[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 12/56 (2006.01)
H04Q 11/04 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200380108007.5

[43] 公开日 2006 年 10 月 25 日

[11] 公开号 CN 1853379A

[22] 申请日 2003.12.29
[21] 申请号 200380108007.5
[30] 优先权
[32] 2002.12.31 [33] US [31] 60/436,997
[86] 国际申请 PCT/US2003/041063 2003.12.29
[87] 国际公布 WO2004/062214 英 2004.7.22
[85] 进入国家阶段日期 2005.6.30
[71] 申请人 肯奈克斯特公司
地址 美国新泽西
[72] 发明人 戴维·米尔维 威廉·思托伊

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李 勇

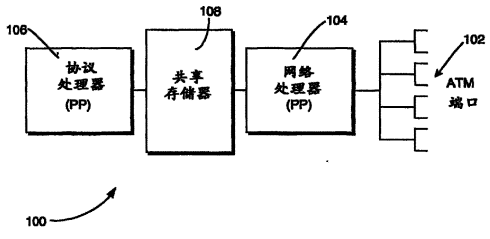
权利要求书 3 页 说明书 21 页 附图 7 页

[54] 发明名称

在异步传输模式信元传输中提供服务质量的系统和方法

[57] 摘要

公开了一种用于调度信元发送至网络，或者其他数据包交换网络的方法和设备。 调度机制的主要特征是一个服务质量引擎(QoS 引擎)，该引擎通过协助多个 ATM 虚拟电路的精确步长调节，加速在数据包交换网络，比如 ATM 网络中的数据包的处理。QoS 引擎允许对多个不同端口速度，使用不同优先级的通信量分类和通信量参数的同时支持。 服务质量引擎(QoS 引擎)与网络处理器(NP)协同工作，允许它保持软件灵活性，使其获得精确步长调节。



1. 加速在数据包交换网络中基于数据包的通信量处理的方法，包含:

提供服务质量引擎，用于通过精确调节多个虚拟电路的步长来加速基于数据包的通信量的处理;

分配数据条目值来调度在一个或多个虚拟路径上数据包的传输，其中每个数据进一步包含：一个时间戳字段，一个索引字段，一个端口字段和一个优先级字段

当特定的端口准备发送时，拣选出数据条目值来确定下一个最佳的可用条目。

2. 如权利要求 1 所述的方法，其中索引字段用于定位特定的虚拟电路。

3. 如权利要求 1 所述的方法，其中时间戳字段表明一个条目可能返回的最早的绝对时间。

4. 如权利要求 1 所述的方法，其中端口字段表明一个物理端口号。

5. 如权利要求 1 所述的方法，其中优先级字段是一个布尔类型的高或低值，并且表明条目的优先级。

6. 如权利要求 1 所述的方法,其中服务质量引擎提供一个掩码，当具有没有准备好被处理的条目的端口被服务质量引擎处理时，所述掩码用来防止该端口被显示。

7. 如权利要求 1 所述的方法，其中服务质量引擎通过网络处理器被访问，以确定是从它的存储器写入还是读出。

8. 如权利要求 7 所述的方法，其中网络处理器执行移除读取。

9. 如权利要求 8 所述的方法，其中每个移除读取在移除之前被检查优先级。

10. 如权利要求 8 所述的方法，其中每个移除读取是一个有条件的或者无条件的移除。

11. 用于加速在数据包交换网络中基于数据包的通信量处理的设备，包含:

通过精确调节多个虚拟电路的步长，用来加速基于数据包的通信的处理的服务质量引擎;

用来调度在一个或多个虚拟电路上的数据包的传输的数据条目值，其中，每个数据进一步包含: 一个时间戳字段，一个索引字段，一个端口字段和一个优先级字段;

当特定的端口准备传输时，用于确定下一个最佳的可用条目的数据条目值拣选器。

12. 如权利要求 11 所述的设备，其中索引字段被用于定位特定的虚拟电路。

13. 如权利要求 11 所述的设备，其中时间戳字段表明条目可能返回的最早绝对时间。

14. 如权利要求 11 所述的设备，其中端口字段表明一个物理端口号。

15. 如权利要求 11 所述的设备，其中优先级字段是一个布尔类型的高或低值，并且表明条目的优先级。

16. 如权利要求 11 所述的设备，其中服务质量引擎提供一个掩码，当具有没有准备好被处理的条目的端口被服务质量引擎处理时，所述掩码用来防止该端口被显示。

17. 如权利要求 11 所述的设备，其中服务质量引擎通过网络处理器被访问，以确定是从它的存储器写入还是读出。

18. 一种计算机可读的介质，包含一个用于在数据包交换网络中加速基于数据包的通信量处理的计算机可执行程序，这个程序包含:

一个或多个用于提供用来加速基于数据包的通信量处理的服务质量引擎的指令;

一个或多个用于分配用来调度在一个或者多个虚拟电路上的数据包传输的数据条目值的指令;

一个或多个用于拣选数据条目值的指令，以当一个特定端口准备

传输的时候确定下一个最佳的可用数据条目。

19. 如权利要求 18 所述的计算机可读介质，其中一个或多个用于分配在一个或者多个虚拟电路上调度数据包的传输的数据条目值的指令进一步包含一个或多个指令，用于识别一个时间戳字段、一个索引字段，一个端口字段和一个优先级字段。

20. 如权利要求 18 所述的计算机可读介质，用于提供一个掩码，当具有没有准备好被处理的条目的端口被服务质量引擎处理时，所述掩码用来防止该端口被显示。

21. 如权利要求 18 所述的计算机可读介质，进一步包含一个或多个指令，用于通过网络处理器访问服务质量引擎，来确定是从它的存储器写入还是读出。

22. 如权利要求 18 所述的计算机可读介质，进一步包含一个或多个指令，用于使网络处理器执行移除读取，这些移除读取在有条件的或者无条件的移除之前被检查优先级。

在异步传输模式信元传输中提供服务质量的系统和方法

对相关申请的交叉引用

本发明要求同时待决的美国临时专利申请 No. 60/436,997 的优先权，该申请提交于 2002 年 12 月 31 日，题为“在异步传输模式信元传输中提供服务质量的系统和方法”，该申请的全部内容被合并在这里以供参考。

背景技术

本发明一般涉及数据通信网络，更特别地，涉及发送控制机制，包含 ATM 通信处理器和交换机，以及在异步传输模式的系统/网络下的信元接收和信头解释。

随着数字时代的崛起，能够在不同的网络环境间高速有效地传输多种信息的一个单一通用的网络技术已经出现了增长的需求。为了响应这种需求，国际电报电话咨询委员会（CCITT）和它的继任组织国际电信联盟电信标准化局（ITU-T）开发了异步传输模式，一般称之为 ATM，作为能够通过公共和私人网络高速传输语音、视频和数据的一种技术。

ATM 采用超大规模集成（VLSI）技术将数据分割成独立的数据包，比如 B-ISDN 要求数据包有 53 个字节的固定大小。这些数据包一般被称为信元。使用 B-ISDN 的 53 字节的数据包是出于描述的目的，每一个 ATM 信元包含一个由前 5 个字节组成的信头部分和一个由其余 48 个字节组成的载荷部分。ATM 信元通过 ATM 交换机穿越各种各样的网络，该 ATM 交换机读取包含在信头里的寻址信息，并且将信元发送到其中所引用的目的地。与其他类型的网络协议不同的是，ATM 不需要依赖时分复用来建立对每个信元的识别。也就是说，ATM

