(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 106134147 A
(43)申请公布日 2016.11.16

(21)申请号 201580013619.9

(22)申请日 2015.03.17

(30)优先权数据
61/954,970 2014.03.18 US
14/289,403 2014.05.28 US

(85)PCT国际申请进入国家阶段日
2016.09.13

(86)PCT国际申请的申请数据
PCT/US2015/021056 2015.03.17

(87)PCT国际申请的公布数据
WO2015/142913 EN 2015.09.24

(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 M·G·卢比 L·C·明德 Y·毛

(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)
H04L 29/08(2006.01)

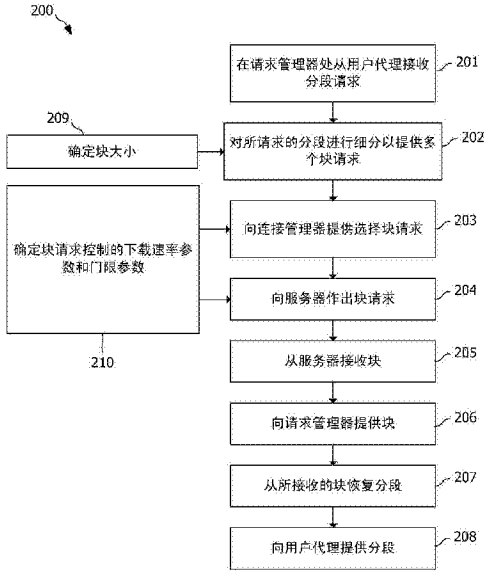
权利要求书6页 说明书41页 附图16页

(54)发明名称

实现请求管理器和连接管理器功能的传输加速器

(57)摘要

根据本公开内容的实施例提供了用于从内容服务器向客户端设备的用户代理(UA)传递内容的传输加速器(TA)系统和方法。TA的实施例操作为:由所述TA的请求管理器(RM)将由UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求;以及,由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块。因此可以由所述CM经由在所述CM与所述内容服务器之间建立的多个连接作出对来自所述内容服务器的所述内容的所述块的请求。



CN 106134147 A

1.一种用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的方法,所述方法包括:

由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率;

由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块;以及

由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块。

2.根据权利要求1所述的方法,还包括:

由所述CM确定所述块请求的所述内容块的所述大小;以及

用信号从所述CM向所述RM发送所确定的所述块请求的所述内容块的大小。

3.根据权利要求1所述的方法,还包括:

将所述块请求的所述内容块的所述大小确定为基于对由所述内容服务器提供的块请求的响应的大小。

4.根据权利要求1所述的方法,还包括:

将所述块请求的所述内容块的所述大小确定为所述内容服务器能够立即通过该连接来发送的数据的量的大小。

5.根据权利要求1所述的方法,还包括:

基于由所述TA向所述内容服务器提供的上游数据的开销来确定所述块请求的所述内容块的所述大小。

6.根据权利要求1所述的方法,还包括:

确定所述块请求的所述内容块的最小大小,其中,大小小于所述最小大小的分段是使用一个块请求来请求的。

7.根据权利要求1所述的方法,还包括:

确定所述块请求的所述内容块的最大大小,其中,分段被划分成大小小于或者等于所述最大大小的块请求。

8.根据权利要求1所述的方法,还包括:

在获得所述分段请求的对应的分段的大小之前确定至少第一块请求。

9.根据权利要求1所述的方法,还包括:

独立于所述分段请求的分段的大小来确定所述块请求的所述内容块的所述大小。

10.根据权利要求1所述的方法,还包括:

将所述块请求的所述内容块的所述大小确定为促进所述内容服务器在所述内容服务器一接收到超文本传输协议(HTTP)块请求时就立即通过所述CM与所述内容服务器之间的传输控制协议(TCP)连接来发送整个HTTP响应的大小。

11.根据权利要求1所述的方法,还包括:

将所述块请求的所述内容块的所述大小确定为适于促进内容的块按照由所述CM作出的针对块的所述请求的次序到达所述TA处的大小。

12.根据权利要求1所述的方法,还包括:

由所述CM动态地调整在所述CM与所述内容服务器之间建立的所述多个连接中的连接的数量。

13.根据权利要求1所述的方法，其中，所述经由多个连接从所述内容服务器请求所述多个块中的所述块包括：经由所述多个连接并行地请求所述多个块。

14.根据权利要求13所述的方法，其中，对所述多个块的连续的块请求的大小被选择为不同，以便降低块请求同时完成的可能性。

15.根据权利要求13所述的方法，其中，所述多个连接中的第一连接上的还未由所述内容服务器完全地进行服务的块请求被至少部分地使用所述多个连接中的一个或多个不同的连接来重新发送。

16.根据权利要求13所述的方法，还包括：

由所述CM控制所述多个连接中的每个连接的接收窗口大小以为所述连接中的每个连接提供近似相同的下载速率。

17.根据权利要求1所述的方法，还包括：

由所述CM计算在所述CM作出另一个块请求之前在所述一个或多个连接上允许的已请求但还未被接收的数据的最大量。

18.根据权利要求17所述的方法，其中，所述计算已请求但还未被接收的数据的所述最大量包括：

利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定当未完成的已请求数据(B)剩余时请求更多数据的门限(Thresh)。

19.根据权利要求1所述的方法，还包括：

由所述CM来计算目标块大小。

20.根据权利要求19所述的方法，其中，所述计算所述目标块大小包括：

利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定将在确定要请求的块的大小时使用的目标块大小(T)。

21.根据权利要求1所述的方法，其中，所述请求所述多个块中的所述块包括：

阻止在所述多个连接中的特定连接上作出针对内容的块的下一个请求，直到该连接上的任何之前的请求已完成为止。

22.根据权利要求1所述的方法，其中，所述请求所述多个块中的所述块包括：

当所述多个连接中的特定连接上的一个或多个之前的请求还未完成时，在该连接上作出针对内容的块的下一个请求。

23.一种被配置为用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的装置，所述装置包括：

用于由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求的单元，其中，所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率；

用于由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块的单元；以及

用于由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块的单元。

24.根据权利要求23所述的装置,还包括:

用于由所述CM确定所述块请求的所述内容块的所述大小的单元;以及

用于用信号从所述CM向所述RM发送所确定的所述块请求的所述内容块的大小的单元。

25.根据权利要求23所述的装置,还包括:

用于将所述块请求的所述内容块的所述大小确定为基于对由所述内容服务器提供的块请求的响应的大小的单元。

26.根据权利要求23所述的装置,还包括:

用于将所述块请求的所述内容块的所述大小确定为所述内容服务器能够立即通过该连接来发送的数据的量的大小的单元。

27.根据权利要求23所述的装置,还包括:

用于基于由所述TA向所述内容服务器提供的上游数据的开销来确定所述块请求的所述内容块的所述大小的单元。

28.根据权利要求23所述的装置,还包括:

用于确定所述块请求的所述内容块的最小大小的单元,其中,大小小于所述最小大小的分段是使用一个块请求来请求的。

29.根据权利要求23所述的装置,还包括:

用于确定所述块请求的所述内容块的最大大小的单元,其中,分段被划分成大小小于或者等于所述最大大小的块请求。

30.根据权利要求23所述的装置,还包括:

用于在获得所述分段请求的对应的分段的大小之前确定至少第一块请求的单元。

31.根据权利要求23所述的装置,还包括:

用于独立于所述分段请求的分段的大小来确定所述块请求的所述内容块的所述大小的单元。

32.根据权利要求23所述的装置,还包括:

用于将所述块请求的所述内容块的所述大小确定为促进所述内容服务器在所述内容服务器一接收到超文本传输协议(HTTP)块请求时就立即通过所述CM与所述内容服务器之间的传输控制协议(TCP)连接来发送整个HTTP响应的大小的单元。

33.根据权利要求23所述的装置,还包括:

用于将所述块请求的所述内容块的所述大小确定为适于促进内容的块按照由所述CM作出的针对块的所述请求的次序到达所述TA处的大小的单元。

34.根据权利要求23所述的装置,还包括:

用于由所述CM动态地调整在所述CM与所述内容服务器之间建立的所述多个连接中的连接的数量的单元。

35.根据权利要求23所述的装置,其中,所述用于经由多个连接从所述内容服务器请求所述多个块中的所述块的单元包括:用于经由所述多个连接并行地请求所述多个块的单元。

36.根据权利要求35所述的装置,其中,对所述多个块的连续的块请求的大小被选择为不同,以便降低块请求同时完成的可能性。

37.根据权利要求35所述的装置,其中,所述多个连接中的第一连接上的还未由所述内

容服务器完全地进行服务的块请求被至少部分地使用所述多个连接中的一个或多个不同的连接来重新发送。

38.根据权利要求35所述的装置,还包括:

用于由所述CM控制所述多个连接中的每个连接的接收窗口大小以为所述连接中的每个连接提供近似相同的下载速率的单元。

39.根据权利要求23所述的装置,还包括:

用于由所述CM计算在所述CM作出另一个块请求之前在所述一个或多个连接上允许的已请求但还未被接收的数据的最大量的单元。

40.根据权利要求39所述的装置,其中,所述用于计算已请求但还未被接收的数据的所述最大量的单元包括:

用于利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定当未完成的已请求数据(B)剩余时请求更多数据的门限(Thresh)的单元。

41.根据权利要求23所述的装置,还包括:

用于由所述CM来计算目标块大小的单元。

42.根据权利要求41所述的装置,其中,所述用于计算所述目标块大小的单元包括:

用于利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定将在确定要请求的块的大小时使用的目标块大小(T)的单元。

43.根据权利要求23所述的装置,其中,所述用于请求所述多个块中的所述块的单元包括:

用于阻止在所述多个连接中的特定连接上作出针对内容的块的下一个请求,直到该连接上的任何之前的请求已完成为止的单元。

44.根据权利要求23所述的装置,其中,所述用于请求所述多个块中的所述块的单元包括:

用于在所述多个连接中的特定连接上的一个或多个之前的请求还未完成时,在该连接上作出针对内容的块的下一个请求的单元。

45.一种用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的计算机程序产品,所述计算机程序产品包括:

非暂时性计算机可读介质,其具有记录在其上的程序代码,所述程序代码包括:

用于由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求的程序代码,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率;

用于由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块的程序代码;以及

用于由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块的程序代码。

46.根据权利要求45所述的计算机程序产品,还包括:

用于由所述CM确定所述块请求的所述内容块的所述大小的程序代码;以及

用于用信号从所述CM向所述RM发送所确定的所述块请求的所述内容块的大小的程序代码。

47.根据权利要求45所述的计算机程序产品，还包括：

用于将所述块请求的所述内容块的所述大小确定为基于对由所述内容服务器提供的块请求的响应的大小的程序代码。

48.根据权利要求45所述的计算机程序产品，还包括：

用于将所述块请求的所述内容块的所述大小确定为所述内容服务器能够立即通过该连接来发送的数据的量的大小的程序代码。

49.根据权利要求45所述的计算机程序产品，还包括：

用于在获得所述分段请求的对应的分段的大小之前确定至少第一块请求的程序代码。

50.根据权利要求45所述的计算机程序产品，还包括：

用于独立于所述分段请求的分段的大小来确定所述块请求的所述内容块的所述大小的程序代码。

51.根据权利要求45所述的计算机程序产品，还包括：

用于将所述块请求的所述内容块的所述大小确定为促进所述内容服务器在所述内容服务器一接收到超文本传输协议(HTTP)块请求时就立即通过所述CM与所述内容服务器之间的传输控制协议(TCP)连接来发送整个HTTP响应的大小的程序代码。

52.根据权利要求45所述的计算机程序产品，还包括：

用于将所述块请求的所述内容块的所述大小确定为适于促进内容的块按照由所述CM作出的针对块的所述请求的次序到达所述TA处的大小的程序代码。

53.根据权利要求45所述的计算机程序产品，还包括：

用于由所述CM动态地调整在所述CM与所述内容服务器之间建立的所述多个连接中的连接的数量的程序代码。

54.根据权利要求45所述的计算机程序产品，其中，所述用于经由多个连接从所述内容服务器请求所述多个块中的所述块的程序代码包括：用于经由所述多个连接并行地请求所述多个块的程序代码，以及其中，对所述多个块的连续的块请求的大小被选择为不同，以便降低块请求同时完成的可能性。

55.根据权利要求45所述的计算机程序产品，还包括：

用于由所述CM计算在所述CM作出另一个块请求之前在所述一个或多个连接上允许的已请求但还未被接收的数据的最大量的程序代码。

56.根据权利要求55所述的计算机程序产品，其中，所述用于计算已请求但还未被接收的数据的所述最大量的程序代码包括：

用于利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定当未完成的已请求数据(B)剩余时的请求更多数据的门限(Thresh)的程序代码。

57.根据权利要求45所述的计算机程序产品，还包括：

用于由所述CM使用下载流水线速率(DPR)度量和下载块速率(DCR)度量来计算目标块大小以确定目标块大小(T)的程序代码。

58.一种被配置为用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的装置，所述装置包括：

至少一个处理器；以及

存储器，其耦合到所述至少一个处理器，

其中,所述至少一个处理器被配置为:

由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率;

由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块;以及

由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块。

59.根据权利要求58所述的装置,其中,所述至少一个处理器还被配置为:

确定所述块请求的所述内容块的所述大小;以及

用信号从所述CM向所述RM发送所确定的所述块请求的所述内容块的大小。

60.根据权利要求58所述的装置,其中,所述至少一个处理器还被配置为:

动态地调整在所述CM与所述内容服务器之间建立的所述多个连接中的连接的数量。

61.根据权利要求58所述的装置,其中,所述至少一个处理器还被配置为:

计算在所述CM作出另一个块请求之前在所述一个或多个连接上允许的已请求但还未被接收的数据的最大量,其中,对已请求但还未被接收的数据的所述最大量的计算利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来确定当未完成的已请求数据(B)剩余时请求更多数据的门限(Thresh)。

62.根据权利要求58所述的装置,其中,所述至少一个处理器还被配置为:

利用下载流水线速率(DPR)度量和下载块速率(DCR)度量来计算目标块大小,以确定将在确定要请求的块的大小时使用的目标块大小(T)。

实现请求管理器和连接管理器功能的传输加速器

[0001] 优先权和相关申请声明

[0002] 本申请要求于2014年3月18日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING REQUEST MANAGER AND CONNECTION MANAGER FUNCTIONALITY”、编号为61/954,970的共同未决的美国临时专利申请和于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING REQUEST MANAGER AND CONNECTION MANAGER FUNCTIONALITY”、序列号为14/289,403的美国专利申请的优先权;故以引用方式将所述美国临时专利申请和美国专利申请的公开内容并入本文。本申请涉及共同让与的于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING EXTENDED TRANSMISSION CONTROL FUNCTIONALITY”、序列号为14/289,016的美国专利申请;于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING EXTENDED TRANSMISSION CONTROL FUNCTIONALITY”、序列号为14/289,181的申请;于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING ENHANCED SIGNALING”、序列号为14/289,348的申请;于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING SELECTIVE UTILIZATION OF REDUNDANT ENCODED CONTENT DATA FUNCTIONALITY”、序列号为14/289,458的申请;于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING A MULTIPLE INTERFACE ARCHITECTURE”、序列号14/289,476的申请;以及于2014年5月28日递交的、名称为“TRANSPORT ACCELERATOR IMPLEMENTING CLIENT SIDE TRANSMISSION FUNCTIONALITY”、序列号14/289,499的申请;所述申请中的每个申请被与本文一起并发地递交,并且以引用方式将所述申请的全部公开内容明确地并入本文。

背景技术

[0003] 越来越多的内容正通过可用的通信网络被传输。通常,该内容包括许多类型的数据,所述数据包括例如音频数据、视频数据、图像数据等。视频内容,特别是高分辨率视频内容,通常包括相对大的数据文件或者数据的其它集合。相应地,消费这样的内容的端用户设备或者其它客户端设备上的用户代理(UA)通常请求并且接收包括期望的视频内容的内容的分段的序列。例如,UA可以包括在用户设备上执行的请求数据(通常是多媒体数据)并且接收所请求的数据以用于进行进一步处理以及可能用于显示在用户设备上的客户端应用或者过程。

[0004] 许多类型的应用目前依赖于HTTP来进行前述的内容传递。在许多这样的应用中,HTTP传输的性能对于伴随应用的用户体验是至关重要的。例如,直播流传送具有几个可能妨碍视频流传送客户端的性能的限制。两个限制特别突出。第一,媒体片段随时间逐个地变得可用。该限制防止客户端连续地下载大量数据,这继而影响下载速率估计的准确度。由于多数流传送客户端根据“请求-下载-估计”的循环来操作,所以其当下载估计不准确时通常运转得不好。第二,当观看直播事件流传送时,用户一般不想遭受与实际直播事件时间轴的长延迟。这样的行为防止流传送客户端建立大缓冲器,这继而可能导致更多重新缓冲。

发明内容

[0005] 根据本公开内容的实施例提供了用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的方法。实施例的方法包括:由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块(chunk)的多个块请求,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率。实施例的方法还包括:由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以便请求所述内容的块;以及,由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块。

[0006] 根据本公开内容的实施例提供了被配置为用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的装置。实施例的装置包括:用于由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求的单元,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率。实施例的装置还包括:用于由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块的单元;以及,用于由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块的单元。

[0007] 根据本公开内容的实施例提供了用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的计算机程序产品。实施例的计算机程序产品包括具有记录在其上的程序代码的非暂时性计算机可读介质。实施例的程序代码包括:用于由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求的程序代码,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率。实施例的程序代码还包括:用于由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块的程序代码;以及,用于由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块的程序代码。

[0008] 根据本公开内容的实施例提供了被配置为用于由客户端设备的传输加速器(TA)加速从内容服务器向所述客户端设备的用户代理(UA)传递内容的装置。实施例的装置包括至少一个处理器和耦合到所述至少一个处理器的存储器。实施例的所述至少一个处理器被配置为:由所述TA的请求管理器(RM)将由所述UA提供的分段请求各自细分成用于请求所述内容的块的多个块请求,其中,所述块请求的内容块的大小被确定为独立于所述内容服务器的网络拥塞避免操作来提高网络传输速率。实施例的所述至少一个处理器还被配置为:由所述RM向所述TA的连接管理器(CM)提供所述多个块请求中的块请求以用于请求所述内容的块;以及,由所述CM经由在所述CM与所述内容服务器之间建立的多个连接从所述内容服务器请求所述内容的所述块。

附图说明

[0009] 图1A示出了根据本公开内容的实施例的适于传输加速操作的系统。

[0010] 图1B示出了关于如可以关于根据本公开内容的实施例的传输加速器的配置来实现的请求管理器和连接管理器的实施例的细节。

[0011] 图1C示出了关于根据本公开内容的实施例的传输加速器配置的细节。

[0012] 图2示出了说明根据本公开内容的实施例的提供请求管理器和连接管理器功能的传输加速器的操作的流程图。

[0013] 图3示出了使用门限参数来确定何时可以在连接上请求另一个数据块的连接管理器的示例性实施例。

[0014] 图4示出了使用门限参数来确定何时连接管理器当前能够立即作出块请求的连接管理器的示例性实施例。

[0015] 图5示出了根据本公开内容的实施例的利用关于多个内容服务器来使用的多个连接来进行对于客户端设备的内容传输。

[0016] 图6示出了根据本公开内容的实施例的在其中将请求管理器与多个连接管理器接合的配置。

[0017] 图7示出了根据本公开内容的实施例的下载块速率减损。

[0018] 图8示出了根据本公开内容的实施例的将块请求分类到纪元中。

[0019] 图9示出了根据本公开内容的实施例的下载流水线速率减损。

[0020] 图10A至图10C示出了根据本公开内容的各种T值选择的结果。

[0021] 图11示出了根据本公开内容的实施例的各种下载块速率和下载流水线速率组合的平均间隙。

[0022] 图12示出了用于根据本公开内容的实施例的重新排序层的逻辑单元。

[0023] 图13A至图13C示出了根据本公开内容的实施例的请求管理器与连接管理器之间的算法执行的高级调用流。

[0024] 图14A至图14C示出了针对使用根据本公开内容的实施例的传输加速器连接到单个起源服务器的单个用户代理所生成的图表。

具体实施方式

[0025] 术语“示例性”在本文中用于表示“充当示例、实例或者说明”。任何在本文中被描述为“示例性”的方面不必理解为是比其它方面优选或者有优势的。

[0026] 在本描述内容中，术语“应用”可以还包括具有可执行内容的文件，可执行内容诸如是：目标代码、脚本、字节码、标记语言文件和补丁。另外，在本文中引用的“应用”可以还包括本质上不是可执行的的文件，所述文件诸如是可能需要被打开的文档或者需要被访问的其它数据文件。

[0027] 如在本描述内容中使用的，术语“内容”可以包括具有视频、音频、视频和音频的组合的数据、或者处于一个或多个质量水平的其它数据，所述质量水平通过比特率、分辨率或者其它因素被确定。内容可以还包括可执行内容，可执行内容诸如是：目标代码、脚本、字节码、标记语言文件和补丁。另外，“内容”可以还包括本质上不可执行的文件，所述文件诸如是可能需要被打开的文档或者需要被访问的其它数据文件。

[0028] 如本描述内容中使用的，术语“分段”指可以被用户设备请求和/或在用户设备处被接收的内容的一个或多个部分。

[0029] 如本描述内容中使用的，术语“流传送内容”指可以根据实现内容的实时传输或者在一段时间中的内容的传输的一个或多个标准来从服务器设备被发送并且在用户设备处被接收的内容。流传送内容标准的示例包括那些支持经解交织的信道(或者多个信道)的流传送内容标准和那些不支持经解交织的信道(或者多个信道)的流传送内容标准。

[0030] 如本描述内容中使用的，术语“部件”、“数据库”、“模块”、“系统”等旨在指计算机相关的实体，计算机相关的实体是硬件、固件、硬件和软件的组合、软件、或者执行中的软件。例如，部件可以但不限于是运行在处理器上的过程、处理器、对象、可执行文件、执行的线程、程序和/或计算机。作为说明，运行在计算设备上的应用和计算设备两者都可以是部件。一个或多个部件可以存在在过程和/或执行的线程内，并且部件可以位于一个计算机上和/或被分布到两个或多个计算机之中。另外，这些部件可以从具有存储在其上的各种数据结构的各种计算机可读介质执行。部件可以诸如根据具有一个或多个数据分组的信号通过本地和/或远程过程进行通信(例如，来自一个部件的数据通过信号在本地系统中、分布式系统中和/或与其它系统跨诸如是互联网的网络地与另一个部件交互)。

[0031] 如本文中使用的，术语“用户装备”、“用户设备”和“客户端设备”包括能够从web服务器请求和接收内容以及向web服务器发送信息的设备。这样的设备可以是固定设备或者移动设备。可以可互换地使用术语“用户装备”、“用户设备”和“客户端设备”。

[0032] 如本文中使用的，术语“用户”指在用户设备上或者客户端设备上接收内容并且向网站发送信息的个人。

[0033] 图1A示出了根据本文中的概念适合于通过通信网络来提供对诸如可以包括音频数据、视频数据、图像数据、文件数据等的内容的传输的系统100。相应地，客户端设备110被示为经由网络150与服务器130相通信，由此服务器130可以根据本公开内容的概念向客户端设备110传输存储在数据库140中的各种内容。应当认识到，尽管仅单个客户端设备和单个服务器和数据库在图1A中被表示，但系统100可以包括多个的任意或全部这样的设备。例如，服务器130可以包括服务器群的服务器，其中，多个服务器可以被集中地和/或以分布式配置来放置，以为内容传输的高水平需求提供服务。替代地，诸如当内容中的一些或全部内容存在于也被并置在该设备上的数据库140(高速缓存)中并且通过服务器130被提供给传输加速器120时，服务器130可以被并置在与传输加速器120相同的设备上(例如，取代通过网络150，直接通过I/O单元113连接到传输加速器120)。同样地，用户可以拥有多个客户端设备，和/或多个用户可以各自拥有一个或多个客户端设备，所述客户端设备中的任何或全部客户端设备适合于根据本文中的概念的内容传输。

[0034] 客户端设备110可以包括对可操作为经由网络150接收内容的传输的设备的各种配置。例如，客户端设备110可以包括有线设备、无线设备、个人计算设备、平板或者垫板型计算设备、便携式蜂窝电话、启用WiFi的设备、启用蓝牙的设备、电视、具有显示器的一副眼镜、一副增强现实眼镜或者任何其它可以使用任何可用的方法或者基础设施与服务器130通信的被连接到网络150的通信、计算或者接口设备。客户端设备110被称为“客户端设备”，因为它可以运行为或者被连接到运行为服务器130的客户端的设备。

