(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1581067 B
(45)授权公告日 2010.10.06

(21)申请号 03158834.4

(22)申请日 2003.08.22

(30)优先权数据
10/635,730 2003.08.06 US

(73)专利权人 微软公司
地址 美国华盛顿州

(72)发明人 K·德比克 R·C·B·斯皮德
C·A·路德维格 G·T·邓巴

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈斌

(51)Int.Cl.
G06F 9/30(2006.01)
G06F 3/00(2006.01)
G06F 9/45(2006.01)

(56)对比文件
US 5455910 A,1995.10.03, 说明书第9栏第42-50行, 第9栏第52-60行、附图10A、附图2.
说明书第9栏第42-50行, 第9栏第52-60行、附图10A、附图2.
CN 1162890 A,1997.10.22, 全文.
US 6172988 B1, 全文.
Sun MicroSystem.JAVA Media Framework API Guide.1999, 第85-87页、第194页、第184页、第186页、第230页。.
第85-87页、第194页、第184页、第186页、第230页。.

审查员 冯慧萍

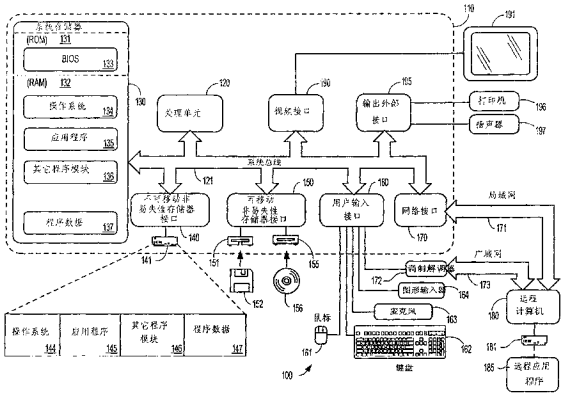
权利要求书 2 页 说明书 12 页 附图 6 页

(54)发明名称

一种与多路分配器对象通信的方法和装置

(57)摘要

一组接口和数据结构(例如,多路分配器API),用于再现一个被表示的多媒体数据的多路分配器。该数据结构利用多个字段,每一个字段包括一个命令元素。在一个实施例中,至少形成七个命令,用于多路分配器的适当的操作,包括初始化,设置表达描述符,获得表达描述符,获得未决表达描述符号,处理输入,处理输出,以及刷新命令。多路分配器API允许用户使用混合流数据,例如统一格式的DV,以产生例如音频和视频(压缩的和非压缩的)这样的基本流数据,并且允许多路分配器作为一个独立的组件。



CN 1581067 B

1. 一种用于提供与多路分配器对象通信的方法，该方法包括：

配置多路分配器对象；

在与当前激活的表达关联的所有输出被服务的情况下，动态地将多路分配器对象的一激活的表达描述符设置给下一个未决表达；

向多路分配器对象提供一新的输入混合流；

恢复至少一个来自一激活的表达的基本流；以及

刷新当前排队的输入和输出样本。

2. 根据权利要求 1 的方法，其中所述方法进一步包括在动态地将多路分配器对象的一激活的表达描述符设置给下一个未决表达之后，恢复多路分配器对象上当前激活的表达描述符的一个副本。

3. 根据权利要求 2 的方法，其中恢复多路分配器对象上当前激活的表达描述符的一个副本包括定义一个指向表达描述符的指针。

4. 根据权利要求 1 的方法，其中所述方法进一步包括在恢复至少一个来自一激活的表达的基本流之后，恢复下一个未决表达。

5. 根据权利要求 4 的方法，其中恢复下一个未决表达包括定义一个指向未决表达描述符的指针。

6. 根据权利要求 1 的方法，其中配置多路分配器对象包括定义参数，这些参数包括：

混合流描述符；

所选的媒体类型，用于混合流描述符的所选的媒体类型；

基本流的主类型阵列；以及

主类型阵列中的主类型的计数。

7. 根据权利要求 1 的方法，其中动态地将多路分配器对象的一激活的表达描述符设置给下一个未决表达包括定义一个指向表达描述符对象的指针。

8. 根据权利要求 1 的方法，其中向多路分配器对象提供一新的输入混合流包括定义一个指向样本对象的指针。

9. 根据权利要求 8 的方法，其中向多路分配器对象提供一新的输入混合流进一步包括返回一个具有新的表达标志的返回值。

10. 根据权利要求 9 的方法，进一步包括在向多路分配器对象提供一新的输入混合流之后：

如果新的表达标志具有 TRUE 值：

恢复下一个未决表达；

选择期望的流；以及

允许来自多路分配器的输入队列的样本的处理。

11. 根据权利要求 1 的方法，其中恢复至少一个来自一激活的表达的基本流包括定义一个流识别器和一个指向样本对象指针的指针。

12. 根据权利要求 11 的方法，其中恢复至少一个来自一激活的表达的基本流进一步包括返回一个输出返回值。

13. 根据权利要求 12 的方法，其中输出返回值包括流错误代码的终点和非数据错误代码之一。

14. 根据权利要求 1 的方法，其中所述方法将多路复用的数据作为数据的内存储缓冲器。

15. 根据权利要求 14 的方法，其中多路分配的数据具有一格式，所述格式包括下列格式中的至少一种 ：数字视频，MPEG2，以及 ASF。

16. 一种用于提供与多路分配器对象的通信的计算机系统，该计算机系统包括 ：

用于配置多路分配器对象的装置 ；

用于在与当前激活的表达关联的所有输出被服务的情况下，动态地将多路分配器对象的一激活的表达描述符设置给下一个未决表达的装置 ；

用于向多路分配器对象提供一个新的输入混合流的装置 ；

用于恢复至少一个来自一个激活的表达的基本流的装置 ；以及

用于刷新当前排队的输入和输出样本的装置。

一种与多路分配器对象通信的方法和装置

发明领域

[0001] 本发明涉及电数据处理，尤其是涉及在计算环境中控制多媒体数据。

[0002] 发明背景

[0003] 数字多媒体，在数字设备上观看的数字格式视频和音频的结合在容量和增殖方面快速增长。目前几乎所有新生产的个人计算机都包括一些多媒体形式。诸如摄像机，视频记录装置，电话及电视等数字产品的销售量稳步增长。由于 Internet 的持续稳定和快速壮大，在 Internet 领域，多媒体也变得更加普遍。随着这一壮大，这些计算机设备的用户提高了对其性能的期望。这些提高的用户期望不仅仅是硬件设备性能方面，也有数据本身处理能力方面。

[0004] 现有技术中，已经开发了数据流用于多媒体应用，以满足这些增长的期望。数据流允许对数据进行传递，以便可以作为稳定和持续的数据流处理。有一个好处是，可以在整个文件被传输之前将数据显示或监听，对于大的多媒体文件这是必须的。

[0005] 起初，数据流结构包括数据处理模块链（例如，俘获滤波器，转换滤波器，补偿滤波器），它们几乎不具有来自链管理者的智能。该数据处理模块也称为滤波器，作出关于如何连接、使用什么格式的数据、以及互相如何控制的决定。在链中的滤波器连接期间，协议定义了一个预定义的固定数据流序列和控制连接通信。典型的通信序列是以下列顺利通信的：接口，媒体，数据格式，分配器，以及主机时钟。在计算机系统中数据处理链提供端对端解决方案。

[0006] 随着数据流复杂度的增加，（数据处理）业界认识到必须优化以实时约束处理数据的处理链，例如视频和音频处理链。一个解决方案是Microsoft®公司的DirectShow®，其提供多媒体数据流从本地滤波器或 Internet 服务器的回放，从设备俘获多媒体数据流，以及多媒体数据流的格式转换。DirectShow®允许视频和音频文件类型内容的回放，例如 Windows Media Audio，Windows Media Video，MPEG，Apple®QuickTime®，Audio-video interleaved(AVI)，WAV(WindowsWave)。DirectShow®包括一个可插入的滤波器组件系统。滤波器为支持DirectShow®接口的对象，也能通过读取，更改和向文件写数据实现在数据流上操作。滤波器的基本类型包括源滤波器，从某一源获取数据，例如在磁盘上的文件，卫星反馈，Internet 服务器，或者 VCR，并且将其引入滤波器图表，该图表是滤波器的连接。滤波器图表中的滤波器包括转换滤波器，用于转换数据格式，以及同步和源滤波器，用于接收数据和传输数据；以及补偿滤波器，用于补偿数据，例如将数据补偿至显示设备。数据也能够被补偿到任何接收媒体的位置。包括在DirectShow®中的其它的滤波器类型包括效果滤波器，它不需要改变数据类型就可以加强效果，分析滤波器，它理解源数据的格式，并且知道如何读取校正类型，创建次数标记，以及执行查找。