技术不是通过在多路数据流中其时间位置来识别信元，而是仅根据包含在信头里的信息来识别 ATM 信元。

此外，ATM 不同于基于诸如以太网和令牌环这样的传统网络结构的系统，这不是因为它在共享的线路上广播数据包让所有网络用户接收，而是 ATM 信元通过包含在信头里的信息来指示信元的连续接收者。也就是说，在任何数据被发送之前，两个终端节点之间已经建立一个称之为虚拟路径（VP）或者虚拟电路（VC）的特定的路由通道。用特定的虚拟电路识别的信元只被传递到这个虚拟电路上的点。用这种方法，只有在信头里被识别的目的地才能收到被发送的信元。

除了其它信息之外，还包括寻址信息，该寻址信息主要描述了信元的来源也就是信元从哪儿来，和它被指派的目的地。尽管 ATM 由时分复用（TDM）这个概念演变而来，但是来自多个源的信元在统计学上经多路选择后进入一个单一传输设备。信元是通过它们信头里的内容来识别的，而不是通过它们在多路数据流中的时间位置来识别的。一个单一 ATM 传输设备每秒可以运载几十万个来自于多个源、送至多个目的地的 ATM 信元。

一个 ATM 网络的中枢包括能够处理高速 ATM 信元流的交换装置。这些装置的交换部分一般被称之为交换结构，它执行所要求的交换功能来实现一个虚拟电路，通过从输入端口接收 ATM 信元，实时地分析即将到来的信元的信头中的信息，然后将它们路由至适当的目标端口。一个单一装置每秒需要交换数百万个信元。

重要的是，这种面向连接的方案允许一个 ATM 网络保证每个连接要求的最小带宽。当连接建立的时候，这样的保证也建立起来。当一个连接被请求的时候，分析已经存在的连接来确定在网络内部是否还留有足够的带宽来按照它要求的容量服务于新的连接。如果必需的带宽不可用，则连接被拒绝。

为了使网络资源得到有效的利用，在一个统计上的多路复用方案下，带宽被分配给已经建立的连接。因此在 ATM 网络中可能偶尔存在导致信元传输延迟甚至信元丢失的拥塞现象。为了保证网络拥塞的

负担是置于那些最有能力处理的连接上，ATM 提供多级服务。这些多级服务支持各种格式的要求不同程度的信元丢失概率的通信量，传输延迟，和传输延迟的偏差，一般称为延迟抖动。众所周知，例如许多多媒体连接，比如视频流，能够忍受相对较大的信元丢失，但是对于从一个信元到下一个信元的延迟偏差非常敏感。相反，传统形式的数据通信量更能够忍受大的传输延迟和延迟偏差，但是要求很低的信元丢失。这种要求上的不同能够被用来提高网络的性能。

特别地，下面的各级服务更适宜在现代 ATM 网络中得到支持：恒定比特率（“CBR”）电路、可变比特率（“VBR”）电路和不定比特率（“UBR”）电路。这些分类定义了可用于特定连接的服务质量，并且当连接建立的时候被选中。每个分类的更明确的定义在下面提出。

一个 CBR 虚拟电路被认为是沿着它的整个路径的带宽的永久分配。相应于需要的带宽，发送端保证以一个精确的时间间隔，或者是固定的速率来发送数据，并且网络保证用最小的延迟和延迟抖动来传输这些数据。一个 CBR 电路最适合实时视频和音频多媒体流，它们要求网络服务等同于由同步传输网络提供的服务。从来源和目的地的角度来看，它必须好像是一条虚拟的线存在于两点之间。这要求属于此数据流的每个信元的传输都出现在精确的时间间隔。

一个 VBR 虚拟电路起初被用一个平均带宽和一个峰值信元速率来指定。这种电路适用于包含一些突发性的高优先级连续通信量，如压缩的视频流。基于所有的 VBR 电路不会同时处理处于峰值信元速率的通信量的假设基础上，网络可能“超额预定”这些连接。然而，虽然传输速率可能变化，使用 VBR 服务的应用程序通常要求低的延迟和延迟抖动。VBR 服务被进一步划分为实时 VBR(rt-VBR)和非实时 VBR(nrt-VBR)。这两类按照对上界延迟(Max CTD)的需要区分。Rt-VBR 提供了 MaxCTD，而对于 nrt-VBR 没有延迟界可用。

一个 UBR 虚拟电路，有时候指无连接数据通信量，被用于最低优先级数据传输；它没有指定的关联带宽。发送端可以在它愿意的时候发送它的数据，但是网络不保证数据在任何特定时间内到达目的地。

这个服务是为具有最小服务需求的应用程序，如在工作站后台提交的文件传输设计的。

网络上的某个末端节点可能具有许多这些不同种类的在任何时刻开路的虚拟电路。在末端节点的网络界面负责从这些虚拟电路中的每个以某种规则方式调度信元的传输的任务。至少，这使得从 CBR 电路的信元隔离以固定的速率来获得虚拟同步传输。另外，某种方式的调度可能在一些或者全部形成 ATM 网络的交换机内实现。由于网络中的异常而从它们的理想传输状态偏离的连接可以被返回到一个可接受的服务级别。

传统 ATM 交换系统的设计包括在哪些操作应该以硬件实现和哪些应该以软件实现之间的折衷。通常，但并不是没有例外，硬件给出最佳性能，而软件允许更大的灵活性和对调度和缓冲的控制，使得具有更加复杂的信元处理（如，OAM 信元抽取等）变得更可行。

关于 ATM 的附加背景信息可以在许多来源找到，在这里不必直接重复。例如，转让给本发明所有人的美国专利 No. 6,122,279（Milway 等）提供了 ATM 的完整描述，通过引用合并在这里。另外，也已转让给本发明所有人的美国专利 No. 5,953,336（Moore 等）除其它内容外还提供了在 ATM 通信量整形上的背景，也同样通过引用被合并在这里。

相对于通信量整形，ATM 信元的小尺寸允许在一个单独物理连接上对多个数据流的微粒交织，这意味着保持对每个流的各自约定的服务质量是可能的。然而，实际实现并不容易，因为数据流会有不同的传输量参数、不同的优先级，并且传输的数据可能来自不同的源，可能是已经格式化好的信元和必须被分割的缓冲的混合。

因此，在 ATM 网络技术中需要更加灵活的方法和系统来整形 ATM 流通信量并确保服务保证的足够质量。

发明内容

本发明克服了上面提到的问题，并通过提供一个用于调节异步传

输模式（ATM）信元的步长的系统和方法，其中整形加速引擎（以后被称为 QoS 引擎）加速了 ATM 通信量的处理，提供了额外的优势。QoS 引擎 205 通过帮助在许多物理端口上对许多 ATM 虚拟电路(VC's)精确调节步长，加速了 ATM 通信量的处理。它允许对多种端口速度、通信量种类、和通信量参数的同时支持。它与网络处理器一起联合工作，以便获得精确的步长调节，以维持网络处理器的软件灵活性。QoS 引擎操作数据条目，数目条目的值包含了代表绝对时间的时间戳，代表新建条目的索引字段，代表物理端口号的端口，代表一个高或低优先级的优先级字段，通过不断拣选那些条目的值，使得当某个特定端口准备好传输的时候，下一个最佳的条目立即可用。在一个实施例中，本发明公开了一个合并在本发明的通信量整形系统中的双处理器硬件配置。特别地，这个硬件配置包含几个用于接收并发送 ATM 信元到临近的网络节点的 ATM 端口。两个处理器，即一个网络处理器和一个协议处理器，共享一个存储器。在一个实施例中，网络处理器处理低层发送和接收 ATM 信元，该信元例如可能包含分割和重组功能，以及端口服务的调度。网络处理器可能还处理其他网络端口，并对其软件行为有严格的实时要求。协议处理器相反地处理较高层的协议并实现功能，如桥接和路由。在本发明的另一个实施例中，通用速率控制算法 GRCA（漏斗桶），用逻辑信元时间（LCT）和系统时钟来决定什么时候桶满、过满和什么时候桶空。

