计算机网络笔记

陈鸿峥

2019.06*

目录

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）》。

*Build 20190612

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

1.1.1 直接连接的网络

- 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

1.1.2 间接连接的网络

- 中间节点、路由器(router)
- 包(packet)
- 存储转发(store-and-forward)
- 路由选择(routing)
- 路由表(routing table)
- 目的地(destination)、下一跳(next hop)

1.2 因特网

网络互连：用路由器（或网关gateway）连接起来构成的网络称为互**连**网络(internetwork)。用实际的物理通信介质及相应的设备把两个或两个以上的网络连接起来的一种网络，如LAN和WAN都可看作是互连网络。

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[1]：

- 应用层：提供对某些专门应用的支持，如FTP、SMTP、HTTP
- (OSI)表示层(presentationn)：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hop，结点到结点），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。

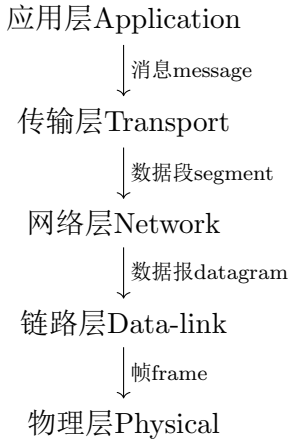
协议(protocol)：在网络实体(entities)之间传送消息的规则，如消息的格式、收发消息的次序等。

协议栈：发送时封装(encaptulation)，接收时拆封。

协议簇(protocol family)：应用层FTP、HTTP、DNS，传输层TCP/UDP，网络层IP。

[1]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

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。路由做的事情是拆一层封装，然后重新加一层。同一个互联网络中网络层协议需要相同，链路层协议可以不同。

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)技术则延迟很大
- 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

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

总时长：$(24 + 30) * 2 + 2 = 110ms$

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

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

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：$24 + 30 * 2 = 84ms$

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：

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

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

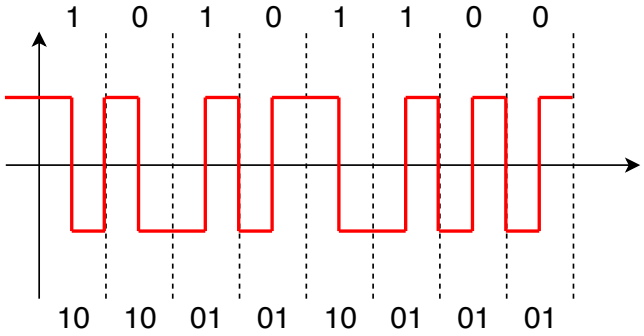
2.1.1 模拟信号

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$000 \sim 111$

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
   - 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$-E \sim +E$，$0 \to 10, 1 \to 01$，**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[2]
     - 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、WiFi、3G网络、卫星

2.2.2 多路复用

- 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

[2]1KB(Kilobyte，千字节)，1MB(Megabyte，兆字节，简称“兆”)，1GB(Gigabyte，吉字节，又称“千兆”)

3 数据链路层

数据链路层把数据包，即**帧(frame)**，从一个节点通过链路（直连网络或**物理网络**）传给相邻另一个节点（主机和路由器）。

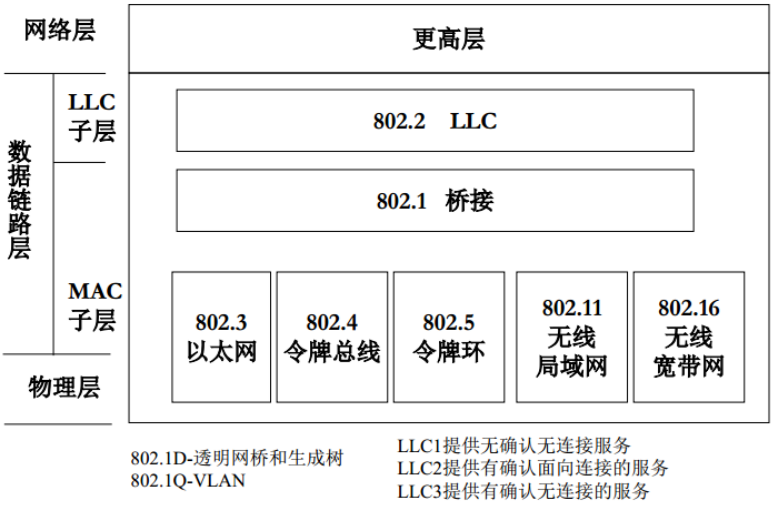
数据链路层的功能如下：

- 成帧(framing)
- 差错检测(error detect)：比特错，纠错
- 差错控制(error control)：丢包、重复、错序、流控制(flow control)
- 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分别制定了两个子层：

