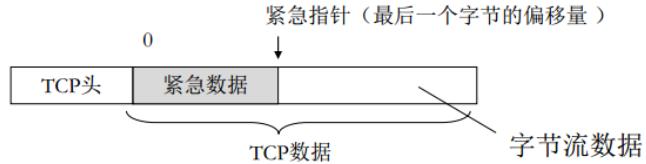
### 5.2.2 数据段格式



- 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。
- 头部长度：以字(32b)为单位，同IP数据报。
- 标志：标记为1时有效

<sup>14</sup>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 $1500 - 20 - 20 = 1460$ 。

- 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
- 通知窗口(advertised window, advWin)大小: 接收方告知发送方, 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)<sup>15</sup>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。



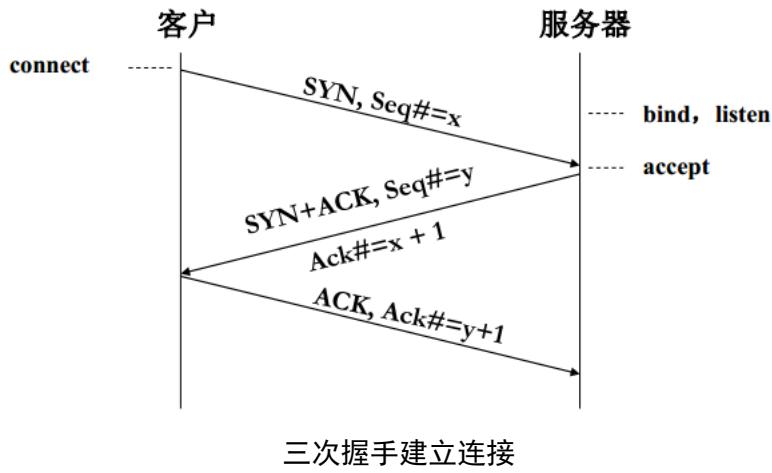
- 选项: 建立连接时有MSS、窗口比例(Scale)、是否使用选择性确认(SACK-Permitted)等。数据传送时有选择性确认的序号范围(Selective ACK, SACK)、时间戳等。

### 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

<sup>15</sup>带外数据不属于字节流



- 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）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。
  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。
- 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 22. 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。

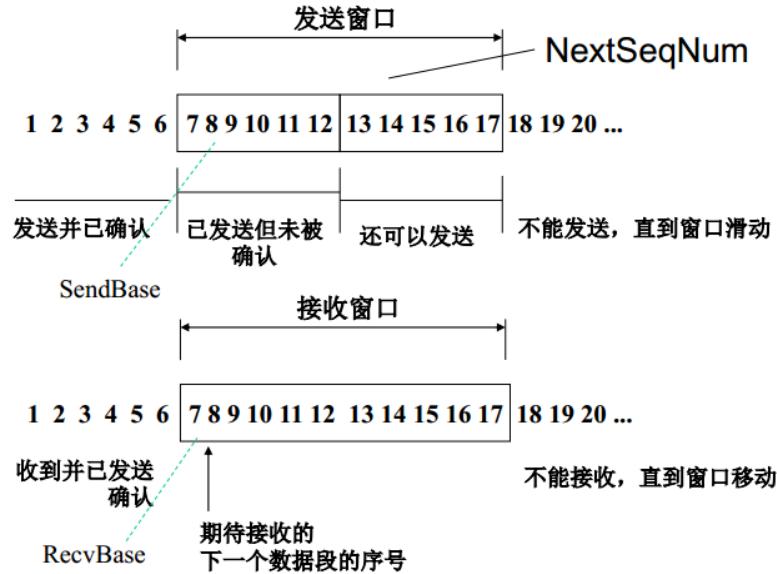
即使是三次握手也无法避免DoS攻击和DDoS攻击，因为依然可以实现大量客户端同时向服务器发送连接请求，然后接收下服务器发来的确认数据包后，不再向服务器发送确认数据包（即客户端主动不进行第三次握手），那么服务器在短时间内会维护这样的半连接队列，等待队列中的客户端确认。但由于每个半连接都会耗费服务器的资源，故最终会导致资源不够用，拒绝正常的连接请求，使得正常服务无法提供。

