



## (12)发明专利申请

(10)申请公布号 CN 107911332 A

(43)申请公布日 2018.04.13

(21)申请号 201710149319.4

H04N 21/2343(2011.01)

(22)申请日 2010.11.04

H04N 21/6587(2011.01)

(30)优先权数据

61/258,162 2009.11.04 US

H04N 21/845(2011.01)

(62)分案原申请数据

201080027024.6 2010.11.04

(71)申请人 阿莫泰克有限公司

地址 韩国仁川

(72)发明人 王业奎 李红兵 尹月静

(74)专利代理机构 北京弘权知识产权代理事务

所(普通合伙) 11363

代理人 李少丹 许伟群

(51)Int.Cl.

H04L 29/06(2006.01)

H04L 29/08(2006.01)

权利要求书2页 说明书22页 附图2页

(54)发明名称

媒体内容流播的系统和方法

(57)摘要

根据实施例之一，方法包括以电子方式从网络中接收媒体展现描述(MPD)。MPD描述其中包括多种媒体类型其它表示方式的多媒体内容，而且MPD包括可说明其它表示方式编码方法的信息。本方法还包括，基于MPD中包含的信息为一种以上的媒体类型选择多种其它表示方式，分别请求所选的多种其它表示方式之一，以及以电子方式接收一份媒体数据。



1. 一种方法，包括：

超文本传输协议HTTP流客户端以电子方式接收描述媒体展现的媒体展现描述MPD元素，其中，所述媒体展现包括多种其它展现方式，每种其它展现方式包括与媒体内容的其它编码选择相对应的片段，以及其中，所述MPD元素包括用于访问所述多种其它展现方式中的片段的元数据；以及

HTTP流客户端构建用于访问所述多种其它展现方式中的第一其它展现方式的第一片段的至少第一统一资源定位符URL，其中，第一URL根据所述MPD元素中的元数据来构建而不依赖于从超文本传输协议HTTP 1.1即HTTP/1.1协议排除的专用扩展。

2. 如权利要求1所述的方法，其中，所述MPD元素是可扩展标记语言XML文件。

3. 如权利要求1所述的方法，其中，所述MPD元素包括媒体展现持续时间属性，所述媒体展现持续时间属性指定所述媒体展现的持续时间。

4. 如权利要求1所述的方法，其中，所述MPD元素包括宽度属性，所述宽度属性指定第一其它展现方式中的视频媒体类型的水平可视展现尺寸。

5. 如权利要求1所述的方法，其中，所述MPD元素包括高度属性，所述高度属性指定第一其它展现方式中的视频媒体类型的水平可视展现尺寸。

6. 如权利要求1所述的方法，其中，所述MPD元素包括帧率属性，所述帧率属性指定第一其它展现方式中的视频媒体类型的帧率。

7. 如权利要求1所述的方法，其中，所述MPD元素包括语言属性，所述语言属性指定与第一其它展现方式有关的一种或更多种语言代码。

8. 如权利要求1所述的方法，其中，所述MPD元素包括多用途互联网邮件扩展MIME类型属性，所述MIME类型属性指定与第一片段有关的MIME类型。

9. 如权利要求1所述的方法，其中，所述MPD元素包括编解码属性，所述编解码属性指定包括在第一其它展现方式中的媒体类型。

10. 如权利要求1所述的方法，其中，所述MPD元素包括字节范围属性，所述字节范围属性指定第一片段的字节范围。

11. 如权利要求1所述的方法，其中，所述MPD元素包括播出率属性，所述播出率属性指定第一其它展现形式是否适于有效快进或快退。

12. 如权利要求1所述的方法，还包括：

HTTP流客户端将携带第一URL的HTTP GET请求发送至HTTP服务器以请求第一片段。

13. 如权利要求1所述的方法，其中，构建至少第一URL包括：

构建与第一其它展现方式中的片段集相对应的URL集。

14. 如权利要求13所述的方法，还包括：

HTTP流客户端将一系列HTTP GET请求发送到HTTP服务器以请求所述片段集，其中，所述一系列HTTP GET请求中的每个HTTP GET请求携带所述URL集中的不同URL。

15. 一种方法，包括：

超文本传输协议HTTP服务器收集描述媒体展现的媒体展现描述MPD元素，其中，所述媒体展现包括多种其它展现方式，每种其它展现方式包括与媒体内容的其它编码选择相对应的片段，以及其中，所述MPD元素包括用于访问所述多种其它展现方式中的片段的元数据；以及

HTTP服务器将MPD元素发送至HTTP流客户端，其中，MPD元素的元数据被配置为用于在

不依赖于从超文本传输协议HTTP 1.1即HTTP/1.1协议排除的专用扩展的情况下,构建其它展现方式中的各个片段的统一资源定位符URL。

16. 如权利要求15所述的方法,其中,所述MPD元素包括媒体展现持续时间属性,所述媒体展现持续时间属性指定所述媒体展现的持续时间。

17. 如权利要求15所述的方法,其中,所述MPD元素包括宽度属性、高度属性和帧率属性,所述宽度属性指定其它展现方式中的至少第一其它展现方式中的视频媒体类型的水平可视展现尺寸,所述高度属性指定第一其它展现方式中的视频媒体类型的水平可视展现尺寸,所述帧率属性指定第一其它展现方式中的视频媒体类型的帧率。

18. 一种装置,包括:

处理器;以及

非暂时性计算机可读介质,储存由所述处理器执行的程序,所述程序包括指令来:

传送媒体展现描述MPD元素,所述MPD元素描述储存在超文本传输协议HTTP服务器中的多媒体内容,所述多媒体内容包括多种媒体类型的其它展现方式,其中,所述MPD元素包括说明如何编码其它展现方式的信息,以及

其中,所述MPD元素包括用于访问所述多种其它展现方式中的各个片段的元数据,所述元数据被配置为用于在不依赖于从超文本传输协议HTTP 1.1即HTTP/1.1协议排除的专用扩展的情况下,构建其它展现方式中的各个片段的各个统一资源定位符URL。

19. 如权利要求18所述的装置,其中,所述装置包括HTTP服务器。

20. 如权利要求18所述的装置,其中,所述装置包括HTTP流客户端。

## 媒体内容流播的系统和方法

[0001] 本申请是申请日为2010年11月04日、于2011年12月26日进入中华人民共和国国家阶段的申请号为CN 201080027024.6、发明名称为“媒体内容流播的系统和方法”的中国发明专利申请的分案申请。

[0002] 相关专利申请的交叉引用

[0003] 本发明要求2009年11月4日提交的名为“HTTP流播”的第61/258,162号美国临时专利的优先权，该应用以引入的方式并入本文本中。

### 技术领域

[0004] 本发明主要涉及通信系统，具体涉及媒体内容流播的系统及方法。

### 背景技术

[0005] 因为IP网络和网络基础架构的功能得以增强，基于互联网的视频流播技术的质量出现提升，对视频传输适用互联网流播技术的情况已出现大幅增长。实施流播视频的一种常用方法是将超文本传输协议(HTTP)服务器中的视频程序传输至基于网络(Web)的HTTP客户端。基于网络流播视频由于其存取简单的特性而广为使用，因此用户仅须单击链接即可观看视频内容。

[0006] 但视频流播技术面临一些挑战。在某些情况下，当用户请求视频内容时，可能出现很长时间的迟延，而内容只能在迟延之后开始播放。在某些情况下，由于网络负荷重和/或较差的链接条件，流播视频内容遭到阻断或终止。在其它情况下，即使网络链接一开始表现良好，之后由于网络条件的变化，观看体验可能受到影响。

### 发明内容

[0007] 根据实施例之一，方法包括以电子方式从网络中接收媒体展现描述(MPD)。MPD描述其中包括多种媒体类型其它展现方式的多媒体内容，而且MPD包括可说明其它展现方式编码方法的信息。本方法还包括，基于MPD中包含的信息为一种以上的媒体类型选择多种其它展现方式，分别请求所选的多种其它展现方式之一，以及以电子方式接收一份媒体数据。

