(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104170334 B
(45)授权公告日 2017.11.07

(21)申请号 201380003841.1

(22)申请日 2013.04.18

(65)同一申请的已公布的文献号
申请公布号 CN 104170334 A

(43)申请公布日 2014.11.26

(30)优先权数据
61/635,056 2012.04.18 US
61/635,226 2012.04.18 US
61/647,516 2012.05.16 US
61/684,693 2012.08.17 US

(85)PCT国际申请进入国家阶段日
2014.05.13

(86)PCT国际申请的申请数据
PCT/US2013/037231 2013.04.18

(87)PCT国际申请的公布数据
WO2013/158917 EN 2013.10.24

(73)专利权人 NICIRA股份有限公司
地址 美国加利福尼亚

(72)发明人 T·考珀内恩 A·帕德马纳班

(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 11038
代理人 李渤

(51)Int.Cl.
H04L 12/751(2006.01)
H04L 12/715(2006.01)
H04L 12/931(2006.01)

(56)对比文件
CN 101136866 A,2008.03.05,
WO 2011083780 A1,2011.07.14,
US 2011305167 A1,2011.12.15,

审查员 段燕辉

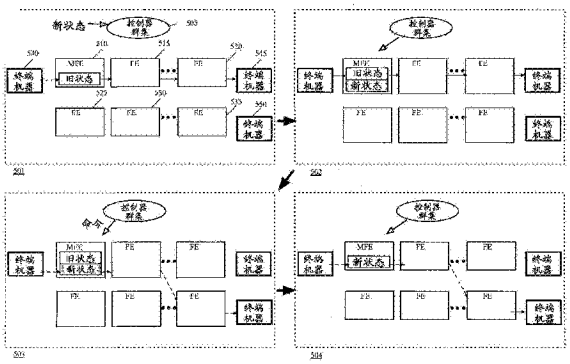
权利要求书2页 说明书23页 附图14页

(54)发明名称

一种用于管理网络的控制器的配置托管元件的方法及设备

(57)摘要

说明了一种用于管理网络的控制器的配置托管转发元件的方法，所述网络包含在网络中转发数据的多个托管转发元件，所述方法包括根据在所述多个托管转发元件中实现的逻辑网络的当前网络策略，产生用于定义托管转发元件的转发行为的第一组流表项。所述方法把第一组流表项发送给托管转发元件，以便使托管转发元件可以根据当前网络策略，转发该托管转发元件直接从终端机器接收的数据。所述方法根据逻辑网络的新的网络策略，产生用于修改托管转发元件的转发行为的第二组流表项。所述方法把第二组流表项发送给托管转发元件，以便使该托管转发元件可以根据新的网络策略，转发数据。



CN 104170334 B

1.一种用于管理网络的控制器的配置托管转发元件的方法，所述网络包含在网络中转发数据的多个托管转发元件，所述方法包括：

生成用于配置作为第一跳转发元件的特定托管转发元件，以便(i)直接从作为分组的源的终端机器接收分组，和(ii)把所述分组转发给作为所述分组的目的地的终端机器的第一组配置数据；

生成用于配置作为一组非第一跳转发元件的一组托管转发元件，以便(i)不直接从源终端机器接收所述分组，和(ii)把所述分组转发给目的地终端机器的第二组配置数据；和

在把所述第一组配置数据发送给作为第一跳转发元件的所述特定托管转发元件之前，把所述第二组配置数据发送给作为非第一跳转发元件的所述一组托管转发元件。

2.按照权利要求1所述的方法，还包括：

在把所述第二组配置数据发送给作为非第一跳转发元件的所述一组托管转发元件之前：

生成用于配置作为第一跳转发元件的所述特定托管转发元件的第三组配置数据，以便把版本信息附加在作为第一跳转发元件的所述特定托管转发元件接收和转发的特定分组上；

生成用于配置作为非第一跳转发元件的所述一组托管转发元件的第四组配置数据；

把所述第三组配置数据发送给作为第一跳转发元件的所述特定托管转发元件；和

把所述第四组配置数据发送给作为非第一跳转发元件的所述一组托管转发元件，

其中在从控制器收到所述第二组配置数据之后，所述一组非第一跳转发元件利用版本信息选择使用所述第四组配置数据，而不是所述第二组配置数据，来转发所述非第一跳转发元件接收和转发的所述特定分组。

3.按照权利要求2所述的方法，其中所述版本信息具有单个二进制位的大小。

4.按照权利要求2所述的方法，还包括：

把第一组配置数据发送给作为第一跳转发元件的所述特定托管转发元件，

其中所述第一组配置数据进一步用于配置作为第一跳转发元件的所述特定托管转发元件，以便把不同的版本信息附加到作为第一跳转发元件的所述特定托管转发元件接收和转发的特定分组上，

其中在收到第二组配置数据之后，作为非第一跳转发元件的所述一组托管转发元件利用所述不同的版本信息选择使用所述第二组配置数据，而不是第四组配置数据，来转发作为非第一跳转发元件的所述一组托管转发元件接收的特定分组。

5.按照权利要求2所述的方法，还包括配置作为第一跳转发元件的所述特定托管转发元件，以便在收到所述第一组配置数据之后，除去所述第三组配置数据。

6.按照权利要求2所述的方法，还包括向作为第一跳转发元件的所述特定托管转发元件发送命令，以便在收到所述第一组配置数据之后，除去所述第三组配置数据。

7.按照权利要求1所述的方法，其中作为第一跳转发元件的所述特定托管转发元件在主机中工作，其中源终端机器在相同的主机内工作。

8.按照权利要求1所述的方法，其中所述特定托管转发元件是第一托管转发元件，其中所述第二组配置数据进一步用于配置所述一组托管转发元件中作为最后一跳转发元件的第二托管转发元件，所述最后一跳转发元件用于把作为非第一跳转发元件的所述第二托管

转发元件接收的分组直接发送给目的地终端机器。

9.按照权利要求8所述的方法，其中所述第二托管转发元件在主机中工作，其中所述目的地终端机器在相同主机内工作。

10.一种用于管理网络的控制器的配置一组托管转发元件的方法，所述网络包含在网络中转发数据的托管转发元件，所述方法包括：

生成(i)用于配置所述一组托管转发元件以作为第一跳转发元件操作、用于直接从一组终端机器接收的数据的第一组流表项，和(ii)用于配置所述一组托管转发元件以作为非第一跳转发元件操作、用于从其他托管转发元件接收的数据的第二组流表项；和

在把所述第一组流表项发送给所述一组托管转发元件之前，把所述第二组流表项发送给所述一组托管转发元件。

11.按照权利要求10所述的方法，还包括：

在发送所述第二组流表项之前：

生成用于配置所述一组托管转发元件，以把版本信息附加到当作为第一跳转发元件操作时的所述一组托管转发元件接收和转发的数据上的第三组流表项；

生成用于配置所述一组托管转发元件以作为非第一跳转发元件操作，用于附加有版本信息的数据的第四组流表项；以及

把第三和第四组流表项发送给所述一组托管转发元件，

其中在从控制器收到所述第二组流表项之后，所述一组托管转发元件利用版本信息选择第四组流表项，而不是第二组流表项，用于所述一组托管转发元件从其他托管转发元件接收的数据。

12.按照权利要求11所述的方法，其中所述版本信息具有单个二进制位的大小。

13.按照权利要求11所述的方法，还包括：

把所述第一组流表项发送给所述一组托管转发元件，

其中所述第一组流表项进一步用于配置所述一组托管转发元件，以把不同的版本信息附加到当作为第一跳转发元件操作时的所述一组托管转发元件接收和转发的数据上，

其中在收到所述第二组流表项之后，所述一组托管转发元件利用所述不同的版本信息选择第二组流表项，而不是第四组流表项，用于所述一组托管转发元件从其他托管转发元件接收的数据。

14.按照权利要求11所述的方法，还包括配置所述一组托管转发元件，以便在收到所述第一组流表项之后，除去所述第三和第四组流表项。

15.按照权利要求11所述的方法，还包括向所述一组托管转发元件发送命令，以便在收到所述第一组流表项之后，除去所述第三和第四组流表项。

16.一种包括实现按照权利要求1-15任意之一所述方法的装置的设备。

一种用于管理网络的控制器的配置托管元件的方法及设备

背景技术

[0001] 在网络内，是网络转发状态把分组从分组的网络入口点运送到它们的出口点。逐跳地，转发状态使网络元件把分组转发给更靠近分组的目的地一步的元件。显然，计算与配置的网络策略一致的转发状态对网络的运行至关紧要。这是因为如果没有适当的转发状态，那么网络将不会把分组递送给其目的地，也不会按照配置的策略进行转发。

发明内容

[0002] 本发明的一些实施例提供一种更新转发状态，以指定新的网络策略的控制器群集。所述控制器群集按照转发元件始终如一地把新的网络策略应用于分组，而不应用新旧策略的混合的方式，把更新的转发状态发送给一组转发元件。

[0003] 在一些实施例中，控制器群集配置在分组的路径的起点的第一跳托管转发元件，以作出转发分组的所有逻辑转发决策(例如，找出逻辑出口端口，并识别逻辑出口端口的物理出口端口)。在分组的路径中的其它托管和非托管转发元件不作出关于所述分组的逻辑转发决策，从而不需要接收转发状态。这些其它转发元件仅仅用作根据分组的源信息和目的地信息，把分组发送给目的地的架构。分组不需要携带任何版本信息，以指示该分组应利用更新的转发状态来转发。这是因为关于该分组的所有转发决策都由第一跳的托管转发元件进行，非第一跳的转发元件不接收更新的转发状态。当分组由根据新策略作出所有逻辑转发决策的第一跳的托管转发元件转发时，所述分组仅仅依据新策略被转发。

[0004] 在一些实施例中，控制器群集按照逻辑转发决策被分散在第一跳转发元件以及非第一跳转发元件内，并且由第一跳转发元件以及非第一跳转发元件进行的方式，配置托管转发元件。在这些实施例中，控制器群集首先把更新的转发状态发送给在分组的路径中的非第一跳转发元件。只有在控制器群集把更新的转发状态发送给非第一跳转发元件之后，控制器群集才把更新的转发状态发送给分组的第一跳转发元件。控制器群集随后指令第一跳转发元件利用更新的转发状态转发该分组。在一些实施例中，由第一跳转发元件转发的分组携带指示应利用更新的转发状态，转发该分组的版本信息。按照这种方式，确保将按照新的网络策略，转发由第一跳转发元件转发给非第一跳转发元件的分组。

[0005] 本发明的一些实施例还提供一种托管转发元件，所述托管转发元件被配置成跨该托管转发元件与网络控制器群集中的几个网络控制器建立的一组通道，实现事务。特别地，在通过特定通道收到分界线之前，一些实施例的托管转发元件不提交通过各个通道接收的转发状态。托管转发元件通过其它通道收到的分界线不会使托管转发元件提交收到的转发状态。即，只有在从特定通道收到分界线之后，托管转发元件才提交转发状态。这样，托管转发批处理通过其它通道到来的事务输入和通过所述特定通道到来的事务输入。

[0006] 上面的发明内容用来简要介绍本发明的一些实施例。并不意图介绍或概述在本文中公开的所有发明主题。下面的具体实施方式和在具体实施方式中参照的附图将进一步说明在发明内容中说明的各个实施例，以及其它实施例。因而，为了理解本文描述的所有实施例，需要完整地回顾发明内容、具体实施方式和附图。此外，要求保护的主题不受发明内容、

具体实施方式和附图中的例证细节限定制，而是由附加的权利要求限定，因为要求保护的主题可用其它具体形式体现，而不脱离所述主题的范围。

附图说明

[0007] 附加权利要求中记载了本发明的新颖特征。然而，为了便于说明，在以下附图中例示了本发明的几个实施例。

[0008] 图1描述网络控制器的例证分层结构。

[0009] 图2图解说明一些实施例的网络控制器的体系结构。

[0010] 图3图解说明跨一组托管转发元件实现的多个逻辑交换元件的例子。

[0011] 图4图解说明在物理基础结构中实现的几个逻辑转发元件。

[0012] 图5图解说明把更新的转发状态信息发送给一组托管转发元件。

[0013] 图6概念地图解说明一些实施例进行的向一组托管转发元件发送更新的转发状态信息的过程。

[0014] 图7概念地图解说明一些实施例进行的在托管转发元件，接收更新的转发状态的过程。

[0015] 图8图解说明向一组托管转发元件发送更新的转发状态信息。

[0016] 图9概念地图解说明一些实施例进行的向一组托管转发元件发送更新的转发状态信息的过程。

[0017] 图10概念地图解说明一些实施例进行的在托管转发元件，接收更新的转发状态的过程。

[0018] 图11概念地图解说明一些实施例进行的事务性地计算并把转发状态发送给一组托管转发元件的过程。

[0019] 图12图解说明控制器群集的几个控制器建立了到其的几个通信通道，以便向其发送更新的托管转发元件。

[0020] 图13概念地图解说明一些实施例进行的把通过一组辅通道接收的事务批量集中(batching)到通过主通道接收的事务中的过程。

[0021] 图14概述地图解说明实现本发明的一些实施例的电子系统。

具体实施方式

[0022] 在本发明的以下详细说明中，记载和说明了本发明的众多细节、例子和实施例。然而，对本领域的技术人员来说，本发明显然并不局限于记载的实施例，在没有所讨论的一些具体细节和例子的情况下，也可以实践本发明。

[0023] 一些实施例提供其中网络控制器计算转发状态信息，以推送给一组托管转发元件，以便定义所述一组托管转发元件的转发行为的网络控制系统。控制器还更新转发状态信息，以修改托管转发元件的转发行为。当网络控制器更新转发状态信息时，控制器把更新的转发状态信息向下推送给托管转发元件，以致托管转发元件按照更新的转发状态信息，在网络中转发数据(例如，呈数据分组的形式)。

[0024] 在一些实施例中，控制器按照在分组的路径中的所有托管转发元件都应用更新的转发状态信息的方式，把更新的转发状态信息推送给托管转发元件。例如，在一些实施例

