(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 112468412 A
(43)申请公布日 2021.03.09

(21)申请号 202010940099.9

(22)申请日 2020.09.09

(30)优先权数据
19196173.9 2019.09.09 EP

(71)申请人 TTTECH 电脑科技公司
地址 奥地利维也纳

(72)发明人 A·芬齐 S·S·克勒丘纳什
R·S·奥利弗

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 李湘 闫小龙

(51)Int.Cl.
H04L 12/853(2013.01)
H04L 12/875(2013.01)
H04L 12/865(2013.01)

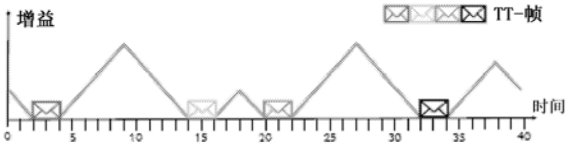
权利要求书2页 说明书21页 附图4页

(54)发明名称

用于为混合临界计算机网络生成调度表的方法

(57)摘要

一种用于生成用于例如在TT以太网或TSN网络中的网络中的时间触发的TT消息的传输的调度表的方法，其中所述网络包括部件，例如节点和星形耦合器，其中所述网络的部件根据所述调度表并基于全局网络范围的时间来传送时间触发的消息，并且其中所述部件传送速率受限的RC消息，其中对于所述RC消息中的每个规定了实时要求，其中所述方法包括以下步骤-步骤1：设置在所述网络中传送的所有TT消息的传输时间，以及-步骤2：执行搜索函数以找到TT传输时间的集合，使得满足所有RC消息的所述实时要求，并且假使，满足了所有实时要求或者满足至少针对定义的RC消息的实时要求，则在以下中生成-步骤3：基于在步骤2中检索的所述传输时间的调度表，-或者假使未满足所有实时要求或者未满足针对所述定义的RC消息的所有实时要求，则再次执行步骤2。



CN 112468412 A

1.一种用于生成用于例如在TT以太网或TSN网络中的在网络中的时间触发的TT消息的传输的调度表的方法,其中所述网络包括部件,例如节点和星形耦合器,其中所述网络的部件根据所述调度表并基于全局网络范围的时间来传送时间触发的消息,并且其中所述部件传送速率受限的RC消息,其中对于所述RC消息中的每个规定了实时要求,

其中所述方法包括以下步骤

-步骤1:设置在所述网络中传送的所有TT消息的传输时间,以及

-步骤2:执行搜索函数以找到TT传输时间的集合,使得满足所有RC消息的所述实时要求,并且

假使,满足了所有实时要求或者满足至少针对定义的RC消息的实时要求,则在以下中生成

-步骤3:基于在步骤2中检索的所述传输时间的所述调度表,

-或者假使未满足所有实时要求或者未满足针对所述定义的RC消息的所有实时要求,则再次执行步骤2。

2.根据权利要求1所述的方法,其中使用优化函数,优选地利用SMT-求解器,来设置所有所述TT消息的所述传输时间,特别是针对一个TT流之后的另一个TT流。

3.根据权利要求1或2所述的方法,其中所述RC消息的所述实时要求包括端到端延迟限制,并且其中步骤2中的所述搜索函数是基于以下循环步骤的循环:

-循环步骤1:将每个RC流的最坏情况的端到端延迟限制与其对应的截止时间进行比较,其中,假使根据所述RC流的所述RC要求,所述延迟限制大于所述对应的截止时间,则执行接下来的两个循环步骤:

-循环步骤2:标识要修改的TT传输时间,以及

-循环步骤3:优选地针对选择的TT流的流路径的每个输出端口,例如使用所述SMT-求解器内的优化函数,计算在循环步骤2中的所述选择的TT流的所述传输时间。

4.根据权利要求1至3中的任一项所述的方法,其中可以保留先前探索的选项的记录,所述记录可用于将所述搜索引导到解决方案空间的未探索部分中和/或用作搜索的附加停止条件。

5.根据权利要求1至4中的任一项所述的方法,其中所述搜索函数(搜索算法)包括要基于网络演算框架修改的TT传输时间的标识以及每个输出端口中的到达曲线,其表示可以到达在详述的TT业务的任何时间间隔中的输出端口的最大数据量。

6.根据权利要求5所述的方法,其中计算被TT流交叉的所述网络的每个输出端口的阶梯曲线,并且其中所述阶梯曲线由线性曲线近似,并且其中具有影响、特别是对RC流、特别是对所述RC流的所述实时要求具有大的影响的TT流通过将阶梯曲线与其线性近似相交来标识,最有可能对RC流具有大的影响的所述流被标识。

7.根据权利要求1至7中的任一项所述的方法,其中所述TT传输时间的所述计算利用优化函数来执行,所述函数优选地添加在所述SMT-求解器中,并且其中考虑了具有超周期HP的TT流的集合,所述HP是所有TT流的所有流周期中的最小公倍数,并且其中对于其中TT流发生的每个输出端口,定义增益函数。

8.根据权利要求7所述的方法,其中对于传输时间的所述计算,确定已调度的TT帧之间的增益函数,其中t^{end}_k是帧传输的结束,而t^{start}_{k+1}是下一次传输的开始,对于每个已调度的

TT帧，所述增益函数由以下确定：

$$\forall t_k^{end} \leq t < \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t - t_k^{end}$$

$$\forall t_{k+1}^{start} \geq t \geq \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t_{k+1}^{start} - t$$

其中对于每个感兴趣的流的周期，优选地是在每个输出端口中，对所述增益函数求和，并优选地由所述SMT-求解器选择所求和的增益的最大值的切除作为所述输出端口中的所述传输时间。

9.根据权利要求7或8所述的方法，其中仅对可接受时间内的增益函数求和，并且其中在每个输出端口中对所产生的所求和的增益函数进行近似，因为优选地由所述SMT-求解器选择所近似的增益的所述最大值的所述切除作为所述输出端口中的所述传输时间。

10.计算机网络，例如TT以太网或TSN网络，其中所述网络包括部件，例如节点和星形耦合器，其中所述网络的部件根据所述调度表并基于全局网络范围的时间来传送时间触发的消息，并且其中所述部件传送速率受限的RC消息，其中对于每个所述RC消息规定了实时要求，

其特征在于

所述调度表利用根据权利要求1至9中任一项所述的方法生成。

11.一种包括程序代码装置的计算机程序，所述程序代码装置用于当所述程序在计算机上运行时执行权利要求1至9中任一项的所有步骤。

12.一种计算机程序产品，包括存储在计算机可读介质上的程序代码装置，所述程序代码装置用于当所述程序产品在计算机上运行时执行权利要求1至9中任一项所述的方法。

用于为混合临界计算机网络生成调度表的方法

技术领域

[0001] 本发明涉及一种用于生成调度表的方法，该调度表用于在网络中(例如在TT以太网或TSN网络中)传输时间触发(TT)消息，其中该网络包括部件，例如节点和星形耦合器，其中所述网络的部件根据所述调度表并基于全局网络范围时间来传送时间触发的消息，并且其中所述部件传送速率受限的RC消息，其中为所述RC消息中的每个规定实时要求。

[0002] 此外，本发明涉及一种计算机网络，例如TT以太网或TSN网络，其中该网络包括部件，例如节点和星形耦合器，其中所述网络的部件根据所述调度表并基于全局网络范围时间来传送时间触发的消息，并且其中所述部件传送速率受限的RC消息，其中为所述RC消息中的每个规定实时要求。

[0004] 计算机网络，例如IEEE 802.3以太网(电气与电子工程师协会2018)或TSN(电气与电子工程师协会2017)，可以承载调度的和未调度的通信消息两者，由此调度的通信(时间触发的或TT)消息在网络范围的定义明确的时间中的预定时间点处从发送实体(部件)传输到一个或多个接收部件(实体)，而未调度的通信消息根据其他标准传输。可以应用用于消息处理和消息优先化的适当的传输协议和机制，由此确保不可能发生任何调度的或未调度的通信消息与任何给定的调度的通信消息的干扰。

[0005] 未调度的消息可以有两种类型：速率受限(RC)和尽力而为(BE)。RC业务以称为BAG(电气与电子工程师协会2018)的最小到达间隔时间发送，并且通常对最大允许的端到端等待时间和抖动具有实时要求。BE业务不具有任何实时要求。

[0006] 在混合临界网络(即，具有TT、RC和BE业务的网络)中，由于TT消息具有比RC和BE消息更高的优先级并且被首先发送的事实，TT消息的通信调度表影响RC和BE消息的及时行为，因此延迟了RC和BE消息的传输。因此，TT通信调度表可能不利地影响RC消息的传输，使得它们不遵循其实时要求。

[0007] 本发明的目的是为网络生成调度表，其中至少传送时间触发的消息和RC消息。

[0008] 该目的通过如上所述的方法实现，其中根据本发明，所述方法由以下步骤表征：

-步骤1：设置在网络中传送的所有TT消息的传输时间，以及

-步骤2：执行搜索函数以找到TT传输时间的集合，使得满足所有RC消息的实时要求，以及

在满足所有实时要求或者至少满足针对定义的RC消息的实时要求的情况下，在以下中生成-步骤3：基于在步骤2中检索的传输时间的调度表，

-或者在未满足所有实时要求或者未满足针对定义的RC消息的所有实时要求的情况下，再次执行步骤2。

[0009] 例如，可以规定实时要求需要或者至少需要针对定义的RC流，特别是针对每个RC流的端到端行进时间满足其截止时间。

[0010] 根据本发明，可以规定均匀地间隔TT传输时间以减少TT流对RC流的影响，使得例如RC最坏情况的端到端行进时间(消息从其来源到其目的地所需的时间)可能小于截止时间。

[0011] 下面描述了本发明的其他优点，这些优点可以单独实现或以任何任意组合实现：

• 可以使用优化函数(优选地利用SMT-求解器)来设置所有TT消息的传输时间，特别是针对一个TT流之后的另一个TT流。

[0012] • RC消息的实时要求可包括端到端延迟限制，并且其中步骤2中的搜索函数是基于以下循环步骤的循环：

-循环步骤1：将每个RC流的最坏情况的端到端延迟限制与其对应的截止时间进行比较，其中，在根据所述RC流的RC要求，该延迟限制大于对应的截止时间的情况下，执行接下来的两个循环步骤：

