计算机网络

Kajih Du

计算机网络

1 绪论

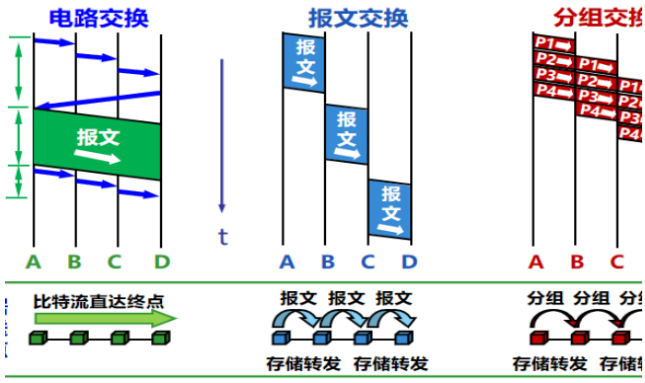
1.1 基本组成

网络的组成：

- 边缘设备：端系统（host），功能为接收和发送数据
- 网络核心：路由器和交换机，功能为路由和转发

传输单元：位（bit），在物理介质上传播的数据最小单元。数据传输使用bit表示，**K/M/G之间为10^3进制**。

交换方式：



- 电路交换：需要建立连接并预留资源
  - 缺点：难以实现灵活复用
- 报文交换：不常用
- 分组交换：适合有大量突发数据传输需求的互联网
  - 优点：具有较好的交换灵活性，分组长度小于报文长度，分组交换的时延比报文交换要短

1.2 协议和服务

协议：同一层对等实体的进行通讯的规定。包含**语法、语义、时序**三个要素。（“水平”的）
服务：上层实体通过接口调用下层实体的服务。（“垂直”的）

- **面向连接传输服务**：按电话系统模型

- **无连接传输服务**：按邮政系统模型

实体可以通过协议来实现其定义的服务

面向链接服务的六个核心服务原语：连接请求、接受响应、请求数据、应答、请求断开、断开连接。

1.3 OSI参考模型

OSI模型先有模型后有设计协议，支持无连接和面向连接的通信

- 物理层：定义在信道上如何传输0和1
- 数据链路层：实现相邻网络实体间的数据传输
    - **成帧：**从物理层比特流提取出完整的帧
    - **错误检测和纠正**
    - **流量控制**
    - **物理地址（MAC Address）**
    - 共享信道的访问控制
- 网络层：将数据包跨越网络从源设备发送到目的设备（host to host）
    - **进行路由和转发**
    - **IP地址**
- 传输层：将数据从源端口发送到目的端口（进程到进程）
- 会话层：在应用程序之间建立和维持会话，并能使会话获得同步
- 表示层：关注所传递信息的语法和语义，管理数据的表示方法，传输的数据结构
- 应用层：通过应用层协议，提供应用程序便捷的网络服务调用

1.4 TCP/IP参考模型

TCP/IP参考模型是对已有协议的描述，只支持无连接通信

- 应用层
- 传输层
- 互联网层
- 网络接口层

采用IP分组交换：可在各种底层物理网络上运行；可支持各类上层应用；每个IP分组携带各自的目的地址，网络核心功能简单（通过路由表转发分组），适应爆炸性增长

1.5 网络设备

交换机：物理层、链路层 （两层，无法实现路由）
路由器：物理层、链路层、网络层 （三层）

只有物理层的网络设备：集线器、放大器

1.6 度量单位

比特率：数字信道上传送数据的速率，单位b/s或bps

带宽：网络内某通道传输数据能力，单位时间内网络中某信道能通过的“最高数据率”，单位bit/s

包转发率（PPS）：Packet Per Second(包/秒)，表示交换机或路由器等网络设备以包为单位的转发速率

时延：数据（一个报文或分组）从网络（或链路）的一端传送到另一端所需的时间。

- **传输时延**：数据从结点进入到传输媒体介质所需要的时间

  $$d_{\text{trans}} = \frac{Length}{R}$$

  $Length$为文件长度，R为传输速率
- **传播时延**：电磁波在信道中需要传播一定距离而花费的时间

  $$d_{\text{prop}} = \frac{Distance}{c}$$

  $Distance$为源点和接受点距离，c为信号传播速率
- **处理时延**：主机或路由器在收到分组时，为处理分组所花费的时间，与路由器有关
- **排队时延**：分组在路由器输入输出队列中排队等待处理所经历的时延，和路由算法有关

往返时延（RTT）：从发送方发送数据开始，到发送方收到来自接收方的确认，经历的总时间

时延带宽积：传播时延和带宽的乘积，即按比特计数的链路长度

吞吐量：单位时间内通过某个网络(或信道、接口)的数据量，单位是 b/s

有效吞吐量：单位时间内，目的地正确接收到的**有用信息的数目**（以 bit 为单位）

2 物理层

2.1 功能

位置：位于网络体系最底层（不是连接计算机或传输信号的具体的物理设备）

功能：在连接各计算机的传输媒体上**传输数据比特流**

作用：尽可能屏蔽掉不同传输媒体和通信手段的差异

2.2 特性

接口特性：物理层协议是DTE和DCE间的约定，规定了两者之间的接口特性

- 机械特性：定义接线器的形状和尺寸、引线数目和排列、固定和锁定装置等
- 电气特性：规定了DTE/DCE之间多条信号线的电气连接及有关电路特性
- 功能特性：描述接口执行的功能，定义接线器的每一引脚(针，Pin)的作用
- 过程特性：指明对于不同功能的各种可能事件的出现顺序

2.3 数据通信基本概念

通信：在源点和终点之间传递信息或消息
消息：通信过程中表达客观物质运动和主观思维活动的文字、符号、数据等
信息：消息中对通信者有意义的部分内容
数据：对某一事实的不经解释并赋予一定含义的数字、字母、文字等符号及其组合的原始表达
信号：消息的载体

信息量：一条消息所含信息的多少，与描述事件出现的概率有关，单位为比特

$$I = -\log_a(p)$$

平均信息量：每个符号包含信息量的统计平均，单位为比特/符号

$$H = \sum_i -p_i \log p_i$$

模拟数据：一段时间内具有连续的值
离散数据：一段时间内具有分散的值

2.4 传输方式

2.4.1 基带传输和频带传输

基带传输：不搬移信号频谱的传输机制
频带传输：通过调制调节器搬移信号频谱的传输机制（为了适应信道的频率特性）

2.4.2 串行传输和并行传输

串行传输：数据在一个信道上按位依次传输的方式
并行传输：数据在多个信道上同时传输的方式

2.4.3 传输模式

单工：指两个站之间只能沿一个指定的方向传送数据信号
半双工：指两个站之间可以在两个方向上传送数据信号，但不能同时进行
全双工：指两个站之间可以在两个方向上同时传送数据信号

2.5 性能度量

传输速率：单位时间内传送信息量
调制速率（波特率）：单位时间内调制信号波形变换次数，单位为波特（baud）

$$R_B = 1/T$$

数据信号速率（比特率）：单位时间内通过信道的信息量，单位bit/s

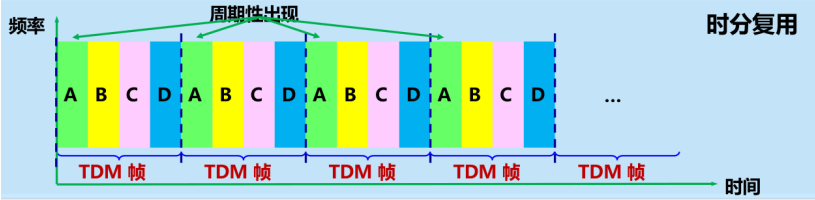
$$R_b = \sum_i \frac{1}{T_i} \log_2 M_i = R_B \log_2 M_i$$

相同条件下，进制越大，比特速率越大，但差错率也大。

2.6 复用技术

复用：允许用户使用一个共享信道进行通信，避免相互干扰，降低成本，提高利用率。

2.6.1 时分复用（Time Division Multiplexing）



- 每一个时分复用的用户在每一个时分复用（TDM）帧占用固定序号的时隙
- 每一个用户占用时隙周期性出现（周期为TDM帧长度）
- TDM信号为等时信号
- 所有用户在不同时间占据同样频带宽度

