计算机基础主干知识详细拓展版（学习|复习|面试）

> **我不会直接把书上的知识点复制过来，而是加上相关理解和拓展问题，你有一桶水，才就能洒出半桶水！加油！**
>
> **有更新和优化会在公众号【程序员宝藏】提醒！请注意查收！（持续更新，不细你来找我要红包！）**

目录：

一、计算机网络

1、TCP三次握手与四次挥手

1.TCP 协议的特点

TCP是在不可靠的IP层之上实现的可靠的数据传输协议，它主要解决传输的可靠、有序、无丢失和不重复问题。

TCP 是TCP/IP 体系中非常复杂的一个协议，主要特点如下：
1) TCP 是面向连接的传输层协议。
2) 每条TCP 连接只能有两个端点，每条TCP 连接只能是点对点的（一对一）。
3) TCP 提供可靠的交付服务，保证传送的数据无差错、不丢失、不重复且有序。

> 这里是一个提问点：如何保证数据无差错、不丢失、不重复且有序的？有哪些机制来保证？
> 答：TCP 使用了校验、序号、确认和重传等机制来达到这一目的。

4) TCP 提供全双工通信，允许通信双方的应用进程在任何时候都能发送数据，为此TCP 连接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据。

> 这里是一个提问点：为什么需要设置缓存，缓存的作用？
>
> 答：发送缓存用来暂时存放以下数据：1.发送应用程序传送给发送方TCP 准备发送的数据；2.TCP 已发送但尚未收到确认的数据。
>
> 接收缓存用来暂时存放以下数据：1.按序到达但尚未被接收应用程序读取的数据；2.不按序到达的数据。

5) TCP是面向字节流的，虽然应用程序和TCP的交互是一次一个数据块（大小不等），但TCP把应用程序交下来的数据仅视为一连串的无结构的字节流。

> 一个字节占一个序号，每个报文段用第一个字节的序号来标识,例如，一报文段的序号字段值是301, 而携带的数据共有l00B, 表明本报文段的数据的最后一个字节的序号是400, 因此下一个报文段的数据序号应从401开始，也就是期望的下一个序号（确认号）。

2.TCP报文段格式

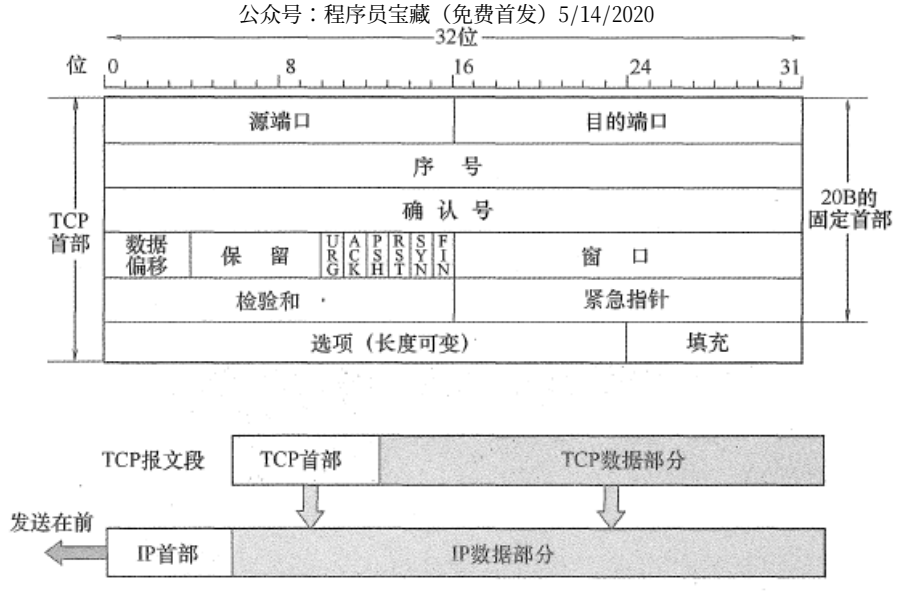


图 5.6 TCP 报文段

> 部分字段解释：
>
> 1) 序号字段（就是seq）：序号字段的值指的是本报文段所发送的数据的第一个字节的序号。
>
> 2) 确认号字段（就是ack）：是期望收到对方的下一个报文段的数据的第一个字节的序号。若确认号为N, 则表明到序号N-1为止的所有数据都已正确收到。（累积确认）
>
> 3) 确认位ACK：只有当ACK=1时确认号字段才有效。当ACK=0时，确认号无效。TCP 规定，在连接建立后所有传送的报文段都必须把ACK置1。
>
> 4) 同步位SYN：同步SYN=1表示这是一个连接请求或连接接收报文。当SYN= 1, ACK=0 时，表明这是一个连接请求报文，对方若同意建立连接，则在响应报文中使用SYN= 1, ACK=1。即SYN=1表示这是一个连接请求或连接接收报文。
>
> 5) 终止位FIN (Finish)：用来释放一个连接。FIN=1表明此报文段的发送方的数据已发送完毕了并要求释放传输连接。

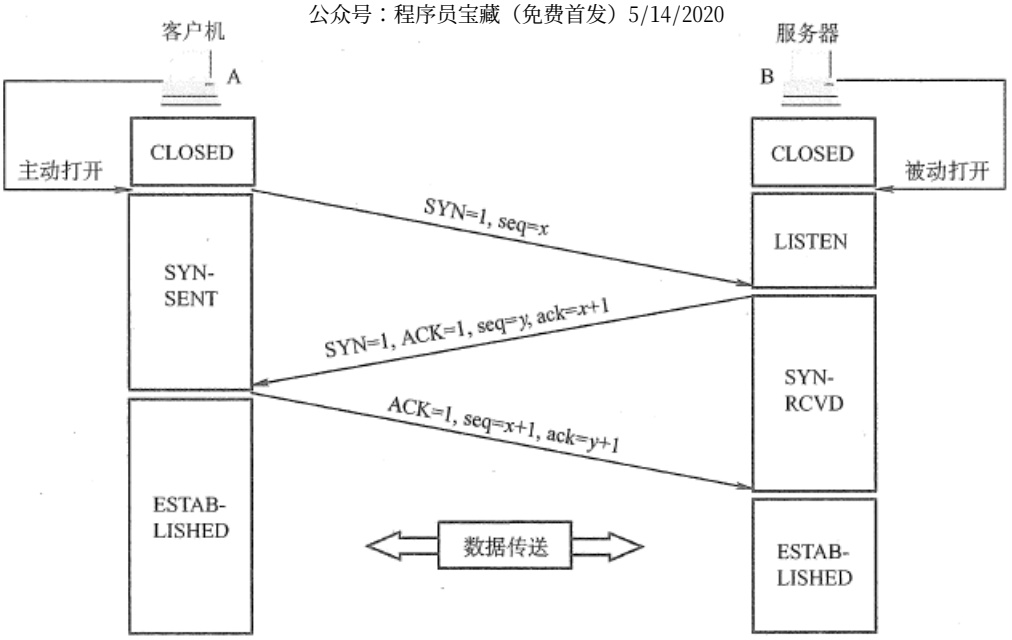
3.TCP连接管理

TCP 是面向连接的协议，因此每个TCP 连接都有三个阶段：连接建立、数据传送和连接释放。

TCP 连接的管理就是使运输连接的建立和释放都能正常进行。
在TCP 连接建立的过程中，要解决以下三个问题：
1) 要使每一方都能够确知对方的存在。
2) 要允许双方协商一些参数（如最大窗口值、是否使用窗口扩大选项、时间戳选项及服务质量等）。
3) 能够对运输实体资源（如缓存大小、连接表中的项目等）进行分配。

每条TCP 连接唯一地被通信两端的两个端点（即两个套接字）确定。`端口拼接到IP地址即为套接字`

4.三次握手建立连接



1) 第一次握手：客户机的TCP首先向服务器的TCP发送一个连接请求报文段。这个特殊的报文段中不含应用层数据，其首部中的SYN标志位被置为1。另外，客户机会随机选择一个起始序号seq = x（连接请求报文不携带数据，但要消耗一个序号）。
2) 第二次握手：服务器的TCP 收到连接请求报文段后，如同意建立连接，就向客户机发回确认，并为该TCP连接分配TCP缓存和变量。在确认报文段中，SYN 和ACK 位都被置为1, 确认号字段的值为x+1,并且服务器随机产生起始序号seq= y(确认报文不携带数据，但也要消耗一个序号）。确认报文段同样不包含应用层数据。
3) 第三次握手：当客户机收到确认报文段后，还要向服务器给出确认，并且也要给该连接分配缓存和变量。这个报文段的ACK 标志位被置1, 序号字段为x+1, 确认号字段ack=y+1。该报文段可以携带数据，若不携带数据则不消耗序号`http中的tcp连接的第三次握手的报文段中就捎带了客户对万维网文档的请求`。

成功进行以上三步后，就建立了TCP 连接，接下来就可以传送应用层数据。TCP 提供的是全双工通信，因此通信双方的应用进程在任何时候都能发送数据。

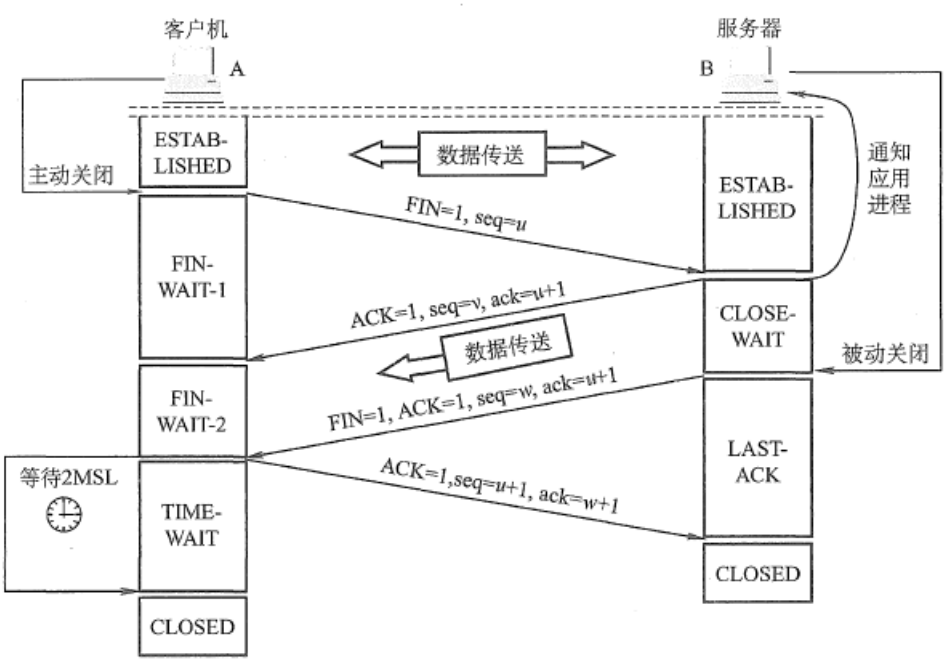
【总结】：
1) SYN = 1,ACK = 0,seq = x;
2) SYN = 1,ACK = 1,seq = y,ack = x+1;
3) SYN = 0,ACK = 1,seq = x+1,ack=y+1。

> 【拓展问题1】：什么是SYN洪泛攻击？（三次握手机制有什么问题？）
>
> 答：由于服务器端的资源是在完成第二次握手时分配的，而客户端的资源是在完成第三次握手时分配的，攻击者发送TCP的SYN报文段，SYN是TCP三次握手中的第一个数据包，而当服务器返回ACK后，该攻击者就不对其进行再确认，那这个TCP连接就处于挂起状态，也就是所谓的半连接状态，服务器收不到再确认的话，还会重复发送ACK给攻击者。这样更加会浪费服务器的资源。攻击者就对服务器发送非常大量的这种TCP连接，由于每一个都没法完成三次握手，所以在服务器上，这些TCP连接会因为挂起状态而消耗CPU和内存，最后服务器可能死机，就无法为正常用户提供服务了。

> 【拓展问题2】：如果已经建立了连接，但是客户端突然出现故障了怎么办?
> TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

> 【拓展问题3】：为什么不采用“两次握手”建立连接呢？
> 答：这主要是为了防止两次握手情况下已失效的连接请求报文段突然又传送到服务器而产生错误。考虑下面这种情况。客户A 向服务器B 发出TCP 连接请求，第一个连接请求报文在网络的某个结点长时间滞留， A 超时后认为报文丢失，于是再重传一次连接请求， B 收到后建立连接。数据传输完毕后双方断开连接。而此时，前一个滞留在网络中的连接请求到达服务器B, 而B 认为A 又发来连接请求，此时若使用“三次握手”，则B 向A 返回确认报文段，由于是一个失效的请求，因此A 不予理睬，建立连接失败。若采用的是“两次握手”，则这种情况下B 认为传输连接已经建立，并一直等待A 传输数据，而A 此时并无连接请求，因此不予理睬，这样就造成了B的资源白白浪费。

5.四次握手释放连接



1) 第一次握手：客户机打算关闭连接时，向其TCP发送一个连接释放报文段，并停止发送数据，主动关闭TCP 连接，该报文段的FIN 标志位被置1, seq= u, 它等于前面已传送过的数据的最后一个字节的序号加1 (FIN 报文段即使不携带数据，也要消耗一个序号）。TCP是全双工的，即可以想象为一条TCP 连接上有两条数据通路。发送FIN 报文时，发送FIN 的一端不能再发送数据，即关闭了其中一条数据通路，但对方还可以发送数据。
2) 第二次握手：服务器收到连接释放报文段后即发出确认，确认号是ack = u + 1, 而这个报文段自己的序号是v, 等千它前面已传送过的数据的最后一个字节的序号加1 。此时，从客户机到服务器这个方向的连接就释放了，TCP连接处千半关闭状态。但服务器若发送数据，客户机仍要接收，即从服务器到客户机这个方向的连接并未关闭。
3) 第三次握手：若服务器已经没有要向客户机发送的数据，就通知TCP释放连接，此时其发出FIN=1的连接释放报文段。
4) 第四次握手：客户机收到连接释放报文段后，必须发出确认。在确认报文段中，ACK字段被置为1, 确认号ack= w + 1, 序号seq= u + 1 。此时TCP连接还未释放，必须经过时间等待计时器设置的时间2MSL（最长报文段寿命）后，A才进入连接关闭状态。

【总结】：
1) FIN = 1,seq = u;
2) ACK = 1,seq = v,ack = u+1;
3) FIN = 1,ACK = 1,seq = w,ack =u+1;(确认第一次的u)
4) ACK = 1,seq = u+1,ack = w+1。

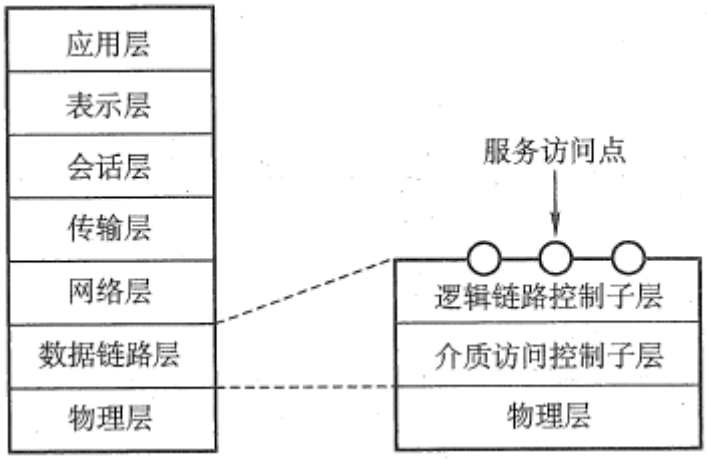
> 【拓展问题1】为什么连接的时候是三次握手，关闭的时候却是四次握手？
> 答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四次握手。

> 【拓展问题2】为什么TIME_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？
> 答：1)虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是网络是不可靠的，有可能最后一个ACK丢失。所以TIME_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。
>
> 2)防止出现"已失效的连接请求报文段"（和上面的为啥不用二次握手类似）。A 在发送最后一个确认报文段后，再经过2MSL可保证本连接持续的时间内所产生的所有报文段从网络中消失。

2.介质访问控制

1.介质访问控制综述

局域网的数据链路层分为逻辑链路层LLC和介质访问控制MAC两个子层。逻辑链路控制（Logical Link Control或简称LLC）是局域网中数据链路层的上层部分，IEEE 802.2中定义了逻辑链路控制协议。用户的数据链路服务通过LLC子层为网络层提供统一的接口。在LLC子层下面是MAC子层。MAC(medium access control)属于LLC（Logical Link Control）下的一个子层，提供介质访问控控制的功能。模型图如下：



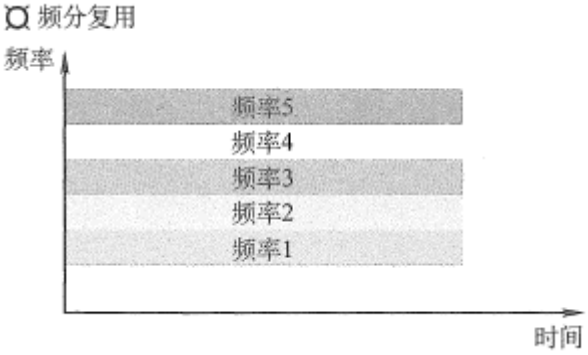
1. 为什么需要介质访问控制？ 因为局域网是一种广播式的网络（广域网是一种点对点的网络），所有联网计算机都共享一个公共信道，所以，需要一种方法能有效地分配传输介质的使用权，使得两对结点之间的通信不会发生相互干扰的情况，这种功能就叫介质访问控制。
2. 介质访问控制的分类？ 常见的介质访问控制方法有信道划分介质访问控制、随机访问介质访问控制和轮询访问介质访问控制。其中前者是静态划分信道的方法，而后两者是动态分配信道的方法。

2.信道划分介质访问控制

信道划分介质访问控制将使用介质的每个设备与来自同一通信信道上的其他设备的通信隔离开来，把时域和频域资源合理地分配给网络上的设备。信道划分的实质就是通过分时、分频、分码等方法把原来的一条广播信道，逻辑上分为几条用于两个结点之间通信的互不干扰的子信道，实际上就是把广播信道转变为点对点信道。 信道划分介质访问控制分为以下4 种：

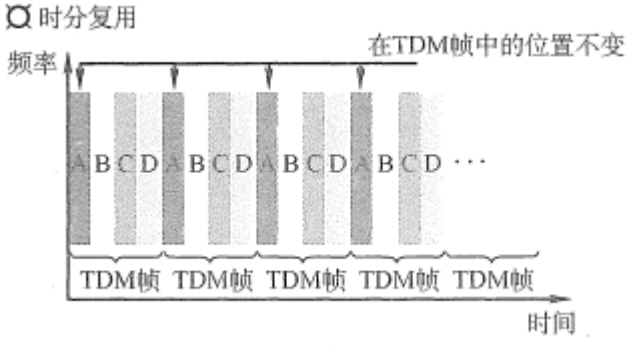
频分多路复用（Frequency division multiplexing FDM）

- 频分多路复用是一种将多路基带信号调制到不同频率载波上，再叠加形成一个复合信号的多路复用技术。
- 每个子信道分配的带宽可不相同，但它们的总和必须不超过信道的总带宽。在实际应用中，为了防止子信道之间的干扰，相邻信道之间需要加入“保护频带”。
- 频分多路复用的优点在于充分利用了传输介质的带宽，系统效率较高；由于技术比较成熟，实现也较容易。缺点在于无法灵活地适应站点数及其通信量的变化。



时分多路复用（Time division multiplexing TDM）

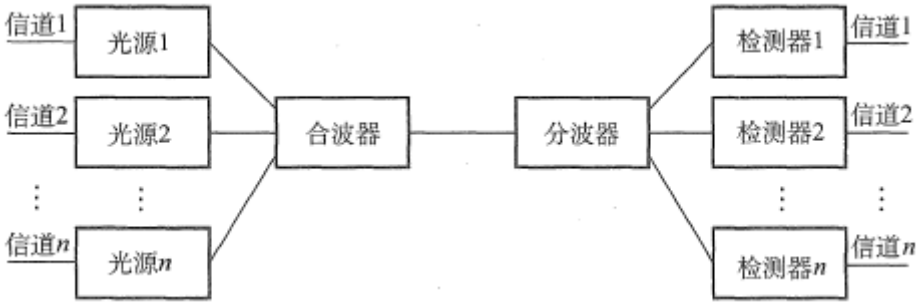
- 时分多路复用是将一条物理信道按时间分成若干时间片，轮流地分配给多个信号使用。每个时间片由复用的一个信号占用。
- 就某个时刻来看，时分多路复用信道上传送的仅是某一对设备之间的信号；就某段时间而言，传送的是按时间分割的多路复用信号。
- 但由于计算机数据的突发性，一个用户对已经分配到的子信道的利用率一般不高。所以对TDM进行改进，有了统计时分多路复用（STDM），它采用STDM 帧， STDM 帧并不固定分配时隙，而按需动态地分配时隙，当终端有数据要传送时，才会分配到时间片，因此可以提高线路的利用率。



波分多路复用（Wavelength division multiplexing　WDM）

- 波分多路复用即光的频分多路复用，它在一根光纤中传输多种不同波长（频率）的光信号，由于波长（频率）不同，各路光信号互不干扰，最后再用波长分解复用器将各路波长分解出来。由于光波

处于频谱的高频段，有很高的带宽，因而可以实现多路的波分复用.