-循环步骤2：标识要修改的TT传输时间，以及

-循环步骤3：优选地针对选择的TT流的流路径的每个输出端口，例如使用SMT-求解器内的优化函数，计算在循环步骤2中选择的TT流的传输时间。

[0013] 在这里，循环步骤2将有助于将搜索引导到解决方案空间的最有希望的部分。

[0014] 如果在循环步骤1中延迟限制较小，则找到解决方案并且搜索停止。

[0015] • 可以保留先前探索的选项的记录，该记录可用于将搜索引导到解决方案空间的未探索部分中和/或用作搜索的附加停止条件。

[0016] • 搜索函数(搜索算法)包括要基于网络演算框架以及每个输出端口中的到达曲线修改的TT传输时间的标识，其表示在详细的TT业务的任何时间间隔中可以到达输出端口中的最大数据量。该方法在L.X.Zhao、H.G.Xiong、Z.Zheng和Q.Li的Improving worst-case latency analysis for rate-constrained traffic in the time-triggered Ethernet network中详细描述。

[0017] • 可以计算所述网络的每个输出端口的与TT流交叉的阶梯曲线，并且其中所述阶梯曲线由线性曲线近似，并且其中具有影响、特别是对RC流有大的影响、特别是对RC流的实时要求具有大的影响的TT流通过将阶梯曲线与其线性近似相交而被标识，最有可能对RC流具有大的影响的流被标识(另参见“findBestFlow()”下的详细描述)。该计算例如在L.X.Zhao中详细描述。

[0018] • TT传输时间的计算可以利用优化函数的使用而被执行，该函数优选地添加在SMT-求解器内，并且其中考虑了具有超周期(HP)的TT流的集合，所述HP是所有TT流的所有流周期中的最小公倍数，并且其中对于其中TT流发生的每个输出端口，定义增益函数。定义增益函数可以引导搜索良好的传输时间，即将对RC端到端延迟具有小的影响的时间。

[0019] • 对于传输时间的计算，可以确定已调度的TT帧之间的增益函数，其中t^{end}_k是帧传输的结束，而t^{start}_{k+1}是下一次传输的开始，对于每个已调度的TT帧，增益函数由以下确定：

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t_{k+1}^{start} - t$$

其中对于每个感兴趣的流周期，优选地是在每个输出端口中，对增益函数求和，并优选地由SMT-求解器选择所求和的增益的最大值的切除(abscise)作为输出端口中的传输时间。

[0020] • 可以规定，仅对可接受时间内的增益函数求和，并且其中在每个输出端口中对所产生的所求和增益函数进行近似，因为优选地由SMT-求解器选择所近似的增益的最大值的切除作为输出端口中的传输时间。“可接受的”时间意指尚未调度帧的时间间隔，并且其中该时间间隔足够大以调度当前帧。因此，每个间隔最大只能保留两个线性函数。

[0021] 特别地，实时属性的保存或实时要求的满足是指经由定义良好的通信信道(称为虚拟链路(VL))在一个发送器节点与一个或若干接收器节点之间周期性地传输消息的保证的端到端等待时间。如果传输方法确保消息在其调度的时间点(从时钟同步不精确性导出的小偏差内)处传输，而不会与其他调度或未调度的传输发生争用，则端到端等待时间是被限制的。

[0022] 所提出的方法的第二步骤(特别是搜索算法)可以涉及改变用于TT消息传输的现有调度表，由此该改变保存了已调度的TT消息的传输并保证RC消息的实时要求。

[0023] 在本文的上下文中，术语TT消息的“传输时间”是指传输时间点/传输瞬间，特别是传输所述TT消息的最早传输瞬间/时间点。特别地，这些是输出端口内部允许的最早的传输开始，这意味着消息的传输将在该时间点之后尽快开始(传输有时可被其他消息延迟)。

[0024] 特别地，规定：(a)TT消息的传输不干扰其他已经建立的TT消息传输的传输；(b)修改将不会使此类TT消息的实时要求无效，并且(c)此类TT消息的所修改的传输时间允许RC消息满足其实时要求。

[0025] 在网络中，优选地，每个节点经由物理链路连接到至少一个星形耦合器。经由设备中的每个上的物理端口实现连接。节点和星形耦合器具有有限数量的端口，并且因此具有连接物理链路的最大数量。

[0026] 所有节点和星型耦合器借助于以下各项来共享时间的公共概念：例如时间同步协议，像例如SAE AS6802(SAE国际2011)或IEEE 802.1AS(电气与电子工程师协会kein Datum)。

[0027] 节点通过交换周期性时间触发的(TT)消息和速率受限的(RC)消息相互传送。

[0028] 由虚拟链路表征的时间触发的消息具有以下属性：

-发送器节点

-一个或若干接收器节点

-一周期

-最大的消息大小

-最大的端到端等待时间

由虚拟链路表征的速率受限的消息具有以下属性：

-发送器节点

-一个或若干接收器节点

-最小的到达间隔时间

-最大的消息大小

-最大的端到端等待时间

-最大的抖动

时间触发的消息的传输可以由其VL来表征并遵循调度表。每个VL通过网络路由。路由过程包括找到将发送器节点与接收器中的每个连接起来的多跳网络路径(使用例如

Steiner Trees[Steiner,W.2010."An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks."RTSS]。

[0029] 对于网络路径上的每个物理链路,可以根据相应的消息属性来调度帧以进行周期性传输。

[0030] 依次对帧进行调度,使得在相应的跳跃处接收到前一帧之前,不对任何中间跳跃的传输时间点进行调度。

[0031] 每个帧的调度时间点可以是相对于将在其中传输帧的周期的偏移。以给定的偏移周期性地重复传输。

[0032] 如果在最大端到端等待时间内在接收器节点处接收到最后一帧,则保证了端到端等待时间。

[0033] 在操作期间,节点和星形耦合器遵循相应链路的调度表在它们的调度时间点(在给定的时间精度内)在每个链路上传输TT帧。

[0034] 链路的调度表包括该链路上调度的所有TT帧的调度的时间点。

[0035] 该调度表根据网络循环(通常是所有VL的周期的最小公倍数)循环地重复。

[0036] 每当没有更高优先级的TT消息准备发送时,发送RC帧。因此,RC消息的及时行为受TT传输调度表影响。

[0037] 包括满足所有虚拟链路的端到端等待时间且无争用的所有虚拟链路的帧的周期性传输的网络调度表的生成是复杂的操作。因此,通常是脱机(在操作之前)进行计算和分发。这意味着关于节点之间的通信的信息在操作之前也是可用的。

[0038] 如上所表征,在操作期间传输时间触发的消息本身是现有技术。

[0039] 给定在操作中如上所述的时间触发的计算机网络,本发明例如涉及一种标识不利地影响RC消息传输的TT消息并修改所述TT消息的传输时间以便保证RC实时要求(这些要求可能包括端到端等待时间和/或抖动)的方法,其中修改了相应TT帧的传输时间点而不会更改已调度的传输的实时属性(端到端等待时间),并保证了所述新VL的实时属性。

[0040] 附加地,本发明涉及一种包括程序代码装置的计算机程序,以用于当所述程序在计算机上运行时执行根据本发明的方法。

[0041] 最后,本发明涉及一种计算机程序产品,其包括存储在计算机可读介质上的程序代码装置,以用于当所述程序产品在计算机上运行时执行根据本发明的方法。

附图说明

[0042] 在下文中,为了进一步说明本发明,如附图所示,讨论了说明性和非限制性的实施例,所述附图示出了:

图1时间触发的网络的示例,

图2TT消息的周期性传输,

图3图2中所描绘的传输循环的详细表示,

图4计算TT流到达曲线的示例,

图5增益函数的计算,

图6增益函数的近似,

图7交换机架构的示例,

图8业务模型的示例，以及

图9使用情况的示例。

具体实施方式

[0043] 接下来，我们讨论本发明的许多实现方式中的一些。如果没有另外说明，则结合具体示例描述的所有细节不仅结合该示例有效，而且适用于本发明的总体保护范围。

[0044] 图1示出了时间触发的网络的示例，其包括三个时间触发的星型耦合器(即交换机)110、120和130以及六个节点(即终端系统)111、112、121、122、131、132或由其组成。如附图中所示，所有系统都经由线路(其优选地是双向的)连接并且具有公共的时基，例如如TT以太网(电气与电子工程师协会2018)中定义的时基。时间触发的(TT)消息遵循与其他业务(速率受限和尽力而为业务)共存的预配置的全局循环调度表而传输。

[0045] 全局分布式调度表确定了在网络系统(也称为“设备”或“部件”；此类系统例如是上述节点和星形耦合器)之间传输TT消息的准确时间点，以此方式实现了通过共享线路的传输而无需争用。调度表的计算在计算上是密集的，并且因此通常是脱机(即，在系统启动之前)执行的，并且全部或部分地分发给网络的系统中的每个。在运行时间处，已知精度内的全局时基对所有系统可用，并用于以循环和协调的方式执行调度表。

[0046] 未调度的业务在调度传输之间的稀疏时间期间传输，以此方式避免对调度的传输的干扰或使其限制于已知的最大延迟。

[0047] 调度的消息的传输根据虚拟链路(VL)的概念在逻辑上进行编组。VL定义一个发送器节点(即终端系统)和一个或多个接收器，以及它们之间的物理路径。VL中的消息的传输在发送器处发起，并通过物理路径(通信线路)传播，直到到达接收器终端系统节点为止。这些传播步骤中的每一个意味着在接收到先前消息之后的调度的传输。可以针对VL规定附加的限制，例如最大端到端传输截止时间，其是指消息从发送器到(一个或多个)接收器传播的最大允许间隔。