中，控制器把在分组的路径中的第一跳转发元件配置成进行所有的逻辑转发决策，以致所述路径中的其它转发元件仅仅起把分组转发到目的地的架构的作用。在这些实施例中，控制器只把更新的转发状态信息发送给第一跳转发元件。从而消除了对分组中的指示非第一跳转发元件应利用更新的转发信息向目的地转发所述分组的任何版本信息的需要。

[0025] 在一些实施例中，控制器配置转发元件，以致逻辑转发决策由分组的路径中的第一跳转发元件以及非第一跳转发元件作出。在这些实施例中的一些实施例中，控制器首先把更新的转发状态信息发送给在分组的路径(即，在分组的入口点和出口点之间的路径)中的所有托管转发元件，作为该分组的第一跳转发元件的一个托管转发元件除外。分组的第一跳转发元件直接从源头机器接收所述分组。即，分组的第一跳转发元件在所述路径的起点。

[0026] 控制器随后向第一跳托管转发元件发送更新的转发状态信息。在一些实施例中，当第一跳托管转发元件把分组转发给下一跳转发元件时，第一跳托管转发元件把版本信息包含在分组中。所述版本信息指示应根据更新的转发状态信息，而不是旧的转发状态信息，转发所述分组。这样，由利用更新的转发状态信息的第一跳托管转发元件接收和转发的分组被在所述分组的路径中的、已准备好利用更新的转发状态信息的非第一跳托管转发元件进一步转发。

[0027] 在以下各节中，说明了更详细的实施例。具体地，节I首先说明用于控制逻辑网络和物理网络的一些实施例的网络控制系统。之后，节II 说明按照本发明的一些实施例的生成、更新和推送转发状态信息的网络控制器。节III接下来说明利用几个通信通道，从控制器接收转发状态信息的托管转发元件。最后，节IV说明实现本发明的一些实施例的电子系统。

[0028] I.网络控制系统

[0029] 图1图解说明其中网络控制器计算转发状态信息，以推送给一组托管转发元件，以便定义所述一组托管转发元件的转发行为的网络控制系统100。网络控制系统100包括控制器群集105和3个托管转发元件 125-135。网络控制器群集105包括3个网络控制器-逻辑控制器110和2 个物理控制器115及120。网络控制系统100表示具有向下向3个托管转发元件推送状态的一个控制器群集105的简化例子。在许多情况下，一些实施例的网络控制系统会包括众多的控制器群集，和成百上千个托管转发元件，每个控制器群集包括众多的控制器。

[0030] 在一些实施例中，网络控制器群集105进行转发状态的计算，并以流表项的形式，把所述状态向下推送给托管转发元件。一些实施例的网络控制器群集接收定义逻辑网络的逻辑控制平面(LCP)数据，并把该LCP 数据转换成物理控制平面(PCP)数据，以发送给托管转发元件125-135。在一些实施例中，逻辑网络的逻辑控制平面定义连接逻辑拓扑中的终端机器(例如，虚拟机)的一个或多个逻辑转发元件(例如，逻辑交换机、逻辑路由器)。逻辑转发元件定义来自源机器的分组在逻辑空间中，应被如何转发给目的地机器(例如，相对于逻辑端口的虚拟机MAC地址的绑定)。另外，在一些实施例中，LCP定义由逻辑转发元件实现的逻辑策略(例如，访问控制列表)。LCP及其结构对通过其实现的物理网络来说是不可知的。

[0031] 一些实施例的网络控制器群集进行LCP数据的几种不同变换，以获得被向下推送到托管转发元件的PCP数据。在一些实施例中，控制器群集把LCP数据转换成逻辑转发平面(LFP)数据，随后把LFP数据转换成PCO 数据。LFP数据定义在逻辑空间中，转发分组的转发

表项。即，除简单地把地址绑定到逻辑端口以外，LFP数据还包括规定如果地址匹配，那么把分组转发给逻辑端口的表项。

[0032] LFP数据到PCP数据的变换把逻辑转发表项结合到物理网络中。PCP 表项包含在物理网络内的逻辑地址空间中进行转发的信息(例如，把逻辑端口映射到物理端口等)。

[0033] 在一些实施例中，推送到托管转发元件的PCP的计算分布在控制器群集的不同层的控制器之间。例如，在一些实施例中，逻辑控制器110 管理至少一个逻辑转发元件。逻辑控制器110进行LCP-LFP变换和随后的LFP-通用PCP(UPCP)变换，如该图的右半部分所示。UPCP数据包括未被定制成包括特定于任何托管转发元件的数据，而是只包括特定于特定物理实现的这种数据的抽象(例如，端口号、隧道标识符等)的流表项。

[0034] 在一些实施例中，管理特定逻辑转发元件的逻辑控制器把UPCP数据发送给任意数目的物理控制器。例如，逻辑控制器110把UPCP数据发送给所述2个物理控制器115和120。每个托管转发元件由主物理控制器管理。从而，用于跨几个托管转发元件实现的逻辑转发元件的UPCP数据可被发送给管理这些转发元件的几个不同的主物理控制器。如图所示，物理控制器115是管理2个托管转发元件125和130的主控制器。物理控制器120是管理托管转发元件135的主控制器。

[0035] 在物理控制器或者与托管转发元件在相同物理机中的机架控制器 (本图中未示出)，UPCP数据被转换成定制的PCP(CPCP)数据。CPCP数据是填充有为特定的托管转发元件特有的定制数据的物理控制平面数据。如上所述，在一些实施例中，物理控制器利用从托管转发元件接收的信息，进行这种变换。在其它实施例中，物理控制器起通道(pass-through)作用，以把UPCP数据发送给托管转发元件所驻留于的主机，在所述主机，控制器逻辑(机架控制器)进行UPCP-CPCP变换。

[0036] 托管转发元件125-135是由网络控制器管理(例如，接收来自网络控制器的转发状态信息)的软件或硬件转发元件。在一些实施例中，托管转发元件是在主机上(例如，在主机的用户空间和/或内核内)工作的软件转发元件。这些托管转发元件接收来自终端机器140-160的分组，对分组进行逻辑处理，和跨过物理网络把分组发送给其目的地(例如，在也连接到不同的托管转发元件的另一个终端机器)。

[0037] 终端机器140-160可以是物理机或虚拟机。在一些实施例中，作为虚拟机的终端机器在具有为所述终端机器转发分组的托管转发元件的相同主机中工作。由于属于多个物理网络的虚拟机可能位于单个主机中(例如，终端机器140和145可以位于托管转发元件125所位于的相同主机内)，因此每个托管转发元件可以实现多个不同的逻辑转发元件。另外，如上所述，通常跨众多的托管转发元件，实现单个逻辑转发元件。

[0038] 除了位于网络边缘的托管转发元件以外，在具有虚拟机的主机上，一些实施例另外包括第二级的非边缘托管转发元件(在一些情况下，称为池节点或服务节点)。当边缘托管转发元件不能进行关于分组的所有处理时(例如，由于它不具有把目的地MAC地址绑定到逻辑端口的流表项)，边缘托管转发元件把分组发送给池节点，以便使池节点可以处理该分组，并把分组发送给其目的地。

[0039] 图2概念地图解说明一些实施例的网络控制器200的例证体系结构。取决于网络控制器200处理的数据的种类，网络控制器200能够起逻辑控制器、物理控制器或机架控制器的作用。

[0040] 作为逻辑控制器，网络控制器200把LCP数据作为输入。在一些实施例中，网络控制器200把LCP数据转换成LFP数据，随后转换成UPCP 数据。网络控制器200把UPCP数据推送给一组物理控制器，所述一组物理控制器是实现作为逻辑控制器的网络控制器200所管理的逻辑转发元件的托管转发元件的控制者(mas ter)。

[0041] 作为一些实施例的物理控制器，网络控制器200把UPCP数据作为输入，把UPCP数据转换成CPCP数据。网络控制器随后把CPCP数据推送给网络控制器200为其控制者的一组托管转发元件。在其它实施例中，作为物理控制器的网络控制器200把UPCP中继给一组机架控制器，所述一组机架控制器在一组托管转发元件在其中工作的主机中工作。在这些实施例中，网络控制器200是该组托管转发元件的控制者。

[0042] 作为机架控制器，网络控制器200把来自一组物理控制器的UPCP数据作为输入。网络控制器200为机架控制器管理的托管转发元件，把UPCP 数据转换成CPCP数据，随后把CPCP数据发送给所述托管转发元件。

[0043] 如图2中所示，网络控制器200包括一组规则引擎输入表210，一组函数和常数表215，导入器220，规则引擎225，一组规则引擎输出表245，转换器250，导出器255，持久性事务数据库(PTD)260，和编译器235。编译器235是控制器的与控制器的其它组件在不同的时刻工作的组件。当开发人员需要为特定的网络控制器和/或虚拟化环境指定规则引擎时，编译器工作，而当控制器面接(interface)其它控制器或托管转发元件时，控制器的其余模块在运行时工作。

[0044] 在一些实施例中，编译器235获得用声明性语言指定的较小一组(例如，几百行)声明性指令240，并把这些指令转换成规定规则引擎225的操作的较大一组(例如，数千行)代码(即，目标代码)，规则引擎225进行控制器的表映射。因而，编译器大大简化网络控制器开发人员的定义和更新网络控制器的过程。这是因为编译器允许开发人员利用允许简明地定义网络控制器的复杂映射操作的高级编程语言，随后响应于任意数目的变化(即，由网络控制器支持的逻辑连网功能的变化，对网络控制器的期望行为的变化等等)，更新该映射操作。此外，当开发人员定义映射操作时，编译器可以使开发人员免于考虑事件到达网络控制器的顺序。另外，开发人员用不同的规则集编程网络控制器200，以使网络控制器 200起逻辑控制器、物理控制器或机架控制器的作用。

[0045] 在一些实施例中，规则引擎(RE)输入表210包括根据网络控制器200 用作的网络控制器的类型，而具有不同种类的数据的表格。输入表210 包括需要映射到LFP数据的LCP数据，并且当网络控制器200起逻辑控制器作用时，包括需要映射到UPCP数据的LFP数据。当网络控制器200 起物理控制器或机架控制器作用时，输入表210包括需要被映射到CPCP 数据的UPCP数据。

[0046] 除了RE输入表210以外，网络控制器200还包括规则引擎225用于为其表映射操作收集输入的其它各种表215。这些表215包括保存规则引擎225为进行其表映射操作而需要的常数的规定值的常数表。例如，常数表215可包括被定义为值0的常数“0”，被定义为值4000的常数“dispatch_port_no”，和被定义为值0xFF:FF:FF:FF:FF:FF的常数“broadcast_MAC_addr”。

[0047] 当规则引擎225引用常数时，实际上取回并使用关于所述常数定义的对应值。另外，可以修改和/或更新常数表215中关于各个常数定义的值。这样，常数表215提供修改关

于规则引擎225引用的常数定义的值，而不需要重写或重新编译指定规则引擎225的操作的代码的能力。表215 还包括函数表，函数表保存规则引擎225为了计算用于填充输出表245的值而需要使用的函数。

[0048] 规则引擎225进行表映射操作，表映射操作指定把输入数据转换成输出数据的一种方式。每当规则引擎(RE)输入表之一被修改时，规则引擎进行一组表映射操作，所述一组表映射操作可导致一个或多个RE输出表中的一个或多数据元组的更改。在一些实施例中，网络控制系统利用称为nLog的datalog数据库语言的变形来创建规则引擎225。类似于datalog，nLog提供允许开发人员指定当发生不同事件时，要进行的不同操作的少许声明规则和算符。在一些实施例中，nLog提供由datalog提供的算符的有限子集，以便提高nLog的运算速度。例如，在一些实施例中，在任意声明规则中，nLog只允许使用AND算符。

[0049] 如图2中所示，规则引擎225包括事件处理器222、几个查询计划 227，和表处理器230。每个查询计划是规定当发生对RE输入表之一的更改时，要进行的一组连接操作的一组规则。下面把这样的更改称为输入表事件。每个查询计划由编译器235根据一组声明240中的一个声明规则产生。在一些实施例中，根据一个声明规则，产生不止一个查询计划。例如，为由一个声明规则连接的各个表创建查询计划。即，当声明规则指定连接4个表时，根据所述一个声明，将创建4个不同的查询计划。在一些实施例中，通过利用nLog声明式语言，定义查询计划。

[0050] 规则引擎225的事件处理器222检测每个输入表事件的发生。不同实施例的事件处理器不同地检测输入表事件的发生。在一些实施例中，事件处理器向RE输入表登记回叫，以便通知对RE输入表的记录的改变。在这样的实施例中，当从RE输入表收到其记录之一已改变的通知时，事件处理器222检测输入表事件。

[0051] 响应于检测的输入表事件，事件处理器222(1)为检测的表事件，选择适当的查询计划，和(2)指令表处理器230执行该查询计划。在一些实施例中，为了执行查询计划，表处理器230进行由查询计划指定的连接操作，以根据一个或多个输入以及各种表210和215，产生表示一组或多组数据值的一个或多个记录。一些实施例的表处理器230随后(1)进行选择操作，以从利用连接操作产生的记录中选择数据值的子集，和(2)把选择的数据值的子集写入一个或多个RE输出表245中。

[0052] 在一些实施例中，RE输出表245保存逻辑和物理网络元件数据属性。表245被称为RE输出表，因为它们保存规则引擎225的表映射操作的输出。在一些实施例中，RE输出表可被分为几种不同的类别。例如，在一些实施例中，这些表可以是RE输入表和/或控制器输出表。当表中的变化致使规则引擎检测到要求查询计划的执行的输入事件时，表是RE输入表。RE输出表245也可是产生使规则引擎进行另一个查询计划的事件的 RE输入表210。这种事件被称为内部输入事件，与外部输入事件形成对照，外部输入事件是由导入器220产生的RE输入表更改引起的事件。

[0053] 当表中的变化致使导出器255把变化导出到另外的控制器或者托管转发元件时，表是控制器输出表。RE输出表245中的表可以是RE输入表、控制器输出表，或者RE输入表和控制器输出表这两者。在一些实施例中， RE输入表和RE输出表是关系数据库管理系统(RDBMS)的表。这些表被保存为关系数据库数据结构，所述关系数据库数据结构是网络控制器的主要数据存储结构。