码分多路复用(Code division multiplexing CDM)

- 码分多路复用是采用不同的编码来区分各路原始信号的一种复用方式。与FDM 和TDM 不同，它既共享信道的频率，又共享时间。
- 码分多址(Code Division Multiple Access, CDMA) 是码分复用的一种方式，其原理是每比特时间被分成m个更短的时间槽，称为码片(Chip), 通常情况下每比特有64 或128 个码片。每个站点被指定一个唯一的m 位代码或码片序列。发送1 时，站点发送码片序列；发送0 时，站点发送码片序列的反码。当两个或多个站点同时发送时，各路数据在信道中线性相加。为从信道中分离出各路信号，要求各个站点的码片序列相互正交。简单理解就是， A 站向C 站发出的信号用一个向量来表示， B 站向C 站发出的信号用另一个向量来表示，两个向量要求相互正交。向量中的分量，就是所谓的码片。(相关计算细节感兴趣的可自行百度)
- 码分多路复用技术具有频谱利用率高、抗干扰能力强、保密性强、语音质量好等优点，还可以减少投资和降低运行成本，主要用千无线通信系统，特别是移动通信系统。

> “
>
> 【一个生动形象的例子帮忙理解和总结】假设A 站要向C 站运输黄豆， B 站要向C 站运输绿豆，A 与C 、B 与C 之间有一条公共的道路，可以类比为广播信道。 在频分复用方式下，公共道路被划分为两个车道，分别提供给A 到C 的车和B 到C 的车行走，两类车可以同时行走，但只分到了公共车道的一半，因此频分复用（波分复用也一样）共享时间而不共享空间。 在时分复用方式下，先让A 到C 的车走一趟，再让B 到C 的车走一趟，两类车交替地占用公共车道。公共车道没有划分，因此两车共享了空间，但不共享时间。 码分复用与另外两种信道划分方式大为不同，在码分复用情况下，黄豆与绿豆放在同一辆车上运送，到达C 后，由C 站负责把车上的黄豆和绿豆分开。因此，黄豆和绿豆的运送，在码分复用的情况下，既共享了空间，也共享了时间。
>
> ”

3.随机访问介质访问控制

在随机访问协议中，不采用集中控制方式解决发送信息的次序问题，所有用户能根据自己的意愿随机地发送信息，占用信道全部速率。在总线形网络中，当有两个或多个用户同时发送信息时，就会产生帧的冲突（碰撞），导致所有冲突用户的发送均以失败告终。为了解决随机接入发生的碰撞，每个用户需要按照一定的规则反复地重传它的帧，直到该帧无碰撞地通过。这些规则就是随机访问介质访问控制协议，常用的协议有ALOHA 协议、CSMA协议、CSMA/CD 协议和CSMA/CA 协议等，它们的核心思想都是：胜利者通过争用获得信道，从而获得信息的发送权。因此，随机访问介质访问控制协议又称争用型协议。随机介质访问控制实质上是一种将广播信道转化为点到点信道的行为。

ALOHA协议

- 纯ALOHA 基本思想: 1.用户有数据要发送时，就让它们发送 2.然后监听信道是否产生冲突，若产生冲突，则等待一段随机的时间重发
- 分槽(时隙)ALOHA 基本思想: 1.把时间分成离散的间隔，每个间隔对应于发送一帧所需时间 2.每个站点只能等到下一个时槽开始时才允许发送 3.其他过程与纯ALOHA相同
- 分槽(时隙)ALOHA网络的吞吐量比纯ALOHA协议大了一倍。

CSMA协议（Carrier sense multiple access）

- 时隙ALOHA 系统的效率虽然是纯ALOHA 系统的两倍，但每个站点都是随心所欲地发送数据的，即使其他站点正在发送也照发不误，因此发送碰撞的概率很大。若每个站点在发送前都先侦听一下共用信道，发现信道空闲后再发送，则就会大大降低冲突的可能，从而提高信道的利用率，载波侦听多路访问协议依据的正是这一思想。CSMA 协议是在ALOHA 协议基础上提出的一种改进协议，它与ALOHA协议的主要区别是多了一个载波侦听装置。（先听后发）

表 3.1　三种不同类型的 CSMA 协议比较

信道状态	1-坚持	非坚持	p-坚持
空闲	立即发送数据	立即发送数据	以概率 p 发送数据，以概率 $1-p$ 推迟到下一个时隙
忙	继续坚持侦听	放弃侦听，等待一个随机的时间后再侦听	持续侦听，直至信道空闲

CSMA/CD 协议

载波侦听多路访问 / 碰撞检测(Carrier Sense Multiple Access with Collision Detection, CSMA/CD)(这些单词大家背一背没有坏处)协议是CSMA 协议的改进方案，适用于总线形网络或半双工网络环境。”载波帧听”就是发送前先侦听，即每个站在发送数据之前先要检测一下总线上是否有其他站点正在发送数据，若有则暂时不发送数据，等待信道变为空闲时再发送。”碰撞检测”就是边发送边侦听，即适配器边发送数据边检测信道上信号电压的变化情况，以便判断自己在发送数据时其他站点是否也在发送数据。引入原因:当两个帧发生冲突时，两个被损坏帧继续传送毫无意义，而且信道无法被其他站点使用，对于有限的信道来讲，造成很大的浪费。如果站点边发送边监听，并在监听到冲突之后立即停止发送，可以提高信道的利用率。

> “
>
> CSMA/CD 的【工作流程】可简单概括为“先听后发，边听边发（区别于CSMA协议），冲突停发，随机重发”。
>
> 1) 适配器从其父结点获得一个网络层数据报，准备一个以太网帧，并把该帧放到适配器缓 冲区中。
>
> 2) 如果适配器侦听到信道空闲，那么它开始传输该帧。如果适配器侦听到信道忙，那么它 将等待直至侦听到没有信号能量，然后开始传输该帧。
>
> 3) 在传输过程中，适配器检测来自其他适配器的信号能量。如果这个适配器传输了整个帧，而没有检测到来自其他适配器的信号能量，那么这个适配器完成该帧的传输。否则，适配器就须停止传输它的帧，取而代之传输一个48 比特的拥塞信号。
>
> 4) 在中止（即传输拥塞信号）后，适配器采用截断二进制指数退避算法等待一段随机时间 后返回到步骤2)。
>
> ”

> “
>
> 【何为截断二进制指数退避算法】
>
> 1) 确定基本退避时间,一般取两倍的总线端到端传播时延(即争用期）。
>
> 2) 定义参数k, 它等于重传次数，但k不超过10, 即k=min【重传次数， 10】。当重传次数不超过10 时，k等于重传次数；当重传次数大于10时，k就不再增大而一直等于10(这个条件往往容易忽略，请注意）。
>
> 3) 从离散的整数集合【0, 1, …, 2^k-1】 中随机取出一个数r, 重传所需要退避的时间就是r倍的基本退避时间。

> 4) 当重传达16 次仍不能成功时，说明网络太拥挤，认为此帧永远无法正确发出，抛弃此帧并向高层报告出错（这个条件也容易忽略，请注意）。使用二进制指数退避算法可使重传需要推迟的平均时间随重传次数的增大而增大（这也称动态退避），因而能降低发生碰撞的概率，有利于整个系统的稳定。
>
> ”

总线的传播时延对 CSMA/CD 的影响很大。如图 3.20 所示，设τ 为单程传播时延。在$t=0$ 时，A 发送数据，B 检测到信道空闲。在$t=\tau-\delta$ 时，A 发送的数据还未到达 B，由于 B 检测到信道空闲而发送数据。经过时间$\delta/2$ 后，即在$t=\tau-\delta/2$ 时，A 发送的数据和 B 发送的数据发生碰撞，但这时 A 和 B 都不知道。在$t=\tau$ 时，B 检测到碰撞，于是停止发送数据。在$t=2\tau-\delta$ 时，A 检测到碰撞，也停止发送数据。显然，CSMA/CD 中的站不可能同时进行发送和接收，因此采用 CSMA/CD 协议的以太网不可能进行全双工通信，而只能进行半双工通信。

由图 3.22 可知，站 A 在发送帧后至多经过时间2τ 就能知道所发送的帧是否发生碰撞（$\delta\to0$ 时）。因此把以太网端到端往返时间 2τ 称为争用期（又称冲突窗口或碰撞窗口）。每个站在自己发送数据之后的一小段时间内，存在发生冲突的可能性，只有经过争用期这段时间还未检测到冲突时，才能确定这次发送不会发生冲突。

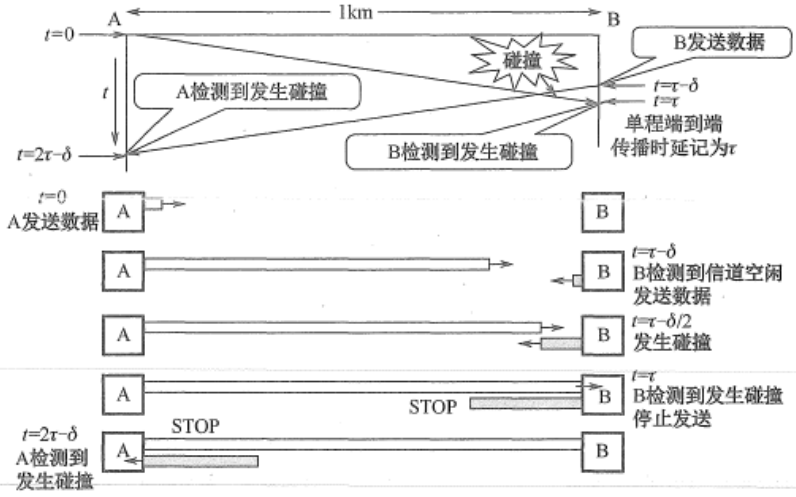


图 3.22　传播时延对载波帧听的影响

CSMA/CA协议(Collision Avoidance, 碰撞避免CA)

CSMA/CD 协议已成功应用于使用有线连接的局域网，但在无线局域网环境下，却不能简单地搬用 CSMA/CD 协议，特别是碰撞检测部分。主要有两个原因：

1. 接收信号的强度往往会远小于发送信号的强度，且在无线介质上信号强度的动态变化范围很大，因此若要实现碰撞检测，则硬件上的花费就会过大。
2. 在无线通信中，并非所有的站点都能够听见对方，即存在“隐蔽站”问题。

CSMA/CA使用预约信道、ACK 帧、RTS/CTS 帧等三种机制来实现碰撞避免：

1. 预约信道。发送方在发送数据的同时向其他站点通知自己传输数据需要的时间长度，以便让其他站点在这段时间内不发送数据，从而避免碰撞。
2. ACK 帧。所有站点在正确接收到发给自己的数据帧（除广播帧和组播帧）后，都需要向发送方发回一个ACK 帧，如果接收失败，那么不采取任何行动。发送方在发送完一个数据帧后，在规定的时间内如果未收到ACK 帧，那么认为发送失败，此时进行该数据帧的重发，直到收到ACK 帧或达到规定重发次数为止。
3. RTS/CTS 帧。可选的碰撞避免机制，主要用于解决无线网中的“隐蔽站”问题。

> “
>
> 【CSMA/CD和CSMA/CA的区别】：

> 1) CSMA/CD 可以检测冲突，但无法避免， CSMA/CA 发送包的同时不能检测到信道上有无冲突，本结点处没有冲突并不意味着在接收结点处就没有冲突，只能尽量避免。
>
> 2) 传输介质不同。CSMA/CD 用于总线形以太网， CSMA/CA 用于无线局域网802.11a/b/g/n 等。
>
> 3) 检测方式不同。CSMA/CD 通过电缆中的电压变化来检测；而CSMA/CA 采用能量检测、载波检测和能量载波混合检测三种检测信道空闲的方式。
>
> ”

> “
>
> 【总结】: CSMA/CA 协议的基本思想是在发送数据时先广播告知其他结点，让其他结点在某段时间内不要发送数据，以免出现碰撞。CSMA/CD 协议的基本思想是发送前侦听，边发送边侦听，一旦出现碰撞马上停止发送。
>
> ”

4.轮询访问介质访问控制：令牌传递协议

在轮询访问中，用户不能随机地发送信息，而要通过一个集中控制的监控站，以循环方式轮询每个结点，再决定信道的分配。当某结点使用信道时，其他结点都不能使用信道。典型的轮询访问介质访问控制协议是令牌传递协议，它主要用在令牌环局域网中。在令牌传递协议中，一个令牌在各结点间以某个固定次序交换。令牌是由一组特殊的比特组合而成的帧。当环上的一个站希望传送帧时，必须等待令牌。一旦收到令牌，站点便可启动发送帧。帧中包括目的站的地址，以标识哪个站应接收此帧。帧在环上传送时，不管该帧是否是发给本站点的，所有站点都进行转发，直到该帧回到它的始发站，并由该始发站撤销该帧。帧的目的站除转发帧外，应针对该帧维持一个副本，并通过在帧的尾部设置“响应比特”来指示已收到此副本。站点在发送完一帧后，应释放令牌，以便让其他站使用。当计算机都不需要发送数据时，令牌就在环形网上游荡，而需要发送数据的计算机只有在拿到该令牌后才能发送数据帧，因此不会发送冲突（因为令牌只有一个）。 在令牌传递网络中，传输介质的物理拓扑不必是一个环，但是为了把对介质访问的许可从一个设备传递到另一个设备，令牌在设备间的传递通路逻辑上必须是一个环。轮询介质访问控制既不共享时同，也不共享空间，它实际上是在随机介质访问控制的基础仁限定了有权力发送数据的结点只能有一个。

3.路由算法

1.路由算法综述

路由器转发分组是通过路由表转发的，而路由表是通过各种算法得到的。主机通常直接与一台路由器相连接，该路由器即为该主机的默认路由器(defaultrouter) ，又称该主机的第一跳路由器(first-hop router)每当主机发送一个分组时，该分组被传送给它的默认路由器。源主机的默认路由器称作源路由器(sourcerouter) ,目的主机的默认路由器称作目的路由器(destination router)。
一个分组从源主机到目的主机的路由选择问题显然可归结为从源路由器到目的路由器的路由选择问题。

> 【路由选择算法的分类】
>
> 1)静态路由算法（又称非自适应路由算法）
>
> 2)动态路由算法（又称自适应路由算法）;常用的动态路由算法可分为两类：距离 - 向量路由算法和链路状态路由算法。

2.静态路由算法

- 由网络管理员手工配置路由信息。当网络的拓扑结构或链路的状态发生变化时，网络管理员需要手工去修改路由表中相关的静态路由信息。大型和复杂的网络环境通常不宜采用静态路由。一方面，

网络管理员难以全面了解整个网络的拓扑结构，另一方面，当网络的拓扑结构和链路状态发生变化时，路由器中的静态路由信息需要大范围地调整，这一工作的难度和复杂程度非常高。

- 静态路由算法的优点是简便、可靠，在负荷稳定、拓扑变化不大的网络中运行效果很好，因此仍广泛用于高度安全的军事系统和较小的商业网络。
- 动态路由算法能改善网络的性能并有助于流量控制；但算法复杂，会增加网络的负担，有时因对动态变化的反应太快而引起振荡，或反应太慢而影响网络路由的一致性，因此要仔细设计动态路由算法，以发挥其优势。

3.距离-向量路由算法（RIP）

在距离-向量路由算法中，所有结点都定期地将它们的整个路由选择表传送给所有与之直接相邻的结点。这种路由选择表包含：
1) 每条路径的目的地（另一结点）。
2) 路径的代价（也称距离）。

> 【注意】这里的距离是一个抽象的概念，如RIP 就将距离定义为“跳数 " 。跳数指从源端口到达目的端口所经过的路由个数，每经过一个路由器，跳数加1 。

在这种算法中，所有结点都必须参与距离向量交换，以保证路由的有效性和一致性，也就是说，所有的结点都监听从其他结点传来的路由选择更新信息，并在下列情况下更新它们的路由选择表：

1) 被通告一条新的路由，该路由在本结点的路由表中不存在，此时本地系统加入这条新的路由。
2) 发来的路由信息中有一条到达某个目的地的路由，该路由与当前使用的路由相比，有较短的距离（较小的代价）。此种情况下，就用经过发送路由信息的结点的新路由替换路由表中到达那个目的地的现有路由。

> 【距离向量路由算法的实质】是，迭代计算一条路由中的站段数或延迟时间，从而得到到达一个目标的最短（最小代价）通路。它要求每个结点在每次更新时都将它的全部路由表发送给所有相邻的结点。显然，更新报文的大小与通信子网的结点个数成正比，大的通信子网将导致很大的更新报文。由于更新报文发给直接邻接的结点，所以所有结点都将参加路由选择信息交换。基于这些原因，在通信子网上传送的路由选择信息的数量很容易变得非常大。

> RIP(Routing Information Protocol,路由信息协议）,采用距离-向量算法，在实际使用中已经较少适用。在默认情况下，RIP使用一种非常简单的度量制度：距离就是通往目的站点所需经过的链路数，取值为0~16，数值16表示路径无限长。RIP进程使用UDP的520端口来发送和接收RIP分组。RIP分组每隔30s以广播的形式发送一次，为了防止出现“广播风暴”，其后续的分组将做随机延时后发送。在RIP中，如果一个路由在180s内未被刷新，则相应的距离就被设定成无穷大，并从路由表中删除该表项。RIP分组分为两种：请求分组和响应分组。

4.链路状态路由算法(OSPF)

链路状态路由算法要求每个参与该算法的结点都具有完全的网络拓扑信息，它们执行下述两项任务：
1) 第一，主动测试所有邻接结点的状态。两个共享一条链接的结点是相邻结点，它们连接到同一条链路，或者连接到同一广播型物理网络。
2) 第二，定期地将链路状态传播给所有其他结点。`典型的链路状态算法是OSPF算法。`

在一个链路状态路由选择中，一个结点检查所有直接链路的状态，并将所得的状态信息发送给网上的所有其他结点，而不是仅送给那些直接相连的结点。每个结点都用这种方式从网上所有其他的结点接收包含直接链路状态的路由选择信息。

每当链路状态报文到达时，路由结点便使用这些状态信息去更新自己的网络拓扑和状态“视野图”，一旦链路状态发生变化，结点就对更新的网络图利用Dijsktra最短路径算法重新计算路由，从单一的源出发计算到达所有目的结点的最短路径。

> 【注意】这是Dijsktra算法的一个实际应用，别忘了

链路状态路由算法主要有三个特征:
1) 向本自治系统中所有路由器发送信息，这里使用的方法是泛洪法，即路由器通过所有端口向所有相邻的路由器发送信息。而每个相邻路由器又将此信息发往其所有相邻路由器（但不再发送给刚刚发来信息的那个路由器）。

> 【洪泛法小知识】洪泛法（Flooding）是一种简单的路由算法，将收到的封包，往所有的可能连结路径上递送，直到封包到达为止。洪泛法被使用在桥接器上，Usenet以及点对点档案分享等。部份的路由协定也以洪泛法为基础，例如开放式最短路径优先（OSPF）、距离向量群体广播路由协定(DistanceVectorMulticastRoutingProtocol，DVMRP)。无线随意网络也使用洪泛法来进行路由。

2) 发送的信息是与路由器相邻的所有路由器的链路状态，但这只是路由器所知道的部分信息。所谓“链路状态”，是指说明本路由器与哪些路由器相邻及该链路的“度量”。对于OSPF 算法，链路状态的 " 度量” 主要用来表示费用、距离、时延、带宽等。
3) 只有当链路状态发生变化时，路由器才向所有路由器发送此消息。由于一个路由器的链路状态只涉及相邻路由器的连通状态，而与整个互联网的规模并无直接关系，因此链路状态路由算法可以用于大型的或路由信息变化聚敛的互联网环境。

链路状态路由算法的主要优点是，每个路由结点都使用同样的原始状态数据独立地计算路径，而不依赖中间结点的计算；链路状态报文不加改变地传播，因此采用该算法易于查找故障。当一个结点从所有其他结点接收到报文时，它可以在本地立即计算正确的通路，保证一步汇聚。最后，由于链路状态报文仅运载来自单个结点关于直接链路的信息，其大小与网络中的路径结点数目无关，因此链路状态算法比距离-向量算法有更好的规模可伸展性。

> 【距离-向量路由算法与链路状态路由算法的比较】：在距离-向量路由算法中，每个结点仅与它的直接邻居交谈，它为它的邻居提供从自已到网络中所有其他结点的最低费用估计。在链路状态路由算法中，每个结点通过广播的方式与所有其他结点交谈，但它仅告诉它们与它直接相连的链路的费用。相较之下，距离-向量路由算法有可能遇到路由环路等问题。

> 【路由环路问题】：在维护路由表信息的时候，如果在拓扑发生改变后，网络收敛缓慢产生了不协调或者矛盾的路由选择条目，就会发生路由环路的问题，这种条件下，路由器对无法到达的网络路由不予理睬，导致用户的数据包不停在网络上循环发送，最终造成网络资源的严重浪费。（例子和解决方案由于篇幅过大，感兴趣的请自行百度百科（狗头保命））

> OSPF(Open Shortest Path First开放式最短路径优先）是对链路状态路由协议的一种实现，著名的迪克斯加算法被用来计算最短路径树。OSPF支持负载均衡和基于服务类型的选路，也支持多种路由形式，如特定主机路由和子网路由等。OSPF的简单说就是两个相邻的路由器通过发报文的形式成为邻居关系，邻居再相互发送链路状态信息形成邻接关系，之后各自根据最短路径算法算出路由，放在OSPF路由表，OSPF路由与其他路由比较后优的加入全局路由表。

5.层次路由

当网络规模扩大时，路由器的路由表成比例地增大。这不仅会消耗越来越多的路由器缓冲区空间，而且需要用更多CPU 时间来扫描路由表，用更多的带宽来交换路由状态信息。因此路由选择必须按照层次的方式进行。

因特网将整个互联网划分为许多较小的自治系统（注意一个自治系统中包含很多局域网），每个自治系统有权自主地决定本系统内应采用何种路由选择协议。如果两个自治系统需要通信，那么就需要一种在两个自治系统之间的协议来屏蔽这些差异。据此，因特网把路由选择协议划分为两大类
1) 一个自治系统内部所使用的路由选择协议称为内部网关协议(IGP), 也称域内路由选择，具体的协议有RIP 和OSPF 等。
2) 自治系统之间所使用的路由选择协议称为外部网关协议(EGP), 也称域间路由选择，用在不同自治系统的路由器之间交换路由信息，并负责为分组在不同自治系统之间选择最优的路径。具体的协议有BGP 。