[0048] 由虚拟链路表征的时间触发的消息具有以下属性：

-发送器节点

-一个或若干接收器节点

-一周期

-最大的消息大小

-最大的端到端等待时间

由虚拟链路表征的速率受限的消息具有以下属性：

-发送器节点

-一个或若干接收器节点

-最小的到达间隔时间

-最大的消息大小

-最大的端到端等待时间

-最大的抖动

令vl_a为图1中所描绘的网络中具有发送器节点121和接收器节点131的TT虚拟链路。网络路径因此由从图1中提取的序列{121，120，130，131}组成。令周期以及vl_a的端到端延迟

为100ms。

[0049] 图2图示了在符合相应网络路径的三个节点(子图150、160和170)的传输循环内来自vl_a的TT消息的周期性传输。注意,在此示例中,仅调度了传输事件,因此没有出现节点131(接收器)。相应地,150表示来自节点121中的vl_a、节点120中的160以及节点130中的170的消息的循环传输。子图中的每个表示从最高时间点200、300和400)开始的100ms的时间循环(,并且顺时针方向进行。时钟在每个循环全局同步;因此在已知的同步精度内,所有系统的时间进程是一致的。

[0050] 本质上,在时间200,在节点121处发生vl_a的TT消息的传输事件,该事件发起发生消息传输直到时间210为止。节点120在时间310接收消息并传输后续消息,即最多到320为止完成传输。类似地,节点130在410传输后续消息,到420为止完成。此传输循环以协调的方式无限地重复。

[0051] 注意,事件310只能在事件210之后发生,因为节点120中的消息传输直接取决于由节点121传输的消息的先前接收。类似地,事件410取决于事件320的发生。实质上,对于第二个以及在沿着VL的网络路径传播的后面的消息,只能在VL的先前消息已经到达当前节点之后才能发起传输。

[0052] 图3提供了图2中所描绘的传输循环的详细表示,除了已经在图2中描绘的vl_a传输(浅灰色)之外,还包括同一通信线路中节点120的所有其他调度的传输(深灰色)。我们观察到在事件301-310和320-321之间发生了附加的传输。注意,传输循环定义了消息朝向相应节点(通常称为端口)的一条单条通信线路的输出传输。因此,180仅描绘了针对节点120和130之间的通信线路的节点120的传输循环。还应注意,与图2中所描绘的顺序传输类似,图3中所示的消息传输依赖于来自节点121、122或110的先前传输作为穿越节点120的VL路径的部分。

[0053] 时间轴上的TT帧的调度表可能对由RC流经历的延迟具有重要影响,因为TT业务具有较高的优先级。通常,此类网络中RC消息的端到端等待时间通过像网络演算之类的方法(A.Van Bemten、W.Kellerer.2016.Network Calculus:A Comprehensive Guide.TUM.https://mediatum.ub.tum.de/doc/1328613/1328613.pdf.)进行分析。提出了考虑TT消息调度表的这种分析的扩展。可以在本发明中使用的优选的定时分析基于网络演算框架[L.Zhao、P.Pop、Q.Li、J.Chen和H.Xiong,“Timing analysis of rate constrained traffic in TTEthernet using network calculus,”实时系统,2017.],其可用于计算上限延迟和线路阻塞限制。这些限制取决于由所谓的到达曲线α描述的业务到达(其代表在任何时间间隔中可以到达的最大数据量),以及取决于由曲线(所谓的最小服务曲线β)描述的资源可用性(其代表在任何时间间隔中可以发送的最小数据量)。

[0054] 提出的方法

时间轴上的TT帧的调度表可能对由RC流经历的延迟具有重要影响,因为TT业务具有较高的优先级。根据本发明,使用优选地使用RC网络演算分析的反馈回路(A.Van Bemten、W.Kellerer.2016.Network Calculus:A Comprehensive Guide.TUM.https://mediatum.ub.tum.de/doc/1328613/1328613.pdf.)来检查利用当前TT调度表计算的RC端到端等待时间满足RC截止时间,从而检查利用当前TT调度表计算的RC端到端等待时间满足RC截止时间。如果RC等待时间大于RC截止时间,则重新调度有问题的TT消息。

[0055] 均匀间隔开TT消息放置可能导致对RC消息端到端等待时间的较小影响。因此，可以规定，建立优化度量，该度量尝试在时间轴上均匀地间隔开TT帧并且使用优化目标来驱动对TT消息调度表的修改。

[0056] 首先，优选地根据所述优化度量来调度所有TT流。其次，提供RC分析以确定RC业务是否满足截止时间要求。如果是这种情况，则方法/算法停止。如果不是这种情况，则算法标识最有可能引起延迟的TT消息，并尝试寻找更好的偏移，即，优选地使用优化度量来修改所标识的TT消息的传输时间。重复该第二步骤，直到找到适当的调度表或达到停止条件为止。

[0057] 当找到满足截止时间条件的偏移的集合时，或者当搜索算法已经尝试重新调度所有可能的流而未找到未探索的偏移的集合时，搜索可能停止。

[0058] 用于重新调度的TT消息的标识(图4)

TT流的影响可以由TT业务的到达曲线表示，所述曲线可以使用(L.Zhao、P.Pop、Q.Li、J.Chen以及H.Xiong，“Timing analysis of rate constrained traffic in TTEthernet using network calculus，”实时系统，2017.)中详细描述的公式来计算。主要方面是在所有可能的情况下计算TT流的影响并保持最大值，如图4中所图示。图4中的虚线是不同的情况，而平线阶梯表示虚线的最大值，其表示最终的TT流最大到达曲线。根据该阶梯到达曲线，可以近似线性到达曲线。线性曲线的比率[线性曲线的增加]与一周期内阶梯曲线的比率相同。关于线性曲线的初始突发[在t＝0时的线性曲线的值]，计算诸如线性曲线总是优先于或等于阶梯曲线，具有至少一个交点，如图4中所图示。结果，通过修改与阶梯交点相关联的偏移的值，可以减小线性曲线的突发的值，并且因此，减小了TT业务对RC流的影响。可以使用所谓的多样化列表来保持跟踪已经探索和改变的TT流，以便通过尝试修改相同的(一个或多个)TT流的调度表而不会陷入无限循环。为了选择最有可能具有最大影响的流，可以计算流是交点的次数，并且可以选择不在多样化列表中具有最多出现的流。最后，如果未找到流，则将选择不在多样化列表中的TT流列表中第一个流。

[0059] TT偏移的生成(图5)

为了计算导致TT流对RC流的最小影响的偏移，可以规定将帧散布在超周期(即所有调度的TT消息周期的最小公倍数)内，以便减少聚集的TT流的初始突发。在这里，周期是10并且超周期HP是40。

[0060] 为了实现该目标，特别是在每个具有TT流的输出端口中，可以定义已调度的TT帧之间的增益函数，如图5中的示例中所图示。当查看0与超周期之间的调度的传输的时间轴时，我们将t_k^{end}表示为时间轴上的TT帧传输的结束，并将t_{k+1}^{start}表示为时间轴上的TT帧的下一次传输的开始。对于已经调度的每个TT帧k(不包括如上所述的标识的TT流的TT帧，其可以由TT流i表示)，增益函数(gain(t))由以下确定：

$$\forall t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t - t_k^{end}$$

$$\forall t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t_{k+1}^{start} - t$$

对于来自标识步骤的选择的TT流，其沿路线的传输时间可以进行如下修改：

-必须保持以下正确性条件中的至少一个，优选保持所有以下正确性条件：

°无争用限制

°路径相关限制

°同时中继站限制

°端到端传输限制

°应用程序级限制

°协议控制流限制

°特定领域的限制

°添加了用于标识的TT流的帧的优化条件，该条件驱动帧的放置尽可能接近该帧的最大增益。

[0061] 在[Steiner,W.2010."An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks."RTSS.]中详细描述了以上条件。

[0062] °可选地，可以添加新偏移与旧偏移不同的每个端口的条件。

[0063] 近似(图6)

以上提出的方法可能需要大量的优化函数。在每两个调度的TT帧之间，需要多个优化函数来定义增益的两个线性部分，即，来定义新偏移的上限和下限以及与增益相关联的值。当TT流的数量增加时，所需的断言数量也会增加，从而导致调度器的运行时间增加。

[0064] 因此，可以规定以下内容：仅考虑流i(要重新调度的流)的周期Ti，并且计算由当前调度的帧排除的值，即是否正在传输帧，或者是否帧间间隙太小以至于无法传输流i的帧。然后，对增益(优选地仅在可接受的时间上)求和，如图6中所图示。最后，进一步减少增益函数的线性部分的数量(这意味着减少断言的数量并因此减少计算时间)，剩余的增益函数可以近似为每个间隔仅保留最多两个线性函数[在两个已调度的帧之间，或者在0和一个帧之间，或者在一个帧和超周期之间(特别是，在0和超周期的结尾之间)]，如图6中所图示，此处具有Ti＝10并且超周期HP＝40。

[0065] 在图5和图6所描述的示例中，断言的数量已从24个减少到6个，这有可能改进定时性能。

[0066] 示例方法的详细解释

在下文中，将解释根据本发明的方法的具体示例。即使在本示例的上下文中将所有细节描述为强制性的，也可以在本发明的最一般范围内提供或实现所有细节。

[0067] 1理论背景

1.1 TT以太网

TT以太网是对当前在航空航天领域以及新兴工业自动化系统中的混合临界实时应用中使用的标准以太网的扩展。TT以太网特征在于具有代表不同临界级别的三种类型的业务：TT业务(用于具有保证的端到端等待时间和最小抖动的高临界业务)、具有确定的服务质量(QoS)保证的RC业务和非临界业务(即，BE业务)。TT以太网[7]标准基于使用全局时间同步来在精确的预定义时间(在作为全局计算的调度表的部分的本地调度表表格中进行编码)发送时间触发的(TT)帧，以确保最低的争用和延迟。因此，TT流由它们的大小、周期和每个输出端口中的偏移(应开始其传输的时间)定义。同步流在TT以太网网络中具有最高优先级，下一个优先级由TT流使用，两个接下来的优先级分别由AFDX RC_{HIGH}和RC_{LOW}业务使用，而其余4个最低优先级则为尽力而为(BE)通信(参见图7)保留。