[0008] 下文将详细列出本发明实施例的特点，以便使以下发明的详细说明更易懂。后文将说明本发明实施例的其它功能和优势，构成本发明权利要求的主题。本技术领域的人员应深知，本文所透露的概念和指定实施例可被用作为修改或设计与本发明拥有相同目的的其它结构或流程的基础。上述技术人员还应认识到，这些等效结构并不违背追加申请中阐述的发明的实质和范畴。

### 附图说明

[0009] 为提供更完整的本发明见解及其优势说明，现为以下说明提供参考信息，请参阅与其相对应的附图，其中：

[0010] 图1说明了媒体流播系统；

- [0011] 图2说明了实施例数据结构；
- [0012] 图3说明了实施例媒体展现形式描述(MPD)的结构；
- [0013] 图4说明了用于执行实施例方法的计算机系统。
- [0014] 不同图片中的相应数字和符号一般指相对应的部件，除非另有说明。这些图片虽未按比例绘制，但可清晰说明实施例的相关部分。

## 具体实施方式

[0015] 下文将详细讨论多种实施例的制备和使用。但这应理解为，本发明所提供的众多适用的发明概念都能够在多种指定场合中实施。所讨论的指定实施例只是说明本发明的指定利用和使用方法，而不是限制本发明的使用范围。

[0016] 将针对一种具体场合下的实施例，例如，媒体内容流播的系统及方法，对本发明进行描述。尤其是，某些实施例与基于HTTP协议对媒体内容进行流播有关。

[0017] 图1说明了基于HTTP的媒体内容流播系统100，该系统可实现本发明的概念及方法。系统100具有可通过IP网络104将流播媒体传输至HTTP客户端106的HTTP流播服务器102。这应理解为，其它实施例也可适用于HTTP流播系统之外的其它流播系统。

[0018] 本发明实施例利用音频和/或视频和/或其它媒体类型多媒体流的HTTP流播技术的系统及方法。此类系统及方法可灵活且有效地支持基于存储方法，媒体展现描述(MPD)，以及具有或未具有字节范围的HTTP GET请求的点播及直播技术。在MPD中，可包括媒体片断的字节范围和时间范围，以便客户端能高效地请求仅使用字节范围的媒体片断。MPD可包括媒体其它展现的更多编解码信息，以支持采用一种以上编码配置而进行编码的媒体内容。例如，图2中的MPD 202指向其它展现204、206、208和210。这些其它展现中的每一项都可包含一个文件或多个文件的媒体内容数据，其中每一个文件都与一个独特的统一资源定位符(URL)相关联。

[0019] MPD可包括所有片断的最大长度，从而无需MPD中的字节范围的信令即可使用字节范围，并大大缩减MPD的大小，这意味着流播过程的启动迟延降低。为了表明直播流会话结束，服务器可采用异常方式组成下一个预期文件或片断，例如，清空文件或片断。为了成功调至直播流会话并开始请求最新内容，流播服务器中可用的且其中包含最新片断的文件使用了一个特殊的文件名称，继而使用了一个特殊的URL。为了让客户端计算出其欲寻找一个具体临时位置时启动的具体文件，当片断持续时间保持不变，以及MPD中每一个片断都没有任何字节偏移或时间偏移信令，可生成文件的URL其其可具有指明文件开始回放时间的功能。提供了可启动诸如设置、暂停、续播及停止等一般流播流程的高效流播流程，并提供了搜索、快进、快退及媒体流自适应流程。

[0020] 图3根据本发明实施例之一说明了MPD结构300。MPD 300具有字节范围及时间范围信息302、片断持续时间信息304、快进快退信息306、URL信息308及帧率信息310。在某些实施例中，可包括其它信息312。在另一种实施例中，可使用MPD结构300的子集。

[0021] 图4说明经自适应而可使用本发明实施例的计算机系统400，例如，存储和/或运行与实施例相关的软件。中央处理器(CPU)401与系统总线402相耦合。CPU 401可为一般用途的CPU。但是只要CPU 401支持本文所述的创造性操作，本发明的实施例则不受限于CPU 401的架构。总线402与随机存取存储器(RAM)403相耦合，后者可为SRAM、DRAM或SDRAM。诸如

PROM、EPROM或EEPROM等ROM 404也可与总线402相耦合。RAM 403和ROM 404存储本领域内常见的用户及系统数据和程序。

[0022] 总线402也与输入/输出(I/O)适配器405、通信适配器411、用户接口408和多媒体适配器409相耦合。I/O适配器405将存储设备406,例如以下驱动器中的一种或多种;一个硬盘驱动器、一个CD驱动器、一个软盘驱动器、一个磁带驱动器,连接至计算机系统400。I/O适配器405还可连接至打印机(未显示),后者有利于系统打印诸如文件、照片、文章等信息的纸质副本。注意:打印机可为点阵式打印机、激光打印机等打印机、传真机、扫描仪或影印机。用户接口适配器与键盘413和鼠标407及其它设备相耦合。在某些实施例中其可为显示器和/或音频卡的多媒体适配器409连接至显示设备410和音频设备415。显示设备410可为CRT、平板显示器或其它类型的显示设备,而音频设备415可为扬声器、耳机或其它模拟或数字音频系统。

[0023] 在某些实施例中,HTTP流播指代基于HTTP协议的多媒体内容的流播。3GPP版本4及以上版本支持流播分发。3GPP TS 26.234具体说明了基于RTSP over UDP协议和RTP over UDP协议的流播分发。HTTP流播成为了互联网视频分发的主要形式,而且将HTTP用作多媒体分发的主要协议,已成为趋势。在其它实施例中,可采用其它流播系统及标准。

[0024] HTTP流播得到广泛应用的技术原因包括:可使用标准服务器和标准HTTP缓存(或一般而言较廉价的服务器)来分发内容,以便能从CDN或其它任何标准服务器集群分发内容;能将对“流播会话”的控制整体转移至客户端,后者基本上仅打开一个或几个与一个或几个标准HTTP服务器相连的TCP接口。HTTP流播之所以广为应用的原因还在于,它能克服NAT及防火墙穿越问题,从而简单且轻松地提供流播服务。

[0025] HTTP流播的一种方法称为静态内容服务模式。在此模式中,使用标准HTTP服务器无需任何扩展。以一个文件或一组文件的形式提供内容,此类文件可从HTTP服务器获取。客户端通过访问使用HTTP GET请求而具有或未具有字节范围的文件来获取内容。

[0026] 凭借目前基于上述方法的HTTP流播解决方案,还无法解决至少以下问题。首先,在HTTP GET请求中使用时间范围便于客户端从流播服务器请求媒体数据。但是,现有标准HTTP服务器识别字节范围,但不能识别时间范围。

[0027] 第二,还没有相关方法可用于在MPD中发送媒体其它展现形式多种编码配置的消息的信令。第三,为了降低启动迟延,最好具有小尺寸的MPD。因而,在某些情况下,理想情况是MPD中不要具有文件或媒体片断的特定信息,因为展现可在时间层面上划分为大量的文件。但是,还没有一种相关方法可用于促成客户端有效地请求媒体片断,而MPD中无需文件或媒体片断特定的信息,例如,时间范围和/或字节范围。

[0028] 第四,当直播流流话结束时,服务器应通知客户端,以便它们不会继续尝试而不断失败,否则会给用户体验造成负面影响。将此信息通知客户端的一种方法在于使用及时更新的MPD。但是,这要求MPD更新及时为客户端所接收,而这会消耗客户端上的其它资源,并在使用静态MPD是理想且可行的方法的情况下造成负载。

[0029] 第五,当调至直播流时,客户端通常会开始接收最新内容。实现这一点的方法之一为通过及时更新的MPD传输最新媒体片断URL和/或字节范围的信息。但是,这要求MPD更新及时为客户端所接收,而这会消耗客户端上的其它资源,并在使用静态MPD是理想且可行的方法的情况下造成负载。

[0030] 第六,当片断持续时间保持不变且MPD中每一个片断都没有任何字节偏移或时间偏移信令的时候,客户端无法计算出其寻找一个具体临时位置时待启动的具体文件。最后,缺乏相关机制用于支持在HTTP流播过程中进行快进及快退操作。