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。

---

TCP协议传输数据采用**选择性确认**协议（**滑动窗口**），不使用NAK。只有一个**超时定时器**。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。

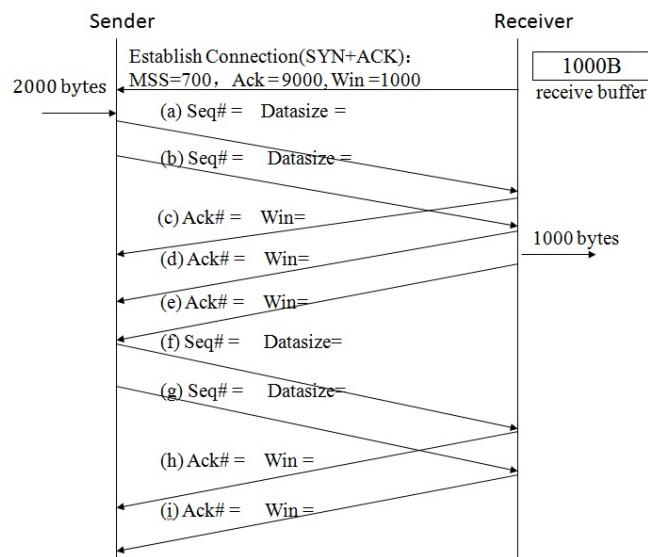
注意TCP协议中没有说明如何处理错序到达的数据段，要取决于具体实现。



\* 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)

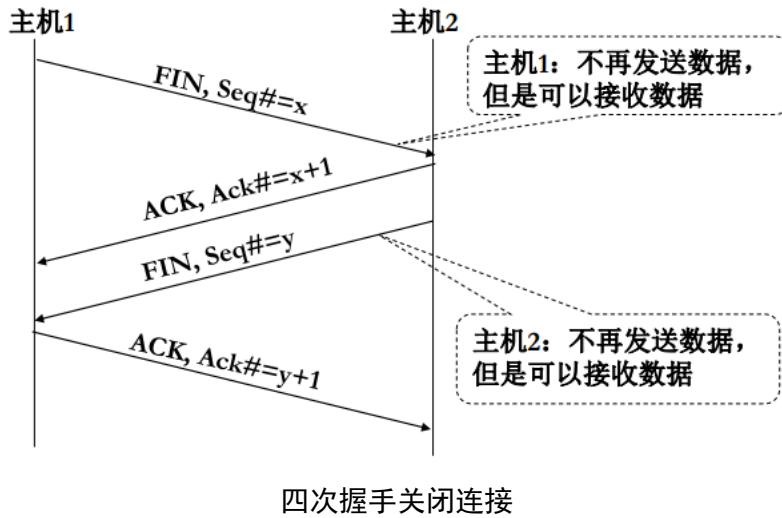
- 接收方先传MSS, x和接收/通知窗口大小(advWin)
- 发送方做发送窗口, 序号为x+1, 大小等同advWin (接收窗口和发送窗口大小都会改变, 流控制)
- 若接收缓冲区已满, 则要等接收方进程将缓冲区取空, 发送方才能继续发, 否则发送窗口大小始终为0
- 超时定时器会自动移动

