华中科技大学

硕士学位论文

多媒体系统任务调度算法研究

姓名：张炫

申请学位级别：硕士

专业：计算机系统结构

指导教师：文远保

20060510

摘　要

随着多媒体编码技术和网络通信技术的发展，出现了很多以“流”为特征的多媒体应用，如视频会议、网络监控等。这些应用具有实时性要求，如视频会议系统的声音和图像都不允许停顿。但与传统的硬实时应用不同，在该类应用中，偶尔的丢失帧不会造成严重的后果，只要求从概率上保证数据处理在限定的时间内完成，所以我们称这样的多媒体流应用为软实时应用。

多媒体流应用对多媒体系统当中的存储、传输和处理机调度都提出了实时化的要求，多媒体系统需要合理地分配处理机、网络带宽、磁盘等资源，从而提供实时性保证。其中，处理机是多媒体系统的核心资源，好的处理机调度算法能在控制任务死线丢失率的前提下，提供较好的处理机资源利用率，这是满足多媒体应用实时要求的关键。由于传统分时操作系统不能很好地支持实时多媒体应用，所以研究新的多媒体流应用的实时处理机调度算法成为目前的研究热点。

在以往的研究中，针对多媒体流任务周期性和突发性的特点，主要的处理机调度方式有直接借鉴传统周期模型的悲观、乐观方法，以及启发式调度和基于周期多帧模型的调度方式。但这些算法存在着没有准确的可调度测试条件、不能保证多媒体任务的实时性或者资源利用率不高等问题。

为了解决上述问题，把多媒体流任务称为不规则周期任务。根据这个概念，给出不规则周期任务模型，将多媒体流任务转换为两部分：传统周期性任务和突发性任务，分别采用单调比率算法和零星服务器算法来进行调度。这种混合调度策略不仅能对周期性任务提供硬实时保证，还能为突发性任务提供软实时性保证，即保证突发性任务丢失死线的概率不超过某一上限值，从而满足多媒体流任务的实时性要求。同时有效地增加了系统可接纳的任务数，提高了处理机的资源利用率。

关键词： 多媒体，实时性，任务调度，单调比率，多媒体流

Abstract

With the development of multimedia coding technology, a lot of multimedia applications with the characteristic of stream are popular. These applications need to be processed in the restricted time, for example, sound and picture can not stop in the video conference system. And It is different with traditional real-time applications, missing frames occasionally will not cause the severe result in the kind of applications which just demand the probabilistic success of the data process finished in the restricted time. So we call the kind of application soft real-time application.

Multimedia stream applications need the real-time support from the storage, transmission and CPU(Center Processing Unit) scheduling in the multimedia system. And CPU is one of the important resources in the system. A good CPU task scheduling algorithm can offer low deadline missing percent of the tasks and high CPU utilization,which are key factors for real-time multimedia application. Due to traditional way of time-sharing scheduling can not satisfy the real-time demand of multimedia application, the real-time scheduling algorithm for multimedia stream was paid special attention.

In the past research, major scheduling ways are pessimistic and optimistic idea which are based on traditional periodic model, periodic multiframe task model and heuristic scheduling. But there are some problems such as not having exact scheduling conditions or low CPU utilization.

For solving the above problems, the multimedia stream task is called irregular periodic task. According to the notion,the irregular periodic model is presented to transform it to a periodic task and a bursting task. The task set is scheduled in the way, that is periodic task scheduled by RM(Rate Monotonic) and bursting task by sporadic server. The way not only offer the hard real-time guarantee for the periodic task ,but also keep the missing deadline percent of the bursting task low to satisfy the soft real-time demand so that guarantee multimedia stream task real-time.At the same time,the system can allow more tasks to enter and improve the CPU resource utilization.

Key words: multimedia, real-time, task scheduling, Rate Monotonic, multimedia stream

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到，本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密? ，在_____年解密后适用本授权书。

　　不保密?。

（请在以上方框内打“v”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

1 绪 论

1.1 课题背景

随着计算机性能和可靠性的不断提高，其应用已深入到社会的各个领域。从军用高技术装备到信息家电，从通信设备到医疗器械，从工业控制到智能仪器，处处都需要计算机。随着应用领域不断拓宽，计算机中的应用程序越来越复杂，类型也越来越多。

在工业控制系统和实验仪器的监控管理系统中要求计算机能对各种外部、异步事件做出立即反应，这一类应用对时间十分敏感，要求系统必须在一定的时间内做出反应，否则将造成严重的后果。这一类应用称为硬实时应用(Hard real-time applications)。

大量的传统应用(如WWW、FTP、Telnet等)对时间并不敏感，这样的应用称为尽力而为应用(Best-effort applications)。

目前，视频压缩和通信技术发展迅速，出现了大量的以“流”为特征的多媒体应用(如：视频会议、远程教育、虚拟现实、VOD等)。该类应用要求连续地播放视频和音频帧，运算结果的正确性与时间相关。与硬实时应用不同的是，在这类应用中偶尔的帧丢失是可以允许的，它们只要求从概率上保证数据处理在限定的时间内完成，所以我们称多媒体应用为软实时应用(Soft real-time applications)[1]。多媒体流应用对存储、传输和处理机调度都有实时要求，更确切地说，它们要求多媒体系统合理地分配CPU、网络带宽、磁盘等资源，从而提供实时性保证。

CPU作为多媒体系统核心资源，CPU任务调度算法是保证多媒体流应用实时要求的重要因素，而传统分时操作系统中CPU调度方式无法满足实时多媒体应用环境。正是在这种背景下，本文提出一种新的支持多媒体流应用的CPU调度策略。

1.2 国内外概况

本节对影响多媒体系统实时性能的相关技术研究状况做简要介绍，并分析了

WINDOWS NT/2000/XP 和 UNIX 等通用操作系统的实时调度策略，以及它们支持实时多媒体应用存在的问题。

1.2.1 多媒体系统实时性问题研究概况

由于多媒体应用实时性的特点，对多媒体系统中的存储、传输和处理机系统都有相应的要求，从而保证多媒体任务在规定时间内完成。下面对涉及到的相关问题如：网络传输、媒体同步、CPU 调度管理、磁盘管理等研究状况逐一进行阐述：

(1) 多媒体通信中实时传输问题

为了确保在网络上多用户能共享多媒体的信息资源，就要保证共享的声音和视频图像的质量，多媒体信息在网络上的实时传输问题至关重要。由于信息数字化后，数据量差别很大，因此对实时传输也提出不同的要求[2]：

① 语音信息的实时传输

由于语音信息数字化后的数据量相对较小，确保声音质量要求的带宽为 64Kb/s，语音采集、处理的硬件也相对便宜，因此大多数多媒体系统都首先保证语音信息的通信质量，如大多采用 ADPCM 编码（Adaptive Differential Pulse Code Modulation）。因此，在一般网络环境下确保语音信息的质量是不成问题的。语音信息实时传输需要注意的问题：

(a) 信息包的大小问题

如果选用的语音包大，对网络的传输利用率就高，而且若发生丢包和错误，产生的语音错误的范围也大；如果选用的语音包较小，当偶尔的丢失或出错时，对语音效果影响不大，但传输利用率也较低。信息包大小的选择通常兼顾网络利用率和语音效果两个方面。

(b) 传输延迟问题

据统计，对语音传输来说，最大可接受的延迟是 0.25s，这是指人可接受的效果而言。若要求高质量的语音效果，则通信延迟应小得多。具体应根据不同条件和具体要求采用不同的延迟限制。

② 视频信息的实时传输

视频信息数字化后数据量很大，这就给传输带来很大困难。彩色全动图像每秒

30 帧需要的传输率至少为 240Mbit。这样大的数据量，如果不压缩的话，传输、处理和存储都会遇到很大问题。对视频图像数据而言，偶尔的丢包是允许的，因为下一帧马上补充过来，在视觉上不会造成很大影响，但是如果一幅图像数据中有错，则有可能在屏幕上出现一些错误图案。因此，若传输中出现错误，最好丢掉此包。为此，在传输中需要进行一些检查报文正确性的工作。

在多媒体通信过程中，由于网络带宽有限，实现全屏幕全色彩动态图像传输比较困难。目前，提高视频质量一般只作为一种锦上添花的措施，在保证语音、图形、正文都能获得较满意效果的前提下才加以考虑。而有些应用场合，如视频会议，由于图像相对运动较少，且动作幅度、速度都不会很大，因此，在网络带宽受限的情况下，适当减少每秒的幅数也是允许的。

(2) 多媒体音频视频同步问题

多媒体同步是多媒体研究和开发应用的关键技术[2]。当前国内外流行的各类多媒体系统中，除了文字和图形之外，最典型的是声音和图像。常见的同步方法是在数据压缩期间就生成三个文件：数字视频信号数据文件、数字声音信号数据文件、同步信号数据文件。在同步信号数据文件中，每一个声音采样记录的时间标记，对应于每个视频帧标记。在回放演播时，对每个视频进行解压缩，把同步文件中当前声音信号的时间标记与这个视频帧信息标记相比较，可出现两种情况：一种情况是可能声音落后于视频，此时视频帧的显示就延迟到使视频与声音信号同步；另一种情况，可能声音超前于视频，当前视频帧放弃，而对下一个视频帧进行解压缩，并且重复地进行比较。下面介绍几种多媒体流同步的方法：

① 分层同步法

在这种同步方法中，多媒体对象被看成一棵树，而树中的每个节点代表了分支的串行或并行表现。分层同步主要基于两种同步操作：一种是串行同步动作，另一种是并行同步动作。动作可以是单一的也可以是复合的，单一的动作用来处理单个媒体对象的表现，复合动作是同步操作和单一动作的结合。分层同步法的层次模型结构清晰，在媒体对象合成后，便于更改，有利于合成多媒体对象的编辑。正是由于其操作简单，使其得到了广泛的应用。然而，由于分层结构带来的限制使得同步只能发生在起点和

终点，这就使得分层同步无法从媒体对象中进行抽取，只能达到粗粒度的同步效果。从而决定了在很多情况下无法用分层结构来描述同步。

② 时间轴同步法

将所有独立的对象都关联到一个时间轴上进行描述的方法称之为时间轴同步法。去掉时间轴上的任何一个对象都不会影响其他对象的同步，这种同步方法需要维持一个整体时间，每个对象可以将整体时间映射到它的局部时间，并依据此局部时间来表现。当局部时间与整体时间的误差超出一个指定限度时，则需要重新与整体时间进行比较并校准。时间轴同步能较好地表达源于媒体对象内部结构的抽象含义。在这方面，它优于分层同步法，它定义了一个与视频中某图像相关说明文字的演示起始位置，而不再要求有相关视频帧的知识。

③ 参考点同步法

时间相关的媒体表现被认为是一个具有固定时片表现的离散子单元序列。子单元在对象中的位置被称为参考点，参考点同步被定义为同时表现的两个不同对象的子单元的关系。与基于时间轴的同步方法类似，基于参考点的同步可以在表现的任何时候实现同步。但是，相对分层同步来说，基于参考点的同步还需要检测媒体之间的不协调，这就增加了操作的复杂性。另外，单纯的基于参考点同步不能实现多媒体表现中的延迟动作。

(3) CPU 实时调度问题

针对处理机资源，多媒体应用的实时性要求在保证较低的任务死线丢失率前提下，提高 CPU 资源利用率。目前的通用操作系统如 WINDOWS NT/2000/XP 和 UNIX 采用分时原则进行调度，它们的目标是在多用户多道程序的环境下提供公平性，优化系统的平均功能。而多媒体任务实时性要求系统的响应时间有界、可预测。因此，通用操作系统的调度策略无法满足具有实时需求的多媒体应用程序的要求。本文后面的内容会对 CPU 实时调度问题做更深入地研究。

(4) 实时磁盘调度

在一些多媒体应用中，如网上视频点播服务(Video On Demand,VOD)，分布式系统中的数据存取，数字地图中的海量数据存储、传输等，磁盘均被认为是其主要的性

能瓶颈[3]。影响磁盘 I/O 性能的因素主要由其数据访问时间决定，对于硬盘数据块的访问，其花费的时间由下列 3 部分组成：

① 寻道时间：磁盘磁头从当前位置移动到读写数据所在磁道所用的时间。通这部分时间是磁盘服务时间中最重要的因素，也是最不容易确定的，磁头的最大寻道时间和最小寻道时间相差很大，一般达到 15ms 左右。

② 旋转等待时间：磁头移动到正确的磁道后，等待所需扇区旋转到磁头下的时间。这部分时间主要由磁盘转速决定，如转速为 7200r/min 的硬盘其平均旋转等待时间为 3.4ms。

③ 数据传输时间：将磁盘上的数据传输到内存所花费的时间。这个时间在整个磁盘服务时间中只占很小一部分，将其忽略基本上不会产生影响。

在实时应用中，一般都会涉及到要求在同一时间段内读出多个不同的磁盘块以服务于多个请求，这样就要求磁头在限定的时间里，移动到每个数据块所在的位置读取数据。一般而言，在一个磁盘系统中，磁盘的转速和数据的读出速率都是固定的。每个数据访问请求的时间开销主要在寻道上，所以若能减少每个请求的磁头寻道开销，就能大大缩短总服务时间，从而提高磁盘的 I/O 性能，以便能提供更多的实时服务。从本质上说，减少磁盘寻道开销的关键在于如何减少磁头的移动距离。

1.2.2 操作系统 CPU 调度发展概况

(1) 调度的基本概念

像内存和终端一样，CPU 也是一个共享的资源系统中的许多进程都争用 CPU，操作系统必须决定在所有进程之间分配 CPU 资源。调度器作为操作系统的一个组成部分，它决定在任一时刻哪个进程去运行，以及这个进程能运行多长的时间。对调度器的研究集中在两个方面：第一个是策略，即决定哪个进程运行及什么时候切换到另一个进程的规则；第二个是实现，即实现这种策略的数据结构和算法[4]。

进程调度有非剥夺（Non-Preemptive Mode）和剥夺(Preemptive Mode)两种基本方式。非剥夺方式规定，调度器一旦把 CPU 分配给进程后便让它一直运行下去，直到进程完成或发生某事件而阻塞时，才把 CPU 分配给另一进程。这种方式的缺点是：

一个紧急任务到达时，不能立即投入运行，以至耽误时机；若干个后到的短作业，须等待长作业运行完毕，导致短作业的周转时间增长。

剥夺方式规定，当一个进程正在运行时，系统可以基于某种原则，剥夺分配给它的 CPU，将之分配给其它进程。剥夺的原则通常有：优先级原则；短作业优先原则；时间片原则。目前，绝大部分操作系统都采用剥夺方式。

(2) 基本的调度策略

基本的调度策略有静态优先权法，动态优先权法，时间片轮转法三种[5]。

静态优先权法在进程创建时按进程执行任务的轻重缓急确定每个进程的优先级，并且在进程运行过程中不再动态的改变优先级。静态优先权的确定方法通常是：按进程的内容确定；按作业要求资源的类型和数量确定；按作业递交的时间次序确定；按用户的类型和要求确定。