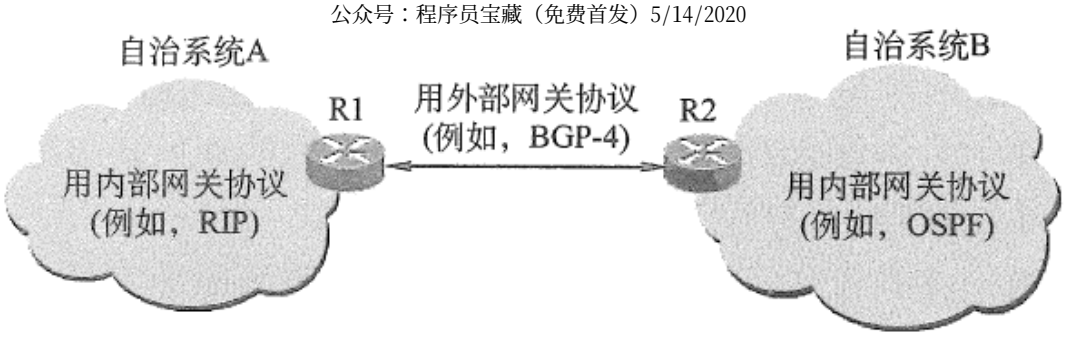


图 4.7　自治系统和内部网关协议、外部网关协议

使用层次路由时， OSPF 将一个自治系统再划分为若干区域(Area), 每个路由器都知道在本区域内如何把分组路由到目的地的细节，但不用知道其他区域的内部结构。采用分层次划分区域的方法虽然会使交换信息的种类增多，但也会使OSPF 协议更加复杂。但这样做却能使每个区域内部交换路由信息的通信量大大减小，因而使OSPF协议能够用于规模很大的自治系统中。

表 4.3　三种路由协议的比较

协　议	RIP	OSPF	BGP	
类型	内部	内部	外部	
路由算法	距离-向量	链路状态	路径-向量	
传递协议	UDP	IP	TCP	
路径选择	跳数最少	代价最低	较好，非最佳	
交换结点	和本结点相邻的路由器	网络中的所有路由器	和本结点相邻的路由器	
交换内容	当前本路由器知道的全部信息，即自己的路由表	与本路由器相邻的所有路由器的链路状态	首次	整个路由表
			非首次	有变化的部分

4.各层主要网络协议

1.各层计算机网络协议综述

协议，就是规则的集合。在网络中要做到有条不紊地交换数据，就必须遵循一些事先约定好的规则。这些规则明确规定了所交换的数据的格式及有关的同步问题。这些为进行网络中的数据交换而建立的规则、标准或约定称为网络协议(Network Protocol), 它是控制两个（或多个）对等实体进行通信的规则的集合，是水平的。不对等实体之间是没有协议的，比如用TCP/IP 协议栈通信的两个结点，结点A 的传输层和结点B 的传输层之间存在协议，但结点A 的传输层和结点B 的网络层之间不存在协议。

协议由语法、语义和同步三部分组成。语法规定了传输数据的格式；语义规定了所要完成的功能，即需要发出何种控制信息、完成何种动作及做出何种应答；同步规定了执行各种操作的条件、时序关系等，即事件实现顺序的详细说明。一个完整的协议通常应具有线路管理（建立、释放连接）、差错控制、数据转换等功能。

【参考图】：

2.物理层协议

> 【注意】本层协议不怎么问和考，就说一下奈奎斯特定理与香农定理吧！

1) 奈奎斯特定理

奈奎斯特(Nyquist)定理又称奈氏准则，它指出在理想低通（没有噪声、带宽有限）的信道中，极限码元传输率为2W 波特，其中W 是理想低通信道的带宽，单位为Hz 。若用V 表示每个码元离散电平的数目（码元的离散电平数目是指有多少种不同的码元，比如有16 种不同的码元，则需要4 位二进制位，因此数据传输率是码元传输率的4 倍），则极限数据率为理想低通信道下的极限数据传输率= 2Wlog2V (单位为b/s)

对于奈氏准则，可以得出以下结论：
1) 在任何信道中，码元传输的速率是有上限的。若传输速率超过此上限，就会出现严重的码间串扰问题（指在接收端收到的信号波形失去了码元之间的清晰界限），使得接收端不可能完全正确识别码元。
2) 信道的频带越宽（即通过的信号高频分量越多），就可用更高的速率进行码元的有效传输。
3) 奈氏准则给出了码元传输速率的限制，但并未对信息传输速率给出限制，即未对一个码元可以对应多少个二进制位给出限制。

由于码元的传输速率受奈氏准则的制约，所以要提高数据的传输速率，就必须设法使每个码元携带更多个比特的信息量，此时就需要采用多元制的调制方法。

2) 香农定理

香农(Shannon) 定理给出了带宽受限且有高斯白噪声于扰的信道的极限数据传输率，当用此速率进行传输时，可以做到不产生误差。香农定理定义为信道的极限数据传输率=Wlog2(1 + S/N) (单位为b/s)
式中， W为信道的带宽， S为信道所传输信号的平均功率， N为信道内部的高斯噪声功率。S/N为信噪比，即信号的平均功率与噪声的平均功率之比，信噪比= 10log10(S/N) (单位为dB), 例如如当S/N= 10 时，信噪比为l0dB, 而当S/N= 1000 时，信噪比为30dB 。

对于香农定理，可以得出以下结论：
1) 信道的带宽或信道中的信噪比越大，信息的极限传输速率越高。
2) 对一定的传输带宽和一定的信噪比，信息传输速率的上限是确定的。
3) 只要信息的传输速率低于信道的极限传输速率，就能找到某种方法来实现无差错的传输。
4) 香农定理得出的是极限信息传输速率，实际信道能达到的传输速率要比它低不少。

从香农定理可以看出，若信道带宽W 或信噪比S/N 没有上限（实际信道当然不可能这样），则信道的极限信息传输速率也没有上限。

> 奈氏准则只考虑了带宽与极限码元传输速率的关系，而香农定理不仅考虑到了带宽，也考虑到了信噪比。这从另一个侧面表明，一个码元对应的二进制位数是有限的。

3.数据链路层协议

> PPP 协议和HDLC 协议是目前最常用的两种广域网数据链路层控制协议。

一、PPP 协议

PPP (Point-to-Point Protocol) 是使用串行线路通信的面向字节的协议，该协议应用在直接连接两个结点的链路上。设计的目的主要是用来通过拨号或专线方式建立点对点连接发送数据，使其成为各种主机、网桥和路由器之间简单连接的一种共同的解决方案。

PPP协议是在SLIP协议的基础上发展而来的，它既可以在异步线路上传输，又可在同步线路上使用；不仅用于Modem 链路，也用于租用的路由器到路由器的线路。

> 【背景】： SLIP 主要完成数据报的传送，但没有寻址、数据检验、分组类型识别和数据压缩等功能，只能传送IP 分组。如果上层不是IP 协议，那么无法传输，并且此协议对一些高层应用也不支持，但实现比较简单。为了改进SLIP 的缺点，于是制定了点对点协议(PPP) 。

PPP 协议有三个组成部分：
1) 链路控制协议(LCP) 。一种扩展链路控制协议，用于建立、配置、测试和管理数据链路。
2) 网络控制协议(NCP) 。PPP 协议允许同时采用多种网络层协议，每个不同的网络层协议要用一个相应的NCP来配置，为网络层协议建立和配置逻辑连接。
3) 一个将IP数据报封装到串行链路的方法。IP 数据报在PPP 帧中就是其信息部分，这个信息部分的长度受最大传送单元(MTU) 的限制。

二、HDLC 协议

高级数据链路控制(High-level Data Link Control, HDLC) 协议是ISO制定的面向比特（记住PPP协议是面向字节的）的数据链路层协议。该协议不依赖于任何一种字符编码集；数据报文可透明传输，用于实现透明传输的"0 比特插入法"易于硬件实现；全双工通信，有较高的数据链路传输效率；所有帧采用CRC 检验，对信息帧进行顺序编号，可防止漏收或重发，传输可靠性高；传输控制功能与处理功能分离，具有较大的灵活性。

HDLC 适用于链路的两种基本配置：非平衡配置和平衡配置。
1) 非平衡配置的特点是由一个主站控制整个链路的工作。
2) 平衡配置的特点是链路两端的两个站都是复合站，每个复合站都可以平等地发起数据传输，而不需要得到对方复合站的允许。

> 【站】
> HDLC有3种站类型：主站、从站和复合站。主站负责控制链路的操作，主站发出的帧称为命令帧。从站受控主站，按主站的命令进行操作；发出的帧称为响应帧。另外，有些站既具有主站的功能又具有从站的功能，所以这类站称为复合站，它可以发出命令帧和响应帧。

> 【HDLC有3种数据操作方式】： 1) 正常响应方式。这是一种非平衡结构操作方式，即主站向从站传输数据，从站响应传输，但从站只有在收到主站的许可后，才可进行响应。2) 异步平衡方式。这是一种平衡结构操作方式。在这种方式中，每个复合站都可以进行对另一站的数据传输。3) 异步响应方式。这是一种非平衡结构操作方式。在这种方式中，从站即使未受到主站的允许，也可进行传输。

4.网络层协议

一、网络地址转换(NAT) 是指通过将专用网络地址（如Intranet) 转换为公用地址（如Internet),从而对外隐藏内部管理的IP 地址。它使得整个专用网只需要一个全球IP 地址就可以与因特网连通，由于专用网本地IP 地址是可重用的，所以NAT 大大节省了IP 地址的消耗。同时，它隐藏了内部网络结构，从而降低了内部网络受到攻击的风险。

使用NAT 时需要在专用网连接到因特网的路由器上安装NAT 软件， NAT 路由器至少有一个有效的外部全球地址。使用本地地址的主机和外界通信时， NAT 路由器使用NAT 转换表将本地地址转换成全球地址，或将全球地址转换成本地地址。NAT 转换表中存放着｛本地IP 地址：端口｝到｛全球IP 地址端口｝的映射。通过{ip 地址：端口｝这样的映射方式，可让多个私有IP 地址映射到同一个全球IP 地址。

> 【注意】：普通路由器在转发IP 数据报时，不改变其源IP 地址和目的IP 地址。而NAT 路由器在转发IP 数据报时，一定要更换其IP 地址（转换源IP 地址或目的IP 地址）。普通路由器仅工作在网络层，而NAT 路由器转发数据报时需要查看和转换传输层的端口号。

二、地址解析协议(ARP)

无论网络层使用什么协议，在实际网络的链路上传送数据帧时，最终必须使用硬件地址。所以需要一种方法来完成IP 地址到MAC 地址的映射，这就是地址解析协议(Address Resolution Protocol, ARP）。每台主机都设有一个ARP 高速缓存，用来存放本局域网上各主机和路由器的IP 地址到MAC 地址的映射表，称ARP 表。使用ARP 来动态维护此ARP 表。

> ARP 工作在网络层，其【工作原理】如下：主机A 欲向本局域网上的某台主机B 发送IP 数据报时，先在其ARP 高速缓存中查看有无主机B 的IP 地址。如有，就可查出其对应的硬件地址，再将此硬件地址写入MAC 帧，然后通过局域网将该MAC 帧发往此硬件地址。如果没有，那么就通过使用目的MAC 地址为FF-FF-FF-FF-FF-FF 的帧来封装并广播ARP 请求分组，使同一个局域网里的所有主机收到ARP 请求。主机B 收到该ARP 请求后，向主机A 发出响应ARP 分组，分组中包含主机B 的IP 与MAC 地址的映射关系，主机A 在收到后将此映射写入ARP 缓存，然后按查询到的硬件地址发送MAC 帧。

> 【注意】： ARP 用于解决同一个局域网上的主机或路由器的IP 地址和硬件地址的映射问题。如果所要找的主机和源主机不在同一个局域网上，那么就要通过ARP 找到一个位于本局域网上的某个路由器的硬件地址（网关），然后把分组发送给这个路由器，让这个路由器把分组转发给下一个网络。剩下的工作就由下一个网络来做，尽管ARP 请求分组是广播发送的，但ARP 响应分组是普通的
> 单播，即从一个源地址发送到一个目的地址。

三、动态主机配置协议(DHCP)

动态主机配置协议(Dynamic Host Configuration Protocol, DHCP) 常用于给主机动态地分配IP 地址，它提供了即插即用联网的机制，这种机制允许一台计算机加入新的网络和获取IP 地址
而不用手工参与。DHCP 是应用层协议，它是基于UDP的。

> 【DHCP 的工作原理】如下：使用客户 / 服务器方式。需要IP 地址的主机在启动时就向DHCP 服务器广播发送发现报文，这时该主机就成为DHCP 客户。本地网络上所有主机都能收到此广播报文，但只有DHCP 服务器才回答此广播报文。DHCP 服务器先在其数据库中查找该计算机的配置信息。若找到，则返回找到的信息。若找不到，则从服务器的IP 地址池中取一个地址分配给该计算机。DHCP 服务器的回答报文称为提供报文。

DHCP 服务器聚合DHCP 客户端的交换过程如下：
1) DHCP 客户机广播"DHCP 发现“消息，试图找到网络中的DHCP 服务器，以便从DHCP 服务器获得一个IP 地址。
2) DHCP 服务器收到"DHCP 发现“消息后，向网络中广播"DHCP 提供“消息，其中包括提供DHCP 客户机的IP 地址和相关配置信息。
3) DHCP 客户机收到"DHCP 提供“消息，如果接收DHCP 服务器所提供的相关参数，那么通过广播"DHCP 请求“消息向DHCP 服务器请求提供IP 地址。
4) DHCP 服务器广播"DHCP 确认“消息，将IP 地址分配给DHCP 客户机。
DHCP 允许网络上配置多台DHCP 服务器，当DHCP 客户机发出DHCP 请求时，有可能收到多个应答消息。这时， DHCP 客户机只会挑选其中的一个，通常挑选最先到达的。
DHCP 服务器分配给DHCP 客户的IP 地址是临时的，因此DHCP 客户只能在一段有限的时间内使用这个分配到的IP 地址。DHCP 称这段时间为租用期。租用期的数值应由DHCP 服务器自己决定， DHCP 客户也可在自己发送的报文中提出对租用期的要求。

> 【注意】DHCP 的客户端和服务器端需要通过广播方式来进行交互，原因是在DHCP 执行期间，客户
> 端和服务器端都没有标识自己身份的IP 地址，因此不可能通过单播的形式进行交互。采用UDP 而不采用TCP 的原因也很明显： TCP 需要建立连接，如果连对方的IP 地址都不知道，那么更不可能通过双方的套接字建立连接。

四、网际控制报文协议(ICMP)

为了提高IP 数据报交付成功的机会，在网络层使用了网际控制报文协议(Internet Control Message Protocol, ICMP) 来让主机或路由器报告差错和异常情况。ICMP 报文作为IP 层数据报的数据，加上数据报的首部，组成IP 数据报发送出去。ICMP 是IP 层协议。ICMP 报文的种类有两种，即ICMP 差错报告报文和ICMP 询问报文。

五种差错报告报文
1) 终点不可达。当路由器或主机不能交付数据报时，就向源点发送终点不可达报文。
2) 源点抑制。当路由器或主机由千拥塞而丢弃数据报时，就向源点发送源点抑制报文，使源点知道应当把数据报的发送速率放慢。
3) 时间超过。当路由器收到生存时间(TTL) 为零的数据报时，除丢弃该数据报外，还要向源点发送时间超过报文。当终点在预先规定的时间内不能收到一个数据报的全部数据报片时，就把已收到的数据报片都丢弃，并向源点发送时间超过报文。
4) 参数问题。当路由器或目的主机收到的数据报的首部中有的字段的值不正确时，就丢弃该数据报，并向源点发送参数问题报文。
5) 改变路由（重定向）。路由器把改变路由报文发送给主机，让主机知道下次应将数据报发送给另外的路由器（可通过更好的路由）。

五、IGMP 与组播路由算法

要使路由器知道组播组成员的信息，需要利用因特网组管理协议(Internet Group Management Protocol, IGMP) 。连接到局域网上的组播路由器还必须和因特网上的其他组播路由器协同工作，以便把组播数据报用最小代价传送给所有组成员，这就需要使用组播路由选择协议。
IGMP 并不是在因特网范围内对所有组播组成员进行管理的协议。IGMP 不知道IP 组播组包含的成员数，也不知道这些成员分布在哪些网络上。IGMP 让连接到本地局域网上的组播路由器知道本局域网上是否有主机参加或退出了某个组播组。

IGMP 应视为TCP/IP 的一部分，其工作可分为两个阶段。

第一阶段当某台主机加入新的组播组时，该主机应向组播组的组播地址发送一个IGMP 报文，声明自己要成为该组的成员。本地的组播路由器收到IGMP 报文后，将组成员关系转发给因特网上的其他组播路由器。

第二阶段因为组成员关系是动态的，本地组播路由器要周期性地探询本地局域网上的主机，以便知道这些主机是否仍继续是组的成员。只要对某个组有一台主机响应，那么组播路由器就认为这个组是活跃的。但一个组在经过几次的探询后仍然没有一台主机响应时，则不再将该组的成员关系转发给其他的组播路由器。

组播路由选择实际上就是要找出以源主机为根结点的组播转发树，其中每个分组在每条链路上只传送一次（即在组播转发树上的路由器不会收到重复的组播数据报）。不同的多播组对应于不同的多播转发树；同一个多播组，对不同的源点也会有不同的多播转发树。

5.传输层

- TCP 提供面向连接的服务，在传送数据之前必须先建立连接，数据传送结束后要释放连接。TCP 不提供广播或组播服务。由于TCP 提供面向连接的可靠传输服务，因此不可避免地增加了许多开销，如确认、流量控制、计时器及连接管理等。这不仅使协议数据单元的头部增大很多，还要占用

许多的处理机资源。因此TCP 主要适用于可靠性更重要的场合，如文件传输协议(FTP) 、超文本传输协议(HTTP) 、远程登录(TELNET) 等。

- UDP 是一个无连接的非可靠传输层协议。它在IP 之上仅提供两个附加服务：多路复用和对数据的错误检查。IP 知道怎样把分组投递给一台主机，但不知道怎样把它们投递给主机上的具体应用。UDP 在传送数据之前不需要先建立连接，远程主机的传输层收到UDP 报文后，不需要给出任何确认。由于UDP 比较简单，因此执行速度比较快、实时性好。使用UDP 的应用主要包括小文件传送协议(TFTP) 、DNS 、SNMP 和实时传输协议(RTP) 。

UDP 提供尽最大努力的交付，即不保证可靠交付，但这并不意味着应用对数据的要求是不可靠的，因此所有维护传输可靠性的工作需要用户在应用层来完成。应用实体可以根据应用的需求来灵活设计自己的可靠性机制。

UDP 是面向报文的。发送方UDP 对应用层交下来的报文，在添加首部后就向下交付给IP 层，既不合并，也不拆分，而是保留这些报文的边界；接收方UDP 对IP 层交上来UDP 用户数据报，在去除首部后就原封不动地交付给上层应用进程，一次交付一个完整的报文。因此报文不可分割，是UDP 数据报处理的最小单位。

> 【注意】： 1) IP 数据报和UDP 数据报的区别： IP 数据报在网络层要经过路由的存储转发；而UDP 数据报在传输层的端到瑞的逻辑信道中传输，封装成IP 数据报在网络层传输时， UDP数据报的信息对路由是不可见的。2) TCP 和网络层虚电路的区别: TCP 报文段在传输层抽象的逻辑信道中传输，对路由器不可见；虚电路所经过的交换结点都必须保存虚电路状态信息。在网络层若采用虚电路万式，则无法提供无连接服务，而传输层采用TCP 不影响网络层提供元连接服务。

6.应用层

一、域名系统（DNS）

域名系统(Domain Name System, DNS) 是因特网使用的命名系统，用来把便于人们记忆的具有特定含义的主机名（如www.cskaoyan.com) 转换为便于机器处理的IP 地址。相对于IP 地址，人们更喜欢使用具有特定含义的字符串来标识因特网上的计算机。值得注意的是， DNS 系统采用客户 / 服务器模型，其协议运行在UDP 之上，使用53 号端口。

从概念上可将DNS 分为3 部分：层次域名空间、域名服务器和解析器。

- 层次域名空间
  因特网采用层次树状结构的命名方法。采用这种命名方法，任何一个连接到因特网的主机或路由器，都有一个唯一的层次结构名称，即域名(Domain Name) 。域(Domain) 是名字空间中一个可被管理的划分。域还可以划分为子域，而子域还可以继续划分为子域的子域，这样就形成了顶级域、二级域、三级域等。每个域名都由标号序列组成，而各标号之间用点(". ")隔开。

图 6.3　一个域名的例子

- 域名服务器
  因特网的域名系统被设计成一个联机分布式的数据库系统，并采用客户 / 服务器模型。域名到IP 地址的解析是由运行在域名服务器上的程序完成的，一个服务器所负责管辖的（或有权限的）范围称为区（不以“域”为单位），各单位根据具体情况来划分自己管辖范围的区，但在一个区中的所有结点必须是能够连通的，每个区设置相应的权限域名服务器，用来保存该区中的所有主机的域名到IP 地址的映射。每个域名服务器不但能够进行一些域名到IP 地址的解析，而且还必须具有连向其他域名服务器的信息。当自己不能进行域名到IP 地址的转换时，能够知道到什么地方去找其他域名服务器。

DNS 使用了大量的域名服务器，它们以层次方式组织。没有一台域名服务器具有因特网上所有主机的映射，相反，该映射分布在所有的DNS 上。采用分布式设计的DNS, 是一个在因特网上实现分布式数据库的精彩范例。主要有4 种类型的域名服务器:

1) 根域名服务器

根域名服务器是最高层次的域名服务器，所有的根域名服务器都知道所有的顶级域名服务器的IP 地址。根域名服务器也是最重要的域名服务器，不管是哪个本地域名服务器，若要对因特网上任何一个域名进行解析，只要自己无法解析，就首先要求助于根域名服务器。

