(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104205768 A
(43)申请公布日 2014.12.10

(21)申请号 201380015992.9
(22)申请日 2013.02.26
(30)优先权数据
61/603,569 2012.02.27 US
13/745,799 2013.01.19 US
(85)PCT国际申请进入国家阶段日
2014.09.23
(86)PCT国际申请的申请数据
PCT/US2013/027807 2013.02.26
(87)PCT国际申请的公布数据
WO2013/130473 EN 2013.09.06
(71)申请人 高通股份有限公司
地址 美国加利福尼亚
(72)发明人 Q·高 M·G·卢比 Y·毛
L·C·明德
(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 张扬 王英
(51)Int.Cl.
H04L 29/06(2006.01)

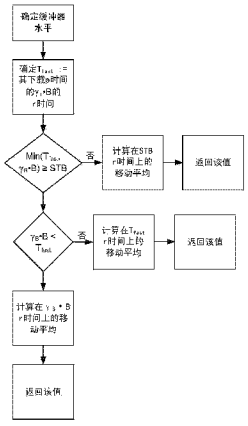
权利要求书2页 说明书40页 附图32页

(54)发明名称

利用下载速率估计器的改善的DASH客户端和接收机

(57)摘要

客户端设备呈现流媒体，并包括用于对流进行控制的流管理器、用于对内容进行网络请求的请求加速器、耦接到流管理器和请求加速器的用于确定进行哪些请求的源组件、网络连接以及媒体播放器。提供了一种用于速率估计的过程，该过程将快速地对接收速率改变做出反应。速率估计器可以使用自适应窗口平均，并且以一种方式考虑视频缓冲器水平和视频缓冲器水平中的改变，以便保证速率在需要时足够快地进行调节，并同时保持窗口宽度大小（并从而保持测量偏差）大小。一种保证可以是当速率发生下降或提高时，估计器在与缓冲器消耗速率或缓冲器填充水平成正比的时间内调节其估计。



CN 104205768 A

1. 一种对通过具有有限带宽的网络路径耦接到数据源的接收机处的下载速率进行估计的方法，其中，所述下载速率是在所述接收机处可以通过所述网络路径接收数据的速率，所述方法包括：

监测所述接收机的呈现缓冲器，其中，所述呈现缓冲器至少在接收到媒体数据的时间和由与所述接收机相关联的呈现元件消耗所述媒体数据的时间之间存储所述媒体数据；

确定对所述下载速率的估计所要基于的非零估计时段；

存储所述估计时段上缓冲器水平的指示，其中，在给定时间的缓冲器水平至少近似地对应于在该时间所述呈现缓冲器被接收到的并且尚未被所述呈现元件消耗的所述媒体数据占用的程度；以及

使用所存储的指示作为所估计的下载速率的测量的一部分。

2. 根据权利要求 1 所述的方法，其中，所述呈现元件包括显示器和音频输出。

3. 根据权利要求 1 所述的方法，其中，对所述下载速率进行测量的所述估计时段具有以预定的比例因子与测量的缓冲器水平成正比的持续时间。

4. 根据权利要求 1 所述的方法，其中，所述估计时段的持续时间被取为与在进行测量时所述呈现缓冲器中的未消耗的媒体数据的字节数量成正比。

5. 根据权利要求 1 所述的方法，其中，所述估计时段至少部分地是将媒体添加到所述呈现缓冲器的添加速率的函数。

6. 根据权利要求 1 所述的方法，其中，所述估计时段与下载所述呈现缓冲器的预定部分所使用的时间成正比。

7. 根据权利要求 6 所述的方法，其中，所估计的下载速率是在先前的等于所述估计时段的持续时间上的平均下载速率。

8. 根据权利要求 1 所述的方法，其中，所述估计时段是下载所述呈现缓冲器的预定比例的内容的时间和所述呈现缓冲器中存在的所述媒体数据的呈现时间中的较小者。

9. 根据权利要求 8 所述的方法，其中，所估计的下载速率是在先前的等于所述估计时段的持续时间上的平均下载速率。

10. 用于由处理器执行以便对通过具有有限带宽的网络路径耦接到数据源的接收机处的下载速率进行估计的非临时性计算机可读介质，其中，所述下载速率是在所述接收机处可以通过所述网络路径接收数据的速率，所述非临时性计算机可读介质包括：

用于监测所述接收机的呈现缓冲器的程序代码，其中，所述呈现缓冲器至少在接收到媒体数据的时间和由与所述接收机相关联的呈现元件消耗所述媒体数据的时间之间存储所述媒体数据；

用于确定对所述下载速率的估计所要基于的非零估计时段的程序代码；

用于存储所述估计时段上缓冲器水平的指示的程序代码，其中，在给定时间的缓冲器水平至少近似地对应于在该时间所述呈现缓冲器被接收到的并且尚未被所述呈现元件消耗的所述媒体数据占用的程度；以及

用于使用所存储的指示作为所估计的下载速率的测量的一部分的程序代码。

11. 根据权利要求 10 所述的非临时性计算机可读介质，还包括：

用于连接到显示器和音频输出的程序代码。

12. 根据权利要求 10 所述的非临时性计算机可读介质，还包括：

用于使用以预定的比例因子与测量的缓冲器水平成正比的持续时间来计算对所述下载速率进行测量的所述估计时段。

13. 根据权利要求 10 所述的非临时性计算机可读介质，还包括：

用于计算与在进行测量时所述呈现缓冲器中的未消耗的媒体数据的字节数量成正比的所述估计时段的持续时间。

14. 根据权利要求 10 所述的非临时性计算机可读介质，还包括：

将所述估计时段至少部分地计算为将媒体添加到所述呈现缓冲器的添加速率的函数和 / 或与用于下载所述呈现缓冲器的预定部分的时间成正比。

15. 根据权利要求 14 所述的非临时性计算机可读介质，其中，所估计的下载速率是在先前的等于所述估计时段的持续时间上的平均下载速率。

16. 根据权利要求 10 所述的非临时性计算机可读介质，其中，所述估计时段是下载所述呈现缓冲器的预定比例的内容的时间和所述呈现缓冲器中存在的所述媒体数据的呈现时间中的较小者。。

17. 根据权利要求 16 所述的非临时性计算机可读介质，其中，所估计的下载速率是在先前的等于所述估计时段的持续时间上的平均下载速率。

利用下载速率估计器的改善的 DASH 客户端和接收机

[0001] 相关申请的交叉引用

[0002] 本申请要求享受 2012 年 2 月 27 日提交的、题目为“Improved DASH Client and Receiver with Rate Adaptation and Downloading for Adaptive Video”的美国临时申请 No.61/603,569 的权益，故出于所有目的，作为整体以引用方式将其全部内容并入本文。

背景技术

[0003] DASH 指的是“动态自适应 HTTP 流”。使用 DASH，内容提供商将内容格式化成分段、片段、表现 (representation)、改编 (adaptation) 等，连同相关联的元数据（诸如 MPD 文件），并将所有这些存储为通过标准 HTTP 服务器或者专用 HTTP 服务器可获得的文件。DASH 客户端是根据需要获得这些文件，以向该 DASH 客户端的用户提供呈现的接收方。

[0004] 由于用户通常在网络受到约束的环境中，在很少或没有提前通知的情况下，想要高质量的流，因此 DASH 客户端具有苛刻的限制。从而，改进的 DASH 客户端令人期望。

发明内容

[0005] 客户端设备呈现流媒体，并包括用于对流进行控制的流管理器、用于对内容进行网络请求的请求加速器、耦接到流管理器和请求加速器的用于确定进行哪些请求的源组件、网络连接以及媒体播放器。请求加速器包括：用于对请求进行缓存的请求数据缓冲器、以及用于向其可以响应的每一个请求返回完整的响应的逻辑。可以将流管理器、请求加速器和源组件实现成处理器指令或程序代码，客户端设备还包括程序存储器、工作存储器、处理器和电源。此外，客户端设备还包括显示器和用户输入设备。在源组件、流管理器和请求加速器之间对客户端任务进行解析，以便对数据进行高效地流传输。

[0006] 在各个方面，如本申请所描述的，客户端可以执行诸如下面的操作：确定何时维持一个表示或者切换到另一个表示，确定请求哪些片段，并确保媒体播放器可以获得充足的数据（在大多数状况下），以便在没有停滞的情况下继续流媒体播放。

[0007] 提供了一种用于速率估计的过程，该过程将快速地对接收速率改变做出反应。下载速率估计器可以使用专门为在媒体流中使用而设计订制的自适应窗口平均。下载速率估计器以一种方式考虑视频缓冲器水平和视频缓冲器水平中的改变，以便保证速率在需要时足够快地进行调节，并同时保持窗口宽度大小（并从而保持测量偏差）大小。该保证由过程通过以下方式来提供：(a) 如果 B 是缓冲器中视频数据的量（以播放时间的秒数为单位），当速率发生下降时，则估计器将调节缓冲器消耗到 B/2 所消耗的时间内的速率估计，以及 (b) 如果 B 是缓冲器中视频数据的量，当速率发生提高时，速率估计器足够快地调节到新的速率，使得可以在至多 3*B 的时间内看到该调节（提供智能速率改变过程）。

[0008] 在示例性方法或装置中，一种接收机监测所述接收机的呈现缓冲器，其中，所述呈现缓冲器至少在接收到媒体数据的时间和由与所述接收机相关联的呈现元件消耗所述媒体数据的时间之间存储所述媒体数据；确定对所述下载速率的估计所要基于的非零估计时段；存储所述估计时段上缓冲器水平的指示，其中，在给定时间的缓冲器水平至少近似地对

应于在该时间所述呈现缓冲器被接收到的并且尚未被所述呈现元件消耗的所述媒体数据占用的程度；以及使用所存储的指示作为所估计的下载速率的测量的一部分。

[0009] 所述所述呈现元件可以包括显示器和音频输出。对所述下载速率进行测量的所述估计时段具有以预定的比例因子与测量的缓冲器水平成正比的持续时间（诸如与在进行测量时所述呈现缓冲器中的未消耗的媒体数据的字节数量成正比持续时间），和/或所述估计时段是将媒体添加到所述呈现缓冲器的添加速率的函数和/或与下载所述呈现缓冲器的预定部分所使用的时间成正比，诸如当所估计的下载速率是在先前的等于所述估计时段的持续时间上的平均下载速率时。

[0010] 所述估计时段可以是下载所述呈现缓冲器的预定比例的内容的时间和所述呈现缓冲器中存在的所述媒体数据的呈现时间中的较小者。所估计的下载速率可以是在先前的等于所述估计时段的持续时间上的平均下载速率。

[0011] 可以使用用于由处理器执行，对通过一个网络路径耦接的源和接收机之间的所述网络路径上的数据下载进行控制的计算机可读介质，来实现各个单元。该计算机可读介质可以是非临时性计算机可读介质。

[0012] 通过本说明书，本发明的其它方面应当是显而易见的。

附图说明

[0013] 图1示出了DASH部署中的包括DASH客户端的各个组成部分，显示了媒体记录如何到达终端用户，其中涉及记录、内容准备以及内容递送阶段。

[0014] 图2示出了具有不同组件的DASH客户端的示例性架构，其包括流管理器、请求加速器、源组件、网络连接和媒体播放器。

[0015] 图3是示出表现切换过程的时序图，其中包括用于回看过程的图3A和用于快进过程的图3B。

[0016] 图4是示出针对切换点对齐的情况的表现切换过程的时序图。

[0017] 图5是示出如由速率估计器所管理的速率随时间变化的图，特别地，该速率估计器是自适应于缓冲器水平的估计器（诸如pker型速率估计器））。

[0018] 图6是示出当使用非自适应指数加权移动平均（“EWMA”）过滤器时，速率增加对比下载时间（r时间）的图。

[0019] 图7是示出当使用非自适应EWMA过滤器时，速率增加对比播放时间（p时间）的图。

[0020] 图8是示出当使用可变窗口大小加权移动平均（“WMA”）过滤器时，速率增加对比下载时间（r时间）的图。

[0021] 图9是示出当使用pker型过程时，速率增加对比播放时间（p时间）的图。

[0022] 图10是示出当使用来自2.1章节的pker型过程时，速率增加对比下载时间的图。

[0023] 图11示出了速率上突然增加的pker过程的行为。

[0024] 图12示出了突然速率下降的pker过程的行为。

[0025] 图13示出了简单（固定宽度的）移动窗口平均与指数加权移动平均的对比。

[0026] 图14是pker速率估计过程的流程图。

[0027] 图15示出了如何根据记录的(Tp,Tr)值的历史连同图16，来确定由pker过程使

用的值 B 和 T_{fast}。

[0028] 图 16 示出了对值进行确定的方面。

[0029] 图 17 示出了“水印”获取过程的行为。

[0030] 图 18 示出了如可以用于选择播放速率的 lambda 和 mu 函数的示例。

[0031] 图 19 示出了使用“保守”设置的 (lambda，mu) 函数的示例性选择。

[0032] 图 20 示出了使用“中等”设置的 (lambda，mu) 函数的示例性选择。

[0033] 图 21 示出了使用“积极”设置的 (lambda，mu) 函数的示例性选择。

[0034] 图 22 示出了使用用于在一定程度上仿真 MLB 过程的过程的 (lambda，mu) 函数的示例性选择。

[0035] 图 23 示出了用于 lambda 设置的并排值的示例。

[0036] 图 24 示出了用于 mu 设置的并排值的示例。

[0037] 图 25 示出了用于进行速率估计、然后进行基于速率的速率选择、然后进行基于缓冲器管理的速率选择的过程。

[0038] 图 26 示出了在没有请求取消的情况下的速率下降。

[0039] 图 27 示出了在具有请求取消的情况下的速率下降。

[0040] 图 28 是示出示例性请求取消过程的流程图。

[0041] 图 29 示出了用于请求取消检测的过程。

[0042] 图 30 是利用多个 TCP 连接但没有接收缓冲器调整的获取的行为的图。

[0043] 图 31 是利用多个 TCP 连接并利用接收缓冲器调整的获取的其它行为的图。

[0044] 图 32 是示例性请求加速器过程的流程图。

[0045] 图 33 示出了用于发现多个子请求，以有助于给定的片段请求的过程。

[0046] 图 34 示出了用于选择各个请求的过程，其中这些请求被选定为具有计算出的大小的源请求的不相交时间间隔。

[0047] 图 35 示出了时间偏移以及通过时间偏移所确定的用于修复分段的片段结构的示例。

[0048] 图 36 包括可以用于速率选择中的 lambda 和 mu 的值的表。

具体实施方式

[0049] 本申请所阐释的 DASH 客户端包括流管理器 (SM)、请求加速器 (RA)、源组件 (SC)、网络连接和媒体播放器，如图 2 中所示。DASH 客户端还可以包括一个或多个媒体数据缓冲器。在一些实现中，RA、SC 和媒体播放器均可以具有其自己的数据缓冲器，或者具有一个大型数据缓冲器的逻辑分区。在其它实现中，或许仅 RA 具有用于缓存请求的数据缓冲器，使得 RA 能够向其可以响应的每一个请求发送完整的响应，并且媒体播放器使用 SC 已建立的任何数据缓冲器。SM 可以具有其自己的本地存储 (物理的或逻辑的)，以便根据需要自己决定来存储元数据。

[0050] 图 1 示出了具有 DASH 客户端的 DASH 部署。

[0051] 图 2 示出了具有不同的组件的 DASH 客户端的示例架构。应当理解的是，SM、RA、SC 和媒体播放器可以在硬件、软件或某种组合中实现。因此，在将功能归结于组件的情况下，其可以实现为处理器指令、程序代码等，在这种情况下，暗含必要的硬件来执行这些指

令（程序存储器、ROM、RAM、处理器、电源、连接器、电路板等）。在描述网络功能时，网络连接应当被理解为是存在的，并且其可以是有线、光纤、无线等，并且当暗示用户交互时，还暗含用户接口能力（显示器、键盘、触摸板、扬声器、麦克风等）。

[0052] DASH客户端维持两个时钟或其逻辑等同物。一个时钟是指示在该客户端中运行的本地时钟的时间的实时时钟电路或软件，而另一个时钟是呈现时间，其表示媒体内容相对于其起始的呈现时间。在本文中，将实时时钟时间称为“r 时间”，而“p 时间”是表示呈现时间的描述符。

[0053] 表现是针对相同的内容，以不同的比特率或者其它差别进行编码的媒体流。因此，用户通常将仅需要一个表现，但客户端可以根据情况和 / 或需求变化，从一个表现切换到另一个表现。例如，如果带宽较高，则流客户端可以选择高质量、高比特率表现。如果带宽降低，则客户端可以通过切换到较低质量、较低比特率的表现来适应这些情况。

[0054] 切换点（或随机接入点）是表现中的一些样本，可以从这些样本开始对媒体样本的解码，而不需要该流之前的数据的知识。具体而言，在视频表现中，由于样本（帧）通常依赖于先前的帧，因此并不是每个样本都是随机接入点。当流客户端想要切换表现时，其应当确保在切换点开始对新的表现进行解码，以避免做无用功。在一些情况下，在分段索引 (sidx) 中以信号方式将切换点发送给流客户端。

[0055] 表现组（有时简称为组）是可切换的表现的集合。一个媒体呈现可以包含一个以上的表现组。例如，针对以不同比特率的视频表现可以具有一个表现组，并且针对音频比特率可以具有另一个表现组。在 DASH 标准中，有时还将表现组称为自适应集。

[0056] 分段是包含针对这些表现中的一个表现的至少一部分中的媒体数据的文件。片段是分段的一部分，其中，从该片段的起始 p 时间到该分段中的该片段的字节范围的映射是可获得的。有时，使用术语子分段来代替片段，这些术语可以被视作为是等同的。一些媒体内容不被分割在片段中 ；在这种情况下，“片段”可以指代分段其自身。

[0057] 图 3 是示出两种可能的表现切换过程的时序图。该切换可以是回看（第一过程 ；图 3A），在该情况下，通过以下方面来寻找切换到的表现中的切换点 ：查找已经在切换自的表现中请求的 p 时间伸缩，并选择在 p 时间上从切换到的表现中向后选择最靠近该伸缩的结束的前一切换点。第二过程（图 3B）是快进 ：其从在切换自的表现中的最后请求的 p 时间处开始，在 p 时间上向前寻找切换到的表现中的下一个切换点。

[0058] 图 4 是示出当切换点对齐时并且当一个切换点紧接在最后请求的片段之后时进行切换的过程的时序图。该图描绘了回看和快进方法二者的行为，这两个过程在这种设置下是表现相同的行为。因此，当这些切换点对齐时，以上的任一过程都不必下载重叠的数据。

[0059] 呈现时间是预期通常按照正常的速度，来播放或者重放媒体的时间周期。例如，30 分钟视频呈现将播放 30 分钟。用户可以对进行快进或快退，这将改变实际花费的时间，但应当理解的是，该呈现仍然是 30 分钟视频呈现。呈现单元在呈现时间上向用户提供该呈现。呈现单元的示例包括视频显示和音频显示，或者被管道化传输到可以对其进行呈现的设备的视频 / 音频流。“播放”是用于描述媒体的消费的术语。例如，智能电话可以下载或者获得在一个呈现的呈现时间 (p 时间）上表示该呈现的媒体数据，对其进行缓存，媒体播放器“消费”该媒体，优选地消费是使得至少在该呈现时间的结束之前，该缓冲器不是完全

地为空，使得当接收机等待获得更多的数据时，用户不会体验到呈现的停滞。当然，“重放”或者“播放”并不暗示对媒体播放一次以上。在很多实例中，其可以是一旦被媒体被消费一次，就不再使用该媒体。

[0060] 呈现缓冲器是接收机、媒体播放器或者可访问一个或二者的部件中的内存单元。为了简化说明，本申请交换地使用术语“呈现缓冲器”、“缓冲器”、“媒体缓冲器”和“播放缓冲器”，并理解其是包括已被下载但还没播放或者消费的数据（通常为媒体数据）的逻辑缓冲器。其可以是下面的情况：将包括呈现缓冲器的数据划分在一个设备中的不同组件之间，即，所下载的数据的一些部分由一个进程（例如，该设备中的接收进程）进行保持，而其它部分可能已经被传送给另一个进程（例如，该设备中的播放进程）。此外，其还可以是下面的情况：可以在不同的进程的不同缓冲器中，对包括呈现缓冲器的数据的至少一部分进行至少部分地重复。在一些情况下，不是所有已被下载但仍没有播放的数据，都被视作为仍然位于呈现缓冲器之中，例如，在一些情况下，一旦将该媒体内容传递给媒体播放器，就不再将其视作为位于呈现缓冲器之中。通常，已被下载但仍没有播放的媒体数据，以及被视作为不位于呈现缓冲器之中的媒体数据（如果有的话）的数量非常的小。

[0061] 呈现缓冲器容纳不均发接收和播放媒体，存储所接收的媒体数据，直到其被消费为止。在媒体数据被消费之后，可以根据配置，将其删除或者继续保存。在一些实现中，呈现缓冲器的大小（如通过可以在该呈现缓冲器中存储的数据的字节数量所测量的）可以随时间发生变化。例如，可以根据需要，从共享内存中动态地分配呈现缓冲器。

[0062] 在本申请所详细描述的很多示例中，可以假定通过大小来示出呈现缓冲器的特性。在固定的存储器大小专用于呈现缓冲器的情况下，可以通过能在可用的存储器中存储的字节数量来测量该大小。当呈现缓冲器是动态分配的时，针对呈现缓冲器的“大小”属性可以等于目前分配给该呈现缓冲器的字节数量、可以分配给该呈现缓冲器的最大字节数量、或者某种其它适当的测量值。有时，还依据呈现缓冲器中当前可用的媒体的呈现播放持续时间，来测量呈现缓冲器的大小。

[0063] 此外，呈现缓冲器还具有另一种特性：其“水平”或者“填充水平”。呈现缓冲器的水平表示在该呈现缓冲器中存在多少未被消费的媒体数据（例如，使用字节或者呈现持续时间来测量的）。可以预期的是，随着接收到媒体数据，该水平增加，随着媒体数据被消费，该水平下降。该水平也可以仅是一种逻辑量，例如，呈现缓冲器可以是不断充满有媒体数据，但该媒体数据中的一些（例如，已被消费的媒体数据）被进行了标记，以便接收到新媒体数据时进行覆盖。可以对一些接收机进行编程，使得“空缓冲器”是存在零个未消费的媒体数据的状况，“满缓冲器”是该呈现缓冲器的100％都填充有未消费的媒体数据的状况。其它接收机可以具有其它限制，使得该水平范围处于与呈现缓冲器大小的0％到100％相比更小的范围之上。在使用共享内存，并且仅当存储未消费的媒体数据时，才向呈现缓冲器进行分配的情况下，使用呈现缓冲器的这种动态分配的内存大小作为在指示水平比率时的分母，可能没有任何意义，这是由于根据规定，该呈现缓冲器始终是满的。更适合的是，可以将该呈现缓冲器的水平测量成：该呈现缓冲器中的未消费媒体数据的数量除以该呈现缓冲器的最大允许大小之比。

[0064] 1、客户端组件的概述

[0065] 再次参见图1-2，示出示例性客户端的各种组件。

[0066] SC 保持元数据（例如，关于哪些表示是可用的、以及其片段都是有哪些之类的信息）的跟踪。此外，SC 还负责对通过网络接收的媒体数据进行缓存，以便将其传递给媒体播放器。SM 负责决定在什么时间点将下载哪些表示，以及进行速率切换决定。最后，给定精确的 URL 以及如 SC 所提供的字节范围信息，RA 负载下载媒体片段。

[0067] SM 是负责进行速率切换决定的软件组件。SM 的目标之一是针对给定的情形，挑选最佳的内容。例如，如果有大量可用的带宽，可以实现高下载速率，则 SM 应当挑选高速率表示。如果下载速率明显地下降，所选择的高表示不再是可维持的，则 SM 应当切换到更低的表示速率，更适合于当前状况。SM 应当足够快地切换速率，以避免使播放缓冲器完全地耗尽（由于这将造成播放停滞），但同时尝试不要切换的太匆忙或者太频繁。此外，应当将目标设定为请求可以通过网络下载、并且在没有停滞的情况下进行播放的最高质量内容。SM 在其决策过程中，可以扩展到考虑不同于下载速率的其它因素。其可以潜在地考虑诸如电池寿命、显示器尺寸之类的事物、以及当关于表示进行决策时的其它因素。可以将这些另外的约束作为过滤器增加到 SM 中，但其不影响本申请所描述的基本速率决策计算。

[0068] 现在描述客户端的典型、高水平操作。假定用户请求一个特定的媒体内容，例如，实时体育广播、预录制的电影、音频流或者其它音视频内容或者其它内容，其可以涉及不同于视频和音频的媒体类型。客户端或许通过用户接口或者计算机接口，向 SM 提供该请求。SM 从 SC 进行请求，并接收关于下面信息的指示：有哪些表示可用、哪些 p 时间跨度被哪些片段覆盖、以及这些表示中的切换点位于什么位置。除了该信息之外，SM 可以具有关于在处理的短期下载速率的某种信息（如下面所解释的），RA 将该数据报告给 SC，SC 将其报告或者提供给 SM。

[0069] SM 使用该信息，以及过去的历史，估计可持续的速率，选择一个表示中的适当切换点，以及从该切换点起始、要从该表示下载的媒体内容的量。随着下载的进行，对媒体内容进行播放，SM 使用提供的信息来决定速率切换是否恰当。如果速率切换不恰当，则 SM 告诉 SC 继续从当前表示获取片段。如果速率切换是恰当的，则 SM 寻找潜在的切换点，决定需要从哪些表示中的哪些片段进行获取，以进行期望的切换。随后，SM 将该信息传递给 SC。SC 和 SM 之间的这种信息交换是定期进行的（只要应当关于要下载的下一节做出决定）。为了做出正确的决定，SM 对缓冲器水平进行监测，在一些情况下，SM 可以决定该缓冲器是足够的满，在一段时间之内不需要再下载任何片段。

[0070] 一旦 SM 决定要下载一个片段，SC 就负责获得 RA 以实际地下载该片段，将所下载的片段保存在媒体缓冲器中，最后当媒体缓冲器中的媒体数据的播放时间到了时，将其传递给媒体播放器。

[0071] 在 SM 告诉了 SC 进行下载之后，SM 就不再活动地涉及到这些片段之中。但是，即使在给定的片段的下载已经开始之后，SM 仍然可以改变其决定，取消其先前已发出的片段请求。这种功能在下面的情形中是有用的：事实证明下载速率极速地下降，截止到媒体缓冲器被完全地耗尽的时间，被下载的片段也不可能是可用的。如果发生这种状况，则 SM 应当负责检测，取消该请求，并切换到更适当的速率。

[0072] 一旦 SC 从 SM 接收到要获取的片段句柄，就在其数据结构中查找该 URL 和相应的片段的字节范围，使用该信息来产生向 RA 传递的请求。此外，SC 还负责从 RA 获取响应数据，将所接收的媒体片段转换成可播放的流。最后，SC 负责对元数据（例如，从 MPD 获得的

数据、分段索引（sidx）盒、或者苹果的HTTP实时流（HLS）情况下的播放列表）进行解析和保持跟踪。

[0073] RA是从SC接收片段和元数据请求，生成相应的HTTP请求，并通过网络连接来发送这些HTTP请求，获取相应的响应并将其传递回SC的组件。网络连接可以是因特网连接、基于蜂窝的连接、WiFi连接或者能够处理HTTP请求和响应的其它网络连接。该网络连接可以是在单一设备的内部，即，其可以是针对已经在该设备中高速缓存的媒体数据的内部接口。此外，还可以存在多种组合，即，媒体内容中的一些可以从有线因特网连接下载，一些通过基于蜂窝的连接来下载，一些通过WiFi连接来下载，一些来自于本地高速缓存。在一些情况下，下载媒体数据的连接可以是混合的，即，一部分是通过蜂窝、一部分通过WiFi、一部分通过有线等。在一些实例中，具体的请求可以是不同于HTTP，但当服务于该媒体内容的服务器是HTTP服务器时，HTTP是优选的。

[0074] 在其最简单的形式，RA是HTTP客户端。但是，可能期望RA与通用HTTP客户端相比更高效。RA的一个目标是实现足够高的下载速率；其应当将目标设定于：与所选定的播放媒体速率相比，下载速率要明显地更快。另一方面，还应当注意不要为了原始吞吐量而使时效性受到处罚：与处于进一步的后面位置的其它片段相比，不久将被播放的片段是更加急迫的，RA应当尝试及时地接收这些片段。因此，需要为了时效性而牺牲一些吞吐量。应当将RA设计为在全部合理的网络状况下能进行很好地工作。

[0075] RA的一种基本设计方案是使用一些连接和可能的FEC（前向纠错）来获得最佳的结果。因此，RA通常需要对一个以上的开放HTTP连接进行管理。RA将请求派发到这些连接上。在一些环境中，RA可以将请求分割成一组更小的请求。当接收到相应的响应时，RA随后将该数据重新组装成一个相干的响应。换言之，RA负责决定要发送的HTTP请求的颗粒度、将这些请求派发到哪些连接、以及决定要请求源片段或者修复片段的哪些部分。这些请求的颗粒度可以取决于多种事物，例如，缓冲器水平、请求的急迫性、可用连接的数量等。

[0076] RA发送出的每一个请求是针对于元数据的HTTP请求，也可以是针对于SC已传送给该RA的片段请求的一部分或者全部。其可以是针对源媒体数据的请求，也可以是针对根据源媒体数据所生成的修复数据的请求。在大多数情况下，对于根据SC片段请求所生成的RA请求的响应，应当是足够用于重建该片段请求中的所有媒体数据，随后RA可以将其传送回SC。因此，RA负责将来自于与媒体片段请求相关联的RA请求的响应，组合回一个针对该片段请求的响应，以提供给SC。RA执行的这种组合可以包括FEC解码（例如，如果存在一些针对FEC修复数据的RA请求的话）。

[0077] 除了对HTTP请求进行管理之外，RA在短期时段上、在一些采样速率的时间片上，对下载速率进行测量。一种示例性采样速率是100ms，即，RA在100ms时段上对下载速率进行测量。SM使用该使用数据来计算其下载速率估计，并最终做出速率决策。其它采样速率也是可以的。

[0078] RA不需要知道关于诸如DASH媒体呈现描述（MPD）或者关于分段结构的元数据。在特定的实现中，RA使用HTTP栈实现的几个同时实例来在一些连接上（甚至在一些情况下，在针对相似或者不同服务器的不同类型的连接上）实现HTTP检索。

[0079] RA负责使SC知道何时可以接受新的请求。SC调用SM来确定要请求的下一个片段，向RA提供该适当的请求。此外，RA还提供一些状态信息。RA通常可以通过SC向SM提

供短期的下载速率、下载所花费的总时间。此外，SM 还可以针对该信息，通过 SC 来间接地向 RA 进行轮询。除了该方面之外，RA 还向 SM 通知关于各个请求已完成的百分比信息。使用 SM 进行调用以获取该信息的 API，来类似地提供该信息。

[0080] 在 RA、SC 和实际的媒体管道之间，应当存在非常紧急的数据，故尽可能地在 RA 或 SC 中缓存很少的数据（除了故意的媒体缓冲器之外）。相同的理由对于具有各种形式的 HTTP 请求来说也是成立的；与通过网络来发送实际相应的 HTTP 请求的时间相比，SM 应当仅只提前很短的时间，必须决定要请求的片段。一种原因在于：SM 必须决定一个请求的提前量更早，保持最新的信息就越不准确，因此其所做出的决策的质量就越低。