[0031] 为了解决以上问题,本发明的实施例提供了以下新功能及未列在此处的其它功能。首先,在MPD中,可包括媒体片断的字节范围和时间范围,以便客户端能高效地请求仅使用字节范围的媒体片断。第二,MPD可包括媒体其它展现形式的其它编解码信息,以支持采用一种以上编码配置而进行编码的媒体内容。

[0032] 第三,MPD可包括所有片断的最大长度,使得在MPD中无需字节范围的信令即可使用字节范围,并大大缩减MPD的大小,这意味着流播过程的启动迟延降低。第四,为了表明直播流会话结束,服务器可采用异常方式组成下一个预期文件或片断,例如,清空文件或片断。第五,为了成功调至直播流会话并开始请求最新内容,流播服务器中可用的且其中包含最新片断的文件使用了一个特殊的文件名称,继而使用了一个特殊的URL。

[0033] 第六,提供有其它展现形式的URL前缀。如果一个文件包含另一展现方式的一个或多个片断,则该文件的URL是另一展现方式的URL前缀和相应的文件索引值之间的级联,例如,在具体形式为五个十进制数位的情况下,00000、00005及00012等。对于其它展现内容中的每一项,第一文件(该文件包含'moov'窗)的文件索引值等于0,而其它文件的文件索引值等于文件本身中第一片断的片断索引值,后者等于与文件本身中第一片断对应的电影片断的电影片断标头窗中的序号字段。这样一来,当片断持续时间保持不变且MPD中每一个片断都没有任何字节偏移或时间偏移信令的时候,客户端可计算出其寻找一个具体临时位置时待启动的具体文件。

[0034] 最后,提供了可启动诸如设置、暂停、续播及停止等一般流播流程的高效流播流程,并提供了搜索、快进、快退及媒体流自适应流程。

[0035] 本发明的实施例包含HTTP流播的一种静态内容服务模式方法,其中包括文件存储、媒体展现形式描述和按需流播及直播流的流播过程。

[0036] 关于文件存储,在某些实施例中,媒体展现形式描述(MPD)自身存储于一个独立的文件中。在实施例之一中,存储媒体文件,以便每一个文件包含另一展现方式的一个或多个片断。例如,在某些实施例中,一个文件可能包含且仅包含另一个音频展现内容中的一个或多个片断,一个文件可能包含且仅包含另一个视频展现内容的一个或多个片断,一个文件可能包含且仅包含另一个音视频展现内容的一个或多个片断,其中的音视频展现内容包含另一个音频展现内容和一个视频展现内容。但在实施例之一中,一个文件包含其它多个视频展现内容的一个或多个片断,而不包含其它多个音频展现内容的一个或多个片断。

[0037] 在实施例之一中,另一个展现内容的第一个片断包含'ftyp'窗和'moov'窗,而不包含'moof'窗。在'moov'窗中,在一个实施例中,不会记录任何媒体样本,即'moov'窗中包含的每一个'stts'窗中的入口计数应等于0,'moov'窗中包含的每一个'stsz'或'stz2'窗中的样本计数应等于0。因为第一个片断中没有记录任何样本,所以无须是一个相关联的'mdat'窗。在实施例之一中,第一片断之外的任何其它片断仅包含一个电影片断。或者,可以使用多个电影片断。另外,对于第一片断之外的任何其它片断,元数据('moof'窗等等)和媒体数据('mdat'窗)应存储在同一文件中。

[0038] 对于另外一些展现方式,对媒体样本的片断分区进行临时调整,从而另一展现方

式A的第n个片断含括其它任何展现方式B的同一时间段。这一点可能不适用于快进及快退的其它展现方式,对于它们来说,一个片断含括正常回放相应的其它展现方式的整数个片断。

[0039] 另一种与上述存储方法(称为存储方法1)效用相当的方法,其中有一个包含其它展现方式的第一片断的“大”'moov'窗,而且它自身存储在一个独立的文件中,而其它展现方式的其它片断按照上述存储方法进行存储。这一替代性方法称为存储方法2。对于存储方法1和2,MPD还可包含于'moov'窗中,或由'moov'窗中包含的URL所引用。

[0040] 对于直播流,除了上述规范,以下内容亦适用:

[0041] 在实施例之一中,当使用H.264/AVC视频时,参数集未存储于样本记录,而存储于独立的参数集磁迹,原因在于视频编码器可能须要在编码过程中计算出最佳编码参数,因而在生成'moov'窗之后,该参数集无法得知,且不可包含在内。

[0042] 在实施例之一中,MPD的另一种存储方式为将其作为媒体文件的一部分进行存储。但MPD自身存储于独立的文件,会使得对MPD的访问变得简单,尤其在有相关需求的情况下,仅一条HTTP GET请求即可用于获取MPD。如果其作为媒体文件的一部分进行存储,则客户端只有掌握MPD在文件中的准确位置,才能准确地单独请求MPD。单独存储MPD的另一个好处在于,因为MPD记录的增加不会对其中包含MPD的媒体文件的媒体部分的偏移产生任何影响,因而对于直播流,该流程会更加简单。

[0043] 还有其它许多方法可用于存储MPD之外的媒体内容。例如,第一替代性方法为,将其它所有展现方式存储在一个文件中,这一方法仅适用于基于RTP/RTSP的流播中。第二种替代性方法为,将所有元数据('moov'窗和'moor'窗等等)存储在一个文件,将所有媒体文件('mdat'窗)存储在其它文件,而未在时间维度上划分为多个文件(即另一种展现内容的所有片断在一个文件中)。在某些实施例中,上述两种存储方法可能不适用于缓存。

[0044] 第三种替代性方法与第二种替代性方法相似,不同的是,在时间维度上划分为多个文件,例如,每一个片断存储于两个独立的文件中,一个适用于元数据('moov'或'moof'窗),一个适用于媒体数据('mdat'窗)。相比所提议的方法,本存储方法的一大劣势在于,文件数量加倍增加,因而HTTP请求的数量加倍增加,以对同一内容进行流播。

[0045] 第四种替代性方法为,将每一个音视频替代项的一个或多个片断存储在一个文件。这一方法的一大劣势在于存储冗余。例如,当有另外两个音频表现方式时,另外的视频表现方式每一个均重复存储两遍。

[0046] 对于直播流H.264/AVC视频的情况,参数集还可存储在样本记录。但这一方法不利于视频编码器在编码过程中更改为更为理想的编码参数。另一种方法为将新参数集放置进其中包含'moof'窗的新窗中。在某些方面,这一方法不是后向兼容,而且现有的H.264/AVC文件阅读器会忽略新窗,其中需要新参数集以对电影片断中包含的样本进行正确解码。

[0047] 一种实施例句法及语义如下所示:

```
media_presentation_description()
{
    // beginning of global information
    [0048]     BOOL live_session;

    byte(4) major_brand;
    byte(4) timescale;
    byte(4) presentation_duration;
```

```
BOOL constant_segment_duration;
if(constant_segment_duration)
byte(4) segment_duration;

BOOL constant_num_segments_per_file;
if(constant_num_segments_per_file)
byte(4) num_segments_in_one_file;

BOOL num_segments_aligned;
BOOL byte_offset_included;
BOOL codec_mime_type_included_for_each_file;

byte(2) num_separate_audio_alternatives;
byte(2) num_separate_video_alternatives;
byte(2) num_av_combined_alternatives;
byte(2) num_video_fastforward_alternatives;
byte(2) num_video_rewind_alternatives;

for (i=0; i<num_separate_audio_alternatives; i++){
string codec_mime_type;
byte(4) avg_bitrate;
byte(2) language_code;
byte(1) channel_count;
string url_prefix;
byte(4) max_segment_len_in_bytes;
}

for (i=0; i<num_separate_video_alternatives; i++){
string codec_mime_type;
byte(4) width;
byte(4) height;
byte(4) avg_framerate;
byte(4) avg_bitrate;
string url_prefix;
byte(4) max_segment_len_in_bytes;
}

for (i=0; i<num_av_combined_alternatives; i++){
string audio_codec_mime_type;
byte(4) audio_avg_bitrate;
byte(2) language_code;
byte(1) channel_count;
```

```

string video_codec_mime_type;
byte(4) width;
byte(4) height;
byte(4) avg_framerate;
byte(4) video_avg_bitrate;

string url_prefix;
byte(4) max_segment_len_in_bytes;
}

for (i=0; i<num_video_fastforward_alternatives; i++){
string codec_mime_type;
byte(4) width;
byte(4) height;
byte(4) avg_framerate;
byte(4) avg_bitrate;
string url_prefix;
byte(4) max_segment_len_in_bytes;
byte(1) num_segments_denominator_ff[i];
}

[0050] for (i=0; i<num_video_rewind_alternatives; i++){
string codec_mime_type;
byte(4) width;
byte(4) height;
byte(4) avg_framerate;
byte(4) avg_bitrate;
string url_prefix;
byte(4) max_segment_len_in_bytes;
byte(1) num_segments_denominator_rw[i];
}

// end of global information, start of segment specific information

// The following six fields are just variables,
// and there are no bits used in the MPD for them
file_index_a = 0;
file_index_v = 0;
file_index_av = 0;
file_index_ff = 0;
file_index_rw = 0;
file_index = -1;

while(!EoMPD) { // EoMPD = End of Media Presentation