缺点：对信道利用率低（某用户暂时无数据发送时，在时分复用帧中分配给该用户的时隙只能处于空闲状态），不同用户之前传输时需要进行同步

2.6.2 统计时分复用

统计时分复用：动态地按需分配共用信道的时隙，只将需要传送数据的终端接入共用信道，以提高信道利用率

在传输数据前进行统计，按需**动态分配时隙**

2.6.3 频分复用

频分复用：将多路基带信号调制到不同频率载波上，再进行叠加形成一个复合信号的多路复用技术

频分复用的所有用户在同样的时间**占用不同的频率带宽资源**

2.6.4 波分复用

波分复用：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

2.6.5 码分复用

码分复用：利用码序列相关性实现的多址通信 , 基本思想是靠不同的地址码来区分的地址

每个站的码片序列各不相同，并且需要正交

缺点：容易受到噪声影响；传输过程中传输的数据量增加，需要的频率带宽增加

2.6.6 正交频分复用

将信道分为若干正交子通道，将高速数据信号转换成并行的低速子数据流，调制到在每个子信道上进行传输

相比于频分复用，其允许不同使用者的频带可以重叠，提高了频带的利用率

2.6.7 空分复用

空分复用：让同一个频段在不同的空间内得到重复利用

3 数据链路层

3.1 位置

向下：利用物理层提供的位流服务
向上：向网络层提供明确接口

实现功能：成帧、差错控制、流量控制

3.2 服务

根据确认（ACK）有无和连接（connection）有无分类

- 无确认无连接服务

  接收方不会确认是否收到，不会记录网络传输状态

  优点：传输速度快；缺点：不能确认数据是否有错误

  适用：误码率低的可靠信道；实时通信
- 有确认无连接服务

  接收方会确认是否收到，不会记录网络传输状态

  优点：网络通信可靠

  适用：不可靠的信道（无线信道）
- 有确认有连接服务

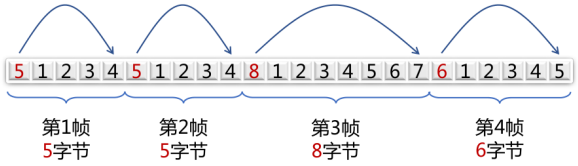
  接收方会确认是否收到，并记录网络传输状态

  适用：长延迟的不可靠的信道

3.3 成帧

目的：为了差错控制和流量控制

3.3.1 字节计数法



每个帧的头标表示帧包含的字节数。

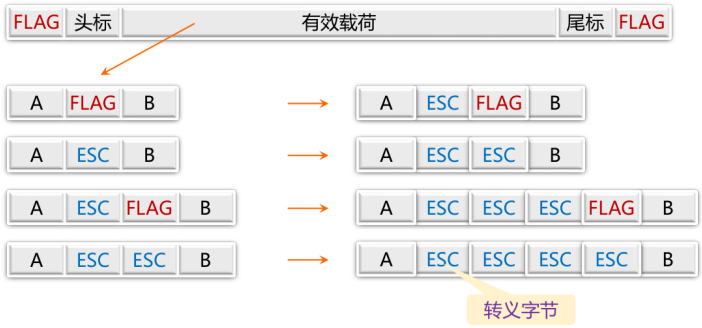
缺点：只能用于无差错传输的情形，方法抗噪性低。

3.3.2 带字节填充定界符法

用定界符区分前后两个帧

缺点：有效载荷部分不能有与定界符相同的字节

解决方法：定义转义字符，接收到转义字符，后一字节无条件成为有效载荷

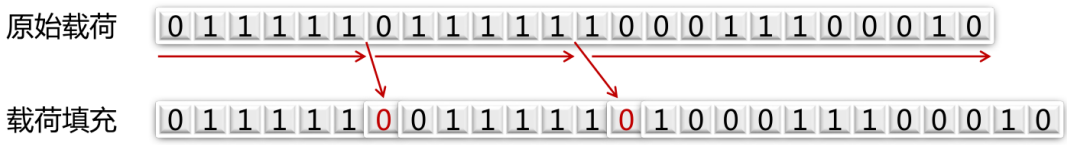


不足：转义字符和flag以字节为单位
改进方式：使用带比特填充定界符法

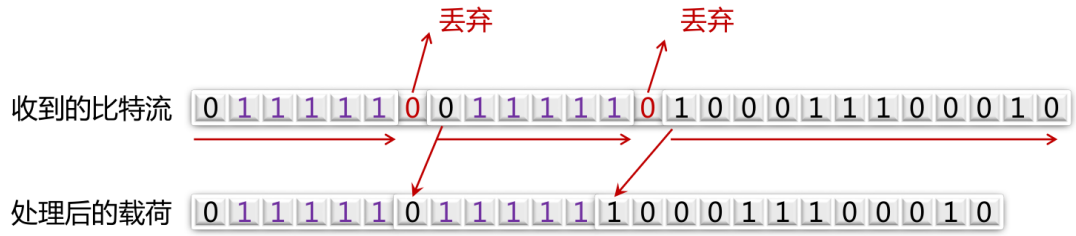
3.3.3 带比特填充定界符法

采用一特定的比特组合01111110（0x7E）来标志帧的边界，也就是一个标志字节。

发送方检查有效载荷：若在有效载荷中出现连续5个1比特，则直接插入1个0比特



接收方处理：若出现连续5个1比特，若下一比特为0，则为有效载荷，直接丢弃0比特；若下一比特为1，则连同后一比特的0，构成定界符，一帧结束



带字节填充定界符法和带比特填充定界符法属于透明传输

3.4 差错控制

主要策略：检错码、纠错码

检错码：在被发送的数据块中，包含一些冗余信息，但这些信息**只能使接收方推断是否发生错误**，但不能推断哪位发生错误，接收方可以请求发送方重传数据

检错码用在高可靠、误码率较低的信道上，差错可以用重传的方法解决

纠错码：发送方在每个数据块中加入足够的冗余信息，使得接收方能够判断接收到的数据是否有错，并能纠正错误

纠错码用在错误发生比较频繁的信道上

3.4.1 基本术语

码字：一个包含m个数据位和r个校验位的n位单元，(n, m)码，$n = m + r$
码率：码字中不含冗余部分所占的比例，可以用m/n表示
Hamming距离：两个码字之间不同对应比特的数目

检查出d个错，可用Hamming距离$d + 1$的编码
纠正d个错，可用Hamming距离$2d + 1$的编码（找Hamming距离最小的有效码字）

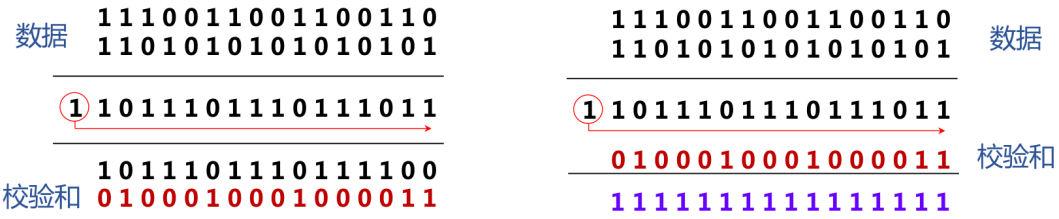
3.4.2 检错码

3.4.2.1 奇偶校验

奇偶校验：增加1位校验位，可以**检查奇数位错误**。

- 偶校验：保证1的个数为偶数个
- 奇校验：保证1的个数为奇数个

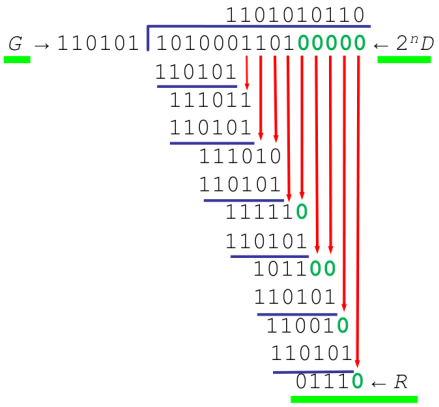
3.4.2.2 校验和



发送方：进行 16 位二进制补码求和运算，计算结果取反，随数据一同发送
接收方：进行 16 位二进制补码求和运算（包含校验和），结果非全1，则检测到错误

3.4.2.3 循环冗余校验（CRC）

- 设原始数据D为k位二进制位模式
- 如果要产生n位CRC校验码，事先选定一个$n+1$位二进制位模式G (称为生成多项式，收发双方提前商定)，G的最高位为1
- 将原始数据D乘以2^n（相当于在D后面添加 n 个0），产生$k+n$位二进制位模式，用G对该位模式做模2除，得到余数R（n位，不足n位前面用0补齐）即为CRC校验码



3.4.3 纠错码

单比特纠错：
m个信息位，r个校验位，对2^m个有效信息中任何一个，有n个与其距离为1的无效码字，于是$(n+1)2^m \leq 2^n$，进而

$$m + r + 1 \leq 2^r$$

3.4.3.1 Hamming码

校验位：2的幂次方位

...