静态优先权法实现简单，但是不能反映系统以及进程在运行过程中发生的各种变化情况。动态优先权法则克服了这种缺陷，在进程运行过程中可以根据变化的情况不断调整进程的优先级，因此能获得较好的效果。在动态地确定进程优先权时，同样要考虑前述一般原则，但实施起来可以更加精确。例如，一个进程执行不同的程序段有轻重缓急之分，动态优先权法可以不断调整该进程的优先级，使之发生相应的高低起伏。

时间片轮转法规定由各个就绪进程顺次轮流使用 CPU，且每次使用的时间长度一般为一定值，这种时间长度称为时间片。如果一个进程运行了一个时间片之后还没有结束，则进入就绪队列末尾，等待下一轮循环。如果由于等待某种事件，进程没有用完时间片就进入睡眠状态，则当它被唤醒后可按其时间片中未用完部分的值插入就绪队列的某个位置。这种方法的优点是各进程能比较均衡的共享使用 CPU，因此适用于分时系统。

1.2.2 传统操作系统实时任务调度策略研究概况

(1) SVR4调度器

基于分时原则的UNIX不能很好地满足实施任务的需求，AT&T为了开拓UNIX在实时环境中的应用，设计了SVR4调度器[6]。

SVR4把进程分为三类：实时类、系统类和分时类。并为每个进程设计了两个优先级：进程调度所依据的全局优先级和进程所在类的类内优先级。全局优先级范围为0-159，0-59为分时类，60-99为系统类，100-159为实时类。与传统UNIX不同的是，SVR4调度器中数值越大优先级越高，因此实时进程的优先级最高。各类进程还具有类内优先级，实时进程为rtproc.rt_pri，分时进程为tsproc.rt_pri。系统参数表rt_dptbl[]和ts_dptbl[]用于类内优先级到全局优先级之间的映射。映射时保证分时类的优先级决不会高于实时类的优先级。类内优先级体现了各类进程的特性，并且方便用户设置进程的优先级。而全局优先级又保证了进程的统一管理。

SVR4调度器实时类的设置可以使实时进程得到最高的优先级，可抢占点的设置可以提高系统对实时应用的响应时间。然而，这种简单的策略没有区分硬实时和软实时应用，并且不能保证软实时应用的实时需求，同时，系统中实时应用负载太重时，会导致普通的分时进程饥饿。

（2）LINUX 调度策略分析

Linux是一个自由的、提供源码的类UNIX操作系统，在提供的用户界面上和UNIX一致，但它的内核却是完全重写的。在Linux中也实现了对实时应用的简单支持。Linux中把用户进程分为非实时进程和实时进程两类。非实时进程有两种优先级，一种是静态优先级，另一种是动态优先级。实时进程又增加了第三种优先级，实时优先级[7][8]。具体而言，Linux选择进程的依据为以下四项：policy，priority，counter，rt_priority。policy是进程的调度策略，实时进程采用FIFO和RR两种调度策略，非实时进程采用动态优先调度，用SCHED_OTHER表示；priority是进程(实时和普通)的优先级；counter的初值由priority赋给，每次时钟滴答时减1；rt_priority为实时进程所特有，用于实时进程的选择。首先，Linux根据policy从整体上区分实时进程和普通进程。对于普通进程，Linux采用动态优先调度，选择的依据是进程counter的大小，counter的初值由静态优先级policy赋给，一般是0-70之间的数字，counter在每个时钟滴答时减l，而policy保持不变，用于在counter变为0时，并且所有处于可运行状态的普通进程的时间片用完了之后，对counter进行重新赋值，从而使该进程有被重新调度的机会。

对于实时进程，Linux采用了两种策略，即FIFO和RR。实时进程的counter只是用

来表示该进程剩余的时间片，并不作为衡量它是否值得运行的标准，这是和普通进程有区别的，它是否值得运行的标准是它的rt_priority。系统遍历可运行队列，用goodness()计算每一个可运行进程的度量值，选择goodness()返回值最大的进程运行。由于实时进程是1000+p->rt_priority，总是大于非实时进程，故保证了实时进程的优先运行，而rt_priority起到的是在实时进程之间区分优先级的大小。

Linux采用了简洁的调度设计，巧妙的保证了实时进程的优先执行，而非实时进程按公平的原则进行分时调度。这种简单的设计在大多数情况下都能保持较好的性能，但面对实时任务和普通任务并存的环境，它和SVR4存在同样的问题。

（3）Windows NT/2000/XP 调度策略概况

Windows NT/2000/XP也采用多极反馈调度技术，其中基本的调度单位是线程[9]。支持32个线程优先级，0优先级由系统专用等待线程使用，1-15给普通线程使用，16-31给实时线程使用。

它们的调度策略较为简单，通过SetThreadPriority()可以生成线程的初始优先级，初始优先级落在16－31的称为固定优先级线程，初始优先级落在1－15之间的称为可变优先级线程。处于固定优先级的实时进程无论处在用户态还是内核态都保持优先级不变。对于处在1－15的可变优先级线程，它在用完时间片时降低优先级，在挂起后又再次就绪时升高优先级，在内核态运行时也可升高优先级(但不会超过15)。

为每个优先级设置一个就绪队列，新的可运行线程被挂在就绪队列后面，如果线程没有用完时间片就被剥夺则仍放在队列前面。调度的规则是选择高优先级的线程，如果同一优先级的线程有多个，则选择就绪队列最前面的。

与SVR4比较，Windows NT/2000/XP内核中没有设置抢占点，当线程运行在内核中时，如果不产生硬件中断，高优先级的线程是无法抢占低优先级的进程，因此对实时响应的效果要比SVR4还差。

1.2.3 传统操作系统调度策略在实时多媒体环境中的问题

虽然在SVR4 UNIX、Linux和WINDOWS NT/2000/XP中都实现了对实时进程的支持，但都是简单地把实时进程的优先级设为最高，优先执行实时任务，这样依然存在许多问题：

(1) 如果简单地把实时多媒体任务的优先级设为最高将会导致该类任务完全霸占CPU，造成普通进程饥饿。例如：在SVR4中进行了一个实验[10]，在系统中并发地运行三个不同的程序：一个交互式键入会话，一个批处理作业，一个电影图像演示程序。把电影图像演示程序设为实时类，结果图像程序本身都不能很好的运行。原因在于播放图像的X服务器得不到足够的CPU时间。如果把图像演示程序和X服务器都设为实时类，则图像演示能很好地运行，但交互式作业和批处理作业就几乎停止了，系统不能响应鼠标和键盘。事实上，多媒体任务的实时性并不能说明其重要性就一定高于普通进程。

(2) 没有区分硬实时和软实时应用。硬实时对时间要求苛刻，而软实时允许在一定概率内丢失死线。如果系统按硬实时定制，会导致浪费大量系统资源。按软实时定制系统，则硬实时应用也可能丢失死线。因此，应该对硬实时和软实时应用区分对待。

(3) 一般操作系统的内核是不可抢占的。实时进程有可能阻塞于系统调用，对于硬实时应用而言，会导致响应时间的不可预测性。同时，在系统中可能因低优先级进程拥有高优先级进程的资源，从而阻塞了高优先级进程的执行。

1.3 本课题主要研究工作

本课题主要研究工作：分析多媒体流编码技术，针对压缩编码造成的多媒体流任务突发性、周期性的特点，研究常见调度方法的不足。采用任务转换的思想，构建出一种不规则周期任务模型以及调度方法。通过理论分析和仿真实验证明，该方法能够在控制任务死线丢失率的同时有效利用 CPU 资源。主要研究内容包括：

(1) 研究流行的多媒体流编码技术以及分析多媒体流任务的特点。

(2) 分析常见多媒体流任务实时调度方式及其存在的问题。

(3) 针对多媒体流任务周期性和突发性的特点提出任务转换的思想。

(4) 给出任务模型和调度方法，以及接入控制算法。

(5) 进行仿真实验。

2　多媒体流编码技术

本章将简单介绍多媒体编码系统的结构以及主要的多媒体流编码标准，并对多媒体流任务的特点进行分析，为后面的课题研究提供必要的背景知识。

2.1　多媒体编码系统综述

多媒体编码技术是多媒体领域的关键技术之一。编码的主要目的是要减少音频视频序列的码率，以便能够在给定的通信信道上实时传输。信道带宽因应用和传输媒体的不同而异。除了通信应用系统之外，储存和检索也需要编码，不同存储介质有不同容量和存取速度，因此需要不同压缩量。由于视频文件数据量大，不利于传输和储存，多媒体视频编码算法成为研究的重点。多媒体视频编码算法的组成在很大程度上是由视频序列建模所采用的信源模型确定的。编码器寻求用它的信源模型描述视频序列的内容。图 2.1 给出视频编码系统的基本组成[11]。在编码器中，首先用信源模型的参数描述数字化的视频序列。下一步，信源模型参数被量化成有限的符号集。量化参数取决于比特率与失真间所期望的折衷。最后，用无损编码技术把量化参数映射成二进制码字；这种技术进一步利用量化参数的统计特性。解码器反向进行编码器的二进制编码和量化过程，重新得到信源模型的量化参数。然后，解码器的合成算法用信源模型的量化参数计算解码的视频帧。

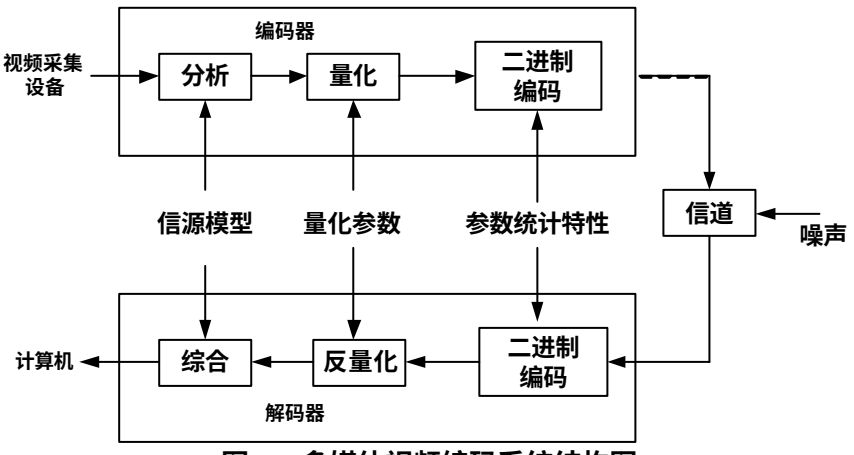

图 2.1 多媒体视频编码系统结构图

2.2 多媒体流编码标准

随着各种多媒体应用的普及，多媒体编码技术最近几年得到蓬勃发展，其中以ISO/IEC 颁布的 MPEG 系列标准和 ITU 颁布的 H 系列标准为代表，它们越来越多地考虑了以“流”为特征的网络应用需求，如视频会议、网络监控等[12-13]。

2.2.1 MPEG 系列标准

运动图像专家小组（Moving Picture Experts Group，MPEG）制定了一系列标准，包括 MPEG-1、MPEG-2、MPEG-4、MPEG-7[14]。它们也越来越多地考虑了多媒体网络传输的特征，编码对网络传输的适应性逐渐得到加强。MPEG-1 用于码率约为 1.5Mbps 的数字视频及其伴音的编码，MPEG-2 在此基础之上为了满足日益增长的多媒体分辨率和传输率等方面的技术要求，首次定义了传输流(Transport Stream)，并支持多路 MPEG-2 码流在网络中的传输复用；MPEG-4 标准旨在为多媒体的传输、存储及其应用环境提供一个基于“对象”的编码方案，并在时域和空域具有灵活的扩展性，针对网络中码率变动的特性[15-16]。MPEG-4 制定了精细可扩展性编码（Fine Granular Scalable, FGS）[17]和渐进的精细可扩展性编码（Progress Fine Granular Scalable, PFGS），以利于动态码率的调整。MPEG-7 被称作是多媒体内容描述接口(Multimedia Content Description Interface)[18]，它将不同类型的多媒体信息进行标准化的描述，并将这种描述与媒体的内容联系起来，以实现基于内容的有效检索。

下面对 MPEG1 和 MPEG4 标准的相关技术进行介绍：

（1）MPEG1

MPEG1 支持的编辑单位是图组，通过对包头图组的信息进行修改，可以完成对视频信号的剪接[19]。由于 MPEG1 的主要应用领域为视频的存储，因此采用逐行扫描，帧频率分别为 30Hz（NTSC 制式）和 25Hz（PAL 制式）。

MPEG1 采用的图像有 3 种类型：帧内编码图（intra-pictures）（I 图），预测编码图（predicted pictures）（P 图）和双向预测编码（Bidirectionally-predictive coded）图即 B 图。帧内图可提供随机存取的存取位置，但压缩比不大；帧间插补可减少时域的冗余信息。在帧间预测编码时，要用到过去的图（帧内图或预测图），当前的预

测图通常又作为后面的预测图的参考值。双向预测图的压缩效果最显著，但是它在预测时需要先前和后续的信息。

这种图的组织结构十分灵活，它们的组合可由应用规定的参数来决定，如随机存取和编码延迟。图 2.2 的图例是在沿时间轴方向的排列中，每 8 帧图像内，有 1 幅帧内图 I，1 幅预测图 P，6 幅双向预测图 B。B 图处于 I 图和 P 图之间。I，P 和 P，I 之间各包括 3 个 B 图。

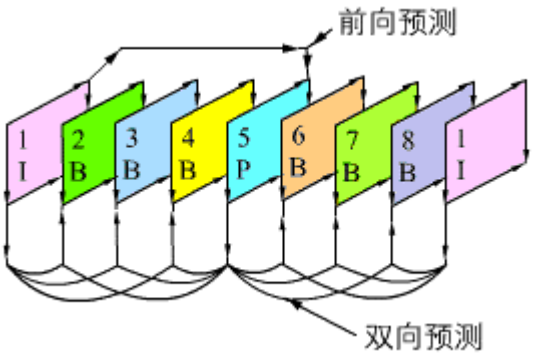

图 2.2 帧间编码

MPEG1 视频位流利用双向预测编码可以获得一个高的压缩比。在一个电视图像的帧序列中，不能全部是 B 图，B 图必须有参考图进行插补，参考图可以是帧内图 I 或预测图 P，B 图不能作为参考图。在两个参考图之间出现双向预测图 B 的频度是可选择的。当增加参考图之间 B 图的数目时，会减少 B 图和参考图之间的相关性。B 图数目的选择与被编码的图像景物具有一定的依赖性，对于大多数景物来说，参考图以大约十分之一秒的间隔隔开较为合适。如图 2.3 所示为一个视频序列中帧图的显示顺序的例子，这也是帧编码器输入帧图的排列顺序。图中第 2 行是编码器输入帧图的编号，第 1 行表示帧图属性。（I 图、P 图、B 图）。