[0007] 运行中，与许多控制信息一起，所有的数据从滤波器传递到滤波器。每一个滤波器包含称为“销 (pin)”的对象，它们用于与其它滤波器相连。当滤波器利用销连接的时候，创建滤波器图表。注意到“滤波器图表”与“Filter Graph”之间的差别，前者是一组相连的滤波器的概念，后者是在DirectShow®中创建的对象，它用于控制相连的一组滤波器，即

“滤波器图表”。“Filter Graph”更直接地称为滤波器图表管理器。为了控制滤波器图表中的数据流和连接，DirectShow®包括一个滤波器图表管理器。滤波器图表管理器协助保证滤波器以适当的顺序互连。但是，数据以及许多控制不通过滤波器图表管理器传递。滤波器必须适当地连接。例如，滤波器图表管理器对于给定的数据类型必须寻找补偿配置，决定有效的滤波器类型，以适当的顺序连接滤波器，并且提供适当的补偿滤波器。

[0008] 而滤波器允许大量的程序重用，这些滤波器的使用也产生了未曾预料的问题。由滤波器产生的问题之一是用于滤波器的大量的 API 出现。每一个滤波器本质上具有独立的 API。因此，一个给定的滤波器必须能够与每一个滤波器所属的 API 交互。同时，滤波器的使用产生了给定的滤波器可疑的闭锁的问题。当图表中的给定滤波器闭锁的时候，与闭锁的滤波器交互的任何一个滤波器都需要不同的辅助接口。通常，对滤波器编程，以适度的控制接口缺陷是困难的，当出现接口丢失的时候，不知道滤波器的状态。因此，接口的丢失，将导致滤波器不可预知的行为，以及最终误导已执行的程序。而且，DirectShow®中的全局控制将被分成两块。滤波器之间的接口控制数据流，而滤波器管理器控制实例化和删除滤波器。以这种方式分配控制使软件设计变得麻烦，因为存在一些交叉于两块边缘的不可避免的控制功能。DirectShow 的另一个问题是，滤波器承担媒体格式通信以及缓冲区管理功能的责任。为了完成该任务，滤波器与其它滤波器通信。依靠滤波器导致在 DirectShow 中建立的应用程序易受调试的影响，并且将被编程到滤波器无效。因此，有害的写滤波器容易降低滤波器图表和与滤波器有关的应用程序。

[0009] 需要改善DirectShow®结构。尤其是需要改善多媒体数据处理的控制。

[0010] 发明概述

[0011] 本发明涉及一组接口，数据结构，以及表示多媒体数据多路分配器的事件。该数据结构利用多个字段，每一个字段包括一个命令元素。在一个实施例中，至少形成 7 个命令，用于多路分配器适当的操作，包括 Initialize（初始化），SetPresentationDescriptor（设置表达描述符），GetPresentationDescriptor（获得表达描述符），GetPendingPresentationDescriptor（获得未决表达描述符），ProcessInput（处理输入），ProcessOutput（处理输出），Flush（刷新）命令。这些接口共同地被称为多路分配器应用程序接口 (Demux API)。Demux API 允许用户使用混合流数据，例如以统一格式的 DV，以产生基本的流数据，例如音频或视频（压缩的或者非压缩的）。

[0012] 初始化方法用于初始化和配置多路分配器对象，并且具有用于配置多路分配器的参数。参数包括混合流描述符对象，其描述混合流，用于混合流描述符的所选的媒体类型，一个用户有兴趣作为从多路分配器的输出而恢复基本流的主类型阵列，以及在主类型阵列中的多个主类型。

[0013] 设置表达描述符方法用于在多路分配器对象上动态设置激活表达描述符。设置表达描述符方法包括一个表达描述符对象的指针。处理输入方法用于提供新的输入混合流至多路分配器对象，还包括一个指向样本对象的指针。处理输入方法具有返回值，该返回值具有新的表达标志。如果新的表达标志具有 TRUE（真）值，该表达根据混合样本中的表达描述符而改变。用户必须调用获得未决表达描述符方法以恢复下一个未决表达，选择预定的流，并且调用设置表达描述符方法以允许处理来自多路分配器输入队列的样本。

[0014] 处理输出方法用于恢复来自激活表达的至少一个基本流。处理输出方法包括流识

别符和指示指向样本对象指针的指针。处理输出方法进一步包括一个输出返回值。该输出返回值包括一个流出错码终端和非数据出错码。

[0015] 刷新方法用于刷新当前队列输入和输出样本。刷新方法不需要参数。

[0016] 获得表达描述符方法用于恢复多路分配器对象上的当前激活表达描述符的副本。获得表达描述符方法包括一个表达描述符。

[0017] 获得未决表达描述符方法用于恢复下一个未决描述符。获得未决表达描述符方法包括一个未决表达描述符号。

[0018] 结合附图对实施例的下列详细说明，使得本发明的附加特征和优点将变得显而易见。

附图说明

[0019] 所附的权利要求专门阐述了本发明的特征，以及其目的和优点，可以结合附图，从以下的详细说明中最佳的理解：

[0020] 图 1 为本发明所驻留的典型计算机系统的一般模块图；

[0021] 图 2 为本发明所运行的典型计算机环境的一般模块图；

[0022] 图 3 为表示根据本发明所示出的多路分配器的模块图；

[0023] 图 4 为本发明的多路分配器的状态转换图；

[0024] 图 5 为允许本发明的调用结构的，表示数据结构模型的数据结构图；

[0025] 图 6 为表示本发明的示例信息结构的简化的数据结构图；

[0026] 图 7 为表示当应用程序与多媒体接收器不通信的时候，装载和运行根据本发明指导的多路分配器的步骤的流程图。

[0027] 发明的详细描述

[0028] 本发明提供了一组接口，数据结构和用于表达多媒体数据多路分配器的事件，这些事件共同称为多路分配器 API。该 API 允许用户使用混合流数据，例如统一格式的 DV，MPEG2，ASF 等等，以产生基本流数据，例如音频和视频（压缩的和非压缩的）。多路分配器 API 根据混合流初始化数据，支持新的流描述符的动态生成，同时，根据混合流样本，也支持新的流描述符的动态生成。此外，多路分配器 API 支持利用关于混合流的元数据的频带初始化输出，也支持利用混合流样本的频带初始化，以及支持多路分配器样本的产生，该样本能够横跨多个缓冲区。在Microsoft®公司的媒体基础结构的内容中，多路分配器 API 最初由下面将描述的媒体处理器控制，输出发送到媒体处理器。多路分配器 API 设计为：便于任何多媒体结构应当能够以定义明确的方式使用多路分配器。

[0029] 转到附图，其中同样的参考标记指同样的元件，本发明表示为在一个适当的计算机环境中实现。尽管不需要，本发明将描述计算机可执行指令的总体环境，例如程序模块，由个人计算机执行。通常，程序模块包括例程，程序，对象，组件，数据结构等等，它们执行特定的任务或者执行特定的抽象数据类型。而且，熟悉本技术领域的人员知道本发明可以由其它的计算机系统配置实现，包括手动设备，多处理器系统，基于微处理器或者可编程的用户电子设备，网络 PC，小型机，大型计算机，以及类似的设备。本发明也可以在分布式计算机环境中实现，其中任务由远程处理设备执行，它们是通过通信网络相连的。在分布式计算机环境中，程序模块可以位于本地存储器设备，也可以位于远程存储器设备。

[0030] 图 1 表示适当的计算机系统环境 100 的实例，本发明在该环境中执行。该计算机系统环境 100 仅是一个合适的计算机环境的实例，并且不规定为对本发明的功能或者使用的范围的任何限制。计算机环境 100 也不能被解释为依赖或者需要涉及基本运行环境 100 的组件中的一个或者器结合。