本发明的其他方面和优点通过下面的详细描述将更为明显，参照附图，通过例子阐明本发明的原理。

附图说明

本发明可通过结合附图阅读下面优选实施例的详尽描述，而理解得更加完整。附图中：

图 1 是显示了根据本发明的一个实施例的双处理器硬件配置的示意框图。

图 2 是显示了根据本发明的一个实施例的控制比特的插入和抽取

的服务质量引擎的示意框图。

图3是显示了根据本发明的一个实施例的选择和决定下一个端口寄存器的服务质量引擎的示意框图。

图4是显示了根据本发明的一个实施例的一个可变位速率传输，以及物理信元传输时隙、不变的信元间隔、峰值信元间隔和最后兼容时间值之间的关系的图。

图5是显示了根据本发明的一个实施例的简化每个数据流的状态机的示意框图。

图6是根据本发明的一个实施例的用于将虚拟电路插入到服务质量引擎的过程。

图7是根据本发明的一个实施例的用于重新将虚拟电路插入到服务质量引擎的过程。

具体实施方式

下面的描述旨在通过提供多个具体的实施例和包括ATM处理和系统的细节来得到对本发明透彻理解。然而，需要明白，本发明并不局限于这些具体的实施例和细节，它们仅仅是例子。需要进一步理解的是，本领域普通技术人员按照已知的系统和方法，将能够理解发明的应用，由于根据特定的设计和其他需要，本发明在许多替代实施例中的预期目的和好处。

现在参照附图，尤其是图1，显示了合并在本发明的通信量整形系统中的一个双处理器硬件配置100的一个实施例的示意框图。尤其，硬件配置100包括多个用于接收和传输ATM信元到临近网络节点的ATM端口102。还包括两个处理器104和106，以及这两个处理器共享的存储器108。在一个实施例中，第一个处理器104（以后称为“网络处理器”或者“NP”）处理ATM信元的低层传输和接收。这可能包含，例如，分割和重新组装功能，以及端口服务的调度。NP 104可能也处理其他网络端口，对它的软件行为具有严格的实时要求。第二个处理器（称为“协议处理器”或“PP”）106反过来处理较高层的协议并

实现功能，如桥接和路由。

在下面详细描述的示例性实施例中，两个普通种类的源可生成由上面的硬件配置处理的 ATM 通信量。第一类源包含本地产生的 ATM 通信量。本地产生的 ATM 通信量被定义为只要涉及到 PP 106 上的一个 ATM 驱动器就本地产生的通信量。例如，这个本地产生的通信量可能通过 PP 106 上的过程创建，或者作为替代，通信量可能包含从另外一个网络接口（在整个系统中可能不被认为是“本地产生的”，但是对即时的例子来说是“本地产生的”）桥接或者路由来的数据包。通常这个桥接或者路由的通信量被当作缓冲器，该缓冲器对应于几个 ATM 信元（如，AAL5 数据包）的分组，并必须在从输出端口发送前被分割为离散的信元。

第二个 ATM 通信量源包括到达 ATM 端口，并在到达时被逐个交换到一个或多个端口的交换 ATM 信元。交换电路可能是单点传输的，对每个输入信元有一个输出信元，或者是多点传输的，其中每个输入信元被复制为多个分支（可能在不同的输出端口）。来自两类源的 ATM 通信量流被虚拟电路或者虚拟路径承载，我们一般称之为“流”。每个流有以下特点：一个优先级，一个通信量类，如 CBR、rt-VBR、nrt-VBR 或者 UBR；和一个指定了信元应该被传输的速率和在传输中允许多少偏差的通信量参数的相应集合。为了简单起见，下面的描述假定优先级遵循具有 CBR 信元的通信量类被给予最高优先级，而 UBR 信元被给予最低优先级。然而，需要明白，这个约定不需要申请。

在本发明的另一个实施例中，参照图 2，公开了用于隔离异步传输模式(ATM)信元的系统和方法，其中一个整形加速引擎（以后称为 QoS 引擎）加速了 ATM 通信量的处理。这里描述的 QoS 引擎 205 是用于 ATM 操作的，然而，它里面没有东西是针对 ATM 特定的，它通常可以用于任何隔离的传输接口。QoS 引擎 205 通过帮助在许多物理端口的许多 ATM 虚拟电路（VC’s）的精确步长调节，加速了 ATM 通信量的处理。它允许同时支持多种端口速度、通信量类、和通信量

参数。为了获得精确步长调节，以保持网络处理器 104 的软件灵活性，它与网络处理器 104 协同工作。QoS 引擎 205 操作数据条目 215，该条目的值显示在表 1 中，包含表示绝对时间的时间戳 210（a），表示对新建条目的索引的索引 210（b），表示物理端口号的端口 210（c），表示一个高或低的优先级的优先级 210（d），通过不断拣选那些条目的值使得当一个特定端口准备好传输时，下一个最佳条目立即可用。驱动软件用“索引”字段 215(b)来定位某个特定的虚拟电路，从该电路传输一个信元，然后将具有与虚拟电路下次可以传输一个信元的时间一致的新更新过的时间戳 215（a）的条目重新插入到 QoS 引擎 205 中。

表 1: QoS 引擎存储的条目

时间戳	20-比特值(微秒计数)
索引	12－比特值
端口	8－比特值（物理端口号）
优先级	布尔－高或低

时间戳 215（a）指示了一个条目需要被返回的最早绝对时间。所有时间与一个系统时钟有关，该时钟是从重置开始的 8 分之一毫秒的 32 比特计数，并被称为“计时器 1”。时间值不可能在超过当前“计时器 1”值之前的可能范围的一半处被插入，否则，有可能在时间戳比较的时候造成反向卷绕。QoS 引擎块 205 不解释“索引”值，而是，索引值典型地被驱动软件用作流表的索引。流通常与单个虚拟电路的一个方向一致。本发明的索引 210(b)被用作第二排序键，为了在许多流具有相同通信量合约的情况下提供稳定性。

在索引值 210(b)比较的本发明的步骤中，计算连接的时间戳和索引值的 32 比特减法，当结果是正数（最高位空）时，认为第一个值在时间上比第二个值要晚，第二个值可能不是传统的带符号或不带符号的比较，但是在可用的数字空间中卷绕存在的情况下，得到了需要的

回答。

QoS 引擎的主要资源是一个大的 SRAM 存储器，对每个存储的条目具有 32 比特字。对每个端口需要一些额外的开销。这个存储器可以在每天开始时分割为每个端口最大数目的条目，这样空间不会在不中断步长调节进程的情况下被实时地重新分配。每个端口可能保存具有高或低的优先级的条目，优先级是绝对的，因为高优先级条目将永远比低优先级条目优先选择。在每个端口需要超过两个优先级的情况下，必须由外部软件实现。