> 需要注意的是，根域名服务器用来管辖顶级域（如com), 通常它并不直接把待查询的域名直接转换成IP 地址，而是告诉本地域名服务器下一步应当找哪个顶级域名服务器进行查询。
>
> 2) 顶级域名服务器
>
> 这些域名服务器负责管理在该顶级域名服务器注册的所有二级域名。收到DNS 查询请求时，就给出相应的回答（可能是最后的结果，也可能是下一步应当查找的域名服务器的IP 地址）。
>
> 3) 授权域名服务器（权限域名服务器）
>
> 每台主机都必须在授权域名服务器处登记。为了更加可靠地工作，一台主机最好至少有两个授权域名服务器。实际上，许多域名服务器都同时充当本地域名服务器和授权域名服务器。授权域名服务器总能将其管辖的主机名转换为该主机的IP 地址。
>
> 4) 本地域名服务器
>
> 本地域名服务器对域名系统非常重要。每个因特网服务提供者(ISP), 或一所大学，甚至一所大学中的各个系，都可以拥有一个本地域名服务器。当一台主机发出DNS 查询请求时，这个查询请求报文就发送给该主机的本地域名服务器。事实上，我们在Windows 系统中配置“本地连接”时，就需要填写DNS 地址，这个地址就是本地DNS (域名服务器）的地址。

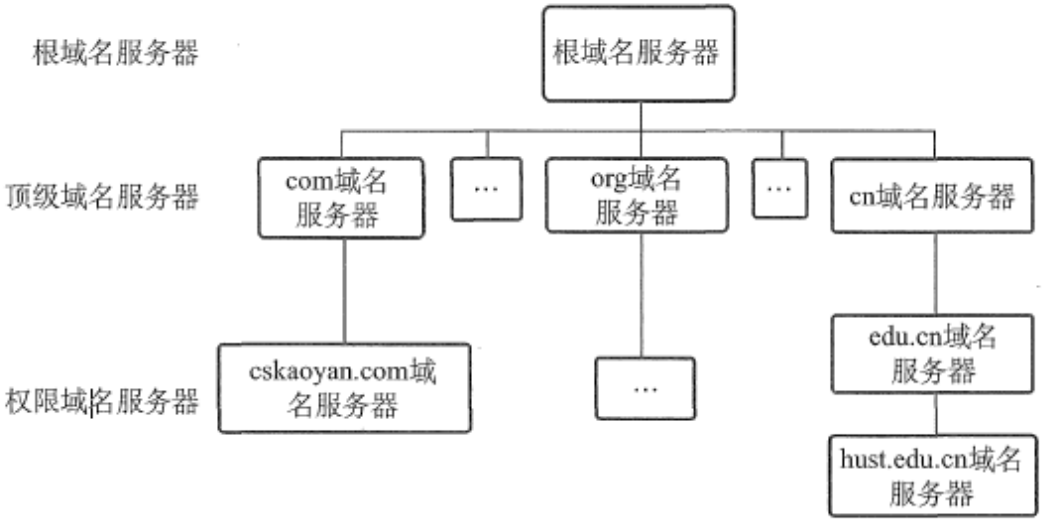


图 6.5　DNS 的层次结构

- 域名解析过程

  域名解析是指把域名映射成为IP 地址或把IP 地址映射成域名的过程。前者称为正向解析，后者称为反向解析。当客户端需要域名解析时，通过本机的DNS 客户端构造一个DNS 请求报文，以UDP 数据报方式发往本地域名服务器。域名解析有两种方式：递归查询和递归与迭代相结合的查询。

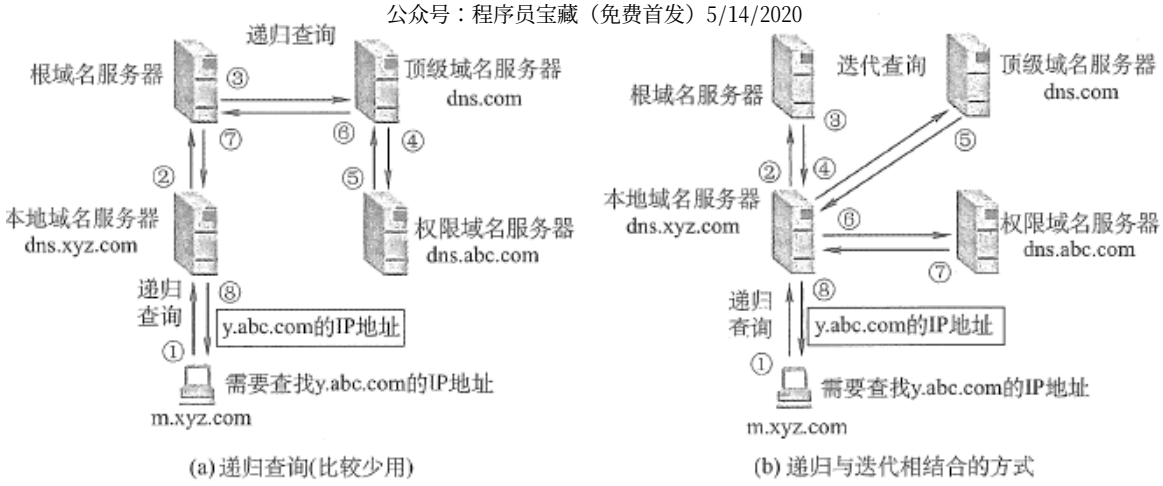


图 6.6　两种域名解析方式工作原理

> 下面举例说明域名解析的过程。假定某客户机想获知域名为y.abc.com 主机的IP 地址，域名解析的过程（共使用8 个UDP 报文）如下：
> 1)客户机向其本地域名服务器发出DNS 请求报文。
> 2)本地域名服务器收到请求后，查询本地缓存，若没有该记录，则以DNS 客户的身份向根域名服务器发出解析请求。
> 3)根域名服务器收到请求后，判断该域名属于com 域，将对应的顶级域名服务器dns.com 的IP 地址返回给本地域名服务器。
> 4)本地域名服务器向顶级域名服务器dns.com 发出解析请求报文。
> 5)顶级域名服务器dns.com 收到请求后，判断该域名属于abc.com 域，因此将对应的授权域名服务器dns.abc.com 的IP 地址返回给本地域名服务器。
> 6)本地域名服务器向授权域名服务器dns.abc.com 发起解析请求报文。
> 7)授权域名服务器dns.abc.com 收到请求后，将查询结果返回给本地域名服务器。
> 8)本地域名服务器将查询结果保存到本地缓存，同时返回给客户机。

> 【注意】：为了提高DNS 的查询效率，并减少因特网上的DNS 查询报文数量，在域名服务器中广泛地使用了高速缓存。当一个DNS 服务器接收到DNS 查询结果时，它能将该DNS 信息缓存在高速
> 缓存中。这样，当另一个相同的域名查询到达该DNS 服务器时，该服务器就能够直接提供所要求的IP 地址，而不需要再去向其他DNS 服务器询问。因为主机名和IP 地址之间的映射不是永久的，所以DNS 服务器将在一段时间后丢弃高速缓存中的信息。

二、文件传输协议（FTP）

文件传输协议（File Transfer Protocol, FTP) 是因特网上使用得最广泛的文件传输协议。FTP提供交互式的访问，允许客户指明文件的类型与格式，并允许文件具有存取权限。它屏蔽了各计算机系统的细节，因而适合于在异构网络中的任意计算机之间传送文件。

FTP 提供以下功能：

1) 提供不同种类主机系统（硬、软件体系等都可以不同）之间的文件传输能力。

2) 以用户权限管理的方式提供用户对远程FTP 服务器上的文件管理能力。

3) 以匿名FTP 的方式提供公用文件共享的能力。

FTP 采用客户 / 服务器的工作方式，它使用TCP 可靠的传输服务。一个FTP 服务器进程可同时为多个客户进程提供服务。FTP 的服务器进程由两大部分组成：一个主进程，负责接收新的请求；另外有若干从属进程，负责处理单个请求。其工作步骤如下：

1) 打开熟知端口21 (控制端口），使客户进程能够连接上。

2) 等待客户进程发连接请求。

3) 启动从属进程来处理客户进程发来的请求。主进程与从属进程并发执行，从属进程对客户进程的请求处理完毕后即终止。

4) 回到等待状态，继续接收其他客户进程的请求。

FTP 服务器必须在整个会话期间保留用户的状态信息。特别是服务器必须把指定的用户账户与控制连接联系起来，服务器必须追踪用户在远程目录树上的当前位置。

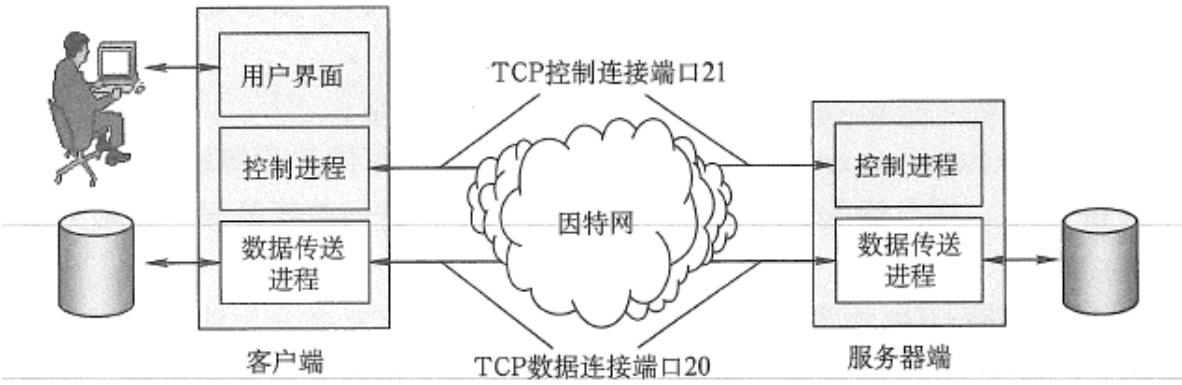


图 6.7　控制连接和数据连接

三、电子邮件系统

自从有了因特网，电子邮件就在因特网上流行起来。电子邮件是一种异步通信方式，通信时不需要双方同时在场。电子邮件把邮件发送到收件人使用的邮件服务器，并放在其中的收件人邮箱中，收件人可以随时上网到自己使用的邮件服务器进行读取。

一个电子邮件系统应具有图所示的三个最主要的组成构件，即用户代理(User Agent) 、邮件服务器和电子邮件使用的协议，如SMTP 、POP3等。

图 6.8　电子邮件系统最主要的组成构件

> 下面简单介绍电子邮件的收发过程。
> 1)发信人调用用户代理来撰写和编辑要发送的邮件。用户代理用SMTP 把邮件传送给发送方邮件服务器。
> 2)发送方邮件服务器将邮件放入邮件缓存队列中，等待发送。
> 3)运行在发送方邮件服务器的SMTP 客户进程，发现邮件缓存中有待发送的邮件，就向运行在接收方邮件服务器的SMTP 服务器进程发起建立TCP 连接。
> 4)TCP 连接建立后， SMTP 客户进程开始向远程SMTP 服务器进程发送邮件。当所有待发送邮件发完后， SMTP 就关闭所建立的TCP 连接。
> 5)运行在接收方邮件服务器中的SMTP 服务器进程收到邮件后，将邮件放入收信人的用户邮箱，等待收信人在方便时进行读取。
> 6)收信人打算收信时，调用用户代理，使用POP3 (或IMAP) 协议将自己的邮件从接收方邮件服务器的用户邮箱中取回（如果邮箱中有来信的话）。

四、超文本传输协议(HTTP)

HTTP 定义了浏览器（万维网客户进程）怎样向万维网服务器请求万维网文档，以及服务器怎样把文档传送给浏览器。从层次的角度看， HTTP 是面向事务的(Transaction-oriented) 应用层协议，它规定了在浏览器和服务器之间的请求和响应的格式与规则，是万维网上能够可靠地交换文件（包括文本、声音、图像等各种多媒体文件）的重要基础。

- HTTP 的操作过程
  从协议执行过程来说，浏览器要访问WWW 服务器时，首先要完成对WWW 服务器的域名解析。一旦获得了服务器的IP 地址，浏览器就通过TCP 向服务器发送连接建立请求。TCP 连接建立后，
  浏览器就向服务器发送请求获取某个Web 页面
  的HTTP 请求。服务器收到HTTP 请求后，将
  构建所请求Web 页的必需信息，并通过HTTP
  响应返回给浏览器。浏览器再将信息进行解释，然后将Web页显示给用户。最后， TCP 连接释放。

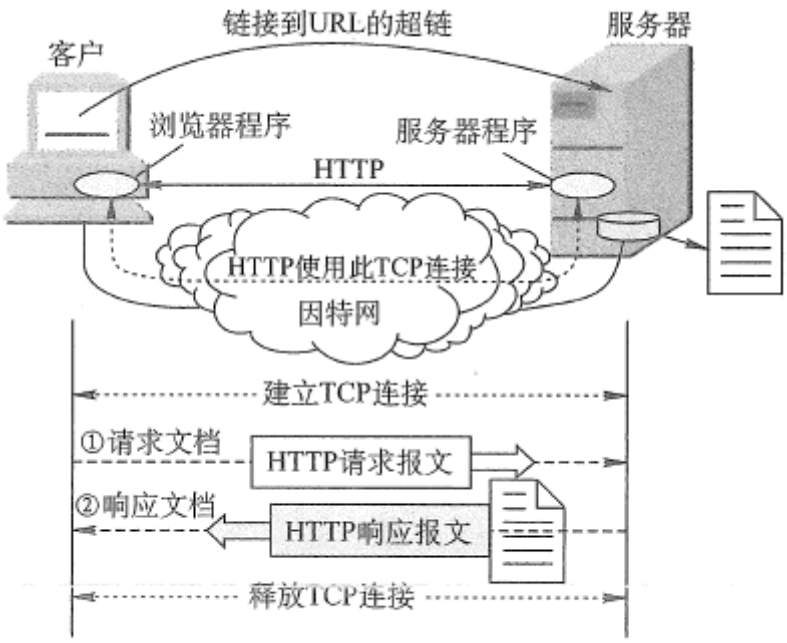


图 6.11　万维网的工作过程

> 【举个栗子】用户单击鼠标后所发生的事件按顺序如下（以访问清华大学的网站为例）：1) 浏览器分析链接指向页面的URL (http://www.tsinghua.edu.cn /chn/index.htm) 。2) 浏览器向DNS请求解析www.tsinghuaa.edu.cn 的IP 地址。3) 域名系统DNS解析出清华大学服务器的IP 地址。4) 浏览器与该服务器建立TCP 连接（默认端口号为80) 。5) 浏览器发出HTTP 请求： GET /chn/index.htm 。6) 服务器通过HTTP 响应把文件index.htm 发送给浏览器。7) TCP 连接释放。8) 浏览器解释文件index.htm, 并将Web 页显示给用户。

> HTTP 既可以使用非持久连接，也可以使用持久连接(HTTP/1.1 支持）。
> 对于非持久连接，每个网页元素对象（如JPEG 图形、Flash 等）的传输都需要单独建立一个TCP 连接，（第三次握手的报文段中捎带了客户对万维网文档的请求）。也就是说，
> 请求一个万维网文档所需的时间是该文档的传输时间（与文档大小成正比）加上两倍往返时间RTT（一个RTT 用于TCP 连接，另一个RTT 用于请求和接收文档）。
> 所谓持久连接，是指万维网服务器在发送响应后仍然保持这条连接，使同一个客户和服务器可以继续在这条连接上传送后续的HTTP 请求与响应报文。

5、各层网络设备

1.物理层设备

一、中继器

- 中继器又称转发器，主要功能是将信号整形并放大再转发出去，以消除信号经过一长段电缆后，因噪声或其他原因而造成的失真和衰减，使信号的波形和强度达到所需要的要求，进而扩大网络传输的距离。其原理是信号再生（而非简单地将衰减的信号放大）。中继器有两个端口，数据从一个端口输入，再从另一个端口发出。端口仅作用于信号的电气部分，而不管数据中是否有错误数据或不适于网段的数据。
- 中继器是局域网环境下用来扩大网络规模的最简单、最廉价的互联设备。使用中继器连接的几个网段仍然是一个局域网。一般情况下，中继器的两端连接的是相同的媒体，但有的中继器也可以完成不同媒体的转接工作。但由于中继器工作在物理层，因此它不能连接两个具有不同速率的局域网。中继器两端的网络部分是网段，而不是子网。中继器若出现故障，对相邻两个网段的工作都将产生影响。
- 从理论上讲，中继器的使用数目是无限的，网络因而也可以无限延长。但事实上这不可能，因为网络标准中对信号的延迟范围做了具体的规定，中继器只能在此规定范围内进行有效的工作．否则会引起网络故障。

> 【小知识】在采用粗同轴电缆的10BASE5 以太网规范中，互相串联的中继器的个数不能超过4个，而且用4 个中继器串联的5 段通信介质中只有3 段可以挂接计算机，其余两段只能用作扩展通信范围的链路段，不能挂接计算机。这就是所谓的"5-4-3 规则”。

> 【注意】：放大器和中继器都起放大作用，只不过放大器放大的是模拟信号，原理是将衰减的信号放大，而中继器放大的是数字信号，原理是将衰减的信号整形再生。如果某个网络设备具有存储转发的功能，那么可以认为它能连接两个不同的协议，如果该网络设备没有存储转发功能，那么认为它不能连接两个不同的协议。中继器是没有存储转发功能的，因此它不能连接两个速率不同的网段，中继器两瑞的网段一定要使用同一个协议。

二、集线器

- 集线器(Hub) 实质上是一个多端口的中继器，它也工作在物理层。当Hub工作时，一个端口接收到数据信号后，由于信号在从端口到Hub 的传输过程中已有衰减，所以Hub便将该信号进行整形放大，使之再生（恢复）到发送时的状态，紧接着转发到其他所有（除输入端口外）处于工作状态的端口。如果同时有两个或多个端口输入，那么输出时会发生冲突，致使这些数据都无效。从Hub的工作方式可以看出，它在网络中只起信号放大和转发作用，目的是扩大网络的传输范围，而不具备信号的定向传送能力，即信号传输的方向是固定的，是一个标准的共享式设备。
- Hub 主要使用双绞线组建共享网络，是从服务器连接到桌面的最经济方案。在交换式网络中，Hub 直接与交换机相连，将交换机端口的数据送到桌面上。使用Hub 组网灵活，它把所有结点的通信集中在以其为中心的结点上，对结点相连的工作站进行集中管理，不让出问题的工作站影响整个网络的正常运行，并且用户的加入和退出也很自由。
- 由Hub组成的网络是共享式网络，但逻辑上仍是一个总线网。Hub的每个端口连接的网络部分是同一个网络的不同网段。同时Hub也只能在半双工状态下工作，网络的吞吐率因而受到限制。

> 【注意】：多台计算机必然会发生同时通信的情形，因此集线器不能分割冲突域，所有集线器的端口都属于同一个冲突域。集线器在一个时钟周期中只能传输一组信息，如果一台集线器连接的机器数目较多，且多台机器经常需要同时通信，那么将导致信息碰撞，使得集线器的工作效率很差。比如，一个带宽为l0Mb/s 的集线器上连接了8 台计算机，当这8 台计算机同时工作时，每台计算机真正所拥有的带宽为10/8Mb/s = l.25Mb/s 。

2.数据链路层设备

一、网桥的概念及其基本原理

- 两个或多个以太网通过网桥连接后，就成为一个覆盖范围更大的以太网，而原来的每个以太网就称为一个网段。网桥工作在链路层的MAC子层，可以使以太网各网段成为隔离开的冲突域。如果把网桥换成工作在物理层的转发器，那么就没有这种过滤通信量的功能。由于各网段相对独立，因此一个网段的故障不会影响到另一个网段的运行。

> 【注意】：网桥处理数据的对象是帧，所以它是工作在数据链路层的设备，中继器、放大器处理数据的对象是信号，所以它是工作在物理层的设备。

- 网络I和网络2 通过网桥连接后，网桥接收网络I发送的数据帧，检查数据帧中的地址，如果是网络2的地址，那么就转发给网络2; 如果是网络I的地址，那么就将其丢弃，因为源站和目的站处在同一个网段，目的站能够直接收到这个帧而不需要借助网桥转发。
- 如图所示，设每个网段的数据率都是I0Mb/s, 那么三个网段合起来的最大吞吐量就变成了30Mb/s 。如果把两个网桥换成集线器或转发器，那么整个网络仍然是一个碰撞域（即冲突域），当A 和B 通信时，所有其他站点都不能通信，整个碰撞域的最大吞吐量仍然是I0Mb/s 。

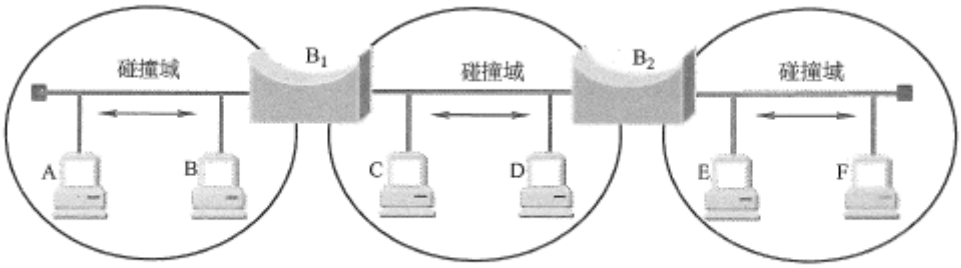


图 3.32　网桥使各网段成为隔离开的碰撞域

> 【网桥的基本特点】：
>
> 1)网桥必须具备寻址和路径选择能力，以确定帧的传输方向；
>
> 2)从源网络接收帧，以目的网络的介质访问控制协议向目的网络转发该帧；
>
> 3)网桥在不同或相同类型的LAN之间存储并转发帧，必要时还进行链路层上的协议转换。【注意】，一般情况下，存储转发类设备都能进行协议转换，即连接的两个网段可以使用不同的协议；
>
> 4)网桥对接收到的帧不做任何修改，或只对帧的封装格式做很少的修改；
>
> 5)网桥可以通过执行帧翻译互联不同类型的局域网，即把原协议的信息段的内容作为另一种协议的信息部分封装在帧中；
>
> 6)网桥应有足够大的缓冲空间，因为在短时间内帧的到达速率可能高于转发速率。