3.5 基本协议

- 分层进程独立假设：网络层、数据链路层、物理层为独立进程，进程间通过传递信息通信
- 提供可靠服务假设：提供可靠、面向连接的服务
- 只处理通信错误假设：仅处理通信错误

声明：

- `void to_physical_layer(frame *s);`
- `void from_network_layer(packet *p);`
- `void from_physical_layer(frame *r);`
- `void to_network_layer(packet *p);`
- 事件：帧到达`frame_arrival`、帧出错`cksum_err`、超时`timeout`
- 帧

```
typedef struct {
    frame_kind kind; // 数据帧或确认帧
    seq_nr seq; // 序列号
    seq_nr ack; // 确认号
    packet info;
} frame;
```

3.5.1 乌托邦式单工协议

假设：

- 完美信道：传输过程中无任何差错
- 始终就绪：发送方/接收方的网络层始终处于就绪状态
- 瞬间完成：发送方/接收方能生成/处理无限多数据

特点：不需要进行流量控制和纠错

实现：

- 发送方：循环中不断发送，从网络层获得数据封装成帧，交给物理层完成一次发送

```
frame s;
packet buffer;
while (true) {
    from_network_layer(&buffer); // 从网络层获得数据
    s.info = buffer; // 封装成帧
    to_physical_layer(&s); // 传送到物理层
}
```

- 接收方：循环中持续接受，等待帧到达`frame_arrival`，从物理层获得帧，解封装将帧的数据传到网络层，完成一次接收

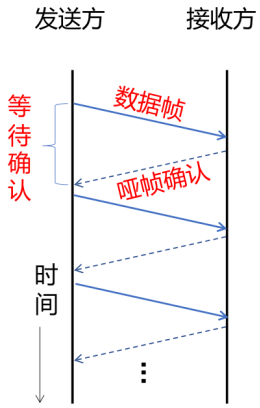
```
frame r;
event_type event;
while (true) {
    wait_for_event(&event);
    from_physical_layer(&r); // 从物理层获得帧
    to_network_layer(&r.info); // 解封装发送到网络层
}
```

3.5.2 无错信道的停等式协议

假设：

- 完美信道：传输过程中无任何差错
- 始终就绪：发送方/接收方的网络层始终处于就绪状态

特点：需要考虑流量控制



实现：

- 发送方：发送一帧后，等待确认到达，确认到达后发送下一帧

```
frame s;
packet buffer;
event_type event;
while (true) {
    from_network_layer(&buffer); // 从网络层获得数据
    s.info = buffer; // 封装成帧
    to_physical_layer(&s); // 传送到物理层
    wait_for_event(&event); // 等待确认到达
}
```

- 接收方：完成一帧接收后交给物理层一个哑帧（帧的内容无意义），作为确认帧

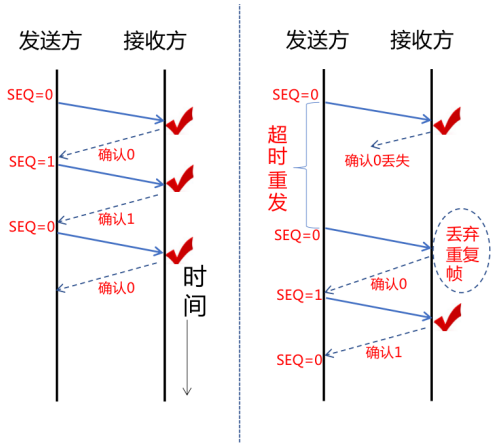
```
frame r;
frame s; // 哑帧
event_type event;
while (true) {
    wait_for_event(&event);
    from_physical_layer(&r); // 从物理层获得帧
    to_network_layer(&r.info); // 解封装发送到网络层
    to_physical_layer(&s);
}
```

3.5.3 有错信道的单工停等式协议

传输过程中出现的问题：差错、丢失、重复帧

解决方法：

- 否定确认帧（NACK）：解决差错
- 差错和丢失同时解决：设置超时（timeout）
- 重复帧出现的原因：1. 超时设置太短；2.确认帧丢失
  解决重复帧：在帧中添加序列号（`SEQ`），1位序号（0或1）即可



实现：

- 发送方：初始化帧序号0，发送帧；等待事件；如果正确确认，发送下一帧，否则重发

```
frame s;
packet buffer;
```

```
event_type event;
next_frame_to_send = 0;
while (true) {
    s.info = buffer; // 封装成帧
    to_physical_layer(&s); // 传送到物理层
    start_timer(s.seq);
    wait_for_event(&event); // 等待事件
    if (event == frame_arrival) {
        from_physical_layer(&s); // 从物理层获得帧
        if (s.ack == next_frame_to_send) { // 确认号确认
            from_network_layer(&buffer);// 从网络层获得数据
            inc(next_frame_to_send); // 修改序列号
        }
        // 如果确认号不一致，会直接回到循环体头部重传帧
    }
    // 如果不是帧到达（超时、帧出错），则回到循环体头部重传帧
}
```

- 接收方：初始化期待0号帧；等待帧到达；如果为正确帧，则交给网络层并发送帧确认，否则发送上一个成功接收帧的确认

```
frame r;
frame s; // 确认帧
event_type event;
frame_expected = 0;
while (true) {
    wait_for_event(&event);
    if (event == frame_arrival) { // 帧到达事件
        from_physical_layer(&r);
        if (r.seq == frame_expected) { // 判断是否为重复帧
            to_network_layer(&r.info);
            inc(frame_expected); // 修改期待的序列号
        }
        // 如果是重复帧，发送前一个帧的序列号，即1 - frame_expected
        // 如果不是重复帧，frame_expected被修改，原来帧的序列号为1 - frame_expected
        s.ack = 1 - frame_expected;
        to_physical_layer(&s);
```

```
    }
    // 帧出错等待下一个事件
}
```

效率评估：

- 帧的大小F，bits
- 信道带宽R
- 传播时延和处理时延I，总延迟$D = 2I$
- 信道利用率：

  用于发送的有效时间为F/R，消息发送使用的总时间为$F/R + D$，利用率为

  $$\frac{\frac{F}{R}}{\frac{F}{R} + D} = \frac{F}{F + RD}$$

  RD为时延带宽积

3.5.4 滑动窗口协议

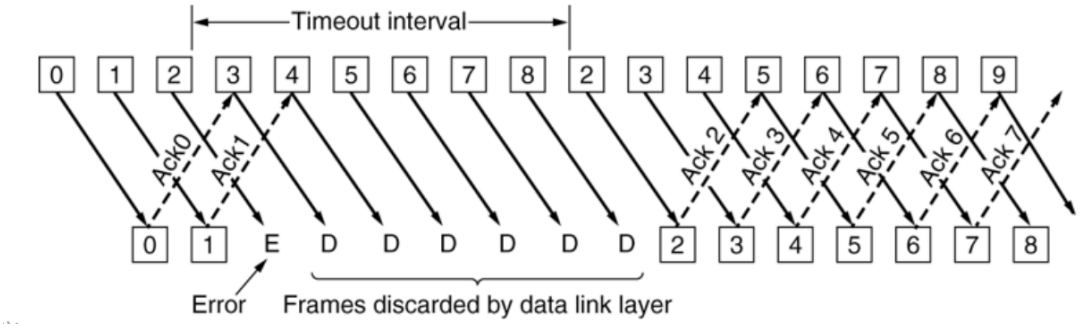
长肥网络：时延带宽积很大（大于10^5 bits）的网络，如卫星通讯网络。
对于长肥网络，信道利用率很低。