[0068] 1.2基于SMT的TT以太网调度表合成

在这里，我们基于[11,6,10]简要描述基于SMT的方法，其用于计算TT以太网的通信调度表。

[0069] 可满足性模理论(SMT)是一种与SAT类似的众所周知的方法，用于基于背景理论(诸如线性整数算术($\mathcal{LA}(\mathbb{Z})$)或位向量($\mathcal{BV}$)[1]，[9]检查一阶逻辑公式的可满足性。当前最先进的TT以太网调度方法[6]使用SMT求解器以用于通过生成对表示必要和足够正确性条件的SMT求解器的上下文的断言来生成静态调度表。SMT求解器的结果是给定限制的解决方案，其表示每个设备上单个帧的偏移值。附加地，现代的SMT求解器(像Z3[2])提供了包括优化标准的可能性，所述优化标准在给定的(一个或多个)最小化或最大化条件的情况下找到优化解决方案。

[0070] 建立在[11]中的工作之上的我们的算法使用SMT求解器来基于固有的TT以太网技术限制和可选的用户限制来找出网络的调度表。Steiner[11]提出了一种增量回溯算法，我们也实现了该算法，该算法以小的增量拆分调度问题，一次调度流的子集。如果找到了子集的部分解决方案，则添加附加帧和限制，直到找到完整的调度表或不能找到部分问题的解决方案。在不可行的情况下，问题会被追溯，并且添加的子集的大小被增加。在最坏的情况下，该算法回溯到根，一个步骤中调度完整的帧集合。我们的算法遵循[11]中描述的算法，将从输入到SMT上下文的每个流依次添加。我们请读者参考[11]，用于对我们在我们方法中使用的基于SMT的TT以太网调度器的正确性限制进行描述和形式化。

[0071] 虽然基于SMT的方法保留了基于SAT或MiP的解决方案的最优属性，但主要缺点在于SMT求解器充当了黑匣子，而我们只能通过定义限制(断言)和优化标准来影响它。而且，我们被限制在一阶逻辑的表达性上。因此，基于最先进的SMT的调度器仅考虑TT流限制。结果，由调度的TT帧的放置可严重影响RC业务，并错过其要求的截止时间。

[0072] 我们提出了一种基于反馈的方法，该方法试图经由优化标准来驱动SMT求解器，以将TT帧放置在时间轴上，使得RC业务将遵循等待时间和线路阻塞要求。

[0073] 请注意，虽然提出的方法是针对TT以太网调整的，但基于主要反馈的方法也可以用于时间敏感的网络(TSN)中，其是用于工业通信系统的新标准化确定性以太网解决方案。对于TSN，可以使用我们提出的方法扩展现有的基于SMT的解决方案(例如[10])，以通过包括来自[12]的基于网络演算的AVB分析结果来考虑由CBS塑造的AVB业务。

[0074] 1.3网络演算

本文中详细描述的定时分析基于网络演算框架[8]，该框架用于计算延迟上限和线路阻塞限制。这些限制取决于由所谓的到达曲线α描述的业务到达(其代表在任何时间间隔内可以到达的最大数据量)，以及取决于由所谓的最小服务曲线β描述的资源可用性(其代表在任何时间间隔中可以发送的最小数据量)。

[0075] 定义1(到达曲线)[8]函数α(t)是具有输入累积函数A(t)的数据流的到达曲线，即，到时间t为止接收到的比特数，当且仅当：

$$\forall t, A(t) \leq (A \otimes {}^{1}\alpha)(t)$$

其中${}^{1}f \otimes g(t) = \inf_{0 \leq s \leq t}\{f(t-s) + g(s)\}$ 。

定义2(严格最小服务曲线)[8]函数β是具有输出累积函数A*的数据流的最小严格服务曲线，如果针对任何线路阻塞的周期]s，t]：$A^*(t) - A^*(s) \geq \beta(t-s)$(其中如果

$A(\tau) - A^*(\tau) > 0, \forall \tau \in]s,t]$，则]s，t]是所谓的线路阻塞的周期)

为了计算主要性能度量，我们需要以下结果：

定理1(性能限制)[8]考虑由与系统S(其提供了最小服务曲线β)交叉的到达曲线α限制的流F。在任何时间t获得的性能限制为：

线路阻塞：$\forall\, t:\ q(t) \le v(\alpha,\beta)$ (其中v是最小垂直距离)

延迟：$\forall\, t:\ d(t) \le h(\alpha,\beta)$ (其中h是最小水平距离)

输出到达曲线：$\alpha^*(t) = (\alpha \oslash \beta)(t)$ (其中$f \oslash g(t) = \sup_{s \ge 0}\{f(t+s) - g(s)\}$)

定理2(仅一次的级联支付突发)[8]假设流与具有相应服务曲线β_1和β_2的两个服务器交叉。由两个服务器的级联组成的系统提供服务曲线$\beta_1 \otimes \beta_2$。

[0076] 定理3(剩余服务曲线)[3]考虑一个系统，其中严格服务曲线β和m流$f_1, f_2, \cdots, f_m$与它相交。f_i的最大分组长度是$l_{i,max}$，并且f_i是α_i受限的。所述流由NP-SP策略调度，其中f_i的优先级>f_j的优先级⇔i<j。对于每个$i \in \{1,\ldots,m\}$，f_i的严格服务曲线由下式给出：

$$(\beta - \sum_{j<i} \alpha_j - \max_{k \ge i} l_{k,max})_{\uparrow}$$

其中$g\uparrow(t) = \max\{0, \sup_{0<s<t} g(s)\}$

业务合同通常使用漏桶整形器来执行，即，业务流是(r，b)受限的，其中r和b是最大速率和突发，并且到达曲线为α(t)＝r•t+b。最小服务曲线的常见模型是速率等待时间曲线$\beta_{R,T}$，定义为$\beta_{R,T}(t) = R \bullet (t-T)^+$，其中R是输出传输容量，T是系统等待时间，以及$(x)^+$表示x与0之间的最大值。所产生的输出到达曲线然后为：$\alpha^*(t) = r \bullet (t+T)+b$

漏桶整形器的改进是考虑输入端口的链路能力，以便获得更准确的流的输入到达曲线[5]。例如，从具有容量为C_{in}的链路以最大突发b和速率r到达的流具有输入到达曲线$\alpha(t) = \min(C_{in} \bullet t, r \bullet t+b)$。

[0077] [14、13、4]中已经介绍了考虑到被TT业务类调度影响的针对TT以太网的RC业务类分析的扩展。

[0078] 2反馈回路

最佳解决方案将是将网络演算框架集成到SMT求解器中，以直接考虑RC业务的限制。然而，由于网络演算的非线性，这是不可能的。特别地，如[14]中所述的TT帧的到达曲线的计算利用了若干最小化、最大化和上限值来计算TT流的整体到达曲线。

[0079] 当前最先进的基于SMT的调度器不考虑RC流。然而，由于TT业务具有较高的优先级，因此时间轴上的TT帧的放置可能对由RC流经历的延迟具有重大影响。我们提出的主要思想(下面更详细地描述)是使用反馈循环，该反馈循环使用RC网络演算分析来检查TT调度，并在必要时重新调度有问题的流。

[0080] 首先，使用SMT求解器的优化函数对所有TT流进行调度，以扩展TT帧。我们的观察是，均匀间隔开的TT帧放置导致对RC流的影响较小。因此，我们建立优化度量，该度量尝试在时间轴上均匀间隔开TT帧，并使用优化目标经由SMT求解器来驱动TT流的调度和重新调度。

[0081] 其次，呼叫RC分析以确定RC业务是否满足截止时间要求。如果是这种情况，则算法

停止。如果不是这种情况，则算法使用SMT的优化函数标识最有可能引起延迟的流，并尝试找到更好的偏移，即重新调度。

[0082] 重复该第二步骤，直到找到适当的调度表或达到停止条件为止。

[0083] 这种解决方案的附加优点在于，它是对现有实现方式的补充，并且可以容易地集成到现有工具中。不需要修改现有的限制，只需添加优化目标并实施反馈循环。

[0084] 2.1常规算法

常规算法在Alg.1中详细介绍。为了修改偏移的值，我们提出向SMT调度器添加优化限制，该限制在第1.2.3节中的函数smtComputeOffset(flow)中实现。特别地，我们提出了两种用于计算优化函数的方法。它们在第1.2.4和第1.2.5节中详细介绍。

[0085] 首先，在Alg.1中，我们使用smtComputeOffset(flow)从第1行到第3行为每个流(从最小到最大周期排序)设置初始偏移。然后，我们检查第4行中是否满足RC截止时间。如果不是这种情况，我们尝试在第8行和第9行中修改由函数findBestFlow()(在第1.2.2节中描述)选择的流的偏移。

[0086] 如果已经探索了新的偏移集合(第10行)，我们使用while循环来探索可能的偏移，直到已经测试所有流或已经找到新的集合为止(第11行)。尽管在第11行中没有找到可接受的解决方案，我们将流添加到多样化列表中，寻找不在多样化列表中的新的流，并像以前一样计算新的偏移(第12至15行)。如果尚未探索新的偏移集合，则重置多样化列表(第16至18行)。最后，我们检查是否现在满足RC截止时间，并更新探索的偏移的列表(第19行和第20行)。

算法1:常规算法

```
Require: flows_TT, flows_RC
 1: for flow in flows_TT do
 2:    smtComputeOffset(flow)
 3: end for
 4: rcDeadlinesFulfilled=computeRCDelays(flows_RC)
 5: listExploredOffset=[]
 6: diversification=[]
 7: while rcDeadlinesFulfilled==False and diversification.size < flows_TT.size do
 8:    flow = findBestFlow()
 9:    smtComputeOffset(flow)
10:    if getCurrentOffsets() in listExploredOffset then
11:       while getCurrentOffsets() in listExploredOffset and
          diversification.size < flows_TT.size do
12:          diversification.add(flow)
13:          flow = findBestFlow(diversification)
14:          smtComputeOffset(flow)
15:       end while
16:    else
17:       diversification==[]
18:    end if
19:    rcDeadlinesFulfilled=computeRCDelays(flows_RC)
20:    listExploredOffset.add(getCurrentOffsets())
21: end while
```