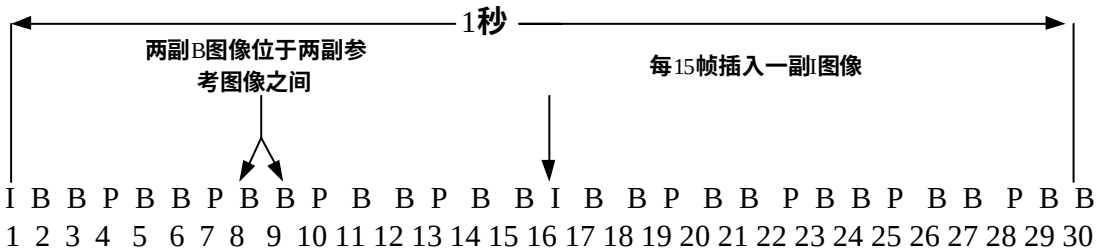

图 2.3 按图像显示次序排列的视频流

本例中共画出 30 帧图，其中包括 2 幅帧内图 I、8 幅预测图 P、20 幅双向预测图 B。在 I 参考图和 P 参考图之间有 2 幅双向预测图，在两个 P 参考图之间也有两个 B 图。由于 I 图、P 图、B 图三者之间存在一定的因果关系，如第 4 帧的 P 图是由第 1 帧的 I 图预测；第 1 帧 I 图和第 4 帧 P 图共同预测出它们之间的双向预测图 B，所以接收端解码器的输入不能按照图 2.3 所示的顺序，而是按照如图 2.4 所示的排列顺序。

1 4 2 3 7 5 6 10 8 9 13 11 12 16 14 15…
I P B B P B B P B B P B B I B B…

图 2.4 解码器输入顺序

MPEG1 视频位流是一个分层结构，目的是把位流中逻辑上独立的实体分开，防止语义模糊，并减轻解码过程的负担。MPEG1 视频位流分层结构如表 2.1 所示。

表 2.1 MPEG1 视频位流语法的六个层次

图像序列层（随机存取单元，上下文）
图组层（随机存取单元，视频编码）
图层（基本编码单元）
宏块片层（重同步单元）
宏块层（运动补偿单元）
块层（DCT单元）

其中，每一层都支持一个确定的函数或者是一个信号处理函数（DCT，MC），或者是一个逻辑函数（同步，随机存储点）等。

(2) MPEG4

活动图像专家组于 1999 年 2 月正式公布了 MPEG4 V1.0 版本，同年 12 月又公布了 MPEG4 V2.0 版本[20]。MPEG4 标准支持以下新的功能：基于内容的交互性；高效的压缩性；通用的访问性。MPEG4 标准主要针对可视电话、视频电子邮件和电子新闻等，其传输码率要求较低，在 4800～6400bps 之间，分辨率为 176×144 像素。

MPEG4 除采用变换编码、运动估计与运动补偿、量化、熵编码等第一代视频编码核心技术外，还提出一些新的有创见性的关键技术，充分利用人眼视觉特性，抓住

图像信息传输的本质，从轮廓、纹理思路出发，支持基于视觉内容的交互功能。MPEG4 标准同以前标准的最显著差别在于它采用基于对象的编码理念，即在压缩之前每个场景被定义成一幅背景图和一个或多个前景音视频对象，然后背景和前景分别进行编码，再经过复用传输到接收端，然后再对背景和前景分别解码，从而组合成所需要的音视频。图 2.5 是 MPEG4 视频编码器框图首先是视频对象（VO, Video Object）的形成，即从原始视频流中分割出 VO，然后由编码控制机制为不同的 VO 以及描述各个 VO 的 3 类信息即运动信息、形状信息和纹理信息分配码率，再将各个 VO 分别独立编码，最后将各个 VO 的码流复合成一个位流。其中，在编码控制和复合阶段可以加入用户的交互控制或智能化算法的控制。解码基本上为编码的逆过程。

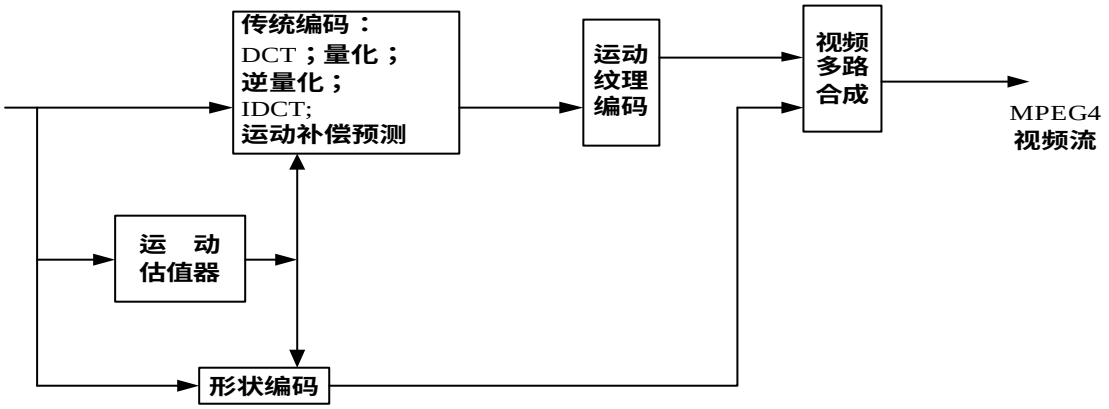

图 2.5 MPEG4 视频编码器框图

2.2.2 H 系列标准

H.261 是第一代的 H 系列标准，受远程电信经营者支持的标准[21]。H.261 只对 CIF（Common Intermediate Format）和 QCIF（Quarter CIF）两种图像格式进行处理。由于 QCIF 格式的受限分辨率以及较低的帧速率 15 或 7.5 帧每秒，实现 H.261 编码与解码器所遇到的技术问题是很少的。如果在实现中并没有应用运动补偿与低通滤波器，那么，应该不存在任何技术问题，尽管在这种情况下质量并不总让人满意。如果使用运动补偿，按每秒 25 或 30 帧速率对图像用 CIF 格式编码，那么质量就可以接受了。H.261 在网络环境中的对话模式里用得很多，如视频电话和视频会议。所达到的连续数据位率在今天利用 ISDN 与租用线的广域网应用中是非常适合的。通常 CIF 格式

用于视频会议，QCIF 格式用于可视电话。

H.261 压缩编码算法是一种采用帧间编码减少时间冗余、变换编码减少空间冗余的混合编码方法，具有压缩比高、算法复杂度低等优点。由于会话双方都需要同样的编码器和解码器，故 H.261 标准的一个特点是编、解码器的复杂程度相当。H.261 仅使用 I 帧和 P 帧，格式为每 1 对 I 帧之间有 3 个 P 帧。在 I 帧和 P 帧中每 6 个 8×8 的像素块(Block)构成一个宏块(Macroblock)，其中包括 4 个亮度(Y)块和 2 个色度(C_b 和 C_r)块，每个宏块都会有一个专门的地址来标识宏块本身，另外还会有一个类字段，用来说明该宏块是独立编码（帧内编码），还是参考前一帧内的宏块进行了帧间编码。一定数量的宏块构成一个块组 GOB(Group of Block)，若干块组构成一帧图像。块层、宏块层、块组层和帧层 4 个层次中每个层次都有说明该层次信息的头（Header），编码后的数据和头信息逐层复用就构成了 H.261 的码流。

除了 H.261 外，ITU-T 陆续发布了 H.262，H.263，H.264 标准。第二代标准的代表 H.264 标准保留了一些先前标准的特点，同时使用了以下视频编码新技术：

（1）熵编码

若是 Slice 层预测残差，H.264 标准采用两种高性能的熵编码方式：基于上下文的自适应变长码（CAVLC）和基于上下文的自适应二进制算术编码（CABAC）。仿真测试结果得到 CABAC 比 CAVLC 压缩率高 10%，但计算复杂度也高。若不是 Slice 层预测残差，H.264 采用 Exp-Golomb 码或 CABAC，视编码器的设置而定。在 CAVLC 中，H.264 采用若干 VLC 码表，不同的码表对应不同的概率模型。编码器能够根据上下文，如周围块的非零系数或系数的绝对值大小，在这些码表中自动的选择，最大可能的与当前数据的概率模型匹配，从而实现上下文自适应的功能。

CABAC 根据过去的观测内容，选择适当的上下文模型提供数据符号的条件概率的估计，并根据编码时数据符号的比特数出现的频率动态地修改概率模型，数据符号可以近似熵率进行编码，提高编码效率。

（2）帧内预测

H.264 采用帧内预测。帧内预测编码具有运算速度快、提高压缩效率的优点。帧内预测编码就是用周围邻近的像素值来预测当前的像素值，然后对预测误差进行编

码。这种预测是基于块的，对于亮度分量，块的大小可以在 4×4 和 16×16 之间选择，4×4 块的预测模式有 9 种（模式 0 到模式 8，其中模式 2 是 DC 预测），16×16 块的预测模式有 4 种（vertical, horizontal, DC, plane）；对于色度分量（chroma），预测是对整个 8×8 块进行的，有 4 种预测模式（vertical, horizontal, DC, plane）。除了 DC 预测外，其他每种预测模式对应不同方向上的预测。

（3）帧间预测

H.264 采用 7 种树型宏块结构作为帧间预测的基本单元，每种结构模式下块的大小和形状都不相同，这样更有利于贴近实际，实现最佳的块匹配，提高了运动补偿精度。在 H.264 中，亮度分量的运动矢量使用 1/4 像素精度，色度分量的运动矢量使用 1/8 像素精度，并详细定义了相应更小分数像素的插值实现算法。因此，帧间运动矢量估值精度的提高，使搜索到的最佳匹配点（块或宏块中心）尽可能接近原图，减小了运动估计的残差，提高了运动视频的时域压缩效率。H.264 支持多参考帧预测，即通过在当前帧之前解码的多个参考帧中进行运动搜索，寻找出当前编码块或宏块的最佳匹配。在出现复杂形状和纹理的物体、快速变化的景物，物体互相遮挡或摄像机快速的场景切换等一些特定情况下，多参考帧的使用会体现更好的时域压缩效果。

（4）灵活的宏块排序

灵活的宏块排序（FMO），指将一幅图像中的宏块分成几个组，分别独立编码，某一组中的宏块不一定是在常规的扫描顺序下前后连续，而可能是随机地分散在图像中的不同位置。这样在传输时如果发生错误，某个组中的某些宏块不能正确解码时，解码器仍然可以根据图像的空间相关性依靠其周围正确译码的像素对其进行恢复。

（5）4×4 块的整数变换

H.264 对帧内或帧间预测的残差进行 DCT 变换编码。为了克服浮点运算带来的硬件设计复杂，新标准对 DCT 定义作了修改，使用变换仅使用整数加减法和移位操作即可实现。这样，在不考虑量化影响的情况下，解码端的输出可以准确地恢复编码端的输入。该变换是针对 4×4 块进行的，也有助于减少块效应。为了进一步利用图像的空间相关性，在对 chroma 的预测残差和 16×16 帧内预测的预测残差进行整数 DCT 变换后，标准还将每个 4×4 变换系数块中的 DC 系数组成 2×2 或 4×4 大

小的块，进一步做 Hadamard 变换。 与 H.263 中 8×8 的 DCT 相比，H.264 的 4×4 的 DCT 有以下几个优点：减少了方块效应，算法简单明了，运算结果精度高，运算速度快，占用内存少。

H 系列，MPEG 系列是数据压缩的不同标准，它们的目标不同，作用互补，大多数算法十分相似但并不完全相同。压缩质量和系统的可用性决定了哪种技术能用于未来的多媒体系统并将导致多种技术的融合。

2.3 多媒体流应用特点

多媒体流应用是目前多媒体最重要的应用形式，如视频会议、网络监控等。本节通过一个例子来分析多媒体流应用的特点。

2.3.1 交互式远程教学系统的例子

交互式远程教学系统在教学中被广泛应用，它可以称作电子教室，由电子白板、音频系统和视频系统组成[22]。每个学生节点上有耳机、麦克风、数字摄像机和监视器。麦克风采集音频，耳机输出音频，数字摄像机采集视频图像，监视器输出视频和白板信息。音频和视频经MPEG压缩后，同白板信息实时地在分布式节点之间传输，视频信息处理的进程通常是周期性的，但由于采用MPEG压缩，视频帧的数据大小是动态变化的，所以每个周期的执行时间也是不一样。交互式远程教学系统结构如图2.6：

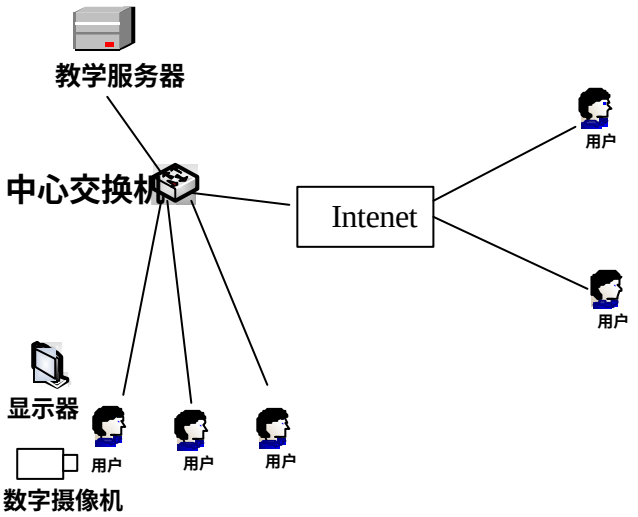

图2.6 交互式远程教学系统结构图

2.3.2 多媒体流应用特点分析

通过上面的例子，我们可以看出多媒体流应用有以下四个特点：

(1)实时性：多媒体是与时间有关的数据，其正确性包括内容正确性和时间正确性两方面，故对多媒体的处理必须满足一定的时限要求，同时，多媒体又具有时间相关性，每个信息单元都要求在一定的时间限度内处理完毕，否则就没有意义。

(2)数据量大：多媒体流应用需要处理音频和视频等信息，即使采用压缩技术，这些信息仍有相当大的数据量。例如，采用MPEG-1标准压缩的VHS质量和HDTV质量的视频流数据量分别为约1.2Mbps和20Mbps，而未经压缩的HDTV视频流数据量则高达1.2Gbps。表2.2给出了采用几种压缩标准压缩后多媒体的带宽要求：

表 2.2 几种压缩标准对带宽的要求

压缩标准	压缩比	压缩后的带宽要求(Mbps)
H.261	24:1	0.064-2
MPEG1	100:1	1.2-2
MPEG2	20-100:1	4-60

(3)周期性：多媒体流是由一系列时间相关的信息单元(information units)组成，如MPEG视频流就是由视频帧(frame)组成。这些信息单元是对源媒体进行周期性采样、量化及压缩而得来，从而其数据在时间上呈现周期性。如本例中的视频采集以及演播任务可以描述为周期为40ms的常数周期任务。