QoS 引擎存储器布局

PP 106 和 NP 104 读写 QoS 引擎 205 的存储器作为存储器－映射区域，该区域被用于设置和管理，因此，高速处理可能对整个操作来说不是特别重要的。在每天开始时，各种值可能被写入，以在物理端口中分割空间。QoS 引擎 205 的存储器的最初几个字被用于提供每个端口队列的通用细节。为了这个目的每个端口队列使用四个字，所以端口 n 的队列的细节将在存储器的第 n*4 个字找到。存储器上的字以表 2 描述的方式被分割。

表 2: NP 存储器字地址，大小和目的

NP 地址 位 偏移	大小 （位）	目的
0	16	队列的基址: 从 QoS 引擎存储器的基址偏移一个字
2	16	队列大小: 字的数目
4	16	队列中当前高优先级元素的数目
6	16	队列中当前低优先级元素的数目
8	12	返回空的可用发送时隙的索引
	20	非激活选择时间: 返回空的可用发送时隙的时间戳
12	32	保留

在 QoS 引擎 205 上实现插入或者移除操作之前，PP 106 或者 NP 104 可正确地设置 QoS 引擎 205 的每个存储器地址值的结构。用于 NP

104 访问的 QoS 引擎 205 映射地址和值被显示在下面的表 3 中。

表 3：用于 NP 访问的 QoS 引擎存储器映射，与 QOS_ENGINE_MEM 有关：

地址	重设值	值
+ 0	未初始化	QoS 引擎 RAM 的位置 0
+4		QoS 引擎 RAM 的位置 1
+n*4		QoS 引擎 RAM 的位置 2

本发明一个其他实施例是对 NP104 访问 QoS 引擎 205 地址的设置，在那里网络处理器(NP)104 访问 QoS 引擎 205 的存储器地址，那些地址的比特大小，和它是否能够写入或者读出，以及在被 NP104 访问后需要实现的指令。完整的说明描述在表 4 中。

表 4：用于 NP 访问的 QoS 引擎寄存器地址，与 QOS_ENGINE 有关

地址	宽度（位）	读/写	重置值	名称	用途
+0			0	CONTROL	需要哪些比特？确实有吗？
+4	8	r/w	0	PORT	当前端口，用于插入或移除。通过从下一个端口读取自动更新
+8	12	r/w	0	INSERT_INDDEX	被任何移除操作自动更新
+12	1	r/w	0	INSERT_PRIO	被任何移除操作自动更新
+16	20	w/o		INSERT_TIME	写到此处会触发在此绝对时间，在一毫秒的 256 分之一时间内的插入
+20	32	r/w	0	PORTMASK0	端口 0..31 选择掩码
+24	32	r/w	0	PORTMASK1	端口 32..63 选择掩码
+28	32	r/w	0	PORTMASK2	端口 64..95 选择掩码
+64 (16 字)		r/o	0	REMOVExx	移除一个条目（详情请看移除操作表）

对于移除读取，每个具有一个高或低值（也就是，1 或者 0）的

地址位允许如下表 5 所描述的操作。

表 5: 地址读取的最低几位和相应的移除操作:

地址位	值	意义
位 2	0	如果存在就是高优先级，否则低优先级
	1	强制在那个端口访问低优先级
位 3	0	屏蔽时间戳（回答的高位为零）
	1	提供时间戳（在回答的位 12..31）
位 4	0	仅当时间戳与现在相比是“准备就绪”时移除，否则返回索引 0
	1	无论是否准备就绪都移除
位 5	0	移除
	1	取数（返回值，不从端口队列移除）

如何插入

在本发明的一个实施例中，参照图 3，NP 104 可通过对 QoS 引擎 305 寄存器执行 STM(多次存储)指令来实现插入操作，其中 STM 的最后地址真正触发插入操作。需要写的字应该是端口（物理端口号）310(c)，索引（新建条目的索引，在那里仅保留末端 12 位）310（b），优先级（仅保留优先级末端位）310(d)和时间戳（绝对时间）310(a)。时间值应该是 256 分之一毫秒，由于计时器 1（系统时钟）值左移 5 位，该值应该大于计时器 1 的当前值，而不大于超过计时器 1 当前值半秒。

如何移除

NP 104 通过进行一次本地目的地迂回路由（LDR）操作来从一个端口队列移除一个条目，该条目提供一个通过 NP 104 转换为代表流结构的地址的“索引”值；每次 NP 104 从下一个端口逻辑 345 读取，都设置端口寄存器，这使得通过下个端口的处理稍微快一些。当希望将所得到的代码再次插入条目时，移除一个条目的动作设置

INSERT_INDEX 和 INSERT_PRO。

选择（Polling）和下一个端口寄存器

位于 QoS 引擎 305 外部的下一个端口逻辑 345 被 NP 104 用于确定下次处理哪个端口。QoS 引擎 305 通过块 330 提供了一个掩码，该掩码提供了一个这个过程的选择掩码，如果一个端口已经被 QoS 引擎处理，但是还没有条目准备好被处理时候，该掩码阻止该端口出现。QoS 引擎不断地检查数据库 310 来看哪个端口是“成熟的”，也就是说，具有需要处理的条目，或者到达了非激活选择时间。

使用与物理端口无关的端口队列

既然没有什么能阻止软件维护与物理端口无关的端口队列。软件可以简单地通过选择一个不由 QoS 引擎隔离的端口 n，保证端口 n 没有在选择掩码中被设置，在 QoS 引擎 305 中分配需要大小的内存，初始化 QoS 引擎 305 位置 n*4 和 n*4+1 为指向分配内存的指针，并按照需要插入或者移除条目。需要注意的是，非常希望下一个端口 345 硬件来提供一个比特，该比特不与物理端口相对应，但是可以在低优先级时宣称“准备好传输”。软件可以利用这个比特，例如与端口队列一起，调度“后台”操作，如 ATM 多点传输信元的复制，这样一个在 QoS 引擎 305 内提供的操作模式不需要特别的硬件，可能希望它提供多个这样的比特。例如，软件可以利用它来实现到 ATM(IMA)链路上的反向多路复用，或者虚拟路径（VP）的隔离。这些应用建议提供几个这样的比特。

GRCA 表现

对于可能用于在虚拟电路(VC)上的时间信元传输的 GRCA(通用速率控制算法)

表 6：通用速率控制算法（GRCA）参数，大小和单位

GRCA 参数	代表	值的大小	值的单位
SCR,不变信元速率	SCI，在不变信元传输时间之间的间隔	32 位	256 分之一毫秒
PCR，峰值信元速率	PCI，峰值信元传输时间之间的间隔	32 位	256 分之一毫秒
BT，突发容错	在峰值速率允许的最大突发长度	30 位	64 分之一毫秒
类	rtP，实时峰值标志	1 位	CBR 和 rt-CBR VC 置位，所有其他清除
	rtS，实时不变标志	1 位	CBR 和所有 VBR 置位，清除 UBR
目前信用等级	LCT，逻辑信元时间	32 位	256 分之一毫秒