[0081] SM 一次发出一个请求。但是，如果所有先前的请求都没有完成，SM 还可以发出新请求；允许同时的请求。SC 将这些请求按照 SM 发出其顺序来传送给 RA。随后，RA 对同时处理保持注意，确保其将接收的数据返回给 SC。

[0082] 同时的请求使 RA 可以实现 HTTP 管道化。事实上，甚至利用多个连接的 RA 也适应这种方案。

[0083] 1.1、流管理器 (SM)

[0084] 响应于用户动作、网络状况和其它因素的组合，SM 确定何时对片段进行请求、以及对哪些片段进行请求。当用户决定开始观看内容时，SM 负责针对该内容，确定要请求的第一片段（其开始于用户或者所提供的服务所指定的 p 时间）。例如，一些实时流服务可能需要所有用户在相同的 r 时间观看该媒体内容的相同的 p 时间部分，而其它实时流和按需服务可以允许终端用户或者应用具有关于在什么 r 时间对什么 p 时间进行播放的灵活性。当媒体缓冲器变满时，SM 临时地暂停提供另外的片段请求。SM 负责根据网络状况和其它因素（例如，显示器的大小、剩余的电池寿命等），来决定在每一个 p 时间点，按照什么质量来播放该内容。

[0085] 当 SM 认为其适合于提供片段请求时，SM 可以只提供一个请求（如果 RA 准备好接收和处理片段请求）。SC 通过对 RA 进行轮询来确定这种情况，并将该信息转发给 SM。

[0086] 当 RA 准备好接收下一个请求时，SM 决定是否应当发出新的请求，并选择下一个片段进行请求。SM 一次针对媒体数据一个片段进行请求，SM 负责请求片段，其允许及时地和无缝地播放该内容。表示中的播放改变，通常只在切换点处发生，在两个连续的切换点之间可以存在多个片段；SM 尊重该限制。

[0087] 通常，SM 只尝试请求有理由认为应当及时地接收以平滑播放的片段。但是，假定网络状况有时可以非常快速地急剧变化，但并不能在所有环境中都保证这种情况。因此，SM 还具有取消请求的能力。如果检测到拥塞，SM 将取消请求，如果不采取任何动作，则有很大的停滞风险。如果不采取任何动作，很可能发生停滞，例如，如果由于在发出片段请求之后不久网络状况就恶化，而造成的下载速率突然急剧地下降。

[0088] SM 对表示 R、以及大多数最近先前选择的片段的结束 p 时间 E 保持跟踪。通常，SM 选择请求起始 P 时间为的下一个片段。一些变化可以是：起始时间根据缓冲器水平和当前播放时间来确定。

[0089] 如果丢弃切换点处的潜在重叠，则 SM 产生旨在生成可以平滑地播放的流的请求序列。SM 生成请求的顺序，与 RA 应当对其划分优先级的顺序相同（虽然不一定是问题）。其还可以是 RA 将所接收的数据传递回 SC、并且 SC 应当对其进行播放的相同顺序。

[0090] 如果 SM 决定其需要切换速率，则在通常情况下，这样做有两个过程。在一个过程中，SM 在新的（“切换目标”）表示中寻找 p 时间小于或等于 E 的切换点（其有时还称为“随机接入点”或“RAP”），一旦识别了这种点，SM 就开始请求该新表示中的片段。第二过程是寻找 p 时间小于或等于 E 的切换点 P，并继续请求旧的（“切换源”）表示中的片段，直到请求了结束时间超过 P 的片段为止的过程。在任一情况下，将这种切换信息发送给 SC 都是有用的。

[0091] 应当注意，这两种过程都具有必须要下载一些重叠的数据的属性。存在着 p 时间的伸缩，需要在该时间，下载从其切换自的表现和切换到的表现的数据。

[0092] 这些切换过程中的哪个是有利的，取决于上述情形。例如，其可以是：在一些特定的情形下，用于这些过程中的一个的重叠是不合理地大，而对于另一个来说却是很短。在所有片段在表示之间都是对齐、并且所有这些片段都以 RAP 起始的简单情形下，这些切换过程简化为更简单的方法，其中在该方法中，SM 只通过从切换到的表现而不是从从其切换自的表现中请求下一个片段来进行切换。此外，还应当注意，在该情况下，不需要下载重叠的数据。

[0093] 1.1.1、SM 片段决策过程

[0094] 该节描述了 SM 片段决策过程，以决定告诉 SC 要请求哪些片段。在这些示例中，假定一个简单的表示组，但这些示例可以被扩展为解决使用多个表示组的过程，例如，从视频表示组中选择一个视频表示，从音频表示组中选择一个音频表示。

[0095] 通常，SM 所选择的下一个片段的起始 p 时间是前一个片段请求的结束 p 时间。下面描述可以在 SM 中实现以便选择要请求的下一个片段的某种详细逻辑。

[0096] 在下面的示例中，假定片段以 RAP 起始，并且在表示之间是对齐的。如果不是这种情况，则本说明书的变化也是有可能的。如果存在这些状况，则 SM 的片段决策降低为速率决策，即，SM 决定是停留在当前表示上，还是切换到不同的表示。在更一般的情况下，片段不是必需在表示之间是对齐的，并且可以不以 RAP 起始，这种决策是类似的，但切换的代价更高，在切换时可以考虑该代价。

[0097] SM 表示处理包括两个逻辑单独的过程：第一过程是速率估计器，其根据 RA 提供的短期样本，计算近似持续的下载速率，第二过程是利用该估计来做出切换决策的决策过程。

[0098] 2、速率估计过程

[0099] 自适应比特率流媒体客户端通常使用下载速率估计器模块，速率决策模块使用该下载速率估计器模块来选择正确的比特率媒体。使用该方法，当下载速率较大时，可以对高质量媒体进行流传输。下载速率的改变可以触发表现切换。速率估计的质量对于流媒体客户端的质量具有很大的影响。

[0100] 用于自适应视频流媒体设备的良好速率估计器应当具有多种属性。首先，其应当具有很小的方差，即使短期下载速率变化很大。第二，其应当在底层的信道上进行快速地速率改变。当信道速率显著下降时，该估计应当反映这种快速的事实，使得设备可以在无需停滞的情况下，对质量进行相应地调整。相应地，应当快速地观测到视频质量的增加，使得可以获取更佳的质量内容。

[0101] 满足这两种需求可能需要一些平衡。通常，具有很小方差的估计器将具有较大的反应时间，反之亦然。例如，假定可以在设备中使用的简单估计器。该估计器对最后 X 秒的

下载（对于某些固定的 X），执行移动平均。挑选较大的 X（例如，X ＝ 30 秒），将导致具有很小方差的相对平滑估计，但其将只反应为下载速率慢速地改变。如果这种估计器是用于速率决策，则最终的播放器可能由于带宽下降而频繁停滞，或者不能及时地切换到更高比特率（当其可以安全地这样做时）。由于这些原因，一种实现可以挑选较小的 X，如 X ＝ 3s。这种选择将导致更快速的速率调整，但以稳定性为代价。速率估计变化很大，因此播放器可能非常快速地改变视频播放速率，其导致很差的用户体验。

[0102] 在图 5 中，颠簸曲线是原始下载速率，其具有很多的短期波动。速率估计器是颠簸下载速率的平滑版本。在速率改变时，其收敛到新的持续速率，只要速率不发生改变，则仍然类似于它。

[0103] 期望的一种属性在于：如果存在很小的缓冲器水平，则调整是快速的，其造成速率的快速调整，使得当下载速率下降时，呈现缓冲器在调整之前不为空。另一方面，如果在媒体缓冲器中存在很多的媒体数据，则速率估计应当是具有更慢调整的更加平滑。与媒体缓冲器中存在很少的媒体数据相比，当在媒体缓冲器中存在更多的媒体数据时，播放速率应当趋向于在下载速率下降时，在更长的时间段保持为较高。

[0104] 下文所给出的速率估计过程（其称为 pker、pker 过程或者 pker 类型过程），针对速率改变进行快速地反应，但其还是稳定的，满足针对低方差和高反应的需求。

[0105] 2.1、pker 过程

[0106] 该节描述了本申请称为 pker、pker 类型过程或者仅“pker 过程”的速率估计过程。基本速率估计器将其估计唯一地基于短期速率测量值，根据该值，使用一种方法或另一种方法来计算较长的平均运行。如上所述的基本移动窗口平均（“MWA”）是该过程的一个示例。

[0107] 图 6-7 示出了为了速率选择目的，使用非自适应（固定系数）指数加权平均的效果。为了简单起见，这些图假定新的速率估计立即触发新的下载选择（即，这些片段是相对较小的），新的速率选择仅是速率估计。

[0108] 图 6 示出了 r 时间方面。如图所示，x 轴是下载时间（实际时间）。当在时间 T1 处发生急剧的速率增加时，缓冲器非常快速地开始增长，这是由于与播放的速率相比，视频数据被更快速地下载。EWMA 估计逐渐收敛到真实的速率。

[0109] 图 7 示出了相同事件的 p 时间方面。在该图中，线 702 描述了在屏幕上显示的比特率。与图 6 的 r 时间图片相比，这里的速率调整是更慢速的。与 r 时间相比的 p 时间的收敛速度，下降了开始中的 NR/OR 的因子（这是由于在该时间点的每一秒的下载，播放器接收了大约 NR/OR 秒的视频）。因此，当使用这种类型的速率估计器时，净效应是媒体可以在显著数量的 p 时间，按照与下载速率相比更慢的速率进行播放。

[0110] 如果为了流媒体的目的来对速率进行估计，则估计器可以利用其它有关的信息。具体而言，对媒体播放器的缓冲器感兴趣，或者通常对媒体播放器的下载历史感兴趣（与当前缓冲器中的内容相比，对更远的过去感兴趣），其包括花费了多长的信息来下载被缓存或者已经播放的各个媒体片段。

[0111] 例如，一种实现可以使用 MWA 估计器，但根据媒体缓冲器来选择窗口大小。

[0112] 如果媒体播放器的缓冲器水平较高，则播放器不会具有立即停滞的风险，所以可以使用较大的窗口来进行长期的估计，这将导致一个更稳定的估计。另一方面，如果缓冲器

水平较低，则播放器应当快速地反应，建议在这种情况下，更短的平均窗口是一种更好的选择。

[0113] 所以速率估计过程的实现可以使用变化的窗口宽度，其利用与当前媒体缓冲器中的 p 时间的数量成比例的 r 时间窗口宽度（也就是说，已下载但没有播放的当前数量的 p 时间）。

[0114] 另一种实现可以对窗口宽度进行选择，以便与媒体缓冲器中当前包含的字节数量成正比。

[0115] 一种实现还可以检查缓冲器的内容，而不是只检查其水平。例如，如果确定缓冲器的很大部分是用与该相同内容的播放持续时间相比更短的时间来下载的，则说明该下载缓冲器正在快速地增长，速率估计器因此可以推断需要对估计进行调整。

[0116] 类似地，速率估计器可以对缓冲器水平的改变速率进行跟踪，将缓冲器水平的快速改变作为需要对速率估计进行快速地调整的指示。

[0117] 图 8-9 示出了当使用可变窗口大小加权移动平均（“WMA”）过滤器时，与图 6-7 相同的场景中的行为。在这些示例中，如同这种可变窗口大小 WMA 过滤器，将“pker”过程解释为程序代码。可以将 pker 过程实现成由处理器执行的程序指令。

[0118] 在图 8 中，线 802 是底层信道的速率突然从速率 OR（旧速率）增加到速率 NR（新速率）的情况下的 pker 速率估计。为了进行速率选择以调整到新速率所花费的 r 时间的量与 OR/NR 成正比。增加越大，则这种调整在实际时间中将更快速地发生。如图所示，在时间 T2 处，Buff@T2 = 2*Buff@T1，T_{fast} = OR/NR*Buff@T1。

[0119] 图 9 显示了 p 时间中的播放行为。pker 估计器花费大约一个缓冲器持续时间（当发生速率增加时，处于该缓冲器中的 p 时间的数量）来调整到新速率，即，在媒体缓冲器具有增加到该媒体缓冲器的 p 时间持续量 B 的媒体内容量的时间之前，pker 估计器调整到新速率，其中 B 是在速率增加到新速率的时间，该媒体缓冲器中的媒体内容的 p 时间持续量。

[0120] 现在将描述执行该处理的具体过程。该处理确定其花费多少 r 时间来下载播放缓冲器的最后 γ_T 比例，其中 γ_T 是适当选择的常量。例如，这可以是其下载全部的当前播放缓冲器（$\gamma_T = 1$）所花费的全部时间，或者其下载播放缓冲器的最后一半（$\gamma_T = 0.5$）所花费的时间。还可以发生 $\gamma_T > 1$。使 T_{fast} 是其下载播放缓冲器的最后比例所花费的 r 时间的量。可以通过对下载时间的前面 T_{fast} 秒上的下载速率进行估计，来计算估计的下载速率。应当注意，其它的 γ_T 值也是可以的。如本申请所解释的，不同的值可以服务于不同的目标。

[0121] 这种类型的在 T_{fast} 宽度窗口上的加窗口平均，具有检测速率快速地增加的卓越性能。事实上，如果使用值 $\gamma_T < 1$ 来确定 T_{fast}，那么估计器具有下面的属性：如果当媒体缓冲器中的媒体内容的 p 时间持续量是 B 时，该速率在某个时刻增加了任何因子的倍数，则在速率估计器收敛到增加的速率之前，该缓冲器将至多增长到 B 的有限倍数。

[0122] 更详细的速率估计方法可以对上面所提及的两种方法进行组合。具体而言，可以将缓冲器水平 B 和 T_{fast} 的最小值使用成平均的窗口宽度（即，在其上对下载速率进行平均的 r 时间的量）。通常而言，可以在 $\gamma_B \cdot B$ 和 T_{fast} 的最小值的前面 r 时间上，对下载速率进行平均，其中 γ_B 是适当选择的常量。这种选择具有下面的属性：当存在具有停滞风险的速率下降时，其将快速地反应，这是由于在这些情况下，B 是最小值，在与媒体缓冲器中的媒体内容的 p 时间持续量成正比的 r 时间上进行平均，从而在媒体缓冲器消耗一半的时间，速

率估计将是新的速率。例如，假定在速率减少的时间，媒体缓冲器中的媒体内容持续时间是B，下载速率减少，使得在该下载速率减少之前下载速率是所选定的呈现的播放速率的一个比例 $\alpha<1$，悲观的是，所选定的表示的播放速率不会减小，直到速率估计减少到新的下载速率为止。随后，随着在发生速率减少时，持续下载超出该时间 x 的 r 时间，则缓冲器水平是 $B' = B - x + \alpha \cdot x$，即，从媒体缓冲器中消耗 x 的 p 时间，向该媒体缓冲器下载了 $\alpha \cdot x$。速率估计将是此时的新速率，使得 $x = B'$，即，在此时，p 时间中的媒体缓冲器水平等于下载已经处于该新速率的 r 时间，这是由于此时，前面 r 时间的下载上的估计将是新速率，在该全部时间期间，下载都已处于新的速率。对于方程 $x = B' = B - x + \alpha \cdot x$ 中的 x 进行求解，得出 $x = B' = B/(2-\alpha)$，即，当缓冲器 B' 仍然至少为 B/2 时，速率估计将达到新速率。如果在此时，速率增加的非常明显，那么 T_{fast} 将是最小值，与前面的B r时间上的平均值相比，在前面的 T_{fast} r 时间上的平均下载速率明显更高。

[0123] 现在基于这种构建，来给出 pker 速率估计过程的一个示例的详细描述。其使用可以从下载模块（例如，请求加速器 (RA)）获得的短期速率测量和缓冲器信息来计算估计。使用缓冲器信息来确定短期速率测量值能获得有用的估计的窗口宽度。

[0124] 图 10 示出了当下载速率急剧下降时，pker 速率估计器如何演进。只要速率一下降，缓冲器水平就开始下降。速率估计也开始调整。在缓冲器水平下降到二分之一时，速率估计达到新的速率 (NR)。在该示例中，不进行中间速率决策，所以缓冲器线性地下降。如果进行了中间决策，则缓冲器水平的下降将逐渐慢慢地下降。

[0125] pker 过程的设计目标是足够大的平均窗口来避免具有嘈杂的数字，但对于反应来说具有足够短的数量。pker 过程通过使用具有动态改变的窗口大小的加窗口平均来实现该目标。RA 在存储器中维持几个变量，以便由 pker 过程使用，其包括 B、播放缓冲器的水平（在 p 时间）、处理参数 γ_B、γ_T 和 T_{fast}、下载该缓冲器的最后 γ_T 比例（在 p 时间）所花费的 r 时间的节省量、以及 R、在 r 时间的下载的最后 C 的持续时间上平均下载速度，其中 $C = \max(STP, \min(\gamma_B \cdot B, T_{fast}))$，STP 是最小可接受窗口大小（其应当超过采样时间周期（如，100ms））。在一些实施例中，$\gamma_B = 1$，$\gamma_T = 0.5$，但其它值也是可以的，并导致性质相似的行为（只要这两个值是正在的，并且 $\gamma_T<1$）。较小的 γ_B 造成 pker 过程对于速率减少反应的很快，而较小的 γ_T 造成其对于速率增加反应的很快。

[0126] 如本申请所解释的，为了计算在持续时间 C 上的下载速率，SM 使用 RA 定期提供的下载速度信息。由于此目的，SM 可以保持 RA 所提供的下载速度信息的历史。进行该平均的持续时间至多是 γ_B 缓冲器持续时间，其有效地限制当存在关于媒体缓冲器水平的上限时，需要保持多久的历史信息。

[0127] 应当注意，如果所选定的播放速率近似地等于下载速率，则缓存值 C 具有缓冲器持续时间的量级，这是由于如果与播放该流所花费的时间相比，对该流下载花费相同数量的时间，则可以具有 $T_{fast} = \gamma_T \cdot B$。对于用于下载速率估计的平滑时间间隔来说，选择某种量级的 r 时间的缓冲器水平是一种自然选择，由于其是远见量，因此流媒体客户端必须决定是否想要避免停滞。

[0128] 在一种简单的实现中，平均的窗口宽度与 B（视频缓冲器中所包含的 p 时间的量）成正比。这种选择能非常好地防止停滞，但具有下面的缺陷：如果下载速率是所选定的媒体的速率的 k 倍，则每一秒的下载导致 k 秒的 p 时间的媒体被下载，其造成速率估计实际调整

地很慢。例如，如果 k = 10，并且存在 10 秒的缓冲器，那么在调整之前，速率估计器将下载大约 k•10s = 100s 的 p 时间，其是一段非常长的时间。这促使向 pker 方法引入了 T_{fast} 参数。事实上，如果使用指数加权移动平均进行平滑，事情可能甚至变得更差，这是由于这些过滤器具有无限冲激响应。由于此原因，pker 过程替代地使用有限冲激响应。简单移动平均工作；一种实现还可以使用更精密的加权移动平均。

[0129] 图 13 示出了这最后一点。其示出了简单（固定宽度）移动窗口平均与指数加权移动平均的比较。该图示出了当观测到速率改变时，固定窗口移动平均首先慢慢地收敛到新速率，但其将一个窗口持续时间之内收敛。指数加权移动平均趋向于在开始时快速地移动，但在后面阶段，其只是慢慢地收敛。与加窗口的移动平均不同，不在固定的窗口之内收敛，而是利用速率改变的幅度的时间对数进行收敛。

[0130] 在 $\gamma_B = 1$ 和 $\gamma_T = 0.5$ 的情况下，pker 过程可以提供各种保证。对于一种情况，如果下载速度下降任何倍数，则在缓冲器收缩到其原始持续时间的一半的时间之内，将估计调整到新的下载速度。对于另一种情况，如果下载速度增加任何倍数，则在 pker 过程收敛到新的速率之前，对至多值得另外的 p 时间的一个缓冲器进行下载。简单明了的计算将示出类似的恒定比例担保，持有 $0<\gamma_B$ 并且 $0<\gamma_T<1$ 的任何选择。

[0131] 用于计算缓冲器水平 B 的一种方法如下所述。使 T 是媒体播放器的当前播放 p 时间，使 $F_{i,1},\cdots,F_{i,n}$ 是表示组 i 中的已经被下载或者正在下载但还没有播放的片段，其以增加的起始时间来排序。仍然被下载的组 i 的任何片段，位于 $F_{i,1},\cdots,F_{i,n}$ 之间。使 $\alpha(F_{i,j})$ 是已经被下载的片段 $F_{i,j}$ 的比例，例如，已经下载的片段 $F_{i,j}$ 的字节数除以片段 $F_{i,j}$ 的字节大小。RA 可以计算用于各种 i 和 j 的 $\alpha(F_{i,j})$ 的值，并将其传递给 SM。对于给定的组 i，将下载的 p 时间的当前总数量规定成如式 1 中所示：

[0132] $$T_{p,i} := \text{starttime}(F_{i,1}) + \sum_{j=1}^{N_i} \text{duration}(F_{i,j}) \cdot \alpha(F_{i,j}) \qquad (式\ 1)$$

[0133] 为了根据式 1 的结果来计算整体 T_p 值，DASH 客户端考虑每一个组的加权因子 w′，后者是根据 MPD（媒体呈现描述元数据）和表示组的数量 G 来确定的，DASH 客户端执行式 2 的计算。随后，将缓冲器水平 B 规定为 B: = T_p - T。

[0134] $$T_p := \sum_{i=1}^{G} w_i' \cdot T_{p,i} \qquad (式\ 2)$$

[0135] 式 2 还捕获属于当前正播放的缓冲器的一部分。应当注意，如果一次下载几个片段，则这种规定也起作用。

[0136] 为了计算 T_{fast}，SM 通常保持一些历史。使 T_r 是 RA（尝试）下载媒体所花费的 r 时间的总数量，使 Z 是 RA 下载的字节的总数量。RA 对 T_r 的值进行计算。SM 保持按照规则时间间隔（例如，每 100ms）所采样的三元组 (T_r^i, Z^i, T_p^i) 的历史 H，其中 i = 1,2,…,K，第 K 个观测值是最后一个。假定以观测顺序来存储该历史；所以具有 $T_{p,j}^1 \le T_{p,j}^2 \le \cdots \le T_{p,j}^K$，以及 $T_r^1 \le T_r^2 \le \cdots \le T_r^K$ 和 $Z^1 \leqslant Z^2 \leqslant \cdots \leqslant Z^K$。

[0137] 现在，为了计算 T_{fast}，假定已经使用上面所给出的方法计算得到了 B。随后，RA 确定 j，使得满足式 3 的不等式（例如，通过使用二进制搜索，对历史进行搜索）。

[0138] $$T_p^K - T_p^{j+1} < \gamma_T \cdot B \le T_p^K - T_p^j \quad (式3)$$

[0139] 则$T_{fast} := T_r^K - T_r^j$。应当注意到，不需要保持无限的历史，只需要 T_i 值足够跨度超过最大缓冲器持续时间的 γ_B 比例。

[0140] 图15(连同图16的放大的变量)示出了如何根据记录的(T_p, T_r)值的历史，来确定pker过程使用的值B和T_{fast}。该附图示出了r时间和p时间同样进展快速的情况(不存在下载中断)，因此播放时间(p时间)是下载时间(r时间)的45度斜线。在该图中画出了(T_p, T_r)值的历史，其导致严格地高于播放时间线的曲线(如果不发生播放停滞的话)。转而，缓冲器水平B是最后记录的T_p值与播放时间的差值。可以通过下面方式在该图中观测到T_{fast}的值：测量在当前(最后)T_p值之下的 $\gamma_T \cdot B$ 电平处，与(T_p, T_r)曲线的水平距离。

[0141] 图11使用了与图15-16相同类型的呈现方式，来示出pker过程对于速率的突然增加的响应。当接收速率观测到播放器还没有进行反应的突然增加时，T_{fast}是相对很小的。其示出了对于高接收速率的快速响应。应当注意，平均窗口完全地位于该图形的高速率部分之内，这是由于其是相对地较窄。因此，在该时间点，pker估计已收敛到更大的速率。

[0142] 图12再次使用图15的呈现方式，来示出针对速率下降的可变窗口大小WMA过滤器(例如，pker)响应。在该情况下，T_{fast}变得相对地较大，但缓冲器耗尽，所以B变得较小，其造成在某个消耗时间之后，平均窗口完全地落入到低速率区域。如图所示，平均窗口的宽度B是使得B小于T_{fast}，但在缓冲器被完全耗尽之前，估计仍然收敛到新的更低的速率。

[0143] 图14是pker速率估计过程的流程图。

[0144] 一旦计算出T_{fast}和B的值，下面的C值将容易计算，最后步骤是在过去的持续时间为C的窗口上计算速率R。由于该目的，使用历史中的Z^i和T_r^i值。

[0145] 为了计算在时间间隔C上的速率，SM或者RA执行下面的操作：(1)寻找最大的j，使得$T_r^K - T_r^j \ge C$；(2)随后计算平均下载速率，如式4中所示。如果在第一步骤中不存在这种j，则SM或者RA设置j:＝0(即，最旧的已知观测值)。可以通过二进制搜索，来高效地确定j的值。

[0146] $$R := \frac{Z^K - Z^j}{T_r^K - T_r^K} \quad (式4)$$

[0147] 每一个组具有相关联的权重w，后者与该组预期使用的总带宽的部分相对应。其取决于MPD所提供的信息，优选地是在过滤完不可用的表示之后。这里，所提议的组g的权重w的规定是w(g):＝maxrate(g)+minrate(g)，其中maxrate()是组g中的最大播放速率，minrate()是最小速率。

[0148] 对于权重w而言，SM或RA可以计算归一化的权重w′，如下所述。假定客户端想要对组1,…,G进行流处理，那么归一化的权重是所述权重除以所有权重之和，如式5中所示。

[0149] $$w_i' = \frac{w_i}{\sum_{j=1}^{G} w_j} \quad (式5)$$

[0150] 这种归一化趋向于关于实际进行流处理的权重来进行。例如，如果存在不被进行

流处理的一个组，则就不应当考虑该组。

[0151] 在 pker 过程的操作中，采取了一些假定。例如，应当将各个表示组的缓冲器水平保持的相对接近。用此方式，pker 过程的效果更好。例如，假定一个组具有很大的缓冲器，另一个组具有很小的缓冲器，二者具有类似的权重。在该情况下，需要快速地调整速率估计，这是由于对于小缓冲器而言，需要在状况发生改变时避免停滞。但 pker 过程仍然容易地平滑其估计，如同针对更大的缓冲器所操作的。相反，对于较大的缓冲器而言，这些测量值具有某种高方差（由于缓冲器水平所允许的），并因此导致紧张的速率决定。

[0152] 在一些情况下，表示组在缓冲器水平中具有很大的差值是不可避免的。由于此原因，另一种实现可以使用 pker 方法的变型，后者当一些缓冲器非常小时，更快速地调整速率，因此在这些情况下更好地防止比特停滞。这种实现可以用与先前所描述的相同方式来计算 T_{fast}，但将窗口大小设置为 $C = \max(STP, \min(T_{fast}, T_{p,1} - T, T_{p,2} - T, \cdots, T_{p,N} - T))$。

[0153] 这些下载速率估计的其它变型包括：针对每一个表示组，使用独立的 pker 估计来针对该组做出决策。

[0154] 3、获取策略

[0155] 通常，流视频播放器具有有限的媒体缓冲器。因此预期在通常的操作中，可以最后达到缓冲器满状态。当缓冲器达到满状态时，流模块应当控制媒体输入，以避免该缓冲器的溢出。一种做到这一点的简单方法是：只要该缓冲器满了就进行等待，直到该缓冲器被消耗足够的多，以便能够保存下一个片段为止，随后继续进行取数。

[0156] 这种方法的效果是对每一个片段进行单独地获取，因此在每一个片段请求之间存在一个时间间隙，即用于对缓冲器消耗足够的多，使得能适应下一个片段并对其进行请求而花费的时间量。

[0157] TCP 协议基于当前网络状况，自动地调整其下载速率。当通过 TCP 连接来发起下载时，初始下载速率通常非常低，随着 TCP 协议探测到是否可以实现更高的下载速率，而下载速率发生增加。TCP 如何快速地增加下载速率、以及 TCP 通常如何对端到端 TCP 连接的属性进行反应，是相当复杂的，并且取决于多种因素，这些因素包括：固有的端到端网络时延、沿着 TCP 传输和确认路径的网络单元的缓冲器容量、沿着这些路径的竞争流量、在使用 TCP 的哪种变型等。通常，TCP 以较慢的下载速率进行开始，随着时间来增加其下载速率，因此该 TCP 连接在整个下载时间上的平均下载速率只接近可持续 TCP 下载速率（当整个下载时间是巨大的时）。例如，如果可持续 TCP 下载速率是 1 兆 / 秒，该 TCP 连接按照下载速率基本为零进行开始，并且在一秒内随时间线性地增加到 1 兆 / 秒，那么在第一秒上的平均下载速率是 500K 比特 / 秒，平均下载速率达到可持续下载速率的 95％要花费 10 秒的下载。由于此原因，在请求之间具有多个下载间隙的获取策略是不理想的，其中这些下载间隙是一个下载请求的完成和下一个下载请求的开始之间的时间段。即使当下载请求之间的间隙为零也不是理想的，这是由于：通常，在前一个请求的完成之后，TCP 要花费一段时间来爬升到针对下一个请求的下载速率。在每一个间隙之后，必须实现新的可持续吞吐量，其减少了整体实现的平均下载速率。

[0158] 这种速率的减小可能导致更小的速率估计，并因此选择更小的媒体速率。转而，这通常导致更小的媒体片段被下载（依据字节大小而言），其进一步增加所述间隙的相对幅度，导致潜在地选择甚至更小的播放速率。换言之，这种效果是自放大。

[0159] 因此，DASH 客户端实现使用使这种问题的影响减到最小的处理是有利的。

[0160] 一种实现可以连续地下载媒体数据，随后定期地对缓冲器水平进行消耗，如下所述。只要请求的但没有播放的 p 时间的数量超过预置的高水印 M_h，那么 SM 不再发出任何请求，直到缓冲器水平下降到低于低水印 M_l 为止。在特定的实现中，M_h = 20 秒，M_l = 10 秒，但在其它实现中，这些值可以更低，也可以更高。在下降到低于低水印之后，重新继续正常的操作，SM 开始再次发出片段决策。

[0161] 另一种实现可以使用以字节而不是呈现时间来指定的水印，来实现类似的效果。

[0162] 系统的其它部分可以以有利于自己的方式，使用缓冲器被定期地耗尽的事实。例如，其可以用于获得 RTT 的新鲜估计，如第 6.1.2 节中所解释的。

[0163] 图 17 示出了一种“水印”获取过程的行为。上图是可见到交替模式的消耗时期和获取时期的缓冲器水平图。在下图中显示出下载速率。在每一个获取时期的开始过程中，TCP 花费一些时间来达到可持续的最大速度，因此平均下载速率（在这些获取时期期间）小于最大可实现下载速率。低水印和高水印之间的差值越大，获取时期就越长，并且平均速率就越高。

[0164] 4、速率选择过程

[0165] 当开始请求媒体数据时，流模块 (SM) 使用某种方法来做出第一播放速率选择。可以选择最低可用的速率，或者可以例如保持网络状况的历史，随后基于该历史，确定在没有停滞的情况下，可以维持的播放速率的估计以便进行选择。当 SM 已经接收到数据，并且在其处理时具有速率估计 R（例如，使用来自节 2 的方法所计算的速率估计中的一个），那么其可以做出决定是停留在该速率，还是对表示进行改变。