[0054] 导出器255检测RE输出表245的控制器输出表的变化。不同实施例的导出器不同地检测控制器输出表事件的发生。在一些实施例中，导出器向控制器输出表登记回叫，以便通知控制器输出表的记录的变化。在这样的实施例中，当它从控制器输出表收到其记录之一已变化的通知时，导出器255检测输出表事件。

[0055] 响应于检测的输出表事件，导出器255获得更改的控制器输出表中的一些或者全部的更改数据元组，并把更改的数据元组传送给其它控制器或托管转发元件。具体地，当网络控制器200起逻辑控制器作用时，导出器255通过与物理控制器建立的一组通信通道(例如，远程过程调用 (RPC)通道)，把UPCP数据传送给一组物理控制器。当网络控制器200起物理控制器作用时，一些实施例的导出器255通过与机架控制器建立的一组通信通道，把UPCP数据传送给一组机架控制器。其它实施例的导出器255通过与各个托管转发元件建立的一对通信通道(例如，OpenFlow 通道和配置通道)，把CPCP数据传送给一组托管转发元件。当网络控制器200起机架控制器作用时，一些实施例的导出器255通过与各个托管转发元件建立的一对通信通道(例如，OpenFlow通道和配置通道)，把CPCP 数据传送给一组托管转发元件。

[0056] 在一些实施例中，网络控制器不在输出表245中保持该网络控制器不负责管理的数据。然而，这样的数据将由转换器250转换成可被保存在PTD中的格式，并被保存在PTD260中。PTD是网络控制器的次要存储结构。网络控制器200的PTD把该数据传送给一个或多个其它的网络控制器，以致负责管理该数据的其它网络控制器中的一些网络控制器能够处理该数据。

[0057] 在一些实施例中，考虑到数据的弹性，网络控制器还把保存在输出表245中的数据(即，网络控制器负责管理的数据)带到PTD。这样的数据也由转换器250转换，保存在PTD中，并被传送给其它控制器实例的其它PTD。于是，在这些实施例中，控制器实例的PTD具有由网络控制系统管理的所有数据的所有配置数据。即，在一些实施例中，每个PTD包含逻辑和物理网络的结构的全局视图。

[0058] 导入器220面接输入数据的许多不同来源，并利用输入数据更改或创建输入表210。当网络控制器200起逻辑控制器作用时，一些实施例的导入器220通过输入转换控制器(未示出)，接收来自用户(租户)的输入数据，所述输入转换控制器把用户输入转换成LCP数据。在一些实施例中，导入器220通过通信通道，接收LCP数据。导入器220还与PTD260 面接，以致通过PTD从其它控制器实例接收的数据可被用作更改或创建输入表210的输入数据。此外，导入器220还检测RE输入表和RE输出表245的控制器输出表中的变化。产生并保存在输出表245中的LFP数据由用于规则引擎225的导入器220，反馈给规则引擎225，以便产生UPCP数据。

[0059] 当网络控制器200起物理控制器作用时，导入器220通过与一组逻辑控制器建立的一组通信通道，从所述一组逻辑控制器获得UPCP数据。当网络控制器200起机架控制器作用时，导入器通过与一组物理控制器建立的一组通信通道，从所述一组物理控制器获得UPCP数据。

[0060] 图3概念地图解说明跨托管转发元件310-330实现的逻辑转发元件 380和390。图3的上半部分表示3个托管转发元件310-330和终端机器 340-365。如图中所示，机器340、350和360属于用户A，机器345、355 和365属于用户B。为了例示和说明的简明起见，图3表示逻

辑转发元件连接到几个终端机器，并且是在几个托管转发元件中实现的。如上所述，逻辑转发元件可连接到众多的终端机器，并且实现在众多的托管转发元件中。

[0061] 一些实施例的托管转发元件310-330在网络中的耦接到托管转发元件310-330的网络元件之间转发网络数据(例如，分组、帧等)。如图所示，托管转发元件310在机器340和345与转发元件320之间转发网络数据。类似地，转发元件320在机器350与托管转发元件310和330之间转发网络数据，转发元件330在机器355-365与转发元件320之间转发网络数据。

[0062] 此外，每个托管转发元件310-330根据转发元件的转发逻辑(在一些实施例中，呈表格的形式)，转发网络数据。在一些实施例中，转发表按照转发标准，确定把网络数据转发到哪里(例如，转发元件上的端口)。例如，层2转发元件的转发表可以根据MAC地址(例如，源MAC地址和/ 或目的地MAC地址)，确定把网络数据转发到哪里。再例如，层3转发元件的转发表可以根据IP地址(例如，源IP地址和/或目的地IP地址)，确定把网络数据转发到哪里。许多其它种类的转发标准也是可能的。

[0063] 如图所示，每个托管转发元件310-330中的转发表包括几条记录。在一些实施例中，每条记录根据转发标准，指定转发网络数据的操作。在一些实施例中，所述记录可被称为流表项，因为所述记录控制数据通过托管转发元件310-330的“流动”。

[0064] 图3的下半部分图解说明每个用户的逻辑网络的概念表示。如图所示，用户A的逻辑网络380包括用户A的机器340、350和360耦接到的逻辑转发元件385。用户B的逻辑网络390包括用户B的机器345、355 和365耦接到的逻辑转发元件395。因而，从用户A的角度来看，用户A 具有只有用户A的机器耦接到的转发元件，从用户B的角度来看，用户B 具有只有用户B的机器耦接到的转发元件。换句话说，对于每个用户，该用户具有自己的只包括该用户的机器的网络。

[0065] 下面说明实现起源于机器340，并且前往机器350的网络数据，和起源于机器340，并且前往机器360的网络数据的流动的概念性流表项。首先，说明用于转发起源于机器340，并且前往机器350的网络数据的流表项，之后说明用于转发起源于机器340，并且前往机器360的网络数据的流表项。

[0066] 托管转发元件310的转发表中的流表项“A1-A2”指令托管转发元件 310把起源于机器310，并且前往机器350的网络数据转发给转发元件 320。托管转发元件320的转发表中的流表项“A1-A2”指令托管转发元件320把起源于机器310，并且前往机器350的网络数据转发给机器350。

[0067] 此外，托管转发元件310的转发表中的流表项“A1-A3”指令托管转发元件310把起源于机器340，并且前往机器360的网络数据转发给转发元件320。托管转发元件320的转发表中的流表项“A1-A3”指令托管转发元件320把起源于机器340，并且前往机器360的网络数据转发给转发元件330。托管转发元件330的转发表中的流表项“A1-A3”指令托管转发元件330把起源于机器340，并且前往机器360的网络数据转发给机器 360。

[0068] 尽管上面说明了用于转发起源于机器340，并且前往机器350的网络数据，和起源于机器340，并且前往机器360的网络数据的概念性流表项，不过在托管转发元件310-330的转发表中，会包含类似的流表项，以便在用户A的逻辑网络380中的其它机器之间转发网络数据。此外，在托管转发元件310-330的转发表中，会包含类似的流表项，以便在用户B 的逻辑网络390中的机器之间转发网络数据。

[0069] 图3中表示的概念性流表项包括托管转发元件的源信息和目的地信息，以推测把分组发送给的下一跳转发元件。不过，源信息不必在流表项中，因为一些实施例的托管转发元件仅仅利用目的地信息(例如，目的地地址等)，就能够推测出下一跳转发元件。

[0070] 在一些实施例中，由隧道协议(例如，无线接入点控制与配置协议 (CAPWAP)、通用转发封装(GRE)、GRE因特网协议安全(IPsec)等)提供的隧道可用于使跨托管转发元件310-330的逻辑转发元件385和395的实现更容易。借助隧穿，分组作为另一个分组的有效负载，被传送通过转发元件。即，隧穿的分组不必暴露其地址(例如，源MAC地址和目的地MAC 地址)，因为该分组是根据包含在封装该隧穿分组的外部分组中的报头中的地址转发的。于是，隧穿允许逻辑地址空间与物理地址空间的分离，因为隧穿的分组可具有在逻辑地址空间中有意义的地址，而外部分组是根据物理地址空间中的地址转发的。这样，隧道可被看作连接网络中的托管转发元件，以便实现逻辑转发元件385和395的“逻辑导线”。

[0071] 用上述各种方式配置转发元件，以跨一组转发元件地实现多个逻辑转发元件，允许多个用户从每个用户的角度来看，都具有独立的网络和/ 或转发元件，尽管用户事实上共享一些或者全部的相同一组转发元件和/ 或所述一组转发元件之间的连接(例如，隧道，物理导线)。

[0072] 虽然图3图解说明了在一组托管转发元件中的逻辑转发元件的实现，不过，通过配置托管转发元件的转发表，可以实现更复杂的逻辑网络(例如，包括几个L3转发元件)。图4概念地图解说明更复杂的逻辑网络的例子。图4图解说明一些实施例的网络体系结构400，它实现3个逻辑转发元件-逻辑路由器425及逻辑交换机420和430。具体地，网络体系结构400表示实现其数据分组由逻辑路由器425及逻辑交换机420和430 转发的逻辑网络的物理网络。图4在其上半部分中，图解说明逻辑路由器425及逻辑交换机420和430。图4在其下半部分中，图解说明托管转发元件455和460。图4在其上部和下部中，都图解说明了终端机器(例如，虚拟机(VM))1-4。

[0073] 在本例中，逻辑交换元件420在逻辑路由器425、终端机器1和终端机器2之间转发数据分组。逻辑交换元件430在逻辑路由器425、终端机器3和终端机器4之间转发数据分组。如上所述，逻辑路由器425在逻辑交换机420和430与其它逻辑路由器和交换机(未示出)之间，路由数据分组。逻辑交换机420和430及逻辑路由器425通过逻辑端口(未示出) 被逻辑耦接，并通过逻辑端口交换数据分组。这些逻辑端口被映射到或附接到托管转发元件455和460的物理端口。

[0074] 在一些实施例中，在托管网络中的每个托管交换元件中，实现逻辑路由器。当托管交换元件从耦接到该托管交换元件的机器收到分组时，该托管交换元件进行逻辑路由。换句话说，在这些实施例中，作为分组的第一跳交换元件的托管交换元件进行逻辑路由。

[0075] 在这个例子中，托管转发元件455和460分别是在主机465和470 中运行的软件交换机。托管转发元件455和460具有流表项，所述流表项实现转发和路由托管转发元件455和460从终端机器1-4接收的分组的逻辑交换机420和430。流表项还实现逻辑路由器425。利用这些流表项，托管转发元件455和460能够在网络中的，耦接到托管转发元件455 和460的网络元件之间，转发和路由分组。

[0076] 如图所示，托管转发元件455和460都具有3个端口(例如，虚拟接口(VIF))，通过这3个端口，与耦接到托管转发元件455和460的网络元件交换数据分组。在一些情况下，这些

实施例中的数据分组将通过在托管转发元件455和460之间建立的隧道(例如,止于托管交换元件455 的端口3和托管交换元件460的端口6的隧道)传播。该隧道使得能够分离逻辑空间中的地址和物理空间中的地址。

[0077] 在本例中,如图所示,每个主机465和470包括托管交换元件和几个终端机器。终端机器1-4都被赋予一组网络地址(例如,L2的MAC地址,网络L3的IP地址等),并且能够往来于其它网络元件,发送和接收网络数据。终端机器由在主机465和470上运行的管理程序(未示出)管理。终端机器1和2分别与相同的逻辑交换机420的逻辑端口1和2相关联。不过,机器1与托管交换元件455的端口4相关联,机器2与托管交换元件460的端口7相关联。逻辑端口1和2于是被分别映射到端口4和7,不过,这种映射不必暴露给网络中的任意网络元件(未示出)。这是因为将根据携带具有映射信息的分组作为有效负载的外部分组的外部报头,通过隧道在机器1和2之间交换该映射信息。

[0078] 上面说明了物理基础结构中的网络控制系统,及逻辑网络的实现。下面的节II说明更新的转发状态到托管转发元件的事务传播。

[0079] II.利用事务性

[0080] 在网络配置被改变之后,更新转发状态(即,从先前计算的状态迁移到新计算的状态)存在几个挑战。下面说明几种解决方案。这些解决方案在两个维度-正确性和效率,考虑该问题。即,这些解决方案考虑网络中目前存在的状态如何确保不仅在更新之前和之后,而且在更新期间,网络策略都能够被正确遵守。就效率来说,这些解决方案考虑如何能够使可能较大的状态更新的成本降至最小。

[0081] 在下面的讨论中,网络控制系统包括计算转发元件的转发状态,以便管理网络转发元件的控制器的集中群集。另外,在下面的讨论中,“网络策略”包括任何配置方面:不仅包括安全策略,而且包括关于如何路由网络流量,以及任何物理(或逻辑)网络配置的策略。从而,在本讨论中,“策略”用于与用户配置的输入有关的一切。

[0082] A.关于事务的要求

[0083] 分组是转发状态的作用对象。必须按照单一的一致策略,而不是代表新旧策略的状态的混合来转发单个分组。随后的分组可以利用该策略的不同版本来处理,只要按照避免用新旧策略的混合来处理的方式,从旧版本过渡到新版本即可。

[0084] 自动过渡到新策略的要求意味对转发状态的更新必须是事务的。然而,如上所述,并不意味整个网络转发状态应被同时自动更新。特别地,一些实施例的网络控制系统在两个方面放松了这种要求。首先,对于从源到一个或多个目的地的分组流,指定策略在什么时刻,从旧策略变成新策略并不重要。唯一必要的是没有分组按照新旧策略的混合被转发。每个分组应是按照旧策略或者按照新策略转发的。其次,一些实施例的网络控制系统允许瞬时对在不同的位置,进入网络中的不同分组流应用不同的策略。同样,这些实施例只要求单个分组仅仅经历单一策略,而不是新旧策略的混合。