[0087] 2.2findBestFlow()

TT流的影响由TT业务的到达曲线表示，该曲线使用[14]中详述的公式计算。主要思想是在所有可能的情况下计算TT流的影响并保持最大值，如图4中所图示。虚线是不同的情况，而平线阶梯是虚线的最大值，表示最终的TT流最大到达曲线，表示为$\alpha_{TT}^{k}(t)$。

[0088] 根据这个阶梯到达曲线，我们能够近似线性到达曲线。线性曲线的比率与一周期内阶梯曲线的比率相同。关于线性曲线的初始突发，计算诸如线性曲线总是优先于或等于阶梯曲线，具有至少一个交点，如图4中所图示。结果，通过修改与阶梯交点相关联的偏移的值，我们可能能够减小线性曲线的突发的值，并且因此，减小TT业务对RC流的影响。交点在findIntersections()函数中计算。

算法2:findBestFlow()

Require: $flows_{TT}$,$flows_{RC}$, $Data_{paths}$,diversification
Ensure: bestFlow
1: listCard={}
2: **for** flowTT in $flows_{TT}$ **do**
3: listCard[flowTT]=0
4: **end for**
5: **for** flowRC in $flows_{RC}$ **do**
6: **for** path in flowRC paths **do**
7: **for** port in path **do**
8: flowsTTInter=findIntersections()
9: **for** flowTT in flowsTTInter **do**
10: listCard[flowTT]+=1
11: **end for**
12: **end for**
13: **end for**
14: **end for**
15: bestFlow=NULL
16: **if** not isEmpty(listCard) **then**
17: bestFlow=findMaxCardinal(listCard,diversification)
18: **end if**
19: **while** bestFlow==NULL or bestFlow in diversification **do**
20: bestFlow = $flows_{TT}$.getNext()
21: **end while**

[0089] 为了选择最有可能具有最佳影响的流(算法在Algo.2中详述)，我们计算了流是交点的次数(从第5行到第14行)，并且我们选择出现次数最多的不在多样化列表中的流，在第17行中。最后，如果未找到流，我们在TT流的列表中选择不在多样化列表中的第一个流，在第19行到21行中。

[0090] 2.3smtComputeOffset()

我们考虑具有超周期HP的TT流的集合(计算为所有流周期的最小公倍数)。我们的目标是利用最大帧大小MFS_i和周期T_i定义流i的偏移x。来自[11]的所有先前存在的限制被执行。在本节中，我们仅呈现为优化偏移而添加的限制。附加地，如果已经调度了流，则我们还添加了x必须与当前偏移不同的限制。

[0091] 我们将C_{out}表示为输出端口的容量。对于已经调度的每个TT帧k，我们定义t_k^{start}和t_k^{end}为传输的开始和结束。我们具有$m_k = \frac{t_k^{end}+t_{k+1}^{start}}{2}$。函数newSymbolicInt()创建了符号

变量，该变量将被SMT求解器用作未知变量。

[0092] 在Alg.3中，对于TT流i的每个路径中的每个端口，我们计算定义增益的限制，然后将它们添加到第6行中的现有SMT限制中。最后，偏移由第7行中的SMT求解器计算。在Algo.4中描述了端口(即第4行)中的新限制的计算。

算法3：用于TT流i的路径的限制计算

Require: *TT flow i, path*
Ensure: $Offsets_i$
1: sumGain=newSymbolicInt()
2: constraints=preexisting constraints from [11]
3: **for** port in path **do**
4: (*constraints*, *sumGain*)=computeConstraintInPort (*TT flow i*, *constraints*, *sumGain*, $Data_{port}$)
5: **end for**
6: constraints.add(maximize(*sumGain*))
7: $Offsets_i$=smtSolve(constraints)

[0093] 2.4 computeConstraintsInPort()：变型1

为了计算导致TT流对RC流的最小影响的偏移量，我们必须在超周期内扩展帧，以便减少聚集的TT流的初始突发。

[0094] 为了实现该目标，在每个具有TT流的输出端口中，我们定义了已调度的TT帧之间的增益函数，如图5中所图示，其中$T_i=10$并且超周期HP=40。我们将t_k^{end}表示为帧传输的结束并且将t_{k+1}^{start}表示为下一传输的开始。对于每个已调度的TT帧$k\neq i$，增益函数由以下确定：

$$\forall t_k^{end} \leq t < \frac{t_k^{end}+t_{k+1}^{start}}{2}, gain(t) = t - t_k^{end}$$

$$\forall t_{k+1}^{start} \geq t \geq \frac{t_k^{end}+t_{k+1}^{start}}{2}, gain(t) = t_{k+1}^{start} - t$$

Alg.4描述了图5中所图示的增益函数的定义。特别地，第7行定义了三角形的外部边界，第8行(对应的第9行)定义了三角形的右侧(对应的左侧)。

[0095] 然而，由于SMT优化限制只能被定义为$\mathcal{LA}(\mathbb{Z})$中的线性曲线，因此该方法需要大量断言。因此，在两个帧之间，算法4：在端口p中的限制计算：computeConstraintsInPort()

Require: *TT flow i*, *constraints*, *sumGain*, $Data_p$
Ensure: *constraints*, *sumGain*
x= newSymbolicInt()
$J = \frac{HP}{T_i}$
for j in range(0,J) **do**
  $gain_j$= newSymbolicInt()
  for frame k in TT flows **do**
    if $t_k^{end} >= j \cdot T_i$ and $t_{k+1}^{start} <= (j+1) \cdot T_i$ **then**
      $A = \text{And}(x + j \cdot T_i + \frac{MFS_i}{C_{out}} <= t_k^{end}, t_{k+1}^{start} <= x + j \cdot T_i)$
      $B = \text{And}(x + j \cdot T_i >= m_k, gain_j = t_{k+1}^{start} - x - j \cdot T_i)$
      $C = \text{And}(x + j \cdot T_i <= m_k, gain_j = x + j \cdot T_i - t_k^{end})$
      $constraints.add(\text{And}(A, \text{Or}(B, C)))$
      $sumGain+ = gain_j$
    end if
  end for
end for

必须有6个断言来定义增益的两个线性部分，即定义x的上限和下限以及与该增益相关联的值。当TT流的数量增加时，所需断言的数量也增加，从而导致调度器的运行时间增加[6]。

[0096] 因此，我们提出了算法的第二种变型，以通过预处理减少断言的数量，并且因此改进我们方法的运行时间。

[0097] 2.5 computeConstraintsInPort()：变型2

改进的主要思想是仅考虑流i的周期T_i并计算由当前调度的帧所排除的值，即，如果正在传输帧，或者如果帧间间隙太小以至于无法传输流i的帧。然后，对增益(仅在可接受的时间上)求和，如图6中所图示。最后，进一步减少增益函数的线性部分的数量(这意味着减少断言的数量并因此减少计算时间)，我们近似剩余的增益函数，以使每个间隔最多仅保留两个线性函数，如图6中所图示，其中T_i＝10并且HP＝40。

[0098] 在图5和图6所描述的示例中，断言的数量已从24个减少到6个，这将改进定时性能。在下一节中，我们将呈现实验结果以验证这些概念并评估两个变型之间的性能增益。

[0099] 3性能分析

在本节中，我们基于真实世界使用情况进行性能分析，以便比较我们解决方案的两个选择的变型。首先，我们提出初步假设，例如，流的模型、TT以太网交换机和终端系统以及延迟计算。然后，在提出了我们的案例研究之后，我们对TT以太网网络的两种方法进行比较。

[0100] 3.1初步和假设

交换机和终端系统模型：我们考虑了图7中描述的TT以太网交换机架构。输入端口延迟是帧i达到速率C_{in}所需的时间量：$\frac{MFS_i}{C_{in}}$。我们考虑了在已经完全接收到帧之后交换机开始时的延迟。转发过程由最小(最佳情况)和最大(最坏情况)延迟(表示为WCD_{fwd}^n和BCD_{fwd}^n)定义，在交换机或终端系统中n∈{es,sw}。所有流在交换机和终端系统中使用置乱集成策略，即TT帧可以被较低优先级的帧延迟。

[0101] TT以太网输出端口：TT业务对RC业务的影响使用[14]中提出的输入到达曲线来计算。然后，使用Th.3计算服务曲线。

[0102] 业务模型：为了计算每个节点(输出端口、交换机sw或终端系统es)内的延迟限制，我们在以下假设下使用Th.1：(i)在节点n的输入处业务流为阶梯到达曲线。对于流i，我们定义最大帧大小MFS_i和带宽分配间隙BAG_i(周期并且通常还包括截止时间)。由业务源发送的初始到达曲线为$\alpha_I(t) = \sum_{i\in I} MFS_i \cdot \lceil \frac{t+J_i}{BAG_i} \rceil$。对于每个类别I，聚集业务在节点$n\in\{es,sw\}$中具有输入到达曲线：$\alpha_I^n(t) = \min\{C_{I,in}^n \cdot t, \sum_{i\in I, i\in n} \alpha_i^n(t)\}$，其中最大输入速率$C_{I,in}^n$是节点n的输入链路的容量$C_{in}$与类别I的业务交叉的总和，并且$\alpha_i^n(t)$是节点n中的流i的输入到达曲线。$\alpha_i^n(t)$是如图8中所图示的阶梯曲线，其中$WCD_{prec}$是先前节点中的延迟。我们考虑在生成节点n中，$C_{I,in}^n = +\infty$；(ii)由节点n向业务类别I提供的服务曲线为如图8中所图示的阶梯曲线。该结果来自使用具有阶梯到达曲线的Th.3。

[0103] 延迟限制计算：利用这种方法，我们可以使用BCD_{fwd}^n在输出端口中获得更紧密的输入到达曲线。节点$n\in\{es,sw\}$中的输出端口port中的聚集的流I的输入到达曲线为：

$$\alpha_I^{port}(t) = \min\{C_{I,in}^n \cdot (t+\delta_{fwd}^n), \alpha_I^n(t+\delta_{fwd}^n)\}$$