```

## Description

```
file_index++;

if((!constant_num_segments_per_file)&&(num_segments_aligned))
byte(4) num_segments_in_one_file;

for (i=0; i<num_separate_audio_alternatives; i++){
if(codec_mime_type_included_for_each_file)
string audio_codec_mime_type_for_one_file;
    mpd_for_one_file();
    file_index_a += num_segments_in_one_file;
}

for (i=0; i<num_separate_video_alternatives; i++) {
if(codec_mime_type_included_for_each_file)
string video_codec_mime_type_for_one_file;
    mpd_for_one_file();
    file_index_v += num_segments_in_one_file;
}

for (i=0; i<num_av_combined_alternatives; i++){
if(codec_mime_type_included_for_each_file){
string audio_codec_mime_type_for_one_file;
string video_codec_mime_type_for_one_file;
}

mpd_for_one_file();
file_index_av += num_segments_in_one_file;
}

for (i=0; i<num_video_fastfoward_alternatives; i++){
if((!file_index)|
(file_index%num_segments_denominator_ff[i]==1)){
if(codec_mime_type_included_for_each_file)
string video_codec_mime_type_for_one_file;
    mpd_for_one_file();
    file_index_ff += num_segments_in_one_file;
}
}

for (i=0; i<num_video_rewind_alternatives; i++){
if((!file_index)|
(file_index%num_segments_denominator_ff[i]==1)){
if(codec_mime_type_included_for_each_file)
string video_codec_mime_type_for_one_file;
    mpd_for_one_file();
    file_index_rw += num_segments_in_one_file;
}
}
```

```

        }
    }
}

mpd_for_one_file()
{
if(!constant_num_segments_per_file)&&(!num_segments_aligned))
byte(4) num_segments_in_one_file;

[0052] for(i=0; i<num_segments_in_one_file; i++){
if(!constant_segment_duration) {
byte(4) segment_start_time;
byte(4) segment_duration;
}
if(byte_offset_included) {
byte(4) segment_start_byte_offset;
byte(4) segment_end_byte_offset;
}
}
}
}

```

[0053] 在实施例之一中,变量定义如下:

[0054] live\_session:本字段等于FALSE (假),说明MPD适用于按需流播会话。值TRUE (真)说明MPD适用于直播流会话。

[0055] major\_brand:主文件格式标识符,说明旨在播放媒体展现内容,客户端须予以支持的文件格式特点。

[0056] 时间标度:一个指定整个展现内容时间标度的整数;其为时间单位(以秒计时)的数量。例如,一个每1/60秒测量一次时间的时间坐标系具有的时间标度等于60。

[0057] presentation\_duration:这类整数可反映出正常回放的另一次展现的展现长度(以指定的时间标度计算)。当该值等于0时,展现长度不可知。在直播流会话的媒体展现描述中,这个值设置为0。

[0058] constant\_segment\_duration:当该值为TRUE (真)时,片断长度为一个恒定的时间值。当该值为FALSE (假)时,片断的时间长度不恒定。

[0059] segment\_duration:说明片断的时间长度(以指定的时间标度计算)。

[0060] constant\_num\_segments\_per\_file:当该值为TRUE (真),其中包含'moov'窗的文件除外,每一个文件包含恒定数量的片断。当该值为FALSE (假),未包含'moov'窗的文件可能包含不同数量的片断。

[0061] num\_segments\_in\_one\_file:说明一个文件中分断的数量(其中包含'moov'窗的文件除外)。

[0062] num\_segments\_aligned:当该值为TRUE (真)时,对于其它所有展现方式,临时校准每个文件中片断的数量。当该值为FALSE (假)时,对于其它所有展现方式,不临时调整每个文件中片断的数量。

[0063] byte\_offset\_included:当该值为TRUE(真)时,每个片断的字节偏移包含在MPD内。当该值为FALSE(假)时,每个片断的字节偏移未包含在MPD内。

[0064] codec\_mime\_type\_included\_for\_each\_file:当该值为TRUE(真)时,一个编解码MIME类型包含在MPD的特定文件部分内。当该值为FALSE(假)时,一个编解码MIME类型信息仅包含在MPD的全局部分内。

[0065] num\_separate\_audio\_alternatives:指定单独存储其它音频展现方式的次数。

[0066] num\_separate\_video\_alternatives:指定单独存储其它视频展现方式的次数。

[0067] num\_av\_combined\_audio\_alternatives:指定单独存储的其它音视频展现方式的次数。

[0068] num\_video\_fastforward\_alternatives:指定单独存储其它视频快进展现方式的次数。

[0069] num\_video\_rewind\_alternatives:指定单独存储其它视频快退展现方式的次数。

[0070] codec\_mime\_type:说明另一展现中音频或视频媒体类型的初始媒体样本的MIME类型参数。对于视频,MIME型参数也包括模型和级别信息。

[0071] avg\_bitrate/audio\_avg\_bitrate/video\_avg\_bitrate:说明另一展现中音频或视频媒体类型的平均比特速率(单位为比特/秒)。

[0072] language\_code:说明此媒体的语言代码。请参阅ISO 639-2/T以了解三字符代码集。每个字符按照其ASCII值与0x60之间的差值予以打包。因为该代码限定为三个小写字母,所以此类值仅为正数。

[0073] channel\_count:说明另一展现中音频媒体类型的音频信道的数量。

[0074] url\_prefix:说明另一展现的URL前缀。如果一个文件包含另一种展现内容的一个或多个分片,则该文件的URL是另一种展现内容的URL前缀的链接和相应的文件索引值,例如,在具体形式为五个十进制数位的情况下,00000、00005及00012等。文件索引值自MPD推出。对于其它展现内容中的每一项,第一文件(该文件包含'moov'窗)的文件索引值等于0,而其它文件的文件索引值等于文件本身中第一片断的片断索引值,后者等于与文件本身中第一分片对应的电影片断的电影片断标头窗中的sequence\_number(序号)字段。当片断持续时间保持不变且MPD中每一个片断都没有任何字节偏移或时间偏移信令的时候,客户端可计算出其寻找一个具体临时位置时待启动的具体文件。

[0075] max\_segment\_len\_in\_bytes:指定片断的最大长度(以字节计)。此值促成在MPD中无需字节范围的信令即可使用合适的字节范围。例如,基于片断的起始位置,在不知片断长度(以字节计)的情况下,客户端可能请求数据块,后者大小等于max\_segment\_len\_in\_bytes,以确保请求整个片断。基于片断的特定位置,在不知片断长度(以字节计量)的情况下,客户端可能请求数据块,后者大小等于max\_segment\_len\_in\_bytes减去特定位置的字节长度,以确保请求整个片断。

[0076] 宽度:说明另一展现中视频媒体类型的水平分辨率(以象素计量)。

[0077] 高度:说明另一展现中视频媒体类型的垂直分辨率(以象素计量)。

[0078] avg\_framerate:说明另一展现中视频媒体类型的平均帧率,以帧/每256秒为单位。对于其它的视频快进或快退展现方式,此值计算过程为正常回放另一展现的展现长度(以指定的时间标度计算)除以所有视频帧的数量,再换算为以帧/每256秒为单位。