[0085] B.实现事务更新

[0086] 已知这些要求和放松,下面将考虑这些事务更新的实现。在M. Reitblatt等的“Cons istent Updates for Software-Defined Networks: Change You Can Believe in!”(Proceedings of the10th ACM Workshop on Hot Topics in Networks,p.1-6, November14-15,2011,Cambridge, Massachusetts(“Reitblatt的论文”)中,提出了在网络

入口，用在该入口使用的转发状态的版本标记分组。从而，当分组在网络内前进时，任何后续网络元件知道使用哪个版本。这为任何网络转发状态，有效地实现事务性的全网络更新。

[0087] 然而，这种方法伴随有几个实际挑战。首先，在不假定网络的分片的情况下，对网络的更新必须被串行化。即，整个网络必须为特定版本作准备，随后入口被更新，以利用准备的版本，只有在此之后，才能够开始对下一个版本的准备。

[0088] 其次，分组需要具有明确的版本标签，从而分组报头中某个地方的足够比特需要被分配给所述标签。如果网络具有利用传统隧道协议工作的要求，那么在报头中，找出用于所述标签的这种空闲比特可能有挑战性。

[0089] 从而，在Reitblatt的论文中描述的全网络的事务更新尽管强大，但是伴随理论上应避免的实际挑战。从而，代替在Reitblatt的论文中描述的这种方法，一些实施例的网络控制系统采用在网络的边缘的托管转发元件的布置。如在上面的节I中所述，一些实施例的网络控制系统在第一跳作出逻辑转发决策(即，哪个或哪些逻辑端口应接收分组的决策)。任何后续步骤仅仅是根据所述转发策略，向选择的目的地转发所述分组。

[0090] 图5概念地图解说明在第一跳，作出逻辑转发决策的网络控制系统的网络控制器群集505。具体地，图5按4个不同的阶段501-504，图解说明网络控制器群集505只把转发状态更新发送给第一跳托管转发元件。网络控制器群集505类似于上面参考图1说明的网络控制器群集105，因为网络控制器群集505包括生成、更新并把转发状态传播给托管转发元件(MFE)510的逻辑控制器和物理控制器(未示出)。托管转发元件510是起源于终端机器540的数据的第一跳转发元件。即，托管转发元件510 直接面接终端机器540，并把来自终端机器540的数据朝着该数据的目的地转发。一组转发元件515-535在一组终端机器540-550之间转发数据。

[0091] 在第一阶段501，托管转发元件510根据该托管转发元件具有的当前转发状态(旧状态)，转发来自终端机器540的网络数据(未示出)。具体地，对于从终端机器540发送给终端机器545的分组，由控制器群集定义的路径跨越转发元件(FE)510、515和520，如用实箭头线所示。另外在阶段501，控制器群集505从用户(例如，通过图中未描述的输入转换控制器)，接收对转发状态的更新。所述更新代表新的网络策略(例如，定义不同的可用带宽的新的QoS策略，新开通的从VM到另一个VM的新路径等)。

[0092] 在第二阶段502，控制器群集505计算转发状态更新(例如，通过把输入的LCP数据转换成UPCP或CPCP数据)。在一些实施例中，控制器群集505识别实现逻辑转发元件的所有托管转发元件。特别地，对于将从第一物理端口转发到第二物理端口(第一物理端口和第二物理端口被映射成逻辑转发元件的逻辑入口端口和逻辑出口端口)的分组的路径，控制器群集505识别具有所述第一物理端口的转发元件(即，第一跳转发元件) 和具有第二物理端口的转发元件(即，最后一跳转发元件)。

[0093] 在阶段502，更新的转发状态具有起源于终端机器540，并且前往终端机器550的分组的新路径，终端机器550是在计算旧的转发状态，并将其传播给网络转发元件之后，加入网络中的终端机器。就该新路径来说，托管转发元件510是第一跳托管转发元件，而转发元件520是最后一跳转发元件。转发元件515是向最后一跳托管转发元件535转发所述分组的“中间”托管和非托管转发元件之一。

[0094] 控制器群集505为受用户更新影响的所有路径，计算更新的转发状态，并识别这些

路径中的每条路径的第一跳托管转发元件。在阶段502，控制器群集505把各条路径的第一跳托管转发元件的更新的转发状态，发送给托管转发元件。为了举例说明的简明起见，阶段502表示始于托管转发元件510的那些路径的新旧转发状态。托管转发元件510具有这些路径的新旧转发状态。托管转发元件510尚未利用新的转发状态，而是根据旧的转发状态转发分组。

[0095] 在一些实施例中，当作为第一跳转发元件的托管转发元件收到新的转发状态时，所述托管转发元件开始利用新的转发状态。然而，在一些实施例中，控制器群集505向托管转发元件发送命令，以作为第一跳转发元件，开始利用更新的转发状态转发分组。在第三阶段503，控制器群集505把这样的命令发送给托管转发元件510。托管转发元件510利用新的转发状态，以起开始于该托管转发元件510的路径的第一跳转发元件的作用。对于从终端机器540发送给终端机器550的分组的新路径，托管转发元件510现在能够根据新的转发状态，转发分组。由于非第一跳转发元件不需要新的转发状态，从而不获得新的转发状态，因此，这些实施例中的分组不需要携带指示非第一跳转发元件应利用新的转发状态的任何版本信息。

[0096] 在第四阶段504，托管转发元件510-535除去旧的转发状态。在一些实施例中，控制器群集505把托管转发元件配置成在收到新的转发状态之后，过去一段时间之后，除去旧的转发状态。在其它实施例中，控制器群集505向托管转发元件发送命令，以除去旧的转发状态。

[0097] 图6概念地图解说明一些实施例进行的更新并向托管转发元件发送转发状态的过程600。具体地，过程600是其中在第一跳托管转发元件，作出转发分组的所有逻辑转发决策的那些实施例的过程。在一些实施例中，过程600由控制器群集(未示出)，比如图1和5的控制器群集105 或505进行。

[0098] 通过(在605)接收输入，开始过程600，所述输入是对由群集控制器管理的托管转发元件的转发状态的更新。由于至少3个原因，会出现对转发状态的这些更新。首先，当逻辑策略因为由逻辑管线强制实施的网络策略被用户重新配置(例如，借助访问控制列表的更新)而变化时，转发状态变化。其次，工作负载操作变化导致转发状态的变化。例如，当虚拟机从第一节点迁移到第二节点时，逻辑视图保持不变。然而，归因于所述迁移，转发状态需要更新，因为VM附接到的逻辑端口现在位于不同的物理位置。第三，物理重构事件，比如托管转发元件增加、去除、升级和重新配置，会导致转发状态的变化。

[0099] 之后，过程600(在610)根据接收的输入，计算更新的转发状态。所述计算涉及LCP数据到LFP数据的变换，随后从LFP数据到UPCP或CPCP 数据的变换。更新的LCP数据可影响几个逻辑转发元件。即，逻辑路径(即，受影响的逻辑转发元件的许多对逻辑端口之间的许多逻辑路径)被除去、增加或更改，于是，实现这些受影响的逻辑路径的物理路径也被除去、增加或更改。

[0100] 在一些实施例中，这些受影响的逻辑转发元件的逻辑转发操作仅仅由第一路托管转发元件进行。例如，控制器群集把第一跳托管转发元件配置成进行第一逻辑交换机的逻辑L2转发操作，逻辑路由器的逻辑L3 路由，和获得逻辑路由器路由的分组的第二逻辑交换机的逻辑L2转发操作。从而，这些实施例的过程600计算的转发状态仅仅用于第一跳转发元件。这些路径的中间和最后一跳转发元件用作把分组转发给目的地机器的架构。于是，转发

状态不会使托管转发元件向分组中添加版本信息。

[0101] 之后，过程600(在610)为起第一跳转发元件作用的托管转发元件，计算更新的转发状态。过程600随后(在615)把更新的转发状态发送给托管转发元件。托管转发元件现在具有旧的转发状态和更新的转发状态。

[0102] 过程600视情况(在625)向托管转发元件发送从托管转发元件中除去旧的转发状态的命令。在一些实施例中，代替发送切换到新的转发状态的明确命令，控制器群集把托管转发元件配置成用新的转发状态代替旧的转发状态，或者一旦托管转发元件收到新的转发状态，就除去旧的转发状态。另一方面或者结合地，控制器群集配置托管转发元件，以在收到新的转发状态之后，过去一段时间之后，除去旧的转发状态，而不是向托管转发元件发送命令。该过程随后结束。

[0103] 尽管图6图解说明了一些实施例的由网络控制器群集进行的过程 600，不过，图7图解说明一些实施例的由托管转发元件进行的过程。图 7概念地图解说明一些实施例进行的转发数据的过程700。过程700由利用转发状态，以便起第一跳转发元件作用的托管转发元件进行。

[0104] 通过在(705)利用当前转发状态(旧的转发状态)，转发到来的分组，开始该过程。到来的分组来自于该托管转发元件直接面接的终端机器。转发状态接收自控制器群集，或者通过把转发状态发送给托管转发元件，管理该托管转发元件的机架控制器。

[0105] 之后，过程700(在710)从控制器群集，接收更新的转发状态。该转发状态由控制器群集更新，并且包括从LCP数据转换来的CPCP数据。在一些实施例中，在起非第一转发元件作用的托管转发元件收到更新的转发状态之后，控制器把更新的转发状态发送给第一托管转发元件。该托管转发元件现在具有旧的转发状态和更新的转发状态。

[0106] 过程700随后(在715)接收来自控制器群集的利用更新的转发状态转发输入数据的命令。当收到该命令时，一些实施例的第一跳托管转发元件从旧的转发状态，切换到更新的转发状态。在一些实施例中，该命令可以不是明示的。即，在没有收到切换到新的转发状态的明确命令的情况下，一旦新的转发状态被安装在第一跳托管转发元件中，第一跳托管转发元件就利用新的转发状态。

[0107] 过程700随后(在720)利用更新的转发状态，转发到来的分组。从第一跳托管转发元件获得所述分组的非第一跳托管转发元件将利用更新的转发状态来转发这些分组。在一些实施例中，过程700向分组中添加版本信息，以致非第一跳托管转发元件能够选择用于转发来自第一跳托管转发元件的分组的新的转发状态。

[0108] 在725，过程700视情况从控制器群集接收除去旧的转发状态的命令。在一些实施例中，托管转发元件不接收除去旧状态的明确命令。托管转发元件改为被控制器群集配置成在收到更新的转发状态之后，在过去一段时间之后，除去旧的转发状态。过程700随后(在730)除去旧的转发状态。该过程随后结束。

[0109] 如在上面的节I中所述，一些实施例的网络控制系统在第一跳以及非第一跳，作出逻辑转发决策(即，关于哪个或哪些逻辑端口应接收分组的决策)。在这些实施例中的一些实施例中，跨越网络的事务更新被分成两部分：(1)对第一跳托管转发元件的事务更新，和(2)对从第一跳托管转发元件经网络，到最后一跳托管转发元件的路径的事务更新。一旦能够实现这两个部分，就能够提供全局事务。即，通过在用新策略更新第一跳之前，准备任何

新的所需路径，整个状态更新变成原子的。在这两个步骤之后，新的第一跳状态配置不需要的任何网络路径可被除去。

[0110] 图8概念地图解说明采用这种两步法的网络控制系统的网络控制器群集805。具体地，图8按4个不同的阶段801-804，图解说明网络控制器群集805分两部分地把转发状态更新发送给两群托管转发元件。网络控制器群集805类似于上面参考图1说明的网络控制器群集105，因为网络控制器群集805包括生成、更新并把转发状态传播给一组托管转发元件810-835的逻辑控制器和物理控制器(未示出)。托管转发元件810-835 根据从网络控制器群集805接收的转发状态，在一组终端机器840-850 之间转发网络数据(未示出)。

[0111] 在第一阶段801，托管转发元件810-835根据托管转发元件具有的当前转发状态(旧状态)，转发网络数据(未示出)。具体地，对于从终端机器840发送给终端机器845的分组，由控制器群集定义的路径跨越托管转发元件(MFE)810、815和820，如用实箭头线所示。另外在阶段801，控制器群集805从用户(例如，通过图中未描述的输入转换控制器)，接收对转发状态的更新。所述更新代表新的网络策略(例如，定义不同的可用带宽的新的QoS策略，新开通的从VM到另一个VM的新路径等)。

[0112] 在第二阶段802，控制器群集805计算转发状态更新(例如，通过把输入的LCP数据转换成UPCP或CPCP数据)。在一些实施例中，控制器群集805识别实现逻辑转发元件的所有托管转发元件。特别地，对于将从第一物理端口转发到第二物理端口(第一物理端口和第二物理端口被映射成逻辑转发元件的逻辑入口端口和逻辑出口端口)的分组的路径，控制器群集805识别具有所述第一物理端口的托管转发元件(即，第一跳托管转发元件)和具有第二物理端口的托管转发元件(即，最后一跳托管转发元件)。对于该路径，控制器群集805随后把第一跳托管转发元件归类到一群中，把最后一跳托管转发元件以及在该分组的路径中的其它托管转发元件归类到另一群中。

[0113] 例如，更新的转发状态具有起源于终端机器840，并且前往终端机器 850的分组的新路径，终端机器850是在计算旧的转发状态，并将其传播给网络转发元件之后，加入网络中的终端机器。就该新路径来说，托管转发元件810是第一跳托管转发元件，而转发元件820是最后一跳托管转发元件。托管转发元件815是向最后一跳托管转发元件835转发所述分组的“中间”托管和非托管转发元件之一。

[0114] 控制器群集805为受用户更新影响的所有路径，计算更新的转发状态，并识别这些路径中的每条路径的第一跳托管转发元件和非第一跳托管转发元件。在阶段802，控制器群集805发送各条路径的非第一跳托管转发元件的更新的转发状态。为了举例说明的简明起见，阶段802表示始于托管转发元件810的那些路径的新旧转发状态。从而，托管转发元件810只具有旧的转发状态，而其它的托管转发元件具有这些路径的新旧转发状态。对于用于从终端机器840发送给终端机器850的分组的新路径，第一跳托管转发元件还不能够正确地转发分组，因为托管转发元件820不具有向目的地机器850转发分组的转发状态(即，在第一跳托管转发元件810，缺少逻辑出口端口与托管转发元件850的物理端口之间的映射)。