[0035] 所示出的实施例的客户端设备110包括多个功能框，所述多个功能框在这里被示为包括处理器111、存储器112和输入/输出(I/O)单元113。尽管为简单起见未在图1A中的表示中被示出，但客户端设备110可以包括额外的功能框，例如，用户接口、射频(RF)模块、照

相机、传感器阵列、显示器、视频播放器、浏览器等，所述各项中的一些或全部项可以被根据本文中的概念的操作利用。前述功能框可以通过一个或多个总线(例如，总线114)被操作地连接。总线114可以包括用于允许所连接的单元、模块和部件通信和互操作的逻辑和物理连接。

[0036] 存储器112可以是任何类型的易失性或者非易失性存储器，并且在一个实施例中可以包括闪速存储器。存储器112可以被永久地安装在客户端设备110中，或者可以是可移除存储器单元(例如，可移除存储器卡)。尽管被示为单个单元，但存储器112可以包括多个分立的存储器和/或存储器类型。

[0037] 存储器112可以存储或者包括诸如可以形成应用、操作系统、文件、电子文档、内容等的各种计算机可读代码段。例如，所示出的实施例的存储器112包括定义传输加速器(TA)120和UA 129的计算机可读代码段，所述计算机可读代码段当由处理器(例如，处理器111)执行时提供可以如本文中描述的那样操作的逻辑电路。由存储器112存储的代码段可以提供除前述的TA 120和UA 129之外的应用。例如，存储器112可以存储在根据本文中的实施例访问来自服务器130的内容时有用的应用(例如，浏览器)。如果服务器130是web服务器的话，这样的浏览器可以是web浏览器，例如，用于通过连接151和152经由网络150访问和查看web内容和经由HTTP与服务器130通信的超文本传输协议(HTTP)web浏览器。作为一个示例，HTTP请求可以通过连接151和152经由网络150从客户端设备110中的浏览器被发送给服务器130。HTTP响应可以通过连接152和151经由网络150从服务器130被发送给客户端设备110中的浏览器。

[0038] UA 129可操作为从诸如是服务器130的服务器请求和/或接收内容。UA 129可以例如包括请求数据(例如，多媒体数据)并且接收所请求的数据以便进行进一步处理以及可能便于显示在客户端设备110的显示器上的客户端应用或者过程(例如，浏览器、DASH客户端、HTTP直播流传送(HLS)客户端等)。例如，客户端设备110可以执行包括用于回放媒体的UA 129的代码，所述代码诸如是独立的媒体回放应用或者被配置为运行在互联网浏览器中的基于浏览器的媒体播放器。在根据实施例的操作中，UA 129决定在流传送内容会话期间的各种时间点处请求内容文件的分段中的哪些分段或者序列以便传输。例如，UA 129的DASH客户端配置可以操作为，诸如基于最近的下载状况来决定在每个时间点处从内容的哪个表示(例如，高分辨率表示、中等分辨率表示、低分辨率表示等)请求哪个分段。同样地，UA 129的web浏览器配置可以操作为作出对web页面或者其部分等的请求。通常，UA使用HTTP请求来请求这样的分段。

[0039] TA 120根据本文中的概念适合于提供对期望的内容的分段中的分段或者序列的增强传递(例如，可以在提供视频流传送、文件下载、基于web的应用、一般web页面等时可以被使用的前述的内容分段)。实施例的TA120适于允许仅支持标准接口(例如，实现标准化TCP传输协议的HTTP 1.1接口)的通用或者传统UA(即，还未被预先设计为与TA交互的UA)作出分段请求，以仍然从使用TA执行那些请求中获益。额外地或者替代地，实施例的TA 120提供增强型接口，增强型接口用于促进向被设计为利用增强型接口的功能的UA提供进一步的益处。实施例的TA 120适于根据现有内容传输协议(诸如是通过实现标准化TCP传输协议的HTTP接口使用TCP)执行分段请求，因此允许通用或者传统媒体服务器(即，还未被预先设计为与TA交互的媒体服务器)为所述请求提供服务，同时向UA和客户端设备提供对分段的增

强型传递。

[0040] 在提供前述的增强型分段传递功能时，本文中的实施例的TA 120包括如本文中描述的架构部件和协议。例如，图1A中所示出的实施例的TA 120包括请求管理器(RM)121和连接管理器(CM)122，其协作以提供如在下面进一步描述的各种增强型分段传递功能。

[0041] 除前述的形成应用、操作系统、文件、电子文档、内容等的代码段之外，存储器112可以包括或者以其它方式提供被客户端设备110的功能框使用的各种寄存器、缓冲器和存储器单元。例如，存储器112可以包括播放缓冲器，诸如，可以提供用于假脱机输入(spool)来自服务器130的流传送的分段的数据的先入/先出(FIFO)存储器，并且被客户端设备110回放。

[0042] 实施例的处理器111可以是任何能够执行指令以控制客户端设备110的操作和功能的通用或者专用处理器。尽管被示为是单个单元，但处理器111可以包括多个处理器或者分布式处理架构。

[0043] I/O单元113可以包括和/或耦合到各种输入/输出部件。例如，I/O单元113可以包括和/或耦合到显示器、扬声器、麦克风、键区、指点设备、触摸敏感屏幕、用户接口控制单元和任何其它允许用户提供输入命令和从客户端设备110接收输出的设备或者系统。任何或者全部这样的部件可以被用于提供客户端设备110的用户接口。额外地或者替代地，I/O单元113可以包括和/或耦合到磁盘控制器、网络接口卡(NIC)、射频(RF)收发机和任何其它促进客户端设备110的输入和/或输出功能的设备或者系统。

[0044] 在用于访问和播放流传送内容的操作中，客户端设备110使用链路151和152经由网络150与服务器130通信以获得内容数据(例如，如前述的分段)，所述内容数据当被渲染时提供对内容的回放。相应地，UA 129可以包括由处理器111执行的用于在客户端设备110中建立内容回放环境的内容播放器应用。当发起对特定内容文件的回放时，UA 129可以与服务器130的内容传递平台通信以获得内容标识符(例如，一个或多个列表、清单、配置文件、或者其它标识期望的内容的一个或多个媒体段或分段以及它们的时序边界的标识符)。关于媒体段以及它们的时序的信息被UA 129的流传送内容逻辑单元用于控制请求用于回放内容的分段。

[0045] 服务器130包括一个或多个可操作为向客户端设备提供期望的内容的系统。例如，服务器130可以包括可操作为经由网络150向各种客户端设备流传送内容的标准HTTP web服务器。服务器130可以包括内容传递平台，内容传递平台包括任何可以向用户设备110传递内容的系统或者方法。内容可以被存储在与服务器130相通信的一个或多个数据库中，例如，所示出的实施例的数据库140。数据库140可以被存储在服务器130上，或者可以被存储在通信地耦合到服务器130的一个或多个服务器上。数据库140的内容可以包括各种形式的数据，数据诸如是视频、音频、流传送文本以及任何其它可以被服务器130在一段时间内传输到客户端设备110的内容，例如，直播网络广播内容和所存储的媒体内容。

[0046] 数据库140可以包括多个不同的源或者内容文件和/或任何特定内容的多个不同的表示(例如，高分辨率表示、中等分辨率表示、低分辨率表示等)。例如，内容文件141可以包括特定多媒体汇编的高分辨率表示，并且因此包括当被传输时的高比特率表示，而内容文件142可以包括相同的特定多媒体汇编的低分辨率表示，并且因此包括当被传输时的低比特率表示。额外地或者替代地，任何特定内容的不同表示可以包括诸如可以由内容文件

143提供的前向纠错(FEC)表示(例如,包括对内容数据的冗余编码的表示)。统一资源定位符(URL)、统一资源标识符(URI)和/或统一资源名称(URN)是与这些根据本文中的实施例的内容文件中的全部内容文件相关联的,并且因此这样的URL、URI和/或URN可以可能与诸如是字节范围的其它信息一起被利用,以标识和访问所请求的数据。

[0047] 网络150可以是无线网络、有线网络、广域网(WAN)、局域网(LAN)或者任何其它适于传输如本文中描述的内容的网络。在一个实施例中,网络150可以包括互联网的至少部分。客户端设备110可以通过诸如由网络连接151表示的双向连接被连接到网络150。替代地,客户端设备110可以经由诸如由启用了多媒体广播多媒体系统(MBMS)的网络提供的单向连接(例如是连接151、152,并且网络150可以包括MBMS网络,并且服务器130可以包括广播组播服务中心(BM-SC)服务器)被连接。连接可以是有线连接,或者可以是无线连接。在一个实施例中,连接151可以无线连接,例如,蜂窝4G连接、无线保真(WiFi)连接、蓝牙连接或者另一种无线连接。服务器130可以通过诸如由网络连接152表示的双向连接被连接到网络150。服务器130可以通过单向连接被连接到网络150(例如,使用在3GPP TS.26.346中描述的协议和服务的MBMS网络或者ATSC 3.0网络)。连接可以是有线连接,或者可以是无线连接。

[0048] 图1A中所示出的实施例的客户端设备110包括可操作为根据本文中的概念提供对期望的内容的分段中的分段或者序列的增强型传递的TA 120。如上面讨论的,所示出的实施例的TA 120包括协作以提供各种增强型分段传递功能的RM 121和CM 122。实施例的UA 129与RM 121之间的接口124和RM 121与CM 122之间的接口123提供类HTTP连接。例如,前述接口可以使用标准HTTP协议以及包括额外的信令(例如,使用与HTTP的信令技术相似的信令技术所提供的),以支持根据本文中的实施例的增强型分段传递的特定功能方面。

[0049] 图1B示出了关于如可以关于如图1A中示出的TA 120的配置来实现的RM 121和CM 122的实施例的细节。特别地说,RM 121被示为包括请求队列(RQ)191a-191c、请求调度器192(包括请求分块算法193)和重新排序层194。CM 122被示为包括T值管理器195、准备就绪计算器196和请求接收机/监控器197。应当认识到,尽管关于图1B中示出的RM 121和CM 122的实施例示出了特定功能框,但可以实现用于执行根据如本文中描述的实施例的功能的额外的或者替代的功能框。

[0050] 在图1B中示出的RM 121的实施例中提供了RQ 191a-191c,以由一个或多个UA(例如,UA 129)提供对由TA 120接收的请求的排队。所示出的实施例中示出的多个RQ中的不同RQ可以被用于提供关于各种请求的排队。例如,RQ中的不同的RQ可以各自与请求优先级的不同级别相关联(例如,直播流传送媒体请求可以接收最高优先级,而流传送媒体接收较低的优先级,并且web页面内容接收更低的优先级)。相似地,RQ中的不同的RQ可以各自与不同的UA、不同类型的UA等相关联。应当认识到,尽管在所示出的实施例中表示了三个这样的队列,但本文中的实施例可以包括任意数量的这样的RQ。

[0051] 实施例的请求调度器192实现一个或多个根据本文中的概念的用于调度分段请求和/或块请求的调度算法。例如,请求调度器192的逻辑单元可以操作为,基于何时当前被RM请求的分段的被接收的或者已请求但还未被接收的数据的量下降到某个门限量以下、何时RM不具有RM可以针对其作出另一个块请求的已被接收的分段请求等,确定RM是否对于来自UA的分段请求准备就绪。额外地或者替代地,请求调度器310的逻辑单元可以操作为,确定

是否将作出块请求，以提供近似是给定当前网络状况下可能的最大下载速率的连接的聚合下载速率，以使得在网络中的被缓存的数据的量尽可能少等。请求调度器310可以例如操作为，诸如每当RM从UA接收新的数据下载请求时、每当RM成功地向CM发出用于检查用于发出对于相同或者不同的起源服务器的更多请求的继续的准备就绪性的块请求时、每当针对已被发出的块请求的数据下载被完成时等时候，向CM查询块请求准备就绪性。

[0052] 在根据实施例的操作中，请求调度器192可以使用信用系统(例如，用于维护与RQ 191a-191c中的每个请求队列相关联的逻辑信用)以用于选择从其发出下一个块请求的下一个RQ。伴随每个被发出的新块请求，请求调度器可以向与每个RQ相关联的现有信用添加特定量的信用，以及向与新块请求从其被生成的RQ相关联的现有信用减除特定量的信用。额外地或者替代地，实施例的请求调度器192可以在选择下一个块请求将从其被发出的下一个RQ时考虑每个RQ的优先级。同样地，请求调度器192可以监控重新排序缓冲器的大小(例如，从对与RQ 191a-191c中的每个请求队列相关联的块请求的部分或者完全响应的被缓存的数据的量)，由此请求调度器可以操作为，当重新排序缓冲器中与给定RQ相关联的被缓存的数据的量超过特定门限时，减缓或者停止从该RQ发出块请求。实施例的请求调度器192可以操作为，当被缓存的数据的量下降到门限以下时，恢复从该RQ的正常请求发出。

[0053] 所示出的实施例的请求调度器192被示为包括采用请求分块算法193的形式的分段请求分块功能。实施例的请求分块算法193提供被用于对所请求的分段进行细分以提供多个对应的较小数据请求的逻辑单元。上面引用的名称为“TRANSPORT ACCELERATOR IMPLEMENTING REQUEST MANAGER AND CONNECTION MANAGER FUNCTIONALITY”的专利申请提供如可以被请求分块算法193实现的根据实施例的关于计算合适的块大小(chunk size)的额外的细节。

[0054] 实施例的重新排序层194提供用于从响应于前述块请求而被提供的块来重构所请求的分段的逻辑单元。应当认识到，响应于块请求而被提供的数据的块可以被TA 120无序地接收，并且因此重新排序层194的逻辑单元可以操作为对数据进行重新排序，可能作出对于缺失的数据的请求，以因此提供所请求的数据分段，以便提供给请求方UA。

[0055] CM 122的所示出的实施例的T值管理器195提供用于确定和/或管理用于提供关于块请求的控制(例如，确定何时将作出块请求)的一个或多个参数(例如，门限参数等)的逻辑。相似地，CM 122的所示出的实施例的准备就绪计算器196提供用于确定和/或管理用于提供关于块请求的控制(例如，在CM 122与RM 121之间信号发送对于下一个块请求的准备就绪性)的一个或多个参数(例如，下载速率参数)的逻辑单元。在上面的名称为“TRANSPORT ACCELERATOR IMPLEMENTING REQUEST MANAGER AND CONNECTION MANAGER FUNCTIONALITY”的引用专利申请中提供了根据实施例的关于计算这样的参数以及它们的使用的细节。

[0056] 实施例的请求接收机/监控器197提供可操作为管理块请求的逻辑单元。例如，请求接收机/监控器197可以操作为从RM 121接收块请求、监控对一个或多个内容服务器所作出的块请求的状态、以及接收响应于块请求而被提供的数据块。

[0057] 在根据实施例的操作中，如由图2的流200示出的，RM 121从UA 129接收针对分段的请求(框201)。根据本文中的实施例，RM 121适于从通用或者传统UA(即，还未被预先设计为与RM交互的UA)接收分段请求并对其作出响应，因此提供与这样的传统UA的兼容性。相应

地，RM 121可以操作为将UA 129与TA 120的扩展传输协议操作隔离。然而，如从下面的讨论中将被更充分地理解的，UA 129可以是适于扩展传输协议操作的，由此RM 121和UA 129(例如，通过使用用于实现这样的特征的RM 121与UA 129之间的信令)协作以实现扩展传输协议操作的一个或多个特征。

[0058] 在根据实施例的操作中，请求接收机/监控器197可以收集关于与每个块请求相关联的下载时间和/或下载速率的统计，并且使用这样的统计来确定正被下载的块请求是否太慢。当块请求下载被认为慢(例如，慢于特定门限)时，请求接收机/监控器197可以发出对于慢进展块请求的未被接收的部分的一个或多个块请求。另外，请求接收机/监控器197可以取消慢进展块请求，并且可以可选地关闭被用于慢进展块请求的底层传输连接(例如，使用传输控制协议或者扩展传输协议)，并且打开新的连接。

[0059] 图1C示出了关于实施例的传输加速器配置的进一步的细节。图1C的示例性实施例包括适于促进关于客户端设备110的传输加速器代理操作的TA 120的配置。图1C中示出的TA 120的实施例包括如本文中描述的可操作为生成块请求和管理向一个或多个服务器作出的对于期望的内容的请求的RM 121和CM 122。此外，所示出的实施例的TA 120包括根据本文中的概念的促进代表一个或多个UA所代理的传输加速器操作的额外的功能。例如，TA 120被示为包括提供关于UA 129a-129c的代理服务器接口的代理服务器1021。所示出的实施例的TA 120还被示为包括提供关于UA 129d的web服务器接口的浏览器适配器1022，其中，UA 129d被示为浏览器型用户代理(例如，用于访问和查看web内容以及经由HTTP与web服务器通信的HTTP web浏览器)。除提供关于UA的代理接口的前述功能框之外，图1C中示出的TA 120的实施例被示为包括根据本文中的概念的在促进内容的加速传输时有用的额外的功能框。特别地说，TA 120被示为包括栈处理1023、TA请求分派器1024、栈处理1025和套接字层1026。在上面引用的名称为“TRANSPORT ACCELERATOR IMPLEMENTING A MULTIPLE INTERFACE ARCHITECTURE”的专利申请中提供关于TA代理配置的实施例的进一步的细节。

[0060] 实施例的TA 120实现请求通常小于由UA所请求的内容分段的内容的部分的数据传输。相应地，实施例的RM 121操作为对所请求的分段进行细分(框202)，以提供多个对应的较小数据请求(在本文中被称为在其中所请求的数据包括“块”的“块请求”)。由实施例的TA 120请求的块的大小通常小于由UA 129请求的分段的大小。因此，来自UA 129的每个分段请求可以触发RM 121生成并向CM 122作出多个块请求，以恢复该分段。这样的块请求可以包括数据对象(包括分段内容或者其某个部分)的某种形式的内容标识符(例如，URL、URI、URN等)，可能具有诸如是包括期望的内容块的字节范围的其它信息，由此块进行聚合以提供所请求的分段。

[0061] 由TA 120请求的块的大小可以是基于多个考虑中的任何考虑的。通常，CM 122可以确定目标块大小或者与块大小选择相关的信息，并且将这些信息提供给RM 121，并且RM 121使用所提供的目标块大小或者与块大小选择相关的信息来确定分段被划分成的实际块大小。CM 122当CM 122用信号发送它对于另一个块请求准备就绪时向RM 121提供目标块大小或者与块大小选择相关的信息。相应地，所示出的流200的实施例提供了RM 121在框209处诸如基于由CM 122向RM 121提供的目标块大小或者与块大小选择相关的信息确定块大小。实施例的CM 122可以实现用于确定将被请求的块的目标大小或者确定与块大小选择相关的信息的一个或多个算法。在根据这样的实施例的操作中，CM 122可以用信号向RM 121

发送适于CM正在提供的传输的块请求的目标大小，并且RM可以使用所提供的目标大小来确定当它基于从UA接收的分段请求形成将向CM作出的请求时的实际块大小。

[0062] 作为用于确定块大小的操作的一个示例，对于可以通过其作出下一个块请求的给定连接，CM可以具有服务器130可以通过该连接立即发送的数据的量的精确或者近似的大小W(应当认识到，块大小在该示例中是独立于分段大小以及甚至可能由RM所使用的分段组装算法的)。CM 122可以例如确定诸如是在W和Cmax中的最小值处的目标块请求大小的下一个块请求的合适目标块大小，其中，Cmax是期望的块大小请求的上界(例如，Cmax可以是块请求的内容块的预定的最大大小，其中，分段被划分成大小小于或者等于最大大小的块请求)。以这种方式设置期望的目标块请求大小的原因在于，服务器一通过那个连接接收该大小的请求，服务器就可以立即通过那个连接发送整个响应。在当向服务器作出请求时以这种方式选择全部块请求大小的情况下，并且如果从服务器向CM发送的全部分组被接收并且被有序地接收，则甚至当CM正在通过多个连接请求数据时，由CM请求的全部数据也将按照当CM作出请求时的次序到达。

[0063] 然而应当认识到，针对任何特定情形的对块大小的选择可以取决于系统的各种方面，尽管块大小可以仍然被设置为使得将作出请求的上行链路业务的比率与响应于所述请求被提供的下行链路业务的量成可接受的比例。当CM可以对请求进行流水线操作(例如，在单个TCP连接上对几个HTTP请求进行流水线操作)时，则相对小的块大小可能是令人满意的。在这样的情况下，CM 122的逻辑单元可以操作为选择相对小的目标块大小，其中，目标块大小仍然是大得足以使得每块的请求和响应报头开销不变得过高的。当不支持或者不允许流水线时，或者当支持或者允许仅有限量的流水线时，CM 122的逻辑单元可以操作为选择较大的目标块大小，因为否则它可能不可能利用全部链路容量(例如，目标块大小在此情况下可以被选择为使得所使用的连接的数量与块大小的乘积超过带宽-延迟乘积(BDP))。

[0064] 如根据实施例被实现的块大小可以被理想地选择为成比例于或者至少小于在传输数据块时所使用的连接(例如，TCP连接)的拥塞窗口。然而，实施例的TA 120被实现在接收机侧(例如，客户端设备110)，并且因此一般不具有拥塞窗口信息。TA 120的逻辑单元仍然可以操作为确定关于任何特定连接的拥塞窗口的近似。例如，实施例的TA 120可以操作为模拟接收机处的TCP发送者行为，以便估计由服务器130所体验的连接的拥塞窗口。

[0065] 下面提供了如可以由本文中的实施例的CM 122和RM 121的逻辑单元实现的用于计算块大小的示例性技术。CM 122可以根据该技术向RM 121提供下面的与块大小选择相关的信息：

[0066]

参数	意义
Cmin	建立块请求的内容块的最小大小的目标最小块大小（例如，8 KB），其中，使用一个块请求来请求大小小于最小大小的分段。
Cmax	建立块请求的内容块的最大大小的目标最大块大小（例如，32 KB 或者 64 KB），其中，分段被划分成大小小于或者等于最大大小的块请求。
N	每分段的块的目标最小数量（可以例如被设置为例如是 8 的连接的目标数量）。

[0067] 基于该与块大小选择相关的信息，RM 121可以确定将把大小F的分段划分成的块的数量P，由此分段被划分成P个大小尽可能相等的块(例如，使用IETF RFC 5053或者IETF RFC 6330中指定的划分函数)，可以然后被计算。例如，RM 121可以如下面所示的那样使用前述参数计算块大小：

[0068] Pmax＝max{1,floor(F/Cmin)}

[0069] Pmin＝max{1,floor(F/Cmax)}

[0070] P＝max{Pmin,min{Pmax,N}}(等价地，P＝min{Pmax,max{Pmin,N}})

[0071] 前述的用于确定块大小的技术确保当F被划分成P个大小相等的块时最小块大小至少是min{F,Cmin}。使用这样的技术计算的块大小特别良好地适于在其中CM可以在其最大程度上使用流水线(例如，HTTP流水线)的情形中使用。

[0072] 当CM不能使用流水线时，可以根据实施例利用计算块大小的不同方法。例如，当例如CM 122用信号向RM 121发送该CM对于另一个块请求准备就绪时，CM 122可以动态地确定目标块大小T，并且向RM 121提供T。每个时间点处的T的值可以是基于多个因素的，包括当前网络状况、当前下载速度和如在本文中的其它地方描述的其它度量。

[0073] RM 121可以使用从实施例的CM 122接收的T的值来如下面那样计算下一个请求的块大小，其中，N是RM 121将从其生成下一个块请求的分段中的未被请求的字节的数量：

[0074] ChunkSize:＝min(N,T)

[0075] 在根据实施例的操作中，使全部块大小相同可能不是可取的。例如，实施例可以当初始向服务器发送请求时使用较小的块大小，并且随时间增大块请求的大小(例如，连接的拥塞窗口通常随时间增大，并且因此块大小可以根据该预期被增大)。通过初始使用较小块大小，TA 120可以在流传送会话的开始处体验关于数据传输的期望的响应性，并且此后通过增大块大小来获得上行链路和下行链路带宽利用之间的较好的折中。

[0076] 使用全都大小相同的块可能同样是不可取的，因为使用这样的同质块大小可能导

致TA与服务器之间的全部连接同时空闲，因此导致低效的信道使用。此外，在其中非常小的块被请求(例如，由于N小)的情形中，前一个块请求较大可能是可取的。下面的技术适于使用下面的用于由RM 122来计算块大小的参数来解决前述考虑：

[0077]

参数	意义
I	计数器，每次作出新请求时被重新使用，以I＝0开始。
X	固定的滞后参数，例如，X=1.5。
A	固定的展宽因子，例如，A＝0.5
N	当前分段中的未被请求的字节的数量