解决方法：使用更大的帧。但帧越大，在传输中出错的概率越高，将导致更多的重传。参考分组交换的方法，允许发送方在没收到确认前发送多个帧

实现要点：

- 序号：循环利用有限的帧序号
- 累计确认：不必对收到的分组逐个发送确认，而是**对按序到达的最后一个分组发送确认**

3.5.5 回退N协议



- 发送方需要缓存多个分组
- 采用出错全部重发的方式：当接收端收到一个出错帧或乱序帧时，丢弃所有的后继帧（无论正确接收与否），并且**不为这些帧发送确认**；发送端超时后，重传所有未被确认的帧（发送方的计时器在第一个未被确认的帧上，超时后从这个帧开始重传）
- 发送方响应的三件事：

  （1）上层的调用：检测有没有可以使用的序号，如果有就发送
  （2）收到ACK：对n号帧的确认采用累积确认的方式
  （3）超时事件：如果出现超时，就重传所有已发送未确认的分组
- 滑动窗口长度：帧序号n位，接受窗口只需要$W_{\mathrm{R}}=1$，发送窗口$W_{\mathrm{T}} \leq 2^{n}-1$

适用场景：该策略对应**接收窗口为1**的情况，即**只能按顺序接收帧**

优点：连续发送提高了信道利用率
缺点：按序接收，出错后即便有正确帧到达也丢弃重传

丢失确认帧（ACK）：不影响，因为是累计确认，不会导致发送方重传

3.5.6 选择性重传协议

发送方发出连续的若干帧后，收到对其中某一帧的否认帧，或某一帧的定时器超时， 则只重传该出错帧或计时器超时的数据帧

适用：该策略对应接收窗口大于1的情况，即暂存接收窗口中序号在出错帧之后的数据帧

窗口最大尺寸：2^{n-1}

优点：避免重传已正确传送的帧
缺点：在接收端需要占用一定容量的缓存

4 介质访问控制子层（MAC子层）

4.1 随机访问协议

4.1.1 ALOHA协议

原理：想发就发
特点：

- 冲突：两个或以上的帧
- 随时可能冲突
- 冲突的帧完全破坏
- 破坏了的帧要重传

帧时T：发送一个标准长的帧所需的时间
满足Poission分布：

- 一个帧时内用户产生新帧：均值N个
- 一个帧时内信道中产生的帧（包括重传）：均值G个
- $0 < N < 1$，轻载：N接近0，重载：N接
- $G \geq N$，轻载$G = N$，重载$G > N$
- $P(k) = G^k \mathrm{e}^{-G} / k!$（Poission分布）

吞吐量S：发送时间T内发送成功的平均帧数
运载负载G：时间T内所有通信站总共发送帧的平均值，$G \geq S$
P_0为一个帧发送成功的概率，于是

$$S = G \times P_0$$

冲突危险期为前后T，即为$2T$，生成帧的均值为$2G$，不受冲突即$2T$时间内没有帧传输，即连续两个帧时T内没有其他帧生成的概率P_0为连续两个帧时都生成0帧的概率

$$P_0 = P(k = 0)^2 = \mathrm{e}^{-2G}$$

于是$S = G\mathrm{e}^{-2G}$，$G = 0.5$取到最大值0.184，信道的利用率最高为18.4%.

4.1.2 分隙ALOHA协议

把时间分成时隙（time slot），时隙长度对应一帧的时间

帧的发送和冲突只能在时隙的起点

冲突危险期：在同一个时隙进行传输，因而冲突危险期为T

吞吐量：$S = G\mathrm{e}^{-G}$，最大值为0.368，为ALOHA协议的两倍

4.1.3 载波侦听多路访问协议（CSMA）

4.1.3.1 非持续式CSMA

- 侦听，如果介质空闲则发送
- 介质忙，则等待随机的时间重复上一步

缺点：信道利用率低
优点：等待一个随机时间可以减少再次碰撞冲突的可能性

4.1.3.2 持续式（1-持续式）CSMA

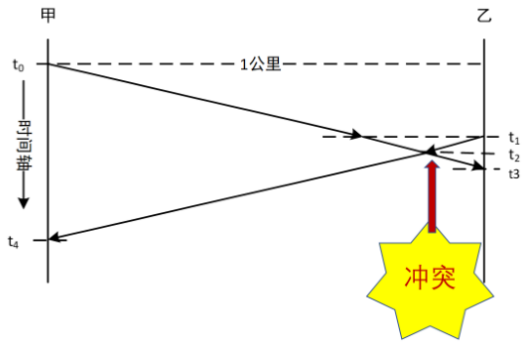
- 侦听，如果介质空闲则发送
- 介质忙，持续侦听，空闲后立即发送
- 如果冲突，等待随机时间再重复第一步

优点：信道利用率高
缺点：如果两个以上的站等待发送，一旦介质空闲就一定会发生冲突

4.1.3.3 p-持续式CSMA

- 侦听，如果介质空闲则以p的概率发送，以$(1-p)$的概率延迟一个单元发送
- 介质忙，持续侦听，空闲后重复上一步
- 如果延迟一个时间单元，重复第一步

CSMA冲突情形：（1）同时传送；（2）传播延迟时间



侦听到冲突：发送帧的时间不能太短，至少要$2D$

4.1.3.4 CSMA/CD

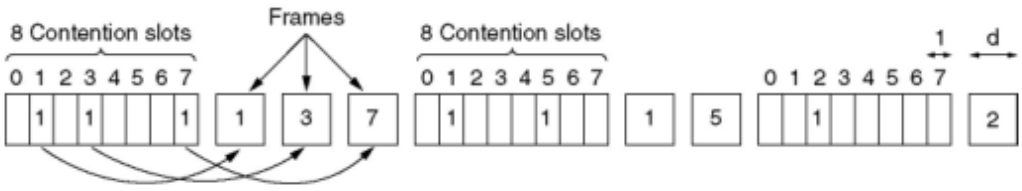
CSMA with Collision Detection

先听后发，边发边听：

- 侦听，如果介质空闲则发送
- 介质忙，持续侦听，空闲后立即发送
- 如果冲突，等待随机时间再重复第一步；并发送Jam（强化）信号（告知冲突产生）

4.2 受控访问协议

4.2.1 位图协议



- 竞争期：在自己的时槽内发送竞争比特（举手示意、资源预留）
- 传输器：按序发送

N个用户，N个时隙，每帧d比特：

- 低负荷信道利用率$d/(d+N)$
- 高负荷信道利用率$d/(d+1)$

缺点：无法考虑优先级

4.2.2 令牌传递

常用于环形拓扑

只有获得令牌才能发送，一次只有一个人能获得令牌。

缺点：负载低的时候，节点需要得到令牌才能通讯，延迟较大的缺点

4.2.3 二进制倒计数协议

站点：编序号，各序号长度相同
竞争期：有数据发送的站点从高序号到低序号排队，高的序号得到发送权（**高序号站点优先**）

优点：存在优先级
缺点：低序号站点可能一直无发送权

4.2.4 随机访问和受控访问对比

低负载时随机访问效率高，高负载时受控访问效率高

4.3 有限竞争协议

4.3.1 自适应树搜索协议

- 在一次成功传输后的第一个竞争时隙，所有站点同时竞争
- 如果只有一个站点申请，则获得信道
- 否则在下一竞争时隙，有一半站点参与竞争（递归），下一时隙由另一半站点参与竞争
- 即所有站点构成一棵完全二叉树