[0031] 本发明可操作的多个其它通用目的或者特定目的的计算机系统环境或配置。公知的计算机系统，环境，和 / 或配置的实例，它们适用于本发明，包括，但是不限于 :个人计算机，服务器计算机，手持或膝上型设备，图形输入设备，多处理器系统，微处理器系统，机顶盒，可编程用户电子设备，网络 PC，小型计算机，大型计算机，分布式计算机环境，包括任何上述系统或者设备，以及类似设备。

[0032] 本发明可以通过计算机可执行指令的通用内容描述，例如程序模块，被计算机执行。通常，程序模块包括例程，程序，对象，组件，数据结构等等，它们执行特定的任务或者执行特定的抽象数据类型。本发明也可以在分布式计算机环境下实现，其中任务由远程处理设备执行，它们通过通信网络连接。在分布式计算机环境中，程序模块可以位于本地和 / 或包括存储器设备的远程计算机存储媒体。

[0033] 参考图 1，用于执行本发明的一个基本系统，包括一个以计算机 110 的形式的通用目的计算设备。计算机 110 的组件可以包括，但是不限于，处理单元 120，系统存储器 130，系统总线 121，它将各个系统组件耦合到包括处理单元 120 的系统存储器。该系统总线 121 可以为几种总线结构中的任何一种，包括一个存储器总线或者存储器控制器，外围总线，利用任何一种总线结构的本地总线。以实例的方式，但是并不作为限制，这样的结构包括工业标准结构 (ISA) 总线，微通道结构 (MCA) 总线，增强 ISA(EISA) 总线，视频电子标准协会 (VESA) 本地总线，以及外围设备互连 (PCI) 总线，也称为中层 (Mezzanine) 总线。

[0034] 通常，计算机 110 包括一种计算机可读存储媒体。计算机可读存储媒体可以是任何可用媒体，它们能被计算机 110 访问，并且包括易失和非易失的媒体，以及可擦除和不可擦除的媒体。以实例的方式，但是并不限于，计算机可读媒体可以包括计算机存储媒体和通信媒体。计算机存储媒体包括易失和非易失的，可擦除的和不可擦除的媒体，它们是以用于信息存储任何方法或技术执行的，例如计算机可读指令，数据结构，程序模块或其它数据。计算机存储媒体包括，但是不限于，RAM，ROM，EEPROM，闪存或其它存储器技术，CD-ROM，数字通用盘 (DVD) 或者其它光盘存储器，卡型盒式磁带，磁带，磁盘存储器或者磁性存储设备，或者任何其它媒体，它们用于存储预期的信息，并且能被计算机 110 访问。通信媒体典型的表现为计算机可读指令，数据结构，程序模块或者其它在调制的数据信号中的数据，所述调制的数据信号是例如载波或者其它传输机制，也包括任何信息发送媒体。术语“调制的数据信号”的含义是一个信号，该信号具有一个或多个特征集或者以这样的方式改变以便在信号中对信息编码。通过实例的方式，但是并不限于，通信媒体包括有线媒体，例如有线网络或者直接有线连接，也包括无线媒体，例如声音的，射频 (RF)，红外线的，以及其它无线媒体。上述的任意结合应当也包括在计算机可读媒体的范围之内。

[0035] 系统存储器 130 包括计算机存储媒体，其形式有易失和 / 或非易失存储器的形式，例如只读存储器 (ROM)131 和随机访问存储器 (RAM)132。基本输入 / 输出系统 133(BIOS)，通常存储在 ROM131 中，所述系统包括基本例程，帮助在计算机 110 的部件之间转换信息，例如在启动期间。RAM132 通常包括数据和 / 或程序模块，能立即访问处理单元 120 和 / 或当

前由处理单元120运行。通过实例的方式，但是并不限于，图1表示操作系统134，应用程序135，其它程序模块136，以及程序数据137。

[0036] 计算机110也可以包括其它可擦除/不可擦除，易失/非易失计算机存储媒体。通过实例的方式，图1表示硬盘驱动器141，它从不可擦除或非易失磁性媒体读取（数据）或向其中写（数据），磁盘驱动器151，从可擦除或非易失磁性媒体读取（数据）或向其中写（数据），非易失磁盘152，以及光盘驱动器155，它从可擦除，非易失光盘156读取（数据）或向其中写（数据），例如CD ROM或者其它光学媒体。其它可擦除/不可擦除，易失/非易失计算机存储媒体，能用于基本操作环境，包括但是并不限于，卡式盒磁带，闪存卡，数字通用盘，数字视频磁带，固态RAM，固态ROM，以及类似设备。硬盘驱动器141通常通过不可擦除存储器接口，例如接口140，连接到系统总线121，而磁盘驱动器151，以及光盘驱动器155通常通过可擦除存储器接口，例如接口150，连接到系统总线121。

[0037] 以上讨论并且在图1中所表示的这些驱动器和他们的相关计算机存储媒体，提供计算机可读指令，数据结构，程序模块和用于计算机110的其它数据的存储。图1中，例如，硬盘驱动器141如图示的存储操作系统144，应用程序145，其它程序模块146，以及程序数据147。应当注意的是，这些组件可以相同或者不同于操作系统134，应用程序135，其它程序模块136，程序数据137。操作系统144，应用程序145，其它程序模块146，以及程序数据147在此图中被给出不同的数字，在最小值的情况下，表明它们为不同的副本。通过输入设备，例如键盘162，指示设备161，通常是指鼠标，轨迹球或者触摸板，麦克风163，以及图形输入板或者电数字转换器164，用户可以输入命令和信息到计算机110中。其它输入设备（未示出）可以包括操纵杆，游戏垫，碟形卫星天线，扫描仪或者类似设备。通过耦合至系统总线的用户输入接口160，这些以及其它输入设备经常被连接到处理单元120，但是也可以由其它接口和总线结构连接，例如并行端口，游戏端口，或者通用串行总线(USB)。监视器191或者其它类型的显示设备也通过接口被连接到系统总线121，例如通过视频接口190。监视器191也可以与触摸屏或者类似的设备集成。应当注意的是监视器和/或触摸屏可以物理连接到外壳，计算机设备110，例如图形输入型个人计算机，也在该外壳中。此外，计算机例如计算机设备110也可以包括其它外围输出设备，例如扬声器197和打印机196，它们可以通过输出外围接口194或类似的设备相连。

[0038] 计算机110可以在网络环境中运行，所述网络环境逻辑连接到一个或多个远程计算机的，例如远程计算机180，。该远程计算机180可以是个人计算机，服务器，路由器，网络PC，对等设备或者其它通用网络结点，以及通常包括多个或者所有的上述涉及到计算机110的部件，尽管图1中仅表示了存储器设备181。图1中描述的逻辑连接包括局域网(LAN)171和广域网(WAN)173，但是也可以包括其它网络。这样的网络环境对于办公室，企业计算机网络，intranet和Internet来说是平常事。例如，计算机系统110可以包括源机器，数据来源于此，而远程计算机180可以包括目标机器。应当注意的是源和目的机器不需要被网络或其它装置连接，但是相反的，数据可以通过任何媒体移动，该媒体被源平台写或者被目的平台或者平台读。

[0039] 当使用LAN网络环境的时候，通过网络接口或适配器170，计算机110被连接到LAN171。当使用WAN网络环境的时候，计算机110通常包括调制解调器172或者其它用于建立覆盖WAN173的通信的装置，例如Internet。该调制解调器172，可以是内部的或者是

外部的，通过用户输入接口 160，或者其它适当的机制，可以连接到系统总线 121 上。在网络环境中，描述的程序模块涉及计算机 110，或者它的一部分，可以被存储在远程存储器设备上。通过实例的方式，但是并不限于，图 1 表示驻留在存储器设备 181 上的远程应用程序 185。可以理解的是，所示的网络连接为基本的并且其它在可以被使用的计算机之间建立通信连接的装置。