[0078] 块大小可以被RM 122如下面所示那样使用前述参数来计算：

[0079] $P=T*(1+A*revbits(I)/2^{32})$

[0080] 如果P*X>N,设置ChunkSize:＝P

[0081] 否则,设置ChunkSize:＝N

[0082] $I=(I+1) \bmod 2^{32}$

[0083] 在上面的计算中，revbits(I)是当I被表示为32比特值时所生成的值，I的32比特以逆序被列出，并且然后经逆转的32比特被看作整数值。例如，如果I＝7，则被表示为32比特值的I是00000000000000000000000000000111，并且因此revbits(I)是整数值11100000000000000000000000000000，该整数值是3,758,096,384，并且因此$revbits(7)/2^{32}=0.875$。应当认识到，如可以根据本文中的实施例被作出的块大小确定可以关于一个或多个内容传输会话(例如，流传送媒体会话)先验地、与内容传输会话的发起同期地、在一个或多个内容传输会话期间动态地等以及在上述各项的组合的情况下被执行。相应地，如在图2的框209处被执行的确定块大小可以诸如通过在关于流200中所表示的其它操作的各种时间处实现前述技术中的一项或多项技术而被执行。

[0084] 此外，尽管已在本文中参考CM 122和RM 121的用于作出块大小确定的逻辑单元的可互操作性描述了实施例，但实施例可以利用用于块大小确定的其它逻辑单元。例如，实施例可以仅在CM 122内或者仅在RM 121内确定块大小，或者使用用于确定块大小的其它模块或者逻辑单元或者信息。例如，RM 121可以利用可以由CM 122提供的下载统计来动态地确定各种块请求的块大小。

[0085] 不考虑块大小确定的时序和用于作出确定的逻辑单元可以被部署在哪里，实施例的RM 121操作为，确定从CM 122请求哪些数据，以可靠地接收和恢复所请求的分段。相应地，实施例的RM 121提供一个或多个所选择的块请求(如可以是与由UA 129时常向CM 122作出的一个或多个分段请求相关联的)(图2的框203)。

[0086] 在根据实施例的操作中，RM 121可以从CM 122接收指示CM对于作出下一个块请求准备就绪的块准备就绪信令。例如，TA 120(例如，CM 122)可以实现用于确定用于提供关于块请求的控制(例如，确定何时将作出块请求和/或在CM 122与RM 121之间用信号发送下一

个块请求的准备就绪性)的一个或多个参数(例如,下载速率参数、门限参数等)的逻辑单元,如由所示出的实施例的框210示出的。下面提供了根据实施例的关于计算这样的参数以及它们的使用的细节。

[0087] 由RM 121向CM 122作出的块请求中的一些块请求可以是针对还未到达并且RM 121已视为可能永远不到达或者可能太晚到达的已被请求的数据的。额外地或者替代地,由RM 121向CM 122作出的块请求中的一些块请求可以是针对从原始分段被生成的经FEC编码的数据的,由此RM 121可以对从CM 122接收的数据进行FEC解码以恢复分段或者其某个部分。RM 121向UA 129传递所恢复的分段。相应地,可以存在根据本发明的实施例的RM的各种配置,各种配置诸如可以包括不使用FEC数据并且因此仅请求来自原始源分段的数据的部分的基本RM配置(RM-基本)和可以请求来自原始源分段的数据的部分以及从源分段被生成的相匹配的FEC分段的FEC RM配置(RM-FEC)。

[0088] 实施例的RM 121可能不知道时序和/或带宽可用性限制,因此促进RM121与CM 122之间的相对简单的接口,并且因此RM 121可以操作为在不由RM 121考虑这样的限制的情况下作出块请求。替代地,RM 121可以适于知道诸如可以由CM 122或者客户端设备110内的其它模块提供给RM121的时序和/或带宽可用性限制,并且因此RM 121可以操作为基于这样的限制作出块请求。

[0089] 实施例的RM 121适用于具有多个不同的CM配置的操作。此外,某些实施例的RM 121可以并发地与多于一个CM接合,诸如用以从多个CM请求分段中的相同分段或者序列的数据块。每个这样的CM可以例如支持不同的网络接口(例如,第一CM可以具有与设备上高速缓存的本地接口,第二CM可以使用与3G网络接口的HTTP/TCP连接,第三CM可以使用与4G/LTE网络接口的HTTP/TCP连接,第四CM可以使用与WiFi网络接口的HTTP/TCP连接,等等)。

[0090] 在根据实施例的操作中,CM 122与RM 121接合以接收块请求,并且通过网络150发送那些请求(图2的框204)。CM 122接收对块请求的响应(框205)并且将响应传递回RM 121(框206),其中,由UA 129请求的分段从由RM 121接收的块中被决定(框207),并且被提供给UA 129(框208)。CM 122的功能操作为决定何时请求由RM 121作出的块请求的数据。根据本文中的实施例,CM 122适于从通用或者传统服务器(即,还未被预先设计为与CA交互的服务器)请求和接收块。例如,CM 122从其请求数据的服务器可以包括在MBMS服务部署中被使用的BM-SC服务器。

[0091] 与上面讨论的RM 121一样,可以存在根据本发明的实施例的CM的各种配置。例如,可以提供多连接CM配置(例如,CM-mHTTP),由此CM适于通过多个TCP连接使用HTTP。多连接CM配置可以操作为诸如取决于网络状况、对数据的需求、拥塞窗口等动态地改变连接(例如,TCP连接)的数量。作为另一个示例,可以提供扩展传输协议CM配置(例如,CM-xTCP),其中,CM在扩展形式的TCP连接(在本文中被称为xTCP)之上使用HTTP。这样的扩展传输协议可以提供适于促进根据本文中的概念的由TA 120进行的分段的增强型传递的操作。例如,xTCP的一个实施例甚至当所发送的分组被丢失时将确认提供回服务器(与当分组被丢失时的TCP的重复确认方案不同)。这样的xTCP数据分组确认方案可以被TA 120采用,以避免服务器响应于确定数据分组缺失而降低数据分组被发送的速率。作为又一个示例,专有协议CM配置(例如,CM-rUDP),其中,CM使用专有用户数据报协议(UDP)协议,并且从服务器发送响应数据的速率可以是在不变的预先配置的速率处,或者协议内可以存在用于确保发送速

率尽可能高而不会不良地拥塞网络的速率管理。这样的专有协议CM可以与支持专有协议的专有服务器协作地操作。

[0092] 应当认识到，尽管已关于从服务器130请求来自源文件的数据的CM122讨论了所示出的实施例，但取决于CM具有的用于访问数据的接口的类型，源文件可以是在服务器上可用的，或者可以被本地存储在客户端设备上。在某些实施例中，包含从相匹配的源文件使用FEC编码所生成的修复符号的FEC文件可以还是在服务器上可用的。在这样的实施例中，可以例如存在针对每个源文件的一个FEC文件，其中，独立于被用于请求数据的CM的特定实施例地使用本领域中已知的FEC编码技术从源文件生成每个FEC文件。

[0093] 进一步地，根据实施例，客户端设备110可以能够(例如，通过WiFi或者蓝牙接口)连接到一个或多个在本文中被称为帮助器设备的其它设备(例如，被放置得邻近的各种配置的设备)，其中，这样的帮助器设备可以具有通过3G或者LTE连接的(对于不同的帮助器设备，潜在地通过不同载波)与诸如是服务器130的一个或多个服务器的连接。因此，客户端设备110可以能够使用帮助器设备的连接来向诸如是服务器130的一个或多个服务器发送块请求。在此情况下，TA 120内可以存在用于向帮助器设备中的每个帮助器设备进行连接和发送块请求以及接收响应的CM。在这样的实施例中，帮助器设备可以将相同分段的不同块请求发送给相同或者不同的服务器(例如，相同分段可以是对于多个服务器上的帮助器设备可用的，其中，例如不同服务器由相同或者不同的内容传递网络提供商来提供)。

[0094] 已在上面描述了实施例，在其中诸如使用数据对象标识符(例如，URL、URI、URN等)和块内容的数据对象内的字节范围来作出多个块请求，以用于将内容的期望的分段作为多个块进行传送。然而，某些内容服务器(例如，服务器130)配置可能不支持这样的字节范围请求，并且因此可以操作为响应于块请求返回完整数据对象而非字节范围的内容的块。相应地，如果仍然作出了多个块请求，则完整数据对象的多个实例将被传送给客户端设备。本文中的传输加速器的实施例操作为，检测是否字节范围请求被操作为内容的源的服务器所支持。例如，实施例的TA 120可以初始发出HTTP字节范围请求，并且分析响应代码以确定字节范围请求是否被支持(例如，接收206响应代码可以用于确定字节范围请求被支持，而接收200响应代码可以用于确定字节范围请求不被支持)。如果字节范围请求被确定为被支持(例如，响应于被发送到主机(起源)服务器的一个或多个块请求，HTTP206被接收)，则TA 120可以继续进行其用于将分段请求划分成块请求以便向该主机服务器进行发送的逻辑单元。然而，如果字节范围请求被确定为不被主机服务器支持(例如，响应于被发送到该主机(起源)服务器的一个或多个块请求，HTTP 200被接收)，则TA 120可以取消指定被发送到该主机服务器的字节范围的全部未完成的请求。在确定了服务器不支持字节范围请求(例如，服务器不是符合HTTP 1.1的)的情况下，TA 120可以操作为修改其操作，以避免使用将导致相同数据对象的多个实例被传输的多个块请求。例如，RM 121可以操作为作出对CM 122的完整数据对象(例如，内容文件)的请求，由此CM 122对应地从服务器130请求完整文件。尽管这样的操作不可以提供相比于在其中块请求被支持的传输加速器操作的最优性能，但在其中内容服务器不支持字节范围请求的情形下，这样的操作避免对内容的重复传送。

[0095] 应当认识到，甚至在字节范围请求被支持的情况下，最优性能也可能由于各种客户端设备和/或服务器配置而不产生。例如，由CM 122向服务器130作出的块请求的大小可以是相当小的，并且关于通过其作出这样的块请求的连接被实现的通信协议可以实现特定

技术，所述特定技术用于通过减少通过网络被发送的分组的数量来提高网络效率。这样的技术可能不良地影响(例如，延迟)本文中的块请求。例如，可以关于TCP连接实现Nagle的算法，以提供对多个小型出站消息的合并，并且一次发送它们全部(例如，只要存在对于其来说发送者未收到任何确认的被发送的分组，则发送者可以保持对其输出进行缓冲，直到它具有值得输出的完整分组为止，以使得输出可以被一次全部发送)。尽管通常提高网络效率，但Nagle的算法的操作可以延迟块请求。块请求的这样的延迟可以诸如通过导致流传送内容中的失速、直播流传送中的不允许的延迟等而导致不可接受的用户体验。相应地，实施例操作为禁用网络效率技术，例如，Nagle的算法，其操作为减少通过网络被发送的分组的数量。

[0096] 在字节范围请求被内容服务器支持的情况下，客户端设备可能仍然不具有关于将被请求的内容文件和/或分段的大小的信息。然而，为了提供根据本文中的实施例的传输加速的优点，实施例的TA 120的操作作出如上面描述的针对内容的块的请求。为获得关于内容文件以及因此其分段的大小的信息，TA 120可以操作为初始发出一个或多个小的块请求(例如，16KB请求)，由此，对那些请求的响应将包含来自服务器的实际内容大小。TA 120的逻辑单元(例如，RM 121和/或CM 122的逻辑单元)此后可以使用该信息来确定如何将分段划分成块和对剩余的请求进行调度。实施例的TA 120当内容文件的大小不是已知的时可以以与前述方式相似的替代方式操作。例如，如果TA 120将作出对于已知大小的内容文件的多个块请求，则当内容文件的大小在第一个块请求的时间处不是已知的时，TA 120可以作出相似数量的具有相似时序的请求的块请求。如果分段大小不是已知的，则RM121可以假设分段大小是无限的(或者无界的)，直到从具有被包括的大小的块请求接收响应为止。伴随该实施例，将潜在存在对于超过分段的结尾的分段的字节范围的块请求。因此，根据实施例，来自支持字节范围请求的服务器的对于块请求的响应可以：(a)如果所请求的字节范围与分段的实际字节范围之间存在完全重叠，则是HTTP 206OK响应，并且因此块请求的字节范围将在响应中从服务器被返回；(b)如果块请求中的所请求的字节范围与分段的实际字节范围之间不存在任何重叠，则是HTTP 404错误响应，并且没有分段的字节将在响应中被服务器提供；(c)是HTTP响应，其指示块请求中的字节范围与分段的字节范围之间存在部分重叠，并且块请求的字节范围与分段的字节范围之间的重叠将在响应中从服务器被返回。在这些情况中的一些或全部情况下，响应将还包括分段的大小。基于在所述响应中的任何响应中被接收的该信息，RM 121可以设置分段的实际大小以用于任何进一步的响应或者处置。

[0097] 从前述内容中应当认识到，实施例的RM 121适于恰当地处置对未知大小的分段的块请求的可能的响应。例如，具有错误码或者警告的HTTP响应将由RM 121无缝地处置，因为RM可以确定这些响应由于请求无效或者分段的部分无效的字节范围(一旦RM在响应中获得实际分段大小RM就可以确定)而本该是所预期的。

[0098] 前述的用于处置未知大小的分段的技术可以在多数环境中正确运行。例如，分段有可能比在单个RTT中被请求以用于HTTP渐进下载流传送的内容长得多，并且因此，由于分段的大小将在整个分段在块请求中被请求之前在响应中被获得，所以通常在已知分段大小之前所作出的块请求中的字节范围中的全部字节范围将是有效的字节范围。相似地，对于DASH流传送，分段的回放时间有可能在持续时间上是几个RTT，并且因此，除了在当下载速

率比当前的回放速率高得多时的情况下(在此情况下,可能存在少量以不与分段重叠的字节范围来发送的块请求,但这通常在所选择的回放显著在下载速率以下时发生,在此情况下,作出具有不与分段重叠的字节范围的块请求的影响被显著降低)之外,相同的分析是有效的。

[0099] 相应地,传输加速器可以操作为,甚至在不充足的信息是对于最优地实现块请求可用的情况下,立即请求内容,并且稍后当信息是可用的时使用信息来确定块大小和调度合适的块请求。

[0100] 如上面讨论的,可以提供多连接CM配置(例如,CM-mHTTP),由此CM适于使用多个连接来根据本文中的概念从一个或多个内容服务器请求和接收内容。在多个连接正在使用中的情况下,关于CM如何迫使不同连接是彼此公平的(例如,用以确保连接的下载速率保持与彼此合理地接近,诸如在二的因子之内)的公平性方面出现。

[0101] 一种用于提供关于多连接的一个水平的公平性的相对简单的技术包括在其中CM控制连接中的每个连接的接收窗口大小的实现方式。例如,CM122可以将全部TCP连接的接收窗口大小设置为是相等的,并且是大得足以使可用带宽可以被TCP连接近似完全利用的,但不是太大得使一些TCP连接不时以比其它TCP连接快得多的速率来下载的。这样的实现方式可以实现对于每个这样的TCP连接的近似相等的下载速率。

[0102] 一种避免控制接收窗口大小的用于提供关于多连接的公平性的替代技术包括适于控制每个连接上的已请求但还未被传递的数据的量的实现方式。该技术的目标在于提高连接的聚合下载速率,确保连接正在以近似相同的速率下载,以及减少在网络中被缓冲的数据的量。实现该技术的实施例利用一个或多个门限参数来决定何时另一个数据块将在多个连接中的特定连接上被请求。可以根据实施例关于多个连接中的每个连接利用前述门限参数的相同值。如果需要,替代实施例可以关于一个或多个连接来利用前述门限参数的不同值。

[0103] 根据控制每个连接上的已请求但还未被传递的数据的量的连接公平性实现方式的实施例,门限参数Thresh被定义为CM控制其值的多个八位字节,并且被用于决定何时数据的另一个块可以在连接上被请求。例如,当TCP连接上的已请求但还未被接收的数据的量在Thresh以下时,则可以在该连接上作出另一个数据块请求。然而,如果TCP连接上的已请求但还未被接收的数据的量在Thresh处或者以上时,则不在该TCP连接上作出另一个数据块请求。

[0104] 图3示出了根据使用门限参数Thresh来决定何时另一个数据块可以在连接上被请求的实施例的操作。在所示出的示例中,假设当对于全部当前请求的将接收的剩余数据的量下降到Thresh以下时,CM对于任何特定连接上的另一个块请求准备就绪。例如,CM 122可以包括前述的CM-mHTTP配置,并且可以利用如图3中所示的三个TCP连接,其中,CM已在全部三个连接上作出针对数据的HTTP块请求,以使得三个TCP连接中的每个TCP连接上的将被接收的剩余数据的量仍然在门限量Thresh以上。在该场景中,CM当前不能在这三个连接中的任何连接上再作出任何HTTP请求。相应地,CM对于来自RM的另一个块请求不是准备就绪的,因为即使CM未从RM接收另一个块请求,CM也不可以立即作出针对该块的请求。

[0105] 图4示出了根据使用门限参数Thresh来决定何时CM当前能够立即作出块请求的实施例的操作。在该示例中,再次假设当针对全部当前的请求将接收的剩余数据的量下降到

门限量Thresh以下时，CM 122对于任何特定连接上的另一个块请求准备就绪。在图4的示例中，针对所述连接中的至少一个连接(例如，TCP连接2和TCP连接3)已接收了足够数据，使得可以作出另一个请求。

[0106] 选择门限参数Thresh的值的目标在于，该值被选择为是大得足以使连接的聚合下载速率近似是给定的当前网络状况下的可能的最大下载速率的，而同时是尽可能小的，以确保不同连接正在以近似相同的速率下载数据，并且使得在网络中被缓冲的数据的量尽可能小。当HTTP流水线被用在个体的TCP连接上时，可以基于本文中描述的方法动态地确定Thresh的值。将块大小C选择为是尽可能小的促进前述内容。然而，如之前讨论的，块大小应当是大得足以使请求块的开销是被用于接收块响应的下载带宽的一小部分的。例如，如果C被设置为8KB，并且HTTP请求的大小是200字节，则针对该值C的块请求的相对开销是大约2.5％。假设TCP确认业务通常是TCP下载业务的少量百分比，则这是合理的折中。

[0107] 应当认识到，当流水线不被实现时，尽管如上面描述的那样直接使用前述门限参数变得有问题，但可以仍然实现基于门限的算法以提供本文中的连接公平性。例如，在固定数量N的TCP连接正被使用的情况下，如果每个块请求的大小是Thresh，则网络上的未完成的字节的总量将是至多N*Thresh。因此，如果链接到服务器的带宽延迟乘积大，则大值的Thresh可能是可取的，而如果带宽延迟乘积不大，则小值的Thresh可能是可取的。应当认识到，在这样的实现方式中，折中通常是与上面描述的流水线的情况下的折中相同的(即，门限参数Thresh的值应当是大得足以允许以接近最大可能速率的速率接收数据的，同时被保持为小得足以避免网络上的不必要的缓冲)。然而相比于流水线的情况，门限参数Thresh的值在该非流水线示例中应当小了大约N/2到N的因子。

[0108] 已概括地描述了用于实现关于由本文中的实施例的传输加速器所利用的多个连接的公平性的门限参数的使用，下面提供关于计算这样的门限参数的合适值的细节。应当理解，下面阐述的技术以替代的方式计算门限参数Thresh的值，并且可以被应用于流水线和非流水线的情况。

[0109] 在根据实施例的操作中，CM 122基于当前的网络状况动态并且连续地调整Thresh的值，因为Thresh的最优值可以根据网络状况变化。在实现根据本文中的实施例的用于确定Thresh的值(例如，图2的框210处的操作)的技术时可以采用多个下载速率参数(例如，DR、DFR和DCR)。这样的下载速率参数提供对数据、数据的分段和/或数据的块多么快地被从服务器下载到客户端的测量，并且可以通过多少个字节已在时间窗口中被接收而被确定。例如，DR(下载速率)被定义为连接(例如，TCP连接)的聚合平均下载速率(例如，在合适的时间窗口上被测量，或者被使用加权移动平均进行平均)。DFR(下载分段速率)被相似地定义为连接的聚合平均下载速率，只是在DFR的情况下，每个分段的第一个分组以及接收分段的第一个分组与接收前面的分组(来自任何分段，通过TCP连接中的任何TCP连接)之间的时间不被包括在平均中(即，“被减损(discount)”)。同样地，DCR(下载块速率)被定义为连接的聚合平均下载速率，除了在DCR的情况下，每个块的第一分组以及接收块的第一个分组与接收前面的分组(来自任何块，通过TCP连接中的任何连接)之间的时间不被包括在平均中(即，“被减损”)。

[0110] DCR通常将是相对高的(即，高于如果下载速率将在包括第一个分组块的字节并且包括第一个分组与前一个分组之间的时间的整个时间窗口上被平均情况下的DCR)，并且在

某种意义上，它可以表示取决于例如网络状况、被使用的TCP连接的数量和其它因素的接口上的真实可用带宽。通常，将是这样的情况：DCR是至少DFR，尽管可能不总是这样的情况，并且有时DFR可以大于DCR。通常，DCR计算的目标在于，在不需要来自服务器或者其它外部网络单元的明确反馈的情况下，给出指示如何控制数据的流水线的值。

[0111] 可以诸如根据以下公式使用这样的下载速率参数来确定和/或调整门限参数Thresh：

[0112] 如果DCR>DFR*1.05，则增大Thresh的值

[0113] 否则，如果DCR<＝DFR*1.05，则减小Thresh的值

[0114] 其中，Thresh的增大/减小值可以是预定的值(例如，2KB)，可以取决于DCR比DFR的相对比率，等等。

[0115] 可以不时地使用这样的用于确定门限参数Thresh的技术。例如，Thresh的值可以在预定的时间段之后(例如，每两秒)、在事件发生时(例如，在诸如是10个RTT的多个RTT之后，或者每当块请求被CM接收时)等时候被调整。

[0116] 实施例可以额外地或者替代地操作为动态地测量所述下载速率参数中的一个或多个下载速率参数，并且相应地确定/调整门限参数。例如，在确定/调整Thresh之后，实施例可以基于在对Thresh的调整已发生之后已被发出的块来测量DCR和DFR。此后，可以使用这些已更新的下载速率参数来调整Thresh的值。DCR和DFR的前述经动态更新的测量可以例如是基于所发出的固定数量T的块请求的。这样的技术具有DFR和DCR的测量周期不取决于RTT的优点，这是可取的，但不需要明确地测量RTT。另一个优点在于，它避免随后的调整将基于在Thresh最后一次被调整之前发生的对DCR和DRF的测量而作出、并且因此不再反映当前的网络状况的可能性。

[0117] 本发明的实施例可以利用除了或者替代前述的下载速率参数(即，DR、DFR和DCR)的下载速率参数。例如，代替在上面的Thresh确定中使用DFR，实施例的CM 122可以使用DDDR(下载减损延迟速率)，所述DDDR被定义为连接的聚合平均下载速率，除了在DDDR的情况下，如下面详述的，当计算DDDR时，每个被延迟的块请求的第一个分组的字节以及每个被延迟的块请求的第一个分组与前一个被接收的分组之间的时间不被计数(即，“被减损”)。

[0118] 在实现利用DDDR下载速率参数的实施例时，CM 122可以操作为，如果块请求在CM用信号通知它对于下一个块请求准备就绪时立即被RM121提供给CM，则将块请求分类为“未被延迟”。相似地，如果块请求未在CM信号通知它对于下一个块请求准备就绪时立即被RM121提供给CM(例如，存在诸如由于当CM信号通知它对于下一个块请求准备就绪时RM不具有它可以作出的块请求所引起的某种延迟)则CM 122可以将块请求分类为“被延迟”。CM 122可以因此将DDDR计算为下载速率，除了CM在计算DDDR的分子时不对每个被延迟的块请求的第一个分组的字节进行计数，并且在计算DDDR的分母时不对每个被延迟的块请求的第一个分组与前一个被接收的分组之间的时间进行计数。实施例的CM可以操作为，为了一致性，将它在一开始时接收的第一个块请求看作被延迟的块请求。

[0119] 从前述内容可以认识到，使用DDDR的实现方式可以使用与其它实现方式(例如，在其中使用DFR的实现方式)稍微不同的CM 122与RM 121之间的API。例如，当CM用信号向RM通知它对于块请求准备就绪时，RM可以立即利用块请求作出响应(假设RM具有可用于作出的块请求)，在此情况下，在计算DDDR时块请求被CM归类为未被延迟的块请求。替代地，如果RM

向CM提供不是响应于来自CM的、关于其对于块请求准备就绪的信号的块请求，则CM将在计算DDDR时把块请求归类为被延迟的。可以在RM与UA之间定义相似的API。

[0120] 可以如下面阐述的那样计算实施例的DDDR下载速率参数：

[0121]