其中$\delta_{fwd}^n = WCD_{fwd}^n - BCD_{fwd}^n$。然后，根据Th.1，我们可以将最坏情况的延迟限制计算为$\alpha_I^{port}(t)$和$\beta_I^{port}(t) = R_I^{port} \cdot (t - T_I^{port})^+$之间的最大水平距离。交换机(或终端系统)n中的延迟然后为：$WCD_I^n = WCD_I^{port} + WCD_{fwd}^n$，以及输出到达曲线为$\alpha_I^{n,*}(t) = \alpha_I^n(t + \delta_{fwd}^n + WCD_I^{port})$。

[0104] 端到端延迟限制：最后，流的端到端延迟通过对沿终端系统、输入端口、交换机和沿流的路径的链路中路径的延迟求和而获得。

[0105] 3.2案例研究

我们考虑图9中所描绘的真实世界项目AERO1(请注意现实的测试案例已经由于合同义务而匿名)，它具有3个终端系统，即PG1、PG2和PX2，以及3个开关，即SW0、SW1、SW2。表1.1中描述了转发延迟，并且链路容量为100Mbps。

[0106] 我们考虑TT(相应的RC)业务的12个流，其中冗余级别为3，即36VL，其中相同周期(相应的BAG)为4ms，以及MFS＝1518字节。RC业务具有的默认最大初始抖动为100μs。在PG1中利用PG2和PX2作为目的地生成六个VL。在PG2中利用PG1和PX2作为目的地生成其它六个。流的默认截止时间等于周期。

节点 n	WCD_{fwd}^n (s)	BCD_{fwd}^n (s)
终端系统	$2.43 \cdot 10^{-6}$	$2.18 \cdot 10^{-6}$
交换机	$2.50 \cdot 10^{-6}$	$2.41 \cdot 10^{-6}$

[0107] 表1.1转发延迟

为了评估两种方法，我们通过变化两个参数(即TT截止时间和RC抖动)来探索不同的场景，如表1.2中所图示。如业务模型中突出显示的那样，增加初始抖动增加了所生成业务的输入到达曲线，从而导致相同TT偏移的RC端到端延迟的增加。因此，增加RC抖动也增加了对RC截止时间和延迟的限制。我们选择变化抖动，因为与MFS和BAG相比，抖动是次要的参数，从而允许我们保持接近默认场景。

[0108] 各种场景的结果将突出显示每种方法的优点和缺点。

[0109] 我们专注于最大迭代次数n_{max}、专注于最大执行时间t_{max}、专注于发现最佳结果的迭代n_{best}、专注于找到最佳结果t_{best}所需的时间以及专注于可调度性sched。我们将可调度性定义为可调度流的百分比。附加地，为了确保算法在可接受的时间帧内结束，我们在Algo.1的第7行中添加了限制以将迭代次数限制为2000。

场景	TT截止时间(s)	RC抖动(s)
默认	0.004	$100 \cdot 10^{-6}$
1.1	0.004	$800 \cdot 10^{-6}$
1.2	0.004	$900 \cdot 10^{-6}$
1.3	0.004	$1000 \cdot 10^{-6}$
1.4	0.004	$1100 \cdot 10^{-6}$
1.5	0.004	$1200 \cdot 10^{-6}$
2.1	0.002	$100 \cdot 10^{-6}$
2.2	0.002	$300 \cdot 10^{-6}$
2.3	0.002	$400 \cdot 10^{-6}$

表1.2场景描述

[0110] 3.3结果

首先，我们在没有优化方法的情况下(即在当前实现方式的情况下)计算所有9种场景的结果。结果表明计算是在大约$t_{max}=1.80$s内完成的，并且36个RC VL中有18个是不可调度的，即，它们的延迟大于其截止时间。因此，在没有针对所有9种场景的优化方法的情况下，RC流的可调度性为50％。

[0111] 在表1.3中，我们呈现了找到可接受解决方案的场景的结果，即，所有RC流是可调度的。

[0112] 在表1.4中，我们呈现了未找到可接受解决方案的场景的结果，即，并非所有RC流是可调度的。

场景	方法	n_{max}	t_{max} (s)
默认	1	1	101
默认	2	1	**2.5**
1.1	1	1	95
1.1	2	3	**3**
1.2	1	4	**174**
1.2	2	691	256
1.3	1	4	**170**
1.3	2	1110	508
2.1	1	8	46
2.1	2	1	**2.3**
2.2	1	1841	3934
2.2	2	123	**76**

表1.3:针对两种方法具有100％可调度性的结果

场景	方法	n_{max}	t_{max} (s)	n_{best}	t_{best} (s)	sched
1.4	1	55	604	55	604	**100%**
1.4	2	2000	1190	1053	616	83%
1.5	1	2000	8621	1	115	50%
1.5	2	2000	1180	1535	882	**66%**
2.3	1	2000	**3968**	255	891	83%
2.3	2	2000	**1023**	660	**350**	83%

表1.4针对两种方法没有100％可调度性的结果

[0113] 首先，关于默认场景，我们从表1.3中可以看到，两种方法在第一次迭代($n_{max}=1$)后找到了解决方案。因此，可调度性从50％增加到100％。附加地，t_{max}表明，如预期的那样，方法2比方法1更快(在场景2.2中快达51倍)，并且比没有优化的当前方法稍慢。

[0114] 当研究所有9个场景时，我们注意到方法1倾向于在较少的迭代中找到最佳解决方案(即场景1.1、1.2、1.3、1.4和2.3中的n_{max}和n_{best})，但并非总是如此(即场景2.1和2.2)。这很可能由于对方法2中增益函数的完成的近似。

[0115] 然而，较少的迭代并不一定导致更好的定时性能(即场景1.1和2.3中的t_{max}和t_{best})，因为如预期的那样，利用方法2的迭代比利用方法1的迭代更快，例如在场景2.3中快约4倍，以及在默认场景中快达40倍。

[0116] 当利用两种方法的可调度性不是100％时(参见表1.4)，我们可以看到最佳方法(即最佳可调度性)取决于场景：对于场景1.4的方法1和对于场景1.5的方法2。这再次突出显示了任何一种方法都可以找到最佳偏移。

[0117] 另外，比较用于探索所有2000个选项的最长时间是有趣的：在场景1.5中，第一种方法花费了$t_{max}=8621s$来完成，而方法2仅花费1180s来评估没有解决方案给出100％的可调度性。附加地，利用方法1(即最长的完成时间)的可调度性低于利用方法2(即最快的方法)的可调度性。

[0118] 最后，我们可以得出结论，虽然方法1实现不太复杂，并且通常以较少的迭代找到结果，但是方法2是在可接受的时间帧中找到解决方案的最佳方法。附加地，我们看到我们基于反馈的方法能够生成TT流的调度表，使得针对所有9种场景增加了RC业务的可调度性(甚至达到100％)。

[0119] 总而言之，本发明涉及一种用于生成用于网络中的流的传输的调度表的方法，所述流包括时间触发的TT流和速率受限的RC流，每个这样的流包括消息，分别是TT消息和RC消息，其中网络包括像节点和星形耦合器之类的部件，或者在网络中不同部件之间传送消息的其他部件，其中网络是时间触发的TT网络或时间敏感的TSN网络，

以及其中所述网络的部件根据所述调度表并基于全局网络范围的时间来传送时间触发的消息，以及

其中所述部件传送速率受限的RC消息，其中对于所述TT和RC消息中的每一个规定实时要求，以及

其中对于RC消息中的每个，所规定的实时要求包括第一要求，该第一要求是针对消息的传输的最坏情况的端到端延迟限制。

[0120] 该方法可以包括以下步骤：

-A1对于满足第一条件的TT传输时间的集合进行第一搜索，

第一个条件是，在将TT传输时间的集合应用于TT消息的情况下，所有TT消息的实时要求被满足，

-A2：对所找到的TT传输时间的集合是否满足第二条件进行第一确定，其应用于在第一搜索期间找到的TT传输时间的集合，

第二条件是，在将传输时间的集合应用于TT消息的情况下，流的第一集合的消息的实时要求被满足，其中所述流的第一集合是RC流的子集，

并且其中，在所找到的TT传输时间的集合不满足第二条件的情况下，步骤A1之后包括以下步骤：

-A11：对在所找到的集合中的TT传输时间的第二集合进行第一选择，TT传输时间的第二集合是鉴于满足流的第一集合的实时要求而需要修改的TT传输时间，

-A12：对TT传输时间的第二集合的修改的传输时间进行计算，该计算的结果是满足所有TT消息的第一条件的TT传输时间的新集合。

[0121] -A3：在所找到的TT传输时间的集合不满足第二条件的情况下，重复步骤A1至A3，直到找到满足第二条件的传输时间的集合为止，以及

-A4：生成调度表，使得调度表包括所找到的TT传输时间的集合。

[0122] 可以规定，第一搜索A1包括第二选择，该第二选择用于在满足第一条件的多个TT传输时间的集合中选择满足第一条件的多个TT传输时间的集合中的一个，其优化第一优化函数，其中利用像可满足性模理论(SMT)求解器或任何其他求解器的求解器来检查第一优化函数。

[0123] 可以规定，步骤A11包括以下步骤：

-A111：超周期HP的计算，所述HP是所有TT流的所有流周期的最小公倍数，

-A112：基于超周期对流的第一集合进行第三选择，以及

-A113：对于被标识为网络节点的输出端口的每个输出端口，使得流的第一集合中的至少一个流经过考虑的输出，计算增益函数。

[0124] 可以规定，步骤A113包括以下步骤：

-A1131：对于已调度的TT帧的增益函数的计算依据以下公式，其中t^{end}_k是帧传输的结束，而t^{start}_{k+1}是下一次传输的开始：

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2}, gain(t) = t_{k+1}^{start} - t$$