[0040] 在下面的说明中，本发明将参考激活和操作的象征性表达来描述，所述操作由一个或多个计算机执行，除非另指出。因此，应当理解，这样的激活或者操作，它们被计算机执行的次数，包括由以结构化形式表达数据的电信号的计算机处理单元的控制。该控制在计算机存储器系统中转换数据或保持数据，它们以熟悉本技术领域的人员好理解的方式，再配置或者否则改变计算机的操作。数据被保持在其中的数据结构为存储器物理定位，存储器具有由数据的格式定义的特定属性。尽管本发明是在前述环境下描述的，但是这并不是限制，因为本领域的的技术人员能够理解前述的对各种激活和操作也可以用硬件执行。

[0041] 转到图 2，本发明的多路分配器可以在媒体基本结构中运行，它是应用 Microsoft 多媒体结构的一个实例。尽管图 2 表示媒体基础结构中的多路分配器，应该认识到，本发明的多路分配器 API 在其它多媒体结构中是有用的。在描述多路分配器之前，将描述该媒体基础。媒体基础为组件化结构。如图所示，媒体基础包括核心层 211 组件，它们在媒体基础中负责一些功能的基础单元，以及控制层 201 组件，利用底层的核心组件，负责执行更一般的任务。

[0042] 核心层 211 组件包括媒体源 210 以及流源 214，通过一般的定义明确的接口，提供多媒体数据。媒体源 210 描述表达，包括将被访问的流。存在许多媒体源的执行，用于提供来自不同多媒体文件类型或者设备的多媒体数据。核心层 211 进一步包括在块 208 中表示的转换，通过一般的定义明确的接口，其对多媒体数据执行一些种类的转换操作。转换实例为多媒体数字信号编解码器，视频载体，音频重取样器，统计处理器，颜色重取样器，以及其它设备。本发明的多路分配器是在图 2 中所示的结构中的转换，所述多路分配器执行交错多媒体数据作为输入，并且将数据分离为独立的有用多媒体数据的媒体流。块 208 进一步包括多路复用器，它执行独立的媒体流，并将它们组合成被插入的多媒体数据。多路复用器共享一个通用的定义明确的接口，并且存在多个实例，用于多路复用到不同的多媒体数据类型。核心层 211 进一步包括流接收器 212 和媒体接收器 230。媒体接收器 230 通过一个一般的定义明确的接口接收多媒体数据作为输入。存在多个媒体接收器，用于执行不同的多媒体数据功能。例如，将多媒体数据写入一个给定的文件类型，或者在使用视频卡的监视器上显示多媒体数据。

[0043] 控制层组件 201 使用核心层 211 组件，以通过一种较简单的方式执行较高级别的任务。通常对于给定的任务，控制层组件将使用多个不同的核心层组件。例如，回放多媒体文件将包括一个媒体源从盘上读取文件，并且解析数据，还包括一个或多个转换以将压缩的多媒体数据解压缩，以及一个或多个媒体接收器以显示多媒体数据。控制层 201 包括媒体引擎 260，与应用程序 202 交互，以接收和发送媒体流，媒体部分 240，媒体处理器 220 和在媒体部分 240 中表示的拓扑下载器 250。拓扑下载器 250 是一个控制层组件，负责描述核心层组件之间的数据流。控制层组件能够被配置为避免控制层所使用的对更多的初始核心层组件的访问。通过系统的数据流以媒体源 210 开始流过媒体部分 240，到达媒体处理器

220，并且在媒体接收器 230 处输出。媒体处理器 230 以拓扑的形式运行媒体源和其它组件的流水线。当发生拓扑事件的时候，媒体部分 240 指引，并且拓扑下载器 250 保证前述拓扑形式的事件的发生。媒体部分 240 也配置媒体处理器 220 并且使用由媒体处理器 210 返回的样本。上述过程在媒体引擎 260 的内容中进行，以及将样本从媒体处理器 220 发送到媒体接收器，它与媒体引擎 260 的调用程序（例如，应用程序 202）通信。拓扑中的组件包括媒体源 210 组件，以及媒体接收器 230 组件以及其它结点。媒体基础系统 200 提供接口，以及连接流媒体对象的安排。该系统允许用户利用拓扑的概念，通过符号提取确定一般或特定源之间的连接，传输，以及接收器对象。

[0044] 下面转到图 3，将描述多路分配器 300 的概况。在下面的说明中，将在描述中参考命令。下面即将描述这些命令。多路分配器 API 将数据格式从数据源中分离出来。多路分配器 API 执行对数据的多路复用，作为数据的内存储器缓冲区，并且执行多路分配操作。这具有预定的效果：即很多不同的数据源能够利用相同的多路分配器实例以执行该操作。例如，DV 数据能够直接来自 DV 摄像机，但是也被存储为硬盘上的一个文件。在这种情况下，有两段代码用于产生多路复用的 DV 数据（例如，一段与摄像机通信，另一段与文件系统通信），但是可以使用同样的多路分配器。多路分配器 300 支持 IMFDemultiplxer（IMF 多路分配器）接口，并且用于将混合流分开成为其基本流部分。多路分配器 300 以同步的方式运行（类似于 DMO），并且处理由混合流变换而引起的在现存的基本流的动态变化。其接收并产生样本，该样本通过 IMFSample（IMF 样本）接口表示，并且调节产生的横跨多个缓冲区的样本。

[0045] 混合流 302 为一个单独流，其包括多个基本流。一个基本流 304，306 是类似元素的流（例如，视频，音频等等）。在混合样本 308 与基本样本 310，312 之间不需要任何的一一对应。例如，在每一个混合样本中，可能或者不可能存在一个应用每一个基本流的完整的基本样本。此外，也不存在特殊的设备，对基本样本以正确的顺序在混合流中的要求。来自流的基本样本可以不共享的同样的时间标记。一个基本流可能与其它流有所偏移。在流的持续时间，一些混合流仅能够获得一组基本流。一些混合流在不同的时间可以具有不同的基本流组。

[0046] 每一组等同的基本流被称为表达 320。每一组具有一个对应的表达描述符 322。该表达描述符 322 具有两个主要目的。其一，它描述每一个基本流的媒体类型。其次，它提供选择哪一个有效流将被多路分配器 300 提取的机制。CurrentPresentation（当前表达）324 始终描述当前输出流的所选的流和数据类型。

[0047] 在样本被多路分配器 300 处理之前，用于转换被多路复用的样本为基本样本的一个分离算法需要知道哪些流将被提取。该信息包含在表达描述符 322 中。在流被选择之前，表达描述符为“未决的”。一旦流被选择，该表达描述符为“激活的”。为了使表达描述符为激活的，通过调用获得未决表达描述符，从未决表达队列 326 中重新取得。该适当的流被选择，然后调用器则调用设置表达描述符方法。在这一点（如果符合某些条件），该表达变成激活表达 324，并且从未决表达队列 326 中删除。如果来自前一个激活表达的所有的输出已经被服务，仅能有一个未决表达变成激活的。

[0048] 多路分配器 300 包括至少两组队列。它们是输入队列 330 和输出队列 340，342。当 ProcessInput() 在多路分配器 300 中被调用的时候，输入立即被处理或者输入进入输入

队列 330。一旦数据被处理，它将被输入到一个输出队列 340，342。在现存的输出队列中，该数据类型和流总是对应于当前激活表达 324。

[0049] 现在已经描述了多路分配器 300 的全部描述符，下面将描述多路分配器 300 的状态和转换。在下面的描述中，将参照命令。这些命令将在下面描述。现在转到图 4，当多路分配器 300 已经被创建但是没有调用 Initialize（初始化）的时候，多路分配器 300 在未初始化状态 400。初始化仅是该状态下在多路分配器对象 300 上的有效操作。调用 Initialize() 将使多路分配器 300 转换到未决状态 402。

[0050] 未决状态 402 表明没有有效的激活表达。ProcessOutput（处理输出）的调用将失败。为了设置激活的表达，媒体处理器 220 调用 GetPendingPresentationDescriptor()，选择适当的流并调用 SetPresentationDescriptor()。如果 GetPendingPresentationDescriptor() 的调用失败，则调用 ProcessOutput()。那么再次调用 GetPendingPresentationDescriptor()，直到获得表达描述符 322。一旦在多路分配器 300 上设置了 PresentationDescriptor（表达描述符），将转换为 Nuetral（中性）状态 404。调用 Flush() 将丢弃所有的队列输入和输出数据，并将多路分配器 300 转换为未决状态 402。