(4)突发性：压缩能大大减小多媒体流的数据量，但同时也使得压缩后的多媒体流具有了突发性。这是由媒体内容、压缩方式等因素引起的，且这种突发性往往难以预测。以 MPEG 视频压缩标准为例，如图 2.7，压缩后的视频流由三种类型的帧组成：I 帧、P 帧和 B 帧。I 帧为帧内压缩帧，数据量最大；P 帧为前向预测压缩帧，数据量其次；B 帧为双向预测压缩帧，数据量最小。三种帧组成一定模式的图组（如 IBBPBBPBBPBBI），再以图组为单位组成 MPEG 压缩视频流。从整体上看，压缩视频流的数据量的变化与图组的模式相关，呈现一定的规律性；但由于各个画面的压缩比不同，即使相同的帧类型，压缩后的数据量也相差很大，再加上场景变换、影片内

容等因素的影响，其突发性更是难以预测。

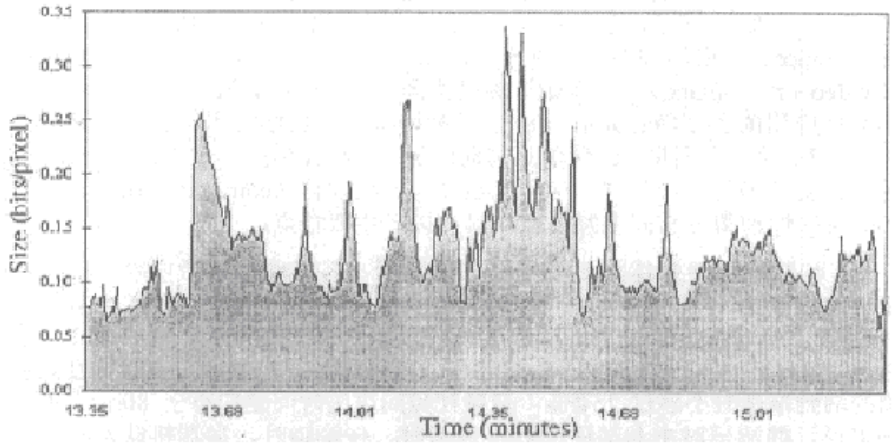

图 2.7 MPEG 压缩视频流示例

2.4 本章小结

本章简要介绍了多媒体编码系统，并对目前主要的多媒体流编码标准 MPEG 和 H 系列进行了分析，进而指出由于压缩编码导致的多媒体流任务周期性和突发性的特点，为后面的内容提供必要的技术知识。

3　常见多媒体流任务实时调度算法

CPU是多媒体系统中最重要的核心资源之一，好的CPU任务调度算法是满足多媒体应用实时要求的关键，目标是控制CPU任务的死线丢失率的同时保证CPU资源利用率。本章将分析一些常见多媒体流任务CPU实时调度方式，并指出存在的问题。

3.1　多媒体流任务实时调度策略

通过上文的介绍，我们知道传统分时系统 CPU 调度的目标是尽可能使吞吐量和资源利用率达到最优及公平性，不能满足多媒体任务的实时性要求。简单地在分时调度中增加实时优先级，如 SVR4 UNIX，WINDOWS NT/2000/XP 等系统，对于支持多媒体应用都存在极大的弊病，高优先级的任务容易恶意地独占 CPU，引起其它任务饿死甚至系统崩溃，所以必须寻找新的调度方式。一条多媒体流由一系列时间相关的信息单元组成，如 MPEG 视频流由视频帧组成。这些信息单元是对源媒体进行周期性采样、量化及压缩而得来，从而其数据在时间上呈现周期性。同时，多媒体流又具有时间相关性，每个信息单元都要求在一定的时间限度内处理完毕，否则就没有意义。因此，处理多媒体流的任务 T 通常可以用周期性实时任务来描述，利用现有的实时调度模型：T= (r, p, c, d) [23]。其中 r 表示任务的起始执行时间；p 表示任务的周期，任务在一个周期内的一次执行称为该任务的一个任务实例；c 表示每个任务实例的执行时间，由于源媒体的内容随时间变化，不同的内容有不同压缩比，故各个信息单元的大小不同，对每个信息单元的处理时间也不同，即不同的实例有不同的 c；d 表示任务的相对时间期限，通常成为相对死线。对于多媒体流任务来说一般有 d=p，故第 i 个实例必须在时刻 r＋ip 之前完成，r＋ip 称为该实例的绝对死线，如图 3.1 所示。

图3.1 多媒体流任务死线示意图

下面，对几种主要的多媒体流 CPU 调度方式进行分析，并指出存在的问题，和目前的研究方向。

3.2 基于传统周期模型的调度方式

3.2.1 经典周期模型

(1) 经典周期模型定义：基于周期调度的一个周期性应用用一个四元组表示：T(r,c,p,d) [24]。

r：任务 T 的到达时间；

c：任务 T 的运行时间；

d：任务 T 的死线(Deadline)；

p：任务 T 的周期，即相邻任务到达时间间隔。

该模型在实时领域被广泛应用，通过它的各种扩展，表征了很多传统实时应用的特征，如数字控制、实时监视等。基于此模型的很多调度算法具有很好的性能和容易理解，如实时经典调度算法单调比率算法 RM(Rate Monotonic) [25]、最早死线完成算法 EDF(Earliest Deadline First) [26]以及单调死线算法 DM(Deadline Monotonic)。

可调度测试条件

(2) 可调度测试条件

可调度测试条件存在于很多经典实时周期调度算法当中，是算法进行调度必须满足的条件。符合条件时，系统中所有请求都会在规定时限内完成。

下面给出 RM 和 EDF 算法的可调度性测试条件[25]：设完全可剥夺的周期性任务集合 $T=\{T_{i=}(r_{i,}c_{i,}p_{i,}d_{i})\,|\,i=1,2,...n\}$ 满足条件①-⑤：

① 所有任务均为周期确定的周期性任务；

② 任务的相对死线与等于任务周期，即 $d_i = p_{i,}(i=1,2,...n)$ ；

③ 所有任务相互独立；

④ 每个任务的运行时间为一常量；

⑤ 系统中的非周期性任务没有严格的时限要求。

则，若满足：

$$\frac{c_1}{p_1}+\frac{c_2}{p_2}+...+\frac{c_n}{p_n}\le n(2^{1/n}-1) \tag{3.1}$$

则该任务集采用RM算法可调度；

若满足：

$$\frac{c_1}{p_1}+\frac{c_2}{p_2}+...+\frac{c_n}{p_n}\le 1 \tag{3.2}$$

则采用 EDF 算法可调度。

为了确保任务的实时性，设计者可以通过选择任务参数使之符合周期模型。将上述可调度性测试条件作为任务进入系统时的接入控制，可保证通过测试的任务都不会丢失死线。接入控制实际上是通过为任务预留适当带宽来实现的，例如为任务T预留的CPU带宽比例为 c_i / p_i。$\sum_{i=1}^{n}\frac{c_i}{p_i}$ 为系统CPU利用率，当n接近无穷大时，$\lim_{n\to\infty}\left[n(2^{1/n}-1)\right]=\ln(2)\approx 0.693$，也就是说只要CPU利用率小于69%，满足周期模型的实时任务集由RM算法调度都不会丢失死线。这个定理经常被应用到调度分析测试当中[27]。

(3) 经典周期调度算法RM、EDF简介

下面介绍EDF算法和RM算法的基本情况[28]：

① EDF算法在实时系统广泛使用。调度进程在每个新的调度准备状态时，在调度任务集合{T}中选择一个死线 d_i 最近的任务 T_i 分配CPU运行。当一个新任务 T_j 到达时，调度进程要重新计算任务集的死线序列，如果新任务 T_j 的死线 d_j 比正在运行任务 T_i 的死线 d_i 短，那么新任务 T_j 就抢占正在运行的任务 T_i 运行，被抢占的任务 T_i 重新排队。可见EDF算法的调度是依据各任务的死线来调度的，是抢占式的算法。它可以

用于周期性任务调度，也可以用于非周期性随机任务插入的调度。

② RM算法基于速率的调度，是按优先级调度周期性任务的抢占式算法。调度进程调度任务集合{T}时，事先就安排了任务的调度时序。任务集中每个任务在开始就由调度进程按着周期赋予了一个优先级，周期越小，优先级越高。即如果对于任T_i，T_j，$p_i < p_j$，则$pr_i < pr_j$，pr是它们的优先级。

3.2.2 基于周期模型的调度方式

(1) 基本思想

根据多媒体流周期性的特性，将处理多媒体流的任务作为传统的周期性任务，根据传统周期调度模型采用 RM 或 EDF 算法及相应的可调度性测试条件进行调度和接入控制。

(2) 性能分析

如果将多媒体流任务直接当作周期性任务处理，因为传统的周期性任务模型及 RM、EDF 算法的可调度性测试中都假设任务在每个周期中的运行时c_i为一常量，但多媒体流的另一个重要特点突发性使得任务各周期内的运行时间变化很大且难以预测。这样，就无法准确的满足周期模型可调度测试条件。

针对这种情况，通常采用两种解决方法：悲观方法和乐观方法[29]。悲观方法中，c_i取最大可能的运行时间$\hat{c}_i$，从而能为所有任务提供硬实时保证。即为每个任务预留 CPU 带宽$\hat{c}_i / p_i$，但是显然这种办法将浪费大量的 CPU 资源，无法提高 CPU 资源利用率。

在乐观方法中，c_i取运行时间的平均值$\bar{c}_i$，这对于运行时间超过平均值的任务，将无法在规定时间内完成，导致在许多情况下丢失死线，使得丢失死线率过高，从而无法提供实时保证。

因此，直接利用经典周期模型，将多媒体流任务当作传统周期性任务处理，不能很好地反映多媒体流任务的特点，必须研究新的调度方法。

3.3 启发式调度算法

另一种是采用启发式调度算法，针对多媒体流应用的特点，对经典的实时调度算法进行改进，以较好地支持多媒体流应用。在文献[30]中给出了这样一种启发式调度算法。

3.3.1 基本思想

在多媒体流服务器上，对每条视频或音频流的处理都可以认为是一个任务，即一个可被调度的实体，并把每条媒体流的处理当做周期性任务，描述为(C,P)。C 为任务的执行时间，P 为任务的执行周期。如视频流的处理速度应达到 30 帧/秒，假设把每帧的处理作为该任务的一个实例，每个实例的处理时间相当于有了约束，需在 33ms 内完成而且；视频流的处理可视为由执行周期为 33ms 的任务序列构成。把在约束时间内完成的实例称为实例成功，否则称为实例失败。

对于每条媒体流的服务质量的控制，用 3 个参数来描述：S，Q，F，即在对 S 个连续实例处理中，至少要成功完成 Q 个，而且连续失败的实例数不能超过 F 个。称 Q/S 为任务的最小成功率，称 F 为任务允许的最大连续失败数。如果服务器调度导致某条流 S 个连续实例中多于 S－Q 个实例失败，或某个失败的实例连续的出现次数超过 F，则任务调度没有满足媒体流的实时要求。对于周期性任务集 $\Gamma=(c_i, p_i)$，i=1,2,…,n，采用 EDF 调度算法，如果 CPU 的总利用率 $\sum_{i=1}^{n}\frac{c_i}{p_i}\leq 1$，那么该任务集可被正确调度。但当任务集的 CPU 利用率超过 1 时，若再采用 EDF 调度算法，结果将无法预测。

在多媒体应用系统中，像视频流和音频流，只要满足一定的任务死线丢失率，就认为是可接受的。因此当系统负载较大时，可在死线丢失率允许的范围内，通过适当降低某些任务实时性要求，来综合保证系统中所有任务的平均死线丢失率。

3.3.2 任务模型

假设系统中有 n 个强实时任务和 m 个多媒体流任务运行于一台单 CPU 服务器上，任务模型如下：

系统中强实时任务的描述：

$$\Gamma \text{ rt}=\{\ T_i^{rt} \mid\ T_i^{rt}=(r_i, c_i, p_i), i=1,2,\cdots,n\}$$

系统中多媒体流任务的描述：

$$\Gamma \text{ mt}=\{T_i^{mt} \mid T_i^{mt}=(r_i, c_i, p_i, S_i, Q_i, F_i), i=1,2,\cdots,m\}$$

任务都定义为周期性执行。其中，r_i 为任务 T_i 的初始释放时间：c_i 为任务 T_i 最坏情况下(worst-case)的执行时间；p_i 为任务 T_i 的执行周期。T_{ij} 为任务 T_i 的第 j 个实例，T_{ij} 处理媒体流的某个数据单位，如视频流中的一帧。由于目前通常使用的数据压缩技术，T_{ij} 的处理时间可能与 $T_{i,k}$ 不同(j≠k).因此，T_{ij} 的释放时间可定义为 r_i+(j-1)× p_i，完成时间期限(deadline)可定义为 r_i+j× p_i。如果 T_{ij} 能在 deadline 之前正确完成，则认为成功，否则认为失败。S_i，Q_i 的意义为在任务 T_i 连续执行的 S_i 个实例中，至少要成功 Q_i 个，因此 Q_i / S_i 为任务 T_i 的成功率；F_i 为任务 T_i 允许的最大连续失败任务数。多媒体流任务的服务质量是由参数 S_i，Q_i 和 F_i 来控制，也就是说对多媒体流任务的调度必须满足 Q_i / S_i 的成功率，而且要防止出现 F_i 个任务连续调度失败。参数 F_i 的引入避免了只用成功率来描述多媒体流任务服务质量的片面性。

3.3.3 调度算法

算法的基本思想是当系统负载较大时，为避免少数任务连续失败，而将可能的任务失败分布于所有的任务中，以尽量保证任务平均死线丢失率。

下面讨论如何将多媒体流的服务质量描述映射为其任务执行的优先级。采用启发式的优先级赋值方法，用函数 $H(T)$ 为每个媒体流任务计算优先级。$H(T)$ 要考虑任务

执行的成功率，还要避免任务连续失败的情况，因此在 $H(T)$ 中要有参数表示上述内容。$H(T)$ 的定义如下：

$$H(T_i)=a \times \frac{f(T_i)/S_i}{1-Q_i/S_i}+b\times(q(T_i)/F_i+g\times\prod(T_i)\ ,\text{i=1,}\cdots\text{,m} \tag{3.3}$$