[0166] 现在描述一种简单的速率决策过程。接收机确定播放速率低于所估计的下载速率 R 的最高带宽表示，挑选该表示作为用于播放（重放）数据的表示。虽然简单，但该方法具有多种问题。首先，其不自然地造成较小的媒体缓冲器增大，并因此容易受到停滞的影响（即使当下载速率仅变化很小时）。第二，变化的估计 R 将导致快速地改变速率决策，这可以是不必要的，并可以造成视觉错觉。第三，其导致启动时间至少近似地是一个片段的持续时间，并因此通常是几秒钟。

[0167] 因此，DASH 客户端可以实现一种速率决策过程，该速率决策过程将其速率决策不仅基于下载估计 R，而且还基于缓冲器水平 B（也就是说，已缓存但没有播放的 p 时间的数量）、以及取决于内容的变量，例如改变速率 D，后者是通常两个连续的切换点之间的 p 时间持续量的估计值。

[0168] 因此，一种实现可以将与 R 成正比的最大播放速率，挑选为该决策速率，其中该比例因子取决于缓冲器水平。

[0169] 通常，该比例因子 λ 是缓冲器水平的递增函数。例如，一种实现可以使 λ 是缓冲器水平的仿射函数。

[0170] 如果 λ 取决于缓冲器水平，那么一种实现可以在缓冲器为空或者较小时，选择使 λ 更小。这种选择是有利的，这是由于其将使得较小的缓冲器增大，并且当没有准确地预测下载速率时，还提供某种安全以防止停滞。

[0171] 对于较大的缓冲器水平而言，一种实现可以选择 λ 的值靠近、等于或者甚至超过 1。这将确保当不存在中间停滞风险时，选择高播放速率来进行下载，从而导致以稳定状态

来流处理高质量媒体。

[0172] 该速率决策过程可以将 λ 实现成 B 的分段仿射函数，而不仅只是简单的仿射函数。分段仿射函数可以将任意连续函数近似成任何所需的精确度，这使其进行一个合适的选择。可以替代地选择具有相同属性的任何其它参数化类型的函数。

[0173] 另一种实现使 λ 取决于以字节计量的缓冲器水平，而不是以 p 时间计量的缓冲器水平。

[0174] 另一种实现使 λ 不仅取决于缓冲器水平 B，而且取决于缓冲器水平 B 和切换机会的频率二者。这样做的原因在于：与具有对速率进行改变的更多机会的播放器相比，具有对速率进行改变的更少机会的播放器将各个决定进一步承诺到未来。因此在后一情况下，将各个决定承诺到更大的时间跨度，故其还具有更高的风险。当缓冲器水平 B 和估计的下载速率 R 相同时，建议在前一种情况而不是后一种情况下挑选低速率，以便将停滞的风险保持的很低。

[0175] 一种用于考虑速率切换机会的频率的速率选择过程的具体方式如下所述。使 D 是流中的两个连续切换点之间的 p 时间的典型量。D 的值取决于编码的视频，例如，可以将其值采取为两个连续切换点之间的 p 时间的最大距离、或者两个连续切换点的 90 个百分点的距离、或者该媒体中的两个连续切换点的 p 时间距离的任何其它适当测量值。给定这种 D，一种方法可以包括选择 λ 为 B/D 的分段仿射函数或者其变型（例如，B/max(u,D)or B/(D+u)），其中增加值 u 以考虑在发出请求时产生的开销。u 的值可以是较小的时间常数量（例如，100ms）。作为进一步的细化，一种实现可以使 u 为所估计的 RTT 的较小倍数。

[0176] 将其速率决策仅基于 λ·R 的处理（例如，上面所描述的方法），具有 R 的甚至相对很小差异也可能导致很多的速率切换的缺陷。这可能不是合乎要求的。当存在足够的缓冲器时，更佳的是，不是针对 R 的较小变化立即进行反应，而是使缓冲器水平相应地变化。

[0177] 为了获得这种行为，一种处理可以使用值 λ 和 μ、相同数量的两种函数（例如，B、B/D 或 B/max(100ms,D)，如上面所解释的），后者连同当前速率来挑选新的速率决策。应当以下面的方式来选择这些函数：λ·R 是低可接受速率选择，μ·R 是高可接受速率选择。随后，可以对该处理进行设计，以便使用这两个值作为用于良好速率决策的向导。

[0178] 在这种设置中，应当对这些函数进行选择，使得通常 $\lambda \leqslant \mu$。

[0179] 如果前一选择已经处于从 λ·R 到 μ·R 的范围之内，则该速率决策过程可以决定将速率保持不变。如果前一选择小于 λ·R，则选择等于或者小于 λ·R 的最大可用播放速率。如果前一选择大于 μ·R，则选择等于或者小于 μ·R 的最大可用播放速率。

[0180] 一种实现可以选择使函数 λ 和 μ 进行硬编码。替代地，其可以根据环境，以更精细的方式来选择这些函数。具体而言，一种实现可以根据客户端至多进行缓存的数量，来选择适当的 λ 和 μ 函数。对于点播内容而言，客户端可以选择对大量的数据（潜在几分钟的媒体数据）进行预缓存。对于低延时的直播内容，客户端可以永远只对至多该端到端延时所预订的媒体量（其可以只是几秒钟）进行缓存。对于具有很少缓存的内容而言，客户端可以决定挑选更保守的 λ 和 μ 函数（即，其具有更小的值）。

[0181] 例如，一种具体实现可以在两个外部函数 λ_1 和 λ_2 之间，线性地插入函数，其中所选定的插值点是低缓冲器水印 M_l（参见第 3 节）。所以其具有两个硬编码函数（λ_1 和 λ_2），其中 λ_1 用于较小的 M_l 的值（其小于某个 m_1），当对于某些值 m_1、m_2（其中 $m_1<m_2$），$M_l \geqslant m_2$

时，使用 λ_2。对于位于从 m_1 到 m_2 的范围之内的值来说，使用函数 $\lambda(x) := \lambda_1(x)(m_2-M_1)/(m_2-m_1)+\lambda_2(x)(M_1-m_1)/(m_2-m_1)$。

[0182] 现在接着上面的描述，给出速率决策过程的详细示例。为此，引入一些注释。

[0183] 1) 使 $S_1, S_2, \cdots, S_L$ 表示一个表示组中的 L 个可用的表示的流速率（以递增顺序给出）。

[0184] 2) 使 $\lambda(x)$ 表示分段线性函数，其采用非负标量作为输入，并返回非负实数缩放系数。函数 $\lambda(x)$ 应当可在编译时间进行设置，或者通过配置文件来设置。对于较大的 x（例如，对于 x 大于 M_1 来说），$\lambda(x)$ 应当不发生改变。

[0185] 这里给出了关于如何实现该函数的一个示例。给定角点 $(0, \lambda_0)$、(x_1, λ_1)、…、(x_N, λ_N)，其中 x_i 具有递增顺序。为了对 $\lambda(x)$ 进行评估，寻找最大的 i，使得 $x_i \leqslant x$。随后，使用式 6，接收机可以对该函数进行评估。

[0186] $$\lambda(x)=\begin{cases}\lambda_i+(\lambda_{i+1}-\lambda_i)\cdot\dfrac{x-x_i}{x_{i+1}-x_i}, & \text{if } i<N\\ \lambda_N, & \text{if } i=N\end{cases} \qquad (式\ 6)$$

[0187] 用于这种 $\lambda(x)$ 函数的适当示例是通过示例参数 N = 1,[(0,0.5),(3,1)] 所规定的函数，也就是说，在 x = 0 处等于 0.5 的该函数，直到 x 达到 3 为止都线性增加，在后一点，该函数等于 1，并且在其后仍然为 1。

[0188] 3) 使 $\mu(x)$ 是另一个这种分段线性函数。一种示例性该函数是在 x = 0 处评估为 0，在 x = 3 处达到 1.5，其后保持不变的函数。

[0189] 4) 使 D 表示从一个切换点到下一个切换点的 p 时间的持续时间的估计（如先前所指定的）。

[0190] 5) 使 x: = min{(Td - T),Ml}/max{D,1second)，其中 T 是当前播放 p 时间，Td 是进行速率决策的 p 时间，D 是如上面所给出的，Ml 是缓冲器水平低水印（参见第 3 节）。

[0191] 6) 使 CURR 是当前所选定的表示（即，在后一片段请求中使用的表示）。使 UP 是速率至多为 $\lambda(x)\cdot R$ 的最高比特率表示的播放速率，如果不存在这种表示，则 UP 是最低比特率表示的播放速率。使 DOWN 是速率至多为 $\mu(x)\cdot R$ 的最高比特率表示的播放速率，如果不存在这种表示，则 DOWN 是最低比特率表示的播放速率。由于通常 $\lambda(x) \leqslant \mu(x)$，因此通常 DOWN ≥ UP。

[0192] 转而，该速率决策过程挑选下一个片段的速率 NEXT，如下所述：(1) 如果 UP<CURR，那么 NEXT: = min(DOWN,CURR)；(2) 否则 NEXT: = UP。

[0193] 在上面的步骤 5 中使用 max{D,1second} 而不是简单的 D 的原因，是由于 RTT；1 的角色是用于充当 RTT 的上限。

[0194] 优选的是，函数 $\lambda(x)$ 和 $\mu(x)$ 是 x 的递增函数。优选的是，对于所有较小的 x，λ 和 μ 函数都 <1，这将确保所选择的播放速率是小于 R，其对于较小缓冲器水平，造成缓冲器增长。应当注意，所选定的播放速率至多等于 $\max(\lambda(B/\max\{D,1\}), \mu(B/\max\{D,1\}))\cdot R$，确保 $\lambda(B/\max\{D,1\})$ 和 $\mu(B/\max\{D,1\})$ 均小于一的所有缓冲器水平 B 的缓冲器增长。

[0195] 一种更简单的处理直接将新表示挑选为播放速率小于 $\lambda(B)\cdot R$ 的最佳表示。这将具有下面的属性：当缓冲器接近于空时，该缓冲器趋向于进行填充。但是，其还造成大量的表现切换，这是由于 R 可能波动的相当大。本申请所描述的更复杂的速率选择过程尝试

避免切换，而不是允许在向下切换到更低的播放速率之前，缓冲器某种程度地耗尽。对于这项工作，应当以某种方式来选择函数 μ 和 λ，使得对于中等到大型缓冲器水平，μ 超过 λ：应当注意，如果所选定的播放速率是 CURR,，所测量的速率是 R，那么只要满足式 7，就不会发生速率改变，其允许在不具有速率切换的情况下，接收速率某种程度地波动。

[0196] $$\frac{CURR}{\mu(B/\max\{D,1\})} \le R \le \frac{CURR}{\lambda(B/\max\{D,1\})} \quad (式\ 7)$$

[0197] 在一些版本中，λ 和 μ 仅是缓冲器水平 B 而不是比率 B/max{D,1} 的函数。用于引入后者的动机如下所述。

[0198] 使 α 表示所选定的表示的播放速率与下载速率之比。想要确定一个适当的 α。花费近似 α •D 的 r 时间来下载到下一个切换点。在将所接收的数据增加到缓冲器中之前，该缓冲器已消耗到 B-α •D。为了避免停滞，需要该数量为正数；作为一个安全垫，其应当与增加到缓冲器的片段的播放持续时间 D 成正比（一旦其被下载），所以对于一些 β>0 来说，该数量应当至少为 β •D。总之，需要 B - αD ≥ β •D。

[0199] 对于 α 的求解给出 B/D - β ≥ α。这建议该表示选择过程应当选择播放速率与下载速率之比不超过 B/D - β。函数和是关于可接受的这种比率的限制；因此，其应当是不超过 x-β 的 x = B/D 的函数。

[0200] 使用 B/max{D,1} 来替代 B/D，在实践中要考虑用于发送一个片段的 RTT 的另外成本。具体而言，1 可以用 RTT 的近似值的某个倍数来替代，或者用其它参数来替代，其中这些其它参数考虑用于发起从服务器下载媒体数据的过程的反应时间。

[0201] 图 18 示出了如可以用于选择播放速率的 λ 和 μ 函数的示例。x 轴是 D 的单位的缓冲器水平，y 轴是接收比例（即，播放表示速率除以当前接收或者下载速率）。如线 1802 所示，如果接收比例小于一，则缓冲器将增长，如果其大于 1，则缓冲器将收缩。标识了三个区域。首先，如果播放器在某个决策点低于 λ 曲线 1804，则其将在速率上向上切换。如果其位于 λ 曲线 1804 和 μ 曲线 1806 之间，则其停留在所选定的速率。如果其在 μ 曲线 1806 之上，则向下切换。

[0202] 图 19 示出了使用“保守”设置的（λ，μ）函数的示例性选择。这种设置是“保守的”，其含义在于：没有使用所有可用的带宽，但在交换停滞中很少见。

[0203] 图 20 示出了使用“中等”设置的（λ，μ）函数的示例性选择。这种设置是“中等的”，其含义在于：与保守设置相比，其使用更多的带宽，但比特更容易停滞。

[0204] 图 21 示出了使用“积极”设置的（λ，μ）函数的示例性选择。这种设置是“积极的”，其含义在于：其尝试积极地使用所有可用的带宽。与其它两种所给出的示例性设置相比，积极设置可能更频繁地停滞。

[0205] 图 22 示出了使用一种处理来仿真 MLB 过程（即，某种程度上类似于工作于美国职业棒球大联盟 (MLB) 的某些研究人员所提议的处理）的（λ，μ）函数的示例性选择。应当注意，（λ，μ）函数并不基于媒体缓冲器充满程度进行变化。

[0206] 图 23 示出了用于 λ 和 μ 设置的并排值的示例。

[0207] 图 24 示出了用于 λ 和 μ 设置的并排值的示例。

[0208] 图 36 包括可以用于速率选择中的 λ 和 μ 的值的表格。

[0209] 图 25 示出了用于速率估计、随后基于速率的速率选择、随后基于缓冲器管理的速

率选择的处理。在这些示例性处理中，使用本申请所描述的方法中的一个或多个来执行速率估计。基于该估计，选择新的播放速率，并可以基于缓冲器管理规则对播放速率进行调整。

[0210] 5、请求取消

[0211] 在一些情况下，甚至一个好的速率选择过程也不能独自地防止视频播放停滞。例如，如果在进行了请求之后但在完成之前，下载速率急剧地下降，则所选定的比特率可能是太大，在甚至到达用于改变播放速率的下一个切换机会之前，这种较慢的下载速率也可能导致播放停滞。

[0212] 再举一个例子，当可用的带宽急剧地增加时（例如，由于从蜂窝连接转换到 WiFi 连接），媒体缓冲器可能填满了相对较低的播放速率媒体。在该情况下，有利的是，丢弃已经被下载但还没有播放的很大部分的媒体，再次下载已丢弃的 p 时间的部分，但这次选择更高播放速率表示来下载。因此，已经下载的低播放速率媒体被取消掉，在其位置从另一个表示下载更高播放速率媒体来进行播放，因此导致更高质量的用户体验。

[0213] 由于此原因，一种流模块实现可以实现用于对下载速率进行监测，并在某些环境下取消早期的决定的模块。如果一个请求被取消，则流模块应当随后基于更新的、更恰当的下载速率的估计，来发出一个新请求。这里，我们将这种监测模块称为请求取消处理。

[0214] 请求取消处理可以由于不同的原因来对请求进行取消。例如，其可以由于下面原因而取消请求：下载速率急剧地下降，故由于不能足够快速地接收到数据，而使播放处于停滞的危险之中。进行取消的另一种原因是：如果确定可以选择更高质量的媒体，并且能及时地获取该媒体以进行播放。进行取消的另一种原因是：不管接收机怎么做，该接收机都确定将发生停滞，并且相对于允许完成未决的请求，估计进行取消将缩短停滞时期。随后，接收机在估计更短的停滞的情况下，选择要执行的动作，其还潜在地考虑要进行播放的媒体表示的质量。当然，是否存在停滞，以及当存在停滞时的其持续时间，可能会与该估计有所出入。

[0215] 一旦决定进行取消，则实际的取消可以通过关闭发出该请求的 TCP 连接来实现。这种关闭具有告诉服务器不要继续发送被取消的片段的数据的效果，因此所关闭的连接所使用的带宽变得可用于获取替代数据。

[0216] 随后，流模块可以发出一个请求以替代被取消的请求。为此目的，可能需要打开一个新的 TCP 连接。

[0217] 一种实现具有选择替代请求的几种选项。哪一种是最适当的一个，取决于要被播放的内容的类型。

[0218] 可以尝试挑选允许实现流的无缝播放的替代请求。在通常情况下，这意味着在前一下载片段的结束时间或者之前，该替代请求必须具有一个切换点。

[0219] 在该情况下，如果发生下面情形，播放器就应当取消：预测当继续进行下载而不取消时，将发生停滞，并且预测在进行取消，选择一个替代分段的情况下，可以避免停滞或者至少减少停滞的持续时间。随后，可以挑选具有针对该替代请求的属性的最高质量媒体请求。

[0220] 速率取消处理可以如下所述地预测停滞：可以通过将片段中的示接收字节的数量除以下载速率的估计值，来计算所发出的请求的估计的完成时间。如果该时间晚于为了实

现平滑播放而需要该片段的期限，则预测发生停滞。

[0221] 当预测即将停滞时，该请求取消处理判断速率的切换是否可能提高用户体验；当可能提高时，才做出取消的决定。

[0222] 一种实现可以只基于速率估计和候选替代片段的大小，对装载该替代片段所花费的时间进行估计。

[0223] 另一种实现也可以考虑由于取消而产生的另外开销：可以增加估计的 RTT 的某种倍数，以说明用于取消现有的请求和发出一个新请求所需要的时间。来自于所取消的请求的在网络上进行排队传输、但还没有到达目的地的数据，可能产生另外的延迟。客户端可以通过将 TCP 接收窗口大小除以所估计的大小来估计该延迟。延迟的另一种估计可以是基于估计的带宽延迟乘积。客户端可以考虑这两种估计的组合（例如，这两个估计中的最大值）。

[0224] 总之，客户端计算用于下载整个的替代片段所需要的时间的总和、通常与 RTT 成比例的数量、加上排队延迟的估计。如果预测发生停滞，并且该时间小于所估计的用于下载当前片段的剩余时间，则发出取消。

[0225] 此外，当播放器通知下载第一片段与所预期的相比要花费更长的时间（这是由于初始的速率选择是不准确的），则请求取消处理还可以在启动时进行取消。

[0226] 此外，另一种速率取消实现还可以挑选不允许无缝播放的替代请求，但意味着跳过多个帧。当播放需要端到端延迟保持的很小的实时内容时，这种要求是必要的。

[0227] 在帧跳过的情况下进行取消的实现，可以以某种方式来挑选替代片段，使得该帧跳过尽可能的小。

[0228] 作为替代请求，该实现可以选择：不超过指定的停滞持续时间或者跳过帧持续时间的能持续地下载的最高质量请求。

[0229] 可以针对已经下载的片段，实现不同类型的取消：如果播放器已经缓存了将进行播放的一些媒体，则可以决定通过网络来获取更高质量的表示，并对其进行流处理，同时丢弃先前缓存的低质量版本。

[0230] 如果确定可以及时地接收更佳质量视频，使得可以在没有停滞的情况下进行播放，则该取消处理可以决定执行这些替代操作。

[0231] 图 26 示出了在紧接着时间 T1 处的新片段请求之后，发生下载速率的强烈下降。在直到该请求为止，接收速率是 OR，随后其下降到 NR。缓冲器水平现在下降。在关于 T2 = T1+OR/NR* 片段持续时间的时间，完全地下载新请求的片段。如果 OR/NR 较大，则在时间 T1，在该缓冲器中可能存在超过 p 时间持续量的媒体内容，其意味着所请求的片段不能在没有停滞的情况下进行播放。应当注意，pker 估计器将更快速地收敛到速率 NR，但由于在 T1 之前做出了所述请求，因此在该估计值有机会收敛到新速率 NR 之前，就进行了该片段的下载。为了避免停滞，并且发出具有校正的估计的新请求，需要取消该请求，并重新发出具有更适当的比特率的请求。

[0232] 图 27 示出了具有请求取消的速率下降。在下载速率的急速下降（线 2702）之后，缓冲器开始消耗数据，所估计的下载速率（例如，pker 过程）开始收敛到新的下载速率。在某个时间点，流管理器通知不能及时地接收该片段，以致于不能在无停滞的情况下进行播放。在图 27 的图形中，将该时间点标记为“取消点”2704。在该时间点，对已经部分接收的

片段进行取消，将其从缓冲器中逐出（因此，在缓冲器水平中发生另外的下降）。但在此之后，可以请求具有正确速率的片段，因此缓冲器水平不会进一步下降。事实上，如果使用平凡的速率选择过程，则其可以再次增长。

[0233] 图 28 是示出一种示例性请求取消过程的流程图。

[0234] 图 29 示出了用于请求取消检测的过程。

[0235] 现在描述基于上面的细节的请求取消实现。

[0236] 在该节中，N_i 指示已请求但还没有接收到的表示组 i 中的片段的数量。将这些标记为$F_{i,1},\ldots,F_{i,N_i}$。此外，假定以递增的起始 p 时间顺序来排序 $F_{i,j}$，$\alpha(F_{i,j})$ 是针对所请求的片段已经下载的字节数量除以其字节的大小。变量表示当前播放 p 时间。请求取消检测处理可以如图 29 的伪代码所示地一样进行执行。

[0237] 当请求取消检测处理运行时，其可以返回零（在该情况下，不采取任何动作），或者标识要取消的组中的片段。在识别了该片段之后，其意味着该片段以及（以 p 时间顺序）在其后到达的同一个组中的所有数据都将被取消，并从缓冲器中强行输出。随后，SM 应当再次调用其速率决策过程，针对该部分发出新的替代请求。

[0238] 为了解释该过程，假定在该时间，仅单一请求是永远优秀的。在该情况下，使 R 是下载速率的准确估计，使 d_{avail} 是直到有问题的片段被播放为止，仍然可以接收的字节的数量。数量 d_{need} 是在该片段中仍然错失的字节的数量。因此，如果 $d_{avail}<d_{need}$，则预测播放器在播放片段 $F_{i,j}$ 之前将停滞。这解释了上面的处理过程中的第一个“如果”条件。

[0239] 即使预测发生停滞，也仅当取消能导致避免该停滞或者至少减少其持续时间时，进行取消才有意义。在取消之后，必须选择一个新的片段，并从头开始下载。如果仅存在一个表示组，速率决策过程选择了正确的速率，则这将花费近似等于 λ 乘以持续时间 $(F_{i,j})$ 的时间，其中 λ 是当前有关的 lambda 因子。另一方面，如果 SM 决定不进行切换，则完成当前片段下载将花费时间 $d_{need}\cdot R^{-1}$。为了简单起见，假定 λ ＝ 1，得到第二状况，以及可能的其它因子。

[0240] 6、请求加速器

[0241] 用于流媒体客户端的简单方法是使用单一 HTTP 连接来获取媒体。这种客户端对片段请求进行顺序地处理。这种方法在视频流媒体中具有一些缺点。首先，通常仅将通用网络软件调谐到用于在长下载上获得最大吞吐量。虽然这适合于接收大型文件，但其与其它重要的流目标（例如，稳定的接收速率）相冲突。第二，由于 TCP 的本质，这种链路的满负荷容量并不必要结合这种 HTTP 连接来使用。如果该信道经历某种延迟和包丢失，则 TCP 对可以实现的实际吞吐量进行限制，其潜在地防止流媒体客户端对良好质量媒体进行流处理。

[0242] 为了避免这些问题，可以实现特殊的 HTTP 客户端，本申请称为请求加速器 (RA)。请求加速器具有特殊的处理，以避免或者减少之前提及的问题。请求加速器的实现可以利用几个关键因素来实现其目标。其可以使用几个 TCP 连接来接收数据。这些连接可以并行地活动。可以将数据请求分割成更小的区块请求，可以在不同的连接上对其进行各自地下载，并在请求加速器中将其重新组合在一起。可以对 TCP 连接参数（例如，特别是 TCP 接收窗口大小）进行调谐，使得这些连接彼此之间是公平的，并具有相对稳定的数据接收。可以基于测量的网络状况和目标播放速率，动态地调整要使用的 TCP 连接的数量。

[0243] 要使用的 TCP 连接的理想数量取决于网络状况，特别是取决于往返时间 (RTT) 和分组丢失行为。因此，RA 可以使用一些方法对这些量进行估计。

[0244] RA 可以通过对于从发出 HTTP 请求到接收到响应为止所花费的时间进行采样，来估计 RTT。一种实现可以使用通过下面方式所获得的 RTT 的估计值 ：对在固定的时间段（如最后几秒钟）上获得的所有这些采样值中取最小值。另一种实现可以使用最后获得的 N 个采样值中的最小值作为该估计值，其中 N 是某种整数。

[0245] 通常困难的是，获得在 TCP 层上的分组丢失的测量值，这是由于 TCP 协议处理分组丢失，将数据的连续前缀传输给应用。因此，有用的是对针对分组丢失的合理值进行确定，以作为 RA 过程的输入。一种实现可以将该丢失率估计成常数。缺少任何分组丢失测量值，RA 可以将该丢失率估计成 1％，或者 RA 可以将该丢失率估计成 0.1％。可以通过连接的类型来确定该估计值，例如，针对 WiFi 连接，可以将该估计值设置为 0.1％，针对蜂窝连接，将该估计值设置为 1％。RA 可以使用诸如 RTT 的方差之类的其它方法，来间接地推断分组丢失。替代地，一种实现可以通过查询操作系统内核以获得其上的信息，来获得分组丢失估计值。

[0246] 另一种实现可以在应用自身中对该丢失率进行估计。为此，可以使用基于对于下面的观测的过程 ：来自网络套接字的数据通常在最大分段大小 (MSS) 块中接收，但分组丢失造成更区块的接收、大小近似为整个 TCP 接收窗口的突发。使 M 是字节的 MSS（很好的猜测是 M ＝ 1500）；那么如果接收到 n 个字节，则发送的分组的数量是大约 n/M。使 z 是读取的套接字的数量（其导致读取超过 k • M 个字节），其中是某个小整数。假定将 k 选择的足够的大，使得在该应用的两次网络读取之间不可能到达 k 或者更多的分组。对于在套接字上经常等待的应用来说，k ＝ 3 应当是很好的。转而，p ＝ z • M/n 是分组丢失概率的估计值。通过从预期的起始点来统计 z 和 n，该过程可以对任何预期的时间范围之上的分组丢失率进行估计。

[0247] 给定 RTT 和分组丢失概率的估计值，应用可以计算需要的良好数量的连接。具体而言，该过程可以选择足够大的连接数量，使得可以使用该数量的连接来实现目标下载速率。单一连接可实现的速率通常受到可实现吞吐量上的 TCP 方程的限制，也就是说，单一 TCP 连接大约可以实现 T ＝ MSS/(RTT • √ p) 的平均下载速率。因此，该过程可以将连接的数量选择为与目标下载速率除以 T 成正比。

[0248] 此外，由于实现原因，RA 还可以对于 TCP 连接的数量设置下限和上限。例如，RA 可以对于其打开的连接的最大数量限制为 8，将连接的最小数量限制为 2。

[0249] 带宽、丢失概率和 RTT 容易发生改变。请求加速器对这些数量进行监测，动态地改变连接的数量。

[0250] 请求加速器可以将 HTTP 请求分割成更小的子请求，将返回的针对每一个子请求的数据响应重新组合成与原始请求相对应的相干响应。将请求分割成一些子请求存在多种优点。首先，为了使用可用的 TCP 连接，需要能够在所有可用的 TCP 连接上发送请求。媒体流播放器不提供足够的请求来使用所有这些连接。请求分割减轻该问题，这是由于其导致更大数量的子请求，随后其可以在不同的连接上进行发送。第二，请求分割导致更短的请求，其减少不及时地数据传输的风险 ：如果一些 TCP 连接比其它 TCP 连接暂时慢，则其仍然可以用于短请求。其与快速连接相比将更慢速地传输响应，但用于完成整个请求的其它相

对延迟通常不是很大，这是由于这些请求很小。

[0251] 通常，如果有更多的连接在使用，则优选的是，每一请求产生更多的子请求。为了实现此目标，当存在 n 个连接时，请求加速器可以将每一个请求分割成 n 个子请求。

[0252] 另一种实现根据请求大小，来挑选每一请求的子请求数量。如果将子请求大小选择为具有预测进行下载要花费的固定数量的时间（如，2 秒）的大小，那么当存在更多的请求时，将请求分割成更多的子请求，以实现期望的效果。

[0253] 这种分割规则应当确保不存在不必要的较小请求。例如，RA 实现可以在其分割过程中，施加最小子请求大小，如果不满足该最小值，则分割成更少的子请求。

[0254] 当使用多个 TCP 连接时，其可能对带宽进行竞争。在一个大的时间尺度，每一个连接与其它连接接收相同的数量，但在较小的尺度上（例如，在几秒钟上），一些 TCP 连接可以与其它 TCP 连接相比明显地更慢。这造成了流处理的问题，这是由于其意味着一些子请求可能与其它子请求相比花费更长的时间，这可能导致播放停滞。

[0255] 为了避免该问题，RA 可以使用 TCP 流控制来“制服”这些连接。其可以对每一个 TCP 连接的最大接收窗口进行充分地限制，使得没有任何连接可以使用明显地多于其吞吐量的公平份额。TCP 连接上的传输中的数据量（已发送但没有被确认的），大约是下载速率除以 RTT。因此，如果将 TCP 接收窗口设置为大约下值或者稍微更大 :用于该连接的目标下载速率除以所估计的 RTT，那么将下载速率限制为大约该目标下载速率，或者与该目标下载速率相比更大。因此，对 TCP 接收窗口大小进行设置可以充当为管理者，确保给定的 TCP 连接不按照这种高速率进行下载，其强迫其它 TCP 连接按照更低的速率进行下载。有了这样的机制，这些连接就趋向于按照大约相同的速度来取数，这是由于慢速连接具有可用于加速到其公平份额的带宽，但同时这些连接可以实现一个合计下载速率，后者至少是总目标接收速率，或者与总目标接收速率相比稍微更高。

[0256] RA 可以通过调整接收缓冲器，来调整客户端中的接收窗口。始终在连续的请求之间，对这些设置进行重新调整。

[0257] 一种实现可以将每一个连接的 TCP 接收窗口设置得与下值相比更高 :所估计的 RTT 和目标下载速率除以连接数量的乘积。

[0258] 例如，可以根据目标设定的用于播放的媒体速率，来确定目标下载速率。另一种实现可以基于当前播放速率来设置该目标速率（例如，当前下载速率的两倍）。

[0259] 6.1 RA 的实施例

[0260] 现在描述含有上面所描述的单元的请求加速器的实施例。

[0261] 图 30 是使得多个 TCP 连接进行获取的行为的图。图 30-31 示出了在不同的状况之下的行为。在该示例中，到 web 服务器的连接是被限制为每秒 2 兆（“mbps”）的带宽，往返时间是 150ms，存在 0.1％的分组丢失。存在活动地获取片段的四个连接。图 30-31 的图形示出了这四个连接的瞬时速率，以及总速率，以及在客户端中获得的 RTT 估计值。