[0051] 在中性状态 404，所有的功能调用（除了初始化之外）是有效的。当在 ProcessInput（处理输入）调用处发现一个新的表达的时候，以及来自当前表达的最后样本从输出队列处被服务的时候，多路分配器 300 将转换到未决状态 402。

[0052] 一些流具有固定的、有限的持续时间，能够根据流内容被检测。当检测到这一条件的时候，以及所有的输出已经被服务的时候，多路分配器将转换到流结束状态 406。所有的其它调用将返回一个适当的出错代码。

[0053] 当一个未回收的错误出现在多路分配器 300 中的时候，将转换到错误状态 408。错误状态 408 能够从任何其它状态到达。当多路分配器 300 将被删除的时候，调用 Release()，然后，在多路分配器 300 从系统中被删除之前，多路分配器 300 转换到结束状态 410。当多路分配器 300 的最后参考被释放时，无论多路分配器 300 处什么状态都将该多路分配器 300 从存储器中删除。可以从任何状态调用 Release()，包括未初始化状态。

[0054] 目前已经描述了多路分配器 300 的状态和转换，下面将描述命令。这些命令包括 Initialize()，SetPresentationDescriptor()，GetPresentationDescriptor()，GetPendingPresentationDescriptor()，ProcessInput()，ProcessOutput()，和 Flush()。

[0055] 图 5 中所示的基本数据结构图，用于构成本发明多路分配器 API 的七个信息的表示基本信息数据结构 460。可以看到，信息数据结构 460 包括多个字段 462_{1-N}。在最佳实施例中，第一字段 462_1 预留为标题。其它字段为参数。

[0056] 根据本发明的数据结构，构成初始化命令。从图 6 所示的数据结构中可以看到，初始化命令 480 根据多个字段 482-490 构成。这些字段具有一个标题字段 482，流描述对象字段 484，媒体类型字段 486，主类型计数字段 488，以及主类型阵列字段 490。每一个不同的命令以类似的方式构成，如图 5 所示，下面将提供逐一描述。

[0057] Initialize() 方法配置和初始化多路分配器对象 300。混合流描述符可以包括媒体数据，适合于初始化该多路分配器的状态（包括任何标题数据等等）。该命令的语法如下：

[0058] HRESULT Initialize(

[0059] IMFStreamDescriptor*pMuxedStreamDescriptor,

[0060] IMFMediaType*pSelectedMediaType,

[0061] DWORD cMajorTypes,

[0062] DUID*aMajorTypes,

[0063]);

[0064] pMuxedStreamDescriptor 参数为一个输入参数,并且是一个指向流描述符对象的指针,其描述混合流。该参数的主要目的是允许任何可以在多路分配器 300 使用的流描述符的元数据。pSelectedMediaType 参数为一个输入参数,为 pMuxedStreamDescriptor 定所选择的媒体类型。这是对应于调用 ProcessInout() 中通过多路分配器 300 的样本的媒体类型。cMajorTypes 参数为一个输入参数,为 aMajorTypes 阵列中的多个主类型。该参数可以为零。aMajorTypes 参数为一个输入参数,并且它是基本流的一个主类型阵列,作为来自多路分配器 300 的输出而检索,所述流是调用者所感兴趣的。如果 cMajorTypes 等于零,该参数可以为空。在阵列中找到的每一个主类型的默认流将在表达描述符中被选择,该描述符由 GetPendingPresstationDescriptor() 返回。如果该方法成功,该方法返回一个 S_OK 值。如果该方法失败,其返回一个错误代码。如果一个表达为有效的,那么它能够通过 GetPendingPresstationDescriptor(获得未决表达描述符)方法恢复。如果表达在初始化之后无效,则在数据通过处理输入反馈给多路分配器之后,表达可以变得有效。

[0065] 表达描述符方法在多路分配器 300 上设置激活的表达描述符,所述表达描述符表明调用者感兴趣的新的流选择。表达描述符必须是一个通过获得表达描述符方法或获得未决表达描述符方法产生的描述符。该命令的语法为 :

[0066] HRESULT SetPresentationDescriptor(

[0067] IMFPresentationDescriptor*pPresentationDescriptor

[0068]);

[0069] pPresentationDescriptor 参数是一个指向表达描述符对象的指针。如果该方法成功,其返回 S_OK。如果表达描述符为无效的,将返回 MF_E_INVALID_PRESENTATION。如果表达描述符为未决表达并且将从当前激活表达仍然未决输出,则返回 MF_E_OUTPUT_PENDING。如果存在从当前激活表达的未决输出,则未决表达可以不调用 SetPresentationDescriptor()。SetPresentationDescriptor() 可以被激活表达调用,以选择或取消选择流。如果流被取消选择,那么来自在输出队列的该流的所有的样本都丢失了。如果选择新的流,那么来自流的样本在将来某时将变得有效。因为不同的流具有不同的输入和输出缓冲区要求,它在新的流样本的有效之处,将依赖单各的多路分配器。

[0070] 获得表达描述符方法恢复多路分配器 300 上的当前激活表达描述符的副本。该命令的语法为 :

[0071] HRESULT GetPresentationDescriptor(

[0072] IMFPresentationDescriptor*ppPresentationDescriptor

[0073]);

[0074] ppPresentationDescriptor 参数是一个指针,指示一个表达描述符对象。如果该方法成功,其返回 S_OK。如果失败,返回一个出错代码。如果存在未决输出,那么获得表达描述符返回对应该输出的表达描述符。

[0075] 获得未决表达描述符方法恢复下一个未决表达。该命令的语法为：

[0076] HRESULTGetPendingPresentationDescriptor(

[0077] IMFPresentationDescriptor*ppPendingPresentationDescriptor

[0078])；

[0079] ppPendingPresentationDescriptor 参数是一个指针，表明表达描述符对象。如果该方法成功，返回 S_OK。如果不存在未决表达，则该方法返回 ME_E_PRESENTATION_NOT_AVAIL_ABLE。如果处理输入被调用几次，可以有几个未决表达队列。该方法将仅返回下一个未决表达。一旦当前激活表达的所有输出都被处理过了，那么通过调用设置表达而使未决表达变成激活的。

[0080] 处理输入方法为调用者将一个新的输入混合流样本提供到多路分配器。如果多路分配器检测到表达中存在新的流，则 *pfNewPresentationAvailable 将被设置为 TRUE（真），并且调用者能够通过：:GetPendingPresentationDescriptor 恢复未决表达描述符。该命令的语法为：

[0081] HRESULT ProcessInput(

[0082] IMFSample*pSample

[0083] BOOL*pfNewPresentationAvailable

[0084])；

[0085] pSample 参数是一个指向样本对象的指针。如果该方法成功，返回 S_OK。如果调用处理输入的结果在新的表达描述符中被加入到未决表达队列，pfNewPresentationAvailable 参数返回 TRUE（真）。关于调用处理输入或者处理输出，多路分配操作可以被多路分配器执行。在任一情况下，数据在调用处理输入将被排队，直到用户调用处理输出，或者数据被调用 Flush() 丢弃。在一次中仅一个表达可以为激活的。当新的表达为有效的时候，调用者必须通过调用处理输出清空所有的未决输出，通过调用 GetPendingPresentationDescriptor（获得未决表达描述符），恢复新的表达，然后则通过调用 GetPresentation（设置表达），设置激活的表达。

[0086] 处理输入方法允许调用者恢复一个基本流或者来自激活表达的流。该命令的语法为：

[0087] HRESULT ProcessOutput(

[0088] DWORD dwStreamIdentifier,

[0089] IMFSample*ppSample

[0090])；

[0091] dwStreamIdentifier 参数是一个 32 位值，包括流识别符，来自激活表达用于样本请求。ppSample 参数是一个指示样本对象指针。如果该方法成功，返回 S_OK。如果失败，则返回出错代码。如果流终点已经到达，将返回 MF_E_ENDOFSTREAM 出错代码。如果返回 E_NO_MORE_DATA，没有有效数据处理。与另一个多路分配器数据样本一起调用 ProcessInput() 可以减轻出错。当有效数据的处理闭塞的时候，因为激活的表达不再有效，该方法将返回 MF_E_NEW_PRESENTATION。当表达改变的时候，存在新的一组有效流。用户必须向多路分配器 300 指明哪一个流将被提取。调用获得未决表达描述符，选择预期的流，并且调用 SetPresentationDescriptor() 将允许来自输入队列的更多样本的处理。如果需要，多路分