[0115] 在第三阶段803，控制器群集805发送计算的用于所有路径的第一跳转发元件的更新。托管转发元件810现在具有新的转发状态，以起从托管转发元件810开始的路径的第一跳转发元件的作用。对于于从终端机器840发送给终端机器850的分组的新路径，托管转发元件810现在能够根据新的转发状态，正确地转发分组。

[0116] 在一些实施例中，当托管转发元件收到新的转发状态时，作为第一跳转发元件的托管转发元件开始利用新的转发状态。不过，在一些实施例中，控制器群集805向托管转发元件发送作为第一跳转发元件，开始利用更新的转发状态转发分组的命令。

[0117] 在一些实施例中，起托管转发元件直接从源头机器接收的那些分组的第一跳转发元件作用的托管转发元件，把版本信息添加到分组中。在一些实施例中，托管转发元件利用分组的特定二进制位作为版本指示符，或者向每个分组再增加一位，以保存版本信息。在一些这样的实施例中，每当托管转发元件切换到更新版本的转发状态更新时，所述版本位就变更其值。非第一跳托管转发元件随后根据分组携带的版本信息，利用旧的或者新的转发状态。按照这种方式，根据旧的或者新的转发状态，而不是根据新旧转发状态的混合，转发特定分组。

[0118] 在第四阶段804，托管转发元件件810-835除去旧的转发状态。在一些实施例中，控制器群集805把托管转发元件配置成在收到新的转发状态之后，过去一段时间之后，除去旧的转发状态。在其它实施例中，控制器群集505向托管转发元件发送命令，以除去旧的转发状态。

[0119] 利用关于一条旧路径和一条新路径的更新，表示了图8中的4个阶段801-804。由于可能存在为实现逻辑转发元件而定义的许多其它路径，因此控制器群集805和托管转发元件件810-835对受用户更新影响的所有路径，进行按这4个阶段801-804说明的两步过程。下一个附图(图9) 概念地图解说明一些实施例对更新或创建的所有路径，进行的向托管转发元件发送更新的过程900。在一些实施例中，过程900由控制器群集(未示出)，比如图1和8的控制器群集105或805进行。

[0120] 通过(在905)接收输入，开始过程900，所述输入是对由控制器群集管理的托管转发元件的转发状态的更新。由于上面说明的3个原因，会出现对转发状态的这些更新。

[0121] 之后，过程900(在910)根据接收的输入，计算更新的转发状态。所述计算涉及LCP数据到LFP数据的变换，以及随后LFP数据到UPCP或CPCP 数据的变换。更新的LCP数据会影响几个逻辑转发元件，包括逻辑L2交换机和逻辑L3路由器。即，逻辑路径(即，受影响的逻辑转发元件的许多对逻辑端口之间的许多逻辑路径)被除去、增加或更改，于是，实现这些受影响的逻辑路径的物理路径也被除去、增加或更改。从而，更新的转发状态用于所有受影响的物理路径的第一跳转发元件和非第一跳托管转发元件。

[0122] 过程900随后(在915)识别用于起非第一跳托管转发元件作用的托管转发元件的新的转发状态。该转发状态用于在受所述输入影响的路径中，但是不是路径的第一跳托管转发元件的那些托管转发元件。

[0123] 在一些实施例中，只有第一跳和最后一跳托管转发元件需要转发状态更新。在这些实施例中的一些实施例中，受所述输入影响的逻辑转发元件仅仅由第一跳和最后一跳托管转发元件实现。例如，控制器群集把第一跳托管转发元件配置成进行第一逻辑交换机的逻辑L2转发操作(例如，根据分组的MAC地址，逻辑地转发分组)，和进行逻辑路由器的逻辑L3路由操作(例如，根据分组的IP地址，逻辑地路由分组)。控制器群集把最后一跳托管转发元件配置成进行获得由逻辑路由器路由的分组的第二逻辑交换机的逻辑L2转发操作。在这些实施例中，(在915)识别的新的转发状态用于受影响路径的最后一跳托管转发元件。在这些路径的中间的转发元件用作连接第一跳托管转发元件和最后一跳托管转发元件的架构。

在915，过程900还把识别的用于非第一跳转发元件的转发状态发送给非第一跳托管转发元件。

[0124] 过程900随后(在920)识别用于起第一跳托管转发元件作用的托管转发元件的新的转发状态。该转发状态用于作为受影响路径中的第一跳转发元件的那些托管转发元件。在步骤920，过程900还把识别的用于第一跳转发元件的转发状态发送给第一跳托管转发元件。

[0125] 在一些实施例中，更新转发状态不必全局地排序。只需要使每个第一跳元件的更新串行化。即，如果多个第一跳元件都需要转发状态更新，那么可以并行地独立处理它们的更新。只有计算需要是事务性的。

[0126] 在一些实施例中，网络控制系统在一些情况下，利用在Reitblatt 的论文中说明的方法更新全网络状态，在所述一些情况下，路径的非第一跳转发元件的转发状态变化相当多，以致新旧路径会被混合。例如，当在(网络控制器的)软件版本之间，路径标志的寻址方案变化时，会发生这种情况。对于这种状况，控制器群集从路径标志/地址的开始，专门指定一个(或几个)全网络版本位，以致如果需要，能够改变路径寻址的结构。说到这，应注意只要标志/地址结构不变，就可通过增加新的路径，随后在路径的剩余部分准备就绪之后，使第一跳托管转发元件迁移到新路径，实现全网络更新。

[0127] 在把第一跳和非第一跳转发元件的更新的转发状态发送给托管转发元件之后，过程900(在925)判定过程900是否已从所述过程向其发送更新的转发状态的所有托管转发元件，收到确认。所述确认指示托管转发元件收到了来自控制器群集的更新的转发状态。在一些实施例中，只有在收到非第一跳转发元件的更新的转发状态的各个托管转发元件回送确认之后，过程900才向托管转发元件发送第一跳转发元件的转发状态。这些实施例的过程900随后等待来自收到第一跳托管转发元件的更新转发状态的各个托管转发元件的确认。

[0128] 当过程900(在925)判定并非所有收到更新的转发状态的托管转发元件都回送了确认时，过程900返回925，以等待所述确认。不过，在一些实施例中，在过程把更新的转发状态发送给托管转发元件之后，过去一段时间之后，过程900进入930。

[0129] 当过程900(在925)判定所有收到更新的转发状态的托管转发元件都回送了确认时，一些实施例的过程900(在930)向托管转发元件发送应用第一跳转发元件的更新的转发状态的命令。在一些实施例中，当托管转发元件利用更新的转发状态，转发分组时，托管转发元件把版本信息(例如，版本位)包含在分组中，以致非第一跳托管转发元件利用更新的转发状态，转发这些分组。

[0130] 过程900随后视情况(在935)向托管转发元件发送从托管转发元件中除去旧的转发状态的命令。在一些实施例中，控制器群集把托管转发元件配置成在收到新的转发状态之后，在过去一段时间之后，除去旧的转发状态，而不是向托管转发元件发送命令。该过程随后结束。

[0131] 图9图解说明一些实施例的由网络控制器群集进行的过程900，下一附图(图7)图解说明一些实施例的由托管转发元件进行的过程。图10概念地图解说明一些实施例进行的转发数据的过程1000。过程1000由利用转发状态，以起非第一跳转发元件作用的托管转发元件进行。

[0132] 通过(在1005)利用当前转发状态(旧的转发状态)，转发到来的分组，开始处理

1000。到来的分组不是来自托管转发元件直接面接的终端机器。即，该托管转发元件在这些分组的路径中，但是不是作为分组的第一跳转发元件。转发状态接收自控制器群集，或者通过把转发状态发送给托管转发元件管理该托管转发元件的机架控制器。

[0133] 之后，过程1000(在1010)从控制器群集，接收更新的转发状态。在一些实施例中，该转发状态由控制器群集更新，并且包括从LCP数据转换来的CPCP数据。托管转发元件现在具有旧的转发状态和更新的转发状态。

[0134] 过程1000随后(在1015)利用旧的或者更新的转发状态，转发到来的分组。在一些实施例中，过程1000根据到来的分组所携带的版本信息，选择使用旧的或者更新的转发状态来转发到来的分组。即，所述版本信息用于匹配安装在托管转发元件中的旧转发状态和更新的转发状态具有的版本信息。

[0135] 在1020，过程1000视情况从控制器群集接收除去旧的转发状态的命令。在一些实施例中，托管转发元件不接收除去旧状态的明确命令。托管转发元件改为被控制器群集配置成在收到更新的转发状态之后，在过去一段时间之后，除去旧的转发状态。过程1000随后(在1025)除去旧的转发状态。该过程随后结束。

[0136] C.模拟外部依赖关系

[0137] 上面的讨论考虑了对网络控制系统中的事务性提出的要求，和跨越网络的事务更新的实现(即，通过分离对第一跳处理的更新和对非第一跳处理的更新)。网络控制系统还事务地计算对网络转发状态的更新。

[0138] 显然，在事务地更新无论什么之前，在已知策略变化的情况下，网络控制系统使计算收敛。如上所述，一些实施例的网络控制系统利用nLog 表映射引擎来实现系统的网络控制器。在一些实施例中，nLog引擎使计算达到其定点-即，nLog引擎根据迄今收到的输入变化，计算转发状态的所有变化。

[0139] 在高层次，达到局部定点较简单：不向计算引擎(即，nLog引擎)提供任何新的更新并等到该引擎不再有工作要做就够了。然而，在联网中，稍微宽泛地解释定点的定义：尽管计算可达到定点，不过不意味计算得到能够被进一步向下推送给托管转发元件的结果。例如，当改变隧道的目的地端口时，UPCP数据可能只具有该目的地端口映射到的物理端口的占位符。

[0140] 结果却是计算可能依赖于在计算可结束并且达到与可以使用并向下推送的转发状态对应的定点之前，必须应用的外部变化。继续该例子，只有在设置了将导致端口号的隧道端口之后，才可填充流表项中的关于端口号的占位符。在这种情况下，在对于任何新的外部状态(例如，归因于创建的隧道的端口号)的依赖关系被满足之前，不能认为UPCP计算已结束。

[0141] 从而，在计算中必须考虑这些外部依赖关系，并将其包含在定点的考虑中。即，在计算局部结束并且没有外部依赖关系仍然未被满足之前，未达到定点。在一些实施例中，nLog计算建立在增加和去除中间结果之上；配置或者对外部状态的每个更改导致计算状态的增加或去除。

[0142] 为了在UPCP计算中考虑外部依赖关系，nLog引擎应：

[0143] (1)当更改导致在新的UPCP数据能够被向下推送之前应被增加的状态时(例如，当为了完成UPCP流表项，必须创建隧道时)，使所述更改立即被应用。在所述更改的结果(例

如，新的端口号）被返回nLog计算引擎之前，nLog计算引擎不得不认为定点是不能达到的。

[0144] (2)当更改导致会影响当前的UPCP数据的状态时（例如，除去旧的隧道），在提交事务（即，实现新的网络转发状态）之前，不能允许更新通过。只有在事务已被提交之后，才应应用所述更新。否则，在事务被提交之前，网络转发会变化。在施行上述规则的情况下，不能支持外部资源的原子级更改。幸运的是，大部分的资源更改可被模拟成增加/去除；例如，当改变代表朝向特定目的地的隧道的端口的配置时，新配置可被视为暂时与旧端口共存的新端口。

[0145] 从而，在高层次，上述方法建立在紧接于旧配置，增加新配置的能力之上。在使路径中的托管资源连网的情况下，情况一般就是这样。在存在约束的情况下（比如说由于某个原因，朝向相同IP的两条隧道不能存在），该方法不起作用，从而不能提供这种变化的原子性。

[0146] 图11概念地图解说明一些实施例进行的事务地计算转发状态，并把转发状态发送给一组托管转发元件的过程1100。在一些实施例中，过程 1100由物理控制器，或者接收UPCP数据并把UPCP数据转换成CPCP数据的机架控制器进行。通过（在1105）从逻辑控制器或物理控制器，接收包含UPCP数据的一组转发状态变化（例如，数据元组），开始该过程。

[0147] 过程1100随后（在1105）判定收到的变化是否具有外部依赖关系。在一些实施例中，当处理变化的控制器不具有处理所述变化的全部信息，从而不得不从另一个控制器或者从托管转发元件获得缺少的信息时，所述变化具有外部依赖关系。例如，为了把指定托管转发元件应从托管转发元件的端口建立隧道的UPCP变化转换成CPCP变化，在CPCP变化中，需要所述端口的端口号。于是，在从托管转发元件收到实际端口号之前，不能创建CPCP变化。

[0148] 当过程1100（在1105）判定收到的变化不具有外部依赖关系时，过程 1100进入将在下面进一步说明的1115。当过程1100（在1105）判定变化具有外部依赖关系时，过程1100（在1110）根据具有外部依赖关系的接收变化，计算一组输出变化，并把计算的变化发送给托管转发元件。所述一组输出变化向托管转发元件请求缺少的信息。过程1100随后返回1105，以便从逻辑控制器或物理控制器，或者从可回送缺少的信息的托管转发元件接收更多的变化，以便解决外部依赖关系。

[0149] 当过程1100（在1105）判定收到的变化没有外部依赖关系时，过程 1100（在1110）计算一组输出变化（例如，通过把UPCP变化转换成CPCP 变换），随后（在1115）判定过程1100是否达到定点，所述定点指示输出变化的事务计算的结束。换句话说，过程1100判定收到的变化是否已被完全处理，并且过程1100目前不再有要处理的变化。

[0150] 当过程1100（在1115）判定过程未到达定点地时，过程1100返回 1115，以继续根据输入变化，计算输出变化。否则，过程1125把输出变化发送给托管转发元件。过程随后结束。