**例 23.** 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

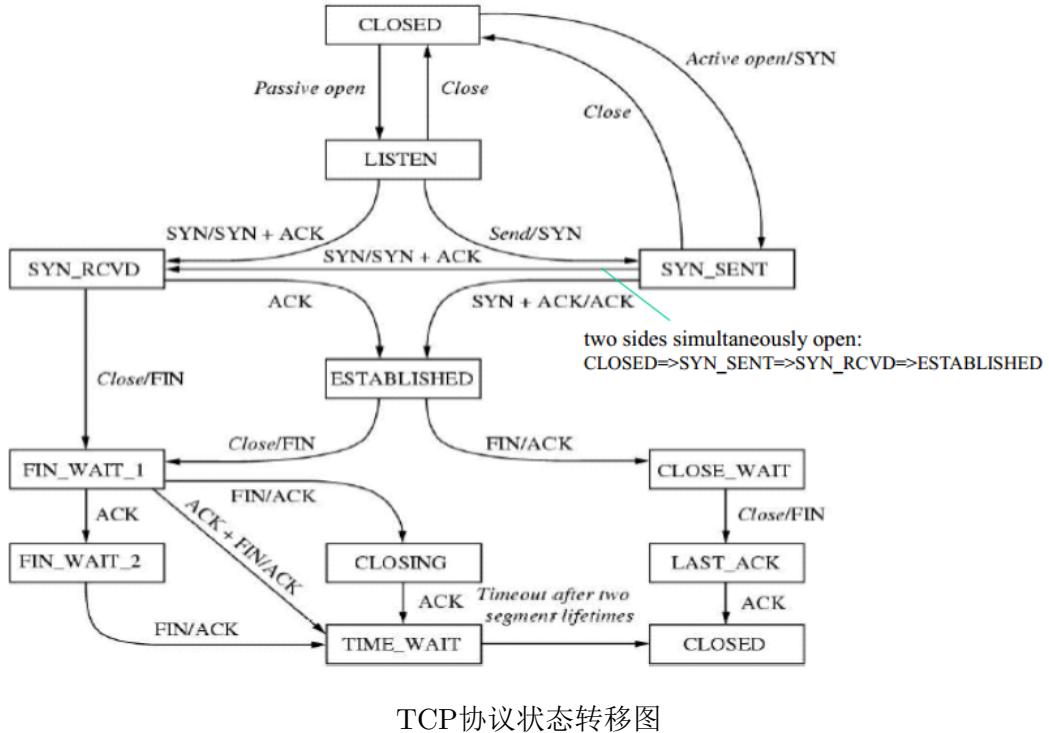


分析. 注意每次发送数据最多就只有MSS, 但发送窗口大小可以大于MSS。

- (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, 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
- 



- 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**。TCP标准中MSL采用60秒，Unix采用30秒。
  - 可以合并中间两次握手(ACK和FIN)或两方同时发出ACK。
-

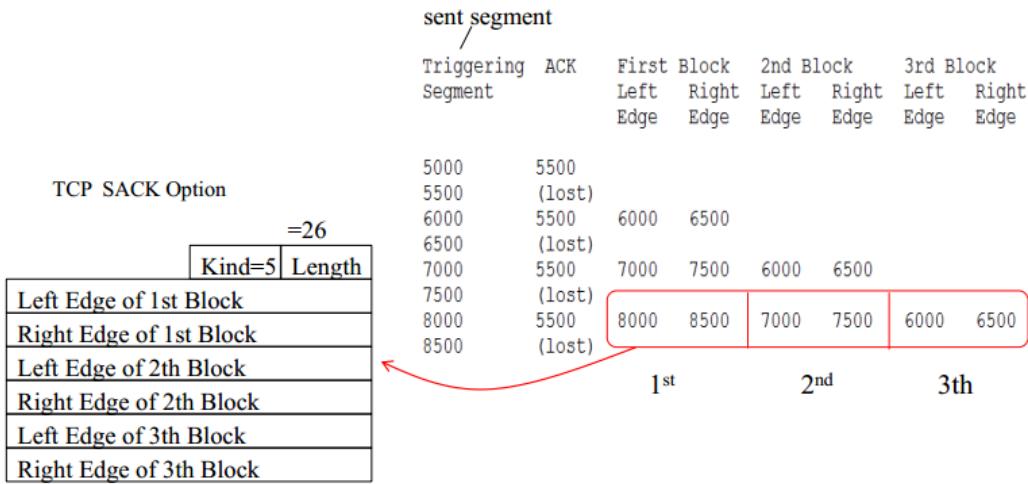


TCP协议状态转移图

#### 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,  $0 < \alpha < 1$ ,  $\alpha$ 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。

$$RTO = \gamma \times RTO$$

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 24.** 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，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：