配器对象 300 将为样本分配空间。多路分配操作可以由多路分配器或者通过调用处理输入或者处理输出来执行。在另一种情况下，关于调用处理输入的数据将被排队，直到用户调用处理输出或者数据随着调用 Flush() 被丢弃。

[0092] 刷新方法允许调用者刷新多路分配器 300 中的所有当前队列输入和输出样本。刷新也清除激活表达描述符。当上游数据被查找到一个新的位置或者调用者简单地需要丢弃所有的缓冲数据的时候做这些。该命令的语法为：

[0093] HRESULTFlush()；

[0094] 刷新命令没有参数。如果该方法成功，返回 S_OK。如果失败，返回出错代码。调用刷新，清除来自多路分配器 300 的所有数据，也清除激活表达描述符。根据特定的多路分配器，通过调用获得未决表达，未决表达可以立即变得有效，或者调用者可以重复调用处理输入，直到未决表达变得有效。

[0095] 现在已经描述了多路分配器 300 的命令，下面将描述典型的操作，其中源（例如，应用程序 202）不与接收混合流的接收器通信。现在转到图 7，源表示了混合流，用于应用程序选择（例如，视频俘获源表示 DV 或者 TV 源表示一个 MPEG2 程序流）（步骤 500）。控制层 201 判断应用程序是否与混合流的接收器通信（步骤 502）。如果应用程序 202 没有与混合流的接收器通信，媒体部分 240 下载适当的多路分配器并且将其以混合流描述符初始化（步骤 504）。

[0096] 如果来自多路分配器的表达有效（步骤 506），媒体部分 240 试图利用基本流描述符通信（步骤 508）。建立其它拓扑（步骤 510）。所表示的基本流被媒体处理器 220 利用 DMO 处理（例如，解码器）。

[0097] 如果表达无效，那么媒体部分使用空媒体接收器终止混合流的拓扑，直到有更多信息为有效（步骤 512）。无论该部分什么时候开始，并且样本为流，多路分配器 300 为给定的样本，直到表达描述符变得有效（步骤 514）。通过调用 IMFMediaStream::ProcessSample 被表示混合流的媒体处理器结点调用来执行。媒体处理器 220 在底层混合流，例如 AVI 源表示 DV 流调用 IMFMediaStream::ProcessSample。当混合流样本被恢复，媒体处理器在多路分配器 300 上调用 IMFDemultiplexer::ProcessInput。处理样本和处理输入调用持续，直到新的表达标志在 IMFDemultiplexer::ProcessInput 的返回值为 TRUE（真）。由于多路分配器的改变，通过当前拓扑需要更新的事件，该媒体处理器 220 则发信号至媒体部分 240。媒体部分 240 调用 IMFDemultiplexer::GetCurrentPresentation，以获得向新的有效基本流描述符的访问。当表达描述符变得有效，该媒体部分能够利用基本流描述符重新通信（步骤 516）。

[0098] 已经描述了用于多媒体数据流的多路分配器 API。多路分配器 API 包括一组接口，数据结构和用于表达多媒体数据的多路分配器的事件。该 API 允许用户使用混合流数据，例如统一方式的 DV，以产生基本流数据，例如音频和视频（压缩的和非压缩的）。该 API 允许多路分配器用作独立的组件。该 API 减小了对大量 API 的需求，因为滤波器和给定的滤波器不再需要与 API 接口连接的能力，用于附属于每一个在系统中的滤波器，其中不支持继承滤波器。同时，多路分配器可以平稳地闭锁，因为媒体处理器控制多路分配器，并且滤波器图表中没有滤波器。

[0099] 以上所有的引用参考，包括专利，专利申请，出版物，在此全部引用作为参考。

[0100] 纵观多个可能的实施例，本发明的基本原理得以实现，应该认识到，此处参考附图所描述的实施例，仅用于解释说明，不能作为对本发明范围的限制。例如，熟悉本技术的人员将认识到以软件形式所表示的实施例的元件可以以硬件的形式实现，反之亦然，或者所表示的实施例可以在设置和细节方面被更改，而不脱离本发明的精神。因此，本发明在此希望在范围内的所有的这些实施例在由下述权利要求及等价内容的范围内表示。

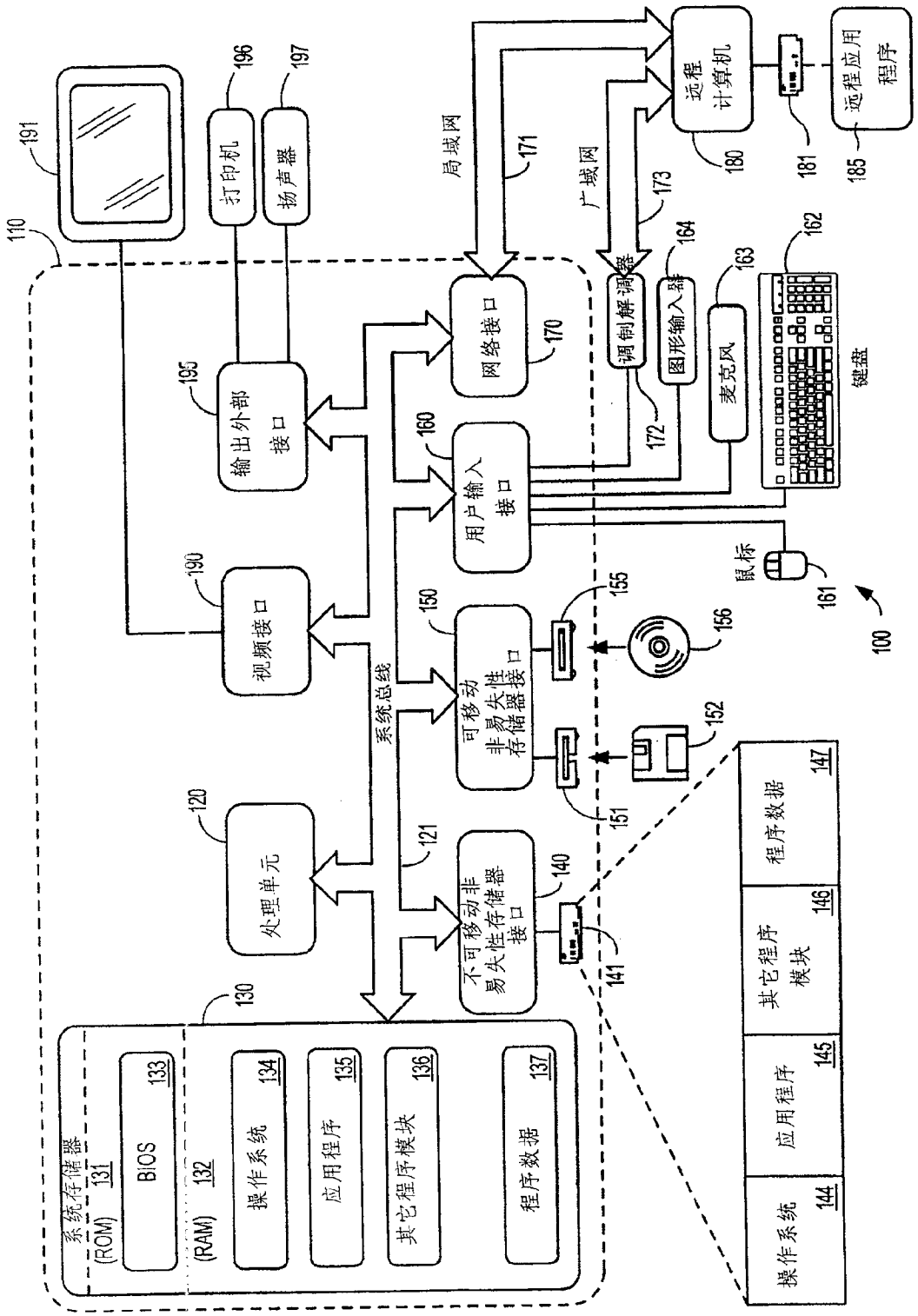

110
191
196
197
打印机
扬声器
局域网
171
远程计算机
180
181
远程应用程序
185
广域网
173
164
162
163
调制解调器
图形输入器
麦克风
键盘
172
网络接口
170
输出外部接口
135
160
用户输入接口
鼠标
161
100
190
视频接口
150
155
156
可移动非易失性存储器接口
系统总线
152
151
120
处理单元
121
140
不可移动非易失性存储器接口
141
130
系统存储器
(ROM) 131
BIOS 133
(RAM) 132
操作系统 134
应用程序 135
其它程序模块 136
程序数据 137
程序数据 147
其它程序模块 146
应用程序 145
操作系统 144