- 逻辑链路控制(Logic 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。



IEEE802系列标准

3.1 逻辑链路控制子层

3.1.1 差错检测

在数据报后加校验码（头部加序号），通过链路传输看是否有数据报/校验码错误。

1. 奇偶校验：若接收方收到奇数个1，则有出错

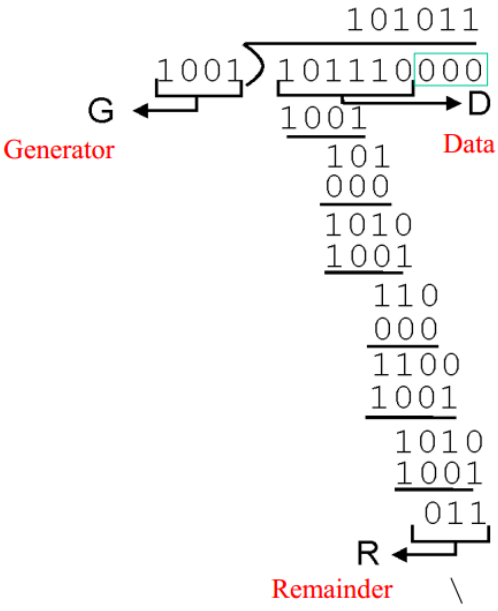
   - 一维偶校验：只能检错；最后补一位使得全部为偶数个1，如010补为010 | 1，而101补为101 | 0
   - 二维偶校验：检错+纠错一位；横纵同时偶校验