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

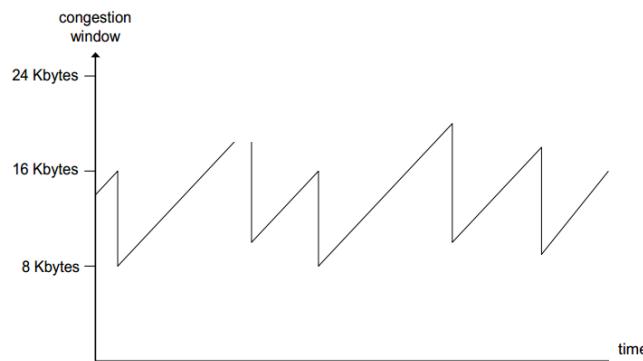
- 通过引入**拥塞窗口变量CongWin**来限制SWS。

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

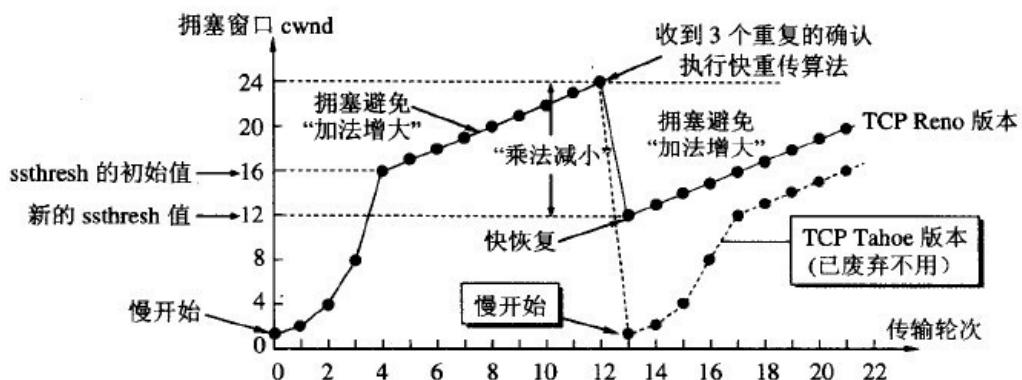
CongWin为发送窗口的流量控制， AdvWin为接收方要求的流量控制

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**



- 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, 上图有误)。
  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

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

- **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)

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

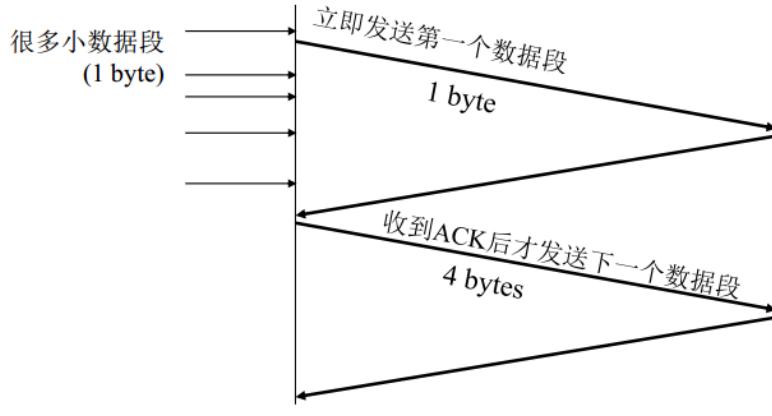
- **发送窗口太小**, 满足不了需求: 当丢包发生时, 由于SWS的限制, 管道将被清空。解决方法: 快速重传、快速恢复。(数据链路不会, 因为直连网)
- 