结合了随机访问和受控访问的优点

4.4 以太网

二进制指数后退的CSMA/CD：

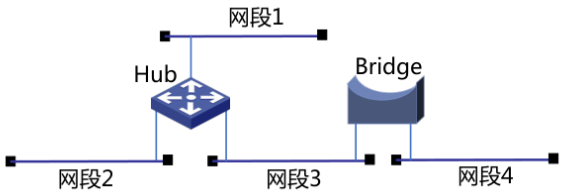
- 基本退避时间槽长度设置为以太介质往返传播时间2τ，设为512比特时间
- 重传次数为k，且$k \leq 10$
- 从整数集合随机$[0, 1, \cdots, 2^k - 1]$取一个数r
- 重传的时延是r倍的时间槽2τ
- 重传16次仍不能成功这丢弃帧，并向高层报告

4.5 数据链路层交换

4.5.1 设备

物理层设备（如集线器Hub）无法隔绝冲突域，只有数据链路层以上的设备可以隔绝冲突域

网桥或交换机：分隔冲突域
网桥的功能：**转发**和**过滤**

> 图中冲突域的个数为?
>
> 
>
>
> 2个冲突域，网桥接口的个数即为冲突域的个数，**集线器无法隔绝冲突域**

理想网桥是透明的：即插即用，无需配置；网络站点无需感知网桥的存在与否

4.5.2 原理

建立MAC地址表：逆向学习发送帧的站的MAC地址，帧到达检查帧的源地址是否在MAC地址表上，如果不在这增加表项，如果在地址表上则更新老化时间

- 增加表项：帧的源地址对应的项不在表中
- 删除表项：老化时间到期

- 更新表项：帧的源地址在表中，更新时间戳

转发过程：逆向学习源地址，根据目的地址，查询MAC地址表

- 如果来源端口和目的端口一样，则丢弃数据帧（在同一冲突域，无需转发）
- 如果来源端口和目的端口不同，根据目的地址查询MAC地址表，找到相应端口转发
- **泛洪**：如果目的地址在MAC地址表上不存在，则将数据帧从所有端口（除了入境口）发送出去

  两种目的地址的帧需要泛洪：

  - 广播帧：MAC地址为FF-FF-FF-FF-FF-FF的帧
  - 未知单播帧

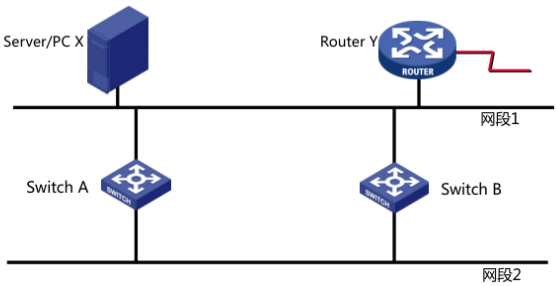
交换模式：

- 存储转发：转发前必须接收整个帧、执行CRC校验

  缺点：延迟大
  优点：不转发出错帧、支持非对称交换
- 直通交换：一旦接收到帧的目的地址，就开始转发

  缺点：可能转发错误帧、不支持非对称交换
  优点：延迟非常小，可以边入边出
- 无碎片交换：接收到帧的前64字节，即开始转发
  缺点：仍可能转发错误帧，不支持非对称交换
  优点：过滤了冲突碎片，延迟和转发错帧介于存储转发和直通交换之间

4.5.3 生成树协议



冗余拓扑：为了保证可靠传输

代价：出现物理环路

引发的问题：

- 广播风暴：物理环路上无休止泛洪广播流量，无限循环，迅速消耗网络资源
- 重复帧：发送到同一网段设备的帧，发到环路的单播帧（假设交换机MAC地址表没有目的路由器的MAC地址）会使得目的设备收到重复帧
- MAC地址表不稳定：
- 当一个帧的多个副本到达不同端口时，交换机会不断修改同一MAC地址对应的端口

生成树选举过程：

- 选举根桥
    - 首先比较优先级，优先级数值最小的交换机胜出成为根桥
    - 如果优先级数值相等，MAC地址最小的交换机成为根桥
- 选举根端口
    - 每个非根桥，通过比较其每个端口到根桥的根路径开销，选出根端口
    - 具有最小根路径开销的端口被选作根端口
    - 如果多个端口的根路径开销相同，则端口ID最小的端口被选作根端口
    - 非根桥只能有一个根端口，根端口处于转发状态
- 为每个网段确定一个指定端口

4.5.4 虚拟局域网

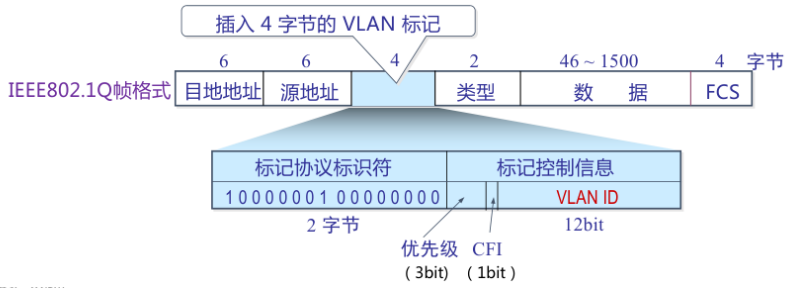
广播域：广播帧能到达的范围

交换机所有端口同属于一个广播域，无法隔离广播域。但支持VLAN的交换机可以隔离广播域，不同VLAN成员之间通讯需要三层设备

VLAN表：记录VLAN ID和端口号/MAC地址/协议/子网的对应关系

区分不同VLAN的数据帧：在数据帧中携带VLAN标记，该标记由支持VLAN的交换机增加和删除，对终端站点透明

帧标记标准：IEEE802.1Q



4.6 无线局域网

4.6.1 CSMA/CD遇到的困难

冲突检测困难：

- 发送功率和接收功率相差太大（有信号衰减）
- 站点的发送时关闭接收功能，无法在发送时同时检测冲突

隐藏终端问题：

- 由于距离太远或障碍物导致站点无法检测到竞争对手的存在
- 隐藏站点不能侦听到发送端但能干扰接收端

暴露终端问题：

- 由于侦听到其他站点的发送而误以为信道忙导致不能发送（导致信道利用率降低）
- 暴露站点能侦听到发送端但不会干扰接收端

4.6.2 CSMA/CA

CA: Collision Avoid

- 信道空闲时间大于帧间隙（IFS），立刻传输
- 信道忙，延迟直到当前传输结束 + IFS
- 开始随机退后过程
    - 从（0，CWindow）中选择一个随机数作为退后计数器
    - 通过侦听确定每个时间槽是否活动

    - 如果没有活动，则减少退后时间
- 退后过程中如果信道忙，则挂起退后过程（解决站点之间的公平问题）
- 在当前帧传输结束后恢复退后过程

IFS用于优先级的调整，IFS短的优先级高

4.6.3 RTS-CTS机制

发送端发送RTS(request to send)，接收端回送CTS(clear to send)

RTS和CTS中的持续时间（Duration）中指明传输所需时间（数据+控制）

其他相关站点能够收到RTS或（和）CTS，维护NAV

通常不使用RTS-CTS，因为会导致信道利用率下降

5 网络层

5.1 功能和服务

功能：

- 路由：选择数据报从源端到目的端的路径
- 转发：将数据报从路由器的输入接口传送到正确的输出接口

服务：

- 无连接服务：不需要提前建立连接（如寄信）

  传输网络**不提供**端到端的可靠传输服务：丢包、乱序、错误的缺点，**不提供服务质量的承诺**
  优点：网络的造价大大降低，运行方式灵活，能够适应多种应用
- 面向连接服务：通信前要建立**逻辑连接**（如打电话）

  先建立连接，再发送数据，最后释放连接

5.2 IP协议（IPv4）

MAC地址（类似身份证）

IP地址（类似门牌）：用于寻址和分片

IPv4协议是**无连接**的协议

IP数据报header:

- 版本： 4bit ， 表示采用的IP协议版本
- 首部长度： 4bit， 表示整个IP数据报首部的长度
- 区分服务： 8bit ， 该字段一般情况下不使用（失败，已不用）
- 总长度： 16bit ， 表示整个IP报文的长度，能表示的最大字节为 $2^{16}-1=65535$字节
- 标识： 16bit ， IP软件通过计数器自动产生，每产生1个数据报计数器加1；在IP分片以后，用来标识同一片分片
- 标志： 3bit，目前只有两位有意义； MF，置1表示后面还有分片，置0表示这是数据报片的最后1个；DF，不能分片标志，置0时表示允许分片
- 片偏移： 13bit，表示IP分片后，相应的IP片在总的IP片的相对位置
- 生存时间TTL(Time To Live)： 8bit,表示数据报在网络中的生命周期，用通过路由器的数量来计量，即跳数（每经过一个路由器会减1，防止无限传输）
- 协议：8bit，标识上层协议（TCP/UDP/ICMP…）
- 首部校验和：16bit ，对数据报**首部**进行校验，不包括数据部分
- 源地址：32bit，标识IP片的发送源IP地址
- 目的地址：32bit，标识IP片的目的地IP地址

- 选项：可扩充部分，具有可变长度，定义了安全性、严格源路由、松散源路由、记录路由、时间戳等选项
- 填充：用全0的填充字段补齐为4字节的整数倍

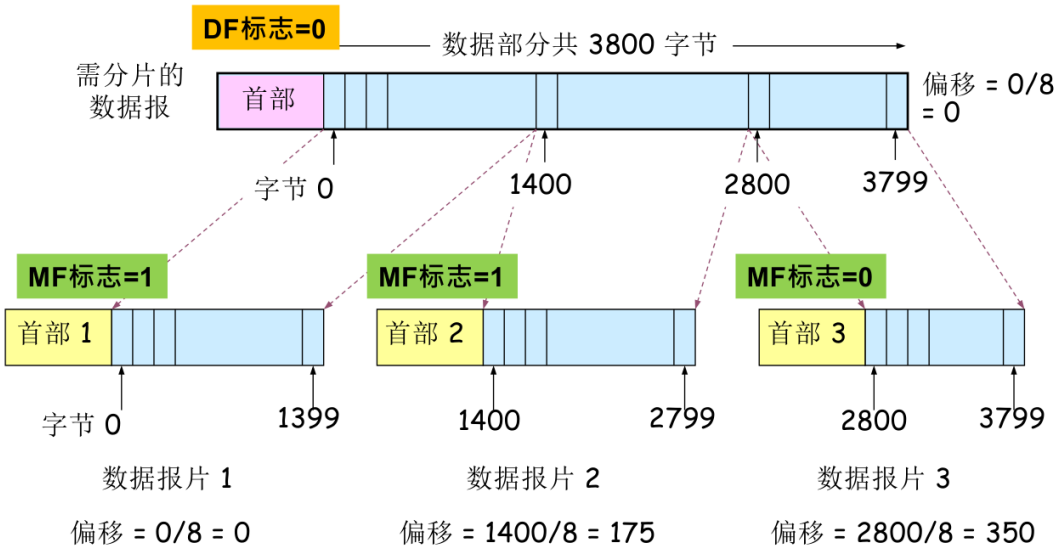
首部中各数据的功能：

- 支持多跳寻路将IP数据报送达目的端：目的IP地址
- 表明发送端身份：源IP地址
- 根据IP头部协议类型，提交给不同上层协议处理：协议
- 数据报长度大于传输链路的MTU的问题，通过分片机制解决：标识、标志、片偏移
- 防止循环转发浪费网络资源（路由错误、设备故障…），通过跳数限制解决：生存时间TTL
- IP报头错误导致无效传输，通过头部机校验解决：首部校验和

5.2.1 分片

最大传输单元MTU：链路MTU、路径MTU

链路MTU大于路径MTU则需要分片



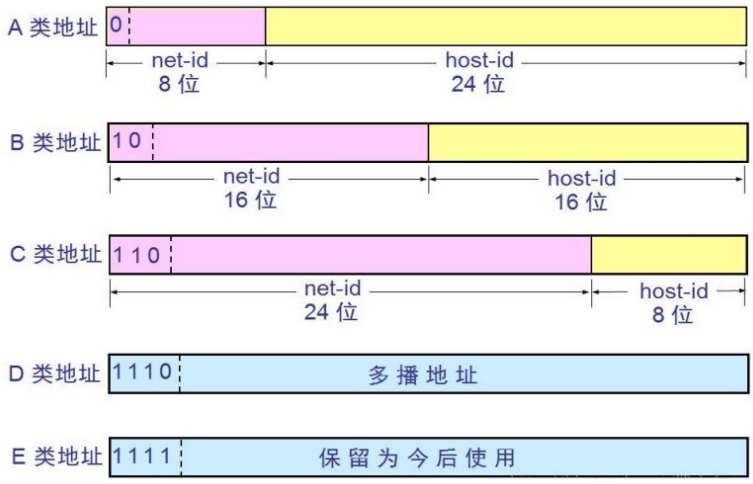
第一次分片修改标志和片偏移，后续再次分片可以直接修改标志和片偏移（片偏移的单位为8B）

分片越多，信道利用率降低（需要传输多余的首部）

重组策略：在**目的端**重组

5.2.2 IP地址

IP地址：网络上的每一台主机（或路由器）的每一个接口都会分配一个全球唯一的32位的标识符



特殊IP地址：

地址	用途
全0网络地址	只在系统启动时有效，用于启动时临时通信，又叫主机地址
网络127.0.0.0	指本地节点(一般为127.0.0.1)，用于测试网卡及TCP/IP软件，这样浪费了1700万个地址
全0主机地址	用于指定网络本身，称之为网络地址或者网络号
全1主机地址	用于广播，也称定向广播，需要指定目标网络
0.0.0.0	指任意地址
255.255.255.255	用于本地广播，也称有限/受限广播，无须知道本地网络地址

子网划分(subnetting)：在网络内部将一个网络块进行划分以供多个内部网络使用，对外仍是一个网络
子网(subnet)：一个网络进行子网划分后得到的一系列结果网络称为子网
子网掩码(subnet mask)：与 IP 地址一一对应，是32 bit 的二进制数，置1表示网络位，置0表示主机位

n位的子网掩码最多有$(2^{32-n}-2)$台主机（注意全0和全1的主机号是保留的，不能作为主机号）

5.2.3 最长前缀匹配

IP地址与IP前缀匹配时，总是选取子网掩码最长的匹配项

5.2.4 DHCP协议（动态主机配置协议）

- DHCP 客户从UDP端口68以**广播形式**向服务器发送发现报文（**DHCPDISCOVER**）
- DHCP 服务器**单播**发出提供报文（**DHCPOFFER**）
- DHCP 客户从多个DHCP服务器中选择一个，并向其以**广播**形式发送
- DHCP请求报文（**DHCPREQUEST**）
- 被选择的DHCP服务器**单播**发送确认报文（**DHCPACK**）

5.2.5 同时需要MAC和IP的原因

- 只有 MAC、不要 IP Address
    - MAC 是固定的，难以用于路由寻址（移动主机后MAC地址仍然是同一个，难以实现寻址）
- 只有 IP、不要 MAC Address
    - IP 属于网络层、MAC 属于数据链路层，没有 MAC 的话每次需要解析网络层才能进行转发
        - 网络层与数据链路层合并
        - 不再有交换机/网桥
    - 帧的发送需要操作系统进入内核态处理（处理时间更长）

5.3 ARP地址解析协议

功能：给定IP地址，获得MAC地址

方法：每一个主机维护一个IP/MAC地址的表

如果在同一个局域网下

- 主机A广播发送ARP请求分组（包含自身的IP地址和MAC地址以及主机B的IP地址）
- 主机B向A单播发送ARP响应分组，A得到了B的MAC地址，记入表中

- 其他主机也可以接收到A的ARP请求分组，可以将A的MAC地址记入表中

不在同一局域网

- 主机A广播发送ARP请求分组（包含自身的IP地址和MAC地址以及主机C的IP地址）
- 在源主机A的路由表中找到路由器R的IP地址
- A根据R的IP地址，使用ARP协议获得R的MAC地址
- A创建数据帧（目的地址为R的MAC地址）
- 数据帧中封装A到C的IP数据包
- A发送数据帧，R接收数据帧