[0262] 在图 30 中，这些连接的接收缓冲器不受限制。在图 31 中，将其限制于大约带宽延迟乘积的两倍。

[0263] 在图 30 和图 31 的示例中，这两种方法均稳定地实现 2mbps 总吞吐量。在这些连接具有有限的接收窗口的情况下（图 31），这些连接之间的传输是更加均匀的 :大多数时间，其按照大约相同的速率进行接收。这种情况对于不具有受限制窗口的连接（图 30）来说并

不全部成立，其中在该情况下，在较长的时间拉伸上，一些连接与其它连接相比更慢。

[0264] 不均匀的连接速度对于流应用来说具有问题，这是由于其可能意味着：一些紧急数据仅非常慢速地到来（在慢速连接上），而带宽转移到更快速连接上，这些更快速连接可能获取不是紧急需要的数据。

[0265] 不受限制接收窗口和受限制接收窗口之间的另一种差别是客户端操作的 RTT。在具有这些限制的情况下，RTT 保持较低，接近传播延迟。在不具有接收窗口限制的情况下，由于传输的数据的量超过底层传播延迟时间乘以该连接的容量，因此排队的延迟可能变得非常显著，故其造成高 RTT。对于媒体流客户端来说，高 RTT 是不期望的，这是由于客户端对于多个事件的反应时间通常是 RTT 的倍数。例如，针对用户寻求事件（其造成新的媒体内容被下载）的客户端反应时间，或者下载速率的减少（其造成请求取消或者表示的切换），通常是当前 RTT 的多倍，因此当 RTT 很大时，客户端对于这些事件的通常响应将会下降。

[0266] 图 32 是一种请求加速器处理的流程图。

[0267] 图 33 示出了用于发现多个子请求，以有助于给定的片段请求的过程。

[0268] 图 34 示出了用于选择各个请求的过程，其中这些请求被选定为具有计算大小的源请求的不相交时间间隔。在该过程中，子请求大小是有意随机化的，使得这些连接空闲的时间在连接和连接之间不同。这避免了所有连接都在相同的时间空闲，这导致更佳的信道使用。还对请求大小进行排序，使得越大的请求越要更早发送，以帮助在受限制的空闲时间中保持差别。

[0269] 图 35 示出了时间偏移以及通过时间偏移所确定的用于修复分段的片段结构的示例。

[0270] 在操作中，请求加速器从 SC 接收 HTTP 请求（每一个请求是 URL 和字节范围）。

[0271] 请求加速器通过 HTTP 下载所请求的字节范围，对该数据进行处理，一旦已完全接收到该数据，就将其返回给 SC。RA 目标针对于实现足够大的下载速度，但同时确保每一个片段都在其期限时间之前被接收到。高下载速度使得可以选择高质量视频表示，而尊重期限确保该播放在没有停滞的情况下进行。

[0272] 为了实现高下载速度的目标，RA 对变化数量的开放 TCP 连接进行管理，所有这些连接用于通过 HTTP 来接收数据。RA 关心要使用多少连接的细节（如果需要的话，对其进行打开或者重新打开），以及如何将请求分派到连接上。

[0273] 在一些情况下，RA 决定将源请求分割成一些更小的所谓 RA 请求，后者随后被分派到不同的连接上，这些请求的响应数据在到达之后，RA 对其进行透明地重新组合。例如，对于包括某个文件的前 64 千字节的源请求来说，RA 可以生成两个 RA 请求：一个用于 32 千字节区块，另一个用于该文件的第二个 32 千字节区块。随后，RA 可以在不同的连接上，并行地请求这两个区块，一旦接收到这两个 32 千字节区块，就生成用于原始请求的相干的 64 千字节响应。

[0274] RA 可以发出 RA 请求，后者超过源请求的简单子范围。例如，除了针对简单视频数据的请求之外，还可以发出针对一个片段的 FEC 数据的请求。在该情况下，一旦接收到 FEC 信息，RA 就对其进行透明地解码，并且只向源呈现最终的解码后的片段。

[0275] RA 自身进行自动调整以适应网络状况。例如，如果 RTT 很大，则 RA 可以决定发出更大的块的请求，以便避免在请求之间具有大量的空闲时间。自动调整的另一个例子在于：

RA 尝试使各个连接的速度保持类似，以便确保其请求的时效性。为了能够做到这些事情，RA 优选地直接访问其连接的套接字。例如，在类似 Unix 的环境中，其能够使用 setsockopt() 函数来设置套接字选项。

[0276] RA 测量和保持网络状态的跟踪，这包括对下载速率和估计的往返时间 (RTT) 进行具体的测量。其收集该信息，第一是由于连接自动调整取决于其可用性，第二是由于需要将带宽信息传送给 SM，后者使用该信息来计算其速率估计。

[0277] RA(通过 SC) 向 SM 转发的另一条信息是关于未完成请求的进展信息，即，已接收到给定的请求的多少数据。SM 使用该信息用于其速率估计以及请求取消决策。

[0278] RA 对 SM 进行带宽估计所需要的信息保持跟踪。该信息是下载所花费的 r 时间的总量 T_r、以及下载的字节总数 Z。这些数字都是单调递增的，SM 频繁地对这些数字进行轮询。如果并且仅当至少一个连接活跃时，定时器 T_r 才运行。如果一个连接在发送 HTTP 请求或者等待响应数据进入，则将该连接视作为活跃的。Z 计数器对输入的字节进行计数，并对所有连接进行汇总。

[0279] 6.1.1 RA 下载速率历史

[0280] 请求加速器通过保存增长数组 (T_r, Z) 对（其以其历史顺序进行存储），来保持一些历史速率的跟踪。我们将其称为数组 mapTrZ。mapTrZ 的更新频繁地发生；至少在时间上按照固定的间隔进行更新（例如，每 100ms)，当接收到新数据时，也可以进行更新。

[0281] RA 可以利用 mapTrZ 来计算加窗口的带宽估计，如下所述。假定感兴趣的窗口具有宽度 t，使 mapTrZ[last] 表示 mapTrZ 中的最后项。那么寻找最大的索引 i，使得 mapTrZ[i].T_r ≤ mapTrZ[last].T_r - t。应当注意，可以使用二进制搜索来高效地寻找 i。随后，如式 8 中所示地给出速率平均值。

[0282] $$R := \frac{\text{mapTrZ[last]}.Z - \text{mapTrZ}[i].Z}{\text{mapTrZ[last]}.T_r - \text{mapTrZ}[i].T_r} \quad \text{（式 8）}$$

[0283] 式 8 假定后续 T_r 中的差值与 t 相比而言很小。这通过进行足够频繁地采样来确保，并且不挑选小窗口宽度 t。

[0284] 在实践中，任意的增长数组是令人讨厌的。可以对向后查找的最大持续时间设置上限，因此存在着一种方式将 mapTrZ 实现成固定大小的环形缓冲器。这可以如下所述地实现。只要 mapTrZ 被更新，并且 mapTrZ 数组已经包含至少两个对，如果 T_r-mapTrZ[last-1].T_r<100ms，则替换最后项，否则增加一个新项。

[0285] 6.1.2 往返时间（“RTT”）估计

[0286] RA 收集带宽估计值。一种先验的用于获得 RTT 采样值的简单方法，是对发送出 HTTP GET 请求的时间和接收到响应的时间的差值进行测量。

[0287] 但是，这些测量包括排队延迟：如果客户端具有其它开放活跃连接，那么向客户端发送数据的最后一跳可以对多个分组进行缓存，如果与其接收数据的速率相比，与客户端的链路具有更低的速率。在该情况下，可以使用与本质相比更长的延迟来传送这些分组。

[0288] 在我们的案例，期望知道由于本客户端自身的活动所引起的排队延迟的 RTT 折扣。为了获得该数量的估计值，可以如下所述地进行操作。

[0289] 在每一个活动时期，使用之前描述的定时方法来收集 RTT 采样值；每一个 GET 都导致一个采样值。随后，当前估计值是所有这些采样值中的最小值。只要 RA 变得不活动，就

对该采样值列表进行清空。(例如,当超过第3节的高水印时,客户端变得不活动,启动的下载已经完成)。在不活动时期,或者在接收到任何RTT采样值之前的活动时期,RTT估计值是最后知道的估计值。

[0290] RTT估计器还可以返回“不知道任何RTT估计值”值,后者可以在例如客户端启动时使用。

[0291] 6.1.3 调整TCP连接的数量

[0292] 对TCP流控制进行调整,允许RA将不同的连接中的带宽保持的大约相同。多个可配置的调整常量可以包括k_R(在RTT中测量的速率测量窗口;建议值:30)、k_N(比例因子;建议值:8192字节)、N_{min}(N_{target}底盖;建议值:1)、N_{max}(N_{target}上盖;建议值:8)。

[0293] 将所估计的bandwidth-delay-product(BDP)规定为BDP:=RTT·R,其中RTT是估计的RTT(如上所述),R是在最后k_R·RTT时间上的平均接收速率(使用窗口方法来估计的)。

[0294] 不断如果,将连接的目标数量规定为如式9中所示,其中k_N是可配置常量。

[0295] $$N_{target} := \max(N_{min}, \min(N_{max}, \lceil BDP/k_N \rceil)) \quad (式9)$$

[0296] 对N_{target}的值定期地进行重新计算。如果当前打开的连接的数量小于N_{target},那么立即打开新的连接以匹配N_{target}。另一方面,如果N_{target}小于当前打开的连接的数量,则不采取任何立即动作。相反,只要完成了RA请求,RA就检查是否打开了太多连接,如果是,则对变得空闲的连接进行关闭。

[0297] 6.1.4 调整连接上的TCP接收窗口

[0298] RA将每一个连接的TCP接收窗口大小设置为$\lceil c_w \cdot BDP/N_{target} \rceil$。这里,$c_w$是可配置的硬编码常量,例如,$c_w=3$。只要RA将在一个连接上发送下一个HTTP请求,RA就对该连接的TCP接收窗口大小进行设置。

[0299] 6.1.5 请求分割过程

[0300] 将传递给RA的每一个源请求,分割成潜在地超过一个RA请求,其每一个对应于所请求的范围的不同部分。一旦完成了所有与给定的源请求相对应的RA请求,RA就将所接收的数据重新组合成一个完整的片段,随后将其返回给SC。

[0301] 对于给定的HTTP请求,RA使用取决于几个可调整值的过程,确定RA请求的数量n。n的值取决于下面的可调整常量:T_{wn}(速率估计窗口宽度;建议值:4s)、D_{min}(最小获取持续时间;建议值:2s)和c_s(RTT中的最小获取持续时间;建议值:6)。

[0302] 随后,如图33的伪代码中所示地,给出了用于寻找针对给定的片段请求的子请求的数量n的过程。

[0303] 随后,使用例如图34中所示的过程(其具有计算的大小),将各个请求选择为源请求的不相交时间间隔。

[0304] 6.1.6 请求分派过程

[0305] 请求加速器维持RA请求集。只要一个连接变得准备好发送下一个请求,就从RA队列中出列一个请求(如果该队列非空的话),并在空闲连接上进行发送。如果该队列为空,则从SC中获得新的片段请求。随后,将该请求分割成一些RA请求,并在RA队列中进行排队。优选地,按照用于寻找针对给定的片段请求的子请求的数量的过程所返回的分片顺序,

来进行这种排队。

[0306] HTTP 连接可以由于各种原因而关闭，例如，由于发生了 web 服务器超时，或者超过了可以在单个连接上发送的请求的数量。RA 应当认真地和透明地处理该情形。只要一个连接被关闭，RA 就自动地重新打开该连接。如果一个请求在被关闭的连接上进行，则使其从该连接中出列，将针对于没有接收到的部分的新 RA 请求放置在 RA 队列中的前面。

[0307] 该过程确保关闭的连接对于性能具有最小影响。

[0308] 6.1.7 特定的实施例中的 RA 参数选择

[0309] TCP 连接受到其流控制的约束：所广告的接收窗口上限对于允许在任何时间点未被确认的数据量进行限制。因此，如果 W 表示接收窗口大小，bdp 表示该连接的带宽延迟乘积，则具有 bdp ≤ W（条件 1）。第 6.1.4 节中的方法描述了选择接收窗口大小，使得倘若 $c_w>1$，则满足该条件 (1)。这确保了各个连接基本上不会获得超过可用带宽的其公平份额部分。为了允许速率增加，并且为了避免速率螺旋下降，优选的是，选择稍微大于 1 的 c_w，例如，$c_w = 2$ 或者 $c_w = 4$。该值越大，则速率可以增长的更快，但这些连接彼此之间就越不公平。

[0310] TCP 拥塞控制过程施加了另一种限制。如果 p 表示分组丢失概率，M 表示 TCP 最大分段大小，则如式 10 所指示的，单个连接的速率 r 受到限制。

[0311] $$r \le \frac{M}{RTT \cdot \sqrt{p}} \quad (式\ 10)$$

[0312] 现在，依据 BDP 和连接的数量 N，对该式进行重写（使用 bdp = r • RTT 和 BDP = N • bdp），可以获得如式 11 所示的公式。

[0313] $$BDP\frac{\sqrt{p}}{M} \le N \quad (式\ 11)$$

[0314] 这建议应当将 k_N 选择为小于式 9 中的 $M/\sqrt{p}$ 的比特，以便确保式 11 中的不等式成立。用于 M 的典型值是 1 千字节，如果设置 p = 0.01，那么 $M/\sqrt{p} = 10$ 千字节。因此，在该示例中，如用于在式 9 中设置 N 的第 6.1.3 节中所建议的，设置 k_N = 8,192 个字节，确保满足式 11 的不等式。可以适当地对接收机进行配置或编程。

[0315] 现在转到上面的第 6.1.3 节的处理过程，以便计算针对给定的源请求的 RA 请求的数量 n。先验的，我们使这些分片尽可能地小，这是由于小分片呈现多种优点：如果一个连接相比其它连接而言较慢，则不太可能对于小请求造成问题，这是由于小请求也能甚至在慢速连接上快速地完成。因此，在小分片设置中，慢速连接本质上来说刚刚结束服务更少的请求。小分片的另一种优点在于：其使得 RA 能在缓冲器中的时间相对较短的部分上工作，所以其趋向于尽力加强最紧急工作区域。

[0316] 但是，使分片较小是有代价的：首先，每一个请求都在上行链路以及下行链路上引起一些开销。第二，在完成一个请求之后，一个连接都会停留大约 RTT 的空闲。因此，请求分割过程应当理想地尝试选择尽可能小的区块，既不造成太多的上行链路业务，也不会在本质上未充分利用每一个可用链路的容量。因此，优选的属性如下所述：

[0317] 1、目标设定为每一 D_{min} 的实际时间，每一连接至多存在一个请求。这造成在最坏情况下，上行链路业务受到与 N_{target} 成正比的值的限制。

[0318] 2、目标设定为每一个 c_s •RTT，每一连接至多存在一个请求。这造成该连接的活动时间至少是大约 $c_s/(c_s+1)$，即，对于中等 c_s 来说，接近于 1。

[0319] D_{min} 的良好选择取决于使用情况。按照期望的端到端延迟的顺序（但小于端到端延迟）来挑选该值，其通常是一个片段的典型持续时间。如果该端到端延迟较大，则可以使用更大的缓冲器，并且更大的分片的不良影响越小。另一方面，在较短的端到端延迟上，缓冲器较小，因此应当使这些分片较小，以避免慢速连接造成停滞。在该场景下，更高成本的更小请求，值得在缓冲器水平中获得稳定性。

[0320] 可以根据 MPD（媒体呈现描述）中的简档指示符，对使用的参数进行调整，这是由于 MPD 是去往该客户端的流媒体的属性的概述。不是下载每一个媒体分段，并将其向终端用户显示，而是客户端可以基于与 MPD 中的简档不同的使用情况，选择对分段进行“跳过”。

[0321] 可以如下所述地设计关于 c_s 的选择的下限。如果有 N 个连接被打开，并且 RA 是活动的，则平均来说存在大约 $N \cdot c_s/(c_s+1)$ 个活动连接。为了确保所有 N 个连接的接收窗口都处于汇总的足够大，以维持总目标速率，则期望 $c_w \cdot c_s/(c_s+1)$ 至少是 1。

[0322] 这种限制是保守的。估计的活动连接的数量 $N \cdot c_s/(c_s+1)$ 仅只是平均值，没有考虑方差，但可能的是存在某种方差。在实践中，有利的是，使 c_s 大约是上面的限制所建议的值的两到三倍，例如，当 $c_w = 3$，$c_s = 6$ 时，那么 $c_w \cdot c_s/(c_s+1)$ 至少是 2.5。

[0323] 6.2 具有前向纠错的 RA

[0324] 当通过几个 TCP 连接接收数据时，其有时临时地具有不同的下载速率。当将一个片段的请求分割成几个子请求时，那么仅当接收到最后一个子请求响应（区块）时，才接收到整个片段。当需要紧急地接收一个片段时，这可能变得有问题，这是由于可能在慢速连接上处理这些子请求中的一个，其阻止该片段被快速地接收。

[0325] 除了视频数据之外，内容提供商可以针对每一个片段，提供另外的前向纠错（“FEC”）修复数据，客户端可以获取该数据以帮助重建原始的片段。例如，假定客户端具有 4 个连接，并且需要紧急地接收大小 4000 个字节的片段。请求加速器可以将该片段分割成 4 个 1000 字节的范围，在这 4 个连接中的每一个连接上发送一个请求。它可能是：连接 1 是快速的，连接 4 是中等速度，但第二连接和第三连接是更慢速的。所以，即使合计下载速率在原则上足够的高，以致于能够及时地下载整个片段，但由于连接 2 和 3 被卡住，因此其也仅到达的非常晚。

[0326] 为了避免该问题，客户端尝试使用连接 1 来获取与连接 2 或连接 3 相同的数据，只要其有自己的子请求进行。这可以是有帮助的，但 RA 必须决定哪个连接需要更多的帮助；无论其是 2 还是 3。如果做出了错误的预测，则不必要下载重复的数据，该片段仍然不能及时地到达。

[0327] 更佳的请求加速器可以使用连接 1 来获取某种修复数据。该修复数据（其是 FEC 编码的）（如果其被成功地下载）可以用于对丢失的数据进行重建，而不管来自于请求 2 或者 3 的数据是否丢失。唯一的约束在于：接收的数据量足够用于对该片段进行重建。换言之，在该示例中，修复数据的数量加上接收的片段字节的数量可以大于或者等于 4000。

[0328] 在一种实现中，内容提供商提供对于编码的视频分段的 FEC 修复数据的访问。其可以以类似于原始视频数据的方式，使该修复数据可获得。例如，可以针对每一个媒体分段文件，提供包含修复信息的另外的 FEC 文件。内容提供商可以提供必要的信息和参数，以便

使用媒体呈现描述中的 FEC。在另一种实现中，媒体呈现描述不包含关于 FEC 的任何信息，但客户端可以使用通用约定来访问该信息，例如，关于如何从分段 URL 中导出 FEC 修复 URL 的名称的规则。

[0329] 客户端实现可以实现关于如何和何时请求修复数据的处理。请求的修复数据的数量可以取决于有多少数据是未接收的。另外，其还可以取决于该片段需要多久才可用。例如，如果有充足的剩余时间，则将希望及时地接收所有源数据，所以请求任何修复数据都可能是多余的。另一方面，如果该片段变得紧急，则可能想要请求大量的修复数据，这是由于倘若停滞是迫在眉睫的，则客户端不能及时地获得用于该片段的足够数据。因此，一种实现可以将请求的修复数据的量设置为 β (B)S，其中 S 是未接收的源数据的量，β (B) 是缓冲器水平的递减函数。

[0330] 另一种实现可以使未接收的数据的量与大多数未完成请求中的未接收数据的量成正比，而不是与总的未接收数据量成正比。

[0331] 6.2.1 修复分段生成器的实施例

[0332] 下面的所有计算涉及：如何使用定点 / 整数算法，来优选地执行使用 FEC(特别是使用 FEC 的 RaptorQ) 的 DASH 标准。这包括下面的计算应当使用定点算法来完成：计算一个表示的一个片段之中的源符号的数量和位置，计算用于该修复分段中的一个片段的修复符号的数量和位置。这是由于需要通过摄取过程与 RA 过程来实现完全相同的结果，其中摄取过程根据源分段来生成 FEC 修复片段，RA 过程使用接收的 FEC 修复片段和源片段的组合来解码源片段，因此这些计算必须具有完全相同的输出。由于在不同的平台上的不同浮点实现具有不同的角情况行为，因此使用浮点计算而不是定点算法在很难追查的场合中，可能产生细微的错误行为，这在两个端点必须产生完全相同的计算结果的标准中是不可接受的。

[0333] 下面所描述的所有其它计算并不涉及使用浮点来完成下面计算 (如果期望的话) (但定点运算应当是精密的)：计算用于一个修复分段中的一个片段的修复符号的数量和位置，这是由于在摄取过程和 RA 过程之间不存在用于计算完全相同的结果的依赖关系。

[0334] 可以基于已经处理的包括 sidx 表的源分段，在单独的进程中生成修复分段。针对这些进程的两个输入 (除了源分段自身之外)，是修复比例 R 和符号大小 S。为了有助于使用定点算法来计算一个分段中的一个修复片段的修复符号的数量和位置，可以用千分率来表达 R 的值，即 R = 500 意味着该比例是 1/2。

[0335] 在每一个分段之中，在源分段的起始处，存在着包括时间 / 字节偏移分段映射的分段索引信息。该时间 / 字节偏移分段映射是时间 / 字节偏移对列表 (T(0),B(0)),(T(1),B(1)),...,(T(i),B(i)),...,(T(n),B(n))，其中 T(i-1) 表示相对于媒体在所有媒体分段之中的最初起始时间，在该分段中针对该媒体的第 i 个片段的播放的起始时间，T(i) 表示第 i 个片段的结束时间 (因此其是下一个片段的起始时间)，字节偏移 B(i-1) 是位于该源分段中的数据的起始的相应字节索引，其中在该位置，媒体的第 i 个片段相对于该源分段的起始进行开始，B(i) 是至到并包括第 i 个片段的该分段中的字节的相应数量 (因此，B(i) 是片段 i+1 的第一字节的索引)。如果该分段包含多个媒体分量，则可以以绝对方式，针对该分段中的每一个分量来提供 T(i) 和 B(i)，或者可以相对于服务参考媒体分量的另一个媒体分量来表示其。无论如何，B(0) 是该分段中的第一片段的起始字节索引，由于 sidx 信

息在分段中的第一片段之前，因此 B(0) 可以大于零。如果 B(0) 不是零，则在与 sidx 相对应的修复分段的起始处存在一些修复符号。根据这种实现方式，这些第一修复符号可以对直到第一片段的起始的该分段中的数据进行保护，或者可以将其零填充到没有使用的数据字节中。

[0336] 修复比例 R 可以在 MPD 连同修复分段元数据中发送，或者通过其它方式 (TBD) 来获得。举一个用于 R 的例子，如果，那么修复分段大小是（非常接近地）近似为 :0.5 乘以生成该修复分段的源分段的相应大小，与该源分段中的源片段相对应的修复分段的修复片段的大小，也是（非常接近地）近似为 0.5 乘以该修复分段的大小。例如，如果一个源分段包含 1,000 千字节的数据，那么相应的修复分段包含近似 500 千字节的修复数据。

[0337] S 的值也可以在 MPD 连同修复分段元数据中发送，或者通过其它方式来获得。例如，S ＝ 64 指示源数据和修复数据包括每一个大小为 64 字节的符号，以用于 FEC 编码和解码目的。可以对 S 的值进行选择，以便与相关联的源分段的表示的流速率成正比。例如，如果流速率是 100Kbps，那么 S ＝ 12 字节是适当的，而当流速率是 1Mbps 时，则 S ＝ 120 字节是适当的，当流速率是 10Mbps 时，则 S ＝ 1,200 字节是适当的。一个目标可以是在下面二者之间具有良好的平衡 :如何将颗粒状片段划分成一些符号、以及与流速率相比的用于 FEC 解码的处理需求。例如，在 1Mbps 的流速率下，片段的大小大约 500ms，每一个片段大约 64KB 的数据，如果 S ＝ 120，那么该片段包括近似 500 个源符号，其意味着每一个符号大约是为了恢复源块而所需要的数据的 0.2%，这意味着由于符号粒度而需要的额外接收量通过下面的值进行上限限制 :0.2% 乘以接收该片段的 HTTP 连接的数量。例如，如果 HTTP 连接的数量是 6，那么符号粒度接收开销受到 1.2% 的限制。

[0338] 可以如下所述地，生成用于源分段的修复分段。将源分段的每一个片段视作为用于 FEC 编码目的的一个源块，因此将每一个片段视作为根据其来生成修复符号的源块的源符号序列。将针对第 i 个片段所生成的总修复符号的数量，计算成 TNRS(i) ＝ divceil(R*B(i),S*1000)，其中 divceil(I,J) 是输出最小整数的函数，其中该最小整数具有至少 I 除以 J 的值，即 divceil(I,J) ＝ (I+J-1)div J，其中 div 是将结果向下取整到最近的整数的定点除法。因此，针对片段 i 所生成的修复符号的数量是 NRS(i) ＝ TNRS(i)-TNRS(i-1)。

[0339] 修复分段包括用于这些片段的修复符号的串联，其中修复符号在修复分段中的顺序具有用于生成这些片段的顺序，在一个片段之中，修复符号具有其编码符号标识符（“ESI”）的顺序。

[0340] 应当注意，通过如上所述地，定义用于一个片段的修复符号的数量，针对所有先前的片段的修复符号的总数、以及因此用于修复片段 i 的符号的字节索引和字节范围，仅取决于 R、S、B(i-1) 和 B(i)，而不取决于该源分段中的片段的前一结构或后续结构里的任何一个。由于其允许客户端仅使用关于源分段（根据其来生成修复块）的相应片段的结构的本地信息，来快速地计算修复块在修复分段中的起始的位置，并且还快速地计算该修复块中的修复符号的数量，因此上述方法是有利的。因此，如果客户端决定开始下载，播放来自于源分段的中间部分的片段，则其还可以快速地生成和访问相应的修复块，后者与相应的修复分段之中的片段相对应。

[0341] 将与片段 i 相对应的源块中的源符号的数量，计算成 NSS(i) ＝

divceil(B(i)-B(i-1),S)。如果 B(i)-B(i-1) 不是 S 的倍数，则为了 FEC 编码和解码的目的，使用零字节来填充最后的源符号，即，使用零字节来填充最后的源符号，使得为了 FEC 编码和解码目的，其大小是 S 个字节，但并不将这些零填充字节填充成源分段的一部分。在该实施例中，用于源符号的 ESI 是 0、1、...、NSS(i)-1，用于修复符号的 ESI 是 NSS(i)、…、NSS(i)+NRS(i)-1。

[0342] 在该实施例中，可以例如通过向源分段的 URL 添加后缀“.repair”，来根据用于相应的源分段的 URL，来生成用于修复分段的 URL。

[0343] 此外，修复分段还可以是相应的源分段的一部分，例如，被添加到末尾部分。组合的分段的结构还可以是：源分段和修复分段在组合后的分段之中是连续的，即，组合后的分段包括第一源分段、接着第一修复分段、接着第二源分段、接着第二修复分段等。如本领域普通技术人员所应当认识到的，上面所描述的方法和处理可以容易地适用于这些组合的分段。

[0344] 6.2.2 使用修复分段的请求加速器的实施例

[0345] 用于修复分段的修复索引信息和 FEC 信息，通过用于相应的源分段的索引信息、并且根据 R 和 S 的值来隐式地规定，其中将 R 表示成指示千分率的 0 和 1000 之间的整数，S 用字节来表达。时间偏移和包括相应的修复分段的片段结构，通过这些时间偏移和相应的源分段的结构来确定。与片段 i 相对应的修复分段中的修复符号的开始和结束的字节偏移，可以分别被计算成 RB(i-1) = S*divceil(R*B(i-1),S*1000) 和 RB(i) = S*divceil(R*B(i),S*1000)。随后，与片段 i 相对应的修复分段中的字节的数量是 RB(i)-RB(i-1)，因此，将与片段 i 相对应的修复符号的数量计算成 NRS(i) = (RB(i)-RB(i-1))/S。（应当注意，这里不需要 divceil 操作，这是由于其保证分子是 S 的倍数，但这里可以使用 divceil，所获得的结果仍然是正确的）。可以将与片段 i 相对应的源符号的数量计算成 NSS(i) = divceil(B(i)-B(i-1),S)，其中使用零对最后的源符号进行填充，以用于解码目的（如果需要的话），其与用于编码的描述相同。因此，可以根据 R、S 和用于相应的源分段的相应片段的索引信息，来导出用于修复分段中的修复块的修复索引信息。

[0346] 举例而言，考虑图 35 中所示出的示例，其示出了在字节偏移 B(1) = 6,410 处开始，在字节偏移 B(2) = 6,770 处结束的片段 2，即，片段 2 的大小是 6,770-6,410 个字节，6,770 是片段 3 的起始字节索引。在该示例中，符号大小是 S = 64 字节，垂直虚线示出了在源分段中的与 S 的倍数相对应的字节偏移。在该示例中，将整体修复分段大小作为源分段大小的比例，设置为 R = 500 千分率（修复是源的近似 1/2)。将用于片段 2 的源块中的源符号的数量，计算成 NSS(2) = divceil(6,770-6,410,64) = (6,770-6,410+64-1) div 64 = 6，这些 6 个源符号分别具有 ESI 0、…、5，其中第一源符号是片段 2 的前 64 字节（其在该源分段中的字节索引 6,410 处开始），第二源符号是片段 2 的下一个 64 字节（其在该源分段中的字节索引 6,474 处开始）等。将与片段 2 相对应的修复块的结束字节偏移，计算成 RB(2) = 64*divceil(500*6,770,64*1,000) = 64*(3,385,000+64,000 - 1)div 64,000 = 64*53 = 3,392，将与片段 2 相对应的修复块的起始字节偏移，计算成 RB(1) = 64*divceil(500*6,410,64*1,000) = 64*(3,205,000+64,000 - 1)div64,000 = 64*51 = 3,264，因此在该示例中，在与具有 ESI 6 和 7 的片段 2 相对应的修复块中分别存在两个修

复符号，其在该修复分段中的字节偏移 3,264 处开始，在字节偏移 3,392 处结束。

[0347] 在图 35 中对此进行了示出。应当注意，在图 35 所示的示例中，即使 R = 500（修复是源的近似 1/2），存在与片段 2 相对应的 6 个源符号，修复符号的数量不是 3，可以预期如果一种简单的方法使用源符号的数量来计算修复符号的数量，但替代地编制为 2。与简单地使用一个片段的源符号的数量来确定修复符号的数量相比，这里执行的这种方法使得可以单独地根据与相应的源分段的相应源块相关联的索引信息，来计算修复分段中的修复块的位置。为此，为了在摄取过程和在 RA 过程中进行一致的计算，重要的是，使用定点算法来计算针对修复分段中的修复片段的修复符号的数量和位置。此外，随着源块中的源符号的数量 K 增长，相应修复块的修复符号的数量 KR 近似地接近于 K*R/1,000，这是由于在通常，KR 至多是 divceil(K*R,1,000)，KR 至少是 divfloor((K-1)*R,1000)，其中 divfloor(I,J) = I div J。

[0348] 7、示出的示例

[0349] 图 25 示出了一种速率选择过程。针对 λ 和 μ 的设置越高，则该设置就越积极。图 23 示出了用于参数 λ 的不同值。图 24 示出了用于参数 μ 的不同值。一种混合设置尝试通过两种主要机制来减少速率波动。第一机制是当 B 更大时，更谨慎地增加速率，第二机制是当 B 更小时，尽更大的努力停留在当前速率。