> 【网桥的优点】：1)能过滤通信量；2)扩大了物理范围；3)可使用不同的物理层；4)可互联不同类型的局域网；5)提高了可靠性和性能

> 【网桥的缺点】：
>
> 1)增大了时延（存储转发）
>
> 2)MAC子层没有流量控制功能（流量控制要用到编号机制，编号机制的实现在LLC子层）
>
> 3)不同的MAC子层的网段桥接在一起时，需要进行帧格式的转换
>
> 4)网桥只适合用户数不多和通信量不大的局域网，否则有时还会因传播过多的广播信息而产生网络拥塞，这就是所谓的网络风暴

> 【网桥的分类】：根据路径选择算法的不同，可将网桥分为透明网桥和源路由网桥。前者选择的不是最佳路由，后者选择的是最佳路由。

二、局域网交换机及其工作原理

1、局域网交换机

桥接器的主要限制是在任一时刻通常只能执行一个帧的转发操作，于是出现了局域网交换机，又称以太网交换机。从本质上说，以太网交换机是一个多端口的网桥，它工作在数据链路层。交换机能经济地将网络分成小的冲突域，为每个工作站提供更高的带宽。以太网交换机对工作站是透明的，因此管理开销低廉，简化了网络结点的增加、移动和网络变化的操作。

2、原理

以太网交换机的原理是，它检测从以太端口来的数据帧的源和目的地的MAC (介质访问层)地址，然后与系统内部的动态查找表（自学习）进行比较，若数据帧的MAC 地址不在查找表中，则将该地址加入查找表，并将数据帧发送给相应的目的端口。

3、特点

以太网交换机的特点如下：
1) 以太网交换机的每个端口都直接与单台主机相连（普通网桥的端口往往连接到以太网的一个网段），并且一般都工作在全双工方式。
2) 以太网交换机能同时连通许多对端口，使每对相互通信的主机都能像独占通信媒体那样，无碰撞地传输数据。
3) 以太网交换机也是一种即插即用设备（和透明网桥一样），其内部的帧的转发表也是通过自学习算法自动地逐渐建立起来的。
4) 以太网交换机由于使用了专用的交换结构芯片，因此交换速率较高。
5) 以太网交换机独占传输媒体的带宽。

> 对于普通l0Mb/s 的共享式以太网，若共有N 个用户，则每个用户占有的平均带宽只有总带宽(l0Mb/s) 的1/N。在使用以太网交换机时，虽然从每个端口到主机的带宽还是l0Mb/s, 但由于一个用户在通信时是独占而不是和其他网络用户共享传输媒体的带宽，因此拥有N 对端口的交换机的总容量为N*10Mb/s 。这正是交换机的最大优点。

以太网交换机一般都具有多种速率的端口，例如可以具有l0Mb/s 、l00Mb/s 和lGb/s 的端口的各种组合，因此大大方便了各种不同情况的用户。

4、两种交换模式

目前，以太网交换机主要采用两种交换模式，即直通式和存储转发式。
1) 直通式交换机只检查帧的目的地址，这使得帧在接收后几乎能马上被传出去。这种方式速度快，但缺乏智能性和安全性，也无法支持具有不同速率的端门的交换。
2) 存储转发式交换机先将接收到的帧缓存到高速缓存器中，并检查数据是否正确，确认无误后通过查找表转换成输出端口将该帧发送出去。如果发现帧有错，那么就将其丢弃。存储转发式的优点是可靠性高，并能支持不同速率端口间的转换，缺点是延迟较大。

3.网络层设备

路由器的组成和功能

路由器是一种具有多个输入 / 输出端口的专用计算机，其任务是连接不同的网络（连接异构网络）并完成路由转发。在多个逻辑网络（即多个广播域）互联时必须使用路由器。

当源主机要向目标主机发送数据报时，路由器先检查源主机与目标主机是否连接在同一个网络上。如果源主机和目标主机在同一个网络上，那么直接交付而无须通过路由器。如果源主机和目标主机不在同一个网络上，那么路由器按照转发表（路由表）指出的路由将数据报转发给下一个路由器，这称为间接交付。可见，在同一个网络中传递数据无须路由器的参与，而跨网络通信必须通过路由器进行转发。例如，路由器可以连接不同的LAN, 连接不同的VLAN, 连接不同的WAN, 或者把LAN 和WAN 互联起来。路由器隔离了广播域。

从结构上看，路由器由路由选择和分组转发两部分构成，如图所示：

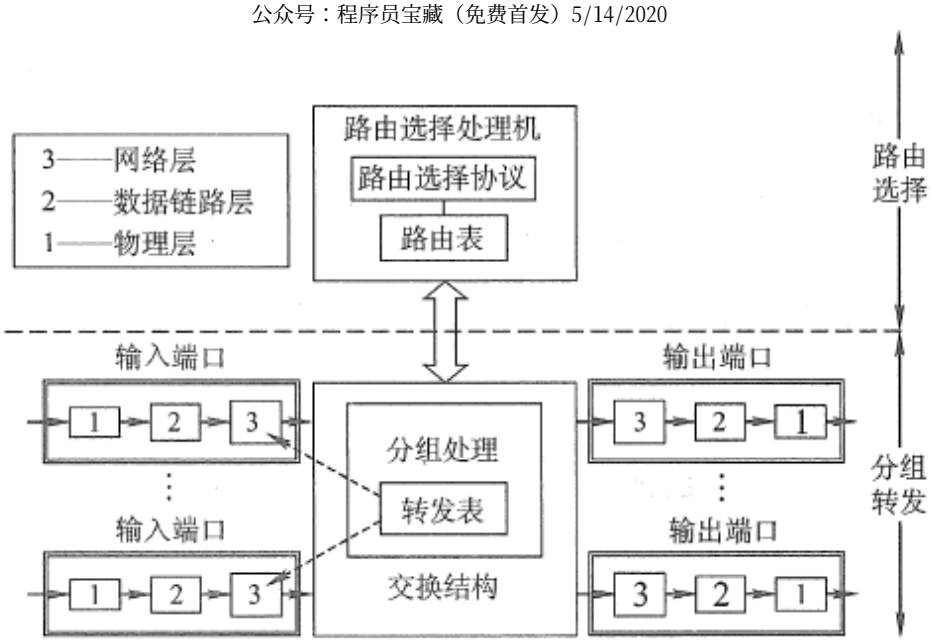


图 4.12　路由器体系结构

而从模型的角度看，路由器是网络层设备，它实现了网络模型的下三层，即物理层、数据链路层和网络层。

> 【注意】如果一个存储转发设备实现了某个层次的功能，那么它就可以互联两个在该层次上使用不同协议的网段（网络）。如果网桥实现了物理层和数据链路层，那么网桥可以互联两个物理层和数据链路层不同的网段；但中继器实现了物理层后，却不能互联两个物理层同的网段，这是因为中继器不是存储转发设备，它属于直通式设备。

路由选择部分也称控制部分，其核心构件是路由选择处理机。路由选择处理机的任务是根据所选定的路由选择协议构造出路由表，同时经常或定期地和其他相邻路由器交换路由信息而不断更新和维护路由表。

分组转发部分由三部分组成：交换结构、一组输入端口和一组输出端口。输入端口在从物理层接收到的比特流中提取出链路层帧，进而从帧中提取出网络层数据报，输出端口则执行恰好相反的操作。交换结构是路由器的关键部件，它根据转发表对分组进行处理，将某个输入端口进入的分组从一个合适的输出端口转发出去。有三种常用的交换方法：通过存储器进行交换、通过总线进行交换和通过互联网络进行交换。交换结构本身就是一个网络。

> 路由器和网桥的重要区别是：网桥与高层协议无关，而路由器是面向协议的，它依据网络地址进行操作，并进行路径选择、分段、帧格式转换、对数据报的生存时间和流量进行控制等。现今的路由器一般都提供多种协议的支持，包括OSI 、TCP/IP 、IPX 等。

路由表与路由转发

1) 路由表是根据路由选择算法得出的，主要用途是路由选择。一般标准的路由表有4 个项目：目的网络IP 地址、子网掩码、下一跳IP 地址、接口。在如图所示的网络拓扑中，RI的路表见表, 该路由表包含到互联网的默认路由。

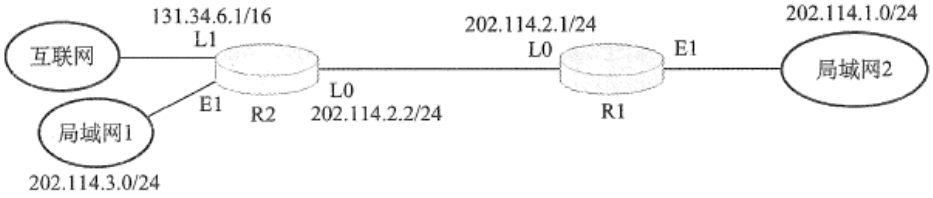


图 4.13　一个简单的网络拓扑

表 4.4　R1 的路由表

目的网络 IP 地址	子 网 掩 码	下一跳 IP 地址	接　口
202.114.1.0	255.255.255.0	Direct	E1
202.114.2.0	255.255.255.0	Direct	L0
202.114.3.0	255.255.255.0	202.114.2.2	L0
0.0.0.0	0.0.0.0	202.114.2.2	L0

2) 转发表是从路由表得出的，其表项和路由表项有直接的对应关系。但转发表的格式和路由表的格式不同，其结构应使查找过程最优化（而路由表则需对网络拓扑变化的计算最优化）。转发表中含有一个分组将要发往的目的地址，以及分组的下一跳（即下一步接收者的目的地址，实际为MAC 地址）。为了减少转发表的重复项目，可以使用一个默认路由代替所有具有相同"下一跳"的项目，并将默认路由设置得比其他项目的优先级低，如图所示。路由表总是用软件来实现的；转发表可以用软件来实现，甚至也可以用特殊的硬件来实现。

目的站	下一跳
1	直接
2	3
3	2
4	3

(a) 未使用默认路由

目的站	下一跳
1	直接
3	2
默认	3

(b) 使用了默认路由

图 4.14　未使用默认路由的转发表和使用了默认路由的转发表的对比

> 【注意】转发和路由选择的区别："转发"是路由器根据转发表把收到的IP 数据报从合适的端口转发出去，它仅涉及一个路由器。而"路由选择"则涉及很多路由器，路由表是许多路由器协同工作的结果。这些路由器按照复杂的路由算法，根据从各相邻路由器得到的关于网络拓扑的变化情况，动态的改变所选择的路由，并由此构造出整个路由表。

4.网关

大家都知道，从一个房间走到另一个房间，必然要经过一扇门。同样，从一个网络向另一个网络发送信息，也必须经过一道"关口"，这道关口就是网关。顾名思义，网关（Gateway） 就是一个网络连接到另一个网络的"关口"。也就是网络关卡。

网关(Gateway)又称网间连接器、协议转换器。默认网关在网络层上以实现网络互连，是最复杂的网络互连设备，仅用于两个高层协议不同的网络互连。网关的结构也和路由器类似，不同的是互连层。网关既可以用于广域网互连，也可以用于局域网互连。

> 【说明】：由于历史的原因，许多有关TCP/IP的文献曾经把网络层使用的路由器称为网关，在今天很多局域网采用都是路由来接入网络，因此通常指的网关就是路由器的IP！

在OSI中，网关有两种：一种是面向连接的网关，一种是无连接的网关。当两个子网之间有一定距离时，往往将一个网关分成两半，中间用一条链路连接起来，我们称之为半网关。

按照不同的分类标准，网关也有很多种。TCP/IP协议里的网关是最常用的，在这里我们所讲的“网关”均指TCP/IP协议下的网关。

那么网关到底是什么呢？网关实质上是一个网络通向其他网络的IP地址。比如有网络A和网络B，网络A的IP地址范围为“192.168.1.1~192. 168.1.254”，子网掩码为255.255.255.0；网络B的IP地址范围为“192.168.2.1~192.168.2.254”，子网掩码为255.255.255.0。在没有路由器的情况下，两个网络之间是不能进行TCP/IP通信的，即使是两个网络连接在同一台交换机（或集线器）上，TCP/IP协议也会根据子网掩码（255.255.255.0）与主机的IP 地址作 “与” 运算的结果不同判定两个网络中的主机处在不同的网络里。而要实现这两个网络之间的通信，则必须通过网关。如果网络A中的主机发现数据包的目的主机不在本地网络中，就把数据包转发给它自己的网关，再由网关转发给网络B的网关，网络B的网关再转发给网络B的某个主机。

所以说，只有设置好网关的IP地址，TCP/IP协议才能实现不同网络之间的相互通信。那么这个IP地址是哪台机器的IP地址呢？网关的IP地址是具有路由功能的设备的IP地址，具有路由功能的设备有路由器、启用了路由协议的服务器（实质上相当于一台路由器）、代理服务器（也相当于一台路由器）。

网关也被称为 IP路由器。

> 【举个栗子】：
> 假设你的名字叫小不点(很小)，你住在一个大院子里，你的邻居有很多小伙伴，父母是你的网关。当你想跟院子里的某个小伙伴玩，只要你在院子里大喊一声他的名字，他听到了就会回应你，并且跑出来跟你玩。
>
> 但是你家长不允许你走出大门，你想与外界发生的一切联系，都必须由父母（网关）用电话帮助你联系。假如你想找你的同学小明聊天，小明家住在很远的另外一个院子里，他家里也有父母（小明的网关）。但是你不知道小明家的电话号码，不过你的班主任老师有一份你们班全体同学的名单和电话号码对照表，你的老师就是你的DNS服务器。
>
> 于是你在家里和父母有了下面的对话：
>
> 小不点：妈妈(或爸爸),我想找班主任查一下小明的电话号码行吗？
>
> 家长：好，你等着。（接着你家长给你的班主任挂了一个电话，问清楚了小明的电话）问到了，他家的号码是211.99.99.99
>
> 小不点：太好了！妈(或爸),我想找小明，你再帮我联系一下小明吧。
>
> 家长：没问题。（接着家长向电话局发出了请求接通小明家电话的请求，最后一关当然是被转接到了小明家家长那里，然后他家长把电话给转到小明）.
>
> 就这样你和小明取得了联系。

如果搞清了什么是网关，默认网关也就好理解了。就好像一个房间可以有多扇门一样，一台主机可以有多个网关。默认网关的意思是一台主机如果找不到可用的网关，就把数据包发给默认指定的网关，由这个网关来处理数据包。默认网关。默认网关一般填写192.168.x.1

二、操作系统

1、死锁

1.死锁的定义

在多道程序系统中，由于多个进程的并发执行，改善了系统资源的利用率并提高了系统的处理能力。然而，多个进程的并发执行也带来了新的问题——死锁。

所谓死锁，是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。

> 【这里可能会问】：外力作用，什么是外力？答：资源剥夺，撤销进程；进程回退（参看后面的死锁解除）

> 【死锁实例】：某计算机系统中只有一台打印机和一台输入设备，进程P1正占用输入设备，同时又提出使用打印机的请求，但此时打印机正被进程P2所占用，而P2在未释放打印机之前，又提出请求使用正被P1占用的输入设备。这样，两个进程相互无休止地等待下去，均无法继续执行，此时两个进程陷入死锁状态。

2.死锁产生的原因

一、系统资源的竞争

通常系统中拥有的不可剥夺资源，其数量不足以满足多个进程运行的需要，使得进程在运行过程中，会因争夺资源而陷入僵局，如磁带机、打印机等。只有对不可剥夺资源的竞争才可能产生死锁，对可剥夺资源的竞争是不会引起死锁的。

二、进程推进顺序非法

进程在运行过程中，请求和释放资源的顺序不当，也同样会导致死锁。例如，并发进程P1, P2分别保持了资源R1,R2, 而进程P1 申请资源R2、进程P2申请资源R1 时，两者都会因为所需资源被占用而阻塞。

> 信号量使用不当也会造成死锁。进程间彼此相互等待对方发来的消息，也会使得这些进程间无法继续向前推进。例如，进程A等待进程B发的消息，进程B又在等待进程A发的消息，可以看出进程A和B不是因为竞争同一资源，而是在等待对方的资源导致死锁。

3.死锁产生的必要条件

产生死锁必须同时满足以下4个条件，只要其中任意一个条件不成立，死锁就不会发生。

1) 互斥条件：进程要求对所分配的资源（如打印机）进行排他性控制，即在一段时间内某资源仅为一个进程所占有。此时若有其他进程请求该资源，则请求进程只能等待。

2) 不剥夺条件：进程所获得的资源在未使用完之前，不能被其他进程强行夺走，即只能由获得该资源的进程自己来释放（只能是主动释放）。

3) 请求并保持条件：进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源已被其他进程占有，此时请求进程被阻塞，但对自己已获得的资源保持不放。

4) 循环等待条件：存在一种进程资源的循环等待链，链中每个进程已获得的资源同时被链中下一个进程所请求。即存在一个处于等待态的进程集合{P1, P2, .., Pn} ，其中Pi 等待的资源被Pi+1 (i=0,1, ... ,n-l) 占有，Pn等待的资源被P0占有，如图所示:

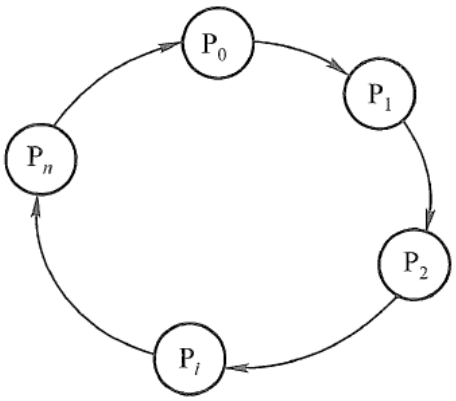


图 2.11　循环等待

> 【注意】：直观上看，循环等待条件似乎和死锁的定义一样，其实不然。按死锁定义构成等待环所要求的条件更严，它要求Pi等待的资源必须由Pi+1来满足，而循环等待条件则无此限制。例如，系统中有两台输出设备， P0占有一台，Pk占有另一台，且k不属于集合{0,1,.. ,n} 。Pn等待一台输出设备，它可
> 从P0获得，也可能从Pk获得。因此，虽然Pn,P0和其他一些进程形成了循环等待圈，但Pk不在圈内，若Pk释放了输出设备，则可打破循环等待，如图所示。因此循环等待只是死锁的必要条件。

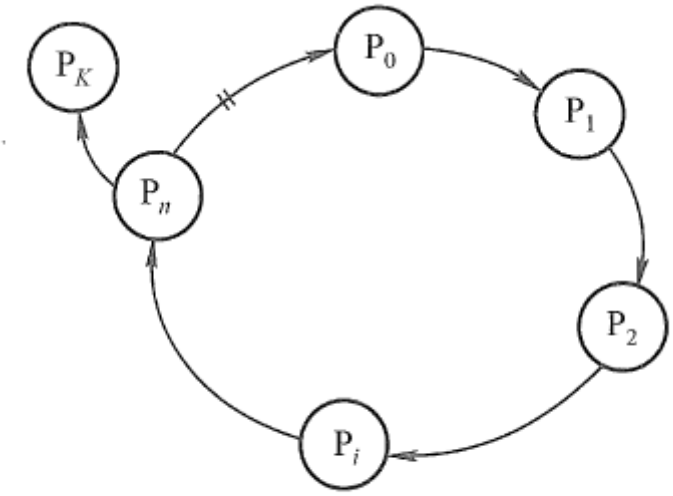


图 2.12　满足条件但无死锁

> 资源分配图含圈而系统又不一定有死锁的原因是，同类资源数大于1 。但若系统中每类资源都只有一个资源，则资源分配图含圈就变成了系统出现死锁的充分必要条件。

> 要注意区分不剥夺条件与请求并保持条件。下面用一个简单的例子进行说明：若你手上拿着一个苹果（即便你不打算吃），别人不能把你手上的苹果拿走，则这就是不剥夺条件；若你左手拿着一个苹果，允许你右手再去拿一个苹果，则这就是请求并保持条件。

4.死锁的处理策略

为使系统不发生死锁，必须设法破坏产生死锁的4个必要条件之一，或允许死锁产生，但当死锁发生时能检测出死锁，并有能力实现恢复。有以下三种处理策略：

1) 死锁预防
设置某些限制条件，破坏产生死锁的4 个必要条件中的一个或几个，以防止发生死锁。

2) 避免死锁
在资源的动态分配过程中，用某种方法防止系统进入不安全状态，从而避免死锁。

3) 死锁的检测及解除
无须采取任何限制性措施，允许进程在运行过程中发生死锁。通过系统的检测机构及时地检测出死锁的发生，然后采取某种措施解除死锁。

> 【比较】预防死锁和避免死锁都属于事先预防策略，预防死锁的限制条件比较严格，实现起来较为简单，但往往导致系统的效率低，资源利用率低；避免死锁的限制条件相对宽松，资源分配后需要通过算法来判断是否进入不安全状态，实现起来较为复杂。死锁的几种处理策略的比较见表：

表 2.4 死锁处理策略的比较

	资源分配策略	各种可能模式	主要优点	主要缺点
死锁预防	保守，宁可资源闲置	一次请求所有资源，资源剥夺，资源按序分配	适用于突发式处理的进程，不必进行剥夺	效率低，进程初始化时间延长；剥夺次数过多；不便灵活申请新资源
死锁避免	是“预防”和“检测”的折中(在运行时判断是否可能死锁)	寻找可能的安全允许顺序	不必进行剥夺	必须知道将来的资源需求；进程不能被长时间阻塞
死锁检测	宽松，只要允许就分配资源	定期检查死锁是否已经发生	不延长进程初始化时间，允许对死锁进行现场处理	通过剥夺解除死锁，造成损失