5.4 NAT网络地址转换

将私有（保留）地址转化为公有IP地址的转换技术

NAT转换表：广域网地址及端口和局域网地址及端口的对照表

私有地址：

- A类地址：10.0.0.0 -10.255.255.255
- B类地址：172.16.0.0 - 172.31.255.555
- C类地址：192.168.0.0 - 192.168.255.255

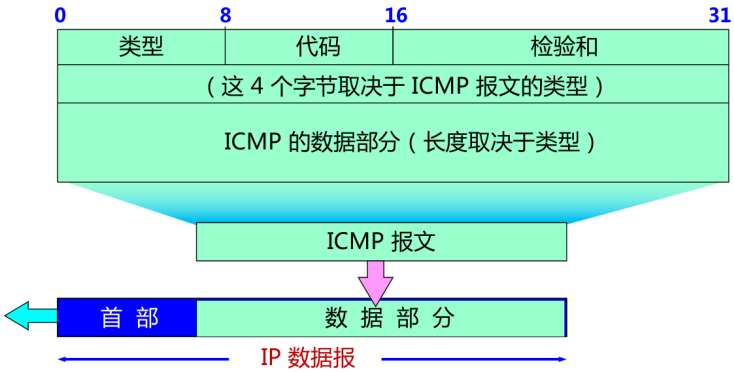
优点：

- 节省合法地址，减少地址冲突
- 灵活连接Internet
- 保护局域网的私密性

缺点：

- 违反了IP的结构模型，路由器处理传输层协议（端口号）
- 违反了端到端的原则（通过路由器进行IP转换，不是端到端的）
- 违反了最基本的协议分层规则
- 不能处理IP报头加密（加密后无法对IP地址进行转换）
- 新型网络应用的设计者必须要考虑 NAT场景，如 P2P应用程序

5.5 ICMP协议



ICMP允许主机或路由器报告差错情况和提供有关异常情况的报告

Ping(Packet InterNet Groper)：检测两个主机的连通性，通过ICMP回送请求和回答报文

Traceroute原理:

- 源向目的地发送一系列UDP段(不可能的端口号)：第一个 TTL =1；第二个 TTL=2, 等
- 当第n个数据报到达第n和路由器:
  - 路由器丢弃数据报
  - 并向源发送一个ICMP报文 (类型 11, 编码0)
  - 报文的源IP地址就是该路由器的IP地址
- 当源收到ICMP报文，计算 RTT
- Tracert针对同一RTT值执行上述过程3次

5.6 路由

汇集树：所有的源节点到一个指定目标节点的最优路径的集合构成一棵以目标节点为根的树

最短路径：Dijkstra算法

路由算法：距离向量路由、链路状态路由

距离向量路由：

- 每个节点周期性地向邻居发送它自己到某些节点的距离向量；

- 当节点x接收到来自邻居的新DV估计，它使用B-F方程更新其自己的DV

$$D_x(y) \leftarrow \min_v\{c(x,v)+D_v(y)\}$$

- 迭代执行直至收敛

某一网络掉线：存在计数到无穷问题，需要更新路由表至最大值才能检测到

链路状态路由：

- 发现邻居，了解他们的网络地址；
- 设置到每个邻居的成本度量；
- 构造一个分组，分组中包含刚收到的所有信息；
- 将此分组发送给其他的路由器；
- 计算到其他路由器的最短路径（Dijkstra算法）。

层次路由：提高路由器查表速度，减少路由表存储空间

广播路由：发送给所有使用者

- 方法1：每个主机单独发一个数据报
- 方法2：多目标路由（难以实现）
- 方法3：泛洪
    - 存在广播风暴问题
    - 解决方案：序号控制泛洪，逆向路径转发
- 方法4：生成树

组播路由：发送给特定使用者

- 生成树
- 稀疏分布：基于核心树
- 密集分布：基于源点树

选播路由：将数据包传送给最近的一个组成员；在有多个服务器的情况下，用户希望快速获得正确信息，而不在乎从哪个服务器获得

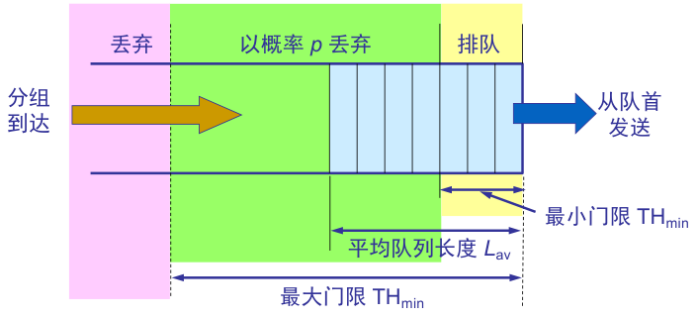
5.7 拥塞控制算法

拥塞：存在太多的数据包导致数据包传输延迟或丢失，从而导致网络吞吐量下降

拥塞控制的途径：

- 提高网络供给（物理层实现）
- 流量感知路由（网络层实现）
- 准入控制（网络层实现）
- 流量调节：隐式流量调节（传输层实现）、显式流量调节（网络层实现）
- 负载丢弃（网络层实现）：随机早期检测RED

5.7.1 随机早期检测RED



实现：

- 使路由器的队列维持两个参数，即队列长度最小门限 $TH_{\min}$ 和最大门限 $TH_{\max}$
- RED对每一个到达的数据报都先计算平均队列长度 L_{AV}
- 若平均队列长度小于最小门限 $TH_{\min}$ ，则将新到达的数据报放入队列进行排队
- 若平均队列长度超过最大门限 $TH_{\max}$，则将新到达的数据报丢弃
- 若平均队列长度在最小门限$TH_{\min}$和最大门限$TH_{\max}$之间，则按照某一概率p将新到达的数据报丢弃