其中 $q(T_i)$ 表示在 T_i 中最近已经连续出现的实例失败数；$f(T_i)$ 表示 T_i 最近 S_i-1 个实例中总共失败的次数，$f(T_i)$ 的统计可通过移位寄存器或软件移位来实现。$\prod(T_i)$ 表示任务 T_i 即将执行实例的重要性，如在视频流中 I 帧的处理应比 P 帧和 B 帧的优先级高，$\prod(T_i)$ 的取值范围在 0 与 1 之间。a , b , g 为调节系数，可根据需要调整。a 与 b 的比值代表了任务的成功率与连续失败数在优先级赋值中的重要性。我们定义 $H(T_i)$ 值越大，表明为当前赋给 T_i 的某个实例的优先级越高。$H(T_i)$ 的计算公式相对地说明了某条媒体流任务当前实例的重要性，也充分地体现了对多媒体流任务的实时控制。如果有多个任务计算得到的 $H(T)$ 值相同，则采用 EDF 算法对它们进行内部的优先级排序。系统的任务调度模型如图 3.4 中所示。

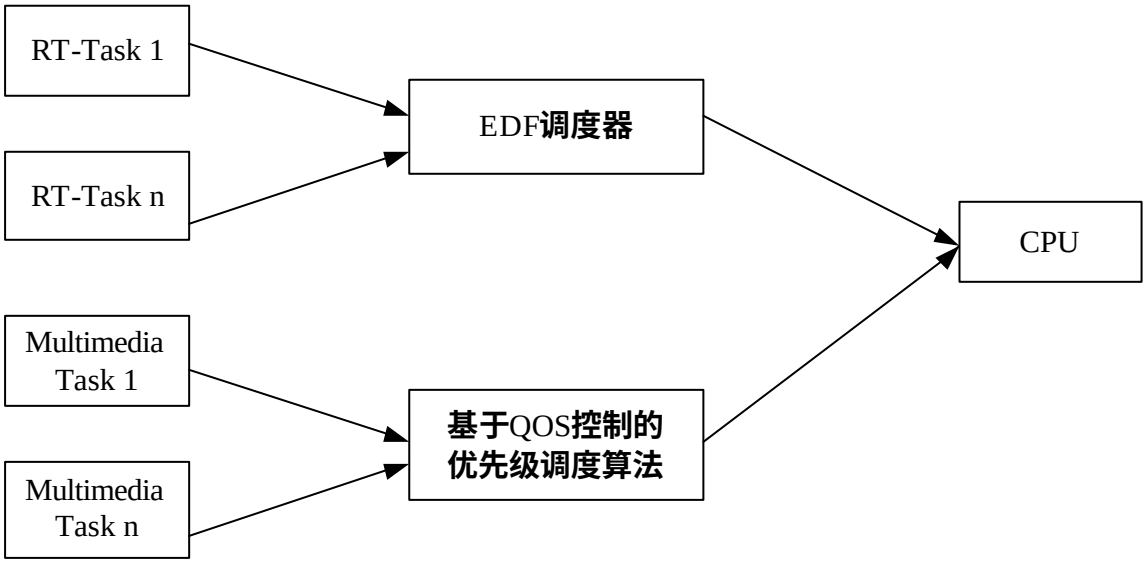

图 3.2 任务调度模型

3.3.4 性能分析

我们通过模拟对基于QoS控制的调度算法进行评价，分别与分时调度算法和EDF调度算法进行比较。

（1）与分时调度算法的比较

模拟设置的参数包括服务器的平均负载和每条流的 QoS 参数，通过模拟求得多媒体流死线丢失率，并与分时调度算法进行比较。

模拟中，连续媒体流任务数总共为15个。为了方便，所有媒体流的参数设置相同：$a=b=1$，S=30，Q=24，F=4。C_i=2~8ms($1\leq i\leq 15$)，p_i=25ms~35ms($1\leq i\leq 15$)。分时调度算法的时间片设为1ms。算法模拟与比较的结果如图3.3所示。

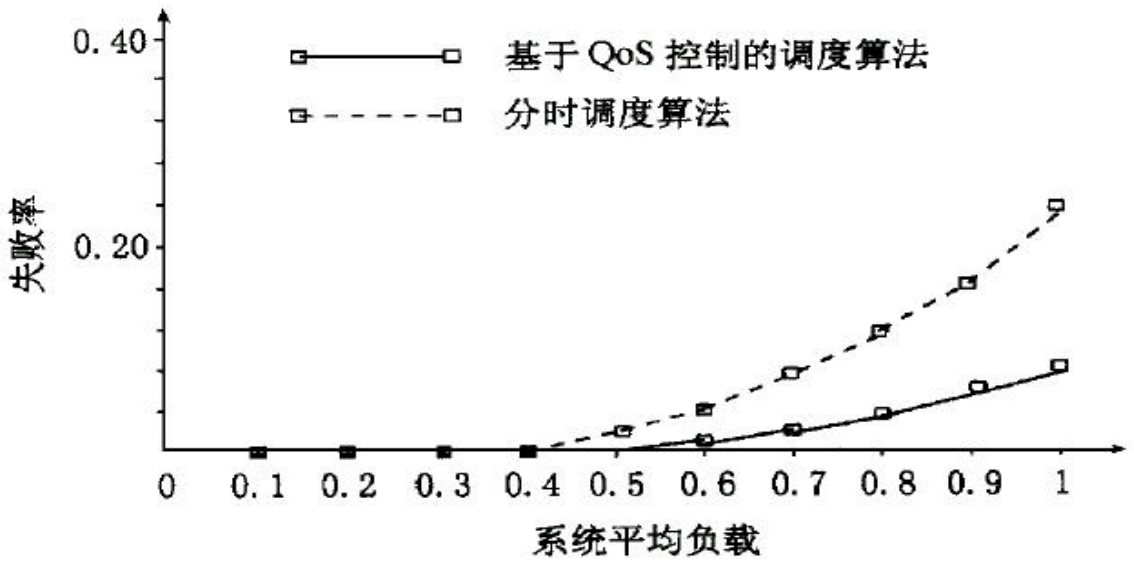

图3.3 算法模拟结果1

从图3.3所知，与普通分时调度算法相比，基于QoS的调度算法更适合于处理连续媒体流任务。尤其当系统负载较大时，算法调度任务的失败率低，能较好的满足多媒体任务实时性要求。

（2）与EDF算法的比较

模拟中，连续媒体流任务数总共为10个。所有媒体流的QoS参数设置相同的值。那么，T_{ij}的时间期限(deadline)为 j× p_i，EDF 算法根据此 deadline 来调度。假设系统

中所有任务的初始释放时间为 0。图 3.4 中纵坐标表示任务被调度的成功率，横坐标为系统中的任务编号。图 3.5 中的纵坐标表示任务连续失败次数超过设置参数的概率。根据这两个图的模拟结果，EDF 算法不能调度编号为 4，7，10 的任务满足期望的参数。而基于 QoS 控制的调度算法能满足所有任务的参数。

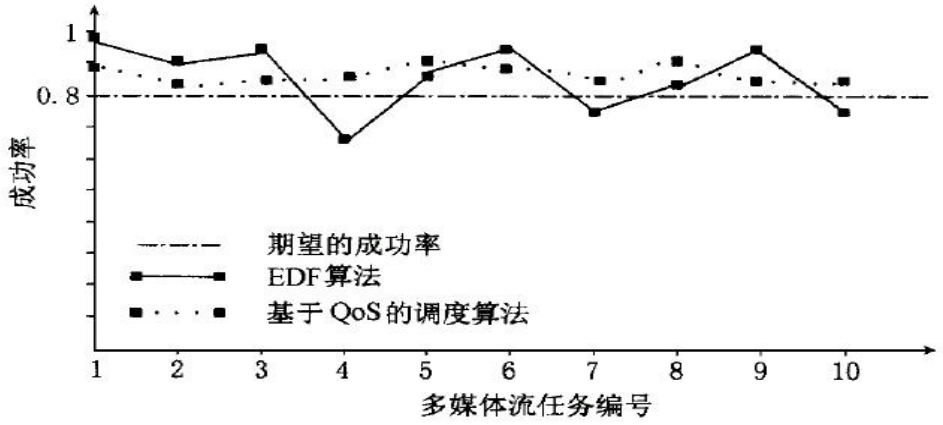

图 3.4 算法模拟结果 2

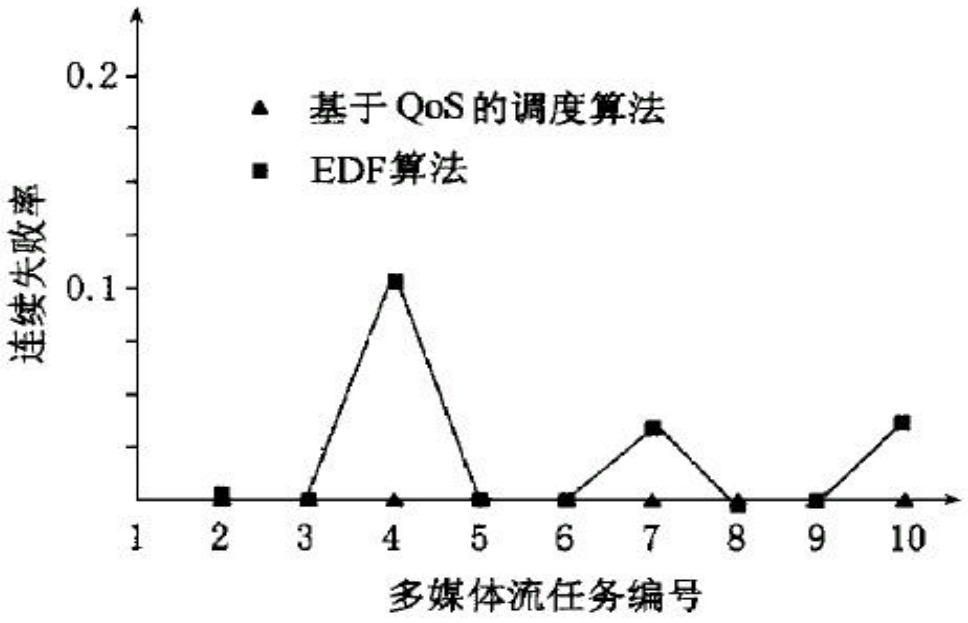

图 3.5 算法模拟结果 3

上述方法采用启发式调度对传统实时调度算法进行改进，提出通过任务的最小成功率和允许的最大连续失败数来刻画多媒体流任务的服务质量，从而提出调度算法。

其基本思想是当系统负载较大时，为避免少数任务连续失败，而将可能的任务失败分布于所有的任务中，避免了某些媒体流任务因连续不能被正确调度执行而导致的服务质量下降，保证了所有任务的实时性，从而较好的支持多媒体流。但是这类启发式算法没有准确的可调度性测试条件，无法预约 CPU 资源，不能提供可预测的实时性保证。

3.4 目前的研究方向

综上所述，为能研究出一种既保证实时性，又提供较好资源利用率的多媒体流任务调度方式，人们开始对多媒体流任务 (运行时间可变的周期性任务)寻求新的任务模型。新的任务模型有两个目标，一是能较好地反映多媒体流任务的特性，二是能充分利用现有的实时调度理论。

针对多媒体流任务运行时间大部分都小于最大运行时间的情况，文献[31]中作者提出了周期多帧任务模型及此模型下的可调度性测试条件。对于在每个周期内运行时间不同的多媒体流任务，周期多帧模型比传统的周期模型更贴近于实际情况。该模型中，运行时间不再是单一的最大运行时间，而是由几个运行时间值组成的一个运行时间向量。基于多帧任务模型的系统一般采用固定优先级调度策略，如静态优先级调度算法DM[32]。实验证明该方法能提高系统资源利用率。但当调度的任务集合中某一任务的运行时间变化较小时，系统利用率就会迅速下降到与悲观方法相近；而任务的每个实例的执行时间发生较大的变化时，又可能导致大部分的任务不能在规定的期限内完成。

在下一章本文借助于任务转换的思想，提出一种新的任务模型来支持多媒体流任务调度。

3.5 本章小结

本章主要针对多媒体流任务周期性和突发性的特点，引入了实时调度策略。并对

常见的多媒体流实时调度算法进行了分析，指出它们存在的问题以及目前的研究方向。在下一章，本文会提出一种新的多媒体流任务模型以及相应的调度策略，来支持多媒体流应用。

4　一种基于单调比率算法的多媒体流混合调度策略

本章将提出任务转换思想，并在此基础上建立多媒体流任务模型以及相应的调度策略，并给出相应的接入控制算法和统计保证理论分析，最后对该算法进行仿真实验。

4.1　任务转换思想

4.1.1 不规则周期任务的概念

多媒体流任务的突发性使其不符合传统周期任务模型。虽然这类应用的所有请求都是周期性地被释放，但是它们的运行时间变化很大。这种变化可能是因为在某个周期处理的数据量变化比较大；或者，不同的请求执行了不同条件分支，这些都是导致运行时间变化大的原因。

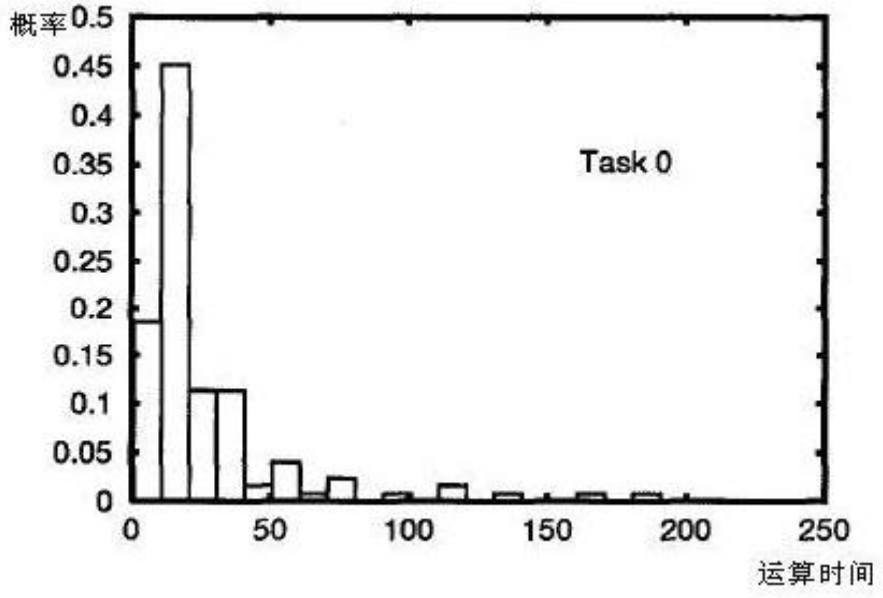

图 4.1　多媒体流任务的运算时间柱状图

上图给出了一个多媒体流任务的运行时间柱状图。在任务执行过程中，所有请求的最大运行时间显然远远大于任务的平均运行时间，如果想通过周期性任务模型来给

这些任务建模，那么每一个任务的资源利用率(最大运行时间除以周期)可能大于 1。根据已有的传统周期模型调度测试条件，即使它的平均资源利用率很小，也不能保证这样的任务能被调度。