死锁预防

防止死锁的发生只需破坏死锁产生的4 个必要条件之一即可。

1) 破坏互斥条件
若允许系统资源都能共享使用，则系统不会进入死锁状态。但有些资源根本不能同时访问，如打印机等临界资源只能互斥使用。

> 所以，破坏互斥条件而预防死锁的方法不太可行，而且在有的场合应该保护这种互斥性。

2) 破坏不剥夺条件
当一个已保持了某些不可剥夺资源的进程请求新的资源而得不到满足时，它必须释放已经保持的所有资源，待以后需要时再重新申请。这意味着，一个进程已占有的资源会被暂时释放，或者说是被剥夺，或从而破坏了不剥夺条件。
该策略实现起来比较复杂，释放已获得的资源可能造成前一阶段工作的失效，反复地申请请和释放资源会增加系统开销，降低系统吞吐量。

> 这种方法常用于状态易于保存和恢复的资源，如CPU的寄存器及内存资源，一般不能用于打印机之类的资源。

3) 破坏请求并保持条件
采用预先静态分配方法，即进程在运行前一次申请完它所需要的全部资源，在它的资源未满足前，不把它投入运行。一旦投入运行，这些资源就一直归它所有，不再提出其他资源请求，这样就可以保证系统不会发生死锁。

> 这种方式实现简单，但缺点也显而易见，系统资源被严重浪费，其中有些资源可能仅在运行初期或运行快结束时才使用，甚至根本不使用。而且还会导致 " 饥饿“现象，由于个别资源长期被其他进程占用时，将致使等待该资源的进程迟迟不能开始运行。

4) 破坏循环等待条件
为了破坏循环等待条件，可采用顺序资源分配法。首先给系统中的资源编号，规定每个进程必须按编号递增的顺序请求资源，同类资源一次申请完。也就是说，只要进程提出申请分配资源Ri, 则该进程在以后的资源中请中就只能申请编号大于Ri的资源。

> 这种方法存在的问题是，编号必须相对稳定，这就限制了新类型设备的增加；尽管在为资源编号时已考虑到大多数作业实际使用这些资源的顺序，但也经常会发生作业使用资源的顺序与系统规定顺序不同的情况，造成资源的浪费；此外，这种按规定次序申请资源的方法也必然会给用户的编程带来麻烦。

死锁避免

避免死锁同样属于事先预防策略，但并不是事先采取某种限制措施破坏死锁的必要条件，而是在资源动态分配过程中，防止系统进入不安全状态，以避免发生死锁。这种方法所施加的限制条件较弱，可以获得较好的系统性能。

避免死锁的方法中，允许进程动态地申请资源，但系统在进行资源分配之前，应先计算此次分配的安全性。若此次分配不会导致系统进入不安全状态，则允许分配；否则让进程等待。

> 【安全状态】是指系统能按某种进程推进顺序(P1, P2, ..., Pn) 为每个进程Pi分配其所需的资源，直至满足每个进程对资源的最大需求，使每个进程都可顺序完成。此时称P1, P2,…, Pn为安全序列。若系统无法找到一个安全序列，则称系统处于不安全状态。

> 【注意】：并非所有的不安全状态都是死锁状态，但当系统进入不安全状态后，便可能进入死锁状态；反之，只要系统处于安全状态，系统便可避免进入死锁状态。

银行家算法

银行家算法是最著名的死锁避免算法，其【思想】是：把操作系统视为银行家，操作系统管理的资源相当于银行家管理的资金，进程向操作系统请求分配资源相当于用户向银行家贷款。操作系统按照银行家制定的规则为进程分配资源。进程运行之前先声明对各种资源的最大需求量，当进程在执行中继续申请资源时，先测试该进程已占用的资源数与本次申请的资源数之和是否超过该进程声明的最大需求量。若超过则拒绝分配资源，若未超过则再测试系统现存的资源能否满足该进程尚需的最大资源量，若能满足则按当前的申请量分配资源，否则也要推迟分配。

> 相关数据结构和实例感兴趣请自行百度（公式括号矩阵表格不太好打出来，嘿嘿嘿！）

死锁的检测和解除

前面介绍的死锁预防和避免算法，都是在为进程分配资源时施加限制条件或进行检测，若系统为进程分配资源时不采取任何措施，则应该提供死锁检测和解除的手段。

- 资源分配图
  系统死锁可利用资源分配图来描述。如下图所示，用圆圈代表一个进程，用框代表一类资源。由于一种类型的资源可能有多个，因此用框中的一个圆代表一类资源中的一个资源。从进程到资源的有向边称为请求边，表示该进程申请一个单位的该类资源；从资源到进程的边称为分配边，表示该类资源已有一个资源分配给了该进程。

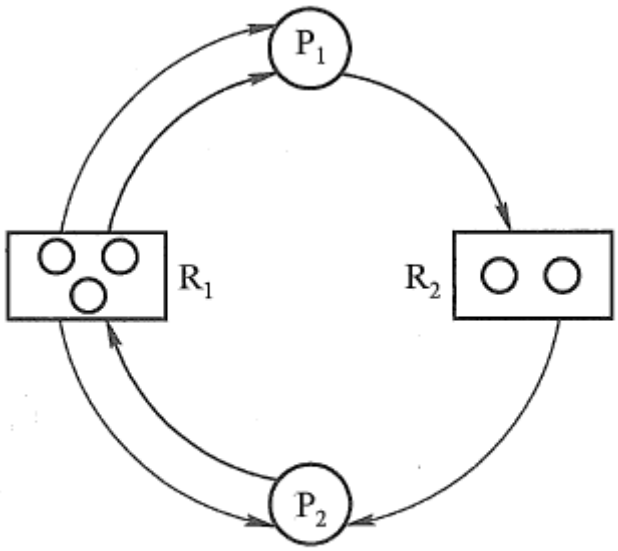


图 2.13　资源分配示例

> 【注意】判断某种资源是否有剩余空间，应用它的资源数量减去它在资源分配图中的出度。

在该图示的资源分配图中，进程P1已经分得了两个R1资源，并又请求一个R2资源；进程P2分得了一个R1资源和一个R2资源，并又请求一个R1资源。

- 死锁定理

  简化资源分配图可检测系统状态S是否为死锁状态。简化方法如下：

  1) 在资源分配图中，找出既不阻塞又不孤点的进程Pi【即找出一条有向边与它相连，且该有向边对应资源的申请数量小于等于系统中已有的空闲资源数量，如在上图中，R1没有空闲资源，R2有一个空闲资源。若所有连接该进程的边均满足上述条件，则这个进程能继续运行直至完成，然后释放它所占有的所有资源】。消去它所有的请求边和分配边，使之成为孤立的结点。在下图(a)中，P1是满足这一条件的进程结点，将P1 的所有边消去，便得到下图(b)所示的情况。继续简化便得到（c）的情况。

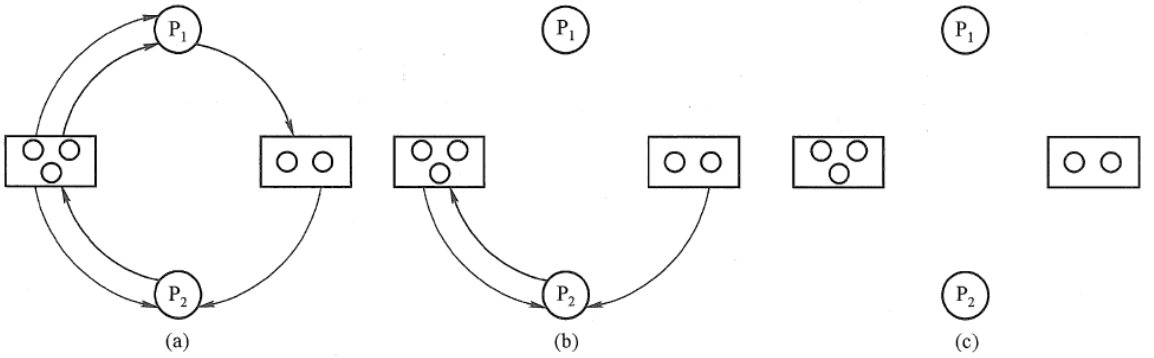


图 2.14　资源分配图的化简

> S为死锁的条件是当且仅当S状态的资源分配图是不可完全简化的，该条件为死锁定理。

死锁解除

一旦检测出死锁，就应立即采取相应的措施来解除死锁。死锁解除的主要方法有：

1) 资源剥夺法。挂起某些死锁进程，并抢占它的资源，将这些资源分配给其他的死锁进程。但应防止被挂起的进程长时间得不到资源而处于资源匮乏的状态。

2) 撤销进程法。强制撤销部分甚至全部死锁进程并剥夺这些进程的资源。撤销的原则可以按进程优先级和撤销进程代价的高低进行。

3) 进程回退法。让一（或多）个进程回退到足以回避死锁的地步，进程回退时自愿释放资源而非被剥夺。要求系统保持进程的历史信息，设置还原点。

> 【拓展】：死锁与饥饿的区别?
> 一组进程处于死锁状态是指组内的每个进程都在等待一个事件，而该事件只可能由组内的另一个进程产生。这里所关心的主要是事件是资源的获取和释放。
> 与死锁相关的另一个问题是无限期阻塞(Indefinite Blocking) 或饥饿(Starvation) , 即进程在信号最内无穷等待的情况。
> 产生饥饿的【主要原因】是：在一个动态系统中，对于每类系统资源，操作系统需要确定一个分配策略，当多个进程同时申请某类资源时，由分配策略确定资源分配给进程的次序。有时资源分配策略可能是不公平的，即不能保证等待时间上界的存在。在这种情况下，即使系统没有发生死锁，某些进程也可能会长时间等待。当等待时间给进程推进和响应带来明显影响时，称发生了进程"饥饿 " ，当"饥饿"到一定程度的进程所赋予的任务即使完成也不再具有实际意义时，称该进程被"饿死"。
> "饥饿"并不表示系统一定会死锁，但至少有一个进程的执行被无限期推迟。 " 饥饿"与死锁的主要差别如下：
> 1) 进入"饥饿"状态的进程可以只有一个，而因循环等待条件而进入死锁状态的进程却必须大于等于两个。
> 2) 处于"饥饿 ' ' 状态的进程可以是一个就绪进程，如静态优先权调度算法时的低优先权进程，而处于死锁状态的进程则必定是阻塞进程。

2、程序|进程|线程比较

1.程序、进程、线程简述

程序的基本概念

程序是含有指令和数据的文件，被存储在磁盘或其他的数据存储设备中，也就是说程序是静态的代码。

进程的基本概念

在多道程序环境下，允许多个程序并发执行，此时它们将失去封闭性，并具有间断性及不可再现性的特征。为此引入了进程(Process) 的概念，以便更好地描述和控制程序的并发执行，实现操作系统的并发性和共享性（最基本的两个特性）。

为了使参与并发执行的程序（含数据）能独立地运行，必须为之配置一个专门的数据结构，称为进程控制块(Process Control Block, PCB) 。系统利用PCB 来描述进程的基本情况和运行状态，进而控制和管理进程。

相应地，由程序段、相关数据段和PCB三部分构成了进程映像（进程实体）。所谓创建进程，实质上是创建进程映像中的PCB; 而撤销进程，实质上是撤销进程的PCB 。值得注意的是，进程映像是静态的，进程则是动态的。

从不同的角度，进程可以有不同的定义，比较典型的定义有：
1) 进程是程序的一次执行过程。
2) 进程是一个程序及其数据在处理机上顺序执行时所发生的活动。
3) 进程是具有独立功能的程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。

引入进程实体的概念后，我们可以把传统操作系统中的进程定义为：“进程是进程实体的运行过程是系统进行资源分配和调度的一个独立单位。（还没引入线程）”

> 【注意】:要准确理解这里说的系统资源。它指处理机、存储器和其他设备服务于某个进程的”时间”，例如把处理机资源理解为处理机的时间片才是准确的。因为进程是这些资源分配和调度的独立单位（还没引入线程），即“时间片”分配的独立单位，这就决定了进程一定是一个动态的、过程性的概念。

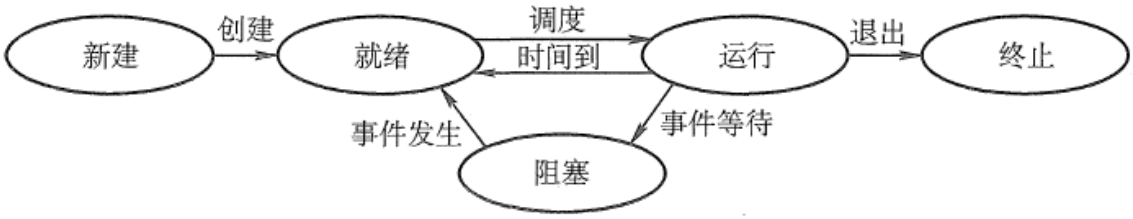


图 2.1　5 种进程状态的转换

线程的基本概念

引入进程的目的是为了更好地使多道程序并发执行，提高资源利用率和系统吞吐量；而引入线程的目的则是为了减小程序在并发执行时所付出的时空开销，提高操作系统的并发性能。

线程最直接的理解就是“轻量级进程”，它是一个基本的CPU执行单元，也是程序执行流的最小单元，由线程ID 、程序计数器、寄存器集合和堆栈组成。线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点儿在运行中必不可少的资源，但它可与同属一个进程的其他线程共享进程所拥有的全部资源。一个线程可以创建和撤销另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有就绪、阻塞和运行三种基本状态。

2.程序和进程

> 【拓展问题1】：为什么要引入进程？
> 答：在多道程序同时运行的背景下，进程之间需要共享系统资源，因此会导致各程序在执行过程中出现相互制约的关系，程序的执行会表现出间断性的特征。这些特征都是在程序的执行过程中发生的，是动态的过程，而传统的程序本身是一组指令的集合，是一个静态的概念，无法描述程序在内存中的执行情况，即我们无法从程序的字面上看出它何时执行、何时停顿，也无法看出它与其他执行程序的关系，因此，程序这个静态概念已不能如实反映程序并发执行过程的特征。为了深刻描述程序动态执行过程的性质乃至更好地支持和管理多道程序的并发执行，人们引入了进程的概念。

> 【拓展问题2】：进程是如何解决问题的？
> 答：进程把能够识别程序运行态的一些变量存放在PCB中，通过这些变量系统能够更好地了解进程的状况，并在适当时进行进程的切换，以避免一些资源的浪费，甚至划分为更小的调度单位线程来提高系统的并发度。

程序和进程的比较

1) 程序是永存的(除非删除)；进程是暂时的，是程序在数据集上的一次执行，有创建有撤销，存在是暂时的；
2) 程序是静态的观念，进程是动态的观念；
3) 进程具有并发性，而程序没有；
4) 进程是竞争计算机资源的基本单位，程序不是。
5) 进程和程序不是一一对应的： 一个程序可对应多个进程即多个进程可执行同一程序； 一个进程可以执行一个或几个程序。

3.进程和线程

> 【拓展问题3】：为什么要引入线程？
> 答：进程存在着很多缺陷，主要集中在两点：
> (1)进程只能在同一时间干一件事情，如果想同时干两件事或多件事情，进程就无能为力了。
> (2)进程在执行的过程中如果由于某种原因阻塞了，例如等待输入，整个进程就会挂起，其他与输入无关的工作也必须等待输入结束后才能顺序执行。
> 为了解决上述两点缺陷，引入了线程这个概念。

线程与进程的比较

1) 调度。在传统的操作系统中，拥有资源和独立调度的基本单位都是进程。在引入线程的操作系统中，线程是独立调度的基本单位，进程是拥有资源的基本单位。在同一进程中，线程的切换不会引起进程切换。在不同进程中进行线程切换，如从一个进程内的线程切换到另一个进程中的线程时，会引起进程切换。
2) 拥有资源。不论是传统操作系统还是设有线程的操作系统，进程都是拥有资源的基本单位，而线程不拥有系统资源（也有一点儿必不可少的资源），但线程可以访问其隶属进程的系统资源。要知道，若线程也是拥有资源的单位，则切换线程就需要较大的时空开销，线程这个概念的提出就没有意义。
3) 并发性。在引入线程的操作系统中，不仅进程之间可以并发执行，而且多个线程之间也可以并发执行，从而使操作系统具有更好的并发性，提高了系统的吞吐量。
4) 系统开销。由于创建或撤销进程时，系统都要为之分配或回收资源，如内存空间、I/0 设备等，因此操作系统所付出的开销远大于创建或撤销线程时的开销。类似地，在进行进程切换时，涉及当前执行进程CPU 环境的保存及新调度到进程CPU 环境的设置，而线程切换时只需保存和设置少量寄存器内容，开销很小。此外，由于同一进程内的多个线程共享进程的地址空间，因此这些线程之间的同步与通信非常容易实现，甚至无须操作系统的干预。
5) 地址空间和其他资源（如打开的文件）。进程的地址空间之间互相独立，同一进程的各线程间共享进程的资源，某进程内的线程对于其他进程不可见。
6) 通信方面。进程间通信(IPC) 需要进程同步和互斥手段的辅助，以保证数据的一致性，而线程间可以直接读 / 写进程数据段（如全局变量）来进行通信。

> 【为什么线程的提出有利于提高系统并发性？】可以这样来理解：由于有了线程，线程切换时，有可能会发生进程切换，也有可能不发生进程切换，平均而言每次切换所需的开销就变小了，因此能够让更多的线程参与并发，而不会影响到响应时间等问题。

3、虚拟内存管理

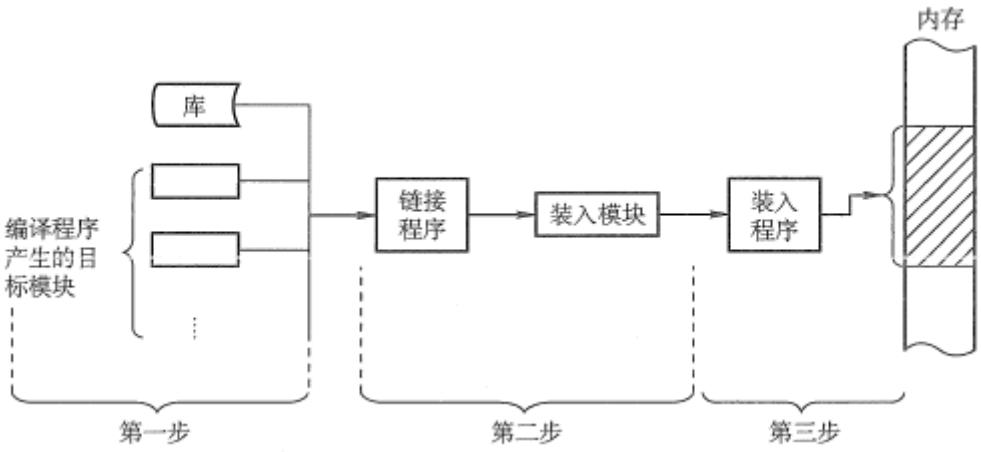
1.前导知识简述

> 【问】：为什么要进行内存管理？
> 答：在单道批处理系统阶段，一个系统在一个时间段内只执行一个程序，内存的分配极其简单，即仅分配给当前运行的进程。引入多道程序的并发执行后，进程之间共享的不仅仅是处理机，还有主存储器。然而，共享主存会形成一些特殊的挑战（越界、缺页等）。若不对内存进行管理，则容易导致内存数据的混乱，以至于限制进程的并发执行。因此，为了更好地支持多道程序并发执行，必须进行内存管理。

链接和装入

创建进程首先要将程序和数据装入内存。将用户源程序变为可在内存中执行的程序，通常需要以下几个步骤：

- 编译。由编译程序将用户源代码编译成若干目标模块。
- 链接。由链接程序将编译后形成的一组目标模块及所需的库函数链接在一起，形成一个完整的装入模块。
- 装入。由装入程序将装入模块装入内存运行。



程序的链接有以下三种方式：
1) 静态链接：程序运行前，先将各目标模块及他们所需要的库函数链接成一个完整的可执行程序，以后不再拆开。
2) 装入时动态链接：在装入时，边装入边链接。
3) 运行时动态链接：对某些目标模块的链接，是在程序执行中需要该目标模块时才进行的。其优点是便于修改和更新，便于实现对目标模块的共享。

程序的装入有以下三种方式：
1) 绝对装入:单道程序环境下，逻辑地址和物理地址完全相同，不需要地址变换。
2) 静态重定位：装入时一次性将逻辑地址转换为物理地址（重定位）
3) 动态重定位：运行时才把逻辑地址转换为物理地址，适合程序在内存中会发生移动的情况。

> 【问】：覆盖与交换？
> 答：覆盖与交换技术是用来扩充内存的两种方法。
> 覆盖的【基本思想】如下：由于程序运行时并非任何时候都要访问程序及数据的各个部分（尤其是大程序），因此可把用户空间分成一个固定区和若干覆盖区。将经常活跃的部分放在固定区，其余部分按调用关系分段。首先将那些即将要访问的段放入覆盖区，其他段放在外存中，在需要

> 调用前，系统再将其调入覆盖区，替换覆盖区中原有的段。覆盖技术的特点是，打破了必须将一个进程的全部信息装入主存后才能运行的限制，但当同时运行程序的代码量大于主存时仍不能运行，此外，内存中能够更新的地方只有覆盖区的段，不在覆盖区中的段会常驻内存。
> 交换（对换）的【基本思想】是，把处于等待状态（或在CPU 调度原则下被剥夺运行权利）的程序从内存移到辅存，把内存空间腾出来，这一过程又称换出；把准备好竞争CPU 运行的程序从辅存移到内存，这一过程又称换入。中级调度采用的就是交换技术。
> 【区别】交换技术主要在不同进程（或作业）之间进行，而覆盖则用于同一个程序或进程中。由于覆盖技术要求给出程序段之间的覆盖结构，使得其对用户和程序员不透明，所以对于主存无法存放用户程序的矛盾，现代操作系统是通过虚拟内存技术来解决的，覆盖技术则已成为历史；而交换技术在现代操作系统中仍具有较强的生命力。