- [0079] num\_segments\_denominator\_ff[i]:第i个视频快进的另一展现的每个片断对应着与正常回放另一次视频展现的片断的num\_segments\_denominator\_ff[i]值相等的数。
- [0080] num\_segments\_denominator\_rw[i]:第i个视频快进的另一展现的每个片断对应着与正常回放另一次视频展现的片断的num\_segments\_denominator\_rw[i]值相等的数。
- [0081] audio\_codec\_mime\_type\_for\_one\_file:说明与特定文件索引值对应的文件中的音频样本的编解码MIME型。
- [0082] video\_codec\_mime\_type\_for\_one\_file:说明与特定文件索引值对应的文件中的视频样本的编解码MIME型。此值包括模型和级别信息。
- [0083] segment\_start\_time:说明相对于展现的起始时间,片断的起始时间(以毫秒计量)。
- [0084] segment\_duration:说明片断的时间长度(以指定的时间标度计算)。
- [0085] segment\_start\_byte\_offset:说明文件(其中包含片断)中的片断的第一个字节的字节偏移。
- [0086] segment\_end\_byte\_offset:说明文件(其中包含片断)中的片断的最后一个字节的字节偏移。
- [0087] 这应理解为,另一个实施例可包含上述命令、变量及定义的子集。
- [0088] 在实施例之一中,MPD可采用XML或SDP加以描述,或根据基于ISO的媒体文件格式将其描述为一个新窗中包含的数据块字段。采用XML或SDP形式的MPD还可包含在窗内,例如,'moov'窗或文件中'ftyp'窗正后面的新窗。
- [0089] 以下所示的XML示例方案说明了XML中任何MPD的格式。

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element
        name="media_presentation_description_for_http_streaming">

[0090]        <xs:sequence>

            <xs:element name="live_session" type="xs:boolean"/>

            <xs:element name="major_brand" type="xs:unsignedLong"/>
            <xs:element name="timescale" type="xs:unsignedLong"/>
            <xs:element name="presentation_duration" type="xs:unsignedLong"/>
```

```

<xs:element name="constant_segment_duration" type="xs:boolean"/>
<xs:element      name="segment_duration"      type="xs:unsignedLong"
minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction>
            <xs:assertion test="/constant_segment_duration[1]=true"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element          name="constant_num_segments_per_file"
type="xs:boolean"/>
<xs:element          name="num_segments_in_one_file"
type="xs:unsignedLong" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction>
            <xs:assertion test="/constant_num_segments_per_file[1]=true"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

[0091]   <xs:element name="num_segments_aligned" type="xs:boolean"/>
<xs:element name="byte_offset_included" type="xs:boolean"/>
<xs:element      name="codec_mime_type_included_for_each_file"
type="xs:boolean"/>

    <xs:element          name="num_separate_audio_alternatives"
type="xs:unsignedShort"/>
    <xs:element          name="num_separate_video_alternatives"
type="xs:unsignedShort"/>
    <xs:element          name="num_av_combined_alternatives"
type="xs:unsignedShort"/>
    <xs:element          name="num_video_fastfoward_alternatives"
type="xs:unsignedShort"/>
    <xs:element          name="num_video_rewind_alternatives"
type="xs:unsignedShort"/>

    <xs:group          minOccurs="/num_separate_audio_alternatives[1]"
maxOccurs="/num_separate_audio_alternatives[1]">
        <xs:sequence>
            <xs:element name="codec_mime_type" type="xs:string"/>
            <xs:element name="avg_bitrate" type="xs:unsignedLong"/>
            <xs:element name="language_code" type="xs:unsignedShort"/>

```

```

<xs:element name="channel_count" type="byte"/>
<xs:element name="url_prefix" type="xs:string"/>
<xs:element      name="max_segment_len_in_bytes"      type
="xs:unsignedLong"/>
</xs:sequence>
</xs:group>

<xs:group      minOccurs="/num_separate_video_alternatives[1]"
maxOccurs="/num_separate_video_alternatives[1]">
<xs:sequence>
<xs:element name="codec_mime_type" type="xs:string"/>
<xs:element name="width" type="xs:unsignedLong"/>
<xs:element name="height" type="xs:unsignedLong"/>
<xs:element name="avg_framerate" type="xs:unsignedLong"/>
<xs:element name="avg_bitrate" type="xs:unsignedLong"/>
<xs:element name="url_prefix" type="xs:string"/>
<xs:element      name="max_segment_len_in_bytes"      type
="xs:unsignedLong"/>
</xs:sequence>
</xs:group>

<xs:group      minOccurs="/num_av_combined_alternatives[1]"
maxOccurs="/num_av_combined_alternatives[1]">
<xs:sequence>
<xs:element name="audio_codec_mime_type" type="xs:string"/>
<xs:element name="audio_avg_bitrate" type="xs:unsignedLong"/>
<xs:element name="language_code" type="xs:unsignedShort"/>
<xs:element name="channel_count" type="byte"/>

<xs:element name="video_codec_mime_type" type="xs:string"/>
<xs:element name="width" type="xs:unsignedLong"/>
<xs:element name="height" type="xs:unsignedLong"/>
<xs:element      name="video_avg_framerate"
type="xs:unsignedLong"/>
<xs:element name="avg_bitrate" type="xs:unsignedLong"/>

<xs:element name="url_prefix" type="xs:string"/>
<xs:element      name="max_segment_len_in_bytes"      type
="xs:unsignedLong"/>
</xs:sequence>
</xs:group>

<xs:group      minOccurs="/num_video_fastfoward_alternatives[1]"
maxOccurs="/num_video_fastfoward_alternatives[1]">

```

```

<xs:sequence>
  <xs:element name="codec_mime_type" type="xs:string"/>
  <xs:element name="width" type="xs:unsignedLong"/>
  <xs:element name="height" type="xs:unsignedLong"/>
  <xs:element name="avg_framerate" type="xs:unsignedLong"/>
  <xs:element name="avg_bitrate" type="xs:unsignedLong"/>
  <xs:element name="url_prefix" type="xs:string"/>
  <xs:element name="max_segment_len_in_bytes" type="xs:unsignedLong"/>
  <xs:element name="num_segments_denominator_ff" type="xs:byte"/>
</xs:sequence>
</xs:group>

<xs:group minOccurs="/num_video_rewind_alternatives[1]" maxOccurs="/num_video_rewind_alternatives[1]">
  <xs:sequence>
    <xs:element name="codec_mime_type" type="xs:string" />
    <xs:element name="width" type="xs:unsignedLong"/>
    <xs:element name="height" type="xs:unsignedLong"/>
    <xs:element name="avg_framerate" type="xs:unsignedLong" />
    <xs:element name="avg_bitrate" type="xs:unsignedLong" />
    <xs:element name="url_prefix" type="xs:string"/>
    <xs:element name="max_segment_len_in_bytes" type="xs:unsignedLong"/>
    <xs:element name="num_segments_denominator_rw" type="xs:byte"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="mpd_for_one_file">
  <xs:sequence>
    <xs:element name="num_segments_in_one_file" type="xs:unsignedLong">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion test="/constant_num_segments_per_file[1]=false"/>
          <xs:assertion test="/num_segments_aligned[1]=false"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

```

<xs:group minOccurs="/num_segments_in_one_file[1]"
maxOccurs="/num_segments_in_one_file[1]>
  <xs:sequence>
    <xs:element name="segment_start_time" type="xs:unsignedLong">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion test="/constant_segment_duration=false"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="segment_duration" type="xs:unsignedLong">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion test="/constant_segment_duration=false"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="segment_start_byte_offset"
type="xs:unsignedLong">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion test="/byte_offset_included[1]=true"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="segment_end_byte_offset"
type="xs:unsignedLong">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion test="/byte_offset_included[1]=true"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:group>

</xs:sequence>
</xs:complexType>

<xs:group maxOccurs="unbounded">

  <xs:element name="num_segments_in_one_file">