其中步骤A12包括以下步骤：

-A121：对于至少一个TT流，基于针对已调度的TT帧计算的增益函数，计算至少一个TT传输时间，针对所考虑的流的每个周期以及针对所考虑的流经过的至少一个输出端口，所述步骤A121包括以下步骤：

-A1211：通过对经过所考虑的输出端口的所有流的增益函数求和来计算总增益函数，以及

-A1212：选择时间t0作为在输出端口中的TT传输时间，针对该时间t0，总增益函数的值在所考虑的流的周期内最大。

[0125] 可以进一步规定，步骤A1211包括

-A12111：仅针对所考虑的流的周期计算增益函数，以及

-A12112：对总增益函数进行近似。

[0126] 可以规定，步骤A1由基于网络演算NC框架的算法指导。

[0127] 可以规定，步骤A2包括存储所找到的TT传输时间的集合的记录。

[0128] 可以规定，步骤A2包括以下步骤：

-A21：对于所找到的集合，计算被标识为网络节点的输出端口的每个输出端口的阶梯曲线，使得流的第一集合经过所考虑的输出端口，

-A22：对于在步骤A21中考虑的每个输出端口，阶梯曲线由线性曲线近似，

-A23：对于在步骤A21中考虑的每个输出端口，阶梯曲线与线性曲线的相交，

-A24：对于在步骤A21中考虑的输出端口，基于阶梯曲线与线性曲线的交点确定经过输出端口的流的影响，以及

其中，步骤A11中的第一选择包括基于被量化为大的所述流的影响来标识流的第一集合中的流中的一个。

[0129] 本发明还涉及一种计算机网络，其中该网络是时间触发的TT网络或时间敏感的TSN网络，其中所述计算机网络包括像节点和星形耦合器之类的部件，或者在系统中不同部件之间传送消息的其他部件，其中所述部件传送包括时间触发的TT流和速率受限的RC流的流，每个这样的流包括消息，分别是TT消息和RC消息，

以及其中所述网络的部件根据调度表并基于全局网络范围的时间来传送时间触发的消息，并且其中，所述部件传送速率受限的RC消息，其中对于所述TT和RC消息中的每个规定了实时要求，其中网络适配成包括用于生成利用上述方法生成的调度表的装置。

[0130] 本发明进一步涉及一种计算机程序，其包括程序代码装置，所述程序代码装置用于当所述计算机程序在计算机上运行时执行所述方法的步骤。

[0131] 最后，本发明涉及一种计算机程序产品，其包括存储在计算机可读介质上的程序代码装置，用于当所述计算机程序产品在计算机上运行时执行该方法。

[0132] 参考文献

[1]Clark Barrett、Roberto Sebastiani、Sanjit Seshia和Cesare Tinelli.Satisfiability modulo theories.在可满足性的手册中.第185卷.IOS出版社.2009(Clark Barrett,Roberto Sebastiani,Sanjit Seshia,and Cesare Tinelli.Satisfiability modulo theories.In Handbook ofSatisfiability,volume 185.IOS Press,2009)。

[2]Nikolaj Bjørner、Anh-Dung Phan和Lars Fleckenstein.vz-an optimizing SMT solver.在会刊TACAS中.Springer.2015(Nikolaj Bjørner,Anh-Dung Phan,and Lars Fleckenstein.vz-an optimizing SMT solver.In Proc.TACAS.Springer,2015)。

[3]Anne Bouillard、Laurent Jouhet和Eric Thierry.Service curves in Network Cal-culus:dos and don'ts.研究报告.INRIA.2009(Anne Bouillard,Laurent Jouhet,and Eric Thierry.Service curves in Network Cal-culus:dos and don'ts.Research report,INRIA,2009)。

[4]M.Boyer、H.Daigmorte、N.Navet和J.Migge.Performance impact of the inter-

actions between time-triggered and rate-constrained transmissions in TTEthernet.在会刊ERTS中.2016(M.Boyer,H.Daigmorte,N.Navet,and J.Migge.Performance impact of the inter-actions between time-triggered and rate-constrained transmissions in TTEthernet.In Proc.ERTS,2016)。

[5]Marc Boyer、JörnMigge和Nicolas Navet.An efficient and simple class of functions to model arrival curve of packetised flows.在会刊WCTT中.2011(Marc Boyer,JörnMigge,and Nicolas Navet.An efficient and simple class of functions to model arrival curve of packetised flows.In Proc.WCTT,2011)。

[6]Silviu S.Craciunas和Ramon Serna Oliver.Combined task-and network-level scheduling for distributed time-triggered systems.实时系统.52(2).2016(Silviu S.Craciunas and Ramon Serna Oliver.Combined task-and network-level scheduling for distributed time-triggered systems.Real-Time Systems,52(2),2016)。

[7]Hermann Kopetz、Astrit Ademaj、Petr Grillinger和Klaus Steinhammer.The Time-Triggered Ethernet(TTE)Design.在会刊ISORC中.2005(Hermann Kopetz,Astrit Ademaj,Petr Grillinger,and Klaus Steinhammer.The Time-Triggered Ethernet(TTE) Design.Proc.ISORC,2005)。

[8]J.Y.Le Boudec和P.Thiran.Network calculus:a theory of deterministic queuing systems for the internet.Springer-Verlag.2001(J.Y.Le Boudec and P.Thiran.Network calculus:a theory of deterministic queuing systems for the internet.Springer-Verlag,2001)。

[9]Roberto Sebastiani.Lazy satisfiability modulo theories.JSAT,3(3-4):141-224,2007。

[10]Ramon Serna Oliver、Silviu S.Craciunas和Wilfried Steiner.IEEE 802.1Qbv Gate Control List Synthesis using Array Theory Encoding.在会刊RTAS中.IEEE.2018(Ramon Serna Oliver,Silviu S.Craciunas and Wilfried Steiner.IEEE 802.1Qbv Gate Control List Synthesis using Array Theory Encoding.In Proc.RTAS.IEEE,2018)。

[11]Wilfried Steiner.An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks.在会刊RTSS中.IEEE.2010(Wilfried Steiner.An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks.In Proc.RTSS.IEEE,2010)。

[12]L.Zhao、P.Pop、Z.Zheng和Q.Li.Timing analysis of AVB traffic in TSN networks using network calculus.在会刊RTAS中.2018(L.Zhao,P.Pop,Z.Zheng,and Q.Li.Timing analysis of AVB traffic in TSN networks using network calculus.In Proc.RTAS,2018)。

[13]L.X.Zhao、H.G.Xiong、Z.Zheng和Q.Li.Improving worst-case latency analysis for rate-constrained traffic in the time-triggered ethernet network.IEEE通信快报.18(11).2014(L.X.Zhao,H.G.Xiong,Z.Zheng,and Q.Li.Improving worst-case latency analysis for rate-constrained traffic in

the time-triggered ethernet network.IEEE Communications Letters,18(11),2014)。

[14]Luxi Zhao、Paul Pop、Qiao Li、Junyan Chen和Huagang Xiong.Timing analysis of rate-constrained traffic in TTEthernet using network calculus.实时系统.53(2):254-287.2017(Luxi Zhao,Paul Pop,Qiao Li,Junyan Chen,and Huagang Xiong.Timing analysis of rate-constrained traffic in TTEthernet using network calculus.Real-Time Systems,53(2):254-287,2017)。