在本发明的一个实施例中，阐述了通用速率控制算法（漏斗桶），其中逻辑信元时间（LCT）是来自 GRCA 的亏蚀，而定时器 1（系统时钟）是 GRCA 的信用。如果 LCT 等于定时器 1，桶是满的，因此有最大信用，当 LCT 小于定时器 1，桶过满，必须被小心削减，当 LCT 大于或者等于定时器 1 和突发容错（BT）的和，桶是空的。当在峰值速率传输时，LCT 比定时器 1 前进得快，直到 BT 超出（没有信用留下）。在这点上，仅需要以持续不变的速率传输，LCT 和定时器 1 将从那时起步调一致地前进。

把比较当作简单的无符号整数比较，在大多数情况下卷绕都巧妙的起作用，但是如果流在 QoS 引擎以外（因为没有数据发送）的停留时间超过 LCT 卷绕所需时间（大约 8 秒）的一半就会产生问题。在这种情况下，流看起来没有信用。这是一个相对较小的影响。如果按照在控制软件中的一个后台线程的要求，该线程至少每隔 8 秒必须访问所有的流并修补 LCT 值，可以克服上述问题。作为一个特别的实例，

GRCA 530 用于接收（管辖）或发送（隔离）信元。在传输中，由于链路层或者物理层引入的偏差，或者由于一个步调不一致的发射器，不使用信元延迟变化(CDV)，这与比峰值信元间隔（PCI）更近的信元的容限是一致的。对于一个接收（管辖）GRCA，在特定时刻接收信元的动作将更新状态，并生成一个布尔返回值用来指明接收是否违反了管辖操作，该操作典型地将导致信元被加标记（如，使用信元丢失优先权，CLP 位）或者丢弃。

不变的信元间隔(SCI)被设定为支持高速端口的高精度，来减少整体上的长期漂移。实际上最大允许值是由在数据库中保持的时间戳（也就是半秒）的范围来决定的。比这慢的步长调节速率不直接被 QoS 引擎支持。慢一些的速率由软件产生，每隔一个流活动发送一个信元。OC3 速率（通常为 155.52Mbps）意味着最小可能信元间空隙是 2.73μs，所以用 256 分之一毫秒意味着在信元中速率的最差可能误差每秒钟约为 700 分之 1。对大多数用途这就足够了，其中 BT 的最低两位被作为标志，当在峰值或者不变速率时给出优先权。

典型虚拟电路状态机

在本发明的另一个实施例中，参考图 5，NP 软件实现了虚拟电路状态机，为了阐述一个代表在单个 ATM 虚拟电路或者 VP 上的单向步长调节通信量的流，其流程由图 5 的开始天 515 开始，那里存在虚拟电路但是没有信元发送。虚拟电路没有插入 QoS 引擎 505，但是如果虚拟电路存在，不在 QoS 引擎 505 中，现在有数据发送，虚拟电路被插入 QoS 引擎 505，请求现在的时间，其允许等待物理发送时隙。用 T 做计时器 1（也就是系统时钟），本发明的一个实施例执行一个插入计算，它由系统检查组成，检查是否系统时钟 T 与突发容错(BT)两倍相加比逻辑信元时间（LCT）小，如果检查的结果为真，则系统时钟 T(也就是计时器 1)和逻辑信元时间是相等的，因此状态是在长时间没有信元后被 LCT 卷绕修补。当 T2 等于 T，那是调度发送的时间。但是如果系统时钟 T 加上突发容错比逻辑信元时间大，意味着在峰值信元速率没有足够的信用来发送信元。如果存在实时峰值标志，那么

意味着优先权为高，因此，指示系统以峰值速率排队发送，否则优先权为低，系统以不变信元速率传输。并且如果存在实时不变标志，优先权可能是高也可能是低，指示系统以不变速率排队发送。当虚拟电路从 QoS 引擎 505 出现，由块 510 决定是否虚拟电路有数据要发送，如果没有数据发送，它被传递到块 520，那里等待直到有信元可以发送，并且 GRCA530 的状态没有更新，虚拟电路被重新插入 QoS 引擎 505。最终，当虚拟电路在时间 T 由 QoS 引擎出现，并被块 510 确定为虚拟电路有数据发送的时候，它被传递到块 525，其中发送一个来自该流的信元，基于使逻辑信元时间等于逻辑信元时间和不变信元间隙相加的和，虚拟电路通过 GRCA 530 在时间 T2 以优先级 P 被重新插入 QoS 引擎 505 中，这解释为信用减少；如果时间 T 比逻辑信元时间大，逻辑信元时间与 T 相等，意味着存在严重的拥塞，因此，信用必须被限制为最大值;如果 T 和突发容限的结合超过了逻辑信元时间，表示有足够的信用，并且那些信元可以以峰值信元率被发送。如果 T2 等于 T 和峰值信元率间隔相加的和，并且如果一个实时的峰值标志存在时，峰值为高，而且 QoS 引擎 505 以峰值率排队发送。否则，峰值为低，而且 QoS 引擎 505 以低于峰值信元率（典型的以不变的信元率）发送。当 T2 等于从突发容错减去逻辑信元时间，并且如果一个实时的不变标志存在时，不管峰值是高还是低，QoS 引擎 505 以不变的速率排队发送。

在本发明的一个实施例中，显示了在每一个端口，QoS 引擎 505 以两个单独的优先级保持经过筛选的流。这些基本的原理揭示了对 ATM 流的分类的映射。在恒定比特率 CBR，虚拟路径代表不变信元速率的间隔，它小于或者等于线速率，只要线速率不大于 SCI 并且峰值信元速率的间隔等于不变信元率的间隔，相等于等于零的突发控制，并且实时峰值和实时不变标志相等并且是真。然后虚拟路径将一直排在高优先级（如果它有信元发送）。如果一个虚拟路径没有可用的信元，则它不建立信用。当一个虚拟路径由于一些其他原因（由于其他 NP 负载；物理发送率间隔尺寸；或者其他的 CBR 通信流）经历一个

小的延迟，它将建立信用，因此整个通信能力将不受影响。在实时变化比特率 VBR，rt-VBR 虚拟电路代表不变信元速率的间隔，它大于峰值信元速率的间隔，峰值信元速率的间隔大于线速率，对一个 n 信元的峰值突发来说，突发容错等于（n+1）×SCI－n×SCI，并且实时峰值和实时不变标志相等并且是真。然后虚拟电路将一直排在高优先级，将实现 PCI 和 SCI。PCI 将不会超额预定，也就是说，所有 CBR 和 rt-VBR PCI 的和应该小于物理线速率。因此，在实际中，PCI 不可能比 SCI 大得多。如果 PCI 被超额预定而 SCI 没有被超额预定，那么所有 CBR 和 rt-VBR 的 SCI 将最终符合。在非实时的可变比特率 VBR，虚拟电路代表不变信元速率的间隔，它大于峰值信元速率的间隔，峰值信元速率的间隔大于或等于线速率，对一个 n 信元的峰值突发来说，突发容错等于（n+1）×SCI－n×SCI，实时峰值标志是假，实时不变标志是真。然后虚拟电路排在低的优先级但以峰值发送，排在高的优先级但以不变值发送。虚拟电路确保实现它的 SCI，但是更高的带宽依赖于在低优先级队列中的负载。高（超额预定）的峰值速率是允许的并且不会中断 CBR 通信流。但是，保证了不变速率。SCI 不应该被超额预定（当加上 CBR 和 rt-VBR 的 SCI 的时候）。如果那样，CBR 的通信能力将受到影响。在前面许多未指定比特率 UBR 的虚拟电路中，它们能够在低优先级找到位置，而在峰值速率处发现整个峰值时间已经过去，实际上甚至不能得到 SCI。如果所有这些通信量分类的混合必须同时得到支持，并且这个行为的可能性是不可接受的，则每个端口应该采用四个优先级。在 UBR＋（以峰值信元速率的 UBR），虚拟电路代表不变信元速率的间隔，它大于峰值信元速率的间隔，峰值信元速率的间隔大于或等于线速率，对一个 n 信元的峰值突发来说，突发容错等于（n+1）×SCI－n×SCI，实时峰值标志是假，实时不变标志是真。然后虚拟电路一直锁定在低的优先级。在这种约束之下，系统将在可用的带宽之内应用 PCI/SCI/BT。高的、且超额预定的峰值速率是允许的。在严重的过负载的时候，带宽与 SCI 而不是 PCI 成比例地被分配。在未指定的比特率 UBR 下，虚拟电路不再占据