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，例如网络层和传输层

```
  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）
   接收方连带校验码一起除，余数为0则没错
   如下图，4位除数补3个0，最后的余数011即为校验码



   链路层常用CRC，因为检错率很高，且容易实现（触发器+异或门）

3.1.2 可靠数据传输

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协议如下：

- 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；最少需要2个序号。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

  效率/吞吐量十分低，信道空闲时间长
- 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)[3]
  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
  - 回退N协议(go back N)：同滑动窗口连续发送，某个ACK没收到则重传在此ACK之后的所有帧（超时重传），丢3则ACK4发2
    - 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
    - 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)要求重传该帧；如3丢失，4发送NAK=3，5发送ACK=2，重传3
    - 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；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
    - 超时时间应该设长，确保帧的2次来回；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；ARQ设得短可以使查出错误的时间变短，同时提升网络的吞吐量（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）。

注意不管哪一种重传机制，序号可以重复使用，因此有最小序号问题。

[3]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(overlow)，后面覆盖前面的

例 2. 序号*8*个，*SWS=RWS=4*，*345670123456*，*5*丢失

分析. 回退*N*：*346705670123456*
选择性重传：*346705123456*

例 3. 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*)*是百分之多少？

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*，则最大利用率位$8 \times 6.25\% = 50\%$

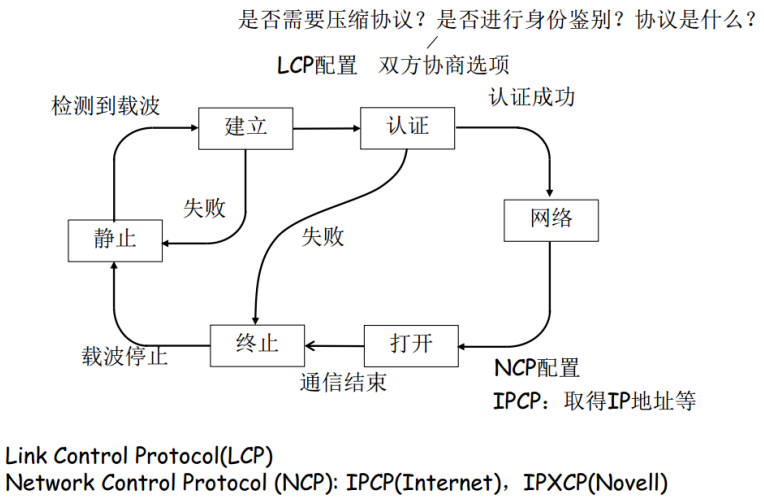
提高滑动窗口协议的效率：

- 选择性确认(selective acknowledgement)：接受方把已收到的帧的序号告诉发送方（收到不用重传告诉发送方哪些已收到，则只用重传某帧）
- 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），需要结合下面一起使用
- 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)：点对点网络**
    - 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



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

2. **以太网：多路访问网络**

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

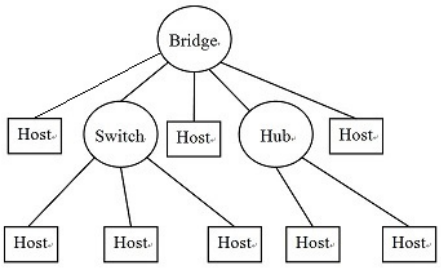
IEEE 802.3规定以太网物理层标准：

- 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

与802.3相比，快速以太网(100BaseTX)只是把传输速率提高到100Mbps

- 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0.96μs

集线器(hub)采用电子线路方法模拟总线方式的以太网，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 4. 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。

集线器、交换机、路由器区别见3.2.8节。

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，造成每帧之间有空隙）。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)τ的长度，最短帧的发送时间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，总之不会与之冲突。也就是说，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，10Mbps的以太网为51.2μs
- 第i次冲突从$0, 1, \ldots, 2j-1$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 5. 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

前导字符(8B)+目的地址(6B)+源地址(6B)+类型/长度(2B)+有效载荷/填充位+帧校验序列(4B)

- 源地址：一般为发送者的单播地址
- 目的地址(6B)：一般为接收者的单播地址

MAC地址

- 单播/网卡/烧录地址：全球唯一，每个网卡/接口一个
- 多播地址：字节0第0位为1，地址非全1
- 广播地址：48位全为1

3.2.4 透明网桥