连续分配管理方式

1. 单一连续分配
2. 固定分区分配
3. 动态分区分配

表 3.1　三种内存分区管理方式的比较

	作业道数	内部碎片	外部碎片	硬件支持	可用空间管理	解决碎片方法	解决空间不足	提高作业道数
单道连续分配	1	有	无	界地址寄存器、越界检查机构	—	—	覆盖	交换
多道固定连续分配	≤N(用户空间划为 N 块)	有	无	① 上下界寄存器、越界检查机构	—	—		
多道可变连续分配	—	无	有	② 基地址寄存器、长度寄存器、动态地址转换机构	① 数组 ② 链表	紧凑		

非连续分配方式

固定分区会产生内部碎片，动态分区会产生外部碎片，这两种技术对内存的利用率都比较低。我们希望内存的使用能尽量避免碎片的产生，这就引入了分页的思想：把主存空间划分为大小相等且固定的块，块相对较小，作为主存的基本单位。每个进程也以块为单位进行划分，进程在执行时，以块为单位逐个申请主存中的块空间。

1. 基本分页存储管理方式
2. 基本分段存储管理方式
3. 基本段页式管理方式

2.页式虚拟内存

基于局部性原理，在程序装入时，将程序的一部分装入内存，而将其余部分留在外存，就可启动程序执行。在程序执行过程中，当所访问的信息不在内存时，由操作系统将所需要的部分调入内存，然后继续执行程序。另一方面，操作系统将内存中暂时不使用的内容换出到外存上，从而腾出空间存放将要调入内存的信息。这样，系统好像为用户提供了一个比实际内存大得多的存储器，称为虚拟存储器。

> 【局部性原理】：快表、页高速缓存及虚拟内存技术从广义上讲，都属于高速缓存技术。这个技术所依赖的原理就是局部性原理。
> 局部性原理表现在以下两个方面：
>
> 1) 时间局部性。程序中的某条指令一旦执行，不久后该指令可能再次执行；某数据被访问过，不久后该数据可能再次被访问。产生时间局部性的典型原因是程序中存在着大量的循环操作。

> 2) 空间局部性。一旦程序访问了某个存储单元，在不久后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址，可能集中在一定的范围之内，因为指令通常是顺序存放、顺序执行的，数据也一般是以向量、数组、表等形式簇聚存储的。
>
> 时间局部性通过将近来使用的指令和数据保存到高速缓冲存储器中，并使用高速缓存的层次结构实现。空间局部性通常使用较大的高速缓存，并将预取机制集成到高速缓存控制逻辑中实现。虚拟内存技术实际上建立了“内存 - 外存”的两级存储器结构，利用局部性原理实现高速缓存。

请求分页系统建立在基本分页系统基础之上，为了支持虚拟存储器功能而增加了请求调页功能和页面置换功能。请求分页是目前最常用的一种实现虚拟存储器的方法。

为了实现请求分页，系统必须提供一定的硬件支持。除了需要一定容量的内存及外存的计算机系统，还需要有页表机制、缺页中断机构和地址变换机构。

页表机制

请求分页系统的页表机制不同于基本分页系统，请求分页系统在一个作业运行之前不要求全部一次性调入内存，因此在作业的运行过程中，必然会出现要访问的页面不在内存中的情况，如何发现和处理这种情况是请求分页系统必须解决的两个基本问题。为此，在请求页表项中增加了4 个字段，如图所示：

页号	物理块号	状态位 P	访问字段 A	修改位 M	外存地址

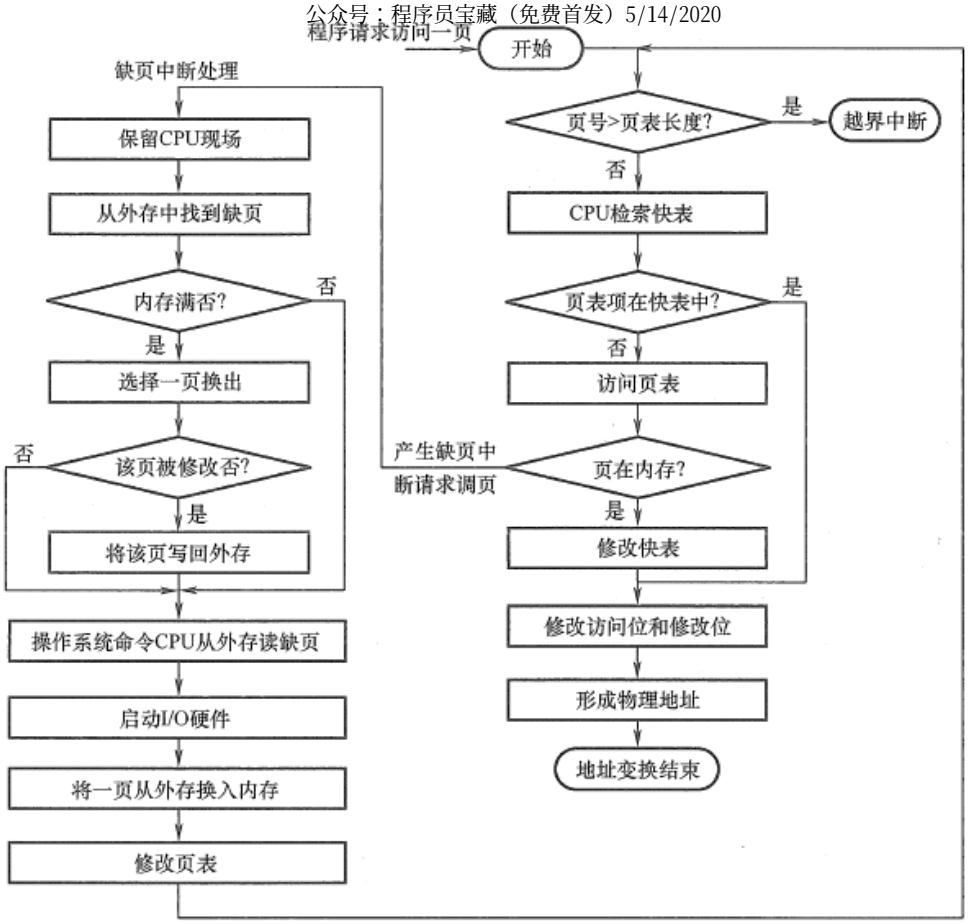
缺页中断机构

在请求分页系统中，匈当所要访问的页面不在内存中时，便产生一个缺页中断，请求操作系统将所缺的页调入内存。此时应将缺页的进程阻塞（调页完成唤醒），若内存中有空闲块，则分配一个块，将要调入的页装入该块，并修改页表中的相应页表项，若此时内存中没有空闲块，则要淘汰某页（若被淘汰页在内存期间被修改过，则要将其写回外存）。

> 【注意】：缺页中断作为中断，同样要经历诸如保护CPU 环境、分析中断原因、转入缺页中断处理程序、恢复CPU 环境等几个步骤。但与一般的中断相比，它有以下两个明显的【区别】：
> 1）在指令执行期间而非一条指令执行完后产生和处理中断信号，属于内部中断。
> 2）一条指令在执行期间，可能产生多次缺页中断。

地址变换机构

一张流程图一目了然：



页面置换算法

1. 最佳(OPT) 置换算法

最佳(Optimal, 0PT) 置换算法选择的被淘汰页面是以后永不使用的页面，或是在最长时间内不再被访问的页面，以便保证获得最低的缺页率。然而，由于人们目前无法预知进程在内存下的若干页面中哪个是未来最长时间内不再被访问的，因而该算法无法实现。

2. 先进先出(FIFO) 页面置换算法

优先淘汰最早进入内存的页面，即在内存中驻留时间最久的页面。该算法实现简单，只需把调入内存的页面根据先后次序链接成队列，设置一个指针总指向最早的页面。但该算法与进程实际运行时的规律不适应，因为在进程中，有的页面经常被访问。

> FIFO 算法还会产生所分配的物理块数增大而页故障数不减反增的异常现象，称为Belady 异常。
> 只有FIFO 算法可能出现Belady 异常， LRU 和OPT 算法永远不会出现Belady 异常。

3. 最近最久未使用（LRU）置换算法

选择最近最长时间未访问过的页面予以淘汰，它认为过去一段时间内未访问过的页面，在最近的将来可能也不会被访问。该算法为每个页面设置一个访问字段，来记录页面自上次被访问以来所经历的时间，淘汰页面时选择现有页面中值最大的予以淘汰。

> LRU 算法的性能较好，但需要寄存器和栈的硬件支持。LRU 是堆栈类的算法。理论上可以证明，堆栈类算法不可能出现Belady 异常。FIFO 算法基于队列实现，不是堆栈类算法。

4. 时钟(CLOCK) 置换算法

简单的CLOCK 算法给每帧关联一个附加位，称为使用位。当某页首次装入主存时，将该帧的使用位设置为1; 当该页随后再被访问到时，其使用位也被置为1 。对于页替换算法，用于替换的候选帧集合可视为一个循环缓冲区，并有一个指针与之相关联。当某一页被替换时，该指针被设置成指向缓冲区中的下一帧。当需要替换一页时，操作系统扫描缓冲区，以查找使用位被置为0 的一帧。每当遇到

一个使用位为1 的帧时，操作系统就将该位重新置为0, 若在这个过程开始时，缓冲区中所有帧的使用位均为0, 则选择遇到的第一个帧替换；若所有帧的使用位均为1,则指针在缓冲区中完整地循环一周，把所有使用位都置为0, 并停留在最初的位置上，替换该帧中的页。由于该算法循环检查各页面的情况，因此称CLOCK算法，又称最近未用(Not Recently
Used, NRU) 算法。

在使用位的基础上再增加一个修改位，则得到改进型CLOCK 置换算法。这样，每帧都处于以下4种情况之一：
1）最近未被访问，也未被修改(u= 0, m = 0) 。
2）最近被访问，但未被修改(u=l,m=0) 。
3）最近未被访问，但被修改(u=0,m= 1) 。
4）最近被访问，被修改(u=l,m=l)

算法执行步骤如下：

1) 从指针的当前位置开始，扫描帧缓冲区。在这次扫描过程中，对使用位不做任何修改。选择遇到的第一个帧(u = 0, m = 0) 用于替换。
2) 若第1) 步失败，则重新扫描，查找(u=0,m= 1) 的帧。选择遇到的第一个这样的帧用于替换。在这个扫描过程中，对每个跳过的帧，把它的使用位设置成0 。
3) 若第2) 步失败，则指针将回到它的最初位置，且集合中所有帧的使用位均为0 。重复第
1) 步，并且若有必要，重复第2) 步，以便可以找到供替换的帧。

> 【注意】：改进型CLOCK 算法优于简单CLOCK 算法的地方在于替换时首选没有变化的页。由于修改过的页在被替换之前必须写回，因而这样做会节省时间。

> 【抖动】：在页面置换过程中，一种最糟糕的情形是，刚刚换出的页面马上又要换入主存，刚刚换入的页面马上又要换出主存，这种频繁的页面调度行为称为抖动或颠簸。若一个进程在换页上用的时间多于执行时间，则这个进程就在颠簸。频繁发生缺页中断（抖动）的主要原因是，某个进程频繁访问的页面数目高于可用的物理页帧数目。

> 虚拟内存是怎么解决问题的？会带来什么问题？
> 答：虚拟内存使用外存上的空间来扩充内存空间，通过一定的换入 / 换出，使得整个系统在逻辑上能够使用一个远远超出其物理内存大小的内存容量。因为虚拟内存技术调换页面时需要访问外存，会导致平均访存时间增加，若使用了不合适的替换算法，则会大大降低系统性能。

> 【问】：多级页表解决了什么问题？又会带来什么问题？
> 答：多级页表解决了当逻辑地址空间过大时，页表的长度会大大增加的问题。多级页表是为了节省空间。（单级页表为了随机访问必须连续存储，如果虚拟内存空间很大，就需要很多页表项，就需要很大的连续内存空间【你能想象页表比分配的页（“运行页”）还多吗？】；但是多级页表不需要，最开始只需一页。）
> 【问题】：而采用多级页表时，一次访盘需要多次访问内存甚至磁盘，会大大增加一次访存的时间。

3.段式虚拟内存

段式虚拟存储器中的段是按程序的逻辑结构划分的，各个段的长度因程序而异。把虚拟地址分为两部分：段号和段内地址。虚拟地址到实地址之间的变换是由段表来实现的。段表是程序的逻辑段和在主存中存放位置的对照表。段表的每行记录与某个段对应的段号、装入位、段起点和段长等信息。由于段的长度可变，所以段表中要给出各段的起始地址与段的长度。

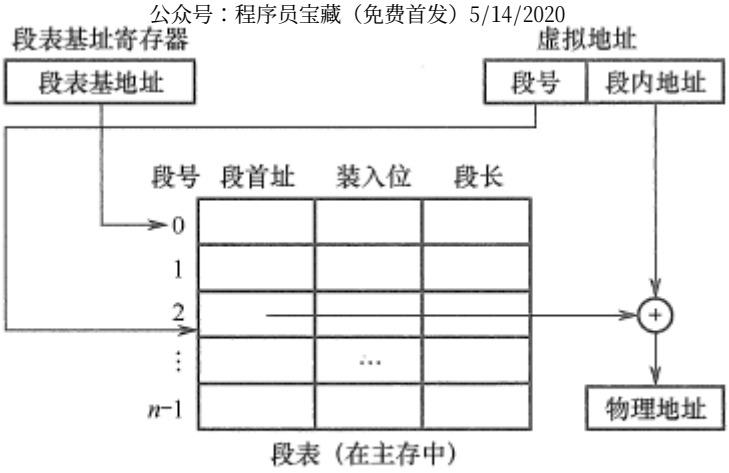


图 3.28　段式虚拟存储器的地址变换过程

段式虚拟存储器的优点是，段的分界与程序的自然分界相对应，因而具有逻辑独立性，使得它易于编译、管理、修改和保护，也便于多道程序的共享；缺点是因为段长度可变，分配空间不便，容易在段间留下碎片，不好利用，造成浪费。

表 3.7　分页管理方式和分段管理方式的比较

	分　页	分　段
目的	页是信息的物理单位，分页是为实现离散分配方式，以削减内存的外零头，提高内存的利用率。或者说，分页仅是由于系统管理的需要而不是用户的需要	段是信息的逻辑单位，它含有一组意义相对完整的信息。分段的目的是为了能更好地满足用户的需要
长度	页的大小固定且由系统决定，由系统把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而在系统中只能有一种大小的页面	段的长度不固定，决定于用户所编写的程序，通常由编译程序在对程序进行编译时，根据信息的性质来划分
地址空间	作业地址空间是一维的，即单一的线性地址空间，程序员利用一个记忆符即可表示一个地址	作业地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址
碎片	有内部碎片，无外部碎片	有外部碎片，无内部碎片
“共享”和“动态链接”	不容易实现	容易实现

4.段页式虚拟内存

把程序按逻辑结构分段，每段再划分为固定大小的页，主存空间也划分为大小相等的页，程序对主存的调入、调出仍以页为基本传送单位，这样的虚拟存储器称为段页式虚拟存储器。在段页式虚拟存储器中，每个程序对应一个段表，每段对应一个页表，段的长度必须是页长的整数倍，段的起点必须是某一页的起点。

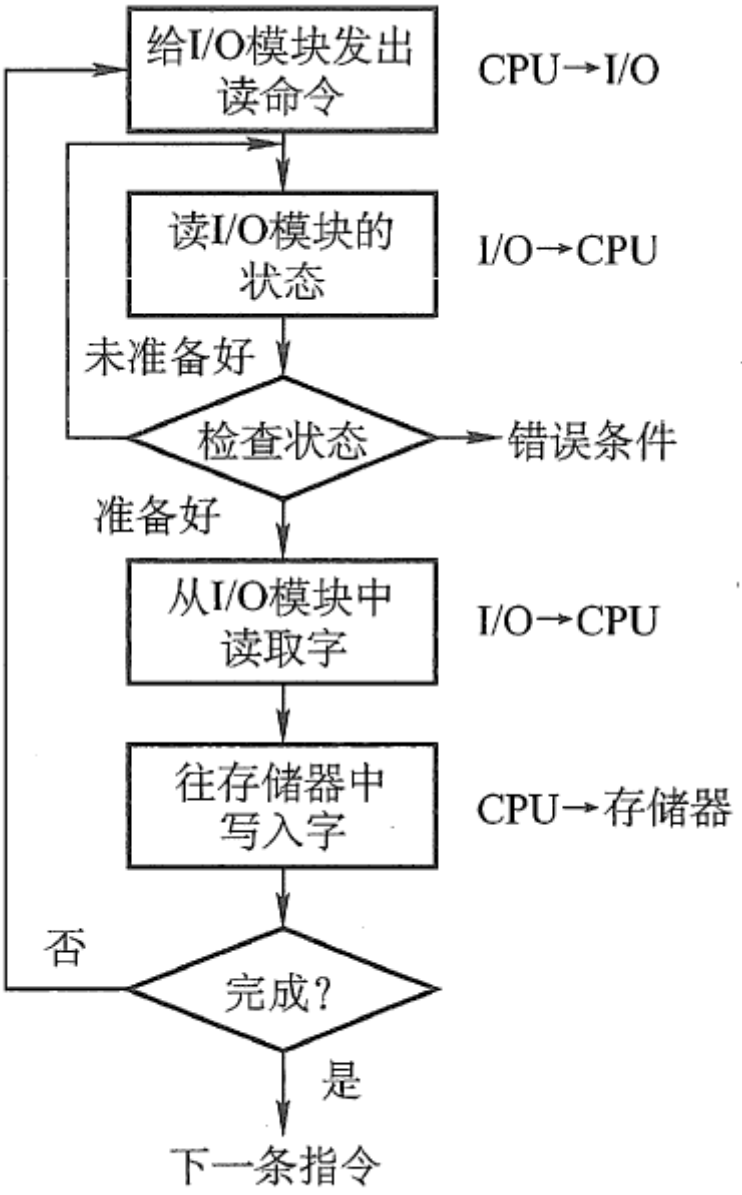
虚地址分为段号、段内页号、页内地址三部分。CPU 根据虚地址访存时，首先根据段号得到段表地址；然后从段表中取出该段的页表起始地址，与虚地址段内页号合成，得到页表地址；最后从页表中取出实页号，与页内地址拼接形成主存实地址。

段页式虚拟存储器的优点是，兼具页式和段式虚拟存储器的优点，可以按段实现共享和保护。缺点是在地址变换过程中需要两次查表，系统开销较大。

4.I/O控制方式

1.程序直接控制方式

如下图所示，计算机从外部设备读取数据到存储器，每次读一个字的数据。对读入的每个字， CPU 需要对外设状态进行循环检查，直到确定该字已经在I/0 控制器的数据寄存器中。在程序直接控制方式中，由于CPU 的高速性和I/0设备的低速性，致使CPU 的绝大部分时间都处于等待I/0 设备完成数据I/0 的循环测试中，造成了CPU 资源的极大浪费。在该方式中， CPU 之所以要不断地测试I/0 设备的状态，就是因为在CPU 中未采用中断机构，使I/0 设备无法向CPU报告它已完成了一个字符的输入操作。



(a) 程序直接控制方式

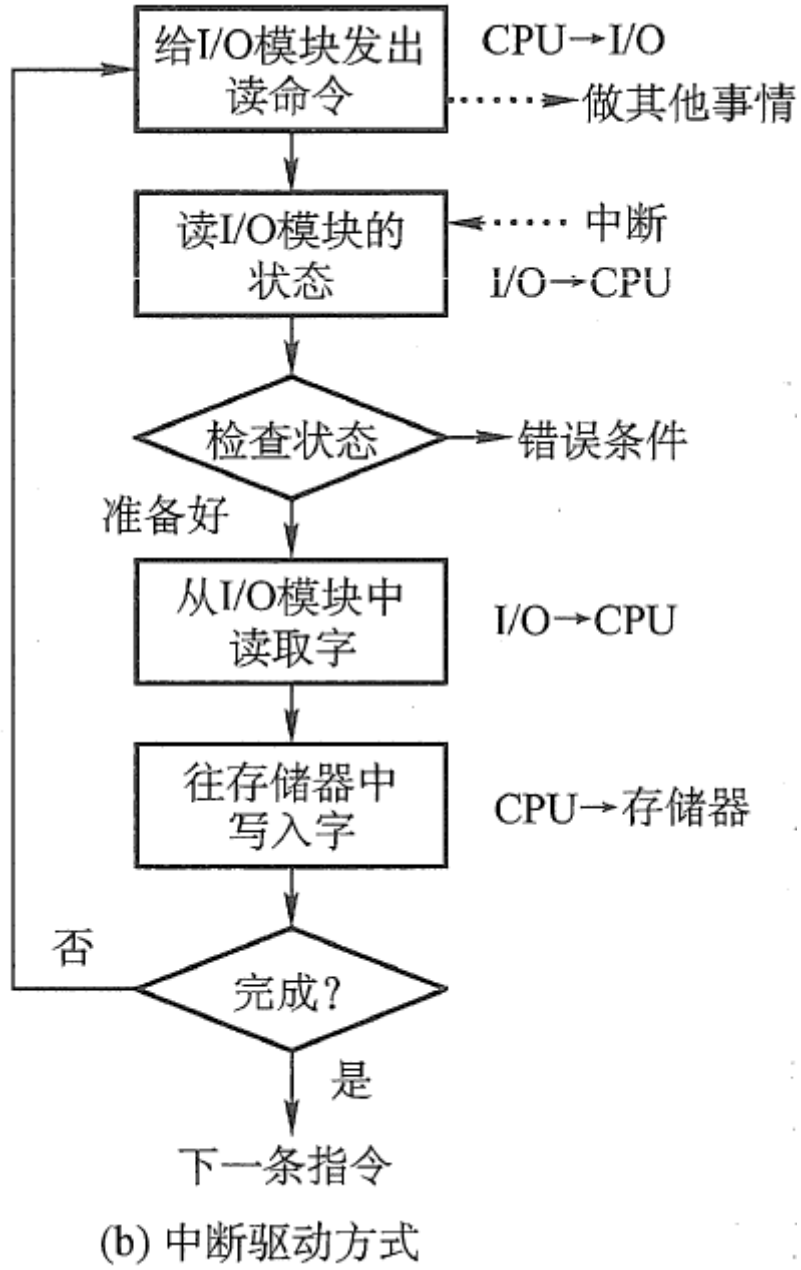
> 【痛点】程序直接控制方式虽然简单且易于实现，但其缺点也显而易见，由于CPU 和I/0 设备只能串行工作，导致CPU 的利用率相当低。