参数	意义
Z	到目前为止所下载的字节数（排除被减损的字节）
Tr	下载在其期间活动的时间（排除被减损的时间）
S	存储临时时间
Tw	当前的挂钟时间

[0122] Z＝0

[0123] Tr＝0

[0124] 每当分组P被接收时(大小为B个字节,在挂钟时间Tw处)：

[0125] 如果P是被延迟的块的第一个分组，则{S＝Tw}

[0126] 否则，如果P不是被延迟的块的第一个分组，则

[0127] {Tr＝Tr+(Tw-S)；

[0128] Z＝Z+B；

[0129] S＝Tw

[0130] }

[0131] DDDR＝Z/Tr

[0132] 在根据实施例的操作中，DDDR可以在不是下载的全部时间的时间窗口上被计算或者求平均。例如，假如DDDR将在持续时间为W秒的最后一个时间窗口上被计算，则分子Z_W和分母Tr_W可以在最后W秒上被计算，由此，DDDR＝Z_W/Tr_W。本领域的技术人员将认识到，可以使用相似的技术来计算DDDR的其它变型，诸如使用具有半衰期W的指数加权移动平均(EWMA)。

[0133] 应当认识到，使用DDDR而不是DFR的优点在于，计算DDDR不利用任何从RM被传递给CM的特殊知识。特别地说，CM可以在RM没有用信号向CM通知哪些块属于新的分段的情况下计算DDDR值。此外，使用DDDR允许CM准确地测量CM实际正在下载的时间，而没有任何特殊情况。相反，使用DFR呈现关于当存在每分段的仅一个块时DFR与DCR之间不存在任何差别的问题。因为使用DDDR不具有这个问题，所以当每分段仅存在一个块时，采用DDDR的实施例可以被使用，而没有特殊例外。作为进一步的优点，当RM由于分段还不可用而将不必然能够为CM提供该分段的第一个块请求时，DDDR是对于直播案例的合适测量，但是更正确地处理当下一个分段的第一个块可以在CM用信号发送对于下一个块请求的准备就绪性时被作出时的情况。相反，采用DFR的实施例将该新分段的第一个分组减损，尽管如果CM对于下一个块请求准备就绪的话不存在任何将该第一个分组减损的原因。相似地，由于与上面相同的原因，DDDR是针对使用低/高水位策略的下载情况的合适测量(即，伴随DDDR和DFR两者，在随着缓

冲器填满低和高水位之间而下载时立即被请求的分段的第一个分组不应当被减损，然而在达到高水位之后和等待直到耗尽到低水位之前被请求的分段的第一个分组应当被减损)。

[0134] 实施例以上面关于下载速率参数DFR描述的全部方法来使用下载速率参数DDDR。例如，实施例的CM 122可以利用DDDR来作出关于如何调整门限值Thresh的决策，其中，Thresh是在其以下作出新的块请求的TCP连接的缓冲器水平。即，如果B是TCP连接上的已请求但还未被接收的数据的量，则当B＜Thresh时，可以由CM作出新的块请求。

[0135] 下载速率参数DPR(下载流水线速率)是可以除了或者替代前述的下载速率参数(即，DR、DFR、DCR和DDDR)而被使用的下载速率参数的另一个示例。DPR可以例如被用作在上面的Thresh确定中使用DFR或者DDDR的替代。根据实施例被利用的DPR与DFR、DCR或DDDR相似，尽管存在如下面阐述的差别。

[0136] DPR被定义为连接(例如，TCP连接)的聚合平均下载速率，除了DPR的计算将在请求被作出时不具有任何未完成的数据的、在连接上被作出的块请求的第一个分组减损。在计算DPR时，如果组请求被延迟(例如，在RM被通知CM对于下一个块请求准备就绪时，没有作出块请求)，则块的一些最先的分组被减损。因此，DPR与真实下载速率几乎相同，并且对于初始时段，DPR与DCR之间可能不存在任何差别。例如，如在下面详细描述的，在会话一开始时在不同连接上作出的最先一些块请求将被分类为不被流水线操作的，(并且因此，这些块的最先的分组将不被DPR或者DCR计数)，并且相似地，如果这样的分段仅在给定的时间轴上可用，并且前一个分段在下一个分段可用之前被完全下载，则直播场景中的分段请求的最先的块请求可以被分类为不被流水线操作的。相似地，当使用高低水位下载方法时，在将媒体缓冲器耗尽到低水位之后立即作出的对于按需内容的块请求可以被分类为不被流水线操作的，因为当这些块请求被作出时，全部TCP连接都不在使用中。

[0137] 对于利用DPR的实施例，当下一个块请求在连接上被作出时，该连接的已请求但还未被接收的数据的量可以被称为经网络缓冲的数据Buff。门限参数Thresh表示该连接的当前的流水线门限。相应地，如上面讨论的，当Buff<Thresh时，新的块请求可以被放到连接上。

[0138] 假设CM在全部连接上使用流水线。在根据实施例的操作中，如果块请求在Buff>＝alpha*Thresh时被接收，则CM可以将对于特定连接的块请求分类为“被流水线操作”(未延迟/按时)以便确定DPR，并且如果块请求在Buff<alpha*Thresh时被接收，则CM可以将块请求分类为“不被流水线操作”(被延迟/不按时)以便确定DPR，其中是alpha是<1的常量(例如，alpha＝1/2或者alpha＝2/3)。

[0139] 在不使用任何流水线的情况下，用于确定DPR的对块请求进行迟分类的技术可能与前述的技术不同。在实施例的迟分类技术中，计算所发出的不完整的块请求的数量R，所述所发出的不完整的块请求在正被考虑的块请求的响应的第一部分被返回的时间点处已接收了部分响应。如果R大于固定的常量fmin，则实施例的CM可以将请求分类为按时的，并且否则将它分类为延迟的。

[0140] 额外地或者替代地，在不使用流水线的情况下的用于确定DPR的对块请求的迟分类的技术包括分析与内容服务器的空闲连接。例如，在请求被发出的时间处，CM 122的逻辑单元可以对与服务器130的空闲TCP连接的数量进行计数。如果空闲连接的数量超过给定的门限(例如，可用连接的一半)，则请求可以被分类为延迟的。否则请求可以被分类为按时

的。

[0141] 不考虑在将请求分类为被流水线操作的或者延迟的时所采用的特定技术，可以利用被流水线操作/延迟的分类来计算DPR。在根据实施例计算DPR时，如果块请求被分类为被流水线操作的则该块请求的第一个分组不被减损，并且如果块请求被分类为不被流水线的则该块请求的第一个分组被减损。如前述内容中使用的，“被减损”表示在该第一个分组中被接收的字节不被计数，并且接收该第一个分组与前一个被接收的分组之间的时间不被计数，而“不被减损”表示在该第一个分组中被接收的字节被计数，

[0142] 并且接收该第一个分组与前一个被接收的分组之间的时间被计数。

[0143] 实施例的DPR下载速率参数可以如下面阐述的那样被计算(应当认识到，下面的示例在确定“被流水线操作”(未延迟/按时)和“不被流水线操作”(延迟或者不按时)分类时利用了前述的经网络缓冲的数据Buff和门限参数Thresh分析技术，而其它实施例可以利用诸如是上面描述那些技术的替代技术来进行这些分类)：

[0144]

参数	意义
Z	到目前为止所下载的字节数(排除被减损的字节)

[0145]

Tr	下载在其期间活动的时间(排除被减损的时间)
S	存储临时时间
Tw	当前的挂钟时间

[0146] 在TCP连接上作出具有Buff个八位字节的已请求但还未被接收的数据的块请求，其中，Thresh是该TCP连接的以八位字节计的当前的流水线门限。

[0147] 如果(Buff>=alpha*Thresh)，则将该块请求分类为“被流水线操作”

[0148] 否则，如果(Buff<alpha*Thresh)则将该块请求分类为“不被流水线操作”

[0149] 响应于块请求，在TCP连接上接收块。

[0150] Z=0

[0151] Tr=0

[0152] 每当分组P被接收时(大小为B字节，在挂钟时间Tw处)：

[0153] 如果P是被分类为“不被流水线操作”的块的第一个分组，则{S=Tw}

[0154] 否则，

[0155] {Tr=Tr+(Tw-S)；

[0156] Z=Z+B；

[0157] S=Tw

[0158] }

[0159] DPR=Z/Tr

[0160] 在根据实施例的操作中，可以在不是下载的全部时间的时间窗口上对DPR进行计

算或者求平均。例如，假如将在持续时间为W秒的最后的时间窗口上计算DPR，则可以在最后W秒上计算分子Z_W和分母Tr_W，由此，DPR＝Z_W/Tr_W。本领域的技术人员将认识到，相似的技术可以被用于计算DCR或者DPR的其它变型，诸如是使用具有半衰期W的EWMA。

[0161] 作为根据本文中的实施例的使用EWMA计算DCR和DPR的示例，假设不被减损的分组(不是块的第一个分组)被接收，该分组包含B比特的数据，并且该分组与(任何类型的)前一个分组的到达之间的时间是dt。可以如下地更新DCR：

[0162] DCR＝DCR*exp(-alpha*dt)

[0163] DCR＝TDCR+B/alpha

[0164] 前述内容中的alpha的值可以被选择为例如使得一段时间上的平均衰减与1/alpha成比例。将alpha的时间单位与实施例的时间单位dt对齐。例如，如果在前述内容中用毫秒来表述dt，并且目标是秒上的1/e的衰减，则根据一个实施例，alpha＝1/1000。作为另一个示例，如果用秒来表述dt，并且目标是秒上的1/e的衰减，则根据一个实施例，alpha＝1。进一步采用前述的示例实施例，对于可能的多秒(例如是5秒)上的1/e的目标衰减，如果用毫秒来表述dt，则alpha＝1/5000，并且如果用秒来表述dt，则根据本文中的实施例，alpha＝1/5。

[0165] 上面的用于使用EWMA来计算DCR的概念可以被应用于使用EWMA对DPR的计算(将块的未被足够地进行流水线操作的第一个分组减损)。例如，用于使用EWMA计算DPR的实施例可以对于DCR和DPR两者使用相同值alpha。然而，替代实施例可以对DCR和DPR使用不同的平均常量(例如，对于DCR使用alphaC，以及对于DPR使用alphaP)。

[0166] 已描述了用于计算下载速率参数DCR和DPR的技术，下面将提供在调整门限值Thresh(即，在其以下作出新的块请求的连接的缓冲器水平)时的它们的使用的示例。应当认识到，存在可以通过其来根据本文中的概念使用DCR和DPR获得Thresh的值的许多技术。

[0167] 作为将DCR和DPR用于调整门限参数Thresh的值的一个示例性示例，让R是当HTTP块请求被CM作出时与当对该请求的第一个响应被CM接收时之间的平均响应时间。对于每个块请求，让RTT是当请求被作出时与当第一个响应被接收时之间的响应时间。使用前述内容，R＝(1-a1)*R+a1*RTT。可以认识到，存在用于计算R的许多方法，诸如，使用EWMA在时间窗口上或者在RTT测量的数量上对所测量的RTT求平均，通过在固定的之前的时间窗口上或者在RTT测量的固定的之前的数量上求平均等。

[0168] 在根据实施例利用DCR和DPR来调整Thresh时，让Thresh是跨被分配给门限的全部连接的字节的总量。可以如下地定期(例如每a2*R秒地)更新Thresh的值：

[0169] 如果(DCR*a3>DPR)，则Thresh＝min{Thmax,Thresh(1+a4)}

[0170] 否则，如果(DCR*a5>DPR>＝DCR*a3)，则Thresh＝min{Thmax,Thresh*(1+a6)}

[0171] 否则如果(DCR*a5<＝DPR)，则Thresh＝max{Thmin,Thresh*(1-a7)}其中，常量a1、a2、a3、a4、a5、a6和a7以及门限最小值和最大值Thmin和Thmax的示例值如下：

[0172] a1＝0.05

[0173] a2＝2

[0174] a3＝0.7

[0175] a4＝0.5

[0176] a5＝0.95

[0177] a6＝0.1

[0178] a7＝0.05

[0179] Thmin＝32KB

[0180] Thmax＝1MB

[0181] 从前述内容中可以认识到,DCR将是至少DPR。在DPR显著小于DCR的情况下,它是对块的第一个分组与前一个分组之间存在大的间隙(例如,块请求未被足够快地作出)的指示。在DCR和DPR是相同或者接近相同的值的情况下,实施例可以操作为保持门限参数Thresh在它的当前值处,或者可能缓慢地升高该值。

[0182] 在根据本文中的实施例调整门限参数Thresh时,目标是获得将与块内的任何其它分组之间的间隔近似相同的块的第一个分组与前一个块的分组之间的分组间间隔(即,分组间间隔≈分组内间隔)。当达到这样的条件时,足够的数据已被流水线操作,并且请求已被提前足够远地作出,使得带宽效率最优化。因此,尽管实现小的门限参数以确保不同连接正在以近似相同的速率下载数据,并且使在网络中被缓冲的数据的量尽可能小可能是可取的,但实施例操作为实现足够大的门限以避免分组间间隔中的显著间隙。

[0183] 应当认识到,根据本文中的概念,可以关于前述内容来利用各种替代方案。例如,在DCR和DPR的某些值处,当作出调整Thresh的值的决策时,Thresh的值可以不改变(例如,当DPR小于但大致上等于DCR的值时,诸如是当DPR/DCR的值在区间(0.9,0.95)内时)。相应地,替代的实施例可以操作为通过设置Thresh＝Thresh*(DCR/DPR)来调整Thresh的值。尽管该技术与上面讨论的方案相似,但它在可以被应用的调整的量方面更灵活,并且因此可以减少配置参数的数量。作为替代实施例的另一个示例,可以通过设置Thresh＝Thresh+delta*(1-DPR/DCR)来调整Thresh的值,其中,delta＝32KB。该替代技术是基于加性的增大的,并且可以较缓慢地收敛。应当认识到,实现用于调整门限参数Thresh的前述替代技术中的任一项替代技术的实施例可以继续应用门限最小值和最大值限制Thmin和Thmax。

[0184] 可以认识到,各种下载速率参数(例如,DPR、DCR和DFR)可以被用于确定允许多少已请求但还未被接收的数据和用于管理每个连接上的已请求但还未被接收的数据的量。此外,如本文中提供的下载速率参数(例如,DPR和DFR)可以被用于向UA提供(例如,用于确定何时请求和/或请求什么分段的)下载统计。

[0185] 从前述内容中可以认识到使用DPR和DCR来计算门限参数值Thresh的各种优点。例如,DPR的值区分流水线值Thresh是否提供用于达到下载速率DCR的足够流水线,因为DPR与DCR之间的差别在于,DPR将被流水线操作的块的最先的分组包括为下载速率的一部分,而DCR不包括这些。因此,如果Thresh是足够大的,则DCR与DPR的值之间应当仅存在小的差别,而如果Thresh不是足够大的,则DPR的值与DCR的值之间应当存在较大差别,其中,通常,DCR的值大于DPR的值。作为进一步的优点,CM可以在不具有来自RM的任何信令或者输入的情况下计算DPR。

[0186] 如前面提到的,可以在确定何时关于多连接CM配置(例如,CM-mHTTP)的连接作出块请求时利用前述的下载速率参数和门限参数,由此,CM适于根据本文中的概念使用多个连接来从一个或多个内容服务器请求和接收内容。例如,在B是全部连接上的已请求但还未被接收的字节的数量的情况下,可以基于将B与Thresh进行的比较来作出关于新的块请求是否可以被发出的确定(例如,如果B<Thresh,则新的块请求可以被发出)。

[0187] 在新请求将在多连接(例如,CM-mHTTP)配置中被发出的情况下,可以选择用于该新请求的多个连接中的合适的连接。例如,根据实施例,可以为新请求选择具有最少数量的未完成的字节(例如,已请求但还未被接收的字节)的连接。然而,替代实施例可以单个地对于每个连接作出关于新请求的决策。实施例可以例如为块请求选择连接以提供负载均衡,由此,针对任何特定连接的块请求可以基于其一个或多个下载速率参数被进行加权。在一个连接上的还未被内容服务器完整地提供服务的块请求被重新发送的情况下,可以整体或者部分地在一个或多个不同的连接上重新发送该块请求,以降低该请求的完成时间变得太迟的可能性。

[0188] 可以例如根据下面的公式对于N个活动连接中的每个活动连接作出关于是否作出新的块请求的决策:

[0189] 如果(B<Thresh/N),则该连接可以接受另一个块请求其中,B是该连接上的已请求但还未被接收的字节的数量。

[0190] 可以由提供多连接配置(例如,CM-mHTTP)的CM来动态地调整将被使用的连接(例如,TCP连接)的数量N。例如,可以预先选择用于CM实例的连接的最小和最大数量(例如,最小连接=2,以及最大连接=8),由此,可以基于一个或多个操作度量来动态地调整在任何给定时间处被使用的连接的特定数量。诸如在分组错误率是与RTT相关的并且因此额外的连接可以被支持的情况下,添加连接可以例如被用于提供额外的吞吐量。

[0191] 通常,实施例操作为使关于任何特定连接的已请求但还未被接收的数据在量上小于该特定连接的拥塞窗口。在拥塞窗口小的情况下,因此会建议比在拥塞窗口大的情况下大的数量的连接。不幸的是,客户端设备(例如,客户端设备110)通常不能访问拥塞窗口信息。相应地,本文中的实施例可以操作为在动态地调整由CM使用的连接的数量时利用如上面描述的下载参数和/或门限参数。

[0192] 在根据实施例动态地调整连接的数量时,让X_i=min{CWND_i,RWIN_i,SWIN_i},其中,分别地,对于连接i,CWND_i是发送者的当前的拥塞窗口大小,RWIN_i是接收者窗口大小,以及SWIN_i是发送者窗口大小。额外地,让RTT是平均RTT,例如,被测量为当块请求被作出的时间与当对该块请求的第一个响应通过连接被接收的时间之间的平均时间的平均RTT。DPR通常是至多X_i/RTT在i上的和,相似地,DPR通常是至多BuffAvg/RTT,其中,BuffAvg是全部连接内的已请求但还未被接收的数据的平均量。实施例的CM具有对RTT和DPR的测量,并且可操作为控制门限参数Thresh和块请求大小(如上面讨论的),并且因此控制BuffAvg。尽管这样的CM可能不直接知道X_i的对于每个i的值,但该CM仍然可以根据以下公式调整连接的数量N:

[0193] 如果DPR小于BuffAvg/RTT,则连接的数量N可以被增大。

[0194] 如果DPR近似等于BufAvg/RTT,则连接的数量N可以被减小。

[0195] 已在上面描述了关于利用用于为客户端设备传送内容的多个连接的CM配置的实施例。可以关于一个或多个内容服务器(例如,如图1A中所示的内容服务器130或者如图5中所示的内容服务器130a和130b)作出这样的连接。相应地,应当认识到,可以关于使用实施例的传输加速器来提供内容的传送的一个或多个内容服务器应用本文中的概念。

[0196] 在根据实施例的操作中,CM负责建立和管理跨越一个或多个接口(例如,LTE接口、WiFi接口等)的与一个或多个内容服务器的全部连接,以及向RM指示CM何时对于将在连接

中的一个连接上被作出的另一个块请求准备就绪。让Thresh为由CM管理的接口的以八位字节(字节)计的门限值,其中,如上面描述的,基于DCR和DPR向上或者向下调整Thresh的值,只是对DPR的定义被如下地修改。让B是跨越由CM管理的接口的全部连接的、已由CM请求但还未被接收的八位字节(字节)的总数。当块请求被CM作出时,如果B<alpha*Thresh,则块请求在DPR计算中被看作被延迟的,而如果B>=alpha*Thresh,则块请求在DPR计算中被看作未被延迟的。让ND是除了与每个活动的内容服务器的一个连接之外CM可以使用的连接的目标总数。ND的值可以由CM动态地确定,或者其可以被设置为常量值(例如,ND=8)。让NO是CM当前正从事于其块请求的内容服务器(起源服务器)的数量。则,CM维护的连接的总计目标数量近似是N=ND+NO,这是基于以下内容的:

[0197] 让B_I是跨越由CM管理的接口的与起源服务器I的全部连接的、已由CM请求但还未被接收的八位字节(字节)的总数。

[0198] 让N_I=ND*(floor(sum_J<=I{B_J}/sum_I{B_I})-floor(sum_J<I{B_J}/sum_I{B_I}))+1是通过由CM管理的接口与起源服务器I的连接的目标数量,并且因此sum_I{N_I}=ND+NO。

[0199] 当B<Thresh时,则CM用信号向RM发送它对于块请求准备就绪。

[0200] 当CM从RM接收块请求时,则

[0201] 确定块请求是针对哪个起源服务器I的

[0202] 如上面描述的来确定N_I

[0203] 确定A_I=与I的活动*连接的数量

[0204] 如果A_I>=N_I,则在具有最少量的已请求但还未被接收的数据的与I的活动连接上作出针对块的HTTP请求

[0205] 如果A_I<N_I,则开启与I的新连接,并且在该连接上作出针对块的HTTP请求

[0206] 如果当针对块的HTTP请求被作出时B<alpha*Thresh(当计算DPR时,不对块的第一个分组和块的第一个分组与前一个分组之间的时间进行计数),则将块请求分类为被延迟的。

[0207] 在根据前述示例性实施例的操作中,如果连接具有已请求但还未被传递的数据、连接不具有已请求的数据但还未超时(如由客户端测量的)或者连接被服务器用信号通知已超时,则连接被认为是活动的。

[0208] 如前面讨论的,图1B提供了TA 120的RM 121和CM 122部件的高层架构。下面参考图1C的传输加速器配置来提供关于TA 120的一个示例性实施例的进一步的细节,以为实现TA 120的这样的实施例的部件提供上下文。

[0209] 该示例性实施例的RM 121优选被配置为在TA 120内提供以下功能。RM 121从TA请求分派器1024接受将被加速的来自UA 129的请求,以进行数据下载。在根据实施例的操作中,RM 121可以用信号向TA请求分派器1024通知对接受将被加速的另一个数据请求的准备就绪性。RM 121将数据请求拆分成块请求。在根据实施例的操作中,针对起源服务器I的每个块请求的优选大小$C_{Target(I)}$由CM 122来指示。实施例的RM 121在优选大小附近均匀地展宽块请求大小。RM 121向CM 122发出块请求。在根据实施例的操作中,块请求在CM 122指示它对于新的请求准备就绪时被发出。RM 121可以在调度算法的限制内调度块请求,以确保跨越来自不同UA 129的请求队列的公平性,并且同时确保合适的请求优先级划分。RM 121

将对块请求的响应组装成对来自UA 129的请求的响应。RM 121可以然后将这些响应传送回UA 129，以使得数据以有序和连续的方式被传递给上层。

[0210] 这个示例性实施例的CM 122优选被配置为在TA 120内提供以下功能。CM 122管理去往每个起源服务器的HTTP块请求的数量。实施例的CM 122发出对多个起源服务器的块请求。在根据实施例的操作中，块请求的数量在所定义的配置参数之间变化。例如，实施例的CM 122计算DCR和DPR，其中，DCR和DPR被用于计算聚合值T，所述聚合值T被用于计算由RM121进行的请求分块的优选块大小$C_{Target(I)}$。根据实施例的这样的操作的目的在于，使块足够大足以获得好的速率，但保持所请求数据的总量尽可能小。实施例的CM 122决定对向栈处理1025发出新的块请求的准备就绪性。在根据实施例的操作中，CM 122将块请求准备就绪性传送给RM 121。根据实施例的这样的操作的目的在于，在栈处理1025中保持连接繁忙，但不提前过远地使块请求排队等候。CM 122可以向栈处理1025作出针对块的请求(例如，HTTP请求)，并且向RM 121提供数据响应。

[0211] 已描述了该示例性实施例的TA 120及其RM 121和CM 122的一般功能，下面提供如可以在根据示例性实施例的TA 120的RM和CM部件内实现的算法。

[0212] 在根据实施例的操作中，CM 122保持对DCR和/或平均DCR的测量。CM 122可以例如通过测量下载速率但减损块响应之间的间隙来完成此。为减损块之间的间隙，实施例的CM 122减损针对每个块所接收的第一个数据中的字节，并且还减损块的第一个数据的到达与针对前一个被接收的块所接收的最后的数据的到达之间的时间。在图7中示出了DCR减损的一个示例，其中，被减损的时间和数据由Xs指定。通常，如图7中所示，针对HTTP请求所接收的响应数据的区块(block)是TCP网络分组的净荷。然而，由于传输错误和其它缺陷，每个单个TCP段可以不是隔离地被接收的，而相反是在较大区块中被接收的(例如，特别是当通过具有小RTT的快速链路下载时)。因此，被一次接收的个体数据区块可以不提供对TCP分组的良好近似，因为被一次接收的个体数据区块是包括由CM 122一次接收的数据的多个TCP分组的大得多的区块。下面描述的由CM 122进行的DCR计算考虑了一次接收数据的大区块的可能性，并且相似地，下面描述的由CM122进行的DPR计算考虑了一次接收数据的大区块的可能性。