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)：由端口P1，扩散到其他所有端口P2，P3（多播、广播一定扩散）扩散不回传收到的部分不会往回转
- 从某一条路发来则不能传回去，过滤/**丢弃**(filter)

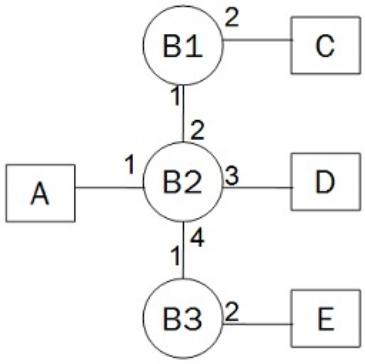
透明网桥有自学习机制：利用**源地址**学习，如信息从A主机-P1端口来，则记为A-P1，同时设好生存期(Time to live, TTL)[4]。如果收到的帧有错则直接丢弃，根本不会学习。如果源地址已经在表中，则更新记录，并重置超时计时器。

扩展/桥接局域网：每一个局域网(LAN)都是一个网段

例 6. 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

[4]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）



B1的MAC地址表

MAC地址	端口
D	1
C	2

B2的MAC地址表

MAC地址	端口
D	3
A	1
C	2

B3的MAC地址表

MAC地址	端口
D	1

分析.

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）。

3.2.5 生成树协议

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）

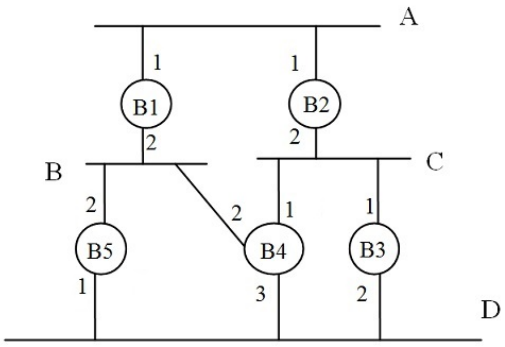
- IEEE 802.1D 生成树协议+透明网桥
- IEEE 802.1w RSTP(Rapid Spanning Tree Protocol)

工作流程如下

- 先确定根网桥，即**BID(Bridge ID)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；相同距离时，BID小的优胜；端口号小的优胜
- 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

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 7. *下图显示了由五个透明网桥(B1 B5)形成的扩展LAN。*



分析. *1. B1*是根网桥

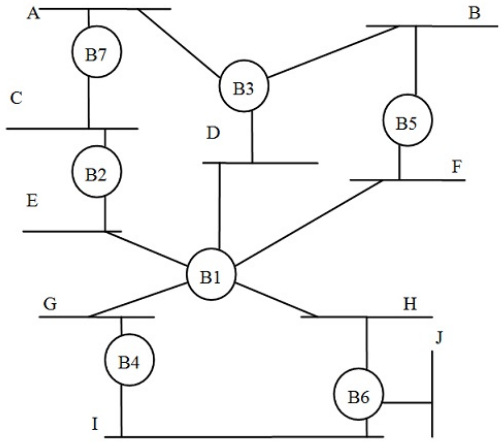
2. 网段*A-D*的指定网桥*(designated bridges)*分别是

A	*B*	*C*	*D*
B1	*B1*	*B2*	*B4*

3. 网桥*B1-B5*的根端口分别是

B1	*B2*	*B3*	*B4*	*B5*
无	*1*	*1*	*2*	*2*

例 8. 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*

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*

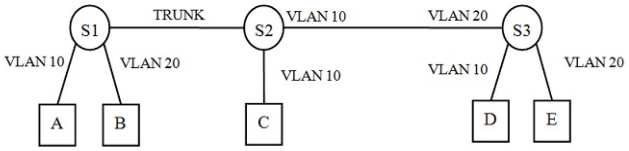
3.2.6 虚拟局域网

虚拟局域网(Virtual LAN, VLAN)将原来的局域网分割成多个相互隔离的局域网，只在具有相同颜色的端口间转发。

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VLAN1。

例 9. 下图中哪些发送的帧将被目的主机收到



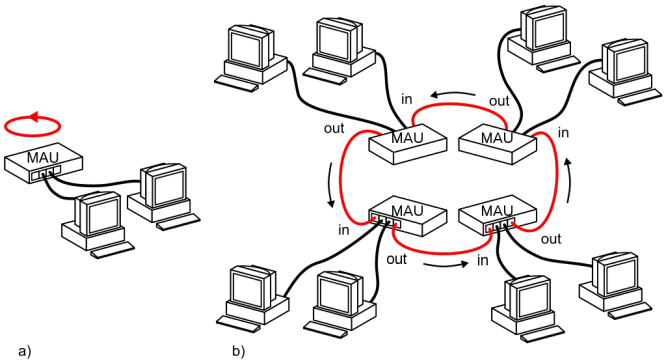
分析. *只有A到E或E到A可以成功发送信息，注意S2和S3的端口设错了（故意的）。如E到A，VLAN20经过S3转发到VLAN20，发到S2。S2误认为是从VLAN10发来的消息，故扩散到干道端口TRUNK加VLAN10，发到S1。S1接收到后转发至VLAN10。*
而D到B没有办法，因为从S3就转发不出去，没有干道端口。

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)

3.2.7 令牌环网

令牌环网(token ring)：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**，其标准为IEEE 802.5，take turns protocol，现在有千兆令牌环网。多站点接入部件称为(multistation access unit, MSAU)



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。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）。

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。

3.2.8 物理设备

交换机是一个把多个网段连接起来的设备，也称为**多端口网桥**(switch=bridge)。注意输入输出端口一样。

交换结构(fabrics)

- 共享总线式交换机：同样有冲突问题
- 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
  目的MAC地址（6B），转也要依照CSMA/CD来转发
- 直通(cut through)：收到一个转一个，出现碎片
- 无碎片(fragment free)：都没冲突才开始转发，**最小帧保证**
  交换机不用收到整个帧而是收到64B（冲突窗口）后自动转发
- 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[5]：

- 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

路由器与其他两者的区别：

[5] http://www.qianjia.com/html/2017-08/09_274208.html

1. 集线器工作在第一层，没有智能处理能力，对它来说，数据只是电流，当一个端口的电流传到集线器中时，它只是简单地将电流传送到其他端口，至于其他端口连接的计算机接收不接收这些数据，它就不管了。交换机工作在第二层，它要比集线器智能一些，对它来说，网络上的数据就是MAC地址的集合，它能分辨出帧中的源MAC地址和目的MAC地址，因此可以在任意两个端口间建立联系，但是交换机并不懂得IP地址，它只知道MAC地址。路由器工作在第三层，它比交换机还要更智能一些，它能理解数据中的IP地址，如果它接收到一个数据包，就检查其中的IP地址，如果目标地址是本地网络的就不理会，如果是其他网络的，就将数据包转发出本地网络。