调度器队列。当一个物理端口有一个发送时隙，而且没有等候的虚拟电路允许发送，则控制软件能够从一个 UBR 虚拟电路发送一个信元。在可用比特率 ABR 下，对 ABR 没有特定的支持，除了在每个信元再次指定 SCR 之外。联系上下文，ABR 应该看作是类似可变比特率，而不是一个影响 SCI/PCI 设置的额外的控制平面。

用四个优先级的通信量分类

一旦当所有上述的通信量分类都被使用时，在一个单端口，四个优先级可用于一个端口，用两个存储器的区域，根据是否在 PCR 或者 SCR 来将 VBR（和 ABR）移动到两个独立的优先级的主意可能是合适的。根据通信量分类和虚拟电路数目的精确混合，每个端口四个优先级可能是在一个独立端口上运行四个优先级的一个要求。软件可以为单个物理端口上的一个队列采用两个端口号来便于实现上述要求。对应于两个逻辑上的端口，QoS 引擎的存储器必须被分配在两个不同的区域。

多点传送：在步长调节方面不需要特别考虑。只要考虑到输出功能，多点传送虚拟电路的每一个输出就表现为一个单点传送虚拟电路。

虚拟路径（VPs）：虚拟路径的输出就像是一个虚拟电路。对于软件来说，可以用这个虚拟路径的一个逻辑端口号（和对应的 QoS 引擎的储存器区域）来调节流进虚拟电路的信元的步长。这个虚拟路径也被认为是在虚拟路径所运行的物理端口上的一个单一条目。在 NP 软件中实现的状态机连接到基于多种组合的各种状态：

无信元发送/在虚拟路径之内有一个或者多个虚拟电路可以发送

无能力/虚拟路径本身可以发送一个信元

仅仅少量的虚拟路径以这种方式控制，因为每一个要求预先分配的 QoS 引擎的存储器区域，和预先分配的软件控制的在下一个端口寄存器中准备发送的比特。

用于 ATM（IMA）的反向复用：运行在 IMA 链路内的虚拟电路占据一个端口队列，就像虚拟路径一样。除此之外，如果需要的话，IMA 链路运行的物理端口能够使用端口队列设备。软件实现了 IMA

的状态机。

QoS 引擎存储器的动态再分配。这部分的目的是允许端口队列区域在使用当中被再分配。这将大大减轻不得不预分配可用空间的痛苦。在流建立中，如果没有足够空间，一个端口的端口队列将会被移到一个较大的连续区域，只要那里有一些空置空间。这意味着一个“端口”的条目被分为多个存储器物理区域。这与用于每个端口的四个优先级级别的硬件支持相结合，该硬件支持对每个端口仍然需要多个物理区域。端口条目的表示被改变如下。它将 Qos 引擎存储器的几个区域映射在单一端口上。

QoS 引擎存储器的动态再分配

本发明的另一个实施例是一种允许端口队列区域在使用中被再分配的方法。在流建立中，如果没有足够的空间，一个端口队列将会被移到一个较大的较大的连续区域，只要那里有一些闲置空间。这意味着一个“端口”的条目被分为多个存储器物理区域。这与用于每个端口的四个优先级级别的硬件支持相结合，该硬件支持对每个端口仍然需要多个物理区域。端口条目的表示被改变如下。它将 QoS 引擎存储器的几个区域映射在单一端口上。

端口 n（可以是物理或者逻辑端口；端口号被下一个端口逻辑所定义）在 QoS 引擎存储器的 32 比特位置 n 上有一个单字指针。这是依次指向存储器区域描述符的一个指针。

表 7：存储器描述符（MD）的布局图

NP 地址字节偏移	大小（位）	命名	目的
0	16	base	队列的基址：从 QoS 引擎存储器的基址偏移一个字
2	16	size	队列的大小：字的数目
4	16	nhi	队列中当前高优先级元素的数目
6	16	nlo	队列中当前低优先级元素的数目
8	12	nulli	返回空的可用发送时隙的索引
	20	nullt	非激活选择时间：返回空的可用发送时隙的时间戳
12	16	next	指向另一个 MD 的指针，或者 0 代表“链的结束”
14	1	same	如果置位，并且 next!=0，出于选择的目的，则链中的下一个存储器描述符与当前的处于同一个优先级。否则下一个的优先级比当前的要低
	15		未用

如果“next”字段是非空，则 nulli 和 nullt 字段被忽略。因此，只是在链的末端得到 nulli/nullt。实际上，它们是所有项目中优先级最低的。

设备的控制软件必须构建这些在每天开始时分配储存器中的描述符链。期望这是在 PP 上运行，而不是在 NP 上。PP 也会记录什么存储器区域没有被使用。

选择硬件必须在每次端口被选择时在描述符链上移动。缺省时，在链中的每个 MD 提供另外两个优先级。或者，如果“same”标识置位，则这二者彼此之间认为是同一个优先级。

插入操作中的端口寄存器被称为 INSERT_ADDR 的寄存器替代，这个 INSERT_ADDR 寄存器是指向插入出现的 MD 的一个指针。移除操作填入正确的值来允许容易的再次插入。如果通信量分类想要支持从一个 MD 移除并且插入另一个，比如改变优先级，那么软件必须用自己的结构指出这两个 MD 的关联。

这种结构的目的是在两个存储器区域内，一个端口可能有四个（或者甚至六个）优先级。因此一个存储器区域能够被重新分配并且在新旧区域都链接到链上时提供服务。这个旧的区域紧靠在新的之前，而且“same”位置位。

为了支持“重新分配”的情形，有一个对移除行为的最终诀窍。当从 MD 中移除并且“same”位置位时，INSERT_ADDR 寄存器被设为在这个端口链中的下一个 MD。以这种方法，条目将从旧的端口队列中移除，并插入新的端口队列中。更详细地，这将允许完全在 PP 的控制下端口队列区域的重新分配。NP，或者 QoS 引擎，均不需要中断服务。假设一个有四个优先级端口 n，被两个 MD 在 m1 和 m2 表示。在 m1 要代替较高优先级的 MD:

1. 分配新的存储器区域。

2. 构建一个新的 MAD，m3，用于代替 m1。

3. m3->next=m2; m1->next=m3，m1->same=true。现在在端口 n 的链上有三个 MAD，依次是 m1，m2，m3。在 MAD 中这些字段的安排意味着 m1->same/true 的更新是原子性的。