[0213] DCR是对如果块请求是任意大的话可以在给定状况下达到的下载速率的估计。应当指出，如果块请求小，则发出每个块请求与开始接收响应之间的空闲时间将负面地影响总体下载速率。增大块大小将因此某种程度上增大有效的下载速率。DCR估计被针对该效应进行校正，并且因此是基本上独立于所选择块大小的。

[0214] 为计算DCR，根据实施例计算运行变量TDCR和ZDCR，其对应于在接收数据上花费的时间的累积量和当每个块的第一个被接收的区块被丢弃时被接收的数据的量。还根据实施例利用帮助器数据结构unseen_chunk_ids，其包含块ID，并且跟踪针对其的请求已被发出，但还没有数据被接收的块ID。

[0215] 前述的根据实施例的变量和数据结构的初始化可以是如下面所示那样的，其中，MSS是优选被设置为接近TCP分组的最大大小的常量(例如，MSS＝1500字节)，并且Cmult是小于1的常量(例如，Cmult＝2/3)：

[0216] 设置T_{DCR}＝0

[0217] 设置Z_{DCR}＝0

[0218] 设置unseen_chunk_ids＝empty_set()

[0219] 设置$T_{last_data_recv}$＝现在(以毫秒计)

[0220] 在根据实施例的操作中,当新的块请求C从RM过来时,它的ID被如下面所示那样添加到未看见的块ID的集合:

[0221] unseen_chunk_ids.add(chunk_id$_C$)

[0222] 然后根据实施例向栈处理1025发出块请求C。

[0223] 当块请求C的数据到达时,根据实施例执行以下步骤以更新TDCR、ZDCR和unseen_chunk_ids:

[0224] 让current_time是当前的时间(以毫秒计)

[0225] 让deltaT＝current_time-$T_{last\_data\_recv}$(自最后被接收的任何块的任何数据以来的时间)

[0226] $T_{last\_data\_recv}$＝current_time

[0227] 让deltaZ＝被接收的数据的量(包括HTTP报头开销)

[0228] 如果chunk_id$_C$在unseen_chunk_ids中:

[0229] 如果deltaZ>MSS则

[0230] T_{DCR}＝T_{DCR}+deltaT*Cmult*(deltaZ-MSS)/deltaZ

[0231] Z_{DCR}＝Z_{DCR}+deltaZ-MSS

[0232] 从unseen_chunk_ids中移除chunk_id$_C$

[0233] 否则:

[0234] T_{DCR}＝T_{DCR}+deltaT

[0235] Z_{DCR}＝Z_{DCR}+deltaZ

[0236] 从前述内容中可以认识到,如果deltaZ<＝MSS是针对对于其来说没有之前的数据已被接收的块所接收的数据的量,则deltaT和deltaZ全部被减损(即,什么都不添加到TDCR和ZDCR)。然而,如果deltaZ>MSS是针对对于其来说没有之前的数据已被接收的块所接收的数据的量,则取代将deltaT和deltaZ全部减损,根据前述内容的操作将deltaT减损一个因子Cmult*(deltaZ-MSS)/deltaZ,并且将被减损的值添加到T_{DCR},并且进一步将deltaZ减损一个因子(deltaZ-MSS)/deltaZ,并且将被减损的值添加到Z_{DCR}。因此,实质上仅被接收的数据的第一个MSS被减损,而剩余部分在DCR参数Z_{DCR}和T_{DCR}中被计数。将deltaT额外减损Cmult确保被减损的deltaZ值除以被减损的deltaT值的比值大于deltaZ除以deltaT的比值,这继而确保当由于Cmult<1,deltaZ>MSS时,因接收针对对于其来说没有之前的数据已被接收的块所接收的数据而产生的对DCR的总体影响总体来说将比对DPR的影响更积极。

[0237] TDCR和ZDCR是单调递增的数。实施例的CM 122进一步操作为记录DCR的一些历史,并且因此CM可以保存(TDCR,ZDCR)对的数组,例如可以按它们的历史次序来进行存储。该数组在本文中被称为mapTDCRZDCR。每当新的数据被接收时,mapTDCRZDCR的更新根据实施例发生。在实践中,从mapTDCRZDCR移除旧的条目以避免历史的无界增长可能是可取的,并且因此mapTDCRZDCR可以根据本文中的实施例被存储在循环队列中。

[0238] 对于对示例性实施例进行的讨论的剩余部分,DCR(dcr_dpr_estimate_win)表示在前一个持续时间为dcr_dpr_estimate_win的窗口上被估计的DCR,其中,dcr_dpr_estimate_win是配置参数。

[0239] HTTP客户端的所达到的下载速率取决于该客户端多么积极地请求数据。例如，如果大文件正在被下载，则可以达到高的下载速率，这是因为什么需要被请求是提前清楚的，并且因此TA 120可以被操作为使得它将永远不会急需将要发出的块请求。另一方面，如果客户端是例如播放直播内容的DASH客户端，则它可能仅能够提前请求数据的小片段(因为在久远的未来视频不可用)，并且因此可能不能够达到非常高的下载速率。

[0240] 根据实施例的所利用的下载流水线速率(DPR)是对当前状态(例如，块大小、并发度)下的可达到的饱和下载速率的估计。换句话说，DPR对以当前的TA 120设置利用大文件下载将达到的速率进行估计。

[0241] 为根据本文中的实施例计算DPR，将块请求分组到纪元(epoch)中。可以例如每当块请求被延迟时(例如，当自从TA 120变为对发送块请求准备就绪并且下一个块请求已发出起，大量时间已过去时)开始新的纪元。可以通过测量下载速率但将针对块或者子请求所下载的第一个数据之间的间隙从新纪元中减损来计算DPR。

[0242] 在CM 122处从RM 121接收的块请求被分类到纪元中。在根据实施例的操作中，针对每个“被延迟”的块请求开始新的纪元。在图8中示出了对将块请求分类到纪元中的说明，其中，块请求在以下条件中的一个或多个条件下通常被指定为迟的或者被延迟的：下载会话的第一个被发出的块请求；当从UA 129去往RM 121的请求是迟的时；在缓冲器的高/低水位耗尽情况下的按需流传送中；以及在直播流传送中，当分段在固定的时间轴上是可用的时。当纪元的第一个数据到达时，其被计数为“被延迟”的，并且被从DPR计算中减损。在图9中示出了DPR减损的一个示例，其中，由Xs指定的时间和数据被减损。通常，如图9中所示，针对HTTP请求所接收的响应数据的区块是TCP网络分组的净荷。然而，与上面描述的DCR技术相似，下面描述的DPR技术考虑了多于一个TCP网络分组可以被一次接收的可能性。

[0243] 可以根据下面描述的算法来计算DPR。应当认识到，关于DPR的计算所采取的方法与上面描述的对DCR的计算相似。如之前一样，让MSS是优选被设置为接近TCP分组的最大大小的常量(例如，MSS＝1500字节)。与上面使用的Cmult相似，假设Pmult是小于1的常量(例如，Pmult＝2/3)。Cmult和Pmult的值可以根据实施例被设置为相同的值或者不同的值。根据实施例的所采用的DPR计算算法取决于配置参数epoch_time_threshold，所述配置参数epoch_time_threshold是请求准备就绪与触发新纪元的请求发出之间的时间(以毫秒计)的最小量(例如，epoch_time_threshold＝20是用于该参数的合理设置)。实施例的DPR算法可以记录TDPR和ZDPR变量，其中，计时器TDPR对TA在下载数据上花费的时间的累积量(将纪元之间的间隙减损)进行计数，并且计数器ZDPR对被接收的字节的累积数量(将在纪元开始处被接收的最先的突发减损)进行计数。该算法可以也记录多个其它变量，所述其它变量诸如是current_epoch_id(即，被分配给新的块请求的当前的纪元号)、awaited_epoch_id(即，对于其来说最先被接收的数据净荷还未被减损的下一个纪元ID)、ready_flag(即，指示是否CM对接受新的请求准备就绪的标志)和last_ready_time(即，如果ready_flag被设置的话，指示当CM变为对接受新的块请求准备就绪时的时间点(如果ready_flag未被设置的话，该值不被使用))。

[0244] 在根据实施例的DPR计算算法启动时，可以如下地设置前述变量：

[0245] 设置T_{DPR}＝0

[0246] 设置Z_{DPR}＝0

[0247] 设置current_epoch_id＝0

[0248] 设置awaited_epoch_id＝0

[0249] 设置ready_flag＝True

[0250] 设置last_ready_time＝现在(以毫秒计)-epoch_time_threshold-1

[0251] 设置$T_{last_data_recv}$＝现在(以毫秒计)

[0252] 当块请求C从RM 121过来时,实施例的DPR计算算法如下地操作：

[0253] 设置delta＝现在(以毫秒计)-last_ready_time

[0254] 如果ready_flag＝＝True and delta>epoch_time_threshold,

[0255] 设置current_epoch_id＝current_epoch_id+1

[0256] C.epoch_id＝current_epoch_id

[0257] ready_flag＝False

[0258] 调用内部的准备就绪性更新算法

[0259] 块请求C可以然后被发给栈处理1025。

[0260] 当块请求D的数据到达时,实施例的DPR计算算法如下地操作：

[0261] 让current_time＝现在(以毫秒计)

[0262] 让deltaT＝current_time-$T_{last\_data\_recv}$(自从最后被接收的任何块的任何数据以来的时间)

[0263] $T_{last\_data\_recv}$＝current_time

[0264] 让deltaZ＝所接收的数据的量,包括HTTP报头开销

[0265] 如果D.epoch_id≥awaited_epoch_id,

[0266] //数据开始新纪元,减损。

[0267] 如果deltaZ>MSS则

[0268] T_{DPR}＝T_{DPR}+deltaT*Pmult*(deltaZ-MSS)/deltaZ

[0269] Z_{DPR}＝Z_{DPR}+deltaZ-MSS

[0270] 设置awaited_epoch_id＝D.epoch_id+1

[0271] 否则：

[0272] T_{DPR}＝T_{DPR}+deltaT

[0273] Z_{DPR}＝Z_{DPR}+deltaZ

[0274] 在根据实施例的操作中,deltaZ确实包括HTTP报头开销,并且对于DCR计算中的对应片段是同样的。不包括报头开销可能通常是重要的,因为报头可能是大到足以影响算法的(例如,如果块大小小的话)。

[0275] 在前述的描述内容中,被用于DCR的变量Tlast_data_recv与被用于DPR的变量是不同的。因为两者算法在相同时间点处被调用,所以公共的变量假如它不被每DPR和DCR调用更新两次的话则也可以被使用。

[0276] TDPR和ZDPR是单调递增的数。CM可以记录那些值的一些历史,并且因此,实施例的CM 122将这些变量存储在在本文中被称为mapTDPRZDPR的循环队列中。可以以与用于DCR的循环队列大致相同的方式来选择那个循环队列的大小。

[0277] 对于对示例性实施例进行的讨论的剩余部分,DPR(dcr_dpr_estimate_win)表示在前一个持续时间为dcr_dpr_estimate_win的窗口上被估计的DPR,其中,dcr_dpr_

estimate_win通常将是与用于DCR的持续时间相同的持续时间。

[0278] 实施例的CM 122保存针对T值的测量，T值是对活动块请求的聚合大小的近似。在根据实施例的操作中，请求被分块，并且CM 122基于它针对T值的当前测量来指示对发出新的块请求的准备就绪性。根据实施例的T值调整算法的目标在于，T应当是大得足以确保接近可能的最大下载速率的，同时受制于前述内容，T应当是尽可能小的。如果T值太小，则CM 122看到大于当当前的HTTP块请求完成时与当来自下一个HTTP块请求的第一个响应开始时之间的平均的“间隙”。这在图10A中被示出。如果T值是正好的，则对下一个HTTP块请求的第一个响应以与当前的HTTP块请求完成之后的块的全部其它分组相同的步调到来。这在图10B中被示出。如果T值太大，大量HTTP块请求被比必要的更快地提交。这在图10C中被示出。

[0279] 实施例的CM 122使用DPR和DCR的当前的值(在前一个持续时间为dcr_dpr_estimate_win的窗口上求平均)来定期地调整T值测量。下面的DCR与DPR之间的关系通常保持为：DCR通常是至少与DPR一样大的；当块的第一个数据区块与前一个被接收的数据区块之间存在大于平均的间隙(这暗示更多数据应当在链路上被请求，并且因此T应当被增大)时，DCR通常大于DPR；以及，当块的第一个数据区块与前一个被接收的数据区块之间存在平均的间隙(这暗示请求是足够早和大的，并且因此，T是足够大的并且可以被减小)时，DCR通常近似等于DPR。在图11中示出了当DCR＝DPR时的平均间隙和当DCR>DPR时的大于平均的间隙的概念。

[0280] 在根据实施例的操作中，T值调整算法使用前述的观察来定期地调整T值，并且维护用于给定的链路状况的最优的T。下面阐述了由实施例的CM122所使用的配置参数和T值调整算法：

[0281]

配置参数	值	意义
cm_target_delta	0.95	目标是设置 T，以使得 DPR = cm_target_delta * DCR
cm_step_alpha	0.25	步进衰减；用于将收敛速度减少因子 1/cm_step_alpha 以换取更好的稳定性。该参数已针对 $t_{MinTAdjInterval}$ 的当前缺省值被调谐，并且

[0282]

		如果后者被改变则可以改变。
cm_cp_ratio_max	2	在计算中 DCR/DPR 的最大允许比值
cm_cp_ratio_min	1	在计算中 DCR/DPR 的最小比值
$t_{MinTAdjInterval}$	500	连续的 T 值更新之间的最小时间间隔（以毫秒计）
C_{Init}	16384	以字节计的初始目标块大小（当 T 历史不可用时被使用）
C_{min}	16384	以字节计的最小目标块大小
T_{max}	3145728	以字节计的最大 T 值。（被设置为TA 120被配置为支持的最大带宽*RTT 的倍数（例如，2倍）——在这里被设置为近似支持 12 Mbps 的最大带宽和1 sec的最大RTT）
$smin_{Total}$	4	跨越全部起源服务器的最小目标请求并发度
$TA_{history\_filename}$	字符串	用于在启动时从其检索 T_{Init} 和在完成时向其存储 T 值的文件的文件名
dcr_dpr_estimate_win	2000	以毫秒计的在其上估计 DCR 和 DPR 测量的窗口

[0283] 在实施例的T值调整算法启动时，可以如下地初始化所述变量：

[0284] 设置T_{Init}=C_{Init}*$smin_{Total}$(或者如果可用的话从历史文件加载它)

[0285] last_tvalue_update_time=现在(以毫秒计)

[0286] 设置T_{min}=C_{min}*$smin_{Total}$

[0287] 设置w=dcr_dpr_estimate_win

[0288] 在根据实施例的操作中,每当数据被接收或者对于活动的CM发生100毫秒的超时(即,CM具有一些未完成的已请求的数据)时:

[0289] 如果DCR(w)是无效的或者DPR(w)是无效的

[0290] 返回

[0291] current_time=现在(以毫秒计)

[0292] 如果current_time-last_tvalue_update_time>$t_{MinTAdjInterval}$,

[0293] 如果DCR(w)>DPR(w)*cm_cp_ratio_max,

[0294] ratio=cm_cp_ratio_max

[0295] 否则如果DCR(w)<DPR(w)*cm_cp_ratio_min,

[0296] ratio=cm_cp_ratio_min

[0297] 否则:

[0298] ratio=DCR(w)/DPR(w)

[0299] z=1+cm_step_alpha*(cm_target_delta*ratio-1)

[0300] 设置T=min(T_{max},max(T*z,T_{min}))

[0301] last_tvalue_update_time=current_time

[0302] 以浮点算术根据实施例执行以上函数中的计算。在终止时,实施例的CM 122操作为,如果TAhistory_filename已通过配置被设置,则将T的最终值写入历史文件。

[0303] 在根据实施例的操作中,RM 121向CM 122查询针对起源服务器I的对发出新的块请求的准备就绪性。CM 122可以继而通过使用准备就绪性算法来确定它对于向起源服务器I发出新的块请求是否准备就绪。在内部,实施例的CM 122记录已向网络栈发出的针对不同起源服务器的块请求。CM可以使用该信息连同关于每起源服务器被网络栈允许的连接的最大数量的配置信息来调节和调度去往栈的块请求。CM可以还使用它的对T值的测量来确保在不过度地致力于比必要的请求得更快的情况下链路被用到它的最大容量。

[0304] 下面阐述了被实施例的CM 122使用的配置参数和准备就绪性算法:

[0305]

配置参数	值	意义
C_{min}	16384	以字节计的最小目标块大小
$smin_{Total}$	4	跨越全部起源服务器的最小目标请求并发度
$smax_{Total}$	256	被网络栈允许的跨越全部起源服务器的最大请求并发度
$smax_{Org}$	12	单个起源服务器的最大请求并发度（对于在Chromium 内被配置的经加速的请求，被设置为每起源的最大套接字池）
ready_threshold	250	准备就绪性门限常量
epoch_time_threshold	20	请求准备就绪与触发新的纪元的请求发出之间的时间的最小量（以毫秒计）

[0306] 当由RM 121向CM查询对于特定起源服务器的准备就绪性时，实施例的CM 122可以执行以下操作：

[0307] Q_I＝去往起源服务器I的活动块请求的当前数量

[0308] Q＝去往全部起源服务器的活动块请求的当前数量＝$sum_I(Q_I)$

[0309] R_{Avg}＝接收来自全部起源服务器的块请求的平均当前数量(使用R_{Avg}计算算法)。

[0310] //找到Qallow，其是

[0311] //max{Z:T>＝Cmin*(Z+1)

[0312] //and(ready_threshold*Z<＝1000*R_{Avg} or Z<$smin_{Total}$)

[0313] //and Z-Q+QI<$smax_{Org}$

[0314] //and Z<$smax_{Total}$}

[0315] bound1＝floor(T/C_{min})-1

[0316] bound2＝max(floor(1000*R_{avg}/ready_threshold),$smin_{Total}$-1)

[0317] bound3＝$smax_{Org}$+Q-Q_I-1

[0318] bound4＝$smax_{Total}$-1

[0319] $Qallow_I = min\{bound1, bound2, bound3, bound4\}$

[0320] 在$Qallow_I >= Q$的情况下，CM对去往任何起源服务器I的块请求准备就绪

[0321] 为理解根据实施例被实现的准备就绪性算法，让deltaI是可以可能对于特定起源服务器I被发出的额外块请求。则QallowI(其针对准备就绪性被计算)是如果我们将成功发出deltaI个块请求的情况下系统中的活动块请求的总数(即，QallowI＝deltaI+Q)。

[0322] 在根据实施例的操作中，CM 122保存对接收数据的活动块请求的平均数量的测量(R)以便在准备就绪性算法中使用。下面阐述了用于由实施例的CM 122计算RAvg的配置参数和算法：

[0323]

配置参数	值	意义
receive_req_avg_const	0.25	在接收请求求平均算法中被使用的求平均常量
$t_{MinRAvgAdjInterval}$	500	连续的 R_{Avg} 值更新之间的最小时间间隔（以毫秒计）
$TA_{history\_filename}$	字符串	用于在启动时从其检索 R_{Avg}、在完成时向其存储 R_{Avg} 的文件的文件名

[0324] 在用于计算RAvg的算法启动时，可以根据实施例如下地初始化所述变量：

[0325] 设置R_{Avg}＝0(或者如果可用的话从历史文件加载它)

[0326] last_Ravg_value_update_time＝现在(以毫秒计)

[0327] 在根据实施例的操作中，每当数据被接收或者对于活动的CM 122发生100毫秒的超时(即，CM 122具有一些未完成的已请求的数据)发生时：

[0328] current_time＝现在(以毫秒计)

[0329] R＝跨越全部起源服务器的接收数据的活动块请求的当前数量

[0330] 如果current_time-last_Ravg_value_update_time>$t_{MinRAvgAdjInterval}$，

[0331] 如果$R > R_{Avg}$，

[0332] $R_{Avg} = R$

[0333] 否则：

[0334] R_{Avg}＝((1-receive_req_avg_const)*RAvg)+

[0335] (receive_req_avg_const*R)

[0336] last_Ravg_value_update_time＝current_time

[0337] 在终止时，如果TAhistory_filename已通过配置被设置，则实施例的CM 122可以将RAvg写入历史文件。

[0338] 根据实施例被实现的内部的准备就绪性算法的目的在于，更新CM 122内的ready_flag和last_ready_time。这些参数可以被CM 122用于将到来的请求分类到前述的纪元中以便进行DPR计算。

[0339] 在根据实施例的操作中，该内部准备就绪性算法由CM 122在块请求被从RM 121接收之后和用于已被发出的块请求中的任何块请求的数据被从栈接收之后立即执行。以下内容提供了根据本文中的实施例被采用的内部准备就绪性更新算法：

[0340] 如果ready_flag＝＝True，

[0341] 返回//如果准备就绪性已被建立则跳过

[0342] 让A是活动的起源服务器(具有至少一个活动的块请求的服务器)的列表

[0343] 对于A中的i：

[0344] 针对起源服务器i确定准备就绪性

[0345] 如果CM对于起源服务器i准备就绪

[0346] ready_flag＝True

[0347] last_ready_time＝现在(以毫秒计)

[0348] 在查询对于特定起源的请求准备就绪性之后并且在将请求分块之前，实施例的RM 121向CM 122查询对于特定起源服务器的块请求的优选块大小。在根据实施例的操作中，CM 122使用优选块大小计算算法来向RM 121传送对于起源服务器请求的优选块大小。在计算优选块大小时可以利用以下参数。

[0349]

配置参数	值	意义
C_{min}	16384	以字节计的最小目标块大小

[0350] 当查询对于起源服务器I的优选块大小时，实施例的CM 122可以如下地操作：

[0351] 计算$Qallow_I$

[0352] T＝当前的T值

[0353] $C_{Target(I)}=\max(T/(Qallow_I+1),C_{min})$

[0354] 在根据实施例的操作中，RM 121使用来自CM 122的优选块大小(CTarget(I))值来将来自UA 129的针对起源服务器I的下一个被调度的请求分块，以便向CM 122发出。相应地，RM可以使用分块算法来提供对请求的分块。由实施例的RM 121使用的分块算法的目标包括：使块大小近似是RM 121从CM 122取回的CTarget(I)值；展宽不同块请求的块大小，以避免请求完成时间的同步(例如，可以与优选块大小CTarget(I)成比例地展宽块大小)；和/或避免作出非常小的块请求。下面阐述了由实施例的RM 121用于分块的配置参数：

[0355]

配置参数	值	意义
chunk_spread	0.5	块展宽。分块算法试图使块大小在范围 $C_{Target(I)}$ 至 $C_{Target(I)}$ *(1 + chunk_spread)中
chunk_hysteresis	1.5	滞后。相对于小的剩余请求，优选以因子 chunk_hysteresis 来增加块大小

[0356] 在根据实施例的操作中，RM 121还使用在启动时被初始化为0的(例如，在TA 120的运行时期间)持续的计数器ctr。计数器ctr可以最方便地被实现为32比特无符号整数。

[0357] N＝将被分块的请求中的剩余的未被请求的字节，或者对于开放端点的字节范围请求是None

[0358] 计算p＝ceil($C_{Target(I)}$*(1+chunk_spread*(reverse_bits(ctr)/2**32)))

[0359] //应当指出：reverse_bits()将字中的比特倒转

[0360] //见下面。

[0361] 如果N不是None，

[0362] 如果chunk_hysteresis*p<N，

[0363] 设置ChunkSize＝p

[0364] 或者：

[0365] 设置ChunkSize＝N

[0366] 或者：

[0367] 设置ChunkSize＝p

[0368] ctr＝(ctr+1)%(2**32)

[0369] 返回ChunkSize

[0370] reverse_bits()函数将32比特无符号整数中的比特倒转，由此，最有效的比特变成最无效的比特，第二有效的比特变成第二无效的比特，等等。该函数具有用于在从0到2^32-1的范围中展宽整数的属性，并且确实展宽具有比随机展宽将具有的距离大的距离的短序列。作为参考，可以如下地实现该函数：

```
uint32_t reverse_bits(uint32_t x)
{
#define flip(x, mask, shift)  \
    ((((x) & mask) << shift) ^ (((x) & ~mask) >> shift))
    x = flip(x, 0x55555555, 1);
    x = flip(x, 0x33333333, 2);
    x = flip(x, 0x0f0f0f0f, 4);
    x = flip(x, 0x00ff00ff, 8);
    x = flip(x, 0x0000ffff, 16);
    return x;
#undef flip
}
```

[0371]