2. 路由器能连接不同类型的网络，我们常见的集线器和交换机一般都是用于连接以太网的，但是如果将两种网络类型连接起来，比如以太网与ATM网，集线器和交换机就派不上用场了。路由器能够连接不同类型的局域网和广域网，如以太网、ATM网、FDDI网、令牌环网等。不同类型的网络，其传送的数据单元—帧的格式和大小是不同的，数据从一种类型的网络传输至另一种类型的网络，必须进行帧格式转换。路由器就有这种能力，而交换机和集线器就没有。而互联网就是由各种路由器连接起来的，因为互联网上存在各种不同类型的网络，集线器和交换机根本不能胜任这个任务，所以必须由路由器来担当这个角色。
3. 路由器具有路径选择能力，在互联网中，从一个节点到另一个节点，可能有许多路径，路由器可以选择通畅快捷的近路，会大大提高通信速度，减轻网络系统通信负荷，节约网络系统资源，这是集线器和二层交换机所根本不具备的性能。

小范围的局域网，如我们的校园网大多采用交换机，路由少。

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

4.1 IP数据报

4.1.1 数据传输技术

- 电路交换(circuit switching)：实际接通一条物理线路，时分多路复用，电话；频分多路复用，电视；一直占用，不管有无数据交互
- 包交换/分组交换(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：需建立连接才可以传输数据（仿照电话系统，因特网之前），好处在于保留带宽
    - 交换式（要交换才建立连接）：建立虚电路(VC)表，虚电路标识符(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**

一般网络的服务模型：Aynchronous 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

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)：无连接无确认的数据报服务
- IP协议可以运行在**任何**网络上，不仅仅是因特网

4.1.2 IP数据报格式

- 4个字节一个字，头部最多$(2^4-1)*4=60$B，除选项20B，IPv4选项最多40B，太少了
- 生存期(TTL)限制在因特网上的停留时间，实际限制为经过的路由器数目，即跳数(hop count)，超过则自动清除，防止兜圈，每次经过路由器减1
  TTL初值默认设置为网络直径的两倍，Windows默认64
  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
- 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
- IP数据报服务类型(type of quality, ToQ)，但路由器都没有实现

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，即数据帧的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
- 只要发出去一定会封装成帧（注意要加头部），帧最长就是MTU，因而要分成多段再分
- 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(Identification)字段加1。
- 分段时用标识的值保持不变，并且用偏移量字段(offset)指出该片段的数据部分相对原来数据报的偏移量(以8字节为单位)，给出原来片段的次序
- MF(More Fragment), DF(Don't Fragment)
- 小于MTU-20B，边界，一定要能被8整除，尽可能大（8字节，一定要除掉）
- IPv6中间不能分段
- 1400B=512B+512B+376B
- Path MTU discovery：找到路径上最小的MTU，发现路径上最小MTU
- 选项最后一定对齐到边界
- 生存期和头部校验（检验和）会变，其他不变

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，是可路由的
IP地址可划分为两个部分：

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

有类网：ABC单播，D多播，E保留，地址范围如下（点分十进制）

- $0 \sim 127$
- $128 \sim 191$
- $192 \sim 223$
- $224 \sim 239$
- $240 \sim 255$

解决IPv4地址不够用的问题

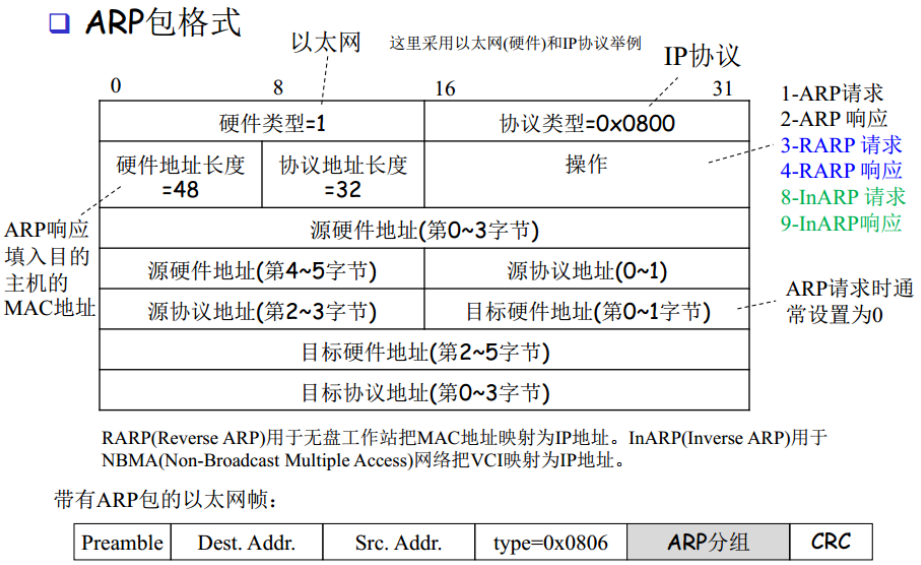
- 将一个有类网可以划分为多个相同大小的子网(subnet)
  用子网掩码(subnet mask)划分边界：主机号全0，剩下的部分（网络号和子网号）全是1
  子网掩码与IP地址**相与**，若相等则在同个子网中
- 变长子网掩码(Variable-length subnet mask, VLSM)：允许把一个有类网划分为多个不同大小的子网，类似变长指令集
  解决主机数目不均匀的问题，如100、50、25、10，则不能等距划分子网
  用长度来表示子网掩码，如/26代表255.255.255.192
- 无类域间路由选择协议(classless inter-domain routing, CIDR)：将多个有类网合并为一个更大的网络，称为超网(supernet)
  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
  NAT将内部源地址转换为外部地址
  NAPT将端口号也加入NAT的映射中

地址解析协议(address resolution protocol, ARP)可以将IP地址映射为MAC地址[6]

- 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$\langle \text{IP address}, \text{MAC address}, \text{TTL} \rangle$，TTL一般为2到20分钟
- 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

[6]也有将也有MAC映射为IP地址的协议

- 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**不知