[0151] D.对于事务更新的计算要求

[0152] 上面的讨论指出事务地计算更新，随后把它们推送给第一跳边缘转发元件就足够了。从而，除了计算之外，对系统再加以一个额外的要求：事务通信通道。

[0153] 因而，在一些实施例中，朝向转发元件的通信通道（例如，从输入转换控制器到逻辑控制器，从逻辑控制器到物理控制器，从物理控制器到机架控制器或托管转发元件，和/或从机架控制器到托管转发元件的通信通道）支持对被完全应用或者根本未被应用的各个

单元的成批变化。在这些实施例中的一些实施例中，通信通道只支持“分界线”(即，开始和结束标记)的概念，所述分界线用信号向接收器通知事务的结束。接收控制器或托管转发元件仅仅对各个更新排队，直到它收到如上所述的分界线为止。另外，通道必须维持发送的更新的顺序，或者至少保证在分界线之前发送的更新不会在分界线之后到达接收器。

[0154] 这样，发送控制器可仅仅把发送更新保持为当计算取得进展时的状态，并且一旦它确定达到了定点，它就用信号向进行接收的第一跳转发元件通知事务的结束。如下进一步所述，一些实施例中的通信通道还支持同步提交，以致发送控制器知道事务何时已被处理(通过达到定点而被计算)和被进一步向下推送(如果需要的话)。应注意在如下所述的嵌套事务的情况下，所述同步提交可在网络控制系统的各个较低层，在内部导致更多的同步提交。

[0155] 上面说明了实现全网络事务，下面的节III说明通过朝向托管转发元件的几个通道，实现事务。

[0156] III.嵌套事务

[0157] 通过如上参考图5-10所述，当提到转发状态更新时，分离网络的起点和网络的剩余部分，一些实施例的网络控制系统有效地创建嵌套事务结构：一个全局事务可被认为包括两个子事务，一个子事务用于第一跳端口，一个子事务用于非第一跳端口。不管该解决方案是以最细的粒度 (通过了解在网络的中间的每个物理跳，并建立所需的状态)管理非第一跳端口，还是假定外部实体能够以事务的方式建立跨网络的连接，该方法都保持相同。

[0158] 在一些实施例中，这推广到允许根据一组更细粒的事务，创建基本的分发式事务的原理。特别地，考虑具有朝向控制器的多个通信通道的托管转发元件，同时每个通道提供事务性，但是不支持跨通道的事务。即，通道不支持分布式事务。在这种情况下，完全相同的组合方法同样起作用。只要通道中的可被视为主通道的一个通道使其事务得到应用，其它通道的状态就都不被使用。借助这种结构，在主通道提交事务之前，可再次“准备”辅通道(正如在第一跳托管转发元件提交其事务之前，准备非第一跳托管转发元件一样)。这样，净结果是当在第一跳托管转发元件的事务被提交时，获得提交的单一的全局事务。

[0159] 图12图解说明控制器1210建立了到其的2个通信通道1215和1220，以便向其发送更新的托管转发元件1205。特别地，图12按4个不同的阶段1201-1204，图解说明在来自通道1215的更新到达之前，托管转发元件1205不利用通过通道接收的更新。

[0160] 控制器1210类似于图2的控制器200。在一些实施例中，控制器1210 是把UPCP数据转换成CPCP数据的物理控制器。在其它实施例中，控制器1210是把从物理控制器接收的UPCP数据转换成CPCP数据的机架控制器。

[0161] 在一些实施例中，控制器1210与托管转发元件1205建立两条通道 1215和1220。通道1215是利用控制托管转发元件1205的转发平面(例如，转发表)的通信协议建立的。例如，OpenFlow协议提供用于向托管转发元件1205中的流表项增加流表项，从托管转发元件1205中的流表项中除去流表项，和更改托管转发元件1205中的流表项的命令。通道1220 是利用配置协议建立的。托管转发元件1205通过通道1220接收配置信息。在一些实施例中，托管转发元件1205把配置信息保存在配置数据库 (未示出)中。在一些实施例中，配置信息包括用于配置托管转发元件1205 的信息，比如配置入口端口、出口端口、端口的QoS配置等的信息。为了例示和讨论的简明起见，图12例示了流表项和配置信息，并将称为转发状态。

[0162] 托管转发元件1205直接面接几个终端机器(未示出),并利用通过这两个通道从控制器1205接收的转发状态,往来于所述终端机器转发数据。这两个通道都不支持分布式事务,不过通过把来自通道1220的事务嵌套 (批量集中)到来自通道1215的事务中,或者反过来,托管转发元件1205 跨这两个通道实现分布式事务。例如,在一些实施例中,托管转发元件 1205把通道1215指定成主通道,把通道1220指定成辅通道。托管转发元件1205保持通过通道接收的转发状态的应用,直到托管转发元件1205 通过主通道收到事务为止。

[0163] 在第一阶段1201,托管转发元件1205已通过通道1220和1220,从控制器1210收到一组变化(例如,数据元组)。通过通道1215收到的这些变化包括流表项。通过通道1220收到的变化包括配置信息。

[0164] 在阶段1201,已通过主通道1215,收到变化1-1。通过辅通道1220,收到变化2-1和2-2。托管转发元件已把这些变化保存在存储结构1230 中,但是还未开始利用这些变化来转发到来的分组(未示出),因为托管转发元件1205还未通过主通道1215,收到完整的事务。托管转发元件利用当前的转发状态,转发到来的分组。

[0165] 第一阶段1201还表示变化1-2通过主通道1215,到达托管转发元件 1205,变化2-3通过辅通道1220,到达托管转发元件1205。变化2-3被描述成具有粗边框的平行四边形,以指示变化2-3是通过辅通道1220接收的事务的最后变化,该事务包括变化2-1、2-2和2-3。

[0166] 在第二阶段1202,托管转发元件1205分别通过主通道1215和辅通道1220,收到变化1-2和2-3。托管转发元件1205已把变化1-2和2-3 保存在存储结构1230中,但是还未开始利用这些变化转发到来的分组,和配置托管转发元件1205,因为托管转发元件1205还未通过主通道 1215,收到完整的事务。

[0167] 第二阶段1202还表示变化1-3通过主通道1215,到达托管转发元件 1205。变化1-3被描述成具有粗边框的平行四边形,以指示变化1-3是通过辅通道1215接收的事务的最后变化,该事务包括变化1-1、1-2和 1-3。

[0168] 在第三阶段1203,托管转发元件1205已通过主通道1215,收到变化1-3,从而从主通道1215,完整地收到事务。托管转发元件1205从而用两种事务中,通过通道1215和1220接收的变化,更新转发状态。

[0169] 第四阶段1204表示托管转发元件1205提交变化。即,托管转发元件1205利用更新的转发状态转发到来的分组,并配置托管转发元件 1205。这样,托管转发元件1205把通过辅通道接收的事务嵌套到通过主通道接收的事务中,以便实现跨这两个通道的全局事务。

[0170] 图13概念地图解说明一些实施例进行的把通过辅通道接收的事务批量集中到通过主通道接收的事务中的过程1300。一些实施例的过程1300 由通过相对于控制器建立的几个通道从控制器接收转发状态的托管转发元件(例如,图12的托管转发元件1205)进行。控制器可以是作为托管转发元件的控制者的物理控制器,或者与托管转发元件在相同的主机中工作的机架控制器。在一些实施例中,通道之一被指定为主通道,另一个通道被指定为辅通道。

[0171] 通过(在1305)借助主通道和辅通道接收转发状态,开始过程1300。在一些实施例中,通过通道从控制器接收的转发状态包括CPCP数据。具体地,通过主通道到来的转发状态包括转到托管转发元件的控制平面的控制数据。通过辅通道到来的转发状态包括配置数据(例如,用于配置入口端口、出口端口、端口的QoS配置、中间盒实例等的数据)。然而,在一些

实施例中，主通道和辅通道的指定不必取决于通过通道接收的数据的种类，只要把通道之一指定为主通道，把另一个通道指定为辅通道即可。

[0172] 之后，过程1300(在1310)判定过程1300是否已通过主通道收到分界线。如上所述，当在接收设备收到分界线时，所述分界线指示接收设备已完全收到输入的一个事务。在一些实施例中，分界线是添加到变化中的信息。其它实施例的分界线是指示变化的发送者已完全发送一组事务输入的变化本身。

[0173] 当过程1300(在1310)判定还未通过主通道收到分界线时，过程 1300(在1320)把该过程迄今收到的转发状态保存在存储结构中。托管转发元件不利用保存在存储结构中的转发状态。过程1300随后返回1305，以通过通道接收来自控制器群集的更多转发状态。

[0174] 当过程1300(在1310)判定已通过主通道收到分界线时，过程用迄今收到的转发状态更新托管转发元件的转发表和配置数据库。托管转发元件随后利用配置数据配置托管转发元件，并根据根据转发表中的更新的流表项，转发到来的分组。过程1300随后结束。

[0175] 注意，该推广允许把事务嵌套到任意深度，如果需要这样的话。特别地，事务系统可用嵌套事务，内部构建其事务性。用嵌套事务构建事务性的能力不仅在控制器可构成的分层结构中有用，而且在考虑转发元件如何内部为管理转发元件的控制器提供事务接口时也有用。

[0176] 一些实施例的网络控制系统同样通过利用相同的嵌套原理，向不明确支持底层托管资源中的事务性的通信通道引入事务性。考虑具有可容易地扩展的表管线的路径。即使流表更新不支持事务，也易于在现有管线的前面增加一级，并使单一流表项决定应使用状态的哪个版本。从而，通过随后更新单一流表项(这是事务的)，可以事务地更新整个流表。该方法的细节不必暴露给上面的控制器；不过，实际上现在存在事务的分层结构。

[0177] 作为上述实施例的使用情况，从一种控制器版本到另一种控制器版本(即，软件版本)的迁移受益于系统中的事务和定点计算支持。在这种使用情况下，外部升级驱动器运行从一种控制器版本到另一种控制器版本的整个升级过程。驱动器的职责是协调升级以不会出现分组丢失的方式发生。

[0178] 驱动器执行的把较小子事务合成为单一全局事务的整个过程如下：

[0179] (1)一旦需要升级转发状态，驱动器就要求开始网络中间部分(架构) 的新状态的计算。对管理网络中间状态的所有控制器都这样做，预期新的中间状态将与旧的中间状态共存。

[0180] (2)驱动器随后等待每个控制器达到定点，随后同步地把事务向下提交给接收控制器/交换元件。驱动器同步地进行所述提交，因为在提交之后，驱动器知道该状态在交换元件中有效，可被分组使用。

[0181] (3)之后，驱动器要求控制器向新的边缘转发状态更新，所述新的边缘转发状态也利用在(1)中为网络的中间部分建立的新路径。

[0182] (4)同样，驱动器向所有控制器要求定点，随后一旦达到定点，也同步提交更新。

[0183] (5)当驱动器要求旧的网络中间状态的去除时，结束更新。这不需要等待定点和提交；所述去除将与控制器最终向下推送的任何其它变化一起被向下推送。

[0184] IV.电子系统

[0185] 许多上述特征和应用被实现成软件过程，所述软件过程被指定成记录在计算机可

读存储介质(也称为计算机可读介质)上的指令集。当这些指令由一个或多个处理单元(例如,一个或多个处理器、处理器的核心、或者其它处理单元)执行时,它们使所述处理单元进行在指令中指示的动作。计算机可读介质的例子包括(但不限于)CD-ROM、闪速驱动器、RAM芯片、硬盘驱动器、EPROM等。计算机可读介质不包括载波和无线地或者通过有线连接传送的电子信号。

[0186] 在本说明书中,术语“软件”意味包括可被读入内存中,以便由处理器处理的驻留在只读存储器中的固件,或者保存在磁存储器中的应用程序。另外,在一些实施例中,在仍然是不同的软件发明的时候,多个软件发明可被实现成更大的程序的子部分。在一些实施例中,多个软件发明也可被实现成独立的程序。最后,一起地实现这里说明的软件发明的独立程序的任意组合在本发明的范围之内。在一些实施例中,当被安装以在一个或多个电子系统上工作时,软件程序定义执行和实现软件程序的操作的一种或多种具体的机器实现。

[0187] 图14概念地图解说明可实现本发明的一些实施例的电子系统1400。电子系统1400可用于执行上面说明的控制、虚拟化或操作系统应用程序任意之一。电子系统1400可以是计算机(例如,桌上型计算机、个人计算机、平板计算机、服务器计算机、大型机、刀片计算机等)、电话机、PDA或任意其它种类的电子设备。这样的电子系统包括各种计算机可读介质,和用于各种其它种类的计算机可读介质的接口。电子系统1400包括总线1405、处理器1410、系统内存1425、只读存储器1430、永久性存储设备1435、输入设备1440和输出设备1445。

[0188] 总线1405总体代表通信地连接电子系统1400的众多内部设备的所有系统总线、外围总线和芯片集总线。例如,总线1405通信地连接处理器1410和只读存储器1430、系统内存1425和永久性存储设备1435。

[0189] 从这些各个存储单元,处理器1410取回待执行的指令和待处理的数据,以便执行本发明的过程。在不同的实施例中,处理器可以是单一处理器或者多核处理器。

[0190] 只读存储器(ROM)1430保存处理器1410和电子系统的其它模块需要的静态数据和指令。另一方面,永久性存储设备1435是读-写存储设备。该设备是即使光电子系统1400关闭时,也保存指令和数据的非易失性存储器。本发明的一些实施例利用大容量存储设备(比如磁盘或光盘及其对应的盘驱动器),作为永久性存储设备1435。

[0191] 其它实施例利用可拆卸的存储设备(比如软盘、闪速驱动器等),作为永久性存储设备。类似于永久性存储设备1435,系统内存1125是读- 写存储设备。然而,不同于存储设备1435,系统存储器是易失性读-写存储器,比如随机存取存储器。系统内存保存在运行时,处理器需要的一些指令和数据。在一些实施例中,本发明的过程被保存在系统内存1425、永久性存储设备1435和/或只读存储器1430中。从这些各个存储单元,处理器1410取回待执行的指令和待处理的数据,以便执行一些实施例的过程。