```

```

        type="xs:unsignedLong">
            <xs:simpleType>
                <xs:restriction>
                    <xs:assertion
test="/constant_num_segments_per_file[1]=false"/>
                    <xs:assertion test="/num_segments_aligned[1]=true"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>

        <xs:group      minOccurs="/num_separate_audio_alternatives[1]"
maxOccurs="/num_separate_audio_alternatives[1]">
            <xs:sequence>
                <xs:element      name="audio_codec_mime_type_for_one_file"
type="xs:string">
                    <xs:simpleType>
                        <xs:restriction>
                            <xs:assertion
test="/codec_mime_type_included_for_each_file[1]=true"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element type="mpd_for_one_file"/>
            </xs:sequence>
        </xs:group>

        <xs:group      minOccurs="/num_separate_video_alternatives[1]"
maxOccurs="/num_separate_video_alternatives[1]">
            <xs:sequence>
                <xs:element      name="video_codec_mime_type_for_one_file"
type="xs:string">
                    <xs:simpleType>
                        <xs:restriction>
                            <xs:assertion
test="/codec_mime_type_included_for_each_file[1]=true"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element type="mpd_for_one_file"/>
            </xs:sequence>
        </xs:group>

        <xs:group      minOccurs="/num_av_combined_alternatives[1]"
maxOccurs="/num_av_combined_alternatives[1]">

```

```

<xs:sequence>
  <xs:element name="audio_codec_mime_type_for_one_file"
  type="xs:string">
    <xs:simpleType>
      <xs:restriction>
        <xs:assertion
          test="/codec_mime_type_included_for_each_file[1]=true"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="video_codec_mime_type_for_one_file"
  type="xs:string">
    <xs:simpleType>
      <xs:restriction>
        <xs:assertion
          test="/codec_mime_type_included_for_each_file[1]=true"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element type="mpd_for_one_file"/>
</xs:sequence>
</xs:group>

```

[0096]

```

<xs:group minOccurs="/num_video_fastfoward_alternatives[1]"
maxOccurs="/num_video_fastfoward_alternatives[1]">
  <xs:sequence>
    <xs:element name="video_codec_mime_type_for_one_file"
    type="xs:string">
      <xs:simpleType>
        <xs:restriction>
          <xs:assertion
            test="/codec_mime_type_included_for_each_file[1]=true"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element type="mpd_for_one_file"/>
    </xs:sequence>
    </xs:group>
    <xs:group minOccurs="/num_video_rewind_alternatives[1]"
maxOccurs="/num_video_rewind_alternatives[1]">
      <xs:sequence>
        <xs:element name="video_codec_mime_type_for_one_file"
        type="xs:string">

```

```

<xs:simpleType>
  <xs:restriction>
    <xs:assertion
      test="/codec_mime_type_included_for_each_file[1]=true"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element type="mpd_for_one_file"/>
</xs:sequence>
</xs:group>

</xs:group>

</xs:sequence>
</xs:element>
</xs:schema>

```

[0097] [0098] 对于某些实施例,MPD的上述实施例句法及语义适应于存储方法1。使用存储方法2时,MPD在某些实施例中可如下略微进行修改。例如,MPD可包含在“giant”’moov’窗中或由“giant”’moov’窗中的URL引用。可从“giant”’moov’窗为其查找到等效信息的字段可从MPD中抽除,例如,major\_brand、时间标度、presentation\_duration。文件索引值初始设置为1,而非0,以保证在环路中全局信息之后,该索引值起始于1,而且为其中包含“giant”’moov’窗的文件保留文件索引值0。

[0099] 在实施例之一中,URL前缀新增为其中包含“giant”’moov’窗的文件的全局信息的一部分,而且该文件的URL是URL前缀和“00000”的链接。或者,文件本身的URL新增为全局信息的一部分。

[0100] 为在MPD中传输编解码信息的信令,一种替代性方法是传输一种媒体类型另一展现的一种全局编解码MIME类型的信令,并指定全局codec\_mine\_type仅说明另一展现中的上述媒体类型的初始编解码MIME类型。相似的方法可适用于流播其中包含诸如计时文本和计时图形等其它媒体类型的多媒体内容。

[0101] 在某些实施例中,按需流播流程执行后,可实现设置、暂停、续播及停止功能。在实施例之一中,流播设置包括以下步骤。首先,客户端获取MPD的URL。客户端获取MPD的URL的方式不属于本文的范围。第二,例如,客户端通过使用HTTP GET请求获取MPD。客户端还可能通过采用一种循序渐进的方式(即使用具有字节范围的多个HTTP GET请求)获取MPD。当适用MPD渐进式请求方式时,以下步骤即可在各个片断的MPD数据可为客户端所用的情况下马上启动,而且客户端可在使用TCP链接请求片断时,使用单独的TCP链接请求MPD。使用存储方法2时,客户端获取其中包含“giant”’moov’窗的文件,而在未包含MPD的情况下,客户端通过使用HTTP GET请求或采用循序渐进的方式获取MPD。

[0102] 第三,客户端解析MPD并选择其中包含客户端所需的所有媒体类型的另一次或其它多次合适展现方式。使用存储方法2时,客户端会在选择其它合适的展现方式的过程中解析其中包含“giant”’moov’窗的文件并使用文件及/或MPD中的信息。

[0103] 第四,始于所选的其它展现方式每一项的第一片断,客户端请求所选的其它展现

方式的片断。使用存储方法2时,始于所选的其它展现方式每一项的第二片断,客户端请求所选的其它展现方式的片断。

[0104] 在实施例之一中,其中包含一个特定片断的文件的URL由MPD指定,其中每一个文件索引值对应着一个文件和多个片断,而且一个文件索引值的URL是相应URL前缀和文件索引值的级联,例如,采用五个十进制数位的形式。这一规则适应于流播过程涉及的所有HTTP GET请求,其中包括直播流情况。使用存储方法2时,为其中包含" giant "'moov'窗的文件保留文件索引值00000。

[0105] 如果请求整个文件,则不须要使用字节范围,这与是否其中包含一个或多个片断无关。如果文件中存储有多个片断,而且客户端请求该文件中包含的所有片断的子集,则须使用字节范围。这一规则适应于流播过程涉及的所有HTTP GET请求,其中包括直播流情况。

[0106] 在实施例之一中,为了暂停或停止,客户端通过发送HTTP GET请求即可停止请求更多数据。为了续播,自下一个片断开始,客户端在上一次所请求的片断之后发送HTTP GET请求以请求片断。

[0107] 在实施例之一中,为了定位具体位置、快进或倒退,客户端始于上述具体位置处起始的片断发送HTTP GET请求,以请求当前其它展现方式的请求片断。注意:定位操作可仅定位在其中片断起始的具体位置。为了定位到具体的倒退位置,如果客户端已自上述具体位置缓冲数据,则它可自上述具体位置回放数据。

[0108] 在实施例之一中,为了以具体的回放速度执行快进操作,则可进行以下步骤。首先,客户端停止请求当前其它展现方式的数据。第二,客户端选择合适的其它视频快进展现,该展现拥有与目标回放速度最为接近的帧率。第三,客户端自上一次所请求的片断之后的临时位置请求所选的其它展现的片断。在实施例之一中,客户端以指定的回放速度播放媒体。

[0109] 在实施例之一中,为了以具体的回放速度执行快退操作,则可进行以下步骤。首先,客户端停止请求当前其它展现方式的数据。第二,客户端选择合适的其它视频快退展现,该展现拥有与目标回放速度最为接近的帧率。第三,客户端自上一次所请求的片断之后的临时位置请求所选的其它展现的片断。在实施例之一中,这要求在生成其它视频快退的过程中,视频帧的解码顺序与显示顺序相反。在实施例之一中,客户端以指定的回放速度播放媒体。