早期检测后丢弃包会使得发送方降低发送速度

以相同概率p丢弃包，每一个包被丢弃概率相同，发的快发的多的使用者的丢包率更大，可以更好的抑制该使用者的发送速率

6 传输层

6.1 功能

实现**端到端**的传输

提供进程之间本地通信的抽象

最低限度的传输服务：

- 将终端-终端的数据交付扩展到进程-进程的数据交付
- 报文检错

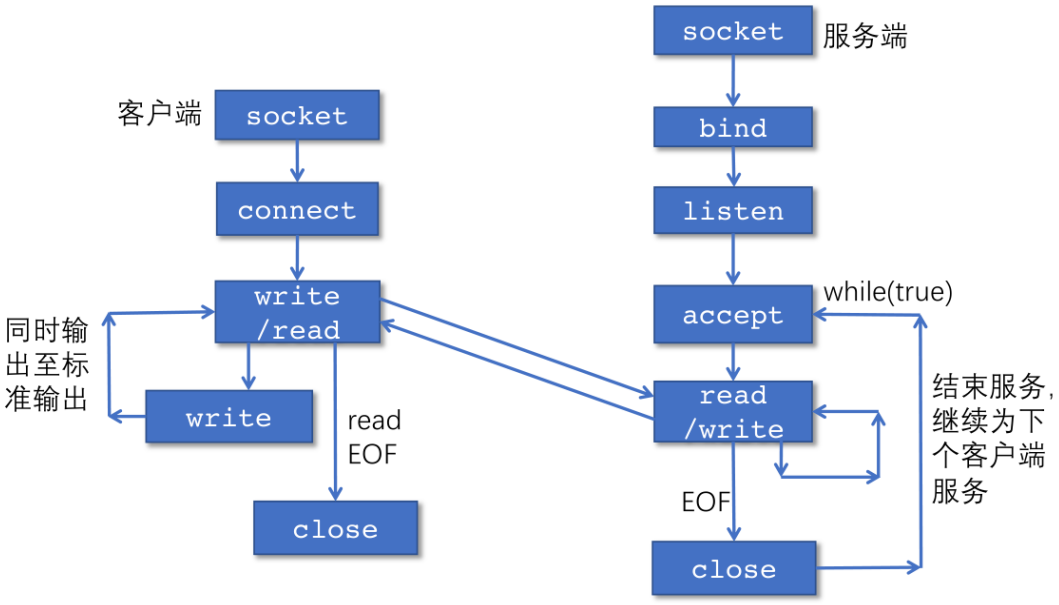
增强服务：

- 可靠数据传输
- 流量控制
- 拥塞控制

协议：UDP、TCP

同一主机进程的标识：端口号

6.2 套接字



复用：发送方传输层将套接字标识置于报文段中，交给网络层
分用：接收方传输层根据报文段中的套接字标识，将报文段交付到正确的套接字

端口号：每个套接字在本地关联一个端口号，是一个16bit数

报文段中有两个字段携带端口号：

- 源端口号：与发送进程关联的本地端口号
- 目的端口号：与接收进程关联的本地端口号

6.3 UDP

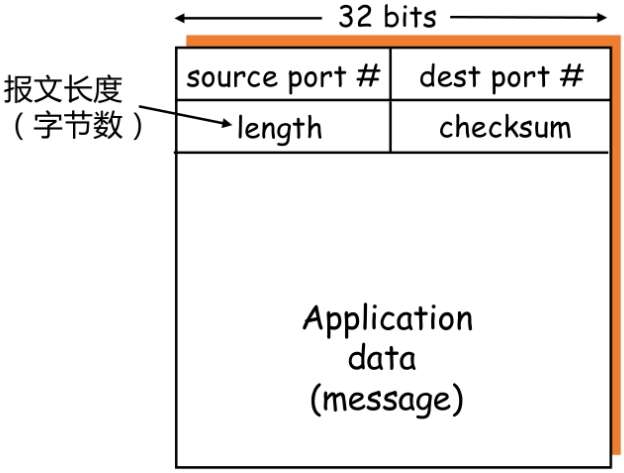
UDP是**无连接**的传输

UDP提供的服务：

- 进程到进程之间的报文交付
- 报文完整性检查（可选）：检测并丢弃出错的报文

实现功能：复用和分用；报文检错

UDP报文：



- 报头：携带协议处理需要的信息
    - 源端口号（用于复用和分用）
    - 目的端口号（用于复用和分用）
    - 校验和（检测报文错误）

    - 报文长度（检测报文错误）
- 载荷：携带上层数据

UDP校验和计算：计算校验和包括伪头（信息来自IP报头，包括源IP地址、目的IP地址、UDP协议号、UDP报文端长度）、UDP头和数据三个部分

UDP校验和的使用是可选的，若不计算校验和，该字段填入0

UDP优点：

- 可以尽可能快发送报文
    - 无建立连接延迟
    - 不限制发送速率：无拥塞控制和流量控制
- 报头开销小
- 协议处理简单

UDP的应用范围：

- 容忍丢包但对延迟敏感的应用
- 以单次请求、响应为主的应用
- 应用要求基于UDP进行可靠传输（应用层实现其可靠性）

6.4 TCP

TCP实现的是**面向连接**的传输

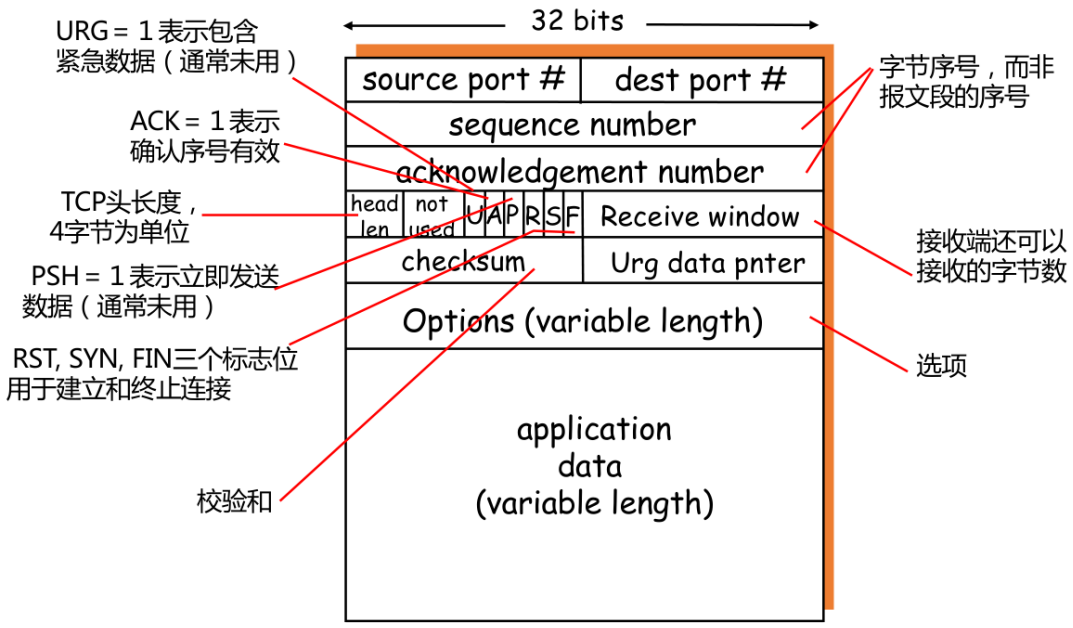
TCP 在不可靠的IP服务上建立可靠的数据传输

模型：在一对通信的进程之间提供一条理想的字节流管道

机制：

- 建立连接：通信双方为本次通信建立数据传输所需的状态（套接字、缓存、变量等）
- 可靠数据传输：流水线式发送，报文段检错，丢失重传
- 流量控制：发送方不会令接收方缓存溢出

TCP报文：



发送序号：数据载荷中第一个字节在字节流中的序号
确认序号：期望接收的下一个字节的序号

初始序号的选取: 每个TCP实体维护一个32位计数器，该计数器每4微秒增1，建立连接时从中读取计数器当前值（避免数字重复）

TCP发送方：

- 收到应用数据：
  - 创建并发送TCP报文段
  - 若当前没有定时器在运行（没有已发送、未确认的报文段），启动定时器
- 超时:
  - 重传包含最小序号的、未确认的报文段
  - 重启定时器
- 收到ACK：
  - 如果确认序号大于基序号（已发送未确认的最小序号）：推进发送窗口（更新基序号）
  - 如果发送窗口中还有未确认的报文段，启动定时器，否则终止定时器

6.4.1 差错恢复机制

接收方：

- 使用累积确认
- 缓存失序的报文段
- 对失序报文段发送重复ACK
- 增加了推迟确认

发送方：

- 超时后仅重传最早未确认的报文段
- 增加了快速重传

结合了GBN和SR的优点

- 定时器的使用：与GBN类似，只对最早未确认的报文段使用一个定时器
- 超时重传：与SR类似，只重传缺失的数据

6.4.2 流量控制

UDP不需要流量控制：接收端UDP将收到的报文载荷放入接收缓存；应用进程每次从接收缓存中读取一个完整的报文载荷；当应用进程消费数据不够快时，接收缓存溢出，报文数据丢失，UDP不负责任

TCP接收端有接受缓存：接收端TCP将收到的数据放入接收缓存，应用进程从接收缓存中读数据，进入接收缓存的数据不一定被立即取走、取完（因此**TCP需要流量控制**）

流量控制方法：将接收窗口大小（缓存中可用空间）RcvWindow放在报头中告知发送方

非零窗口通告：当接收窗口为0时，发送方必须停止发送；当接收窗口变为非0时，接收方应通告增大的接收窗口。发送方收到“零窗口通告”后，可以发送“零窗口探测”报文段

6.4.2.1 AIMD

乘性减（Multiplicative Decrease）：

- 发送方检测到丢包后，将cwnd的大小减半（但不能小于一个MSS）
- 目的：迅速减小发送速率，缓解拥塞

加性增（Additive Increase）：

- 若无丢包，每经过一个RTT，将cwnd增大一个MSS，直到检测到丢包
- 目的：缓慢增大发送速率，避免振荡