[0350] 用于 pker x.y:C = x*min(y*Tdl,B) 的示例设置可以是 :x、y 被设置为 8.1、4.2，2.4、4.4 或者其它 x、y 值。应当注意，pker 的实际平均窗口与 C 相比更长，这是由于跳过了下载暂停时期。不跳过 EWMA，并且假定下载暂停时期中的速率与最后的下载时间间隔的速率相同。

[0351] 对于 MWA（移动窗口平均），$H(z) = (1/D)*((1-z^{-D})/(1-z^{-1}))$，其中 D 是窗口大小。$X_i = \min\{R_k:k \geq i\}$，其中 R_k 是具有权重 W_k 的速率的 EWMA，$W_1<W_2<W_3<\cdots$。对于 EWMA，$H(z) = ((1-\beta)/(1-\beta z^{-1}))$，其中 β 是前一次平均的权重。在一些情况下，MWA 和 EWMA 大约相同。

[0352] 如果自适应估计器具有更长的平均窗口，其减少切换频率，同时为实时流媒体维持大约相同的平均速率。不同的设置针对于不同的场景起很好的作用。积极设置能很好地用于更加静止的场景，而不太积极设置更适合于动荡性场景。如果针对时间的显著部分，该带宽与最高表示速率相比更高一定的余量（例如，当与速率上限相比更高 20 秒平均值，该时间的％值），其有利于用于更积极的设置。理想情况下，设备应当能够检测到场景类型，并应用适当的设置。场景检测可以是基于诸如无线技术类型、在某个单位时间内速率改变的次数、移动速度等之类的因素。更简单的策略可以是基于上面的观测量 :当“整体”带宽高于速率上限时，使用更积极的设置。

[0353] 8、设置速率选择参数

[0354] 在该节中，提供了设置速率选择参数的示例。

[0355] 对于 MLB，EFF = 1 - Rv/Rdl，其中 Rv 是所选定的表示的当前速率，Rdl 是当前下载速率。建议的规则如下所述 :

[0356] 如果 EFF<0，则下降或许超过一个速率。

[0357] 如果 0< = EFF<0.1，则下降一个速率。

[0358] 如果 0.1< = EFF<0.6，则停留在当前速率。

[0359] 如果 0.6< = EFF<0.8,则上升一个速率。

[0360] 如果 0.8< = EFF< = 1,则上升或许超过一个速率。

[0361] 使 alpha = Rv/Rdl。则这粗略地转换成 :

[0362] 如果 alpha< = 0.4,则上升至少一个速率。

[0363] 如果 0.4<alpha< = 0.9,则停留在相同的速率。

[0364] 如果 0.9<alpha,则下降至少一个速率。

[0365] 将这种方式用于 DASH 客户端速率选择过程的上下文 :

[0366] 使 RUP 是与 UP 相对应的表示的速率,使 RDOWN 是与 DOWN 相对应的表示的速率,如上所述,使 Rv 是当前选择的表示的速率。将 RUP 选择地尽可能地大,使得 RUP< = lambda(t)*Rdl,将 RDOWN 选择地尽可能地大,使得 RDOWN< = mu(t)*Rdl。参数 t = B/(D+delta),其中 B 是媒体缓冲器中的当前呈现时间量,D 是关于超过进行当前决策时的时间点,直到下一个可能的切换点为止的时间的限制,delta 是考虑网络延迟和往返时间的较小参数,例如,可以近似地将 delta 设置为 1 秒或者 2 秒,或者可以根据测量的关于当前 RTT 的上限来设置 delta。

[0367] 下一个速率 RNEXT 的整体选择,如下所述 :

[0368] 如果 RUP<Rv,则 RNEXT = min{Rv,RDOWN},否则 RNEXT = RUP。

[0369] 可以通过针对所有 t,设置 lambda(t) = 0.4*R 和 mu(t) = 0.9,对上面的 MLB 参数进行近似处理,其中 R 是下一个更高表示的速率与当前表示的速率之比。例如,如果当前速率是 500Kbps,下一个更高速率是 750Kbps,那么 R = 1.5,因此 lambda(t) = 0.6。该方法如下所述地对 MLB 过程进行近似处理。

[0370] 在判断点,如果 EFF> = 0.6(即,alpha< = 0.4),那么 Rv< = 0.4*Rdl,在该情况下,RUP 至少是 Rv*R(由于对于所有 t 而言,lambda(t) = 0.4*R),因此 RNEXT = RUP,即速率可以上升到速率为 Rv*R 的下一个更高表示,如果 Rdl 甚至与 0.4*Rv 相比更高,那么 RUP 将大于 Rv*R(根据表示速率的粒度),如果例如 EFF 大于 0.8,则 RUP 将高于 Rv*R 超过一个速率。如果 EFF<0.1,那么 Rv>0.9*Rdl,在该情况下,RDOWN 将小于 Rv(这是由于 RDOWN< = 0.9*Rdl),随后该速率将下降,即 RNEXT<Rv。如果 EFF 位于 0.1 和 0.6 之间,那么 RUP< = Rv*R,RDOWN> = Rv,在该情况下,将 RNEXT 选择为等于 Rv。

[0371] 9、速率选择参数集

[0372] 下面的表格指出了一些可能的速率选择参数集。在下面的表格中没有示出的用于 t 的中间值的 lambda 和 mu 的值,应当通过在周围值之间进行线性插值来计算。针对超出下面的表格所示出的值之外的 t 值的 lambda 和 mu 的值,应当被设置为针对所示出的最大 t 值的 lambda 和 mu 值。

[0373] 如果针对所有 t,都满足约束条件 mu(t)< = t 和 lambda(t)< = t,则在理论上,在播放过程中应当不存在停滞,但从实际观点来看,优选的是在播放中具有很小的停滞,而不是不具有停滞,但继续按照大大减少的速率进行播放,例如,与在一秒钟暂停期间从 1Mbp 跳到 250Kbps 相比,从 1Mbps 跳到 20Kbps 可能是更坏的体验。在图 36 的表格中设置了 lambda 和 mu 的最小值,应当注意,对于值 mu(t)>t 和 / 或 lambda(t)>t 来说,很可能发生停滞(虽然独立于 lambda(t) 和 mu(t) 的设置,在该缓冲器为空时,在任何情况下都可能发生停滞)。

[0374] 如已经解释的，客户端设备可以为 HTTP 承载的自适应视频流，提供速率调整和下载处理。在因特网（和其它网络）上对视频进行流处理的客户端，面对带宽波动的问题。如果对高质量视频进行流处理，则链路可能有时不够足够地快，其造成播放器中断和重新缓冲。在其它情况下，低质量视频使用更少的带宽，但却具有更差的用户体验。一种解决方案是自适应地调整视频质量：当吞吐量较高时，选择更佳的质量，自动地向下切换。

[0375] 但是，自适应视频流引起了多种挑战：(1) 用于选择视频速率（质量）的处理或者算法，应当足够快速地动作，以便适应于速率下降以及速率增加。同时，应当避免过早或不稳定的决定，并且避免不必要的速率切换决定。客户端应当将目标设定于按照足够高的速率来获取数据，所以可以实现高视频质量。同时，下载处理应当确保数据被及时地接收。在对每一个帧进行播放之前，应当已完全地接收到该帧。应当在无需不必要的较大播放缓冲器的情况下，就能够实现这些目标。较大缓冲器的一些问题在于：对于直播事件，该缓冲器中的视频的量受到目标端到端延迟的限制，在这些情况下，其严重地限制可能的播放缓冲器。此外，对于较大缓冲器的依赖，可能在播放起始或者搜寻时造成非期望的延迟，这是由于需要对该缓冲器进行预填充。此外，较大的播放缓冲器使用大量的存储器，这些资源在移动电话和其它客户端设备中是稀缺的。

[0376] 为了解决这些问题，一种用于速率估计的处理将快速地对于接收速率改变进行反应。速率估计可以是专门为在媒体流中使用而设计订制的自适应窗口平均。速率估计器以一种方式考虑视频缓冲器水平和视频缓冲器水平中的改变，以便保证速率在需要时足够快地进行调节，并同时保持窗口宽度大小（并从而保持测量偏差）大小。该保证由过程通过以下方式来提供：(a) 如果 B 是缓冲器中视频数据的量（以播放时间的秒数为单位），当速率发生下降时，则估计器将调节缓冲器消耗到 B/2 所消耗的时间内的速率估计，以及 (b) 如果 B 是缓冲器中视频数据的量，当速率发生提高时，速率估计器足够快地调节到新的速率，使得可以在至多 3*B 的时间内看到该调节（提供智能速率改变过程）。

[0377] 速率决策过程可以进行速率决策，以便：(a) 当缓冲器处于低水平时，对其进行填充；(b) 使用该缓冲器来避免不规律的变化率，即使观察到较小的下载速率估计；(c) 在稳定速率场景中，快速地选择正确的稳定状态。多媒体下载策略用于 HTTP：(a) 允许准确的速率估计；(b) 即使网络延迟和分组丢失率较高，也能够实现链路容量；(c) 实现该流的及时传输。为了实现该目标，可以使用多个 HTTP 连接，根据网络状况将媒体请求分解成更小的块请求，使用 TCP 流控制机制对这些连接进行同步，对突发的数据进行请求。此外，还可以使用 HTTP 管道化处理来使连接保持繁忙。

[0378] 现在已解释了多种特征、方面和细节。如上所述，在各个实施例中，方法步骤可以通过相应的编程单元、提供给处理器、硬件或其它装置的指令来执行，如对于本领域普通技术人员来说显而易见的。同样，一些组成部分可以通过处理或程序单元来启用。一个实施例的组成部分的结构可以简单地包括由处理器执行的一组指令，但本申请将其描述成相应的方法步骤。

[0379] 在各个实施例中，可以使用下载速率加速，也可以不使用下载速率加速。下载速率加速的一个例子是：通过在 TCP 连接上使用 HTTP 请求来加速下载的方法或装置。TCP 连接具有特定的窗口大小，位于 TCP 连接的终端的节点可以改变用于窗口大小的设置。一种新颖性是设置用于连续 HTTP 请求的窗口大小，其中该大小取决于目标下载速率。因此，随着

目标下载速率改变，TCP 窗口大小也发生改变。

[0380] 在一个实施例中，一种方法和 / 或装置或者计算机可读介质，用于对通过一个网络路径耦接的源和接收机之间的所述网络路径上的数据下载进行控制，该方法包括：针对所述源和接收机之间的多个 TCP 连接中的每一个 TCP 连接，确定用于该 TCP 连接的 TCP 接收机窗口大小，其中所述源和接收机之间的 TCP 连接可以是直接连接，也可以是间接连接；确定用于媒体内容的目标下载速率，其中对于至少两个连续的 HTTP 请求来说，所述目标下载速率在至少两个值之间变化；使用所述多个 TCP 连接中的每一个 TCP 连接来下载要进行下载的媒体内容的多个媒体数据元素，其中所述媒体内容是针对多个 HTTP 请求的响应的一部分或者全部，其中，针对给定的 TCP 连接所确定的 TCP 接收机窗口大小，是至少部分地基于用于目标下载速率来确定的，其中对于所述至少两个连续的 HTTP 请求来说，所确定的 TCP 接收机窗口大小在至少两个值之间变化。

[0381] 针对当前的 TCP 连接所确定的 TCP 接收机窗口大小，可以是至少部分地基于针对当前 TCP 连接的当前估计往返时间（“ERTT”）与一个乘数速率相乘的乘积来确定的，其中该乘数速率位于下面二者限定的范围之内：针对当前 TCP 连接的目标下载速率、以及与目标下载速率相比更高预定的量的速率。当前 ERTT 可以通过在紧接前一测量周期（例如，一秒、十秒、五十秒等）上观测到的最小 RTT 的测量值来确定。至少部分地根据在静止周期结束时的测量值来确定当前 ERTT，该静止周期跟着下载周期，其是在这些 TCP 连接上，没有呈现活动的 HTTP 请求达到预定的持续时间段的周期。所述目标下载速率可以与下面的值成正比：所有在用的 TCP 连接上的当前合计下载速率除以在用的 TCP 连接的数量，例如，当前合计下载速率的两倍或者三倍。所述目标下载速率可以与媒体内容的播放速率成正比，其中该播放速率是横跨所有在用的 TCP 连接的合计速率除以在用的 TCP 连接的数量。可以将每一个媒体数据元素划分成大小位于预定的方差范围之内的多个区块，其中这些区块的数量是基于在用的 TCP 连接的数量。这些区块的数量还可以是进一步基于下面各项中的至少一项：针对当前 TCP 连接的当前估计往返时间（“ERTT”）、当前下载速率、和 / 或请求的媒体片段的大小。所述预定的方差范围可以是零，因此每一个区块在每一个片段请求时具有相同的大小，其中这些区块的数量等于在用的 TCP 连接的数量乘以预定的因子。每一个区块的大小可以大于或者等于最小字节数。将针对后续媒体数据元素的后续 HTTP 请求，分配给第一可用的 TCP 连接。

[0382] 此外，所述控制还可以包括：确定在所述源和接收机之间使用的 TCP 连接的数量，其中该数量大于 1，至少部分地基于所确定的至少一个网络状况来确定所述使用的 TCP 连接的数量；使用所述多个 TCP 连接中的每一个 TCP 连接来下载要进行下载的媒体内容的多个媒体数据元素，其中该媒体内容是针对多个 HTTP 请求的响应的一部分或者全部。可以至少部分地基于针对 TCP 连接的估计往返时间（“ERTT”）、所述目标下载速率和丢失率的估计，来确定在用的 TCP 连接的数量。可以将所述丢失率估计为 1％或者 0.1％。所述使用的 TCP 连接的数量可以位于两个和十六个之间（包括边界），和 / 或与下面各项的乘积成正比：(a) 所述目标下载速率、(b)ERTT、以及 (c) 估计的丢失率的平方根。针对这些 TCP 连接中的每一个 TCP 连接，至少部分地基于所述目标下载速率来确定用于该 TCP 连接的 TCP 接收机窗口大小，其中对于所述至少两个连续的 HTTP 请求来说，所确定的 TCP 接收机窗口大小在至少两个值之间变化。

[0383] 在一个实施例中，一种方法和 / 或装置或者计算机可读介质，用于对观看呈现缓冲器的下载速率进行估计，并基于该缓冲器有多大 / 多满 / 多空（即，其填充水平怎样）对下载速率进行估计。例如，在通过具有有限带宽的网络路径耦接到数据源的接收机处对下载速率进行估计（其中，该下载速率是接收机可以通过该网络路径接收数据的速率），可以包括：用于对接收机的呈现缓冲器进行监测，其中该呈现缓冲器至少在接收到媒体数据的时间和与该接收机相关联的呈现单元消费该媒体数据的时间之间，对该媒体数据进行存储；确定对下载速率进行估计所基于的非零估计时期；存储在该估计时期上的缓冲器水平的指示，其中给定的时间的缓冲器水平与已接收到但呈现单元还没有消费的媒体数据在该时间至少近似地占用多少呈现缓冲器相对应；使用所存储的指示作为估计的下载速率的测量值的一部分。

[0384] 所述呈现单元可以包括显示器和音频输出。所述估计时期可以具有与测量的缓冲器水平成正比的持续时间，其具有预定的比例因子。可以使该估计时期的持续时间，与在测量时间时呈现缓冲器中的未消费的媒体数据的字节数成正比；和 / 或取决于向呈现缓冲器增加媒体的另外速率；和 / 或与用于下载该呈现缓冲器的预定部分的时间成正比。所述预定的时间持续量可以与对呈现缓冲器的内容的预定比例进行下载的时间持续量相对应。所述估计时期可以与对呈现缓冲器的内容的预定比例进行下载所花费的时间，以及在呈现缓冲器中存在的该媒体数据的呈现时间相比更短。

[0385] 在一个实施例中，一种方法和 / 或装置或者计算机可读介质用于播放速率选择，其中该播放速率是从呈现缓冲器中消费媒体的速率、以存储器单元 / 时间（例如，兆 / 秒）测量的速率。当接收机针对一些媒体进行测量时，存在用于该媒体的播放速率。通常（但或许并不是始终），高质量媒体具有更高的播放速率，因此呈现一种平衡。使用 / 请求哪种播放速率是至少部分地取决于在呈现缓冲器中有多少媒体。接收机可以接收媒体，以便使用该接收机的呈现单元进行播放，其中该播放导致按照播放速率从呈现缓冲器中消费媒体，其中接收机被配置为从多个播放速率中进行选择，其包括：对呈现缓冲器进行监测，其中呈现缓冲器至少在接收到媒体数据的时间和与该接收机相关联的呈现单元消费该媒体数据的时间之间，对该媒体数据进行存储；对缓冲器水平的指示进行存储，其中该缓冲器水平与已接收到但呈现单元还没有消费的媒体数据占用多少呈现缓冲器相对应；使用所存储的指示来确定所估计的下载速率，使用所估计的下载速率来计算目标播放速率；根据目标播放速率，从所述多个播放速率之中进行选择。

[0386] 所选定的播放速率可以小于或者等于所估计的下载速率的预定的乘数，所述预定的乘数是缓冲器水平的递增函数。所述预定的乘数可以是呈现缓冲器中的媒体数据的播放时间持续量的仿射线性函数，当呈现缓冲器的缓冲器水平小于阈值量时，所述预定的乘数可以小于一。当呈现缓冲器中的媒体数据的呈现时间持续量大于或等于预置的最大数量的呈现时间时，所述预定的乘数可以大于或等于一。所述预定的乘数可以是呈现缓冲器中的媒体数据的呈现时间持续量的分段线性函数。所选定的播放速率可以小于或者等于所估计的下载速率的预定的乘数，所述预定的乘数可以是呈现缓冲器中的媒体数据的字节数的递增函数。可以将播放速率选择为所述多个播放速率之中的最大可用播放速率，其小于或等于比例因子乘以下载速率估计值，其中该比例因子是呈现缓冲器中的媒体数据的播放时间持续量除以针对速率改变的反应时间的估计值的递增函数。该反应时间可以是关于该媒体

数据中的切换点之间的呈现时间的上限，和 / 或该反应时间的估计值可以是关于该媒体数据中的切换点之间的呈现时间的平均值。该反应时间的估计值可以大于或者等于预定的常量乘以估计的往返时间（“ERTI”）。

[0387] 接收机接收媒体以便使用该接收机的呈现单元进行播放，其中该播放导致按照播放速率从呈现缓冲器中消费媒体，其中接收机被配置为从多个播放速率中进行选择，其可以包括一种用于对呈现缓冲器进行监测的方法或装置，其中呈现缓冲器至少在接收到媒体数据的时间和与该接收机相关联的呈现单元消费该媒体数据的时间之间，对该媒体数据进行存储；对缓冲器水平的指示进行存储，其中该缓冲器水平与已接收到但呈现单元还没有消费的媒体数据占用多少呈现缓冲器相对应；使用所存储的缓冲器水平的指示来确定该缓冲器水平的允许的方差，使用该缓冲器水平的允许的方差来计算目标播放速率；根据目标播放速率，从所述多个播放速率之中进行选择。

[0388] 可以基于上限比例因子、下限比例因子、下载速率估计值、当前播放速率、缓冲器水平、以及针对速率改变的反应时间的估计，来选择所述播放速率。所述上限比例因子和所述下限比例因子，可以是呈现缓冲器中的媒体数据的播放时间持续量除以针对速率改变的反应时间的估计值的递增函数和 / 或分段线性函数，其中所述上限比例因子大于或者等于下限比例因子。当前一播放速率位于下限比例因子乘以所估计的下载速率和上限比例因子乘以该下载速率估计值之间时，可以将播放速率选择为与前一播放速率相同。当前一播放速率高于上限比例因子乘以所估计的下载速率时，可以将播放速率选择为最大可用的播放速率，后者不大于上限比例因子乘以该下载速率估计值。当前一播放速率低于下限比例因子乘以所估计的下载速率时，可以将播放速率选择为最大可用的播放速率，后者不大于下限比例因子乘以该下载速率估计值。

[0389] 在一个实施例中，一种方法和 / 或装置或者计算机可读介质用于进行请求，但还用于判断在处理请求中是否取消。随着接收机对媒体的分段 / 部分 / 片段进行请求，接收到针对该请求的响应，存储来自该响应的媒体，并可以进行另一个请求，接收机可以确定取消一个请求并发出不同的请求是否是优选的。处于最积极并选择最高播放速率的接收机可以确定媒体的播放速率，其中该接收机预期随着其对媒体进行消费，可以在没有用尽呈现缓冲器中的媒体的情况下获得媒体数据。当下载速率意外下降时，接收机判断是否取消其当前请求，是针对更低的播放速率媒体进行新的请求，还是使当前请求继续进行播放。取消高播放速率请求并使用更低的播放速率请求进行替代，可以导致呈现缓冲器中的内容持续更长时间，但取消请求中间流可能造成针对该请求所接收的任何部分媒体的丢失。

[0390] 在一个这种实施例中，接收机接收媒体以便使用该接收机的呈现单元进行播放，其中该播放导致按照播放速率从呈现缓冲器中消费媒体，其中接收机被配置为从多个播放速率中进行选择。确定请求动作包括：对呈现缓冲器进行监测，其中呈现缓冲器至少在接收到媒体数据的时间和与该接收机相关联的呈现单元消费该媒体数据的时间之间，对该媒体数据进行存储；对缓冲器水平的指示进行存储，其中该缓冲器水平与已接收到但呈现单元还没有消费的媒体数据占用多少呈现缓冲器相对应；维持发出的用于下载所选定的第一区块的媒体数据的请求的状态，当发送的请求是未完成的时，基于网络状况和所发出的请求的状态，来判断是继续该请求还是取消该请求。

[0391] 判断是继续该请求还是取消该请求，可以包括：判断在第一媒体数据被播放完之

前，是否存在足够的时间来完成针对该请求的下载，如果不存在足够的时间，则取消该请求。判断是继续该请求还是取消该请求，还可以包括：判断在所选定的第一区块或者更高速率的第二区块被播放之前，是否存在足够的时间来下载所选定的第二区块，如果不存在足够的时间，则取消该请求，并发出针对第二区块的请求。判断是继续该请求还是取消该请求，还可以包括：基于下载速率和媒体消费速率，检测是否发生停滞；估计下面二者之间的停滞时期：呈现单元不能够按照正在被消费的媒体所指示的速率来消费媒体数据的时间、与呈现单元能够重新继续按照正在被消费的媒体所指示的速率来消费媒体数据的时间；确定继续请求或者取消请求对于该停滞时期的影响，如果取消该请求将缩短停滞时期，则取消该请求。

[0392] 其它特征可以包括：选择第二区块的媒体数据，其中该第二区块的媒体数据具有一个起始呈现时间，该起始呈现时间与第一区块的媒体数据具有相同的起始呈现时间；请求第二区块的媒体数据的下载；选择第二区块的媒体数据，其中该第二区块的媒体数据具有一个起始呈现时间，该起始呈现时间晚于第一区块的媒体数据的起始呈现时间；请求第二区块的媒体数据的下载。接收机可以对第二区块的媒体数据进行选择，使得其起始呈现时间与第一区块的起始呈现时间相比，具有可用于该接收机的最低差值，和/或使得其播放速率是在其呈现时间和第一区块的媒体数据的起始呈现时间之间，具有预定的最大间隙的最大播放速率。

[0393] 一些实施例可以包括：判断是否不能及时地完成第一区块的媒体数据的剩余部分的下载以进行播放；判断是否可以及时地完成第二区块的媒体数据的下载以进行播放；使判断是继续所述请求还是取消针对第一区块的媒体数据的请求，并替代地请求第二区块的媒体数据，基于判断是否不能及时地完成第一区块的媒体数据的剩余部分的下载以进行播放、以及判断是否可以及时地完成第二区块的媒体数据的下载以进行播放。可以将第二区块的数据中的媒体数据的播放速率，选择为该接收机所支持的最高播放速率。接收机可以对覆盖呈现缓冲器中已经存在的至少一些媒体数据的呈现时间的媒体数据进行请求，下载所请求的媒体数据，播放所请求的媒体数据，丢弃呈现缓冲器中已经存在的相应媒体数据里的至少一些。所请求的媒体数据的播放速率可以是最大播放速率，服从关于从呈现缓冲器中丢弃的相应媒体数据的最大呈现时间持续量的约束。可以对所请求的媒体数据进行选择，使得其起始呈现时间是可用于该接收机的最早起始呈现时间。

[0394] 在一些接收机中，下载依赖于缓冲器水平，接收机使用高水印和低水印的概念。在该接收机中，从源下载媒体数据，并将其存储在该接收机的呈现缓冲器中。确定呈现缓冲器的填充水平（或者简单的“水平”），其中该填充水平代表呈现缓冲器中有多少部分包含还没有被呈现单元消费的媒体数据。如果填充水平高于高填充阈值（“高水印”），则下载动作停止，如果填充水平低于低填充阈值（“低水印”），则下载动作重新开始。随着呈现单元对媒体数据进行消费，可以对填充水平进行更新。可以用存储器存储容量的单位和/或呈现时间的单位，对填充水平进行测量。下载动作可以是基于估计的往返时间（“ERTT”），其中当重新开始媒体数据下载时，对 ERTT 进行重置。如果在多个 TCP 连接上发生下载，则当重新开始媒体数据下载时，可以对使用的多个 TCP 连接进行重置。高填充阈值和低填充阈值可以随时间发生改变。

[0395] 本领域普通技术人员在阅读本申请后，还可以想到另外的实施例。在其它实施例

中，有利地，可以实现上面所公开的发明的组合或者子组合。为了说明目的，示出了组成部分的示例性排列，应当理解的是，在本发明的替代实施例中，可以预期对这些组成部分的组合、补充、重新排列等。因此，虽然已经针对示例性实施例描述了本发明，但本领域普通技术人员应当认识到，众多修改是有可能的。

[0396] 例如，本申请所描述的过程可以使用硬件组件、软件组件和／或其任意组合来实现。因此，应当将说明书和附图视作为示例性的，而不是限制意义的。但是，显然，可以在不脱离如权利要求书所阐述的本发明的更广精神和保护范围的基础上，对本发明做出各种修改和改变，本发明旨在覆盖落入所附权利要求书的保护范围之内的所有修改和等同物。

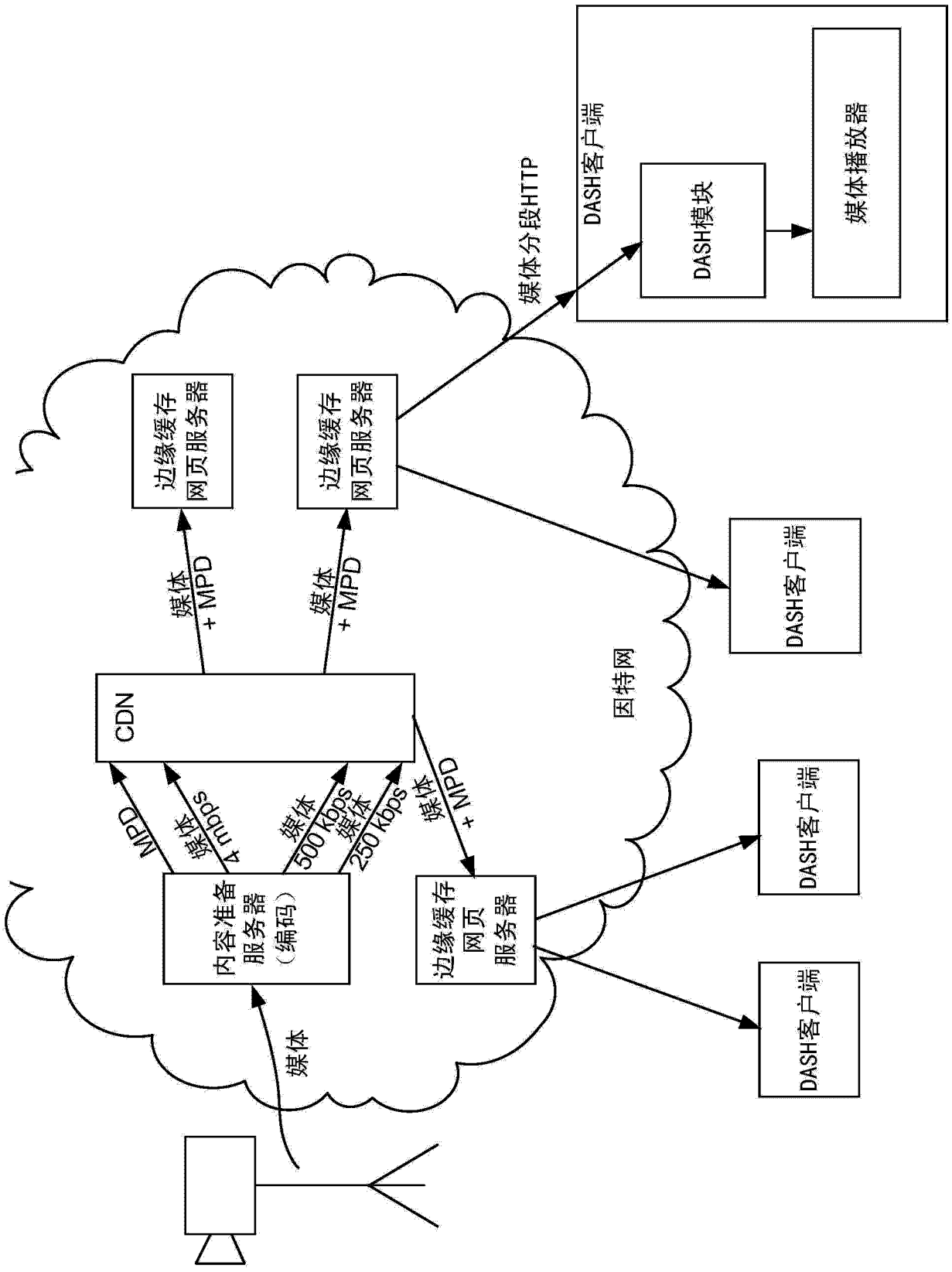

DASH客户端
DASH模块
媒体播放器
媒体分段HTTP
边缘缓存
网页服务器
边缘缓存
网页服务器
媒体
+ MPD
媒体
+ MPD
DASH客户端
因特网
CDN
MPD
媒体
4 mbps
媒体
500 kbps
媒体
250 kbps
媒体
+ MPD
内容准备
服务器
（编码）
边缘缓存
网页
服务器
DASH客户端
DASH客户端
媒体