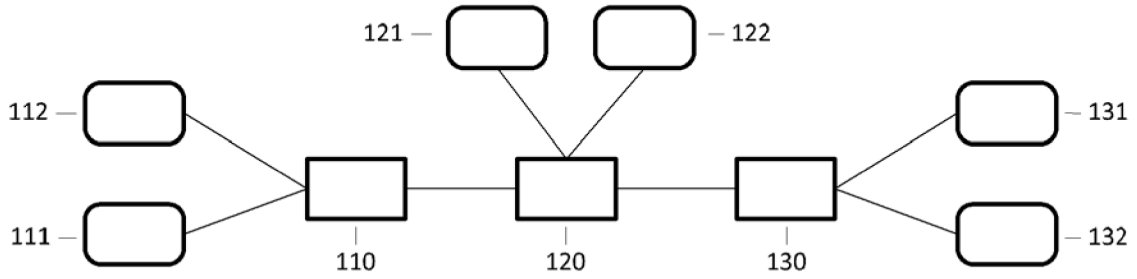

121
122
112
131
111
132
110
120
130

图 1

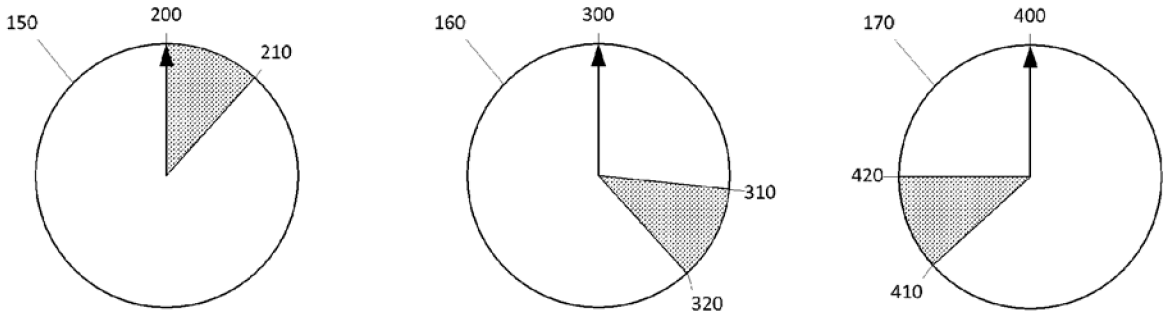

150
200
210
160
300
310
320
170
400
420
410

图 2

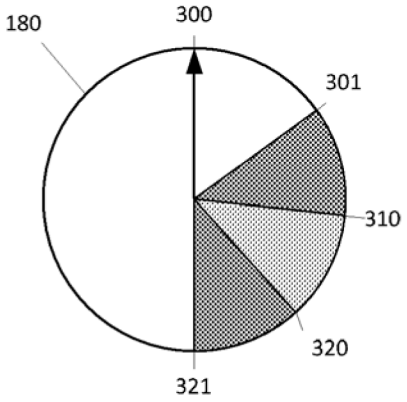

180
300
301
310
320
321

图 3

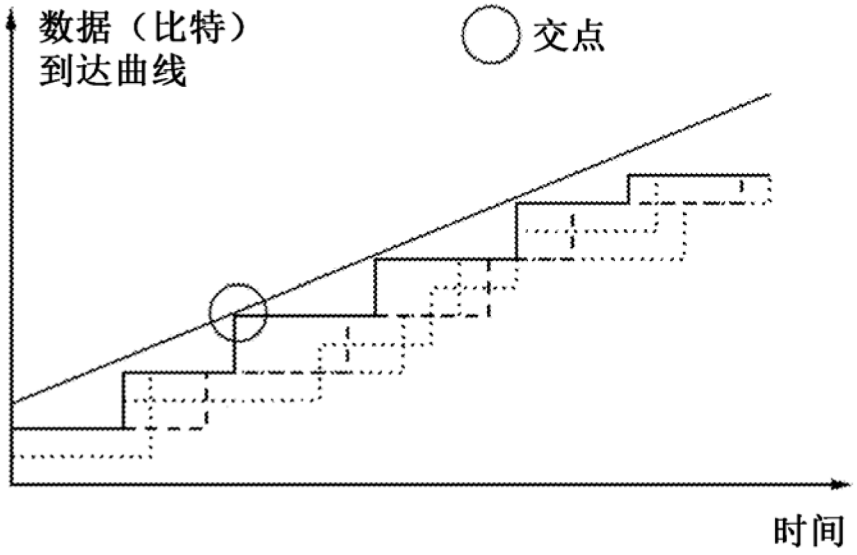


图 4

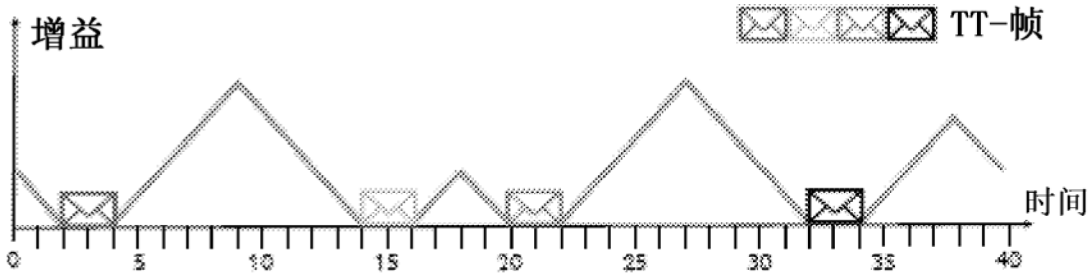


图 5

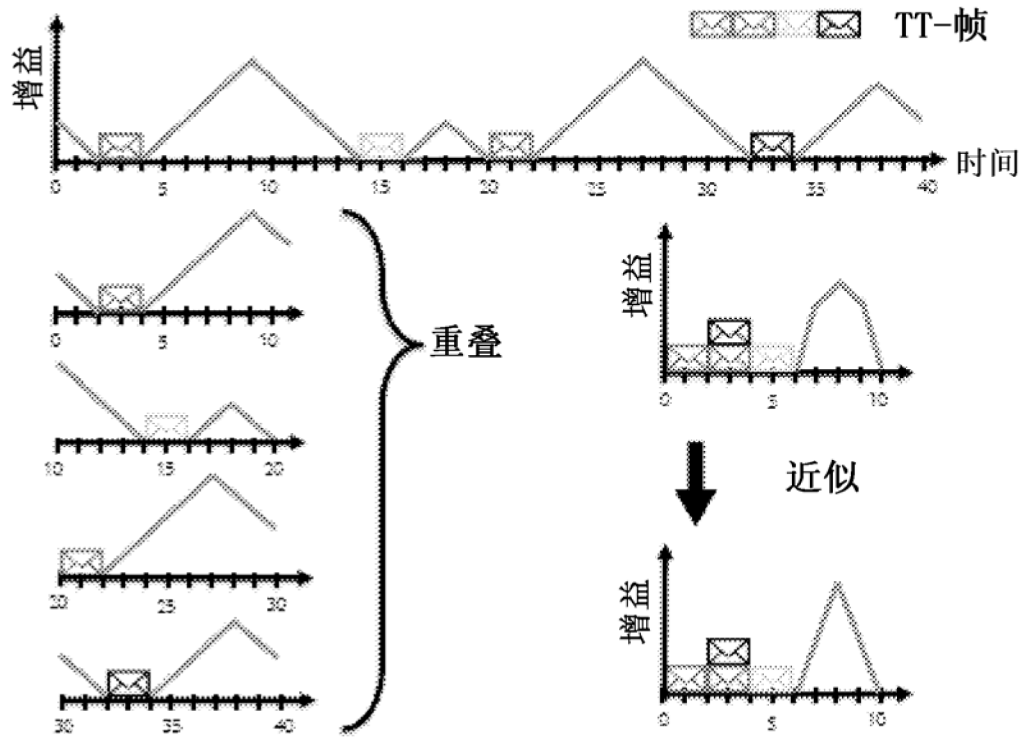

TT-帧
增益
时间
0
5
10
15
20
25
30
35
40
重叠
增益
近似
增益

图 6

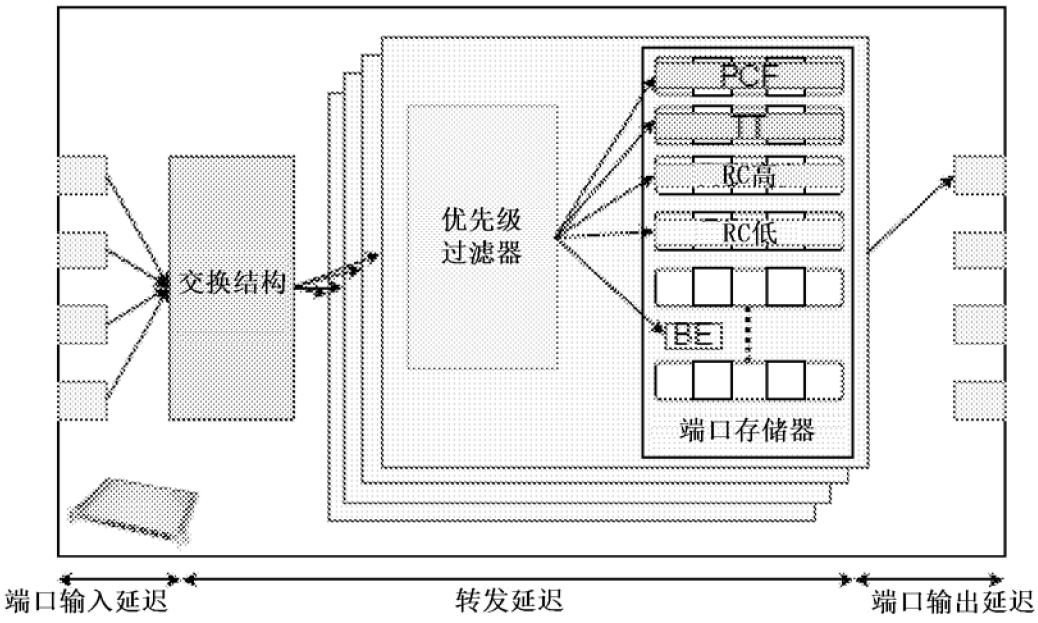

PCF
TT
RC高
RC低
BE
优先级
过滤器
交换结构
端口存储器
端口输入延迟
转发延迟
端口输出延迟

图 7

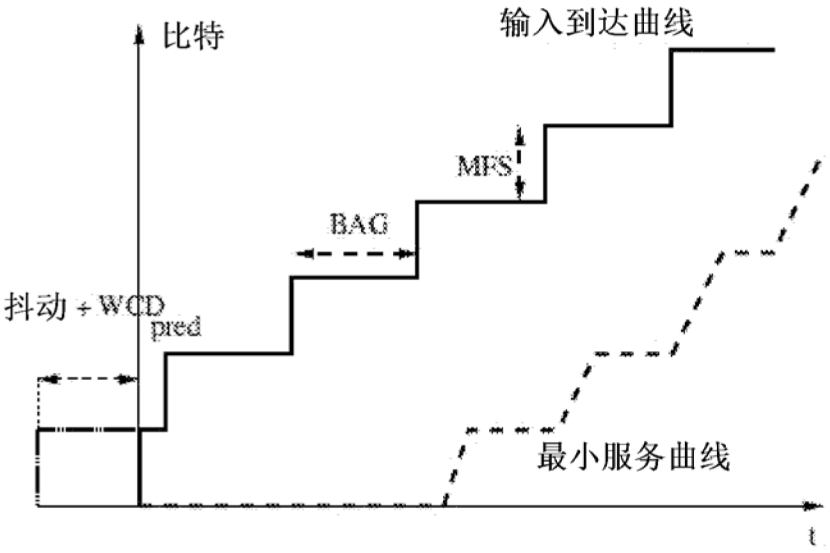


图 8

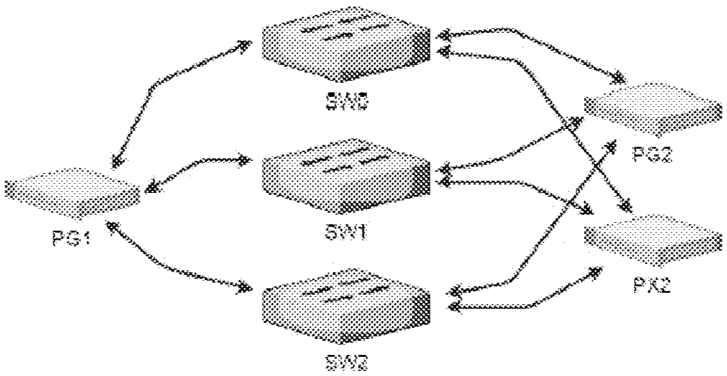


图 9