**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

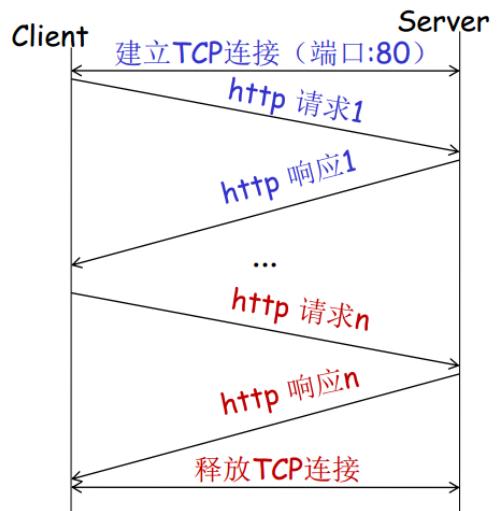
### 6.1 HTTP协议

#### 6.1.1 概述

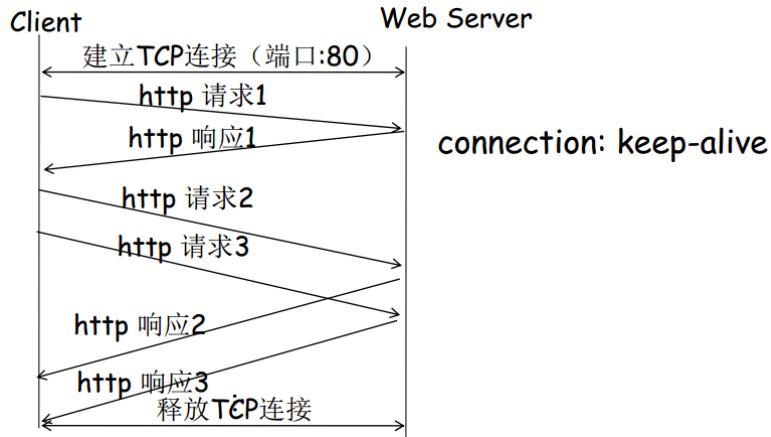
- 网页(Web page)是由对象(objects)构成的。这些对象可以是HTML文件、 JPEG图像文件、 MP4视频文件等。
- 网页的HTML文件中指出了所需的其他对象。
- 每个对象采用URL指明存放地址。URL由主机名和路径名构成。