DHCP协议(Dynamic Host Configuration Protocol)用于主机在加入网络时**动态租用**IP地址，用UDP，四个步骤如下

- DHCP发现(discover)
- DHCP提供(offer)
- DHCP请求(request)
- DHCP确认(ACK)

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（TTL减到0、不可达），如回响请求和答复消息(ping)、不可达消息、时间超时消息（原IP头部+原IP数据部份的头64B）、重定位消息

4.3 路由协议

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

无类网的路由选择：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）
- 直连网将数据报直接**封装成帧**发送，不需要目的地址(PPP)
- 以太网同样要封装成帧，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**，加入帧内
- 如没有下一跳，则直接取**IP数据报中的目的IP地址**（已经到达了）
- 发送时要遵守以太网协议(CSMA/CD)
- 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（上一跳路由器MAC地址）全要发生变化

路由表

- 127.0.0.1是内部的，其他是外部的需要经过防火墙
- 接口不一样，因此要在路由器里写两项，一项内部一项外部
- 往外发，默认路由会匹配
- 选择跃点数(metric)小的一项，如果跃点数也相同，则两个接口都会发送

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。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。

建路由表是记**最短路径**的下一跳。

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，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。

加了网关相当于加了默认路由。

网络层在入口位置有防火墙，未查路由表就丢弃了。

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

- 链接状态(link state, LS)：Dijstra
- 距离向量(distance vector, DV)：BellmanFord

4.3.1 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**[7]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，则更新该路由的距离为新距离，把下一跳改为邻居。如果原来的更大，则**也要进行改动**，因为原来的路可能断了。即路由的下一跳送来的新路由，则必须修改距离。
3. 如果路由存在，就要重置失效定时器

RIP路由表的每一项都有TTL(Time-To-Live)，用失效定时器(invalid timer)计时，超时则让该路由失效

RIP协议存在的问题

- 慢收敛：最短时间接近0（这样看更新时刻），最长时间30$(m-1)$，平均时间15$(m-1)$
- 计数到无穷：N1-R1通路断了，R1收到R2的路由表，更改自己的

RIP协议的技术

- 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

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
- 开销缺乏灵活性
- 存在慢收敛问题和计数到无穷问题
- 每30秒发送完整路由表会消耗大量的带宽
- 实际运行的RIP协议具有如下特性：

[7]太频繁会占用带宽

– 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

4.3.2 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：

- 利用最短路径算法，如Dijkstra，求出一个节点(源节点)到所有其它节点的最短路径
- 利用这些最短路经上的下一个节点作为下一跳得到源节点的转发表(路由表)

OSPF协议的简单描述：

- 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
- 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
- 如果点到点网络没有配置IP地址，则该结点可去除
- **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/网段**都有自己的链路状态通告(Link State Advertisement, LSA)（2种LSA）

详细过程如下

- 发现邻居：每10秒向邻居发送Hello分组，如果40s(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。若通过收到多个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。通过序号，也可以防止扩散形成回路，第二次收到来自相同的发通告路由器、相同LSA类型和相同序号的LSA将丢弃它
- 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

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
- 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

- 当多路访问网络重启时，选择DR的过程就开始了。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到达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10秒(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**的路经存在(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**。