图 1

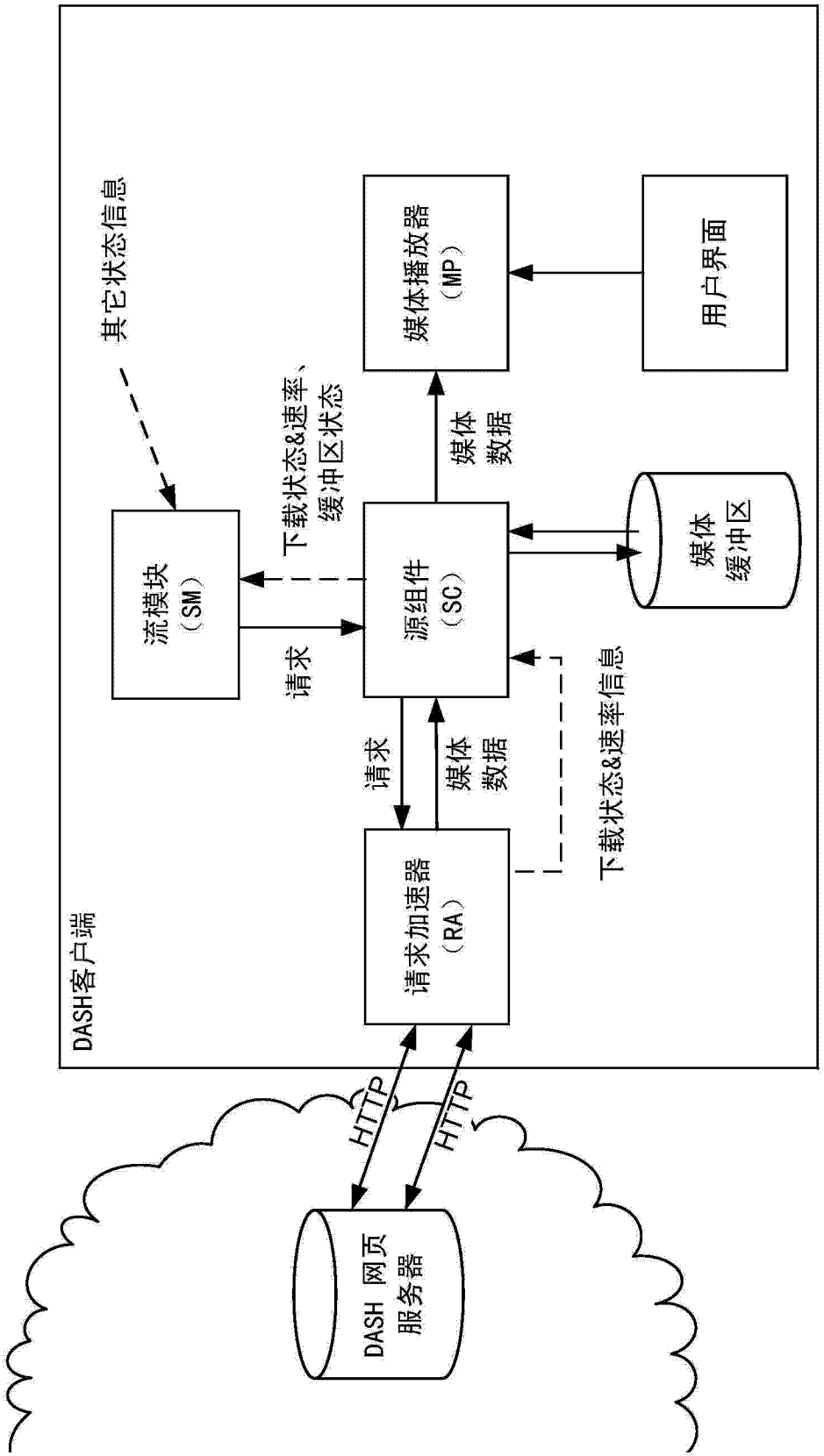
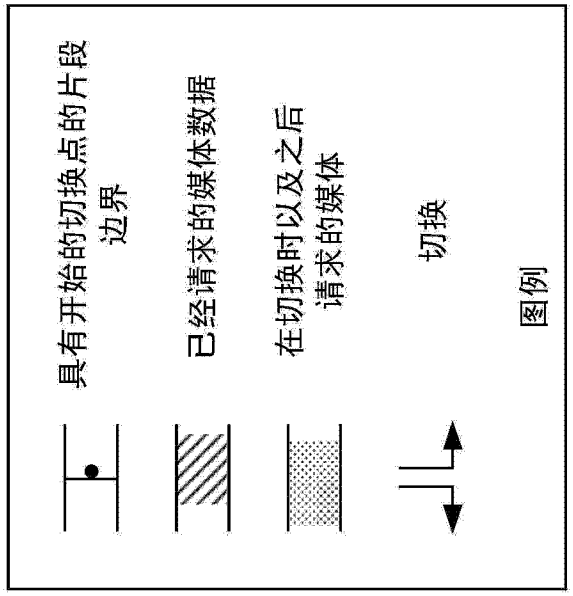

DASH客户端
其它状态信息
流模块（SM）
请求
下载状态&速率、缓冲区状态
源组件（SC）
媒体播放器（MP）
用户界面
媒体数据
媒体缓冲区
请求
媒体数据
请求加速器（RA）
下载状态&速率信息
HTTP
HTTP
DASH 网页服务器

图 2


具有开始的切换点的片段边界
已经请求的媒体数据
在切换时以及之后请求的媒体
切换
图例

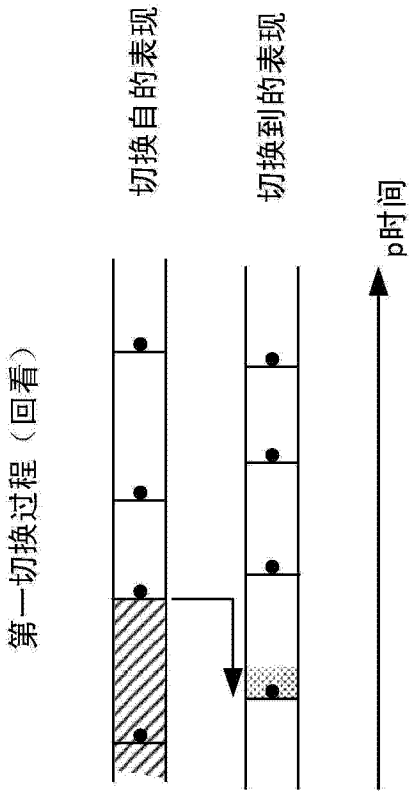

第一切换过程（回看）
切换自的表现
切换到的表现
p时间

图 3A

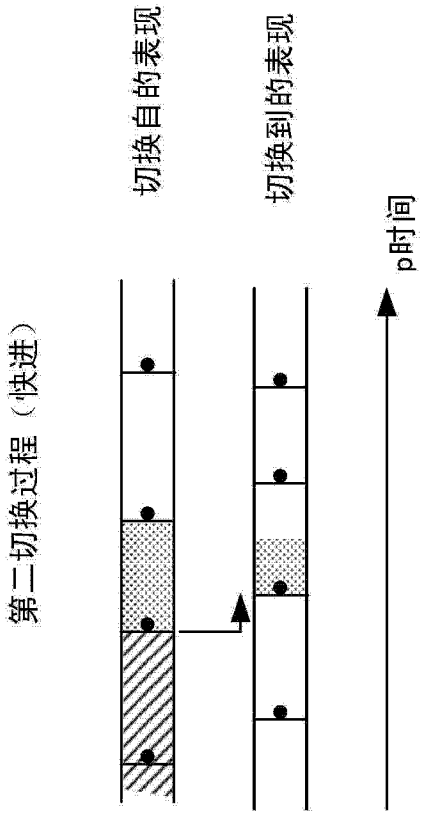

第二切换过程（快进）
切换自的表现
切换到的表现
p时间

图 3B

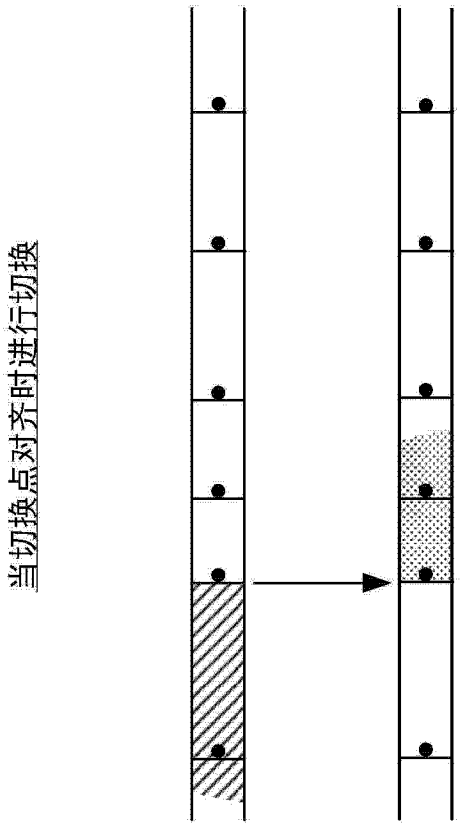


图 4

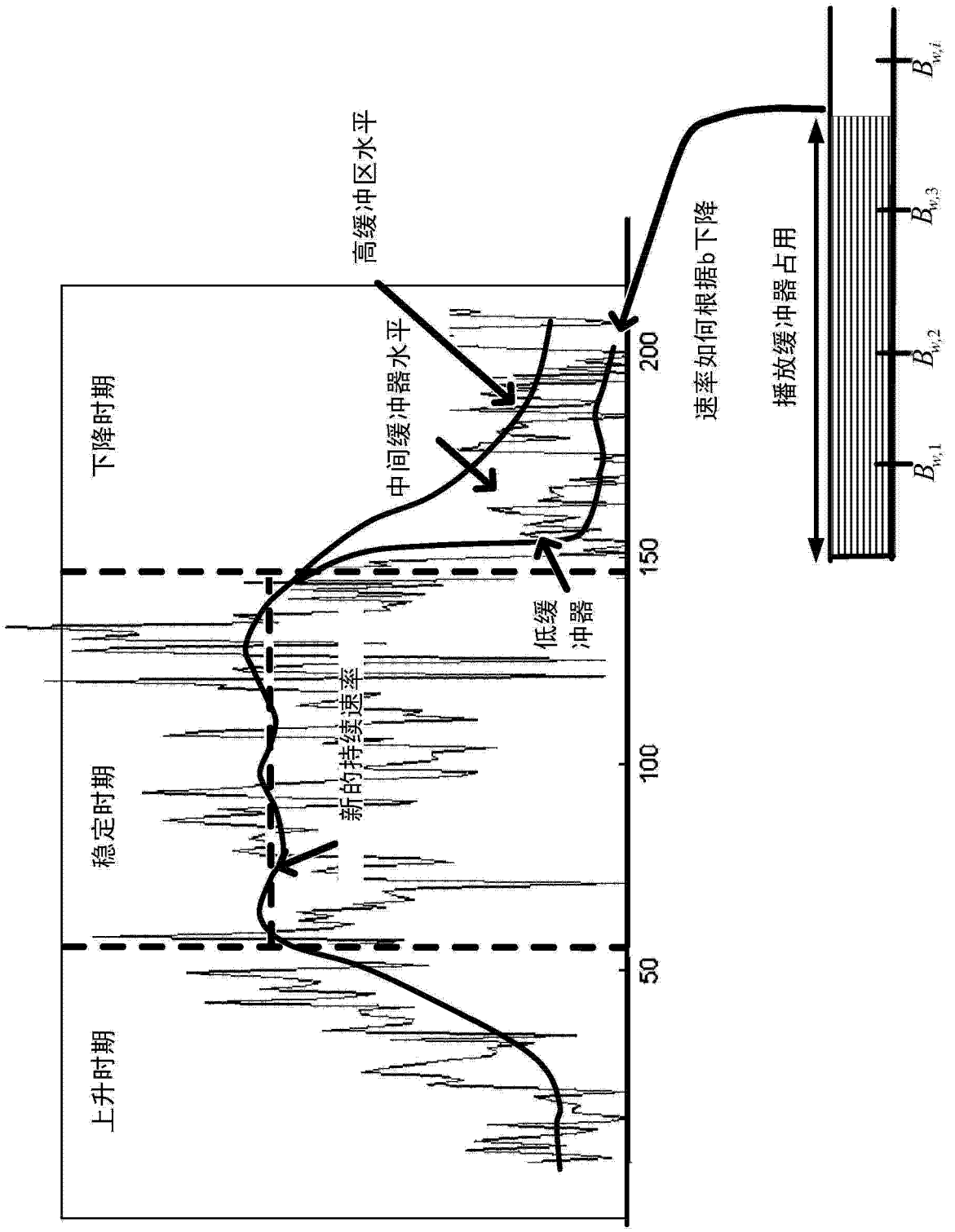

高缓冲区水平
中间缓冲器水平
低缓冲器
新的持续速率
速率如何根据b下降
播放缓冲器占用
下降时期
稳定时期
上升时期
200
150
100
50
$B_{w,1}$
$B_{w,2}$
$B_{w,3}$
$B_{w,i}$

图 5

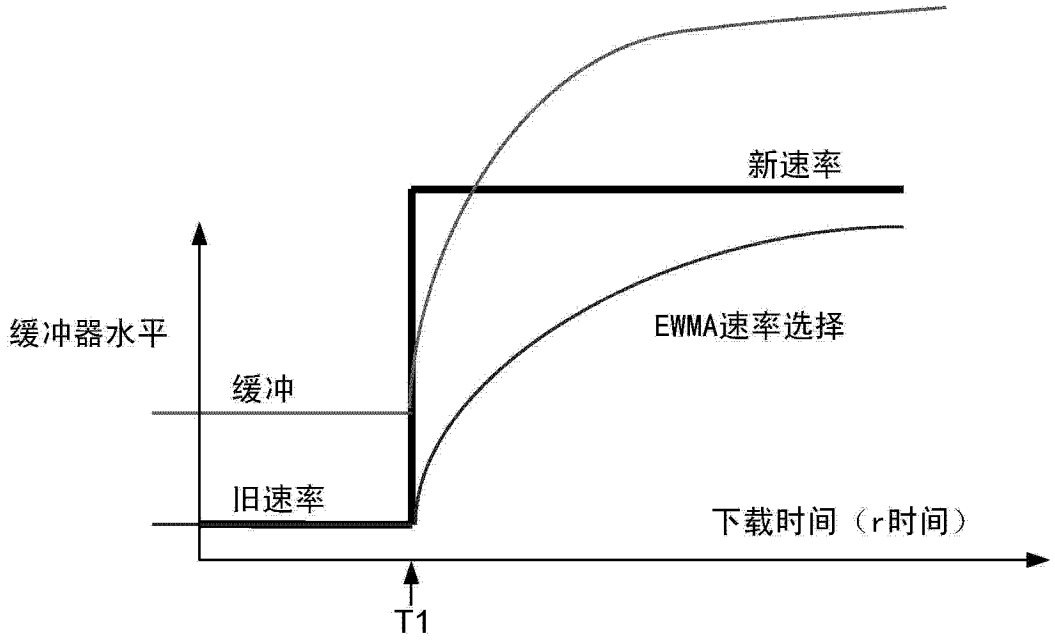

新速率
缓冲器水平
EWMA速率选择
缓冲
旧速率
下载时间（r时间）
T1

图 6

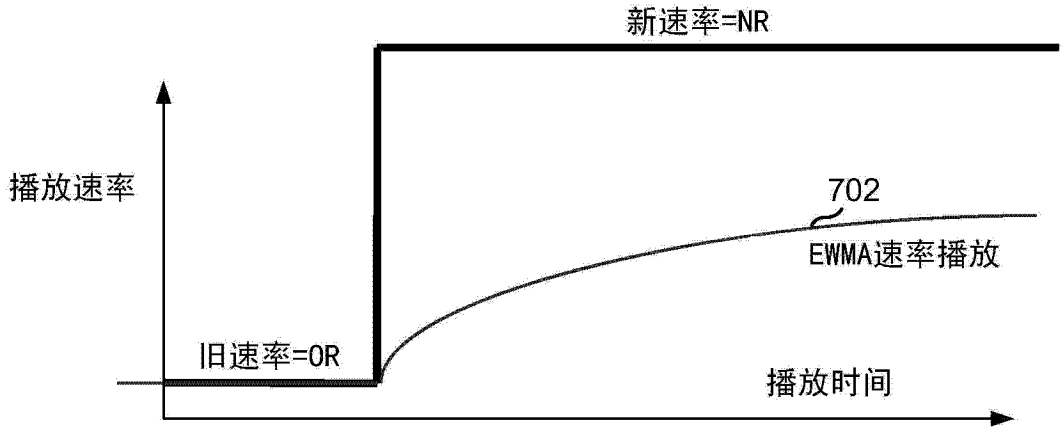

新速率=NR
播放速率
702
EWMA速率播放
旧速率=OR
播放时间

图 7

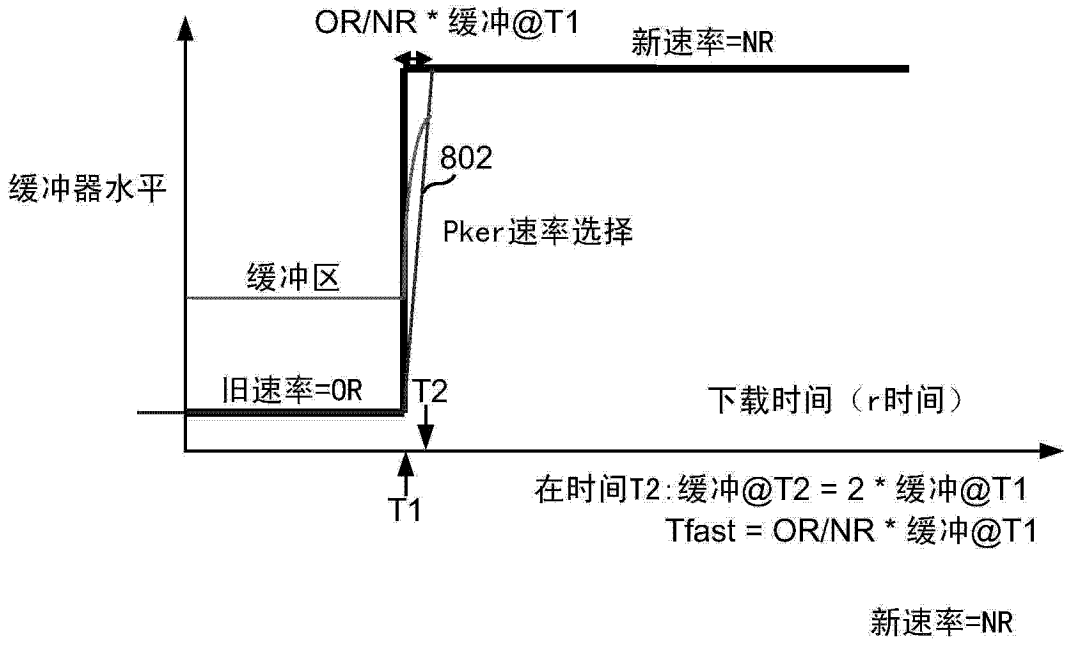

OR/NR * 缓冲@T1
新速率=NR
802
缓冲器水平
Pker速率选择
缓冲区
旧速率=OR
T2
下载时间（r时间）
T1
在时间T2:缓冲@T2 = 2 * 缓冲@T1
Tfast = OR/NR * 缓冲@T1
新速率=NR

图 8

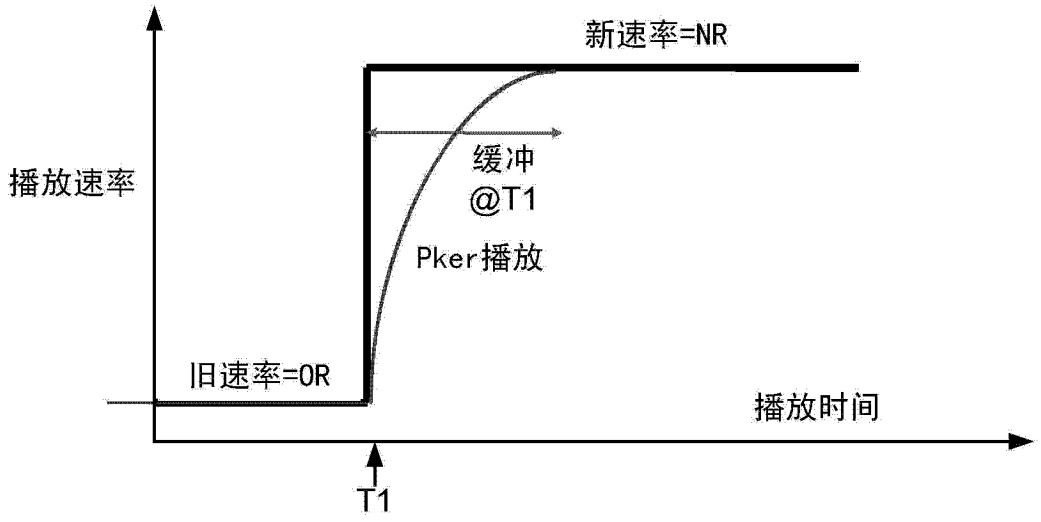

新速率=NR
播放速率
缓冲
@T1
Pker播放
旧速率=OR
播放时间
T1

图 9

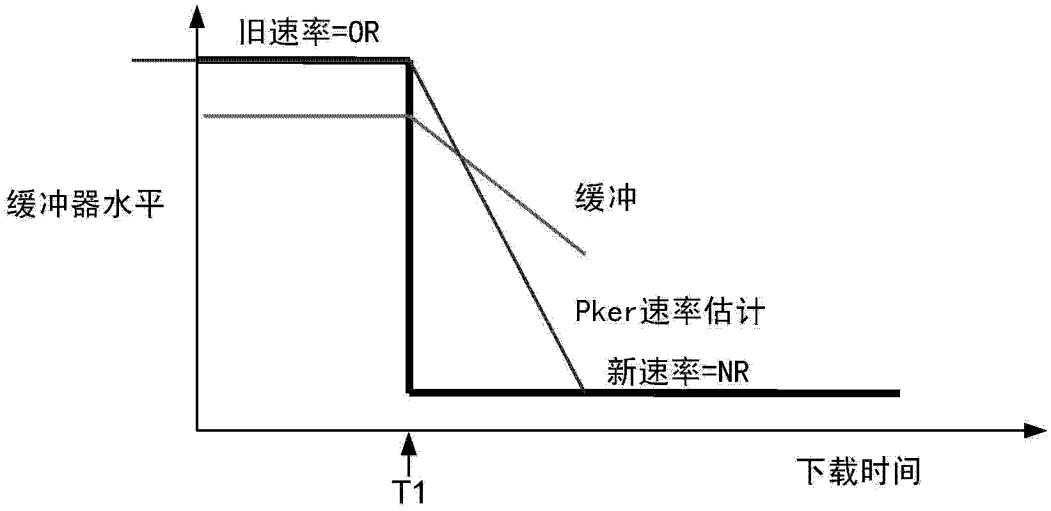

旧速率=OR
缓冲器水平
缓冲
Pker速率估计
新速率=NR
T1
下载时间

图 10

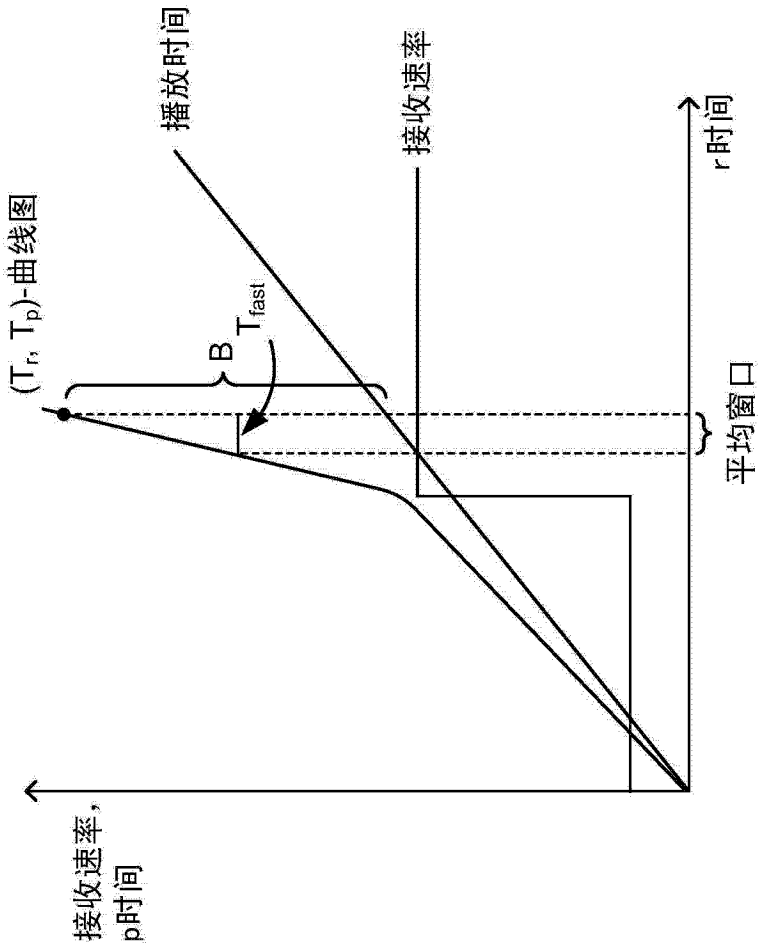

播放时间
接收速率
r时间
(Tr, Tp)-曲线图
B
Tfast
平均窗口
接收速率，
p时间

图 11

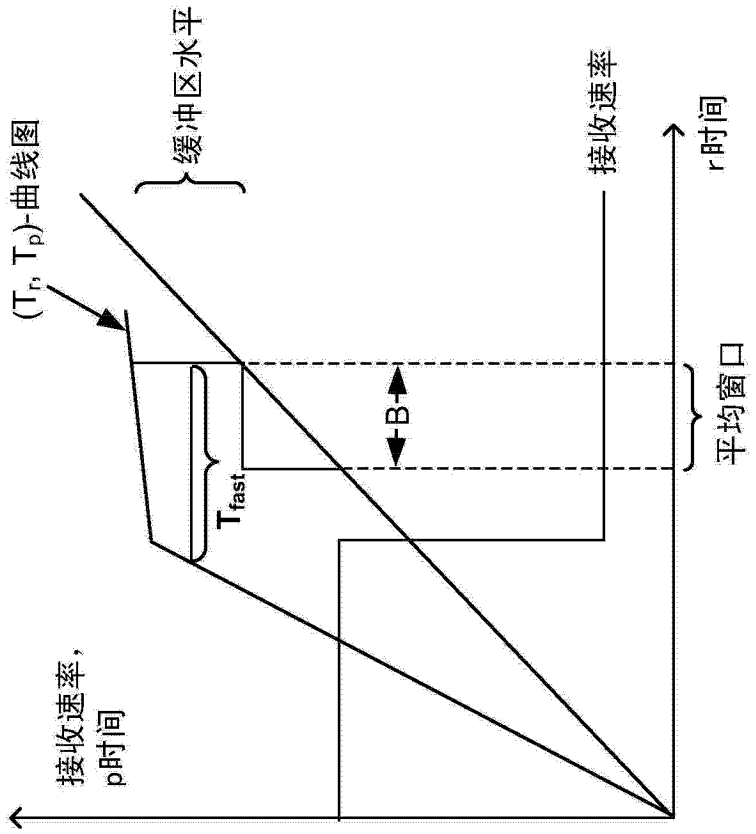


图 12

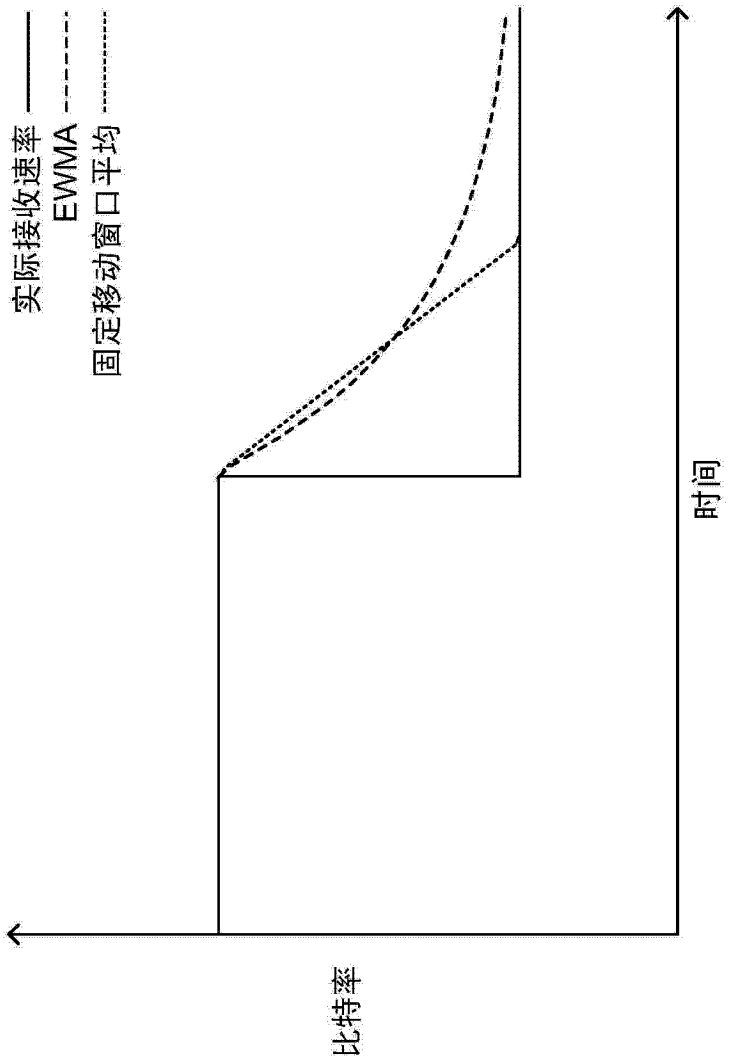


图 13

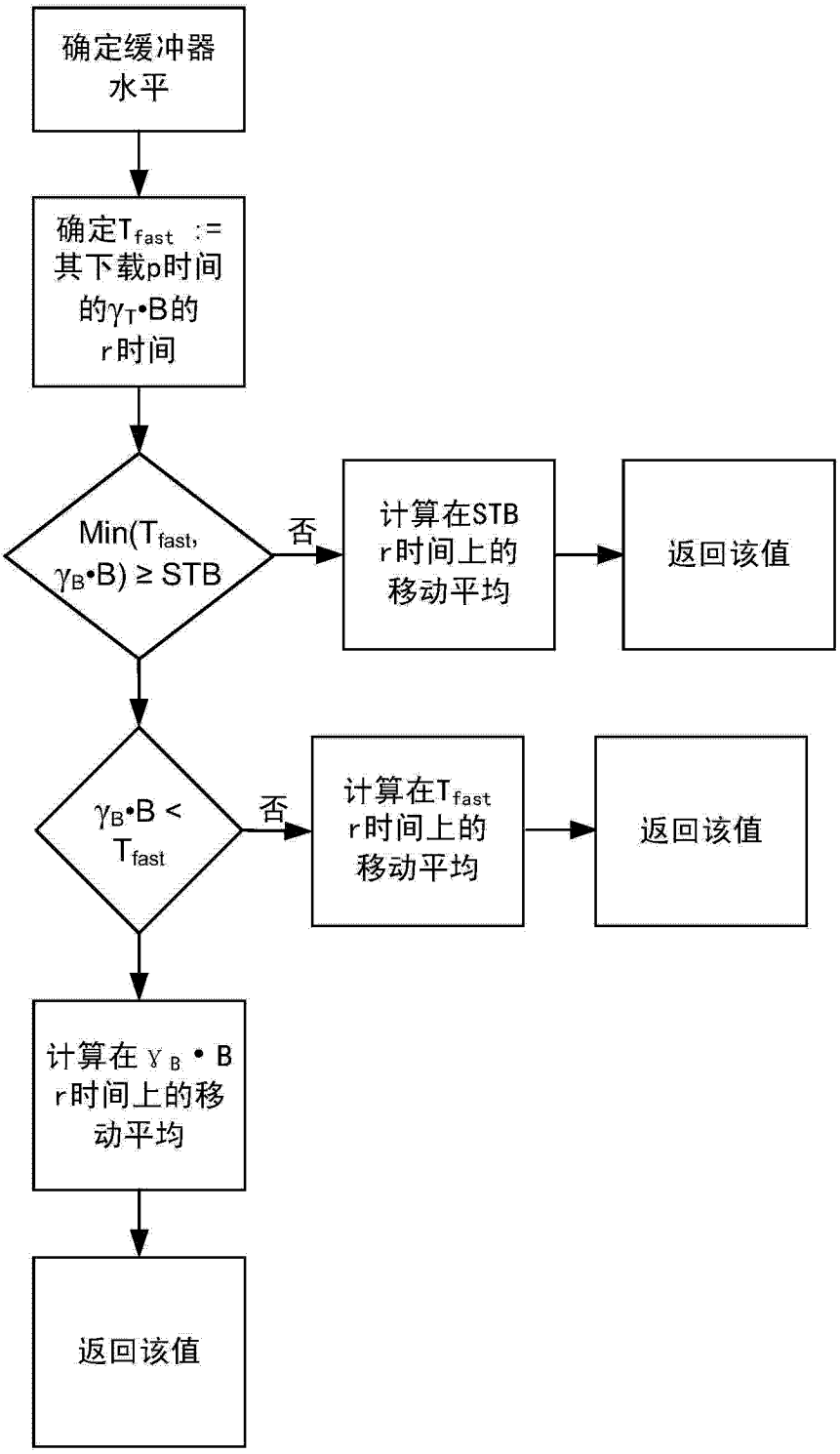

确定缓冲器水平
确定T_{fast} := 其下载p时间的γ_T•B的r时间
Min(T_{fast}, γ_B•B) ≥ STB
否
计算在STB r时间上的移动平均
返回该值
γ_B•B < T_{fast}
否
计算在T_{fast} r时间上的移动平均
返回该值
计算在γ_B•B r时间上的移动平均
返回该值