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



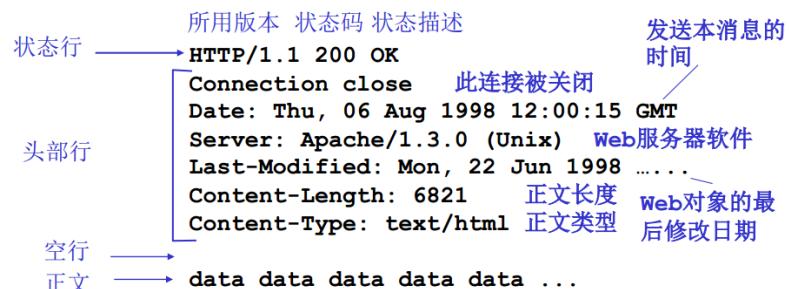
### 6.1.2 消息

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

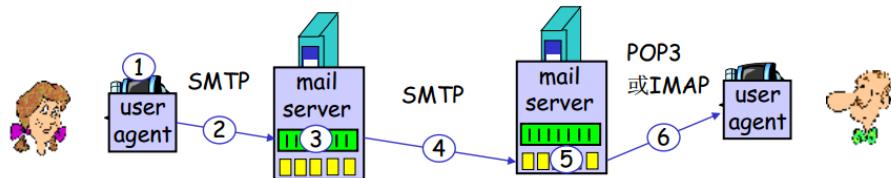
```

### 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

- 1) Alice使用用户代理编写消息给 bob@someschool.edu
- 2) Alice用代理把她编写的消息发送给邮件服务器; 消息放在消息队列中排队
- 3) Alice的邮件服务器与Bob的邮件服务器建立TCP连接
- 4) Alice的邮件服务器把Alice的消息发送给Bob的邮件服务器
- 5) Bob的邮件服务器把这个消息放在Bob的邮箱中
- 6) Bob采用POP3协议通过他的用户代理读取该邮件消息。



### 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

例 25. 如果客户希望获得`www.amazon.com`的IP地址：

- 客户先到根服务器上查询`com`DNS服务器的IP地址
- 然后到`com` DNS服务器上查询`amazon.com`的DNS服务器
- 最后查询`amazon.com` DNS服务器获得`www.amazon.com`的IP地址

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

#### 6.4.3 资源记录

采用分布数据库保存资源记录(resource records, RR)

(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

#### 7.1.2 设施

- 无固定设施(自组织): 独立基本服务集(Basic Service Set, BSS), 不需要路由器, 可以几台PC直接连接
- 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 底层配置

MAC层一般用分布协调功能(DCF)

- 原子操作: 发完数据等待 $28\mu s$ 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
- 

- 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

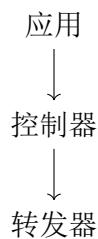
SNMP协议: Simple Network Management Protocol

### 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**

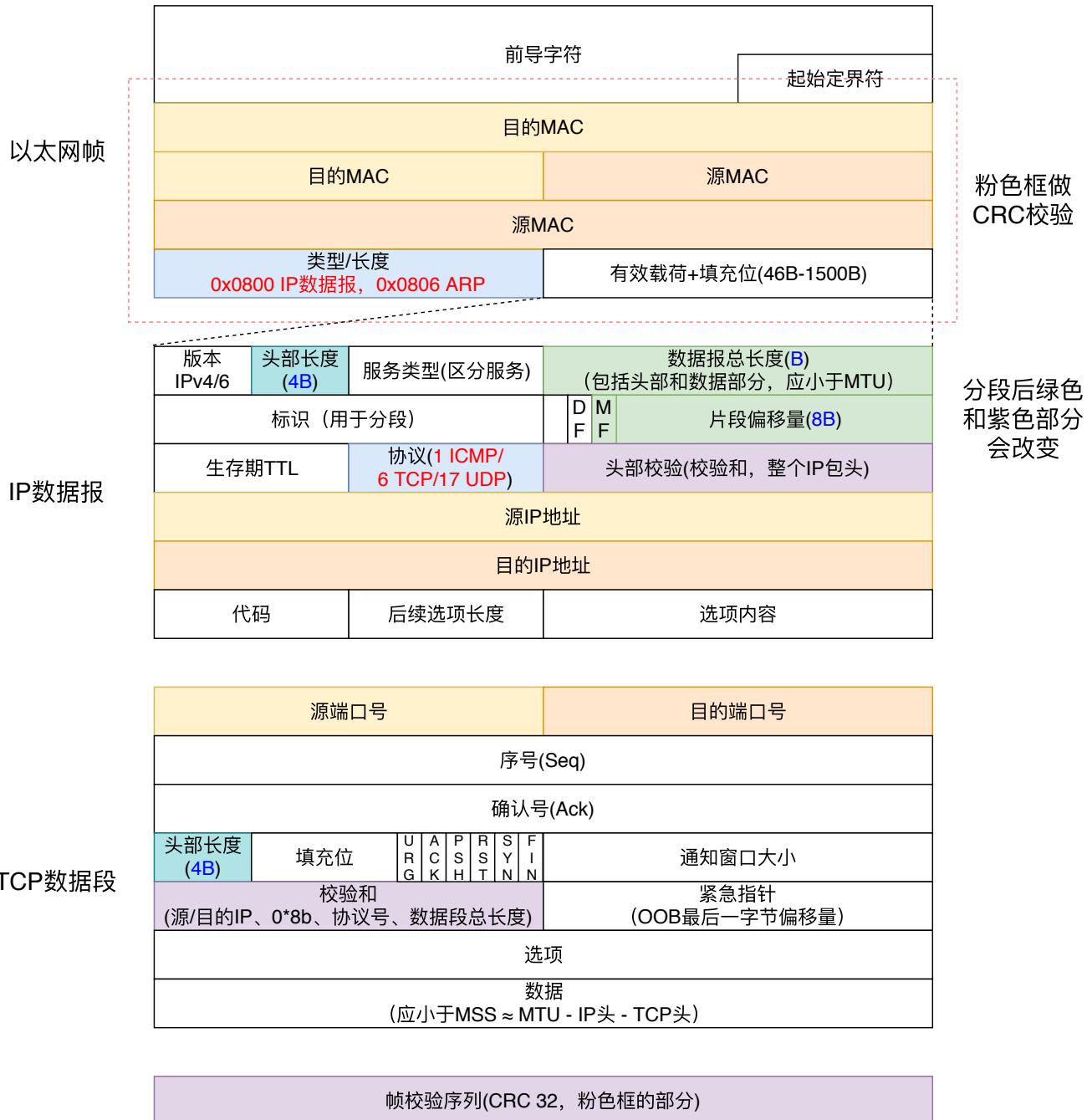


控制器相当于一个电脑，用来修改下面（几十台）转发器的路由表

### 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&>