我们将上图列出的多媒体流任务称为不规则周期任务[33]。这类任务的特点：任务请求被周期性释放，但每一个请求的运行时间是随机变化的。这也正是多媒体流应用周期性和突发性并存的特点。就像任务的其它参数一样，任务请求运行时间是随机分布变量，遵循某个概率分布函数。

4.1.2 任务转换思想

我们下面将要讨论如何扩展目前已有的传统周期任务调度分析方法，来为不规则周期任务提供概率性调度保证。通过概率性的性能保证，设计者可以选择占用较少的 CPU 资源或者保证所有任务都能够在死限内完成。

任务转换的基本思想：采取任务转换的方式，将每一个不规则周期任务转换为一个周期性任务和一个突发性任务之和。周期性任务根据已知的调度测试条件，采用固定优先级的实时调度算法来调度；利用零星服务器算法调度突发性任务。

假设在某个系统中，有 N 个各自独立的不规则周期任务的任务集合要由一个处理机来调度。每一个不规则周期任务有参数周期 p_i 和相对硬死线 d_i。T_{ij} 表示任务 T_i 的第 j 个请求，T_{ij} 的运行时间是 c_{ij}。对给定的 i，c_{ij}，j＝1,2,…根据概率密度函数 $f_i(t)$ 均匀分布。对所有的任务 T_i，参数 p_i，d_i，以及密度函数 $f_i(t)$ 在系统开始运行以前都是已知的。而且，我们假设每一个请求 T_{ij} 在被释放之后，运行时间已知。令 $r_{i,j}$ 是 T_{ij} 的释放时间，且 $d_{i,j} = r_{ij} + d_i$ 是它的绝对硬死线。

任务 T_i 的 CPU 资源平均利用率用 u_i 表示，也就是任务 T_i 拥有的处理器有效时间片。假设条件：我们分解每一个不规则周期任务 T_i 为一个周期任务 P_i 和一个突发性任务 S_i。周期性任务 P_i 的周期是 p_i，运行时间是 c_i。c_i 由设计者选择；范围是 $(0, \hat{c}_i)$，

如图 4.2。P_i 的第 j 个请求 $P_{i,j}$ 在 $r_{i,j}$ 时刻被释放，同时 $r_{i,j}$ 也是是 T_i 的第 j 个请求的释放时间。如果 $T_{i,j}$ 的运行时间 $c_{i,j}$ 大于 c_i ，当 c_i 完成后，S_i 的一个请求到达并且会准备执行。因此，在 T_i 的每一个周期中，S_i 中的每一个突发性请求服务的概率 A_i 等于 $f_i(t)$ 函数下方的阴影部分，我们总是选择合适的 c_i 和 A_i ，以便于所有的周期任务 P_i (1<i<N) 可调度。

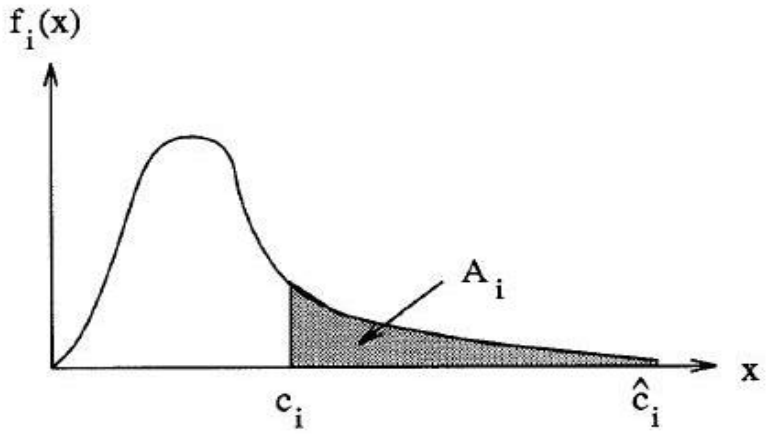

图 4.2 任务 T_i 中 S_i 请求到达的概率密度函数

转换后得到的周期性任务 P_i 和不规则周期任务 T_i 有相同的周期。c_i 是 P_i 的最大运行时间，P_i 的相对死线是 $a_i\ d_i$ ，0< a_i <1。 a_i 和 c_i 要选择确保 P_i 所有的请求都能满足死线。T_i 突发性任务 S_i 请求 $S_{i,j}$ 在第 j 个周期提出请求当且仅当请求 T_{ij} 的运行时间大于 c_i 。这个突发性请求在对应的周期性请求 $P_{i,j}$ 完成时到达。它的运行时间根据概率密度函数分布：

$$G_i(x)=\begin{cases} o & \text{for x< 0} \\ f_i(x+c_i)/A_i & \text{for x} \geq 0 \end{cases} \qquad (4.1)$$

在 T_i 的第 j 个周期当中，突发性任务的到达概率 A_i 等于 $c_{i,j} > c_i$ 的概率。突发性任

务的相对死线S_i是（1－a_i）d_i。这里为了能够对突发性任务提供概率性的保障，我们也指定它的周期也是p_i。

4.2 任务模型的建立

根据上节的思想，本节建立新的多媒体流任务模型。为方便描述，将传统意义上的周期性任务(运行时间不变)称为周期性任务[34]。而将运行时间可变的周期性任务称为不规则周期性任务。多媒体流任务可用不规则周期性模型来描述。

定义1.周期性任务P_i定义为四元组$<r_i, c_i, p_i, d_i>$，r_i表示任务的准备好的时间，c_i表示任务周期内的运行时间，P_i表示任务的周期，d_i表示任务的相对死线(deadline)。

定义 2.不规则周期性任务T_i定义为五元组$<r_i, c_i, c_i^m, p_i, d_i>$，$r_i$表示任务准备好的时间，$c_i$和$c_i^m$分别表示任务周期内的平均运行时间和最大运行时间，$p_i$表示任务的周期，$d_i$表示任务的相对死线。

多媒体流任务同时具有周期性和运行时间不确定的特点，因此多媒体流任务可表示为不规则周期性任务。

定义 3.突发性任务S_i由一系列任务实例的运行请求组成，定义为四元组$<r_i, c_i, p_i, d_i>$，r_i表示任务的准备好时间，c_i表示每次请求的最大运行时间，p_i表示两次请求的最小时间间隔，d_i表示任务的相对死线。我们提出的不规则周期任务模型定义如下：

定义 4.不规则周期任务模型是指不规则周期性任务$T_i = <r_i, c_i, c_i^m, p_i, d_i>$可以用一个周期性任务$P_i = <r_i, c_i, p_i, d_i>$和一个突发性任务$S_i = <r_i + p_i, c_i^m - c_i, p_i, d_i>$来替代。(如图 4.3 所示)。

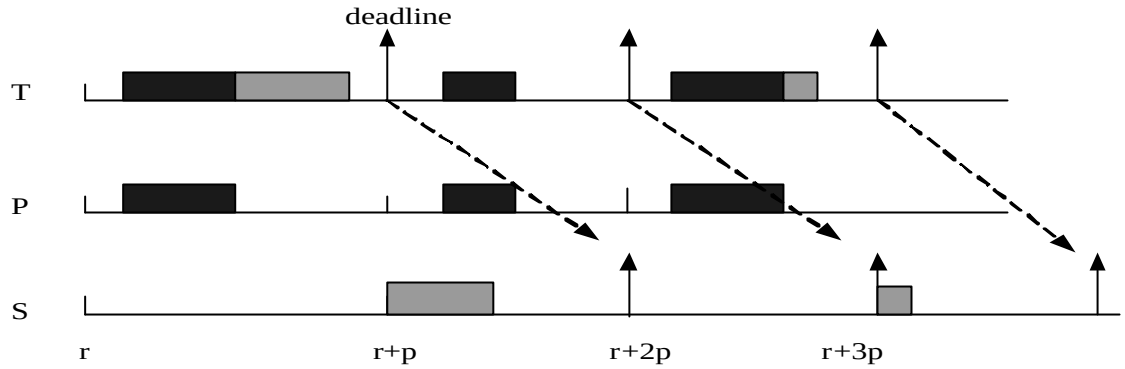

图 4.3 不规则周期任务模型示意图

不规则周期任务模型基于多媒体流任务的两个特性：多媒体流任务个各任务实例潜在的独立性，及其对启动延时容忍度大大而且流性非常敏感的特性。根据第一个特性，在任务转换时将 S_i 的准备好时间向后推延一个周期，从而在调度时不必考虑分解带来的任务相关性，得到了标准的周期任务和突发性任务，以充分利用各种成熟的实时调度算法；而由第二个特性，该模型带来的一个周期的附加延时也是可以容忍的。

4.3 调度方法

经不规则周期任务模型处理所得的任务集合可以采用任何混合调度周期性任务和突发性任务的调度方法来调度。为了保证多媒体任务的实时性，要求不仅能对周期性任务提供硬实时保证，还应能为突发性任务提供统计保证，即保证突发性任务丢失死线的概率不超过某一上限值。我们采用 RM 调度处理周期性任务，而采用零星服务器(sporadic server)来处理突发性任务，并着重分析基于不规则周期任务模型的统计保证理论分析和接入控制算法。

4.3.1 零星服务器算法简介

零星服务器算法采用一个称为服务器(server)的专门的周期性任务来处理非周期性任务的请求，server和其它周期性任务一起通过可调度性测试来预留带宽，然后再调度其它的突发性任务或非周期性任务在自己的运行时间中运行，从而避免其它任务对周期性任务的影响。这种调度方式的目标是在保证周期性任务不丢失死线的情况下尽可能地提高非周期性任务的响应时间。Sporadic Server算法中server与其他周期性任

务一起采用RM算法调度[35]。算法中采用了下列符号定义：

p_{ss}：server 的周期；

c_{ss}：server 的最大执行预算；

c_s：server 当前的执行预算；

P_{ss}：server 的优先级；

P_{sys}：系统中正在运行的任务的优先级；

P_i：系统中的第 i 个优先级，设只表示最高优先级，i=1，2，…，n；

Active：若 P_{sys} 等于或高于优先级 P_i，则称优先级 P_i 处于 active 状态；

Idle：若 P_{sys} 小于 P_i，则称优先级 P_i 处于为 idle 状态；

RT(Replenish Time)：server 的执行预算的刷新时刻；

RTA(Replenish Time Amount)：server 的执行预算的刷新量；

Sporadic Server 算法描述如下：当突发性任务的请求到达时，进入 server 的请求队列，队列的排队方式不作规定，一般采用先来先服务(FIFO)策略；当 c_s >0 时，sever 执行队列头的请求，从 c_s 中减掉所执行的时间，c_s 为 0 时 server 不允许处理请求。Sporadic Server 算法的执行预算刷新策略为：

RT 的确定：当 c_s >0 且 P_{ss} 由 Idle 状态进入 active 状态时(设为时刻 t_a)，则 RT= t_a + p_{ss} ；若 c_s =0，则当 $c_s \neq 0$ 且 P_{ss} 为 active 状态时(设为时刻 t_b)确定 RT= t_b + p_{ss}。

RTA 的确定：当 P_{ss} 进入 Idle 状态或 c_s =0 时确定 RT 的 RTA 等于从确定 RT 的时刻到当前为止被用掉的执行预算量。

在文献[35]中给出了下面的定理：

定理：若周期性任务集 T 可调度，则 T 中的某个任务 T_i 被具有相同周期及相同最大执行时间的 sporadic server 任务替代后，T 仍可调度。

该定理说明 sporadic server 可被当作一个周期及最大执行时间相同的周期性任务，因此可以直接应用 RM 的可调度性测试条件，这也是 sporadic server 算法的一个主要优点。

4.3.2 统计保证

下面给出一个分析模型，根据它我们可以计算出由零星服务器处理的突发性任务丢失死线的概率上限。一个零星服务器来处理周期相近的突发性任务，整个系统根据实际需要可采用一个或多个零星服务器。下面分析由零星服务器处理的突发性任务丢失死线的概率：

设 $S=<r_s, c_s, p_s, d_s>$ 为系统中的一个零星服务器，服务策略为 FIFO，其处理的不规则周期性任务集合为 $T_i=<r_i, c_i, c_i^m, p_i, d_i>$ $(1\le i\le n)$，T_i 对应的突发性任务为 $S_i=<r_i+p_i, c_i^m-c_i, p_i, d_i>$；令 $p_s=\min_{1\le i\le n}\{p_i\}$，$c_s=\max_{1\le i\le n}\{c_i^m-c_i\}$，$d_s=p_s$。假设 T_i 的运行时间概率分布已知(可由经验得出或统计得出)，其运行时间大于平均运行时间的概率为 A_i 此即为 S_i 请求服务的概率。S_i 的请求在周期 p_s 内未能完成，则认为该请求死线丢失，显然，这是最坏情况下的假设。设 l_m 表示 S_i 发出第 j 个请求 $S_{i,j}$ 时，服务器内积压的任务需 m 个服务器周期才能完成的概率，$S_{i,j}$ 丢失死线的概率上限为 1 － l_0，S_i 丢失死线的概率上限为

$$P\{S_i \text{丢失死线}\}=A_i\ (1-l_0) \tag{4.2}$$

设 h_x^i 表示完成 S_i 的请求需 x 个服务器周期的概率；t 表示完成一个服务器周期内到达的所有突发性请求所需的服务器周期数，q_y =P{t=y}。令 $H^i(z)$ 和 Q(z)分别表示 H_x^i 和 q_y 的母函数，则

$$H^i(z)=\sum_{x=0}^{\infty}h_x^i z^i \quad =1-A_i+A_{iz} \tag{4.3}$$

$$Q(z)=\prod_{i=1}^{n}H^{i}(z)\ =\prod_{i=1}^{n}(1-A_i+A_iz) \tag{4.4}$$

对于 l_m **，当** m≥1 **时存在递推式：** $l_m=\sum_{y=0}^{n}q_y l_{m-y+1}$ **；又** m=0 **时有** $(1-q_0-q_1)l_0=q_0l_1$ **；设** L(z)**为** l_m **的母函数，则当** t **的均值** E[t]<1 **时有**

$$L(z)=\sum_{m=0}^{\infty}l_m z^m\ =\frac{(1-E[t])(1-z)}{Q(z)-z} \tag{4.5}$$

当 E[t]≥1 **时，**{ l_m }**的平稳分布不存在，无法求得统计保证上限；因此** E[t]<1 **是零星服务器能提供统计保证的接入控制条件。根据式**(3)**，采用归纳法可得** $E[t]=\sum_{i=1}^{n}A_i$ **；由定义有** $q_0=\prod_{i=0}^{n}(1-A_i)$ **，在式**(4)**中令** z=0**，则可得**

$$l_0=\frac{(1-\sum_{i=1}^{n}A_i)}{\prod_{i=0}^{n}(1-A_i)} \tag{4.6}$$

根据式(4.2)**和式**(4.6)**，我们可得到零星服务器服务的任一突发性任务的死线丢失率上限。显然，这是最坏情况下的估计，实际应用中可根据实际情况放宽限制。**