LS算法和DV算法比较

	LS	DV
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[8]	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	DV节点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

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**，可以提供**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和TCP。 UDP协议提供不可靠的尽力服务， TCP协议提供可靠的字节流服务。协议号：TCP为6，ICMP=1；UDP为17，IGMP=2

TCP/UDP通过数据段(segment)中的目的端口号(2B)确定将收到的数据段交给上层哪个进程。

- 知名端口：0-1023，为提供知名网络服务的系统进程所用。例如: 80-HTTP， 21-ftp Control， 20-ftp Data， 23-telnet，25-SMTP， 110-POP3，53-DNS
- 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

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，在IP协议上增加了**端口号**，把进程关联起来了。

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。

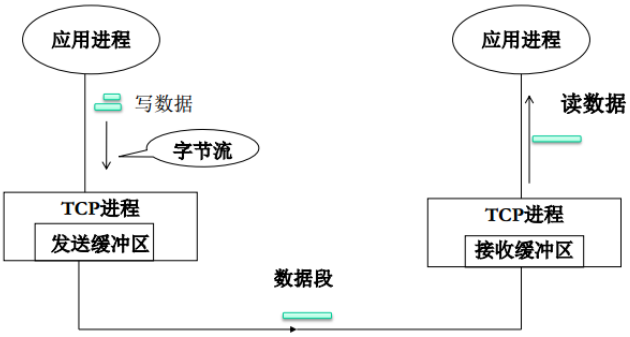
数据报的内容

- 总长度：整个UDP报文长度
- 源端口号和目的端口号：用于关联发送进程和接收进程

- 校验和：由**伪IP头（就是用来算校验和用）**、 UDP头(校验和为0)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头部的总长度。

5.2 TCP协议

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。



- 一个TCP连接提供**可靠的字节流服务**。字节流服务表示**没有消息边界**。例如，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**。
- 每个数据段的数据部分的最大长度(字节)不能超过MSS(Maximum Segment Size)。
- 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，通过**端口号**区分不同进程。
- 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

TCP数据报格式：头部长度以四个字节为单位。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。字节流中的每个字节均被编号。初始序号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初始序号加1。

- SYN：表示建立连接
- 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：表示响应
- PSH：将接收缓冲区的进程全部推给接收方进程
- RST：发现连接可能出了问题，连接重置
- URG：紧急指针用于指出紧急/带外数据(out-of-band)的边界，紧急数据最后一个字节

TCP协议工作过程

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：x,y为初始序号，随机数

- SYN,Seq#=x
- (Client)SYN+ACK,Seq#=y; Ack#=x+1
- ACK,Ack#=y+1

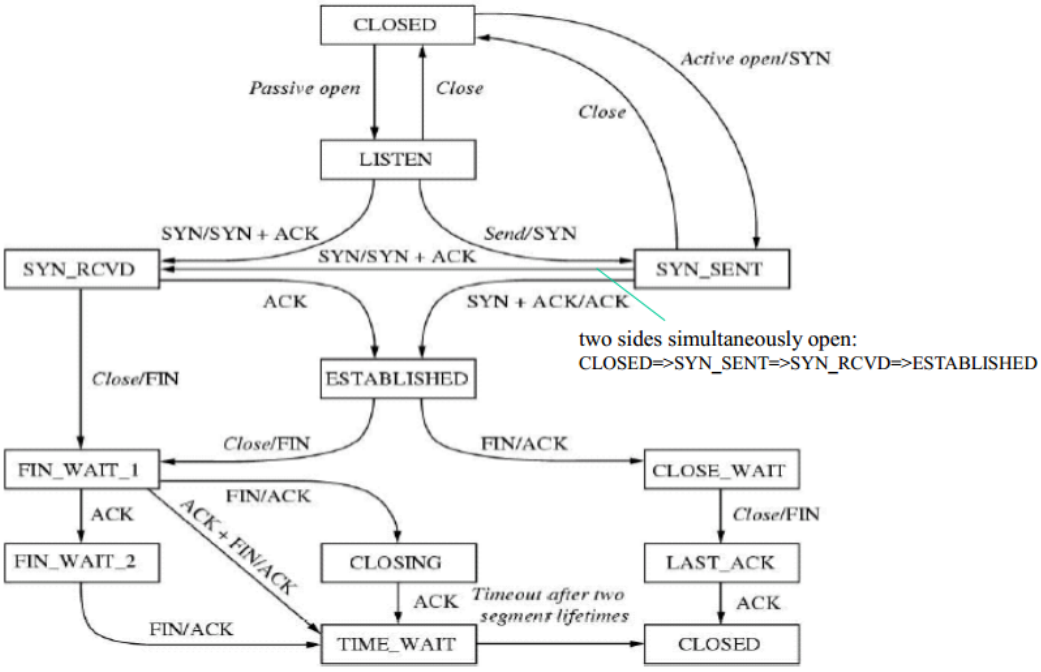
这里确认号含义与数据链路层不同，这里是期待接收的序号

四次握手关闭连接

- FIN,Seq#=x
- (Client)ACK,Ack#=x+1
- (Client)FIN,Seq#=y
- ACK,Ack#=y+1

可以合并中间两次握手(ACK和FIN)或两方同时发出ACK

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）。 TCP标准中MSL采用60秒， Unix采用30秒。**避免通道中还有未到达的数据**。



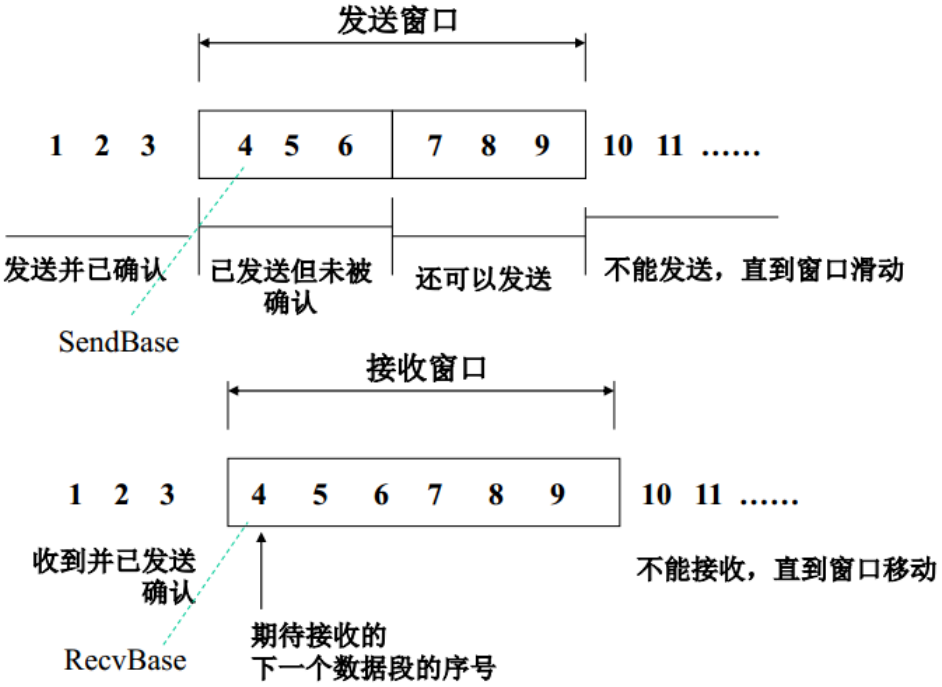
TCP协议使用**选择性确认**协议，不使用NAK。只有**一个超时定时器**。

- 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[9]，按**字节**编号。
- 确认号为**期待接收**[10]的下一个**字节**(下一个数据段)的序号。

[9]数据链路层每一个帧占一个序号

[10]区别于数据链路层的滑动窗口，确认号是当前接收并确认的序号

- 由于TCP是对IP数据报的封装，要获取数据的长度，可以通过IP数据报中的总长度获取。
- TCP协议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和接收窗口大小(advWin)
- 发送方做发送窗口，序号为x+1，大小等同advWin大小（发送窗口大小是会变的）
- 接收窗口大小**会变**，不同于数据链路层；同理发送窗口大小**也会改变**，用接收窗口大小设置发送窗口大小
- 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
- 会自动移动超时定时器
- 一旦发现有问题，立即减慢
- 要考虑错序到达的数据段

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。

- 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，这种方法称为**快速重传**(fast retransmit)。缺点是丢包时间很长，优点是可以减缓网络压力。
- 采用**延迟确认**(delayed ACK)时，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**允许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