图 14

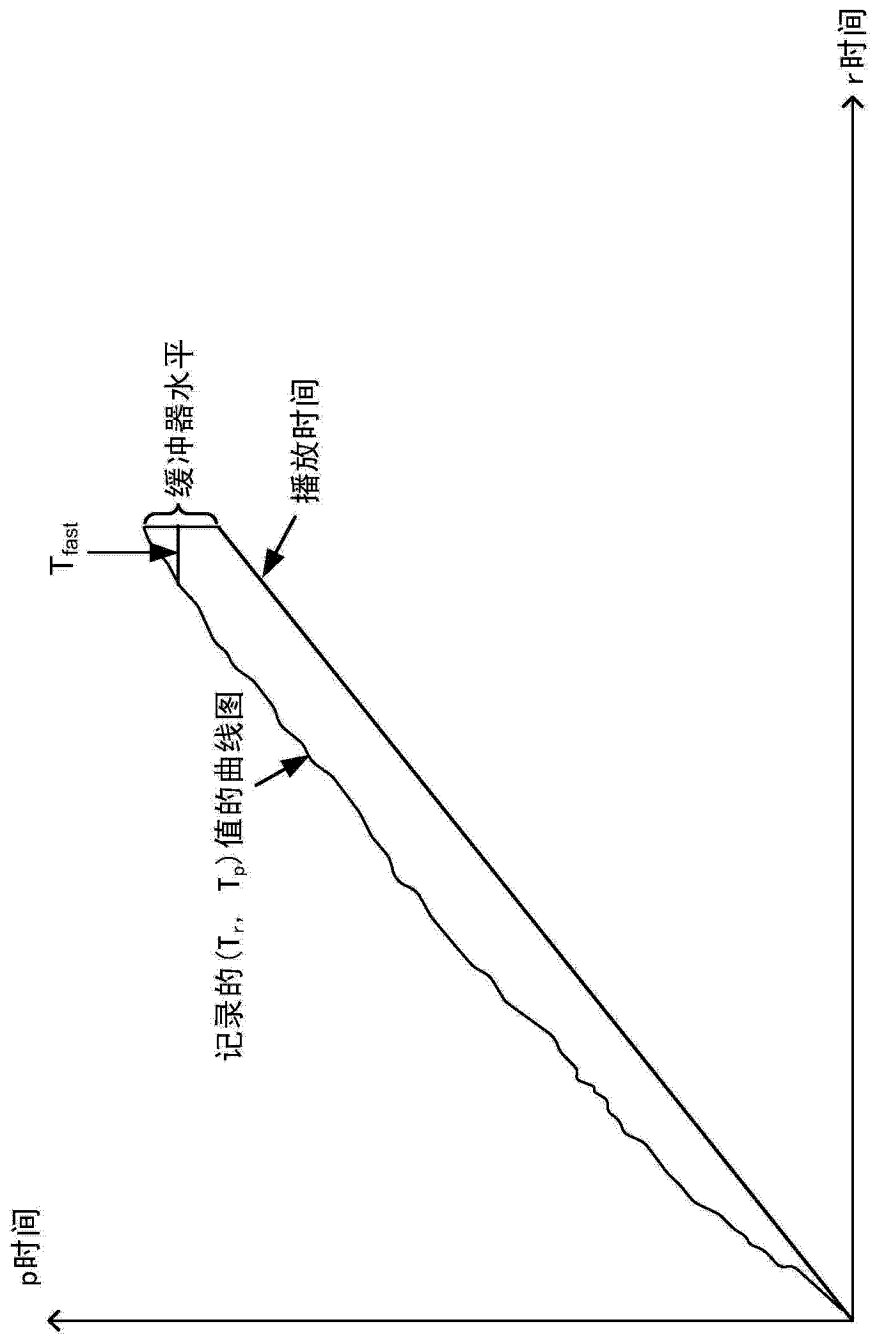


图 15

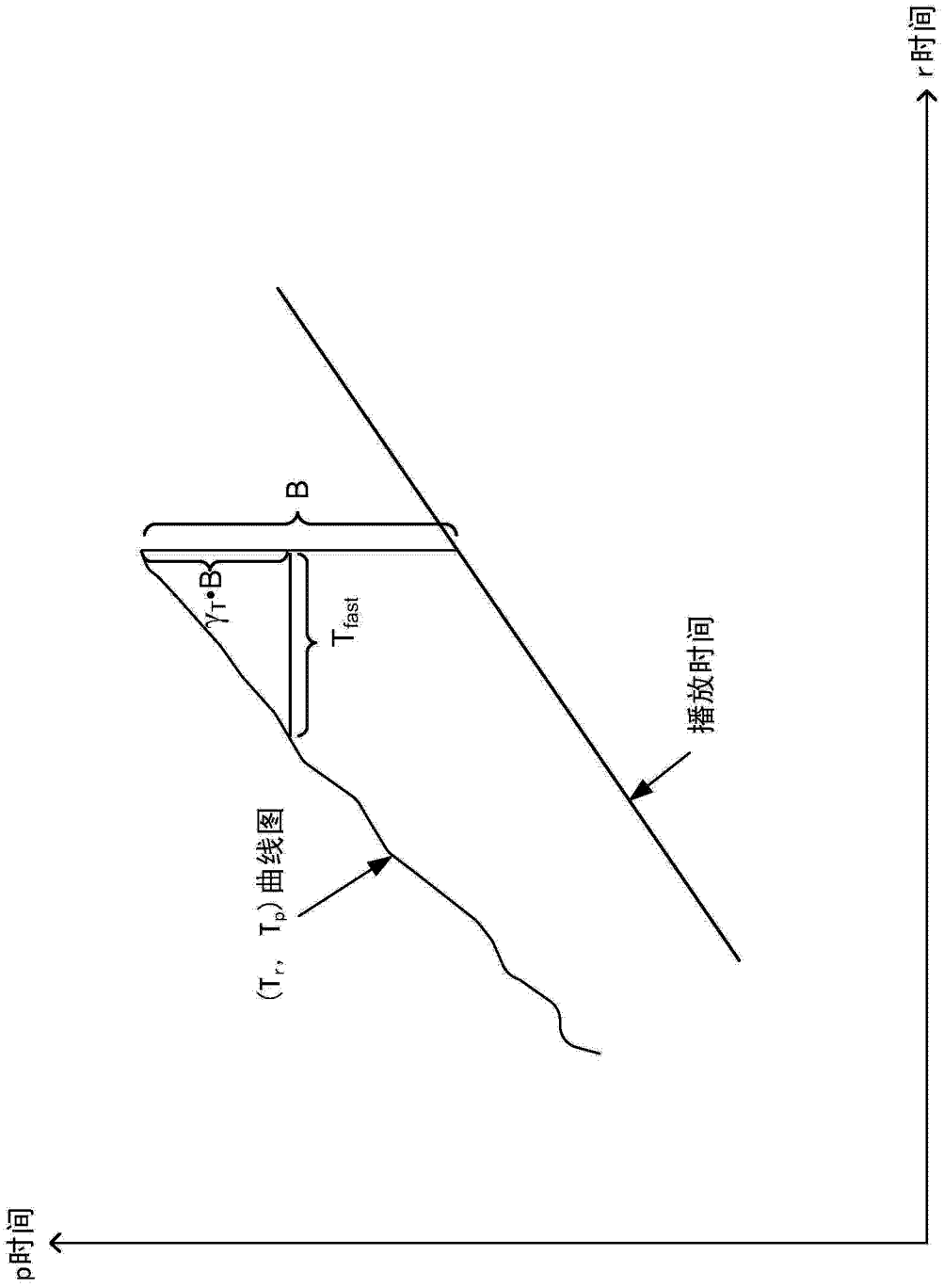


图 16

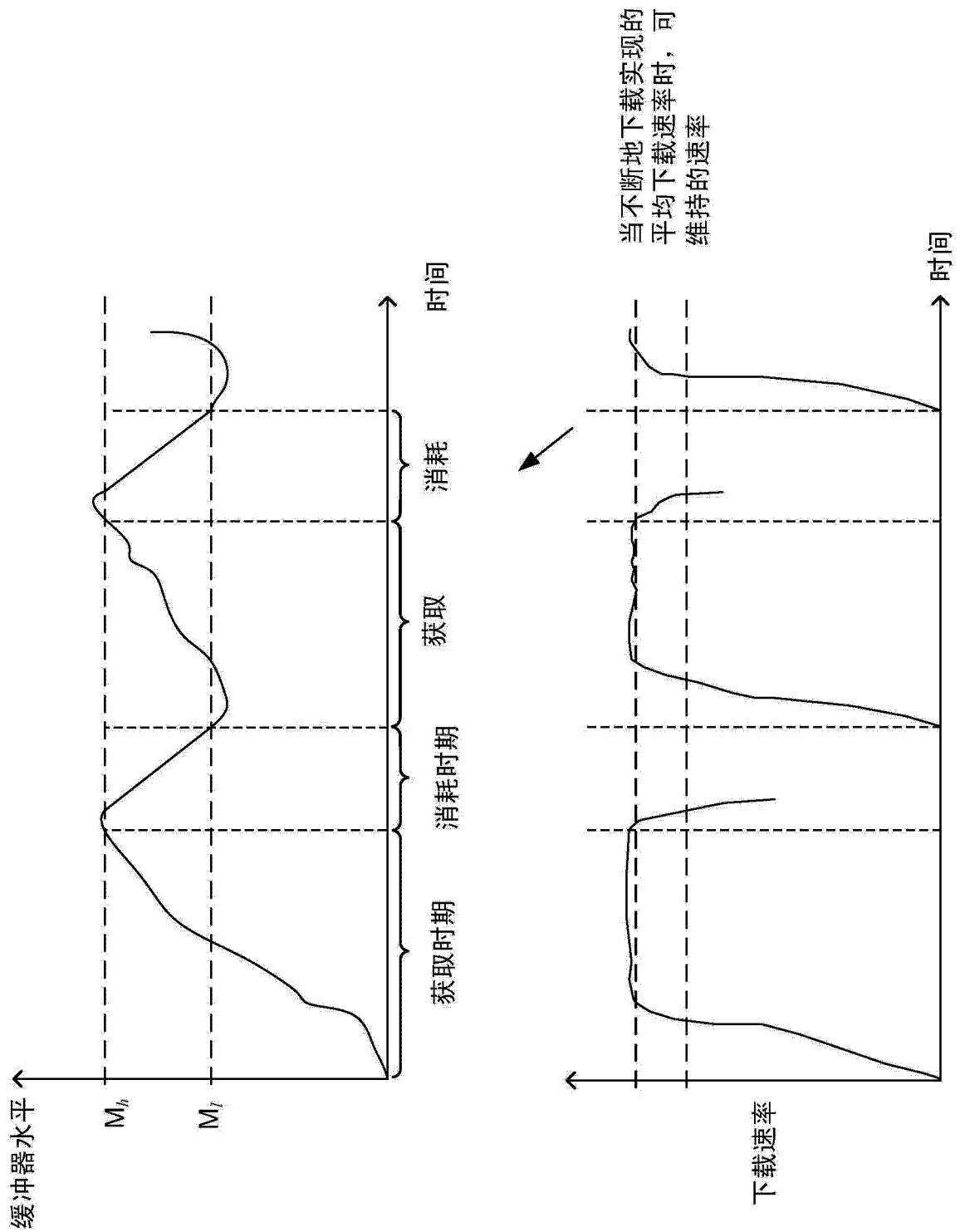

缓冲器水平
M_h
M_l
时间
获取时期
消耗时期
获取
消耗
当不断地下载实现的平均下载速率时，可维持的速率
时间
下载速率

图 17

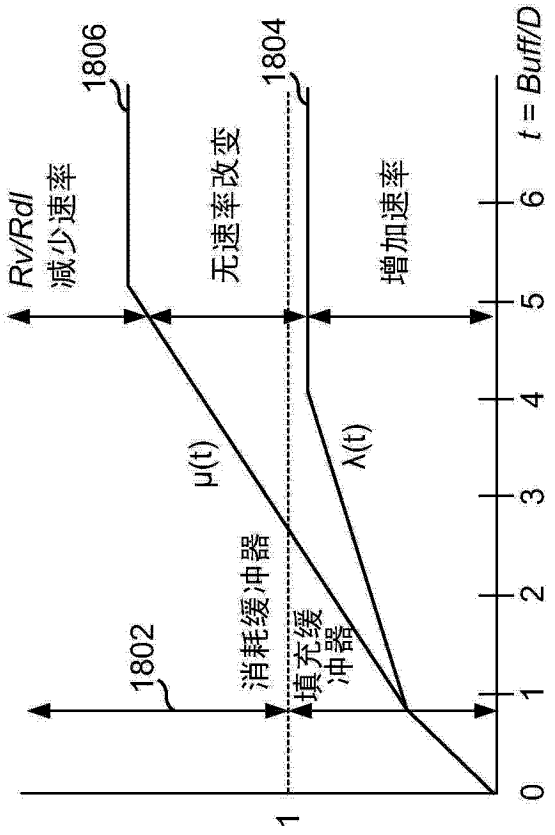

1806
1804
Rv/Rdl
减少速率
无速率改变
增加速率
t = Buff/D
μ(t)
λ(t)
1802
消耗缓冲器
填充缓冲器
0
1
2
3
4
5
6
1