4.3.3 接入控制算法

设系统中现有 n **个周期性任务** $P_i=<r_i,c_i,p_i,d_i>$ (1≤i≤n)**，**m **个零星服务器** $S_i=<r_s^i,c_s^i,p_s^i,d_s^i>$ (1≤i≤m)**。新道达的不规则周期性任务为** $T=<r,c,c^m,p,d>$**，其运行时间大于** c **的概率为** A**，则接入控制算法描述如下：**

(1) 根据 T**的周期** p **选择合适的零星服务器为** S^k **若无合适服务器则转到步骤** 4**；**

(2) 设 S^k **正在处理的不规则周期任务突发性请求的概率为** $\{A_1^k,A_2^k,...A_t^k\}$ **；若** $\sum_{i=1}^{}A_i^k+A<1$ **则继续步骤** 3**；否则转到步骤** 4**；**

(3) 根据 T 的参数，必要时修改 S^k 的周期 p_s^k 和运行时间 c_s^k ；判断是否满足接入控制条件：$\sum_{i=1}^{n}\frac{c_i}{p_i}+\frac{c}{p}+\sum_{i=1}^{m}\frac{c_s^i}{p_s^i}\le(m+n+1)(2^{\frac{1}{m+n+1}}-1)$ ；若满足，则接纳 T；否则拒绝 T 并恢复 S 的原有参数；算法结束；

(4) 为 T 生成一个新的服务器 $S^{m+1}=<r,c^m-c,p,d>$，判断是否满足接入控制条件 $\sum_{i=1}^{n}\frac{c_i}{p_i}+\frac{c}{p}+\sum_{i=1}^{m+1}\frac{c_s^i}{p_s^i}\le(m+n+2)(2^{\frac{1}{m+n+1}}-1)$，若满足，则接纳 T，否则拒绝 T 并删除新生成的服务器；算法结束。

4.4 仿真分析

模拟器方法具有简单快速的优点，但同时也牺牲了一定的灵活性，通常一种模拟器只用于一定领域内的模拟。在DRTSS(Distributed Real-Time System simulator)之前较为成熟的模拟器基本上都是网络模拟器，如NetSim [36]。这些模拟器在网络中的广泛使用说明了特定的模拟器能为特定领域的研究开发带来极大的便利，因此实时系统研究者也开发了一些用于实时系统的模拟器，如STRESS[37]， Scheduler 1-2-3 [38]等，但这些模拟器或者只能用于硬实时系统，或者假定了特定的硬件或软件平台、只能针对特定的系统环境，对实时系统领域支持得很不完善。

为了弥补上述不足，Illinois大学的实时系统小组开发了DRTSS [39]，其主要目的就是提供一个通用的实时系统模拟器，以缩短复杂实时系统的开发周期、方便实时调度算法的验证和推广。

4.4.1 模拟器的结构

DRTSS是一个离散事件模拟器，它采用了清晰自然的面向进程的建模方法，从编程角度说，它又采用了先进的面向对象技术，用 C＋＋语言编写，运行在LINUX平台，其主要结构由五部分组成，如图4.4所示[40]。

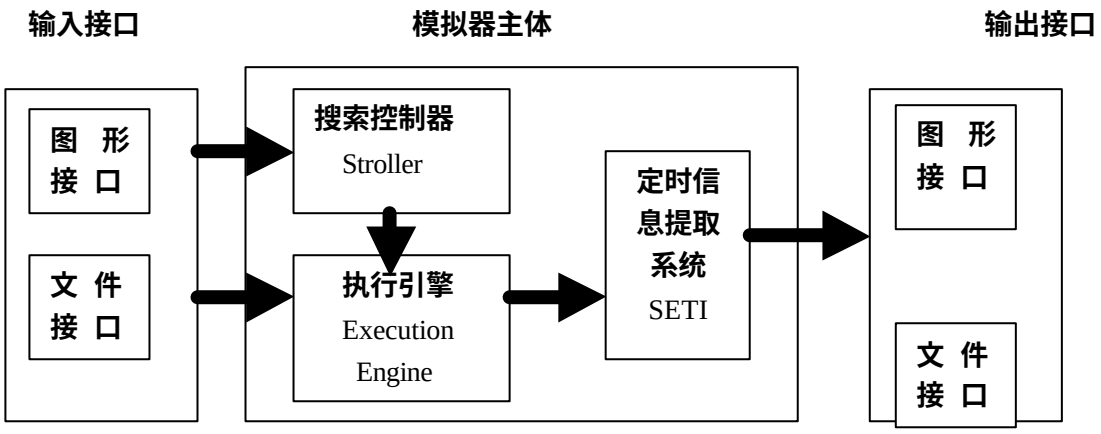

图4.4　DRTSS结构图

输入接口：DRTSS提供了图形和文件两种输入接口，用户通过接口输入模拟系统的配置信息，如模拟系统的结构、任务参数、资源参数、系统定时参数等。输入的信息被分别送往搜索控制器和执行引擎，以设置各种系统参数。

图形接口：使用简单、直观，文件接口则能更灵活地进行参数配置。

搜索控制器(Search controller)： DRTSS允许部分输入参数在一定范围内变化，这些参数的所有可能的组合则构成了模拟系统的参数空间。搜索控制器就是用来指导如何在参数空间中进行搜索，以便对每一种参数组合进行模拟。DRTSS目前采用的是线性搜索器，用户可以嵌入其它搜索器。

执行引擎(Execution Engine)：执行引擎是DRTSS中真正进行模拟的部分，它的主要功能是使用搜索控制器选择的参数组合对用户输入的系统模型进行离散事件模拟，输出模拟过程中发生的原子事件流。原子事件代表执行引擎在模拟过程中发生的各种与调度相关的行为，它只占据瞬时时间而不会引起时间的流逝，如任务实例产生事件、死线丢失事件等。执行引擎在适当的时候调度处理器资源定义的调度器来调度任务运行，并调用资源访问协议来控制任务对资源的访问。

定时信息提取系统(System for Extraction of Timing Information)：用户在输入系统参数的同时定义自己关心的输出事件，输出事件由原子事件组成，称为复合事件(compound event)。 SETI对执行引擎输出的原子事件流进行分析，输出与用户定义的复合事件相关的统计信息。

输出接口：输出接口接收SETI**输出的复合事件统计信息，并转换成图形方式或文件方式输出给用户。**

4.4.2 层次调度器

DRTSS **支持模拟者嵌入自己编写的各类调度算法，以验证新算法的正确性及评价新算法的性能。在** DRTSS **中，所有的调度算法都是一个抽象类** SchedulerInstance **的子类，**SchedulerInstance **类中定义了各种调度算法的公共接口。如图** 4.5 **所示。**

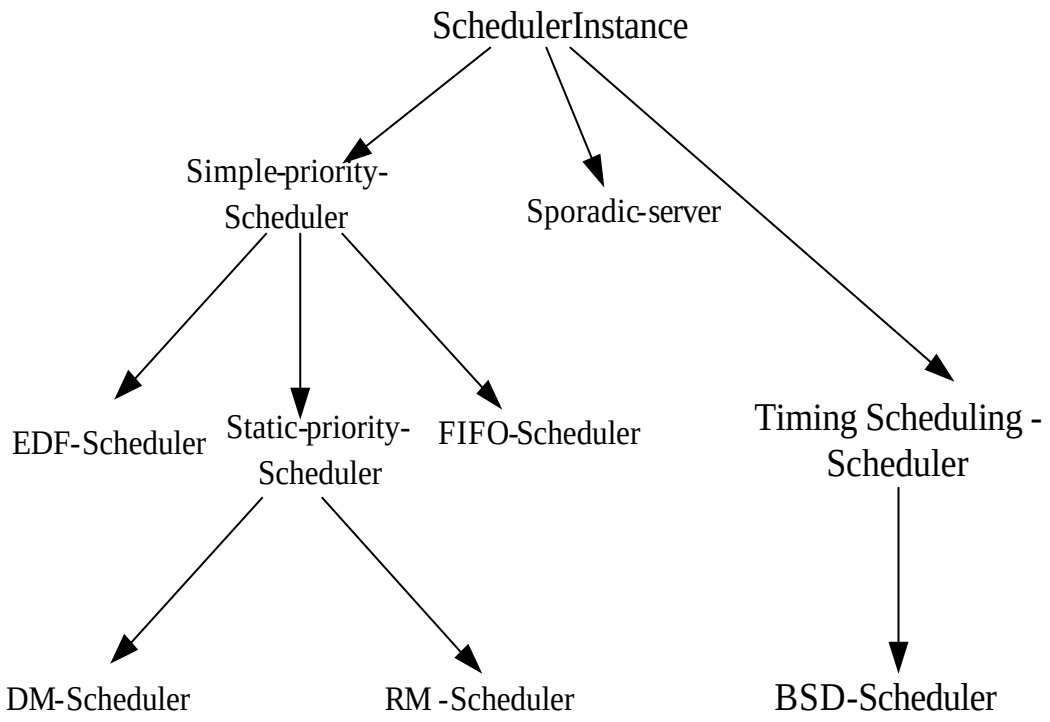

图 4.5 DRTSS **调度器层次结构**

DRTSS **的调度器呈层次化继承结构，各个调度器以** SchedulerInstance **类为公共父类。在** DRTSS **中，调度器实现为回调函数** (callbacks)**的集合，回调函数是用来处理模拟过程中产生的各种原子事件的函数，其形式为** handleXXXX()**，其中** XXXX **表示函数处理的事件名称。**DRTSS **中的原子事件如表** 4.1 **所示。表中第二列表示原子事件的原语表达式及其参数，其** CELL **表达式组合用于表示用户关心的各种组合事件。**

表 4.1 DRTSS **中的原子事件**

原子事件	原语表达式
INSTANCEGENERATED	instance generated[*task*,*instance*,*processor*]
INSTANCEARRIVED	instance arrived[*task*,*instance*,*processor*]
INSTANCERUNNING	instance running[*task*,*instance*,*processor*]
INSTANCESTOPPED	instance stopped[*task*,*instance*,*processor*]
INSTANCEFINISHED	instance finished[*task*,*instance*,*processor*]
RESOURCEREQUESTED	resource requested[*task*,*instance*,*processor*,*resource units*]
RESOURCELOCKED	resourcelocked[*task*,*instance*,*processor*,*resource units*]
RESOURCEUNLOCKED	resource unlocked[*task*,*instance*,*processor*,*resource units*]
DEADLINE	deadline[*task*,*instance*,*processor*]
DEADLINEMISSED	deadline missed[*task*,*instance*,*processor*]
DEADLINEMET	deadline met[*task*,*instance*,*processor*]
SCHEDULERGNERATED	scheduler generated[*task*,*instance*,*processor*]
SCHEDULERSTARTED	scheduler started[*task*,*instance*,*processor*]
SCHEDULERDONE	scheduler done[*task*,*instance*,*processor*]
SCHEDULERSHUTDOWN	scheduler shutdown[*task*,*instance*,*processor*]
TIMESET	Timer set[*task*,*instance*,*processor*]
TIMEEXPIRED	Timer expired[*task*,*instance*,*processor*]
BUDGETREPLENISHED	Budget replenished[*task*,*instance*,*processor*]
BOOT	Boot []

SchedulerInstance **类由公有原子事件的回调函数组成，如** handleInstGen()、handleDead()**等。除此以外，它还包括两个最重要的函数：**Consider(event)**和** Execute()**。上述函数定义了调度器的基本工作流程：内核收集模拟过程中发生的原子事件，并通过调用调度器的** Consider(event)**函数通知各调度器事件的发生；若某调度器需要处理该事件，则将其放入自己的事件队列；内核通过** Execute()**函数来激活调度器以线程**

的方式运行，运行的调度器则根据事件队列中的事件调用相应的回调函数进行处理。

4.4.3 调度算法仿真实现

为了有效地处理运行时间可变的任务，我们在调度算法中嵌入了不规则周期任务转换模块，根据用户输入的参数选择进行不规则周期任务转换。根据调度算法类的继承，实现了 RM 算法和零星服务器算法相结合的混合调度算法。RM- Sporadic-Server 调度算法类的实现如图 4.6 所示。

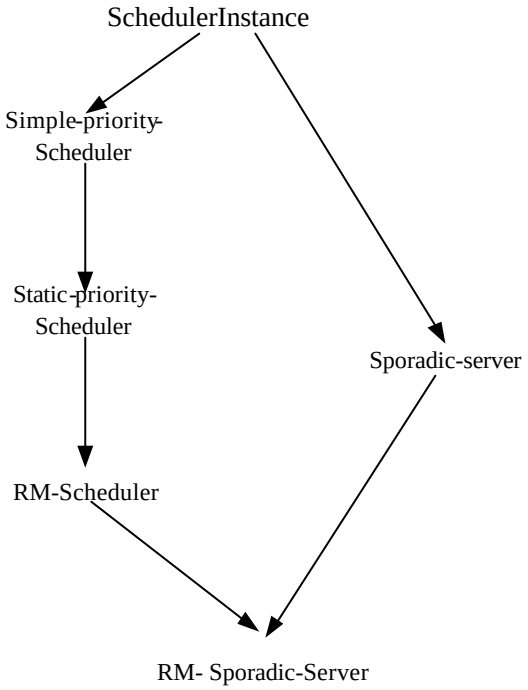

图 4.6 RM- Sporadic-Server 算法类实现

算法调度类中主要的重载的函数：

```
class RM－SS:Public Sporadicserver，Public RM
{
   Protected:
  BOOL handleProIdle(constEventHndC event);
  BOOL handleRePlenished(constEventHndC event);
  //下面几个函数修改 server 的预算(budgetf);
  Void TaskHasBeenRun(TasklnstanceHhd task，PTime workDone);
```

```
    void ResetBudget(PTime timeln);
    void RePlenishBudget(PTime time。PTime amount);
     void AdjustBudget(PTime time);
     void RequestRePlenishment(PTime now,PTime time，PTime amount);
     Public:
     Void TaskDecomPose(SimTask* task); //不规则周期任务转换程序；
     Void AdmissionControl(SimTask* task);//根据算法进行任务允许接入；
  }
```

4.4.4 实验结果

本节对提出的调度策略在 DRTSS 中进行了实现，并与常用的悲观和乐观方法进行了结果比较，证明了所嵌入算法的有效性。

在实验中，不规则周期性任务采用模拟 MPEG-1 视频流处理的任务。为满足不规则周期模型中任务实例无关的要求，任务以图组(GOP)为单位处理 MPEG-1 视频数据[41]。任务运行时间与 GOP 的大小成正比，为模拟简单，假设任务处理 1K 数据的时间为 1ms。实验中模拟的 7 条 MPEG-1 视频流取自实际的节目片段(参见表 4.2)。模拟时，处理这 7 条流的任务循环地加入系统，直到无法再接纳任何流为止。我们分别模拟了悲观方法、乐观方法及不规则周期方法。模拟结果见表 4.3 及表 4.4。