[0372]

[0373] 在根据实施例的操作中，从用户代理(包括浏览器)接收的HTTP请求在RM 121中被拆分，并且块请求被发送给CM 122。实施例的CM 122决定何时新的块请求可以被发送以及哪些起源服务器可用于处理新的块请求。实施例的RM 121在以下条件下获得调度用于向CM 122发出的块请求的机会：每当RM 121从UA 129接收新的数据下载请求时；每当针对已被发出的块请求的数据被接收时；以及每当在CM 122可能准备好发出对于相同或者不同起源服务器的更多请求时RM 121成功地向CM 122发出块请求时。在根据实施例的操作中，RM 121选择来自UA 129的下一个请求以拆分并向CM 122移交块请求，由此RM可以使用请求调度算法来达到这个目的。实施例的请求调度算法的目的在于：确保没有任何来自UA 129的连接在TA 120存在时是急需的(如果在TA 120不存在时它将不是急需的的话)；实现跨连接(UA 129与代理服务器之间的连接)或者跨起源服务器的公平性；以及，是连续工作的(例如，如果至少一个流可以在调度机会处被提供服务，则至少一个请求将被提供服务)。可以由实施例的RM 121跨不同查询地使用简单轮询(round robin)调度。实施例的请求调度器记录最后被提供服务的队列，并且当被给与调度另一个块请求的机会时，它挑选来自下一个队列的开头的请求，并且测试CM 122对于起源服务器的针对该请求的准备就绪性。如果需要，还可以使用由RM 121使用的更复杂的调度算法。

[0374] 在启动时，可以如下地初始化由请求调度器使用的参数：

[0375] 设置current_queue_id＝0//记录最后被提供服务的队列

[0376] 当被给与调度与CM 122的请求的机会时，实施例的RM 121循环地执行以下操作，直到返回false为止：

[0377]

```
// *** 步骤 1：获得活动队列的已排序的列表
// 创建具有未完成的需要被调度的请求的队列的队列 id 的列表
ordered_queues = sort (get_active_queues())
如果 len(ordered_queues) == 0
```

```
        返回 false    //没有要调度的请求

    // ***步骤 2：使队列列表是按照处理次序的
    //    （即，将针对请求检查按照该次序的队列）
    // 找到进入 ordered_queues 中的最小下标 i，以使得
    // ordered_queues[i] >= current_queue_id; 或者设置
    // i := len(ordered_queues) 如果不存在这样的 i
    for i = 0, …, len(ordered_queues):
        if ordered_queues[i] >= current_queue_id,
            break
```

[0378]

```
    //按照处理次序来布置 ordered_queues
    ordered_queues = ordered_queues[i:] + ordered_queues[:i]

    // *** 步骤 3：找到下一个要发出的请求
    // 针对要发出的请求，检查按照处理次序的队列
    对于按照 ordered_queues 的 q
        如果 CM 准备好在队列 q 的开头处发出块请求 r，
            向 CM 发出块请求 r
            current_queue_id = q + 1
            返回 true
    返回 false
```

[0379] 实施例的RM 121内的重新排序层194的目的在于，向上面的部件提供对连续的有序数据的传递。由于从UA 129进来的数据下载请求被分块成跨与起源服务器的多个连接来发送的多个块请求，所以由RM 121接收回的数据可能不是有序的。在根据实施例的操作中，

重新排序层194对在RM 121内被内部地接收的数据进行缓冲，并且仅向上面的层/部件传递连续的数据。实施例的重新排序层194对每个队列进行缓冲，以使得数据可以被有序地传递。然而，根据本文中的实施例，一个队列上的缺失的数据不应当阻塞另一个队列。在图12中示出了用于重新排序层194的逻辑单元。

[0380] 在图13A、13B和13C中示出了根据前述的示例性实施例的RM 121与CM 122之间的算法执行的高层调用流。特别地说，图13A示出了根据本文中的实施例的启动和结束调用流。图13B示出了根据本文中的实施例的当经加速的请求被从传输加速器分派器接收时的调用流。图13C示出了根据本文中的实施例的当块数据从栈被接收或者超时时的调用流。

[0381] 图14A、14B和14C示出了使用smin＝4和smaxOrg＝20针对使用本文中的实施例的TA 120连接到单个起源服务器的单个UA而生成的图表。在图14A中所示的仿真中，可用带宽是6656千比特每秒，分组丢失率是0.1％，并且往返时间是50毫秒。在该仿真中，T的值平均上是近似80千字节，所使用的TCP连接的平均数量近似是4，以及所达到的呈现速率是近似5000千比特，这是在可用带宽以下可得的最高可能呈现速率。在图14B中所示的仿真中，可用带宽是6656千比特每秒，分组丢失率是1％，并且往返时间是50毫秒。在该仿真中，T的值平均上是近似120千字节，所使用的TCP连接的平均数量近似是7或者8，以及所达到的呈现速率是近似5000千比特，这是在可用带宽以下可得的最高可能呈现速率。因此，将图14B的仿真与图14A的仿真进行比较，在具有相同的可用带宽和往返时间，但在图14B中具有比图14A中高的丢失的情况下，所使用的连接的平均数量更高，并且T的值更高，但仍然是合理的，并且仍然达到最高呈现速率。在图14C中所示的仿真中，可用带宽是12288千比特每秒，分组丢失率是1％，以及往返时间是200毫秒。在该仿真中，T的值平均上是近似1兆字节，所使用的TCP连接的平均数量是近似20，以及所达到的呈现速率是近似9500千比特，这是在可用带宽以下可得的最高可能呈现速率。因此，将图14C的仿真与图14B的仿真进行比较，在具有两倍的可用带宽和四倍的往返时间，并且具有相同的丢失率的丢失的情况下，所使用的连接的平均数量更高，并且T的值与往返时间和可用带宽的增加的乘积成比例地更高，并且仍然达到最高呈现速率。

[0382] 应当认识到，可以修改或者适配传输加速器的操作以用于关于特定接口来使用。例如，根据本文中的概念实现的CM的实施例可以操作为，在知道瓶颈通常是无线接入网的情况下，当网络接口是3G/4G/LTE时是关于块请求非常积极的，所述无线接入网由PFAIR(成比例的公平性)排队策略支配(其将不对使用网络的其它用户装备(UE)有害)。对应地，当网络接口是通过共享WiFi公共接入网的时，实施例可以实现较不积极的CM，所述共享WiFi公共接入网使用将潜在对使用网络的其它较不积极的UE有害的FIFO排队策略。如与通过与内容服务器的网络连接被获得相对的，在数据从本地存储器被访问(例如，如可以已从较早的广播被加入队列的)的情况下，传输加速器的实施例可以实现适于访问来自本地高速缓存的数据的CM，所述本地高速缓存是与关于网络连接所使用的设计非常不同的设计。

[0383] 尽管图1A中所示出的实施例的RM 121被示为被与CM 122的单个实例接合，但如图6的实施例中所示的，某些实施例的RM 121可以与多于一个这样的CM接合。这样的CM可以例如支持不同的网络接口(例如，CM122a可以具有至设备上高速缓存622的本地接口，CM 122b可以使用至WiFi网络接口的HTTP/TCP连接，以及CM 122c可以使用至4G/LTE网络接口的HTTP/TCP连接，等等)。额外地或者替代地，这样的CM可以提供本质上相似的网络接口(例

如,不同的WiFi链路)。RM 121可以因此并发地与多个CM接合,由此,RM 121可以例如可操作为从多个CM请求相同分段或者分段的序列的数据块(例如,数据请求的一部分被发送到使用至4G/LTE接口的HTTP/TCP连接的第一CM-xTCP,并且数据请求的一部分被发送到使用至WiFi接口的HTTP/TCP连接的第二CM-mHTTP)。RM可以聚合从所述CM中的每个CM接收的数据,以重构由UA请求的分段,并且将响应提供回UA。

[0384] 额外地或者替代地,尽管图1A中所示出的实施例的CM 122被示为被与RM 121的单个实例接合,但某些实施例的CM 122可以并发地与多于一个这样的RM接合。例如,多个RM(每个RM针对客户端设备110的不同UA的)可以适于使用相同的CM或多个CM,由此,CM可以适于解决任何由于RM的并发操作产生的对连接的竞争。替代地,单个RM可以适于关于多个UA使用,由此,RM适于解决任何由于UA的并发操作产生的对连接的竞争。

[0385] 尽管已详细示出和描述了所选择的方面,但应当理解,可以在其中作出各种替换和改变,而不会脱离如由下面的权利要求书定义的本发明的精神和范围。

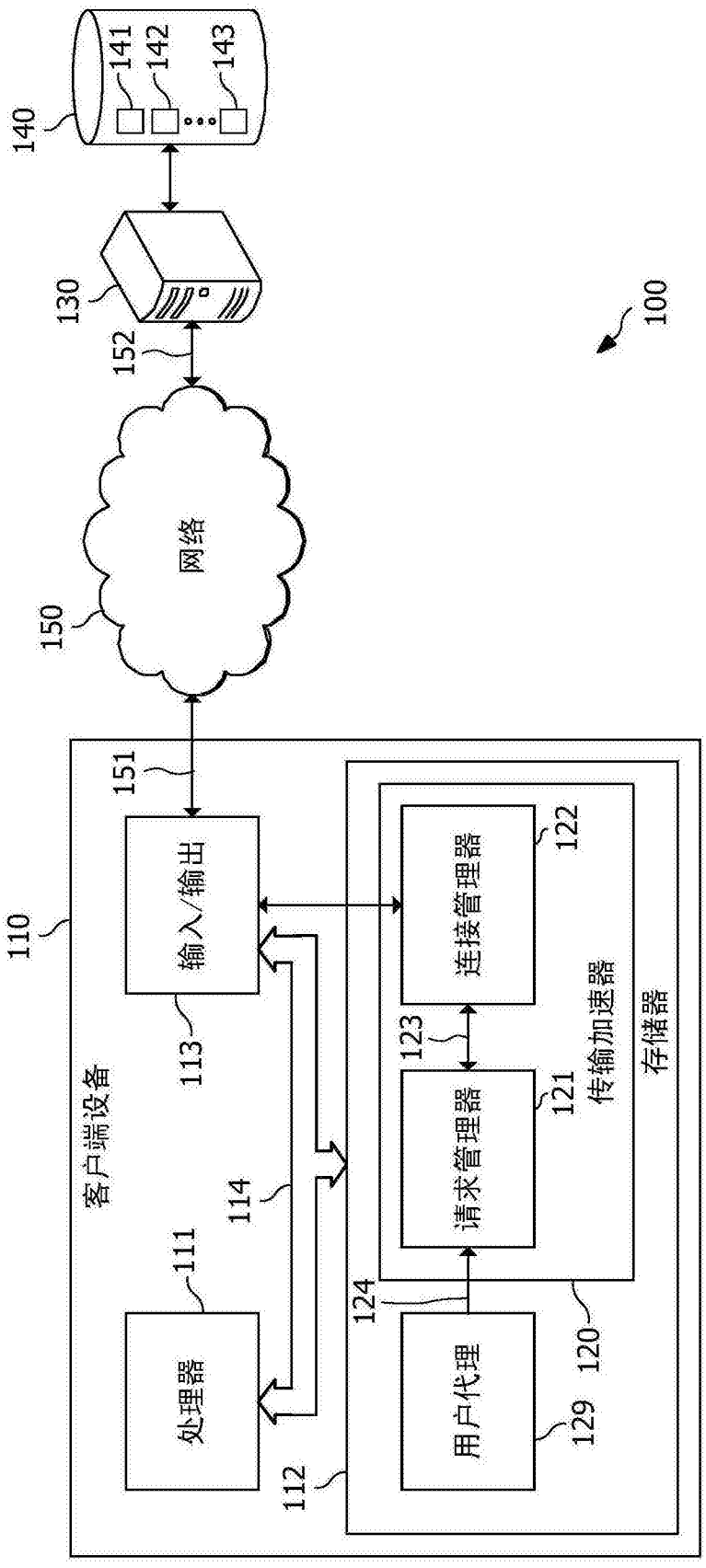

140
141
142
143
130
152
100
网络
150
151
110
输入/输出
113
客户端设备
114
111
处理器
112
连接管理器
122
123
请求管理器
121
传输加速器
存储器
124
用户代理
129
120

图1A

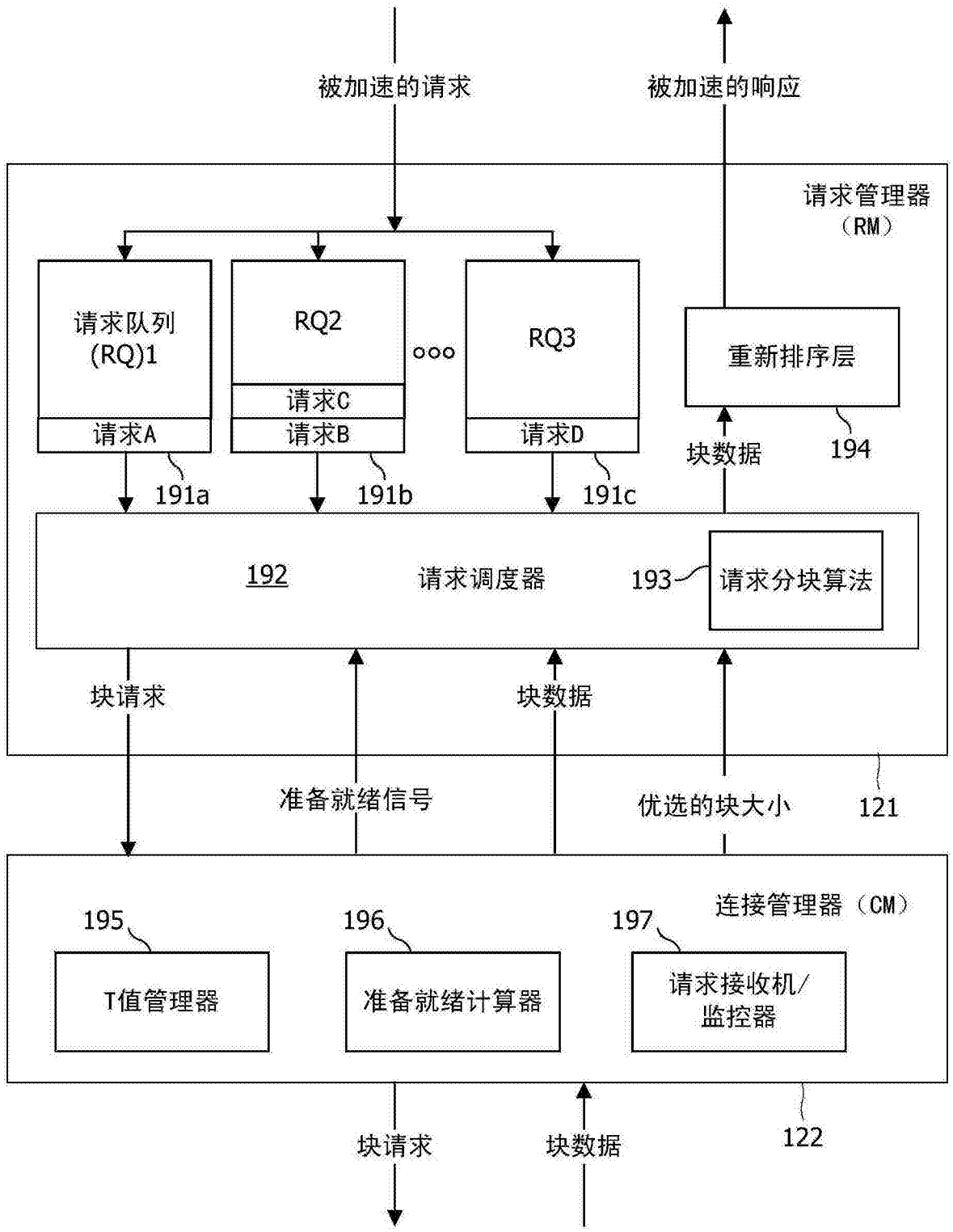

被加速的请求
被加速的响应
请求管理器
（RM）
请求队列
(RQ)1
RQ2
RQ3
请求C
请求A
请求B
请求D
重新排序层
块数据
194
191a
191b
191c
192
请求调度器
193
请求分块算法
块请求
块数据
准备就绪信号
优选的块大小
121
连接管理器（CM）
195
196
197
T值管理器
准备就绪计算器
请求接收机/
监控器
块请求
块数据
122

图1B

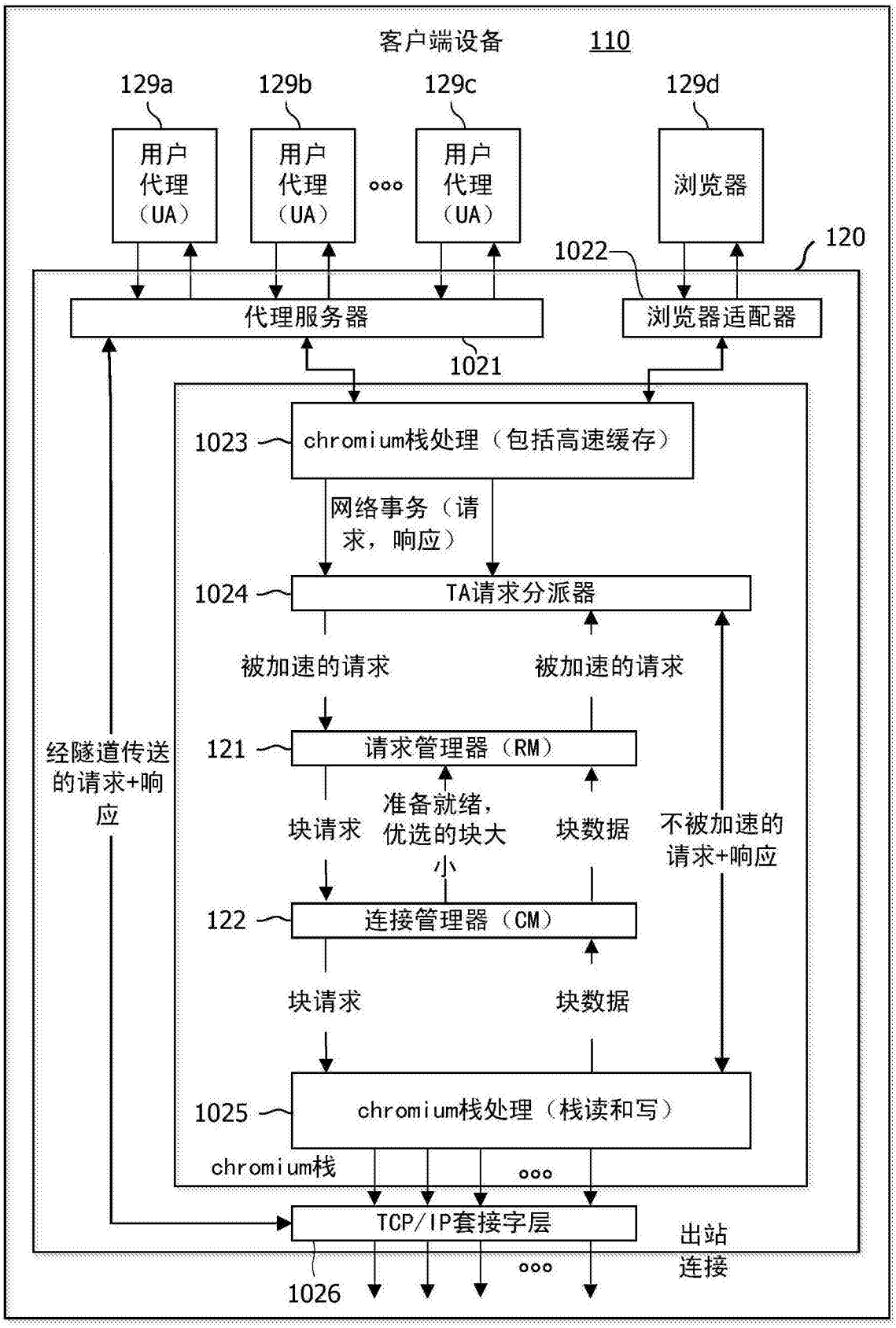

客户端设备 110
129a
129b
129c
129d
用户代理（UA）
用户代理（UA）
用户代理（UA）
浏览器
120
1022
代理服务器
浏览器适配器
1021
1023
chromium栈处理（包括高速缓存）
网络事务（请求，响应）
1024
TA请求分派器
被加速的请求
被加速的请求
121
请求管理器（RM）
经隧道传送的请求+响应
块请求
准备就绪，优选的块大小
块数据
不被加速的请求+响应
122
连接管理器（CM）
块请求
块数据
1025
chromium栈处理（栈读和写）
chromium栈
TCP/IP套接字层
出站连接
1026

图1C

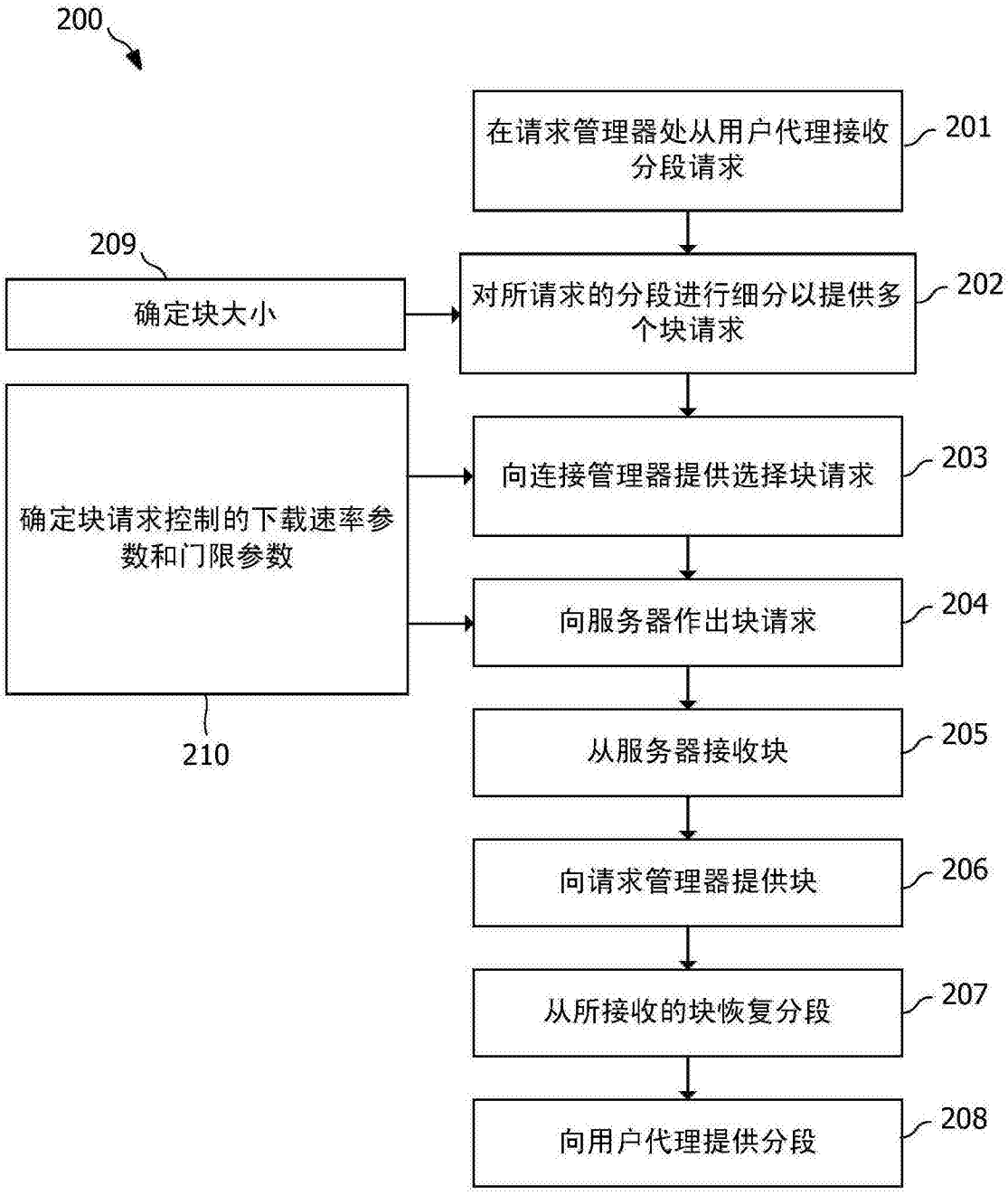

200
在请求管理器处从用户代理接收分段请求
201
209
确定块大小
对所请求的分段进行细分以提供多个块请求
202
确定块请求控制的下载速率参数和门限参数
向连接管理器提供选择块请求
203
向服务器作出块请求
204
210
从服务器接收块
205
向请求管理器提供块
206
从所接收的块恢复分段
207
向用户代理提供分段
208