[0192] 总线1405还连接到输入和输出设备1440和1445。输入设备使用户能够向电子系统传达信息和选择命令。输入设备1440包括字母数字键盘和指示设备(也称为“光标控制设备”)。输出设备1445显示由电子系统生成的图像。输出设备包括打印机和显示设备,比如阴极射线管(CRT)或液晶显示器(LCD)。一些实施例包括同时起输入设备和输出设备作用的设备,比如触摸屏。

[0193] 最后,如图14中所示,总线1405还通过网络适配器(未示出),把电子系统1400耦接到网络1465。按照这种方式,计算机可以是计算机网络(比如局域网(“LAN”)、广域网

(“WAN”)或企业内部网),或者网络之网(比如因特网)的一部分。电子系统1400的任意或全部组件可以和本发明一起使用。

[0194] 一些实施例包括电子组件,比如微处理器,把计算机程序指令保存在机器可读或计算机中读介质(另一方面,称为计算机可读存储介质、机器可读介质或机器可读存储介质)中的存储器和内存。所述计算机可读介质的一些例子包括RAM、ROM、只读光盘(CD-ROM)、可记录光盘(CD-R)、可重写光盘(CD-RW)、只读数字通用光盘(例如,DVD-ROM、双层DVD-ROM)、各种可记录/可重写DVD(例如,DVD-RAM、DVD-RW、DVD+RW等)、闪存(例如,SD卡、小型SD卡、微型SD卡等)、磁性和/或固态硬盘驱动器、只读和可记录蓝光®光盘、超高密度光盘、任何其它光或磁介质、以及软盘。计算机可读介质可保存可由至少一个处理单元执行,并且包含用于实现各种操作的多个指令集的计算机程序。计算机程序或计算机代码的例子包括比如由编译器产生的机器代码,和包括通过利用解释器,由计算机、电子组件或微处理器执行的高级代码的文件。

[0195] 尽管上面的讨论主要涉及执行软件的微处理器或多核处理器,不过一些实施例由一个或多个集成电路,比如专用集成电路(ASIC)或现场可编程门阵列(FPGA)实现。在一些实施例中,所述集成电路执行保存在电路本身上的指令。

[0196] 本说明书中使用的术语“计算机”、“服务器”、“处理器”和“存储器”都指的是电子或其它技术设备。这些术语把人或人群排除在外。对本说明书来说,术语“显示”意味电子设备上的显示。本说明书中使用的术语“计算机可读介质”和“机器可读介质”完全局限于以计算机可读的形式,保存信息的有形物理物体。这些术语排除任何无线信号、有线下载信号和任何其它短暂的信号。

[0197] 尽管关于众多的具体细节说明了本发明,不过,本领域的普通技术人员会认识到,可以用其它具体形式体现本发明,而不脱离本发明的精神。另外,许多附图(包括图9、6、10、7、11和13)概念地图解说明了各种过程。这些过程的具体操作可以不按照表示和说明的严格顺序进行。可以不用连续的一系列操作进行所述具体操作,在不同的实施例中,可以进行不同的具体操作。此外,过程可以利用几个子过程来实现,或者被实现成更大的宏过程的一部分。

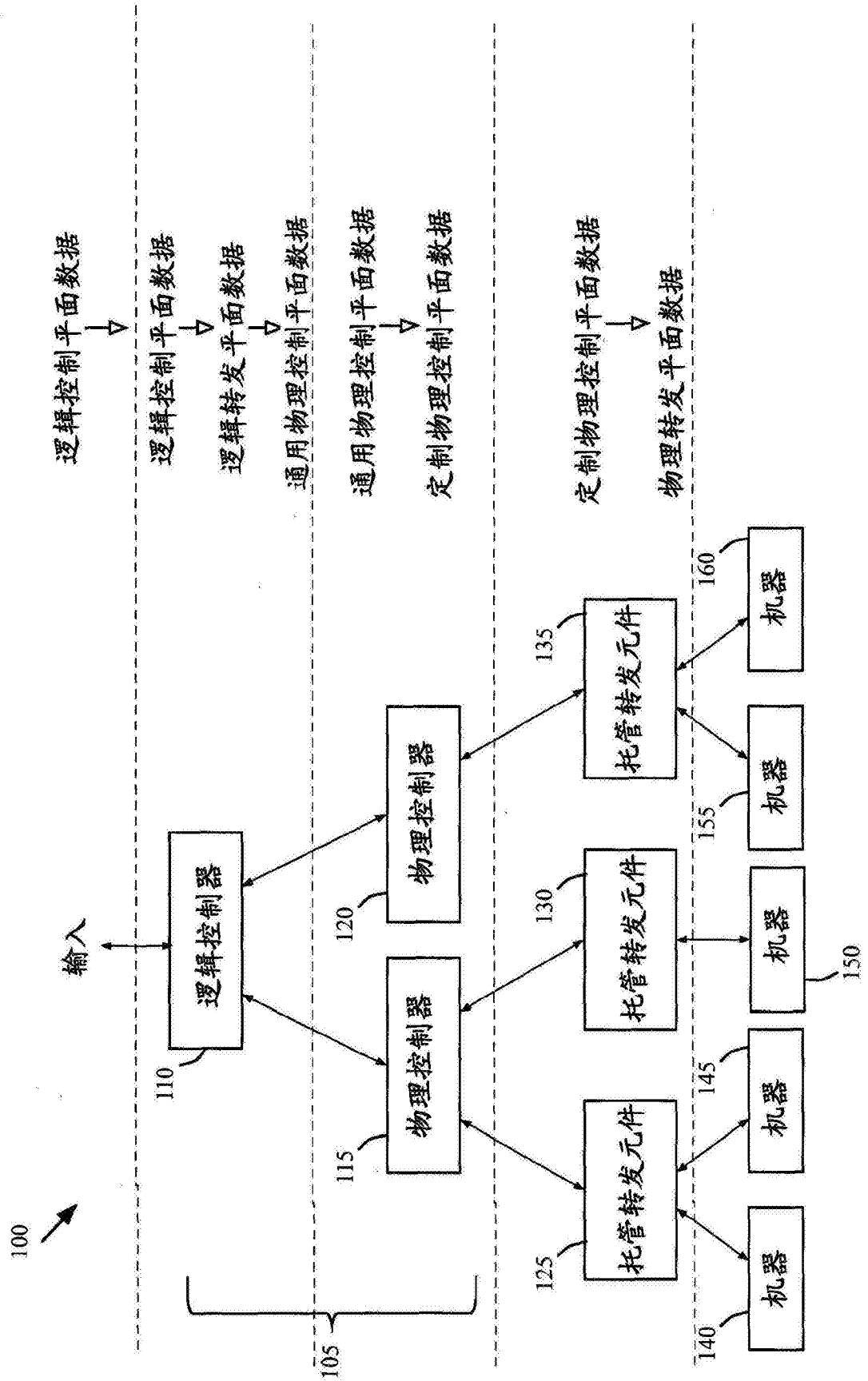

逻辑控制平面数据
逻辑控制平面数据
逻辑转发平面数据
通用物理控制平面数据
通用物理控制平面数据
定制物理控制平面数据
定制物理控制平面数据
物理转发平面数据
100
输入
逻辑控制器
110
物理控制器
120
物理控制器
115
托管转发元件
135
托管转发元件
130
托管转发元件
125
机器
160
机器
155
机器
150
机器
145
机器
140
105

图1

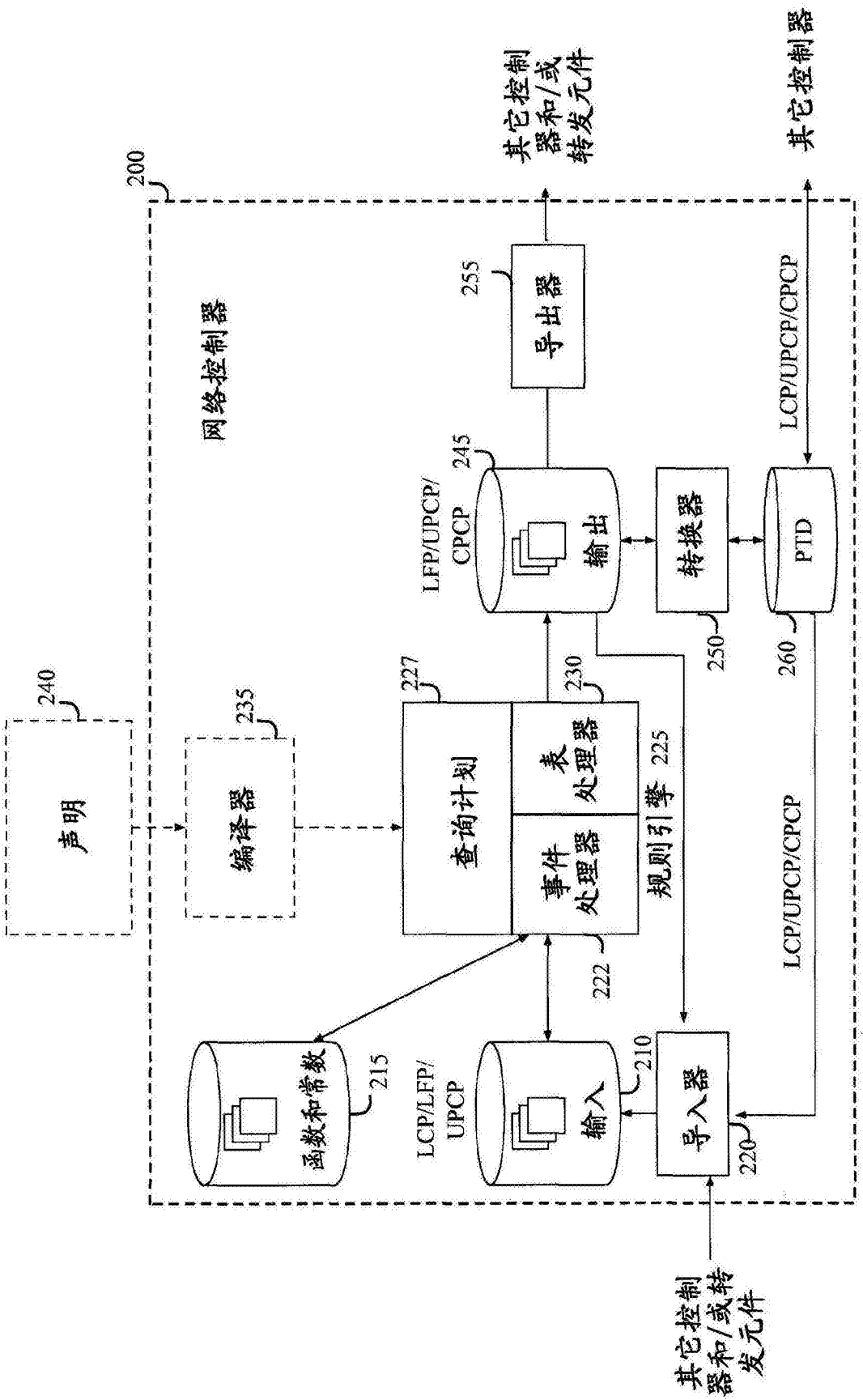

200
网络控制器
其它控制器和/或转发元件
其它控制器
255
导出器
245
LFP/UPCP/CPCP
输出
转换器
250
PTD
260
LCP/UPCP/CPCP
240
声明
235
编译器
227
查询计划
230
表处理器
事件处理器
规则引擎 225
222
LCP/UPCP/CPCP
函数和常数
215
LCP/LFP/UPCP
输入
210
导入器
220
其它控制器和/或转发元件

图2

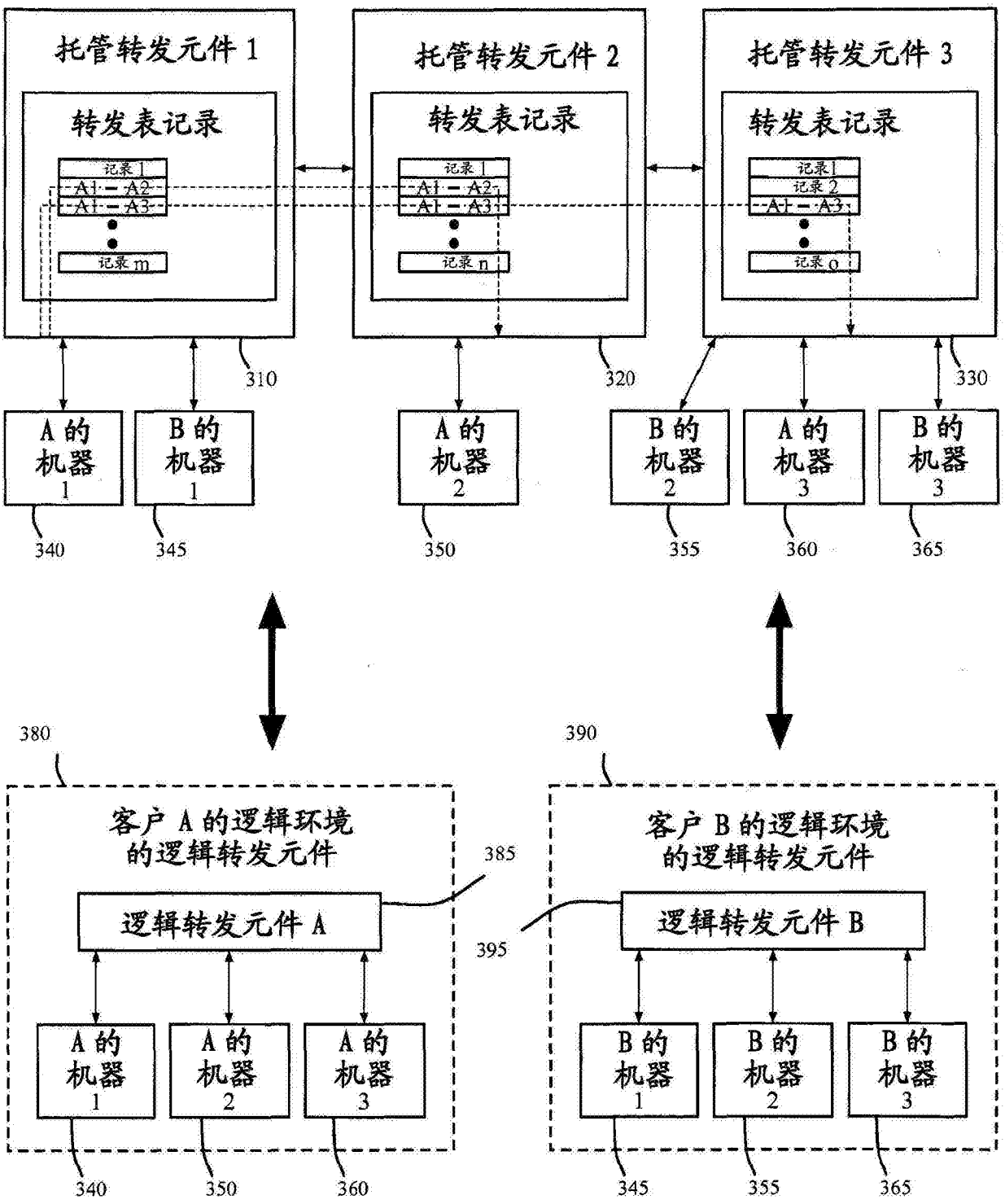

托管转发元件 1
转发表记录
记录 1
A1 = A2
A1 = A3
记录 m
托管转发元件 2
转发表记录
记录 1
A1 = A2
A1 = A3
记录 n
托管转发元件 3
转发表记录
记录 1
记录 2
A1 = A3
记录 o
310
320
330
A 的机器 1
B 的机器 1
A 的机器 2
B 的机器 2
A 的机器 3
B 的机器 3
340
345
350
355
360
365
380
390
客户 A 的逻辑环境的逻辑转发元件
客户 B 的逻辑环境的逻辑转发元件
385
395
逻辑转发元件 A
逻辑转发元件 B
A 的机器 1
A 的机器 2
A 的机器 3
B 的机器 1
B 的机器 2
B 的机器 3
340
350
360
345
355
365