4. 随着条目成熟，m1 的存储器应该慢慢变空，此前一直等待。

5. 设定 QoS 引擎存储器的位置 n 指向 m3，而不是 m1。此时 m1 从列表中被（原子性地）移除。它的存储器能够为今后的使用被再次收回。

指针的初始数组，而不是紧接着 MAD 的初始数组，将明确允许链中第一个 MAD 条目的原子性的异步替代。当这个端口的储存器区域仅有一个时，m1 替代情形也工作。在 m2 替代较低优先级的 MD:

6. 分配一个新的存储区域。

7. 构建一个新的 MAD，m3，用于代替 m2。

8. m2->next=0; m2->next=m3，m2->same=true。现在在端口 n 的链上有三个 MAD，依次是 m1，m2，m3。在 MAD 中这些字段的安排意味着 m2->same/true 的更新是原子性的。m2 的 nulli/nullt 字段现在被忽略，因为 m2->next 不再是 0。

9. 随着条目成熟，m2 的存储器应该慢慢变空，此前一直等待。

10. 设定 m1->next 指向 m3，而不是 m2。此时 m2 从列表中（原子性地）移除。它的存储器能够为今后的使用中被再次回收。

所有这些使得硬件模块更加灵活，也使得每个端口四个优先级更可接受。

虽然前面的描述包含了很多细节和特征，需要明白的是包含这些仅仅是出于解释的目的，而不能解释为本发明的局限。针对上面描述的实施例可作许多修改，而不偏离本发明的主旨和范围。