Go back N超时重传1个RTT，没有中间结点，故可以用固定的超时时间。但由于TCP涉及多个结点，故需要实时变化，进行估计。

总结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
确认号	期待接收的序号	收到哪一个就用哪一个，代表当
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

原始公式

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

- $0 < \alpha < 1$, 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。
- 这个公式也称为指数加权移动平均方法(Exponentially Weighted Moving Average, EWMA)
- RTO(retransmission timeout) $= 2 \times$ EstimatedRTT

TCP超时计算最常用的算法—Jacobson算法

- 修改上一页公式的参数：$\alpha = 1/8$。
- Jacobson/Karels提议RTO计算还要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。

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

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

一般设$\beta = 1/4$

采样频率：如果发送窗口为12MSS，则每12个段取样一次。

Karn算法：在收到重传段确认时不要计算EstimatedRTT。在每次重传时直接把RTO加倍直到数据段首次得到确认，并把这个RTO作为后续段的RTO。这个修正称为Karn算法(Karn and Partridge, 1987)。每次收到非重传段的ACK之后，就计算EstimatedRTT和正常的RTO，并用这个RTO作为后续段的RTO。

在12次重传后TCP协议发送rst数据段并关闭连接。

- 流控制：单一发送方和接收方，控制发送的速度
- 拥塞控制：大家都发数据，使整个网络的数据太多，让每一个都不要传那么快

拥塞简单来说“太多的主机发送太快太多的数据给网络处理”。这不同于流控制！表现为：

- 丢包（路由器上缓冲区溢出）

- 长延迟（在路由器缓冲区中排队）

不主张发送ICMP包，加重拥塞；靠TCP自己发现拥塞。

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

- 超时或收到3个重复ack就认为丢包了，看作拥塞发生了
- TCP协议通过减少发送速率来控制拥塞。发送速率与发送窗口大小有关：

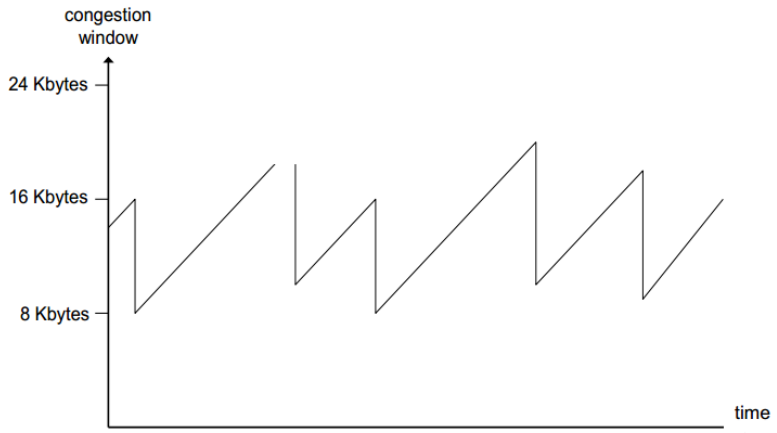
$$\text{发送速率 rate} = \text{SWS(Sending Window Size)/RTT}$$

- 引入拥塞窗口变量CongWin来限制SWS。

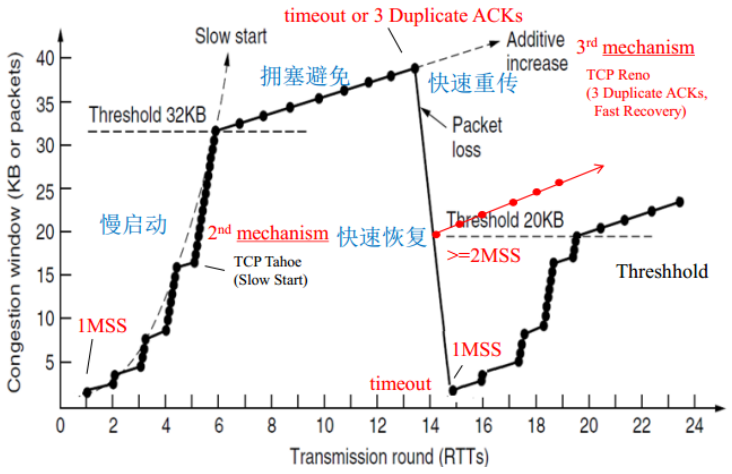
$$SWS = \min(\text{CongWin}, \text{AdvWin})$$

CongWin发送窗口限制，Adcwin接收方要求的流量控制

- TCP协议改变CongWin的三种机制：
  - 加性增乘性减(AIMD)



  - 慢启动(slow start)



1. 初始时， CongWin设为1 MSS, 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通知窗口更慢。
3. 当拥塞发生时，把当前CongWin (或SWS)的一般保存为阈值(threshold)，然后CongWin又从 1MSS开始慢启动。
4. 当 CongWin增长到等于或大于阈值(threshold)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。(Congestion Avoidance)实际上，每收到一个ACK， CongWin增加SegSize*SegSize/CongWin。如果发生拥塞，转 (3)。
5. 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
   * 假设算法开始时通知窗口大小AdvWin=65535

– 在超时时间之后的保守方法

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，未确认的数据量(bandwidth*RTT)

- 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。TS也用来测量RTT。
- 发送窗口太小，满足不了要求：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。

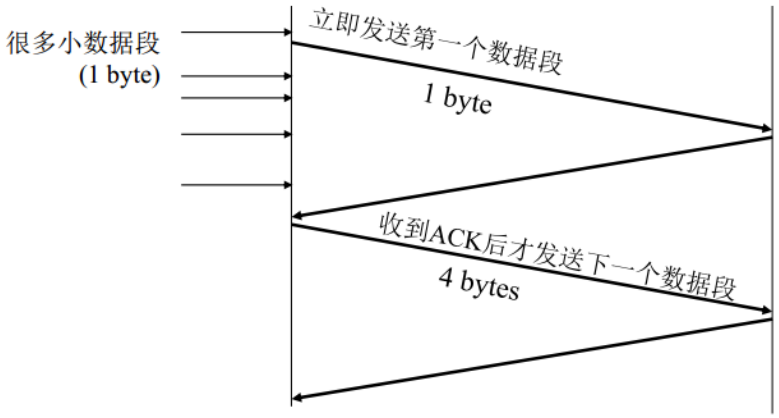
问题二—死锁现象：win为0，接收方取空，再发不为空的ACK丢失了。缓冲区不变化，不会发确认回来

- 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, 发送方如果有数据要发送，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：设个定时器，到时间就把缓冲区内容发送出去

  Nagle算法—启发式算法



  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

- 接收方问题，接收进程频繁取走小批量数据

  Clark算法：当接收缓冲区的空闲块大小变得很小时，要等到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。

TCP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。
- 持续定时器(persist timer)用于保持窗口大小信息流动即使连接的另一端关闭了接收窗口。
- 保活定时器(keepalive 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）
- 处于TIME_WAIT状态的连接一方需要等待2MSL秒，时间到才能关闭连接。