表 4.2 实验中采用的 MPEG-1 视频流

节目	MPEG 流	GOP 平均大小（KB）	GOP 峰值大小（KB）	A_i(%)	周期（ms）
1	Sports	44	165.52	13	400
2	News	21	137.32	30	600
3	Music Video	33	162.28	15	480
4	Football match	36	143.21	16	400
5	Micky	36	128.7	13	400
6	Movie1	38	202.1	30	480
7	Movie2	12	68.8	28	400

表 4.3 实验结果比较

调度方法	接纳任务数	CPU 平均利用率	平均丢失死线率
悲观方法	4	17．6%	0
乐观方法	12	97．8%	62．7%
不规则周期法	9	78．3%	16．28%

表 4.4 不规则周期方法中的死线丢失率

节目号	1	2	3	4	5	6	7
死线丢失率上限(%)	11．8	26．2	24．6	11．4	12．8	11．4	12．6
实际死线丢失率(%)	9．4	22．2	16．7	9．9	7．8	8．9	10．2

任务逐一加入系统时，死线丢失率的变化如图 4.6 所示。从实验可看出，不规则周期法有效地增加了系统可接纳的任务数，提高了资源利用率，同时又能对任务的实时性提供保证。

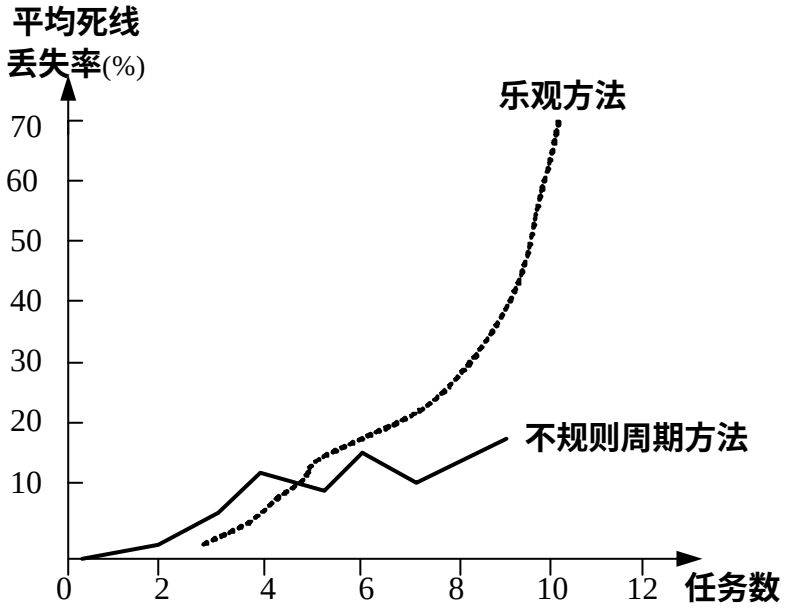

图 4.6 平均死线丢失率随任务数的变化

4.5 本章小结

针对多媒体流任务的特点，本章提出了转换的思想，将多媒体流任务转换为周期

性任务和突发性任务，并提出了任务模型。经该任务模型处理得到的任务集采用单调比率算法和零星服务器算法的混合调度方式。同时给出了接入控制算法和统计保证分析。最后利用实时系统模拟器 DRTSS 进行了仿真实验，给出实验数据验证该调度策略的有效性。

5 总结与展望

5.1 全文总结

多媒体流应用作为目前主要的多媒体应用形式，要求数据处理在一定时限内能够概率性的完成，我们称之为软实时应用。这对多媒体系统提出了相应的实时要求，涉及到的相关问题有：实时传输问题、CPU 实时任务调度管理、实时磁盘调度管理、多种媒体的同步。其中，多媒体流 CPU 实时调度算法一直是研究热点。

为此，本文介绍了传统分时操作系统CPU调度策略在实时多媒体环境存在的问题，分析了多媒体流任务的特点，指出基于实时理论的调度方式是研究的重点。而常见的实时调度方式，如基于周期模型的悲观、乐观方法以及启发式调度算法都不能很好地支持多媒体流任务。为能研究出一种既保证实时性，又提供较好资源利用率的多媒体流任务调度方式，人们开始对多媒体流任务 (运行时间可变的周期性任务)寻求新的任务模型。新的任务模型有两个目标，一是能较好地反映多媒体流任务的特性，二是能充分利用现有的实时调度理论。

本文采用任务转换的思想，构建出一种不规则周期任务模型以及相应的调度策略，用于满足多媒体流任务的软实时要求。主要研究内容包括：

(1)介绍影响多媒体系统实时性的关键问题。

(2)阐述了传统分时操作系统 CPU 调度方法支持实时多媒体应用存在的不足。

(3)分析了多媒体流压缩编码技术及主要的编码标准。

(4)针对多媒体流任务周期性和突发性的特点，分析了常见两种实时调度方式的不足和目前的研究方向。

(5)对多媒体流任务提出任务转换的思想。

(6)提出任务模型和调度方法，以及接入控制算法。

(7)对提出的调度方法进行仿真实验。

5.2 后继工作

在本文研究的基础上，将来的研究工作主要包括：

(1) 采用不同的多媒体流编码格式，做更多的仿真分析，进一步验证本调度策略的有效性；

(2) 利用本文提出的调度策略设计 CPU 任务调度器，并将其嵌入到具体的操作系统如实时 LINUX；

(3) 嵌入了上述任务调度器的实时操作系统中进行多媒体流的调度实验，得出实验数据与仿真结果做比较，对相关的算法做更深入的分析。

致　　谢

三年的研究生生活，是我人生中最宝贵的经历之一。在此论文完成之际，谨向所有关心、支持和帮助过我的老师、同学和亲人表示最崇高的敬意和最诚挚的感谢！

首先，我要特别感谢我的导师文远保教授。导师严谨的治学态度、广博的知识、敏捷的观察和分析能力使我受益匪浅。在这三年的科研生活中，他不仅对我的科研方法给予指导，而且提供了良好的科研条件和环境，使我能够经历多个项目的锻炼。并使我在良好的科研和实践相结合的环境中成长，不断地深化和扩展自己的知识。在生活中，导师也给予细心的帮助和无微不至的关怀，使我能够很好地度过三年的研究生生活。

其次，我很高兴地能够在这和睦、团结的实验室中学习和科研。这里有着宽松、和谐的学习环境，大家能够相互交流和帮助。为此，我要感谢与我一起辛勤工作过的人，包括曹文波、王斌斌、杜冠等人，当我遇到问题时，是他们和我共同探讨，给了我帮助和解答。

永远感谢我的亲人，他(她)们一直默默地关爱着我、支持着我、鼓励着我，没有他(她)们的精神支持和物质帮助我是无法完成学业。

最后，真诚感谢各位评委老师的批评指正。

参考文献

[1] P.Goyal，X.Guo，H.M Vin.A Hierarchical CPU Scheduler for Multimedia Operating Systems，Proceedings of 2nd Symposium on Operating System Design and Implementation，Seattle.WA.OCt.1996:107～121

[2] 杨学良，邵佩英，庞军平.多媒体计算机技术及其应用．北京:电子工业出版社，2004:41～45

[3] 詹磊,李强,潘清.实时磁盘调度算法研究.海军工程大学学报,2004,5(7):38～40

[4] Uresh Vahalia 著，聊鸿斌，曲广之，王元鹏等译.UNIX高级教程.北京:清华大学出版社，1999:101～106

[5] 汤子瀛，杨成忠，哲凤屏.计算机操作系统.西安:西安电子科技大学出版社，1994:54～60

[6] 周兴社，朱庆九，谷建华.UNIX SVR4 实时进程调度分析与改进.计算机研究与发展，1995，11（8）：33～37

[7] S.Maxwell著，冯锐等译.Linux内核源代码分析.北京:机械工业出版社，1999：505～525

[8] 陈莉君.Linux操作系统内核分析.北京:人民邮电出版社，2000:14～16

[9] Laut 著，罗宇等译.操作系统（第三版）.北京:电子工业出版社，2005:127～131

[10] Nieh.J.SVR4 UNIX Scheduler Unacceptable for Multimedia applications.In Proceedings of the Fourth International workshop on Network and Operating Support for Digital Audio and Video，1993:184～197

[11] Yao Wang，Jorn Ostermann，Ya-Qin Zhang 著，侯正信，杨喜，王文全译．北京:电子工业出版社，2003:129～132

[12] T.Sikora.Trends and perspectives in image and video coding. Proceedings of the IEEE，2005，93(1) :6~17

[13] G.J. Conklin， G.S. Greenbaum， K.O. Lillevold， et al. Video coding for streaming media delivery on the Internet. Circuits and Systems for Video Technology，IEEE Transactions， 2001，11(3):269~281

[14] 向健勇，朱学涛，张志伟．MPEG 系列标准的比较及最新进展．电子工程师，2004(07):37~39

[15] S. Battista，F. Casalino，C. Lande. MPEG-4: a multimedia standard for the third millennium，part 1. Multimedia，IEEE，1999，6(4):74~83

[16] S. Battista，F. Casalino， C。 Lande。 MPEG-4: a multimedia standard for the third millennium，part 2. Multimedia，IEEE，2000，7(1):76~84

[17] Weiping Li . Overview of fine granularity scalability in MPEG-4 video standard. Circuits and Systems for Video Technology，IEEE Transactions，2001，11(3):301~317

[18] Shih-Fu Chang， T. Sikora， A.Purl.Overview of the MPEG-7 standard.Circuits and Systems for Video Technology，IEEE Transactions，2001，11(6):688~695

[19] 钟玉琢，向哲，沈洪．流媒体与视频服务器．北京:清华大学出版社，2003:9~15

[20] 沙丽娜，王宁章，覃团发．视频压缩编码国际标准综述．电子科技，2005，(3):56~59

[21] 杨永杰，包志华．一种基于 H．261 建议的远程视频监控系统的实现[J]．电视技术，2003，7(4):41-43

[22] 张占军，杨学良．一种无抖动的分布式多媒体任务调度算法[J]．计算机学报．1999，22(1):24～30

[23] 杨学良，张占军．分布式多媒体计算机系统教程．北京:电子工业出版社，2002:242~244

[24] Ralf Steinmetz，Klara Nahrstedt，潘志庚等译．北京:清华大学出版社．2000:168~170

[25] C.L.Liu and J.W.Layland，Scheduling algorithms for multiprogramming in hard-real-time environment，Journal of Association for Computer Machinery，1973，20(1)：46～61

[26] J .Y.T. Leung and J. Whitehead，“On the Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks，”Performance Evaluation 1982 vol.2，237～250，

[27] Stankovic，Marco Spun，Marco Di Natale and Ciorgio Buttazo，"Implications of classical scheduling results for real-time systems，" Computer，1995，28(6)：16～25

[28] **张占军，韩承德，杨学良.多媒体应用与操作系统支持．小型微型计算机系统**.2001，22（6）：641～643

[29] **阮俊波，李红兵，金惠华．实时连续多媒体任务模型及调度算法．**2005，30（8）：864～868

[30] **张拥军，刘哗，陈福接.基于** QoS **控制的连续媒体服务任务调度.计算机研究与发展．**1999，36 (8) ：996-999

[31] A.K.Mok，D.Chen，A multiframe model for real-time tasks，IEEE Real-Time Systems Symposium，1996

[32] **黄文广.一个改进的实时任务模型．计算机研究与发展．**2001，38(2) ：234~236

[33] Deng Zetal．Probabilistic performance guarantee for real-time tasks with varying computationtimes．IEEE Real-Time Technology and Applications Symposium，1995

[34] J.W.Layland，Scheduling algorithms for multiprogramming in hard-real-time environment，Journal of Association for Computer Machinery，1984，12(1)：13～16

[35] Sprunt B et al. Aperiodic task scheduling for hard real-time systems. Real- Time Systems，1989，23(2)：27～60

[36] Andrew Heybey and Niel Robertson.The network simulator，available in format with the NetSim package．Massachusetts Institute of Technology，Version 3.1，1994

[37] N．C．Audsley,A Bums et al.Stress:A Stimulator for Hard Real-time System.Soft ware Practice and Experince，1994，24(6):543~564

[38] H．Tokuda and M．Kotera.．A Real-time Toolset for the Arts Kernel．In Proceedings of the 9^{th} IEEE Real-time Systems Symposium，1988，9(12) :289～299

[39] M.F.Storch and J.S.Liu，DRTSS: a simulation framework for complex real-time systems， Proc．of the IEEE Real-Time Technology and Applications Symposium，Brookline，1996，8(6):160～169

[40] J.W.S Liu,Z,Deng,M.Shankar.“The use of DRTSS for an Architecture and Timing Study,”Tech,Rep.in preparation.Department of Computer Science,University of Illinois at Urbana-Champain，1995

[41] **钟玉琢，向哲，沈洪．流媒体与视频服务器．北京:清华大学出版社，2003:8～10**

附录 1　攻读硕士学位期间发表的论文

[1] 文远保，张炫．单调比率算法研究及改进．计算机工程与科学，（已录用，2007 年第 1 期，署名单位：华中科技大学）

附录 2　本文所用缩写词汇英汉对照

英文缩写	英文全称	中文译名
API	Application Programe Interface	应用程序接口
ADPCM	Adaptive Differential Pulse Code Modulation	适应微分脉冲编码调制
AT&T	American Telegraph and Telephone Corporation	美国电报电话公司
CATV	Cable Television	有线电视
CBR	Content Based Retrieval	基于内容检索
CPU	Central processing unit	中央处理器
DCT	Discrete Cosine Transform	离散余弦变换
DM	Deadline Monotonic	单调死线
DRTSS	Distributed Real-Time System simulator	分布式实时系统模拟器
EDF	Earliest Deadline First	最早死线优先
FTP	File Transfer Protocol	文件传输协议
FCFS	First Come First Serve	先到先服务
FGSS	Fine-Granularity Spatially Scalable	精细的空域可扩展性
IETF	the Internet Engineering Task Force	因特网工程组
ISO	International Organization for Standardization	国际标准化组织
ITU	International Telecommunications Union	国际电信联盟
ITV	Interactive Television	交互式电视
MPEG	Moving Picture Experts Group	运动图像专家小组
PE	Priority Exchange	优先级交换
PFGS	Progress Fine Granular Scalable	渐进的精细可扩展性编码
QoS	Quality of Service	服务质量
RM	Rate Monotonic	单调比率
RT	Replenish Time	刷新时刻
RTA	Replenish Time Amount	刷新时刻数据量
SVR	System V Release	UNIX 系统版本
VO	Video Object	视频对象
VOD	Video on Demand	视频点播
WWW	World Wide Web	万维网