图 18

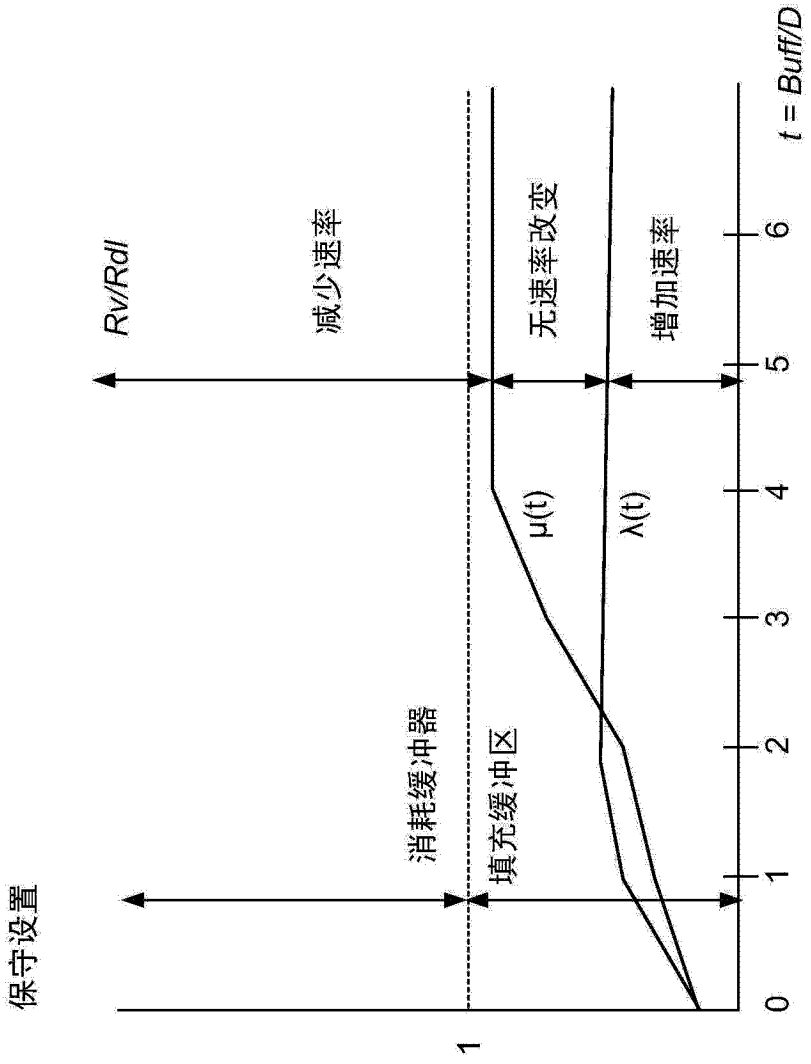

保守设置
消耗缓冲器
填充缓冲区
减少速率
无速率改变
增加速率
Rv/Rdl
μ(t)
λ(t)
1
0
1
2
3
4
5
6
t = Buff/D

图 19

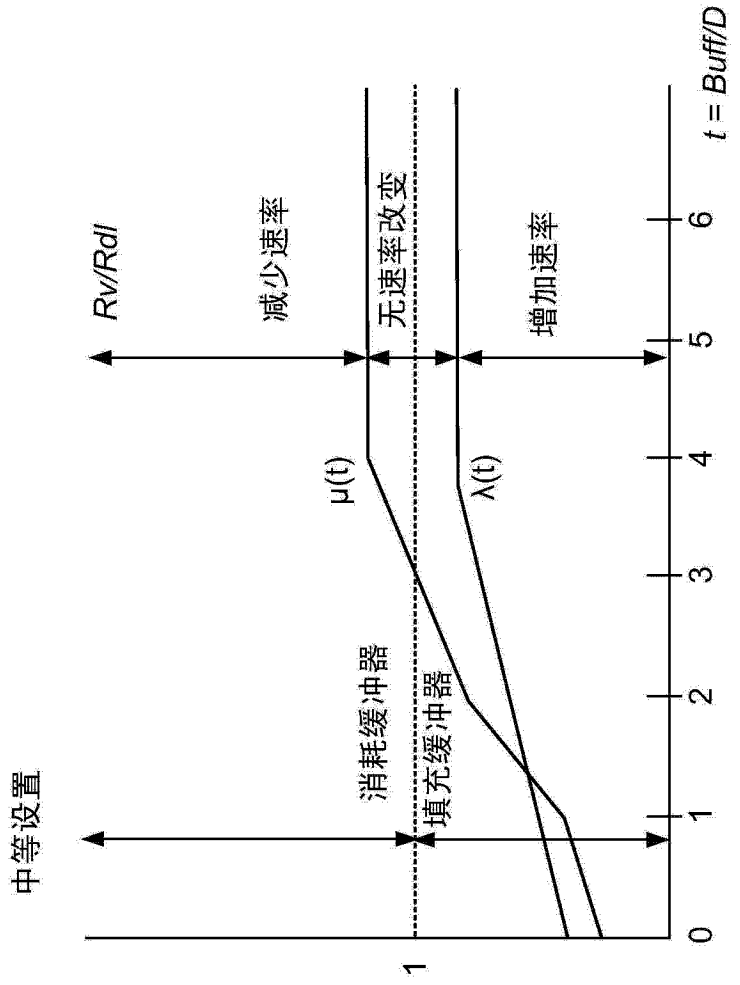

Rv/Rdl
减少速率
无速率改变
增加速率
μ(t)
λ(t)
消耗缓冲器
填充缓冲器
中等设置
1
0
1
2
3
4
5
6
t = Buff/D

图 20

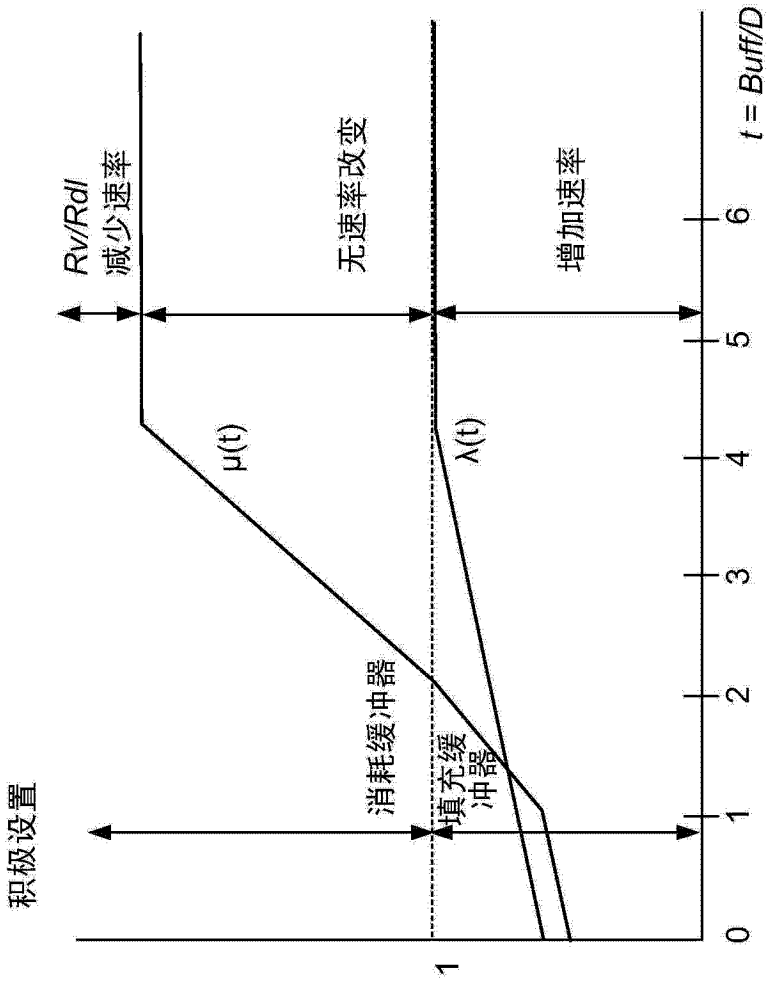

积极设置
消耗缓冲器
填充缓冲器
μ(t)
λ(t)
Rv/Rdl
减少速率
无速率改变
增加速率
t = Buff/D
0
1
2
3
4
5
6
1

图 21

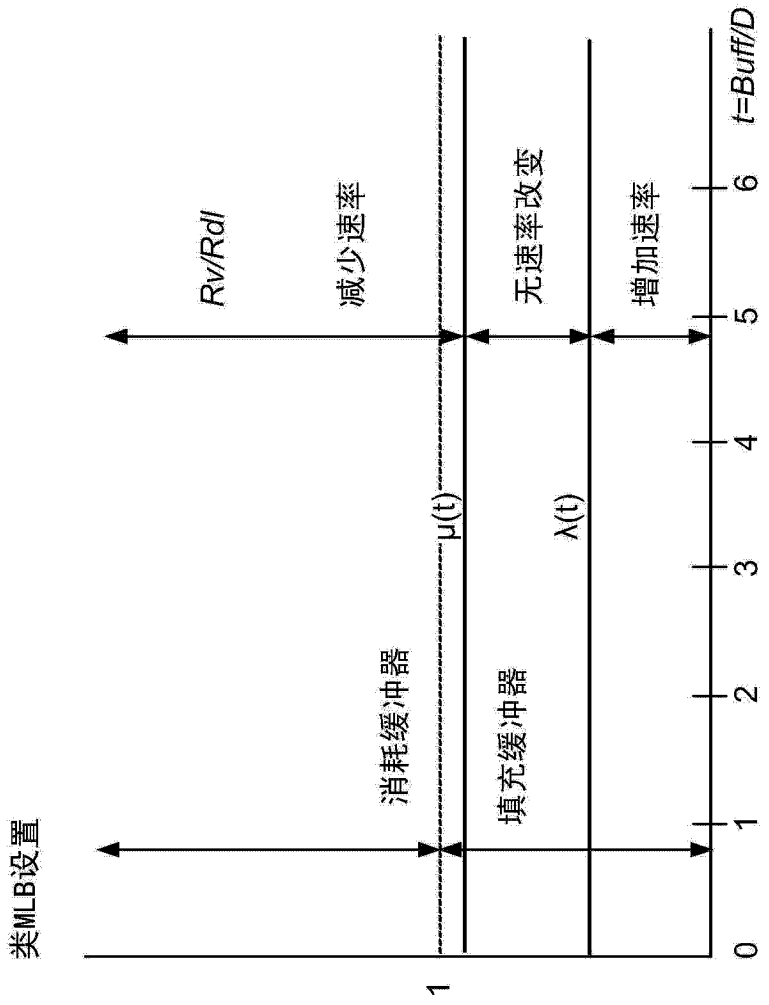

类MLB设置
消耗缓冲器
填充缓冲器
μ(t)
λ(t)
减少速率
无速率改变
增加速率
Rv/Rdl
0
1
2
3
4
5
6
t=Buff/D
1

图 22

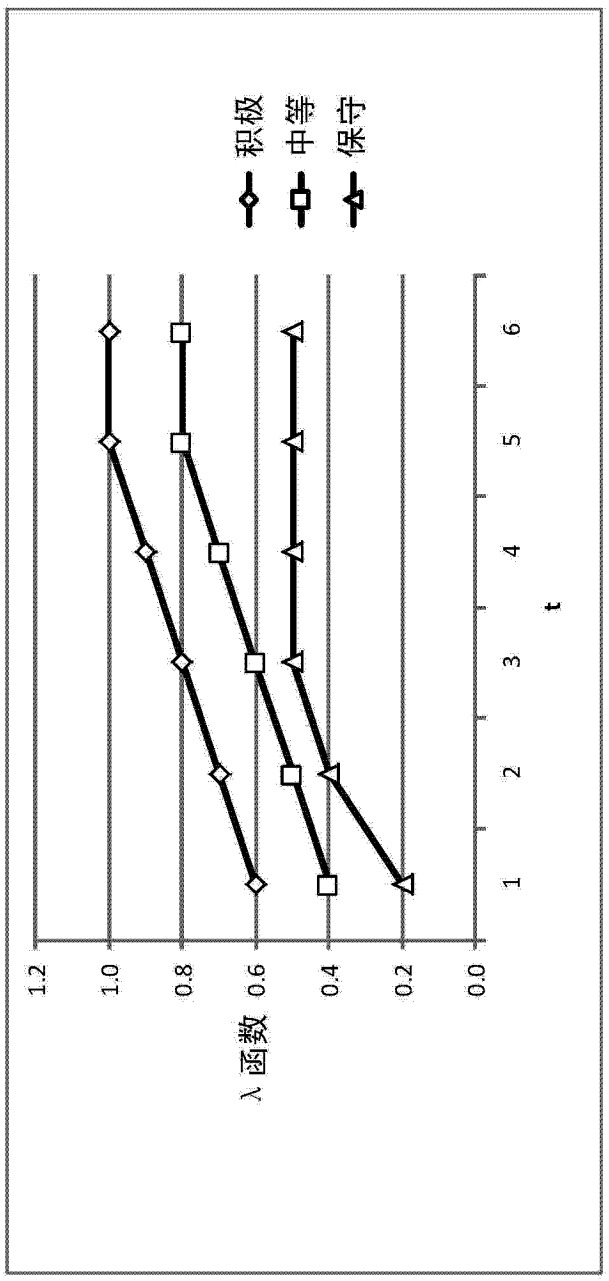


图 23

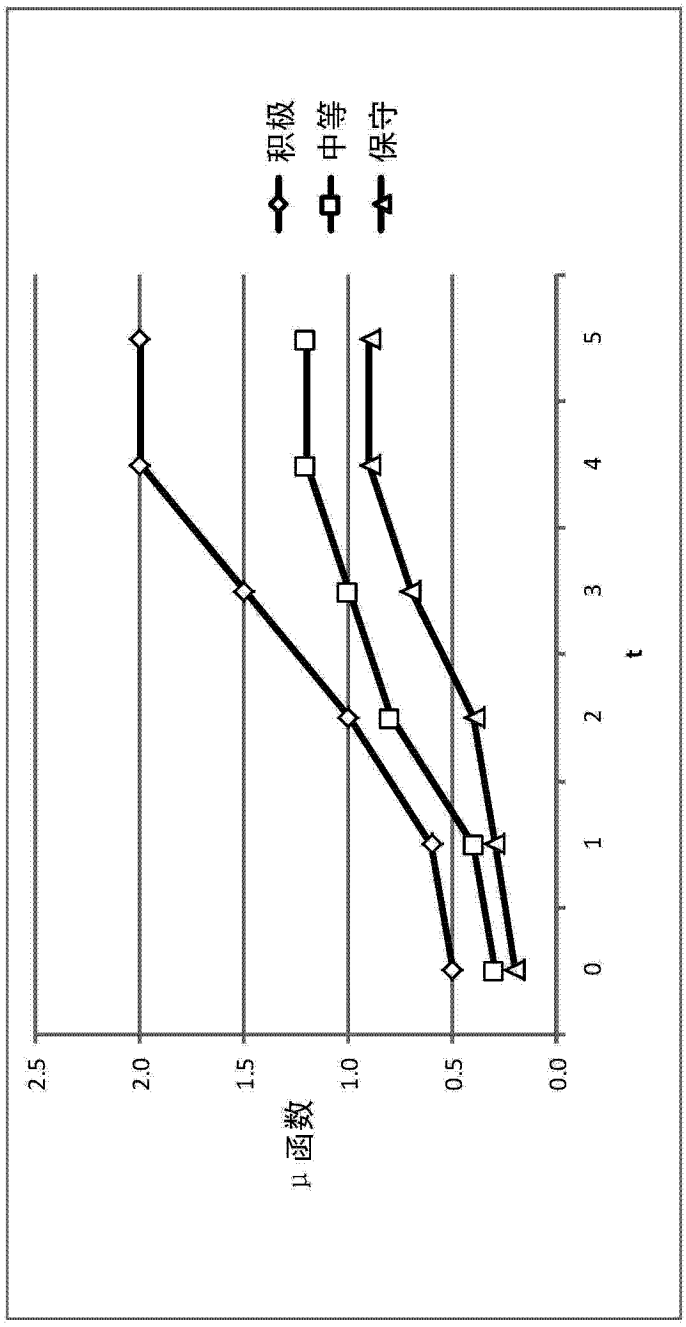


图 24

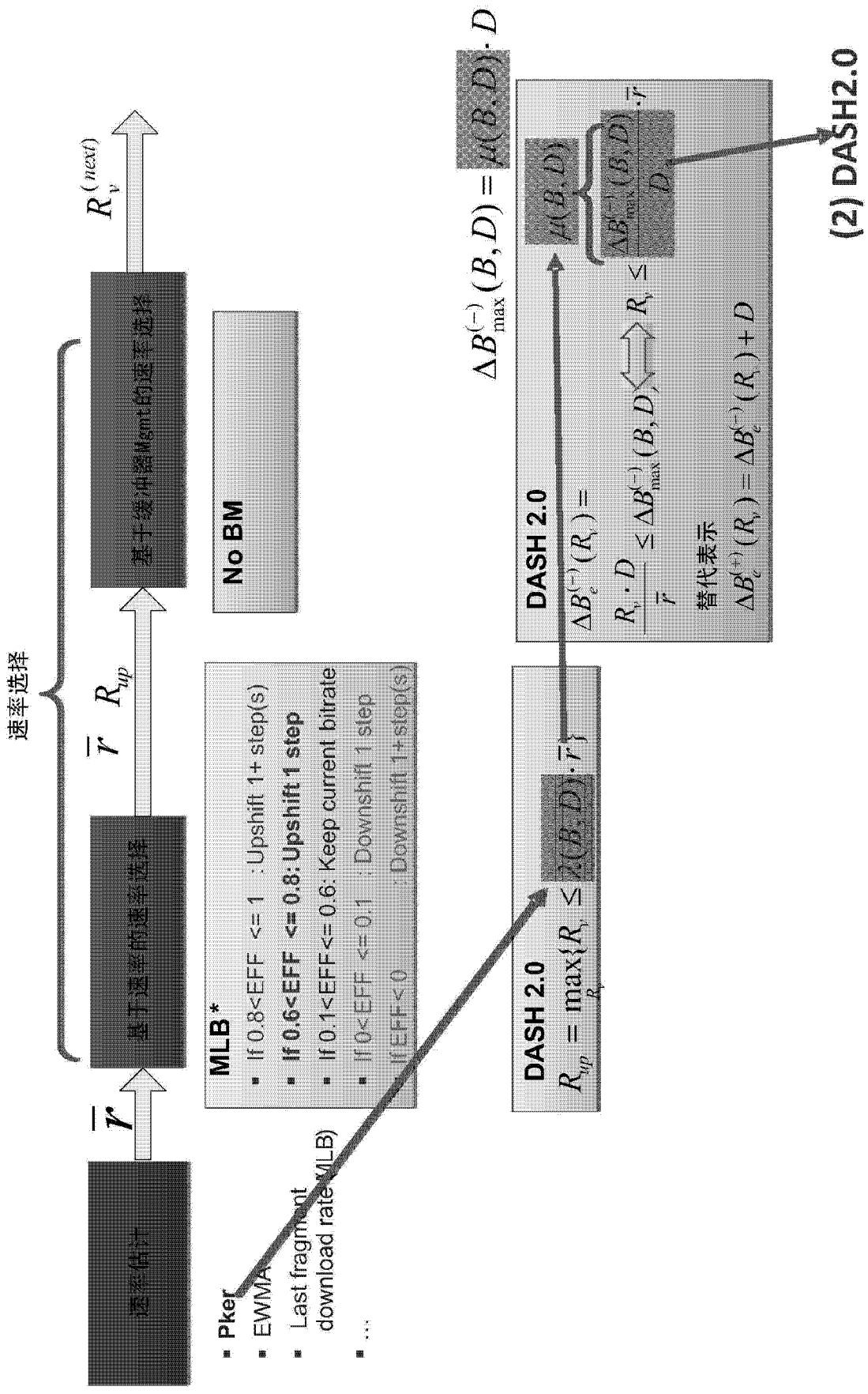

速率选择
速率估计
基于速率的速率选择
基于缓冲器Mgmt的速率选择
$\bar{r}$
R_{up}
$R_v^{(next)}$
Pker
EWMA
Last fragment download rate (MLB)
...
MLB*
If 0.8<EFF <= 1 :Upshift 1+ step(s)
If 0.6<EFF <= 0.8: Upshift 1 step
If 0.1<EFF<= 0.6: Keep current bitrate
If 0<EFF <= 0.1 :Downshift 1 step
If EFF< 0 :Downshift 1+step(s)
No BM
DASH 2.0
$R_{up} = \max_{R_v} \{R_v \le \mu(B,D)\cdot\bar{r}\}$
DASH 2.0
$\Delta B_e^{(-)}(R_v) = \frac{R_v \cdot D}{\bar{r}} \le \Delta B_{\max}^{(-)}(B,D), R_v \le \mu(B,D)\cdot\bar{r}$
替代表示
$\Delta B_e^{(+)}(R_v) = \Delta B_e^{(-)}(R_v) + D$
$\Delta B_{\max}^{(-)}(B,D) = \mu(B,D)\cdot D$
$\mu(B,D)$
$\frac{\Delta B_{\max}^{(-)}(B,D)}{D_n}\cdot\bar{r}$
(2) DASH2.0

图 25

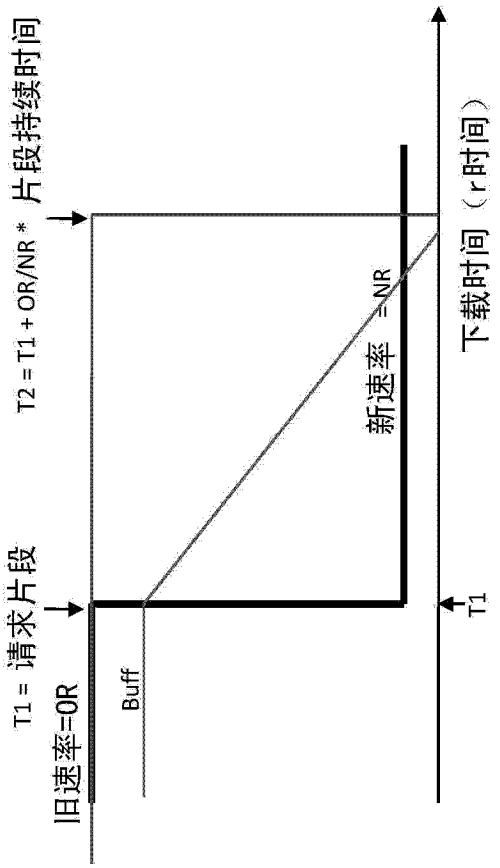


图 26

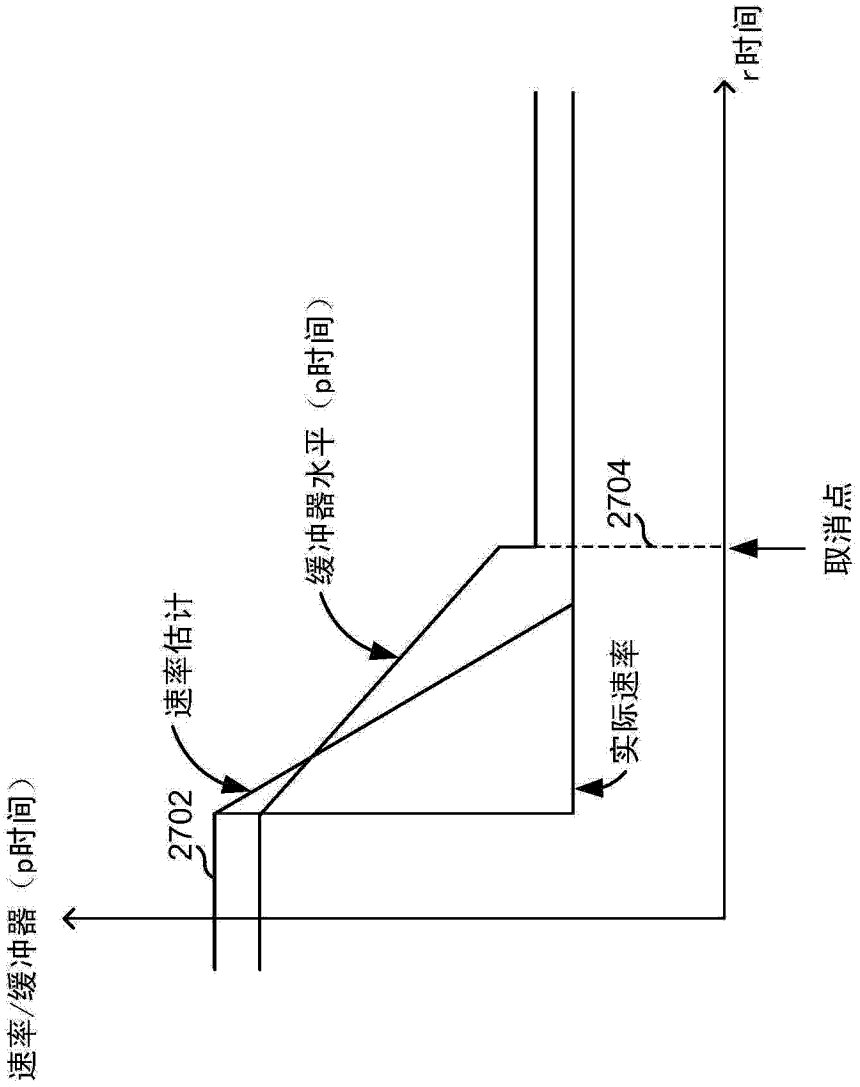


图 27

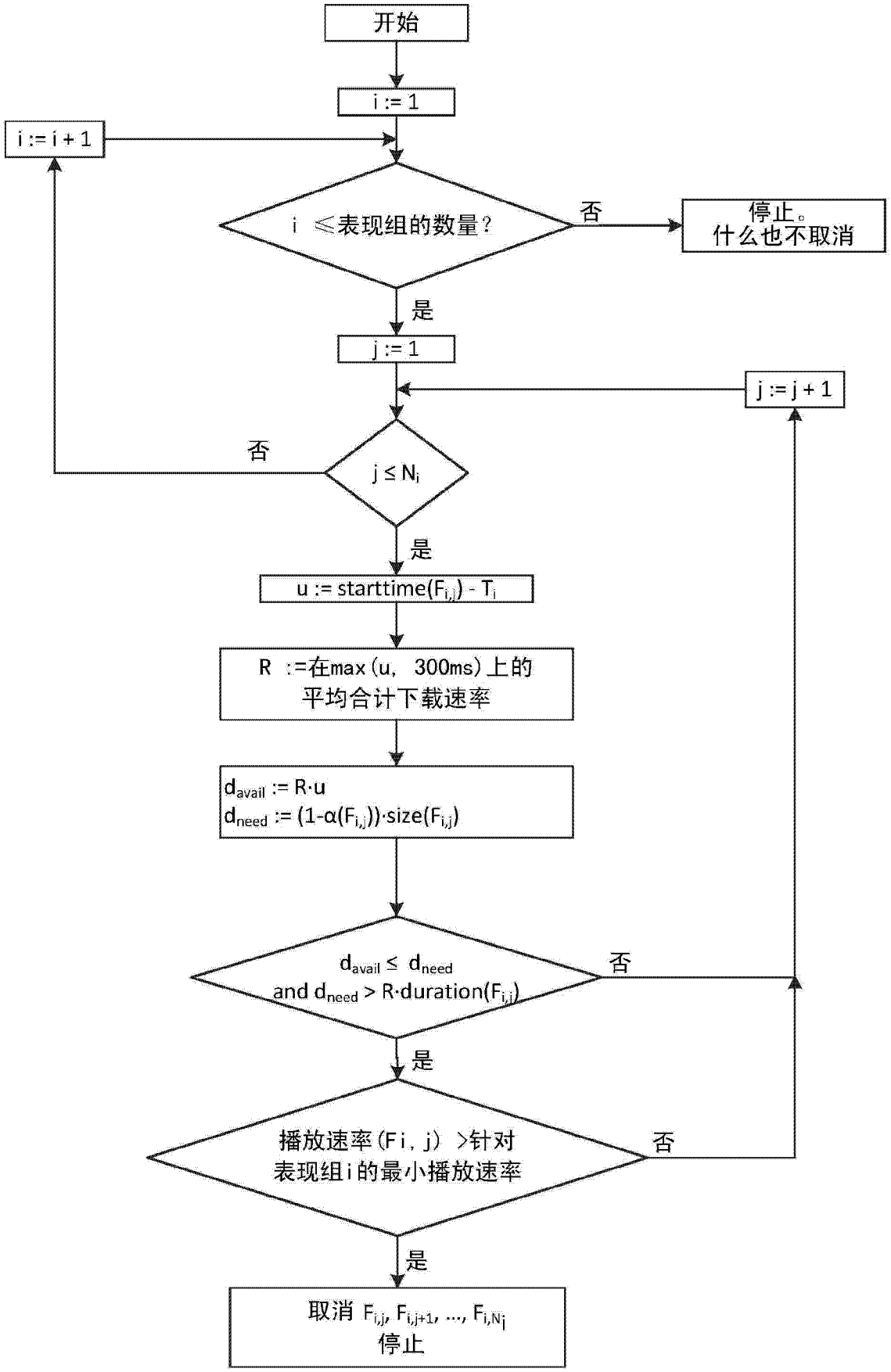

开始
i := 1
i := i + 1
i ≤表现组的数量？
否
停止。
什么也不取消
是
j := 1
j := j + 1
否
j ≤ Ni
是
u := starttime(Fi,j) - Ti
R :=在max(u, 300ms)上的
平均合计下载速率
davail := R·u
dneed := (1-α(Fi,j))·size(Fi,j)
davail ≤ dneed
and dneed > R·duration(Fi,j)
否
是
播放速率(Fi, j) >针对
表现组i的最小播放速率
否
是
取消 Fi,j, Fi,j+1, ..., Fi,Ni
停止

图 28

```
for every representation group i do
    m ← smallest available rate for this group
    for j = 1, ... , N_i do
        if representation-rate(F_{i,j}) = m then
            continue
        end if
        u ← starttime(F_{i,j}) – T
        R ← aggregate rate over max(u, 300ms)
        if R cannot be computed then
            continue
        end if
        d_avail ← R • u
        d_need ← (1 – α (F_{i,j})) •·size(F_{i,j})
        if d_avail < d_need and d_need > R •·duration (F_{i,j}) then
            return (i,j)
        end if
    end for
end for
return nil
```

图 29

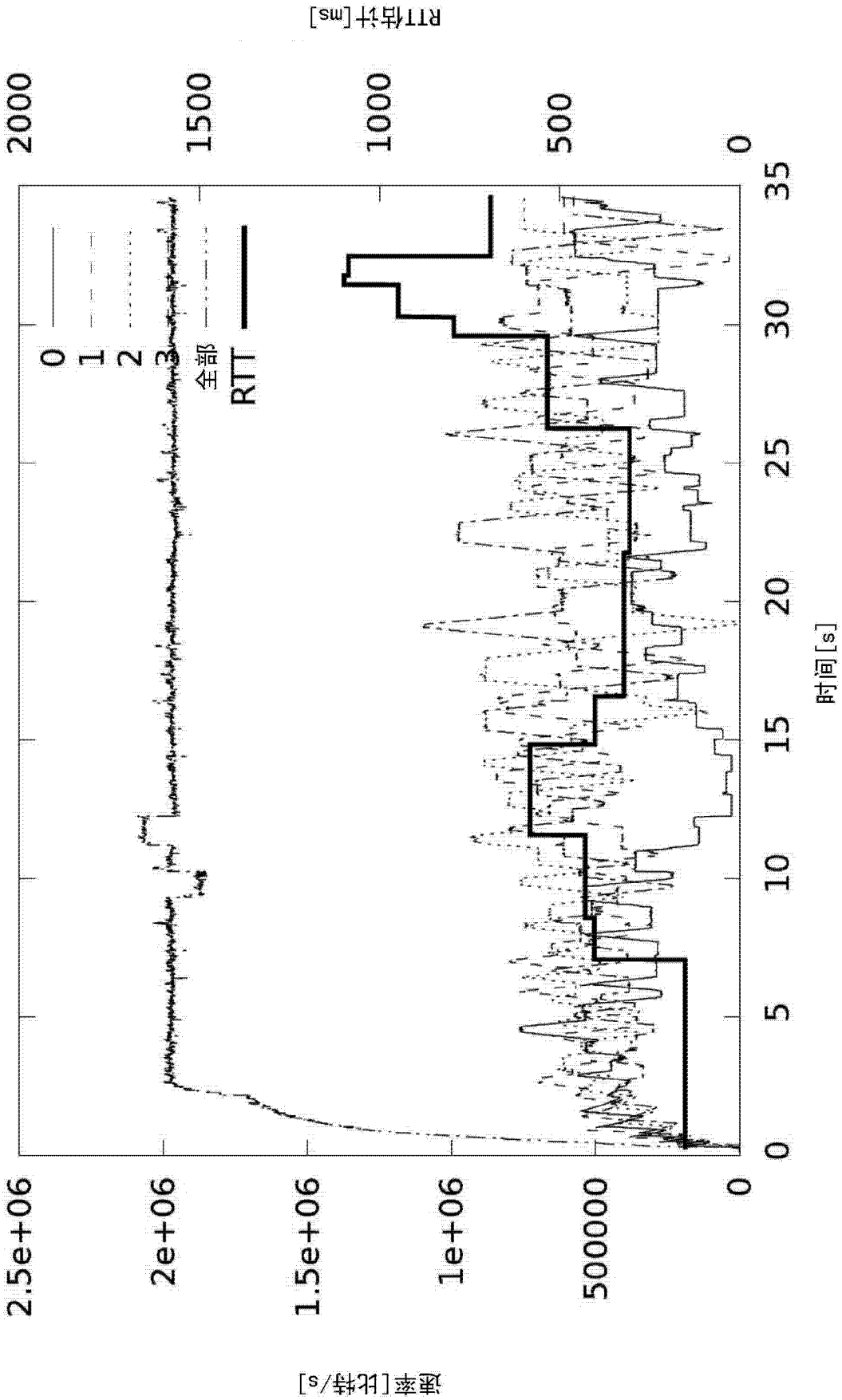

RTT估计[ms]
2000
1500
1000
500
0
0
1
2
3
全部
RTT
2.5e+06
2e+06
1.5e+06
1e+06
500000
0
0
5
10
15
20
25
30
35
时间[s]
速率[比特/s]

图 30

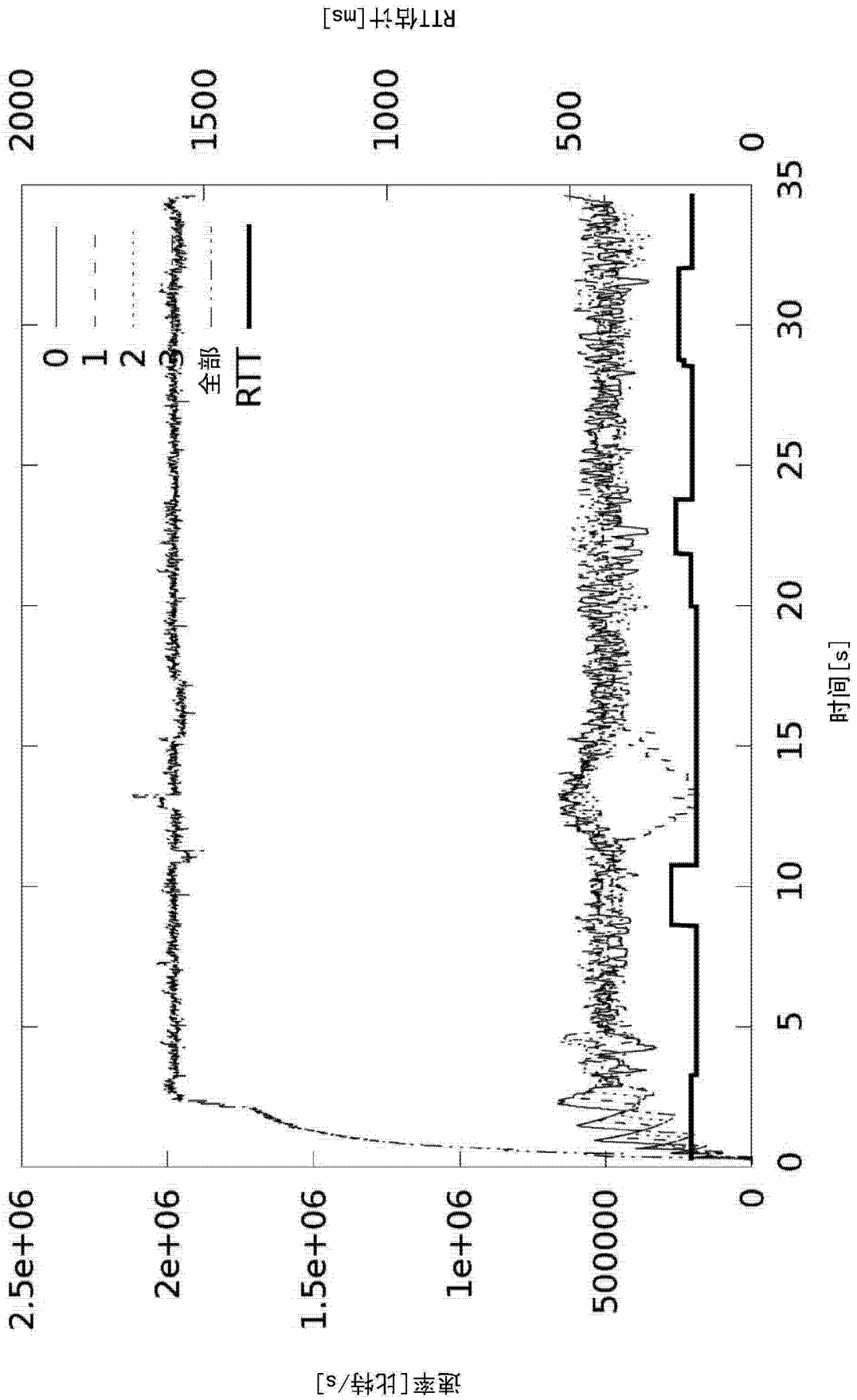

RTT估计[ms]
2000
1500
1000
500
0
0
1
2
3
全部
RTT
2.5e+06
2e+06
1.5e+06
1e+06
500000
0
0
5
10
15
20
25
30
35
时间[s]
速率[比特/s]

图 31

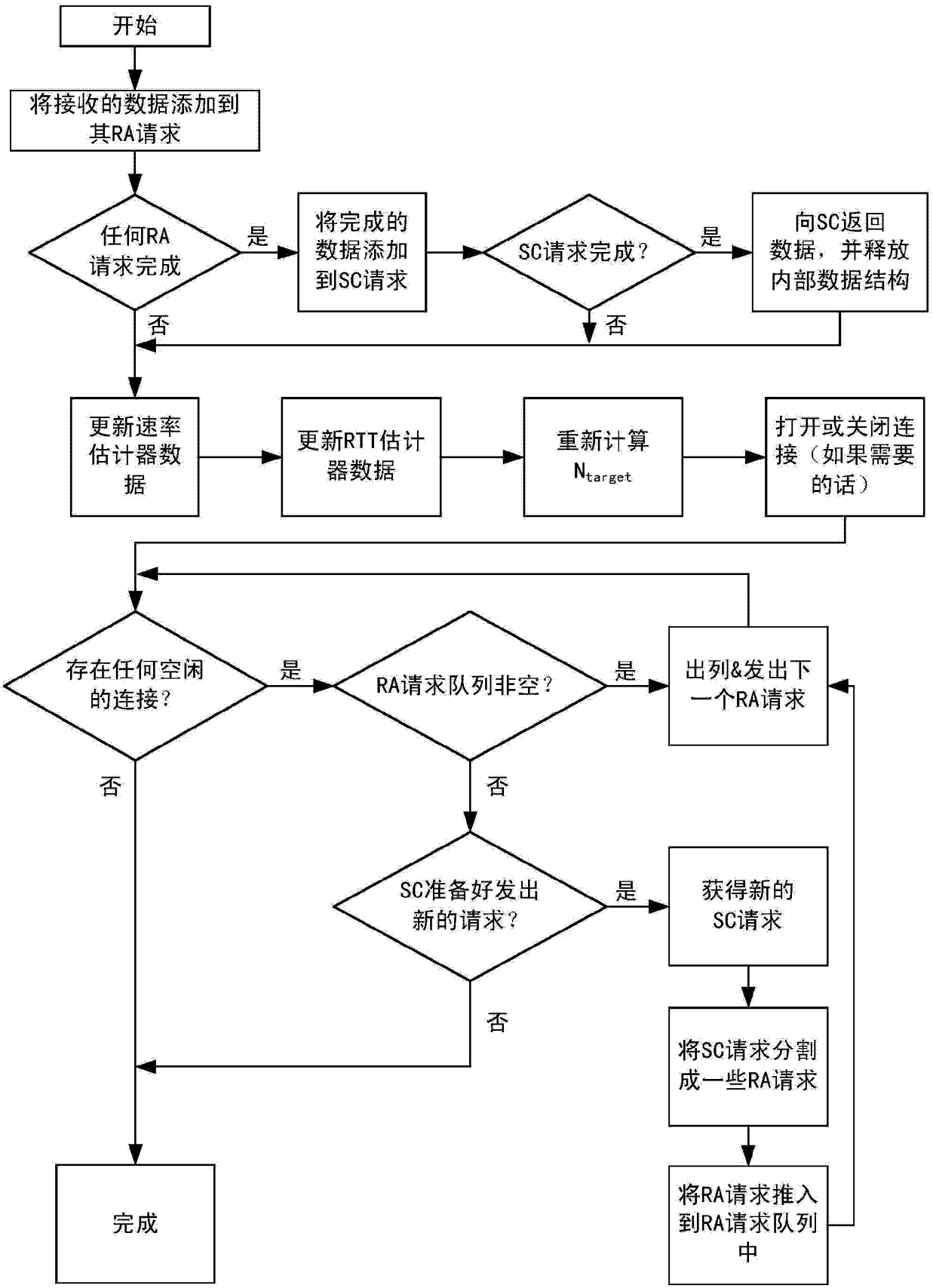

开始
将接收的数据添加到其RA请求
任何RA请求完成
是
将完成的数据添加到SC请求
SC请求完成？
是
向SC返回数据，并释放内部数据结构
否
否
更新速率估计器数据
更新RTT估计器数据
重新计算 N_{target}
打开或关闭连接（如果需要的话）
存在任何空闲的连接？
是
RA请求队列非空？
是
出列&发出下一个RA请求
否
否
SC准备好发出新的请求？
是
获得新的SC请求
否
将SC请求分割成一些RA请求
将RA请求推入到RA请求队列中
完成

图 32

$S \leftarrow$ 原始请求的大小
$N_{target} \leftarrow$ 连接的目标数量
$R \leftarrow$ 在最后的T_{wn}上的速率估计
$RTT \leftarrow$ RTT估计
if no valid RTT estimate or no valid rate estimate is known **then**
    $n \leftarrow N_{target}$
    return n
end if
$D \leftarrow \max(D_{min}, c_s \cdot RTT)$
if $S < R \cdot D$ **then**
    $n \leftarrow \left\lfloor \frac{N_{target} \cdot S}{R \cdot D} \right\rfloor$
else
    $n \leftarrow N_{target}$
end if
return n

图 33

```
a := 1/(n + 1)

Let v[1], ..., v[n - 1] be iid random variables chosen uniformly
in the range [0,1]

Sort the v[] in ascending order

for i = 1, ..., n - 1:

      w[i] := (1 - a)*i/n + a*v[i]

w[n] := 1

prev := 0

for i = 1, ..., n

      x := floor(w[i] * S)

      s[i] := x - prev

      prev := x

Sort the vector s[] in decreasing size order

return s[1], ..., s[n]
```

图 34

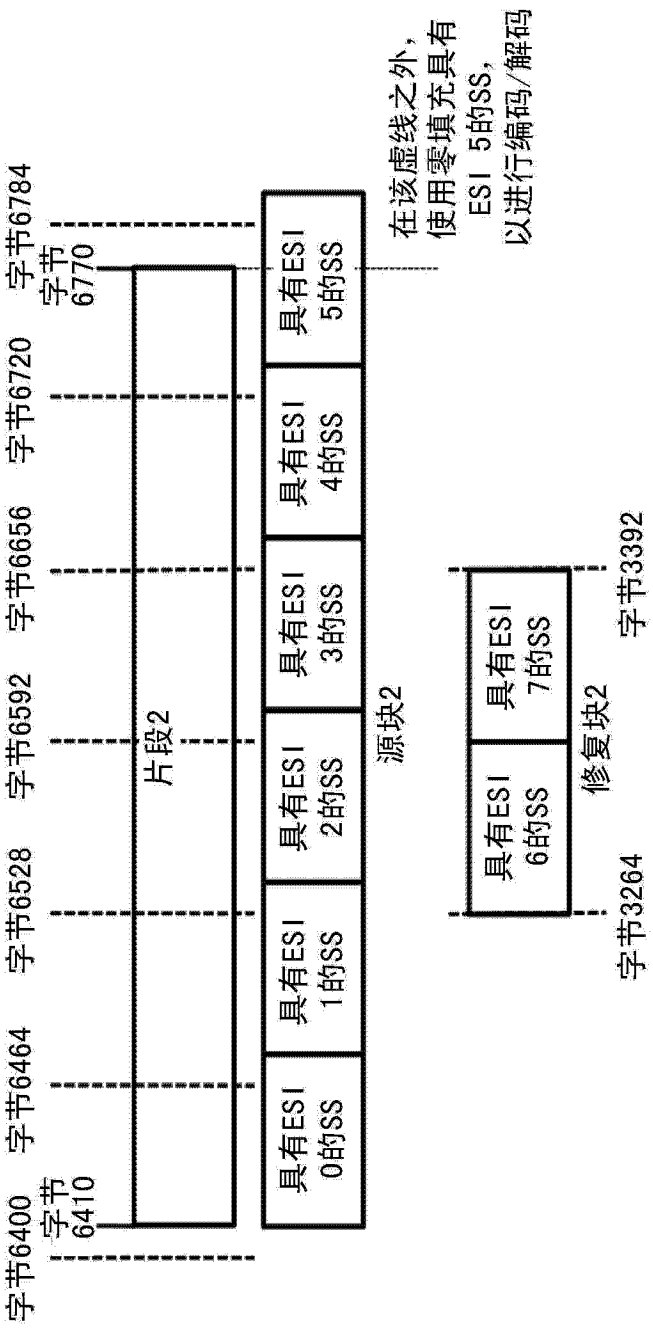

字节6400
字节6410
字节6464
字节6528
字节6592
字节6656
字节6720
字节6770
字节6784
片段2
具有ESI 0的SS
具有ESI 1的SS
具有ESI 2的SS
具有ESI 3的SS
具有ESI 4的SS
具有ESI 5的SS
源块2
在该虚线之外，使用零填充具有ESI 5的SS，以进行编码/解码
具有ESI 6的SS
具有ESI 7的SS
修复块2
字节3264
字节3392

图 35

MLB参数集：

T	lambda	mu
0	0.4*R	0.9

保守参数集：

t	lambda	mu
0	0.2	0.2
1	0.4	0.3
2	0.5	0.4
3	0.5	0.7
4	0.5	0.9

中等参数集：

t	lambda	mu
0	0.4	0.3
1	0.5	0.4
2	0.6	0.8
3	0.7	1.0
4	0.8	1.2

积极参数集：

t	lambda	mu
0	0.6	0.5
1	0.7	0.6
2	0.8	1.0
3	0.9	1.5
4	1.0	2.0

图 36