图2

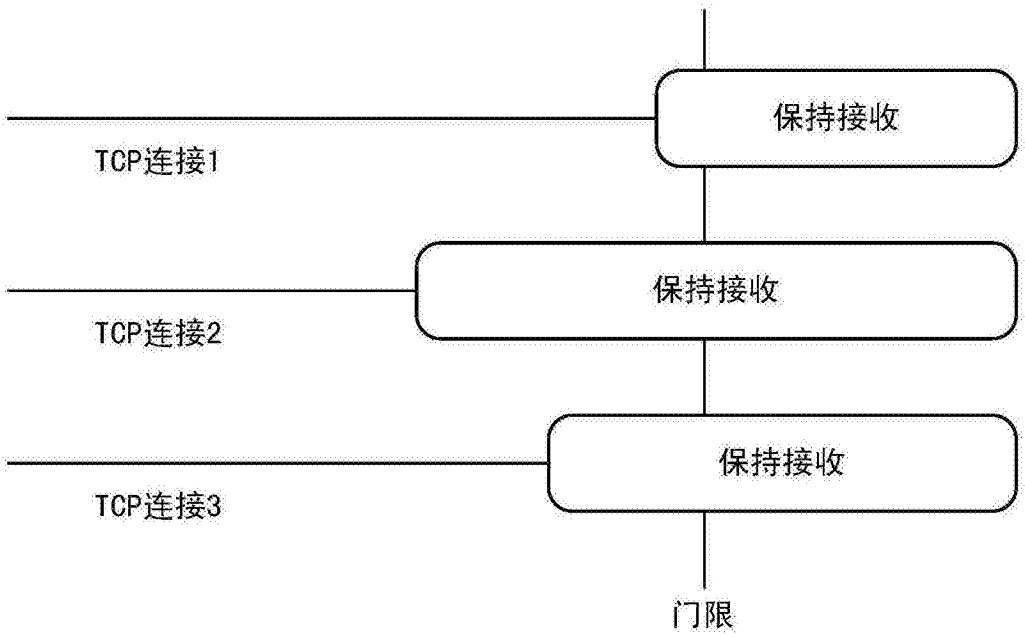

保持接收
TCP连接1
保持接收
TCP连接2
保持接收
TCP连接3
门限

图3

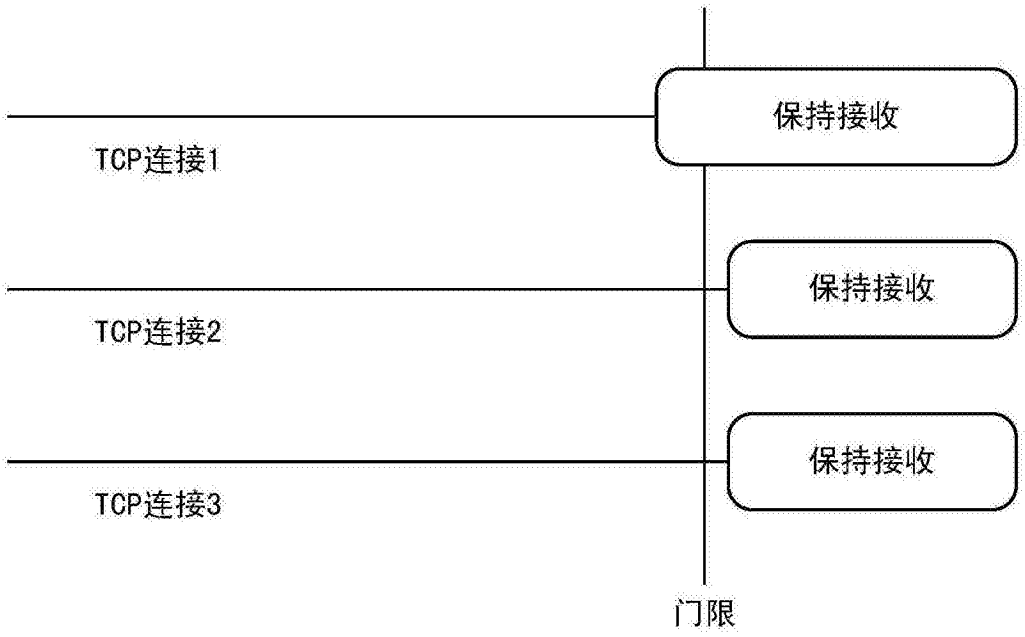

保持接收
TCP连接1
保持接收
TCP连接2
保持接收
TCP连接3
门限

图4

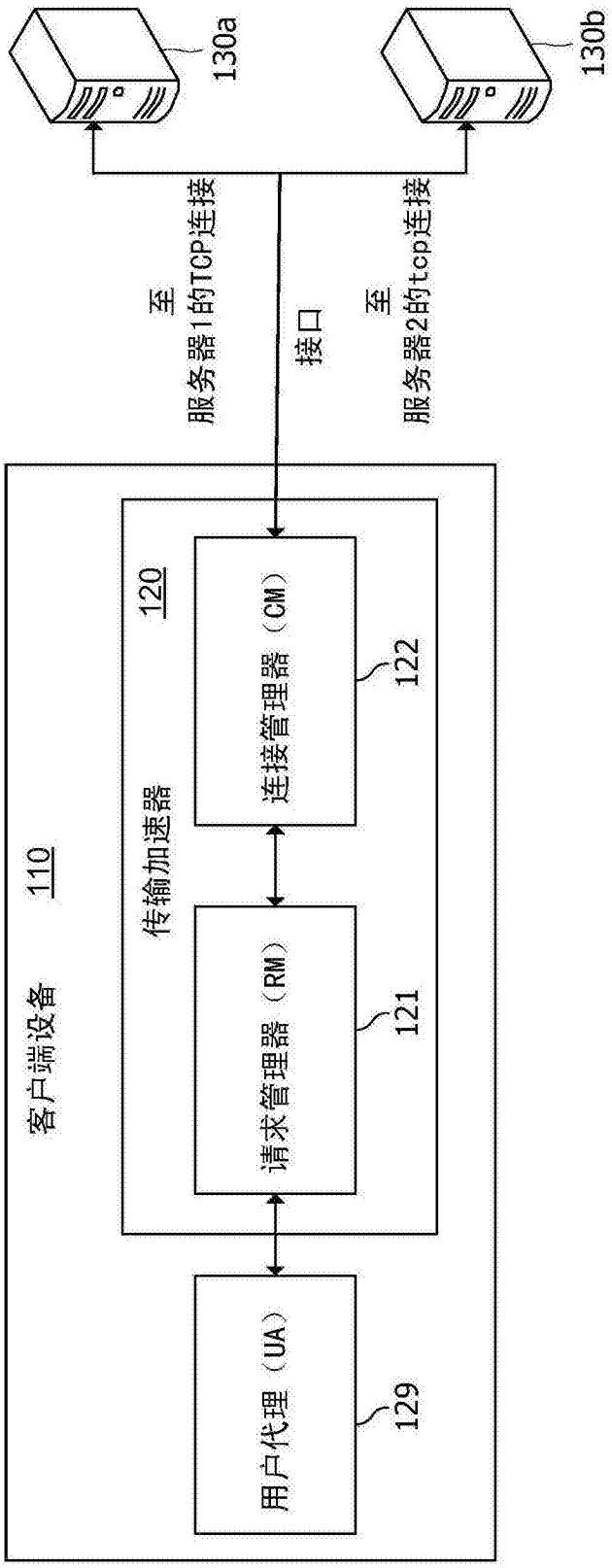

130a
130b
至
服务器1的TCP连接
接口
至
服务器2的tcp连接
客户端设备 110
传输加速器 120
连接管理器（CM）
122
请求管理器（RM）
121
用户代理（UA）
129

图5

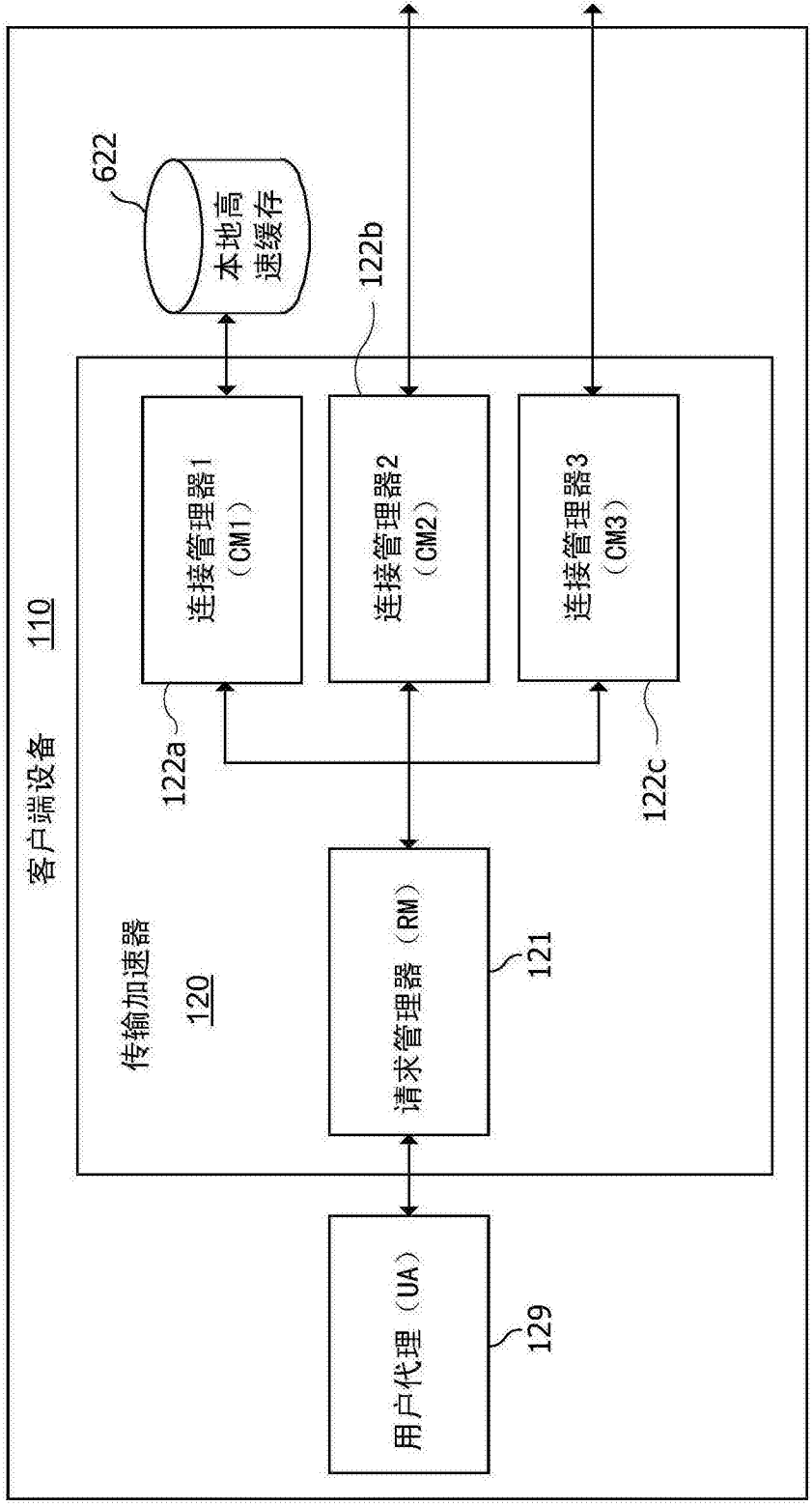

WiFi
4G/LTE
622
本地高速缓存
122b
客户端设备 110
连接管理器1（CM1）
连接管理器2（CM2）
连接管理器3（CM3）
122a
122c
传输加速器 120
请求管理器（RM）
121
用户代理（UA）
129

图6

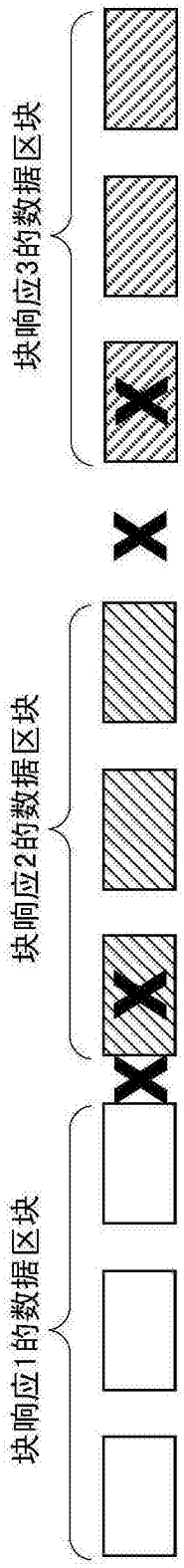

块响应3的数据区块
块响应2的数据区块
块响应1的数据区块

图7

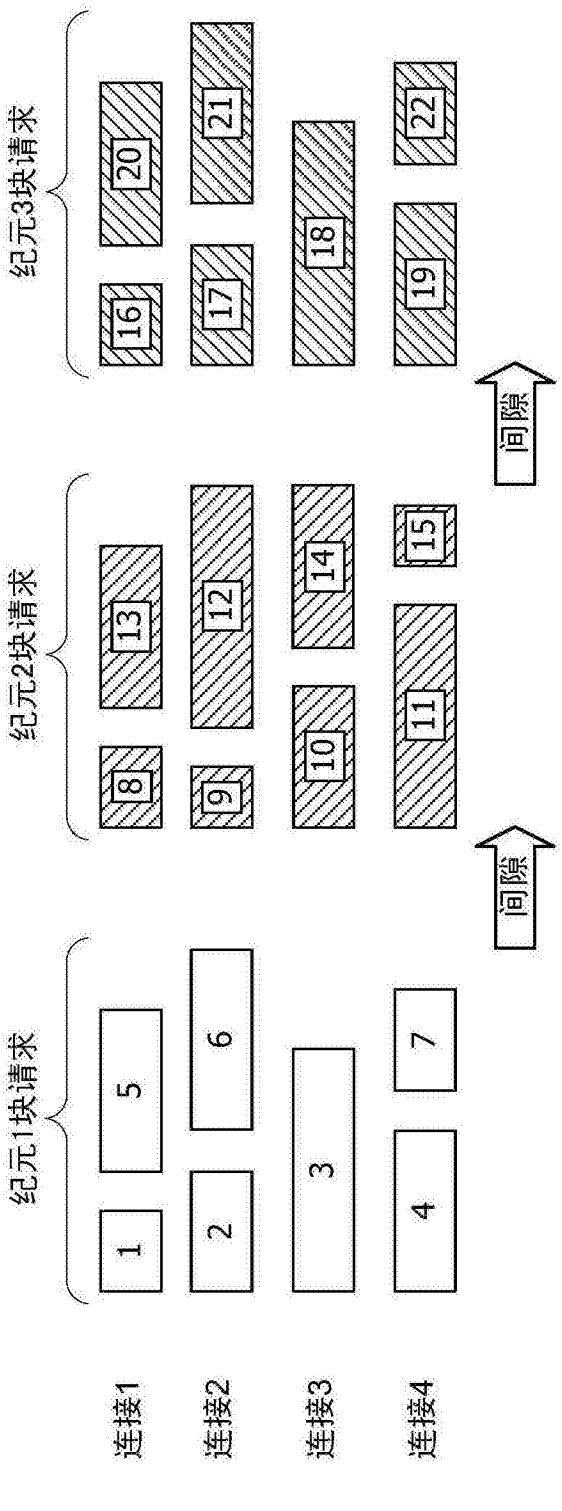

纪元1块请求
纪元2块请求
纪元3块请求
连接1
连接2
连接3
连接4
间隙
间隙
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