图 1

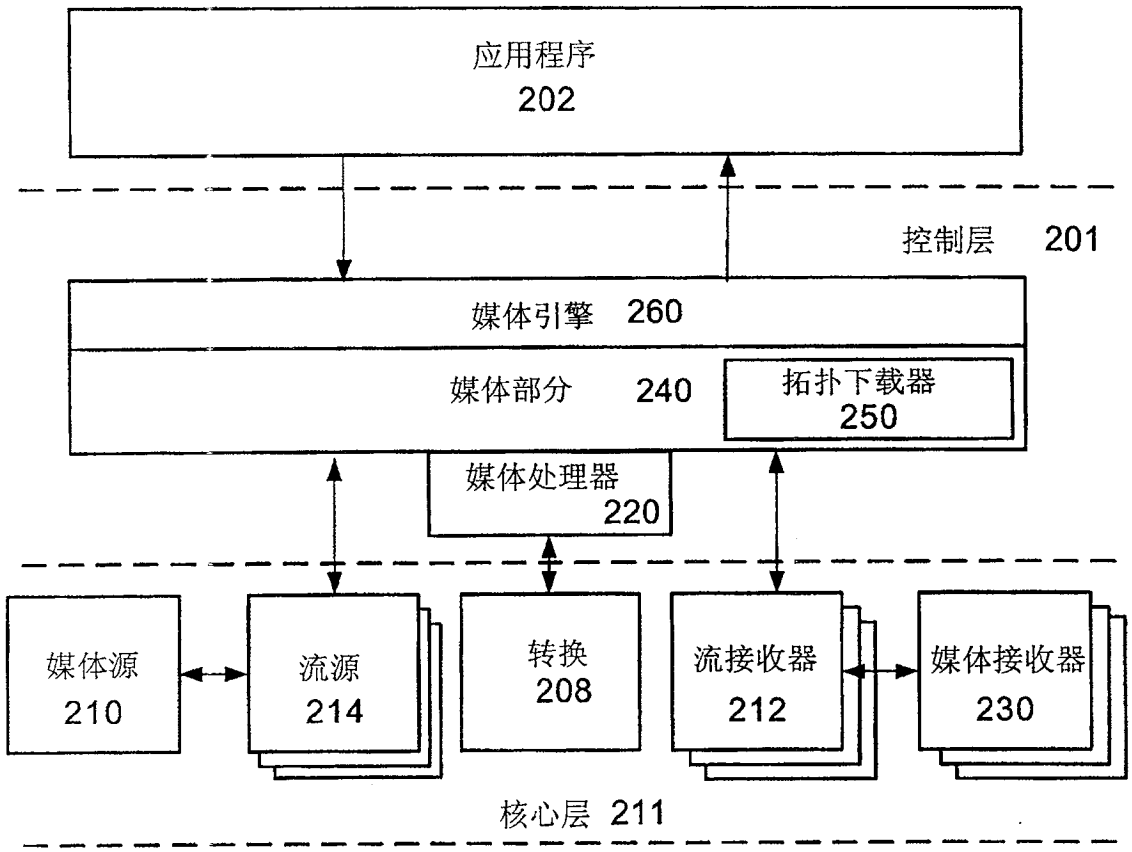

应用程序
202
控制层 201
媒体引擎 260
媒体部分 240
拓扑下载器
250
媒体处理器
220
媒体源
210
流源
214
转换
208
流接收器
212
媒体接收器
230
核心层 211