[0110] 在实施例之一中,为了通过自一种替换性展现A切换至另一种替换性展现B而执行流播自适应,会执行以下步骤。首先,客户端停止请求当前其它展现方式的数据。

[0111] 第二,如果客户端从未接收到B的任何片断,则客户端首先请求B的第一片断,再继上一次所请求片断之后,立即请求始于临时位置的B的片断。同时客户端存储B的第一片断,以便日后在自其它任何替换性展现切回至B的情况下使用。如果客户端已接收到并存储了B的第一片断,则会跳过对第一片断的请求。使用存储方法2时,客户端继上一次所请求的片断之后即可请求始于临时位置的B的片断。

[0112] 在实施例之一中,对于直播流,以下规定适应于某些实施例。首先,除开第一片断(其包含零媒体样本),片断持续时间恒定,例如,constant\_segment\_duration不等于0。

[0113] 第二,如果每个片断都存储在单独的文件中,则MPD不包含每个文件的元素,即,MPD在直播流会话过程中是静态的(不会发生变化)。这意味着constant\_num\_segments\_

per\_file不等于0,codec\_mime\_type\_included\_for\_each\_file应等于0,constant\_segment\_duration不等于0,以及byte\_offset\_included等于0。在此情况下,其中包含其它每次展现最后片断的文件的URL应为其它展现的URL前缀和五个十进制数位99999之间的链接。或者,其中包含其它展现最后片断的文件的URL可为其它展现的URL前缀和特殊字符串(如,"last\_segment")之间的链接。在实施例之一中,如果有文件包含多个片断,则num\_segments\_aligned设置为真。

[0114] 在一个直播流实施例中,设置具有以下步骤。首先,客户端获取MPD的URL。第二,客户端通过使用HTTP GET请求获取MPD。使用存储方法2时,客户端获取其中包含"giants' moov'窗的文件,而在未包含MPD的情况下,客户端获取MPD。

[0115] 第三,客户端解析MPD并选择其中包含客户端所需的所有媒体类型的另一次或其它多次合适展现方式。使用存储方法2时,客户端会在选择其它合适的展现方式的过程中解析其中包含"giants' moov'窗的文件并使用文件及/或MPD中的信息。

[0116] 第四,客户端请求所选的其他每一展现展现方式的第一片断,再请求所选的其他每一展现方式的最后一个片断。使用存储方法2时,客户端直接请求所选的其他每一次展现的最后一个片断。

[0117] 第五,如果每个片断存储在单独的文件中(即constant\_num\_segments\_per\_file等于1),则客户端定期检查与下一个文件索引值对应的下一个文件的可用性,并在可获取文件的情况下,客户端通过使用不具有字节范围的HTTP GET请求,请求下一个文件。在此情况下,如果下一个文件不包含'moof'窗(例如,文件为空),则客户端应断定直播流会话已结束。否则(多个片断可能存储在一个文件中),客户端通过使用具有开放式字节范围的HTTP GET请求,自之前请求最后一个字节后定期请求MPD中已更新的部分,再请求下一个可用的片断。在此情况下,如果下一个片断为空(即其不包含任何媒体样本),则客户端应断定直播流会话已结束。客户端可使用独立的TCP链接用于请求MPD,而使用另一个TCP链接用于请求片断。对下一个文件或下一个片断的可用性定期进行检查的时间,应不少于而应约近片断持续时间,或等于片断持续时间。

[0118] 在实施例之一中,为了暂停或停止,客户端通过发送HTTP GET请求即可停止请求更多数据。为了续播,客户端发送HTTP GET请求,以请求所选的其它每一展现的最后一个片断,再执行上述的第五步。

[0119] 在实施例之一中,为了定位到具体倒退位置,客户端始于上述具体位置处起始的片断发送HTTP GET请求,以请求当前其它展现方式的片断。在某些实施例中,定位操作定位在其中片断起始的具体位置。如果客户端已自上述具体位置缓冲数据,则它可自上述具体位置回放数据。

[0120] 在实施例之一中,快退对于直播流和按需流播发挥同样的作用。

[0121] 在一个直播流实施例中,为了通过自一种替换性展现A切换至另一种替换性展现B而执行流播自适应,会执行以下步骤。

[0122] 首先,客户端停止请求当前其它展现方式的数据。第二,如果客户端从未接收到B的任何片断,则客户端首先请求B的第一片断,再请求B的最后一个片断。同时客户端存储B的第一片断,以便日后在自其它任何替换性展现切回至B的情况下使用。如果客户端已接收到并存储了B的第一片断,则会跳过对第一片断的请求。使用存储方法2时,客户端仅请求B

的最后一个片断。

[0123] 第三,如果每个片断存储在单独的文件中(即constant\_num\_segments\_per\_file等于1),则客户端定期检查与下一个文件索引值对应的下一个文件的可用性,并在可获取文件的情况下,客户端通过使用不具有字节范围的HTTP GET请求,请求下一个文件。在此情况下,如果下一个文件不包含'moof'窗(例如,文件为空),则客户端应断定直播流会话已结束。否则(多个片断可能存储在一个文件中),客户端通过使用具有开放式字节范围的HTTP GET请求,自之前请求最后一个字节后定期请求MPD中已更新的部分,再请求下一个可用的片断。在此情况下,如果下一个片断为空(即其不包含任何媒体样本),则客户端应断定直播流会话已结束。客户端可使用独立的TCP链接用于请求MPD,而使用另一个TCP链接用于请求片断。对下一个文件或下一个片断的可用性定期进行检查的时间,应不少于而应约近片断持续时间,或等于片断持续时间。

[0124] 在实施例之一中,在MPD中,可包括媒体片断的字节范围和时间范围,以便客户端能高效地请求仅使用字节范围的媒体片断。

[0125] 在另一个实施例中,MPD可包括媒体其它展现形式的其它编解码信息,以支持采用一种以上编码配置而进行编码的媒体内容。在实施例之一中,MPD可包括所有片断的最大长度,使得在MPD中无需字节范围的信令即可使用字节范围,并大大缩减MPD的大小,这意味着流播过程的启动迟延降低。

[0126] 在另一个实施例中,为了表明直播流会话结束,服务器可采用异常方式组成下一个预期文件或片断,例如,清空文件或片断。在实施例之一中,为了成功调至直播流会话并开始请求最新内容,流播服务器中可用的且其中包含最新片断的文件使用了一个特殊的文件名称,继而使用了一个特殊的URL。在实施例之一中,为了让客户端计算出其寻找一个具体临时位置时待启动的具体文件,而此时分片持续时间保持不变,以及MPD中每一个分片都没有任何字节偏移或时间偏移信令,文件的URL生成后可具有指明文件开始回放时间的功能。在实施例之一中,提供了可启动诸如设置、暂停、续播及停止等一般流播流程的高效流播流程,并提供了搜索、快进、快退及媒体流自适应流程。

[0127] 虽然详细描述了目前的实施例及其优势,但请理解这一点:在不背离专利申请中定义的发明实质和范畴的情况下,可进行各种变化、变动和替换。例如,上述的许多特性和功能可在软件、硬件、固件或它们的组合中实施。

[0128] 另外,本申请的范围并不局限于规格中描述的流程、机器、制造、物质成分、装置、方法和步骤的特定实施例。这些流程、机器、制造、物质成分、工具、方法或步骤,不管是目前已存在还是有待日后开发,只要是能够与本文描述的相应的实施例发挥本质上相同的功能或取得本质上相同的结果,都可以根据本发明而予以采用,作为相关技术中的一个普通技巧。本发明披露后,技术人员应对这一点有所理解。相应地,随附的权利要求书旨在将这些流程、机器、制造、物质成分、工具、方法或步骤纳入权利要求的范围内。