图8

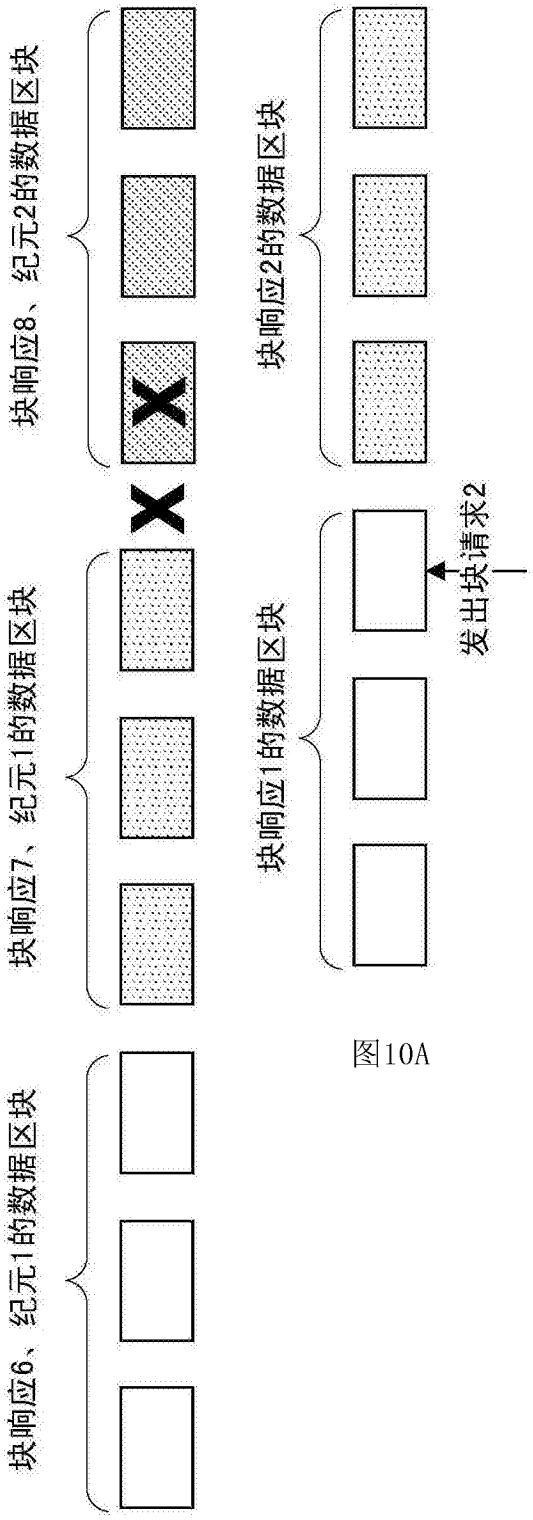

块响应6、纪元1的数据区块
块响应7、纪元1的数据区块
块响应8、纪元2的数据区块
块响应1的数据区块
块响应2的数据区块
发出块请求2

图9

图10A

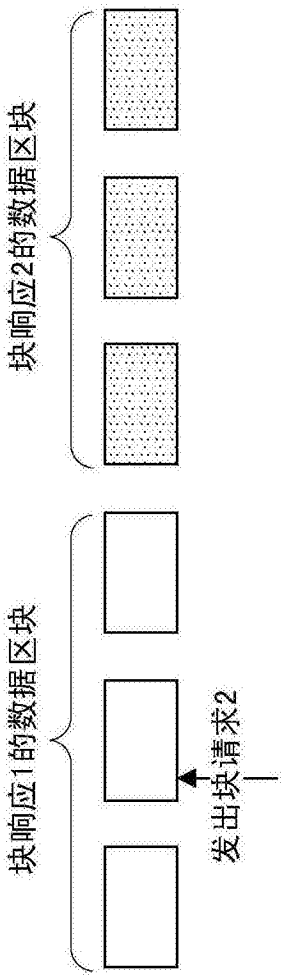


图10B

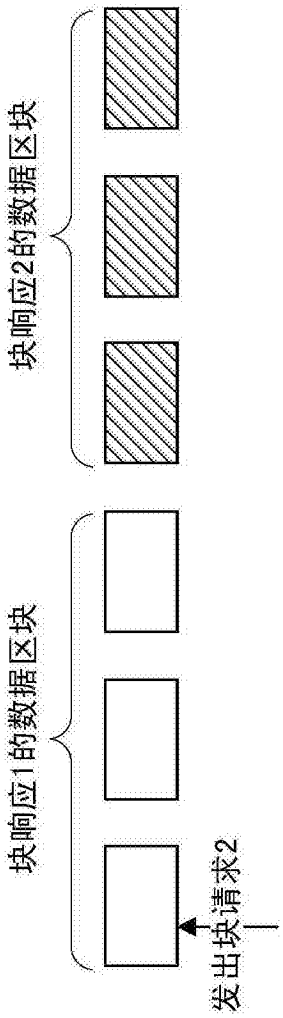


图10C

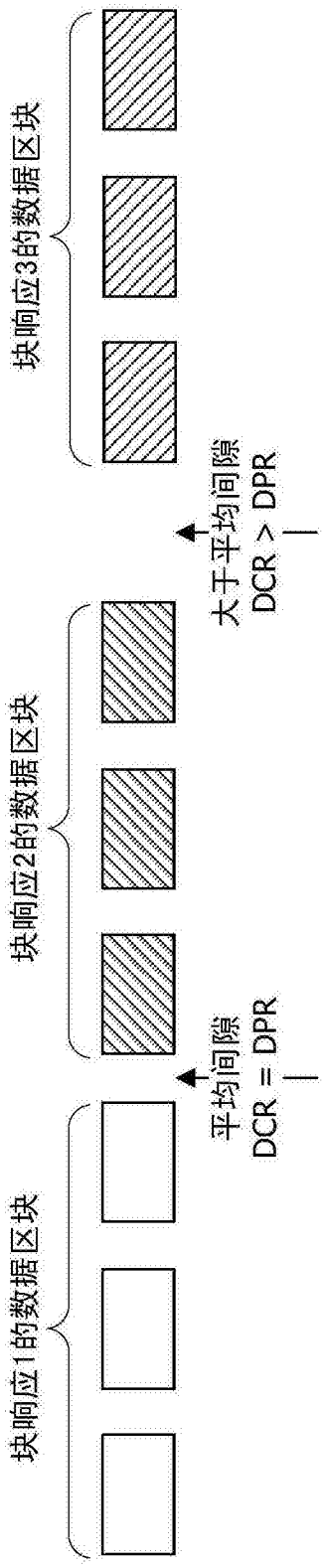


图11

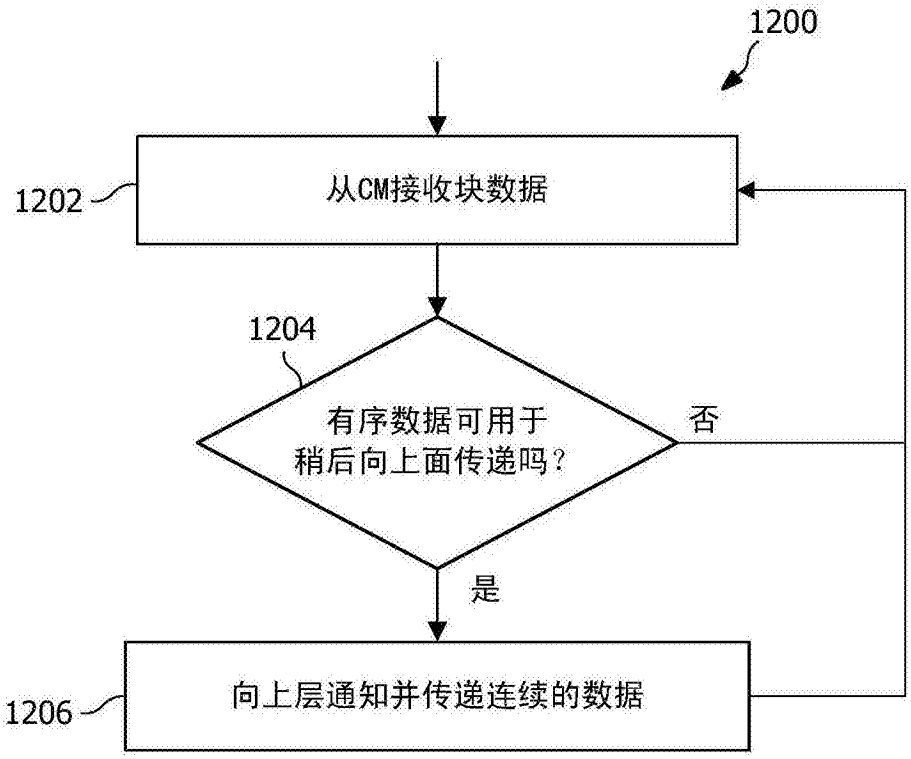

1200
1202
从CM接收块数据
1204
有序数据可用于
稍后向上面传递吗？
否
是
1206
向上层通知并传递连续的数据

图12

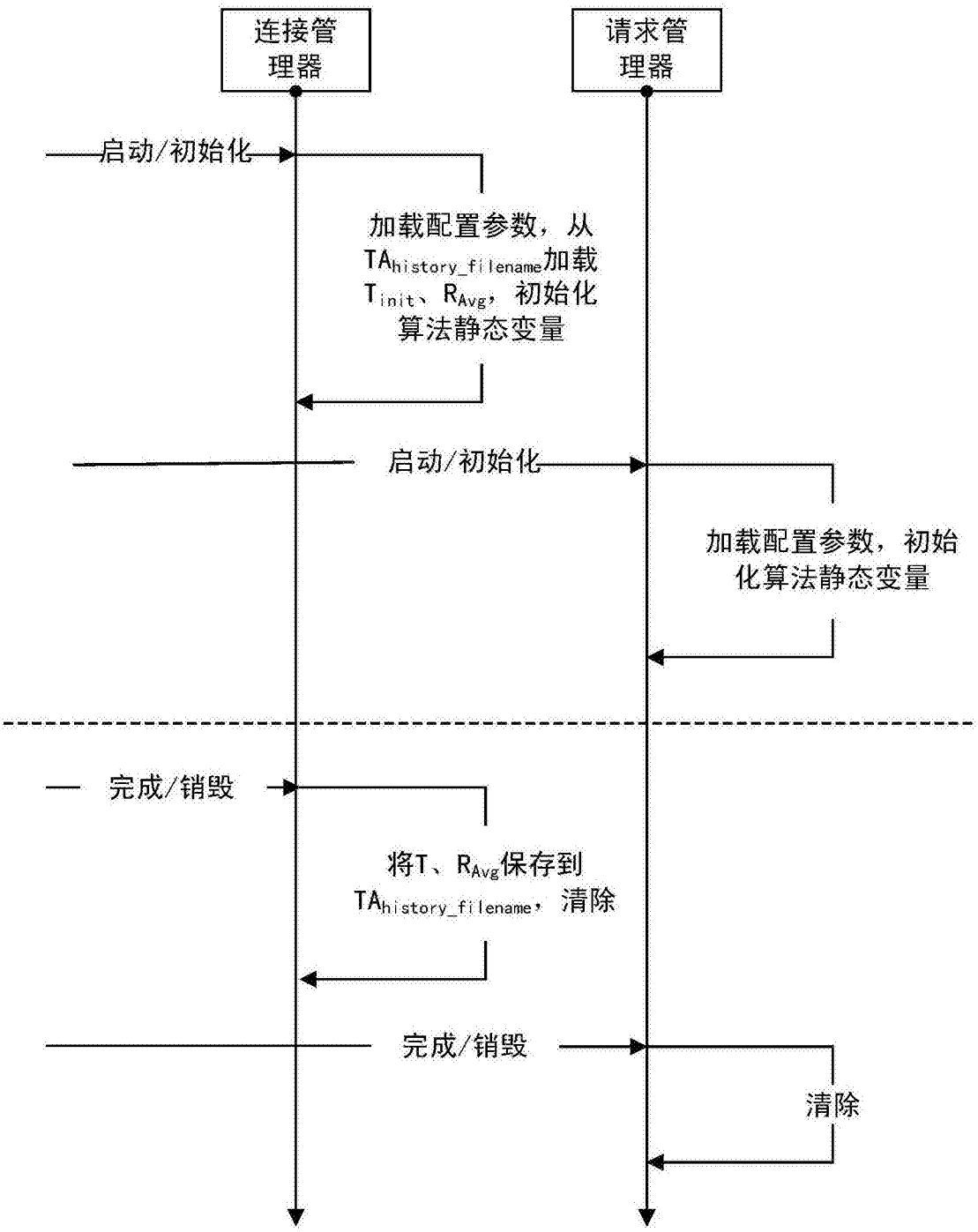

连接管理器
请求管理器
启动/初始化
加载配置参数，从 $TA_{history_filename}$加载 T_{init}、R_{Avg}，初始化算法静态变量
启动/初始化
加载配置参数，初始化算法静态变量
完成/销毁
将T、R_{Avg}保存到 $TA_{history\_filename}$，清除
完成/销毁
清除

图13A

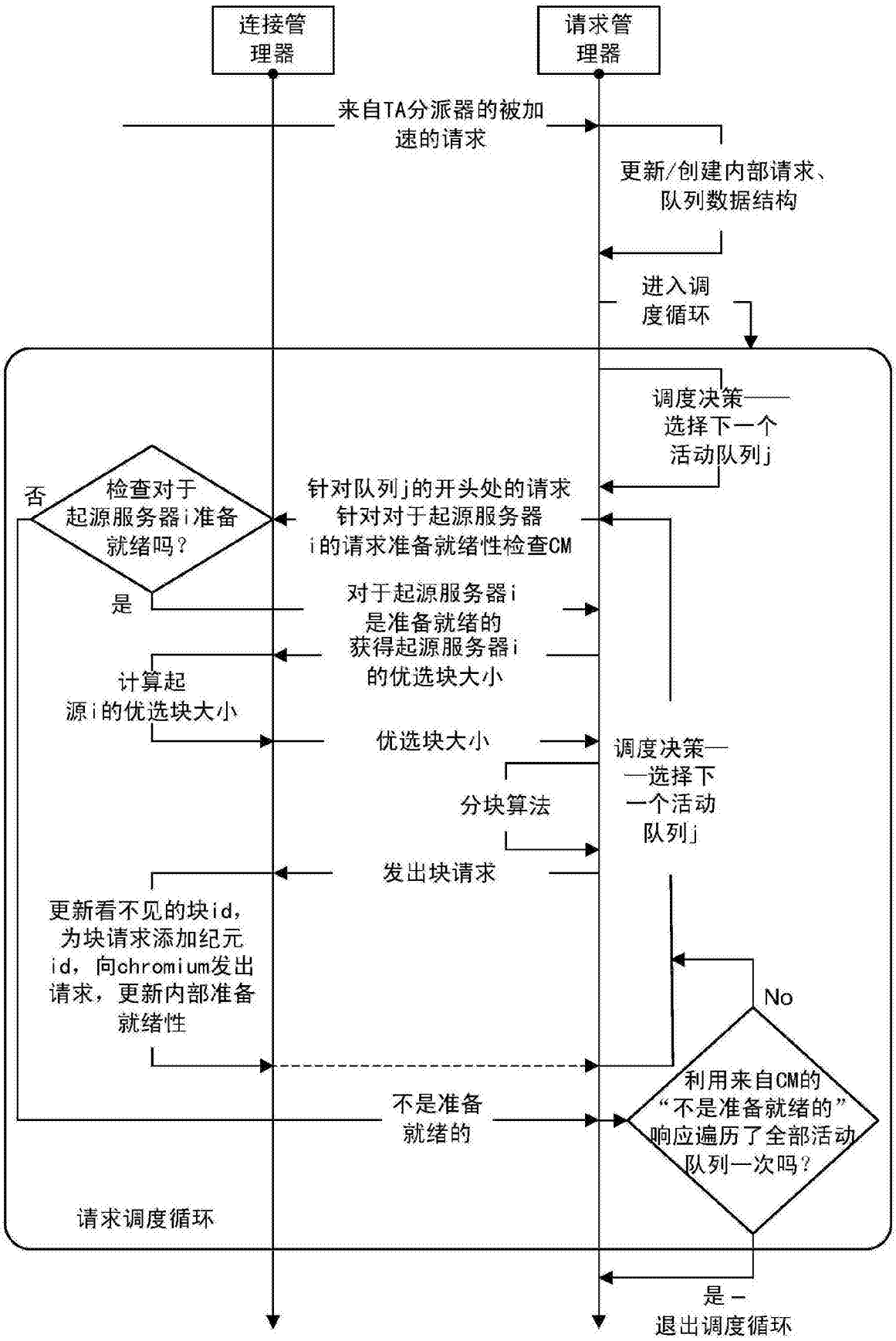

连接管理器
请求管理器
来自TA分派器的被加速的请求
更新/创建内部请求、队列数据结构
进入调度循环
调度决策——选择下一个活动队列j
检查对于起源服务器i准备就绪吗？
否
针对队列j的开头处的请求针对对于起源服务器i的请求准备就绪性检查CM
是
对于起源服务器i是准备就绪的
获得起源服务器i的优选块大小
计算起源i的优选块大小
优选块大小
调度决策——选择下一个活动队列j
分块算法
发出块请求
更新看不见的块id，为块请求添加纪元id，向chromium发出请求，更新内部准备就绪性
No
不是准备就绪的
利用来自CM的“不是准备就绪的”响应遍历了全部活动队列一次吗？
请求调度循环
是 –
退出调度循环

图13B

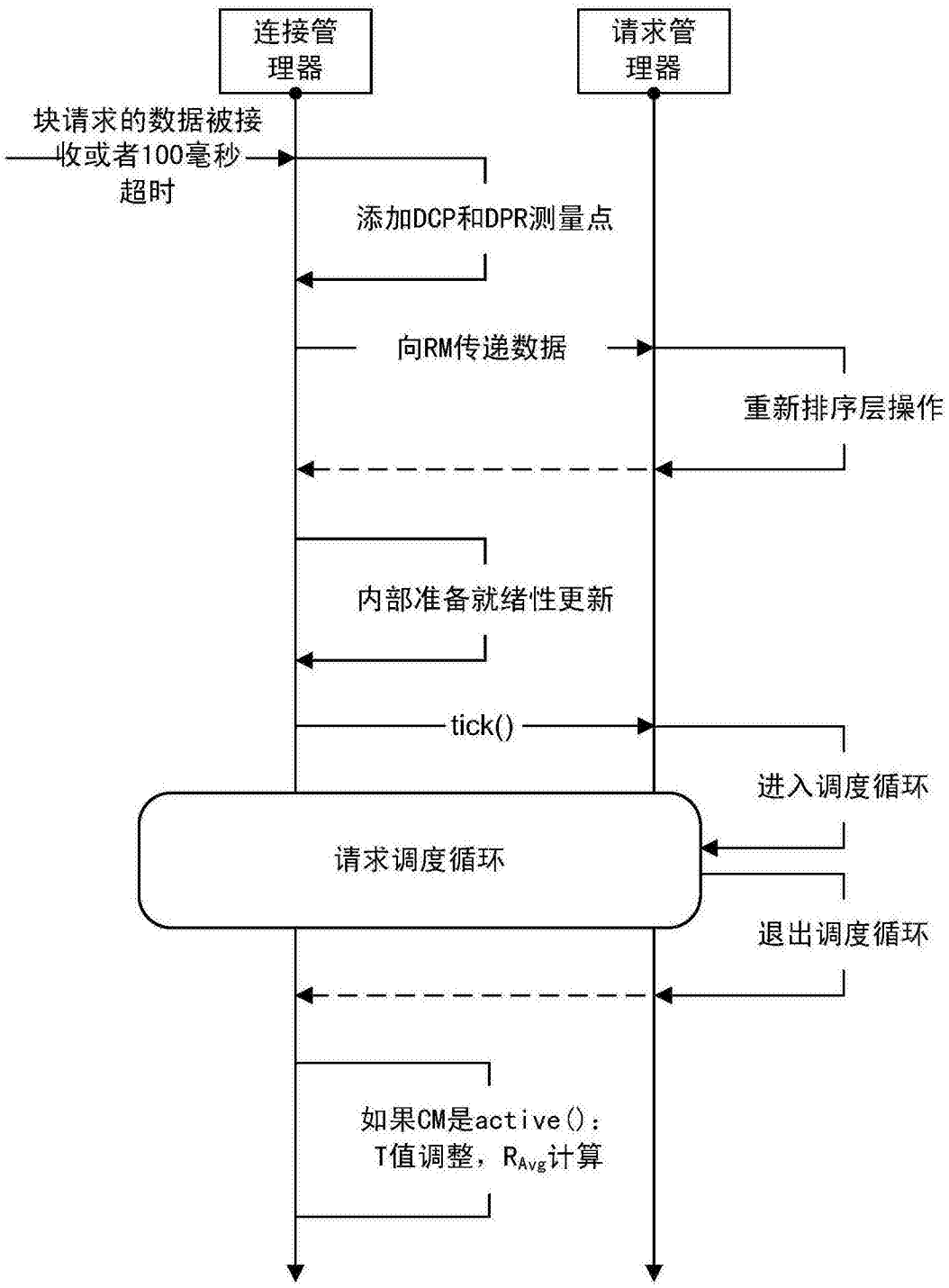

连接管理器
请求管理器
块请求的数据被接收或者100毫秒超时
添加DCP和DPR测量点
向RM传递数据
重新排序层操作
内部准备就绪性更新
tick()
进入调度循环
请求调度循环
退出调度循环
如果CM是active()：
T值调整，R_{Avg}计算

图13C

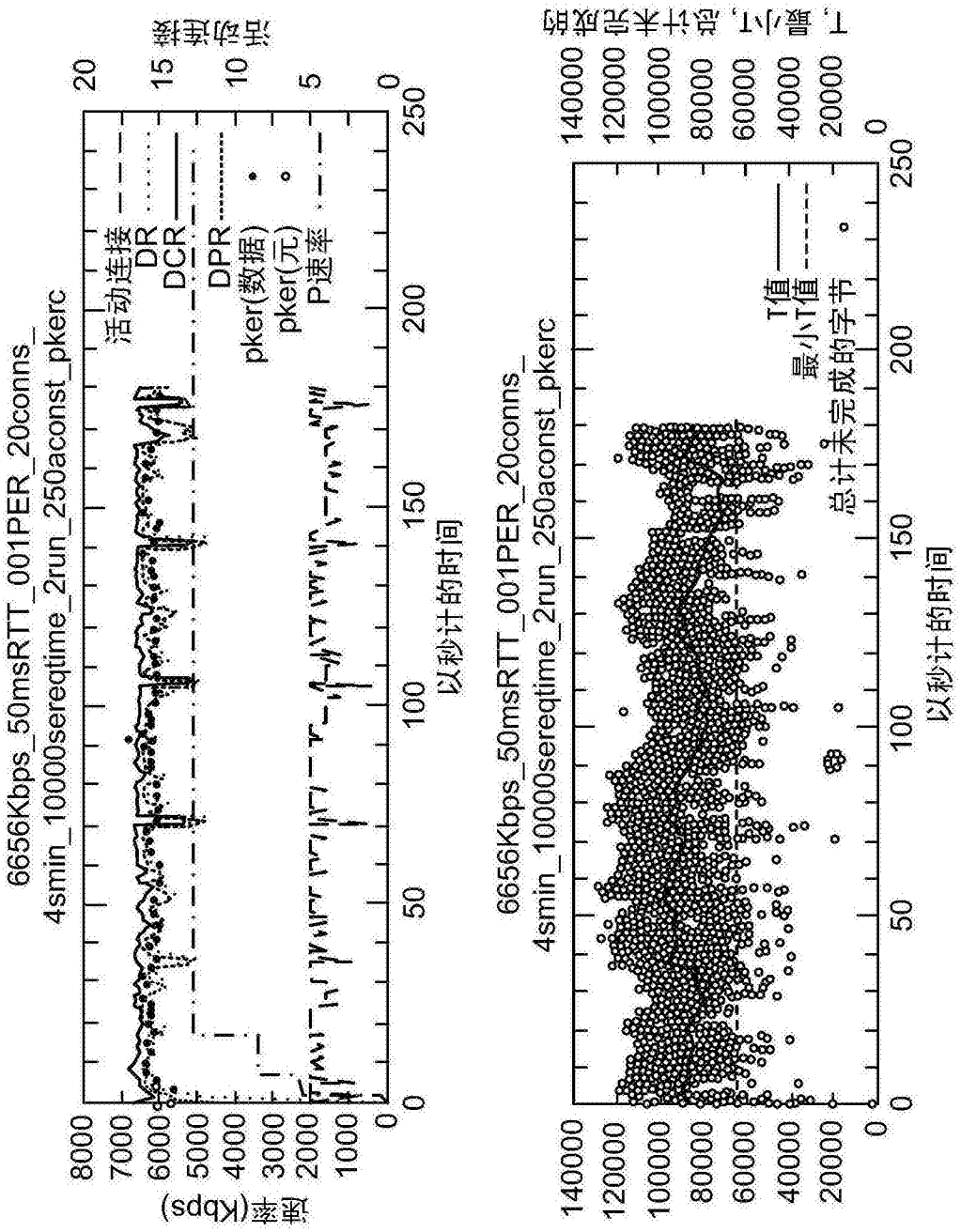

6656Kbps_50msRTT_001PER_20conns_
4smin_10000sereqtime_2run_250aconst_pkerc
速率(Kbps)
活动连接
以秒计的时间
活动连接
DR
DCR
DPR
pker(数据)
pker(元)
P速率
T, 最小T, 总计未完成的
T值
最小T值
总计未完成的字节
以秒计的时间

图14A

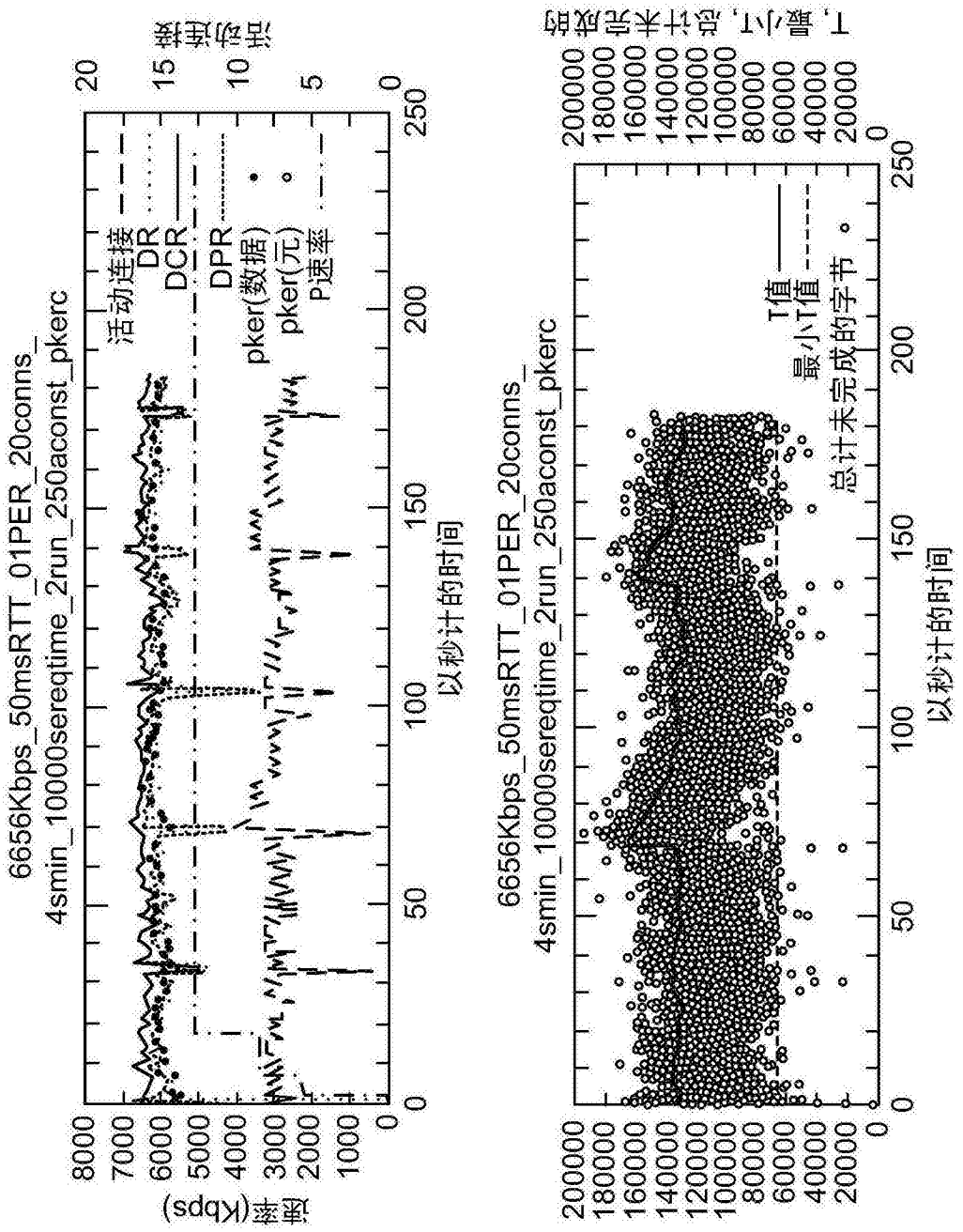

6656Kbps_50msRTT_01PER_20conns_
4smin_10000sereqtime_2run_250aconst_pkerc
速率(Kbps)
以秒计的时间
活动连接
DR
DCR
DPR
pker(数据)
pker(元)
P速率
T, 最小T, 总计未完成的
T值
最小T值
总计未完成的字节

图14B

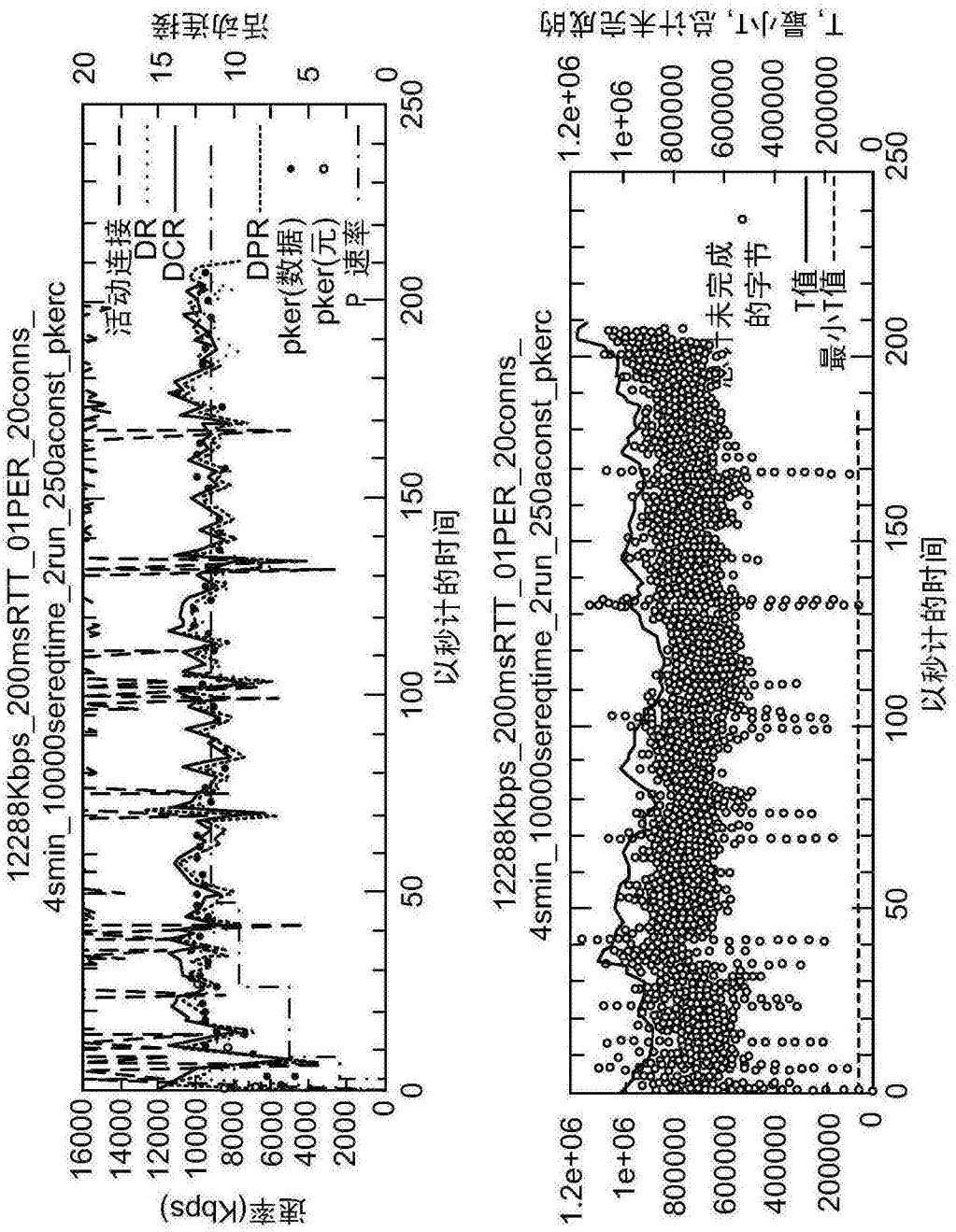

12288Kbps_200msRTT_01PER_20conns_
4smin_10000sereqtime_2run_250aconst_pkerc
速率(Kbps)
活动连接
以秒计的时间
DR
DCR
DPR
pker(数据)
pker(元)
P 速率
T, 最小T, 总计未完成的
T值
最小T值
总计未完成
的字节

图14C