图 2

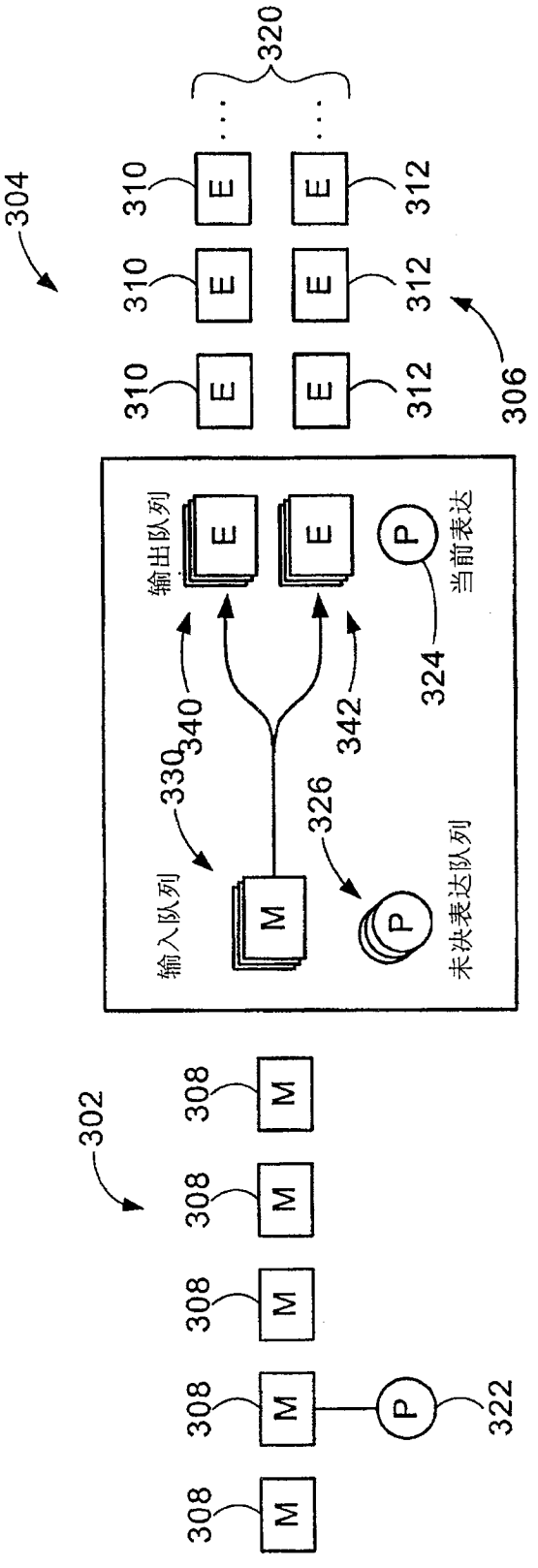

304
320
310
312
306
输出队列
当前表达
324
340
330
342
326
输入队列
未决表达队列
302
308
322
E
M
P

图 3

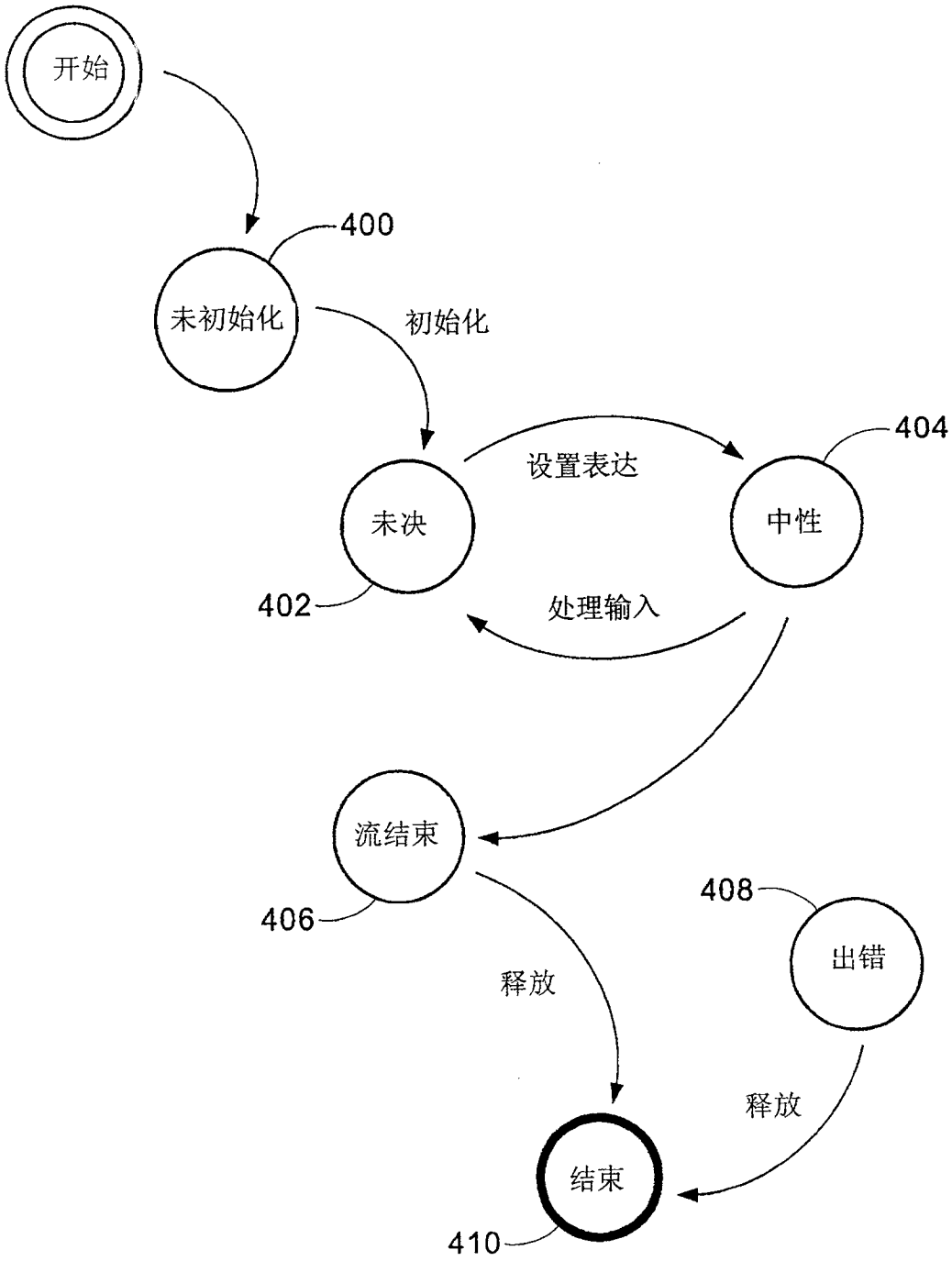

开始
400
未初始化
初始化
404
设置表达
未决
中性
402
处理输入
流结束
406
408
出错
释放
释放
结束
410

图 4

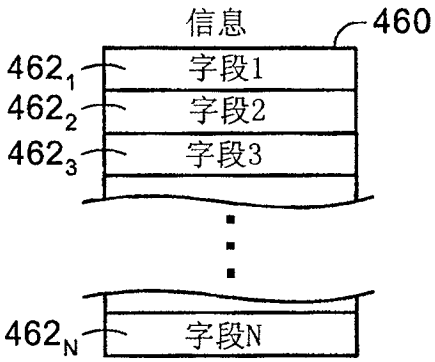

信息
460
462$_1$
字段1
462$_2$
字段2
462$_3$
字段3
462$_N$
字段N

图 5

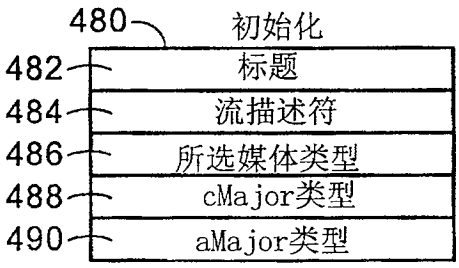

480
初始化
482
标题
484
流描述符
486
所选媒体类型
488
cMajor类型
490
aMajor类型

图 6

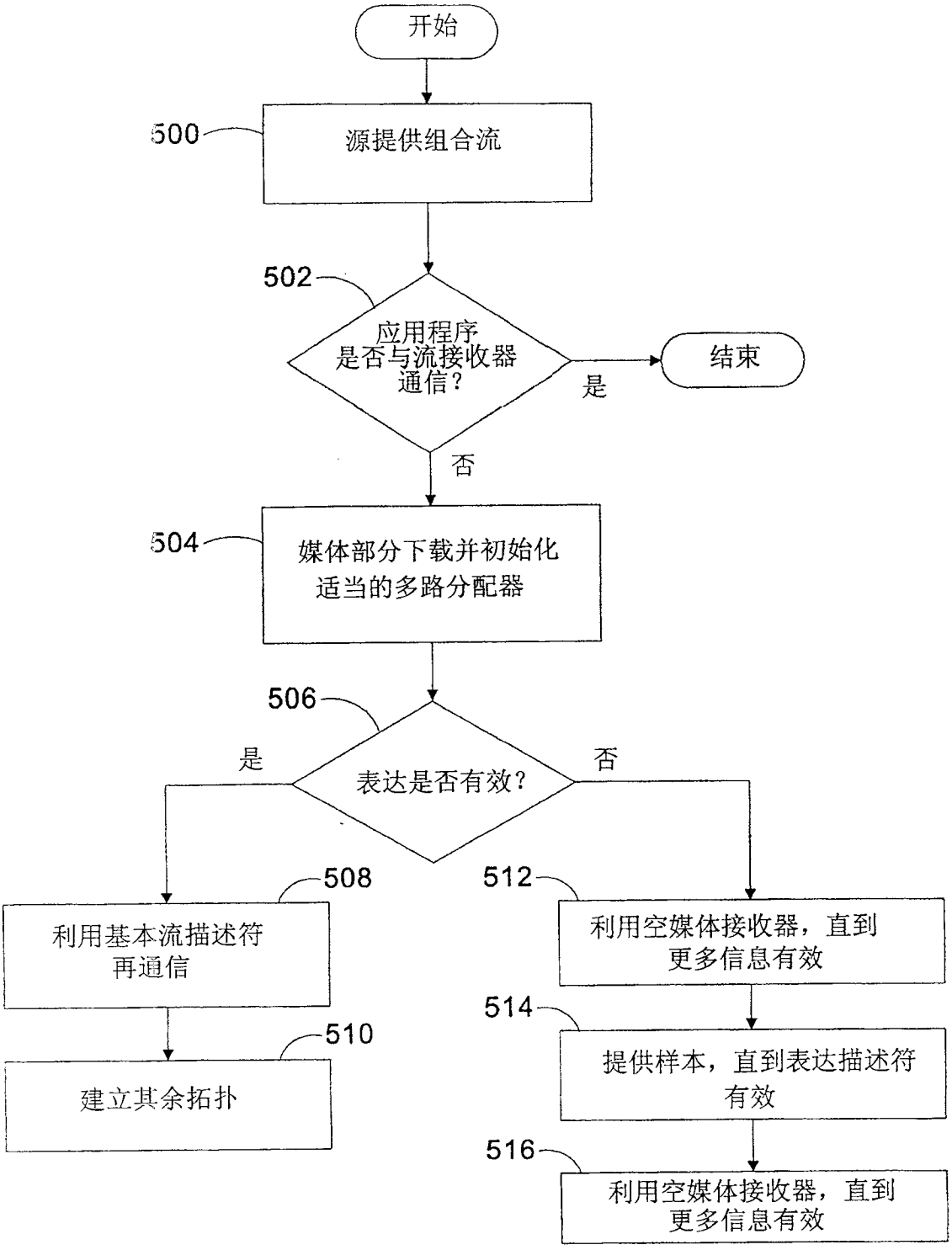

开始
500
源提供组合流
502
应用程序
是否与流接收器
通信?
是
结束
否
504
媒体部分下载并初始化
适当的多路分配器
506
是
表达是否有效?
否
508
利用基本流描述符
再通信
510
建立其余拓扑
512
利用空媒体接收器，直到
更多信息有效
514
提供样本，直到表达描述符
有效
516
利用空媒体接收器，直到
更多信息有效

图 7