图1

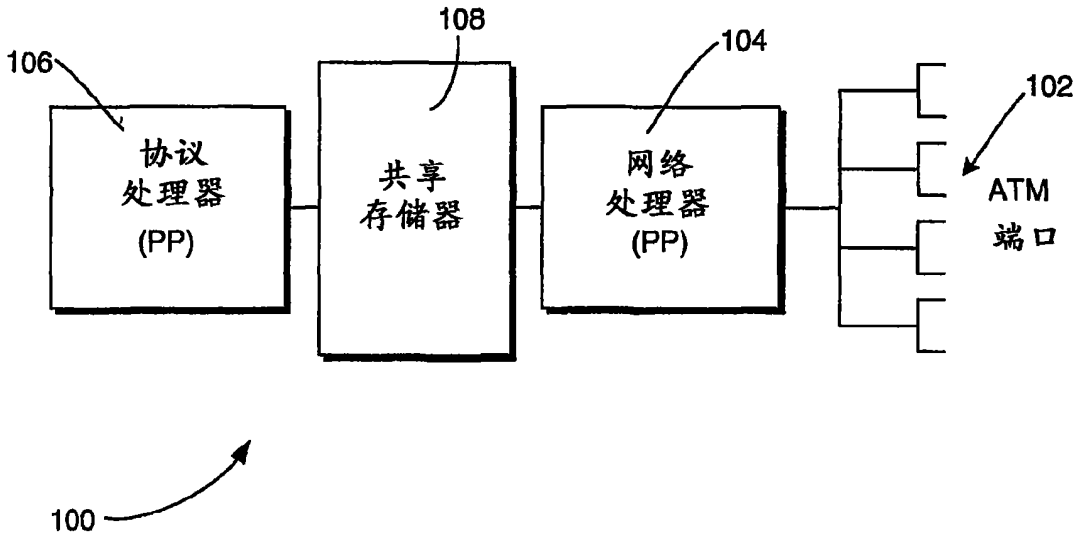


图2

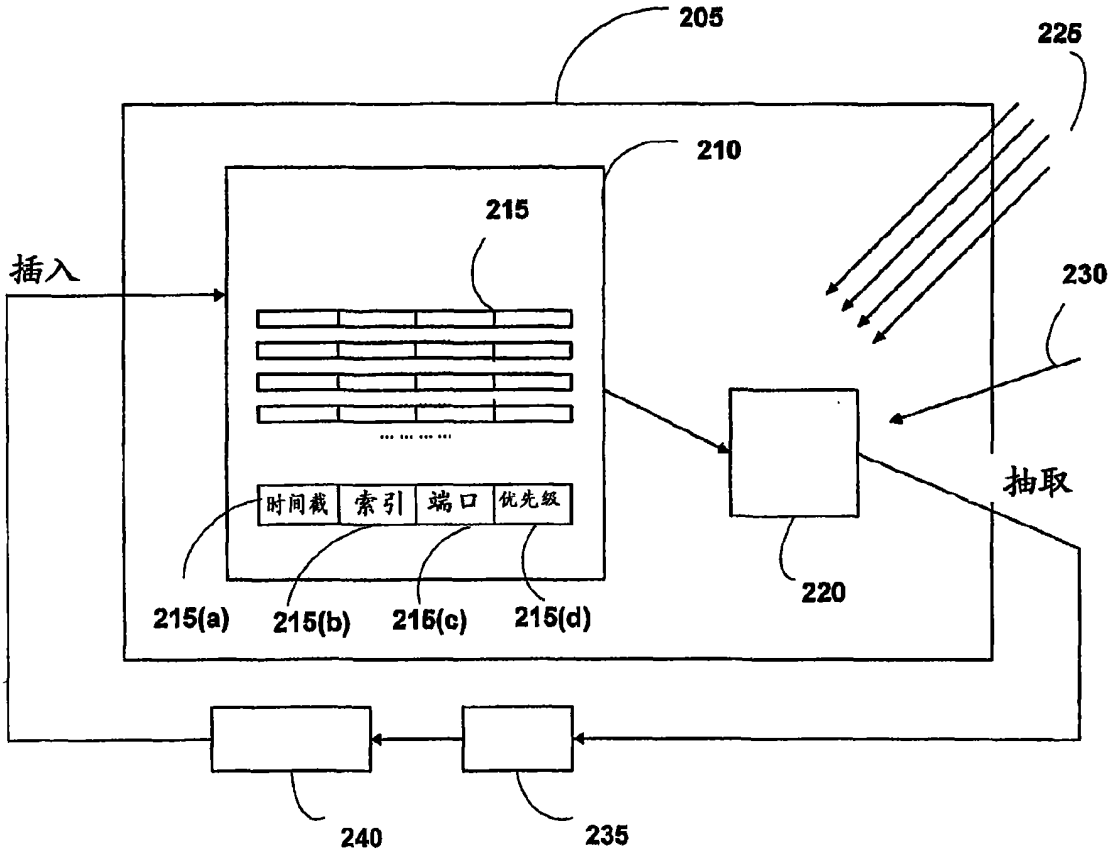

205
225
210
215
插入
230
时间戳
索引
端口
优先级
抽取
220
215(a)
215(b)
215(c)
215(d)
240
235

图3

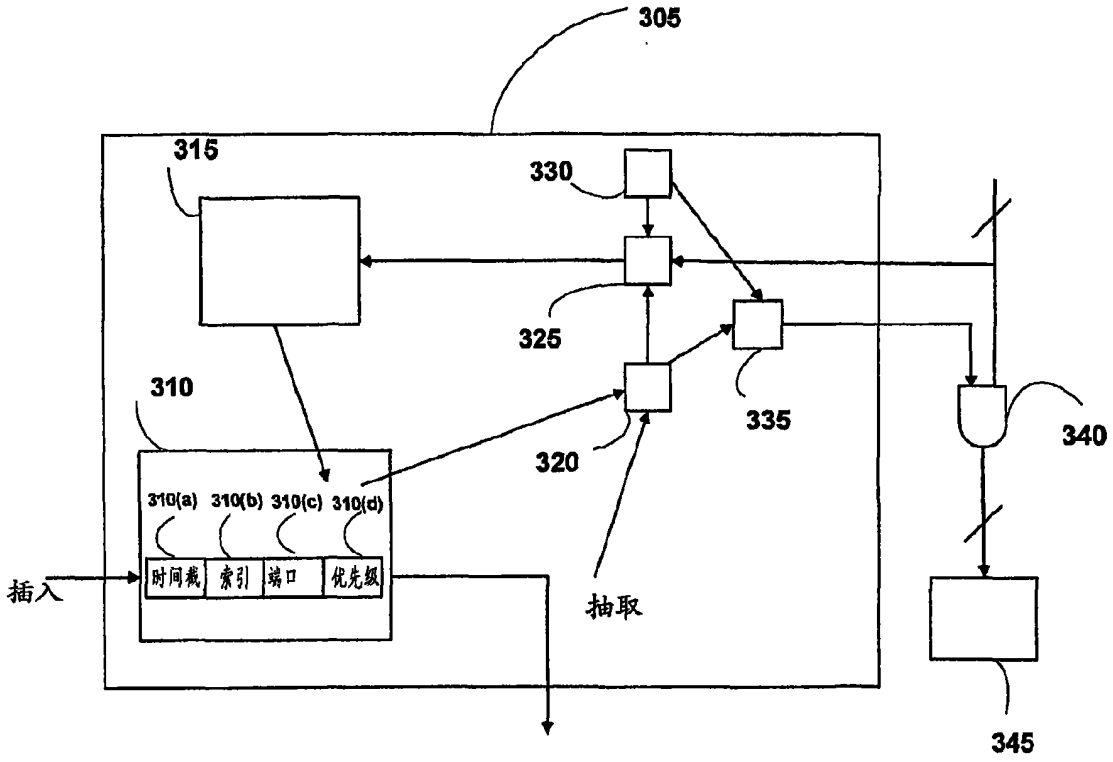

305
315
330
325
335
340
310
320
310(a) 310(b) 310(c) 310(d)
时间戳
索引
端口
优先级
插入
抽取
345

图4

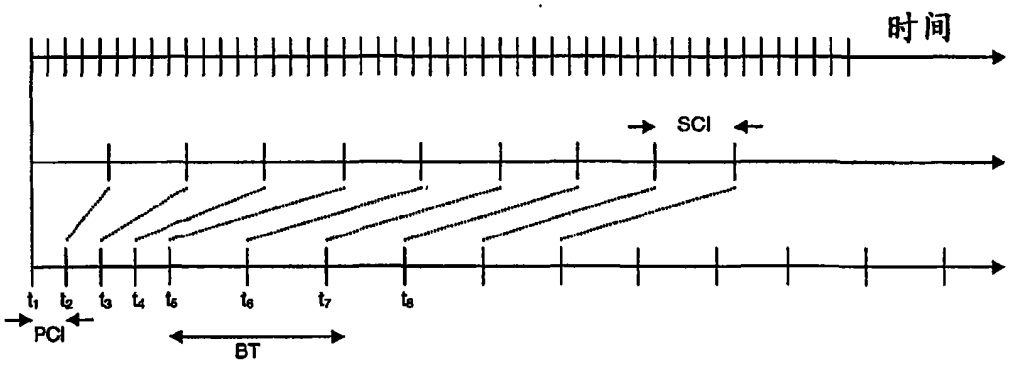


图5

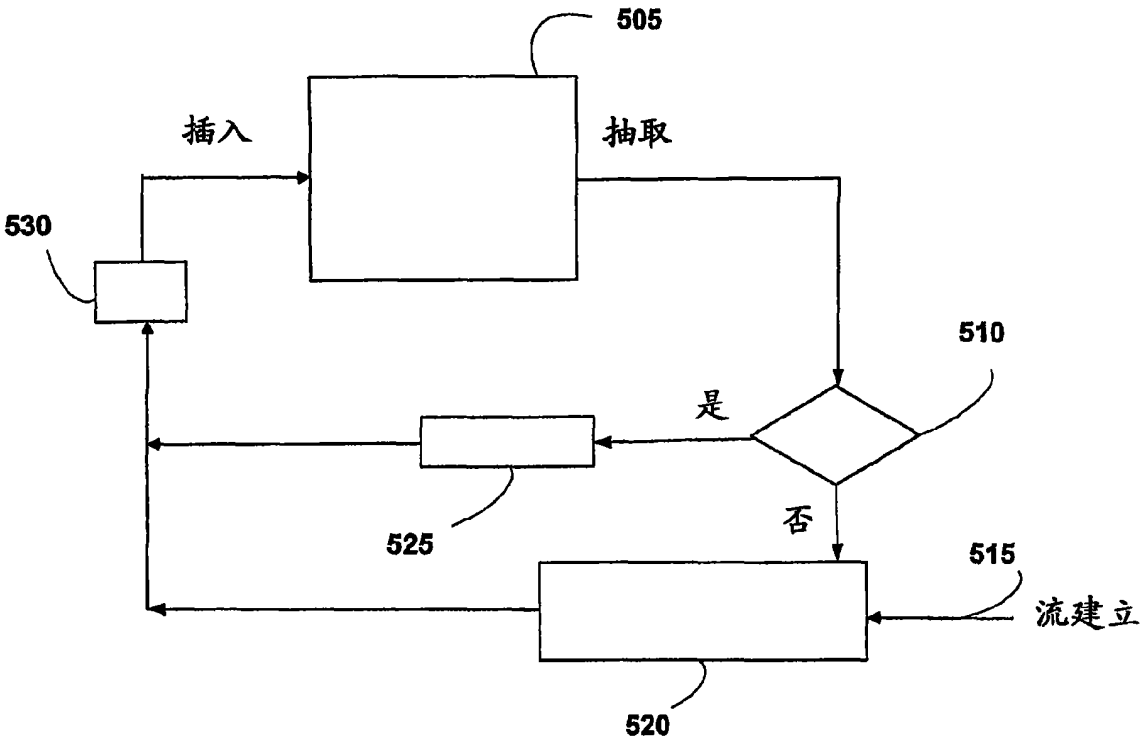


图6

插入计算如下组成（T作为计时器1）

```
if (T + 2*BT < LCT) LCT = T; // 长时间没有信元后LCT卷绕
修补
    T2 = T;                          // 现在调度
    if (T + BT > LCT)                // 在PCR有足够的信用来发送

        if (rtP) P = high; else P = low;
        // 在峰值速率排队发送

    else                             // 在PCR发送

        if (rtS) P = high; else P = low;

        // 以不变速率排队发送
```

图7

带有数据发送的虚拟路径和更新GRCA状态

```
LCT += SCI;                    // 减少信用
if (T > LCT) LCT = T;          // 严重的拥塞-限制信用为最大值

if (T + BT > LCT)              // 在PCR有足够的信用来发送
{
    T2 = T + PCI;
    if (rtP) P = high; else P = low;
    // 以峰值速率排队发送
}
else                           // 以小于PCR（典型在SCR）发送
{
    T2 = LCT - BT;
    if (rtS) P = high; else P = low;
    // 在不变速率排队发送
}
```