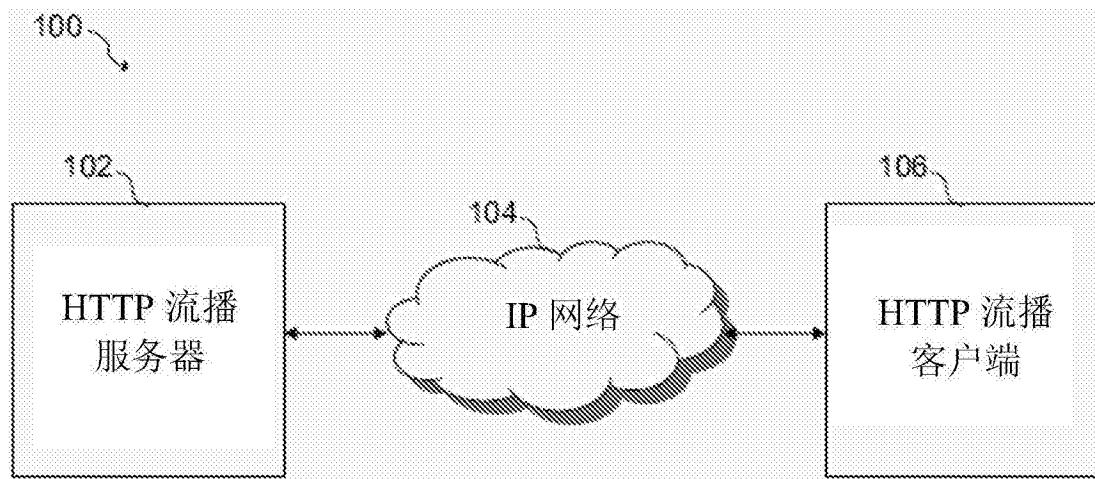


图1

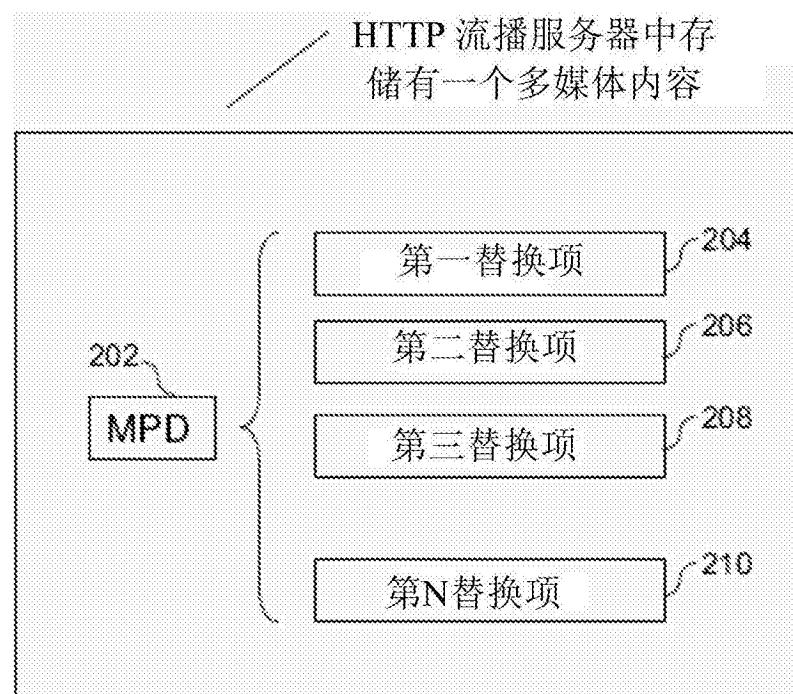


图2

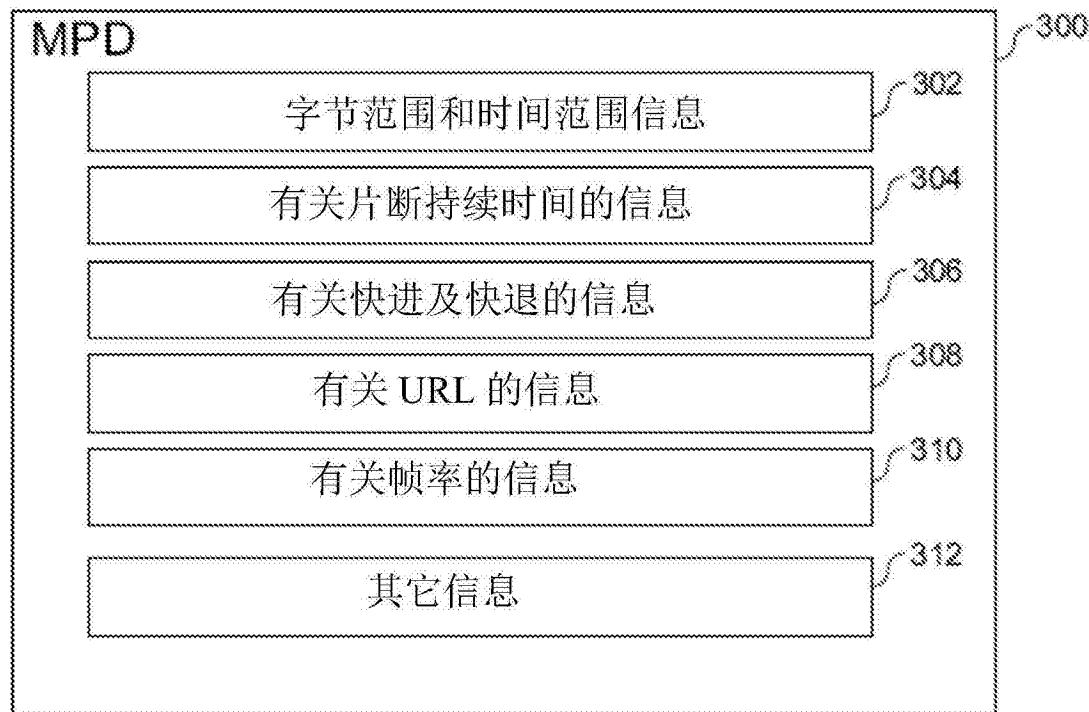


图3

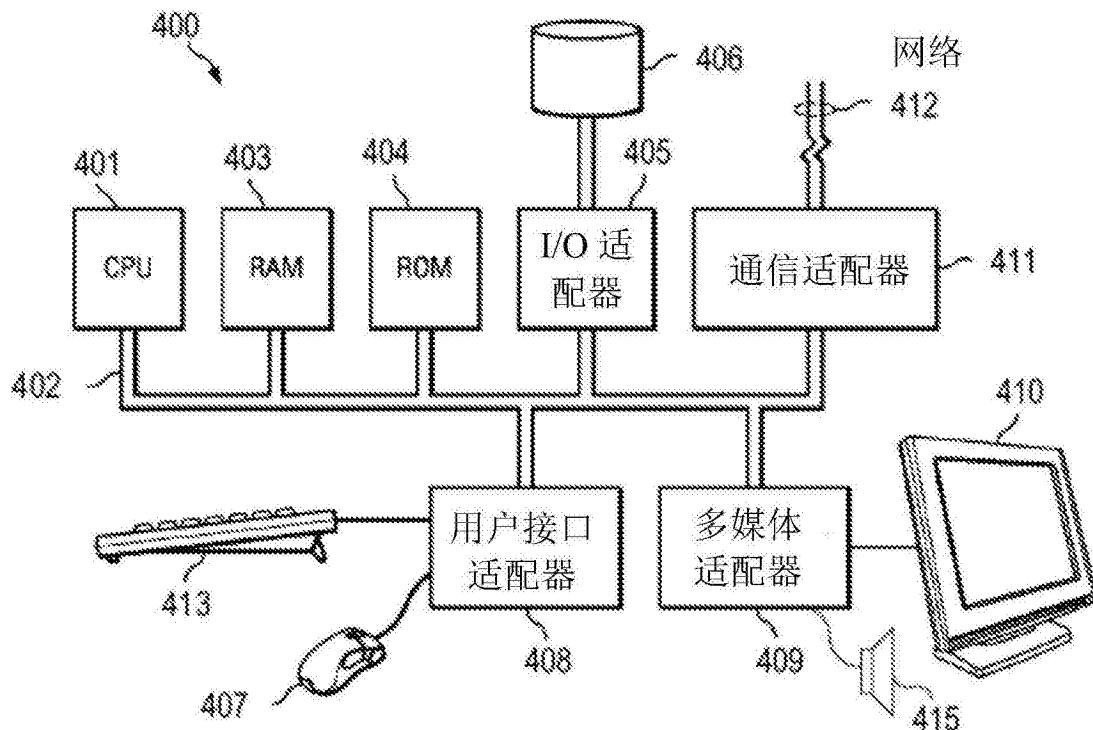


图4