2.中断驱动方式

> 其实后一种的方式都是对前一种的方式的一种改进（OS里面都是为了提高资源利用率和并发性等才改进的，其他也差不多），408的课程很多都可以用这样的方法学习，不要硬背，这些都不是孤立的，回答的时候体现出改进，一步一步的讲出来，条理和逻辑会比较清楚！

中断驱动方式的思想是，允许I/0 设备主动打断CPU 的运行并请求服务，从而"解放"CPU，使得其向I/0 控制器发送读命令后可以继续做其他有用的工作。如下图所示，我们从I/0 控制器和CPU 两个角度分别来看中断驱动方式的工作过程:

> 【1】从I/0 控制器的角度来看，I/0 控制器从CPU 接收一个读命令，然后从外围设备读数据。一旦数据读入该I/0 控制器的数据寄存器，便通过控制线给CPU 发出一个中断信号，表示数据已准备好，然后等待CPU 请求该数据。I/0 控制器收到CPU 发出的取数据请求后，将数据放到数据总线上，传到CPU 的寄存器中。至此，本次I/0 操作完成， I/0 控制器又可开始下一次I/0操作。
> 【2】从CPU 的角度来看， CPU 发出读命令，然后保存当前运行程序的上下文（现场，包括程序计数器及处理机寄存器），转去执行其他程序。在每个指令周期的末尾， CPU 检查中断。当有来自I/0 控制器的中断时， CPU 保存当前正在运行程序的上下文，转去执行中断处理程序以处理该中断。这时， CPU 从I/0 控制器读一个字的数据传送到寄存器，并存入主存。接着， CPU 恢复发出I/0 命令的程序（或其他程序）的上下文，然后继续运行。



(b) 中断驱动方式

> 【痛点】：中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在存储器与I/0 控制器之间的传输都必须经过CPU, 这就导致了中断驱动方式仍然会消耗较多的CPU 时间。

> 【注意】：什么叫经过CPU呢?
> 答：在DMA(Direct memory access 直接存储器访问)之前，输入数据流大概是这样的：
> 【外围设备->I/O控制器的数据寄存器->CPU寄存器->存储器】，这就叫经过CPU，或者说传输数据的过程需要CPU的干预，于是引出了所谓的DMA(直接在I/O设备和内存之间建立数据通路)。

3.DMA方式

为何引入DMA前面提到了，所有改进如下：

1) 基本单位是数据块（前面是一个字）。

2) 所传送的数据，是从设备直接送入内存的，或者相反。

3) 仅在传送一个或多个数据块的开始和结束时，才需CPU 干预，整块数据的传送是在DMA控制器的控制下完成的。

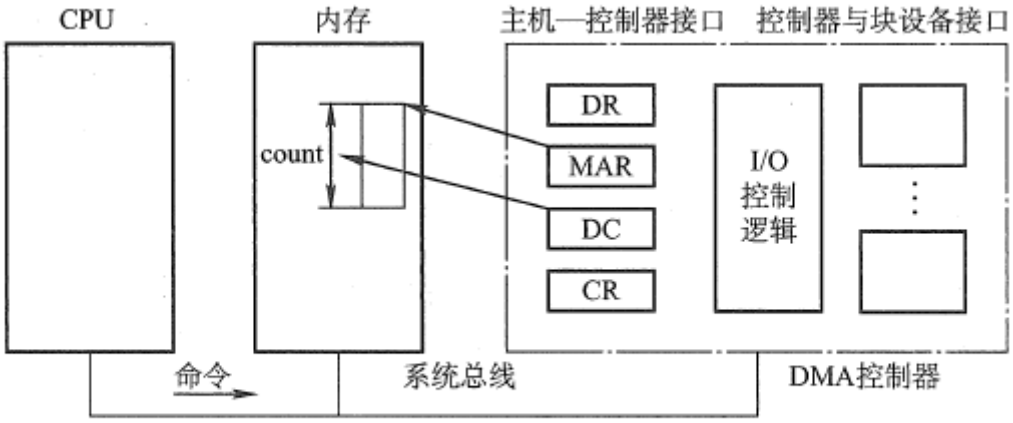


图 5.2 DMA 控制器的组成

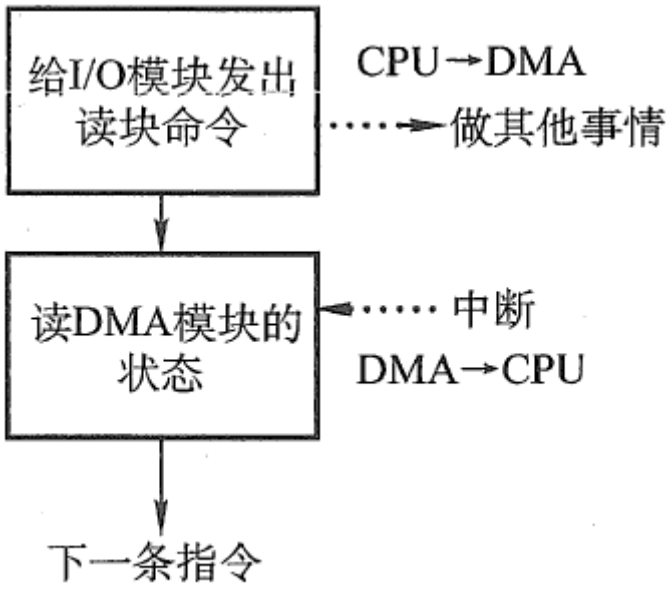
1) 命令 / 状态寄存器(CR) 。用于接收从CPU 发来的I/0 命令或有关控制信息，或设备的状态。

2) 内存地址寄存器(MAR) 。在输入时，它存放把数据从设备传送到内存的起始目标地址；在输出时，它存放由内存到设备的内存源地址。

3) 数据寄存器(DR) 。用于暂存从设备到内存或从内存到设备的数据。

4) 数据计数器(DC) 。存放本次要传送的字（节）数。

如下图所示，DMA方式的工作过程是： CPU 接收到I/O 设备的DMA 请求时，它给I/0 控制器发出一条命令，启动DMA 控制器，然后继续其他工作。之后CPU 就把控制操作委托给DMA 控制器，由该控制器负责处理。DMA 控制器直接与存储器交互，传送整个数据块，每次传送一个字，这个过程不需要CPU 参与。传送完成后，DMA 控制器发送一个中断信号给处理器。因此只有在传送开始和结束时才需要CPU的参与(预处理【设置CR、MAR、DC等】和后处理【中断处理、唤醒因该I/O阻塞的进程程等】)。



> DMA控制方式与中断驱动方式的主要区别是:中断驱动方式在每个数据需要传输时中断CPU, 而DMA 控制方式则是在所要求传送的一批数据全部传送结束时才中断CPU; 此外，中断驱动方式数据传送是在中断处理时由CPU 控制完成的，而DMA 控制方式则是在DMA 控制器的控制下完成的（前面提到了，加深印象！）。

> 【不算痛点】：如何进一步提高资源利用率呢？当然是请更牛逼的秘书（通道），老板（CPU）尽可能的从累活中解放出来。

4.通道控制方式

I/0 通道是指专门负责输入 / 输出的处理机。I/O通道方式是DMA方式的发展，它可以进一步减少CPU的干预，即把对一个数据块的读（或写）为单位的干预，减少为对一组数据块的读（或写）及有关控制和管理为单位的干预。同时，又可以实现CPU、通道和I/0 设备三者的并行操作，从而更有效地提高整个系统的资源利用率。

> 例如，当CPU要完成一组相关的读（或写）操作及有关控制时，只需向I/O 通道发送一条I/O 指令，以给出其所要执行的通道程序的首地址和要访问的I/0 设备，通道接到该指令后，执行通道程序便可完成CPU 指定的I/O任务，数据传送结束时向CPU发中断请求。(都有预处理和后处理，毕竟CPU才是老板！)

> I/O通道与一般处理机的区别是：通道指令的类型单一，没有自己的内存，通道所执行的通道程序是放在主机的内存中的，也就是说通道与CPU 共享内存。

> I/O 通道与DMA方式的区别是:DMA 方式需要CPU 来控制传输的数据块大小、传输的内存位置，而通道方式中这些信息是由通道控制的。另外，每个DMA 控制器对应一台设备与内存传递数据，而一个通道可以控制多台设备与内存的数据交换(包工头；也是上面提到的三者能并行的原因)。

5.总（举个栗子）

科班的同学应该不用看这个栗子也早就理解了，但为了照顾跨考的同学（不来个三连吗？），举个栗子总结以上4种方式：

> 想象一位客户要去裁缝店做一批衣服的情形。
> 1）采用程序直接控制时，裁缝没有客户的联系方式，客户必须每隔一段时间去裁缝店看看裁缝把衣服做好了没有，这就浪费了客户不少的时间。
> 2）采用中断驱动方式时，裁缝有客户的联系方式，每当他完成一件衣服后，给客户打一个电话，让客户去拿，与程序直接控制能省去客户不少麻烦，但每完成一件衣服就让客户去拿一次，仍然比较浪费客户的时间。
> 3）采用DMA 方式时，客户花钱雇一位单线秘书，并向秘书交代好把衣服放在哪里（存放仓库），裁缝要联系就直接联系秘书，秘书负责把衣服取回来并放在合适的位置，每处理完100 件衣服，秘书就要给客户报告一次（大大节省了客户的时间）。
> 4）采用通道方式时，秘书拥有更高的自主权，与DMA方式相比，他可以决定把衣服存放在哪里，而不需要客户操心。而且，何时向客户报告，是处理完100 件衣服就报告，还是处理完10000件衣服才报告，秘书是可以决定的。客户有可能在多个裁缝那里订了货，一位DMA 类的秘书只能负责与一位裁缝沟通，但通道类秘书却可以与多名裁缝进行沟通。

5.进程调度算法

1.前导知识简述

> 【问】：为什么要进行处理机调度?
> 若没有处理机调度，同意味着要等到当前运行的进程执行完毕后，下一个进程才能执行，而实际情况中，进程时常需要等待一些外部设备的输入，而外部设备的速度与处理机相比是非常缓慢的，若让处理机总是等待外部设备，则对处理机的资源是极大的浪费。而引进处理机调度后，可在运行进程等待外部设备时，把处理机调度给其他进程，从而提高处理机的利用率。用一句简单的话说，就是为了合理地处理计算机的软 / 硬件资源。

所谓进程调度方式，是指当某个进程正在处理机上执行时，若有某个更为重要或紧迫的进程需要处理，即有优先权更高的进程进入就绪队列，此时应如何分配处理机。通常有以下两种进程调度方式：
1) 非剥夺调度方式，又称非抢占方式。非剥夺调度方式是指当一个进程正在处理机上执行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在执行的进程继续执行，直到该进程完成或发生某种事件而进入阻塞态时，才把处理机分配给更为重要或紧迫的进程。

> 这种方式的优点是实现简单、系统开销小，适用于大多数的批处理系统，但它不能用于分时系统和大多数的实时系统。

2) 剥夺调度方式，又称抢占方式。剥夺调度方式是指当一个进程正在处理机上执行时，若有某个更为重要或紧迫的进程需要使用处理机，则立即暂停正在执行的进程，将处理机分配给这个更为重要或紧迫的进程。采用剥夺式的调度，对提高系统吞吐率和响应效率都有明显的好处。

> 但“剥夺”不是一种任意的行为，必须遵循一定的原则，主要有优先权、段进程优先和时间片原则等

调度的基本评价准则

1) CPU利用率
2) 系统吞吐量：单位时间内CPU完成作业的数量
3) 周转时间：作业完成时间 - 作业提交时间
4) 等待时间：等待时间指进程处于等处理机状态的时间之和，等待时间越长，用户满意度越低。处理机调度算法实际上并不影响作业执行或输入 / 输出操作的时间，只影响作业在就绪队列中等待所花的时间。因此，衡量一个调度算法的优劣，常常只需简单地考察等待时间。
4) 响应时间：响应时间指从用户提交请求到系统首次产生响应所用的时间。在交互式系统中，周转时间不可能是最好的评价准则，一般采用响应时间作为衡量调度算法的重要准则之一。从用户角度来看，调度策略应尽量降低响应时间，使响应时间处在用户能接受的范围之内。

2.先来先服务调度算法（FCFS)

FCFS 调度算法是一种最简单的调度算法，它既可用于作业调度，又可用于进程调度。在作业调度中，算法每次从后备作业队列中选择最先进入该队列的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。

在过程调度中， FCFS调度算法每次从就绪队列中选择最先进入该队列的进程，将处理机分配给它，使之投入运行，直到完成或因某种原因而阻塞时才释放处理机。

> FCFS调度算法属于不可剥夺算法。从表面上看，它对所有作业都是公平的，但若一个长作业先到达系统，就会使后面的许多短作业等待很长时间，因此它不能作为分时系统和实时系统的主要调度策略。但它常被结合在其他调度策略中使用。例如，在使用优先级作为调度策略的系统中，往往对多个具有相同优先级的进程按FCFS原则处理。

> FCFS 调度算法的特点是算法简单，但效率低；对长作业比较有利，但对短作业不利（相对SPF和高响应比）；有利于CPU繁忙型作业，而不利于I/0繁忙型作业。【跨考解答】：为什么CPU繁忙型是长作业，因为长短是使用CPU的长短，I/O繁忙型使用CPU比较短。

3.短进程优先调度算法（SPF)

短作业（进程）优先调度算法是指对短作业（进程）优先调度的算法。短进程优先(SPF) 调度算法从就绪队列中选择一个估计运行时间最短的进程，将处理机分配给它，使之立即执行，直到完成或发生某事件而阻塞时，才释放处理机。（每次调度都选就绪队列中最短的）

> SPF调度算法也存在不容忽视的缺点：
>
> 1) 该算法对长作业不利。更为严重的是，若有一长进程进入就绪队列，由于调度程序总是优先调度那些（即使是后来进来的）短作业，将导致长作业长期不被调度（饥饿）。
> 2）该算法完全未考虑作业的紧迫程度，因而不能保证紧追性作业会被及时处理。
> 3）由于作业的长短只是根据用户所提供的估计执行时间而定的，而用户又可能会有意或无

> 意地缩短其作业的估计运行时间，致使该算法不一定能真正做到短作业优先调度。

> 【注意】 SPF调度算法的平均等待时间、平均周转时间最少。

4.优先级调度算法

在进程调度中，优先级调度算法每次从就绪队列中选择优先级最高的进程，将处理机分配给它，使之投入运行。

根据新的更高优先级进程能否抢占正在执行的进程，可将该调度算法分为如下两种：
1) 非剥夺式优先级调度算法。当一个进程正在处理机上运行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在运行的进程继续运行，直到由于其自身的原因而主动让出处理机时（任务完成或等待事件），才把处理机分配给更为重要或紧迫的进程。
2) 剥夺式优先级调度算法。当一个进程正在处理机上运行时，若有某个更为重要或紧迫的进程进入就绪队列，则立即暂停正在运行的进程，将处理机分配给更重要或紧迫的进程。

而根据进程创建后其优先级是否可以改变，可以将进程优先级分为以下两种：
1) 静态优先级。优先级是在创建进程时确定的，且在进程的整个运行期间保持不变。确定静态优先级的主要依据有进程类型、进程对资源的要求、用户要求。
2) 动态优先级。在进程运行过程中，根据进程情况的变化动态调整优先级。动态调整优先级的主要依据有进程占有CPU 时间的长短、就绪进程等待CPU 时间的长短。

一般来说，进程优先级的设置可以参照以下原则：
1) 系统进程 > 用户进程。系统进程作为系统的管理者，理应拥有更高的优先级。
2) 交互型进程 > 非交互型进程（或前台进程 > 后台进程）。大家平时在使用手机时，在前台运行的正在和你交互的进程应该更快速地响应你，因此自然需要被优先处理，即要有更高的优先级。
3) I/0 型进程 > 计算(CPU)型进程。我们知道， I/0 设备（如打印机）的处理速度要比CPU 慢得多，因此若将I/0 型进程的优先级设置得更高，就更有可能让I/0 设备尽早开始工作，进而提升系统的整体效率。

5.时间片轮转调度算法

时间片轮转调度算法主要适用于分时系统。在这种算法中，系统将所有就绪进程按到达时间的先后次序排成一个队列，进程调度程序总是选择就绪队列中的第一个进程执行，即先来先服务的原则，但仅能运行一个时间片，如l00ms 。在使用完一个时间片后，即使进程并未完成其运行，它也必须释放出（被剥夺）处理机给下一个就绪的进程，而被剥夺的进程返回到就绪队列的末尾重新排队，等候再次运行。

> 在时间片轮转调度算法中，时间片的大小对系统性能的影响很大。若时间片足够大，以至于所有进程都能在一个时间片内执行完毕，则时间片轮转调度算法就退化为先来先服务调度算法。若时间片很小，则处理机将在进程间过于频繁地切换，使处理机的开销增大，而真正用于运行用户进程的时间将减少。因此，时间片的大小应选择适当。

> 时间片的长短通常由以下因素确定：系统的响应时间、就绪队列中的进程数目和系统的处理能力。

6.高响应比优先调度算法

高响应比优先调度算法是对FCFS调度算法和SPF调度算法的一种综合平衡，同时考虑了每个作业的等待时间和估计的运行时间。在每次进行作业调度时，先计算后备作业队列中每个作业的响应比，从中选出响应比最高的作业投入运行。响应比的变化规律可描述为:
响应比 = (等待时间 + 要求服务时间)/要求服务时间
根据公式可知：
1) 作业的等待时间相同时，要求服务时间越短，响应比越高，有利于短作业。
2) 要求服务时间相同时，作业的响应比由其等待时间决定，等待时间越长，其响应比越高，因而它实现的是先来先服务。
3) 对于长作业，作业的响应比可以随等待时间的增加而提高，等待时间足够长时，其响应比便可升到很高，从而也可获得处理机。因此，克服了饥饿状态，兼顾了长作业。

7.多级反馈队列调度算法

多级反馈队列调度算法是时间片轮转调度算法和优先级调度算法的综合与发展，如下图所示。通过动态调整进程优先级和时间片大小，多级反馈队列调度算法可以兼顾多方面的系统目标。例如，为提高系统吞吐量和缩短平均周转时间而照顾短进程；为获得较好的I/0 设备利用率和缩短响应时间而照顾I/0 型进程；同时，也不必事先估计进程的执行时间。

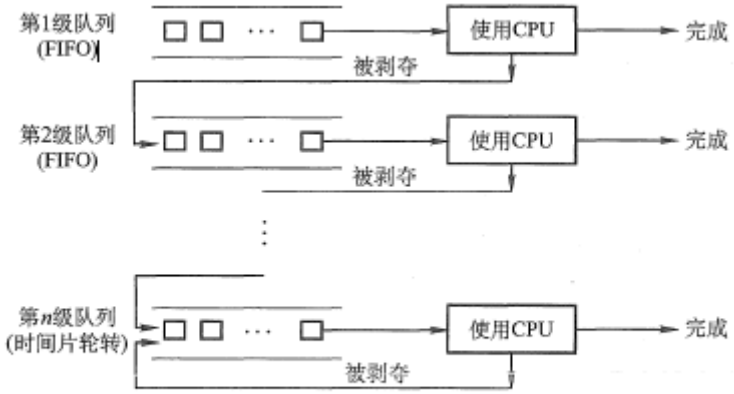


图 2.7　多级反馈队列调度算法

多级反馈队列调度算法的实现思想如下：

1) 设置多个就绪队列，并为各个队列赋予不同的优先级，第1 级队列的优先级最高，第2级队列次之，其余队列的优先级逐次降低。

2) 赋予各个队列中进程执行时间片的大小各不相同。在优先级越高的队列中，每个进程的运行时间片越小。例如，第2级队列的时间片要比第1级队列的时间片长1倍。

3) 一个新进程进入内存后，首先将它放入第1 级队列的末尾，按FCFS 原则排队等待调度。当轮到该进程执行时，如它能在该时间片内完成，便可准备撤离系统；若它在一个时间片结束时尚未完成，调度程序便将该进程转入第2 级队列的末尾，再同样按FCFS 原则等待调度执行；若它在第2 级队列中运行一个时间片后仍未完成，再以同样的方法放入第3级队列……

4) 仅当第1 级队列为空时，调度程序才调度第2 级队列中的进程运行；仅当第1~(i-1)级队列均为空时，才会调度第i 级队列中的进程运行。若处理机正在执行第i 级队列中的某进程，这时又有新进程进入优先级较高的队列［第1~(i-1)中的任何一个队列］，则此时新进程将抢占正在运行进程的处理机，即由调度程序把正在运行的进程放回第i 级队列的末尾，把处理机分配给新到的更高优先级的进程。

> 多级反馈队列的优势有以下几点：
>
> 1) 终端型作业用户：短作业优先。
>
> 2) 短批处理作业用户：周转时间较短。
>
> 3) 长批处理作业用户：经过前面几个队列得到部分执行，不会长期得不到处理。