图3

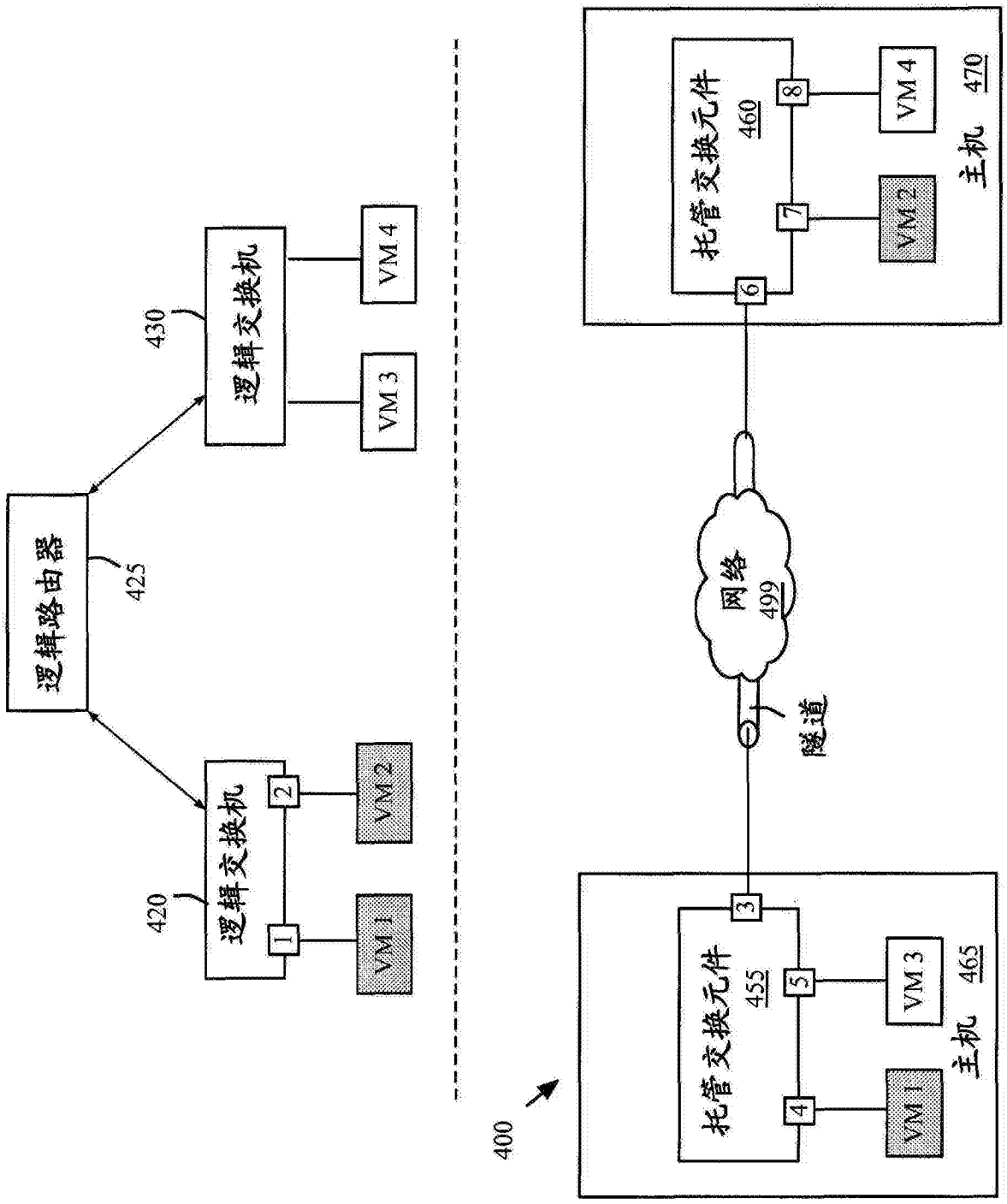

逻辑路由器
425
逻辑交换机
420
1
2
VM 1
VM 2
逻辑交换机
430
VM 3
VM 4
400
托管交换元件
455
3
4
5
VM 1
VM 3
主机
465
网络
499
隧道
托管交换元件
460
6
7
8
VM 2
VM 4
主机
470

图4

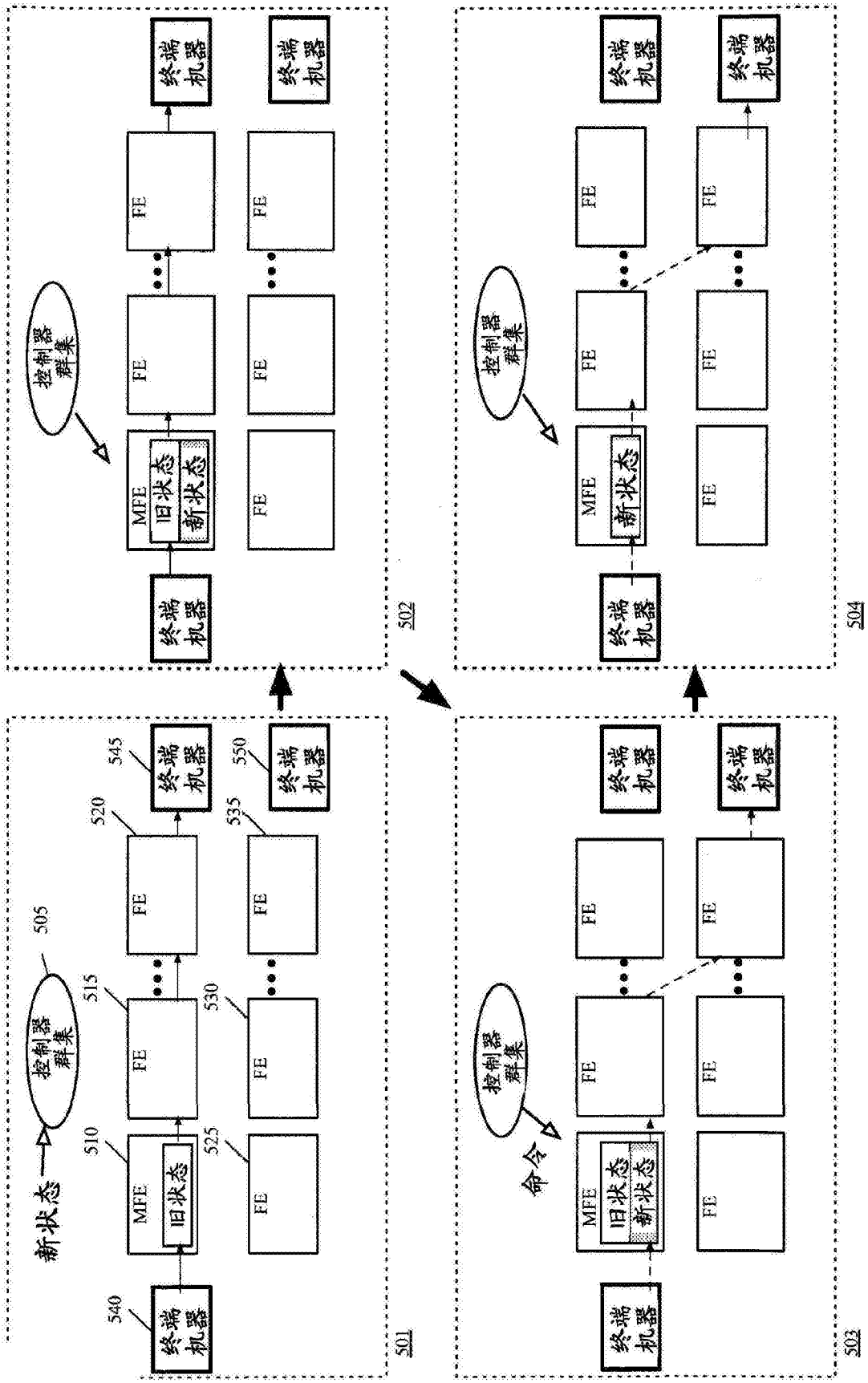

控制器群集
新状态
命令
MFE
旧状态
新状态
FE
终端机器
501
502
503
504
505
510
515
520
525
530
535
540
545
550

图5

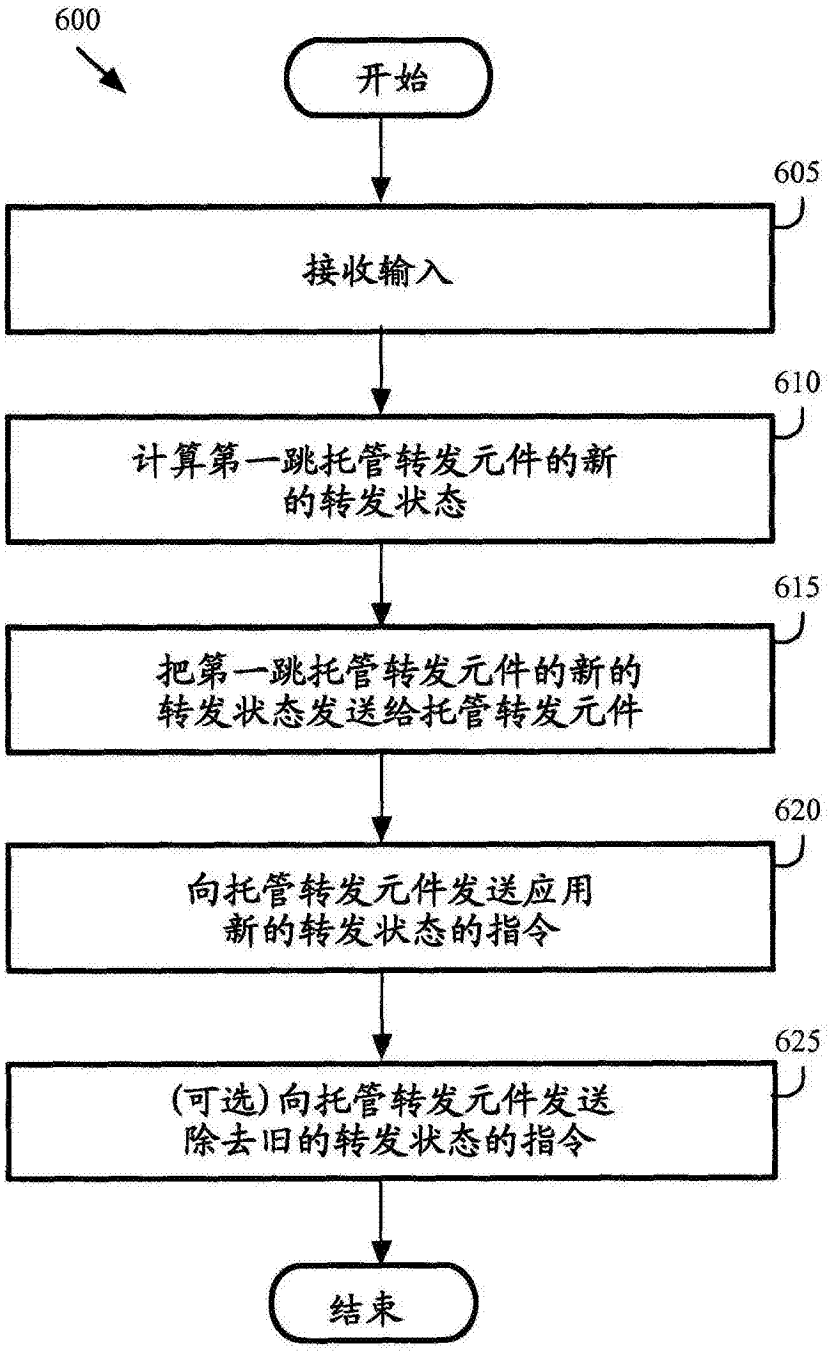

600
开始
605
接收输入
610
计算第一跳托管转发元件的新的转发状态
615
把第一跳托管转发元件的新的转发状态发送给托管转发元件
620
向托管转发元件发送应用新的转发状态的指令
625
(可选)向托管转发元件发送除去旧的转发状态的指令
结束

图6

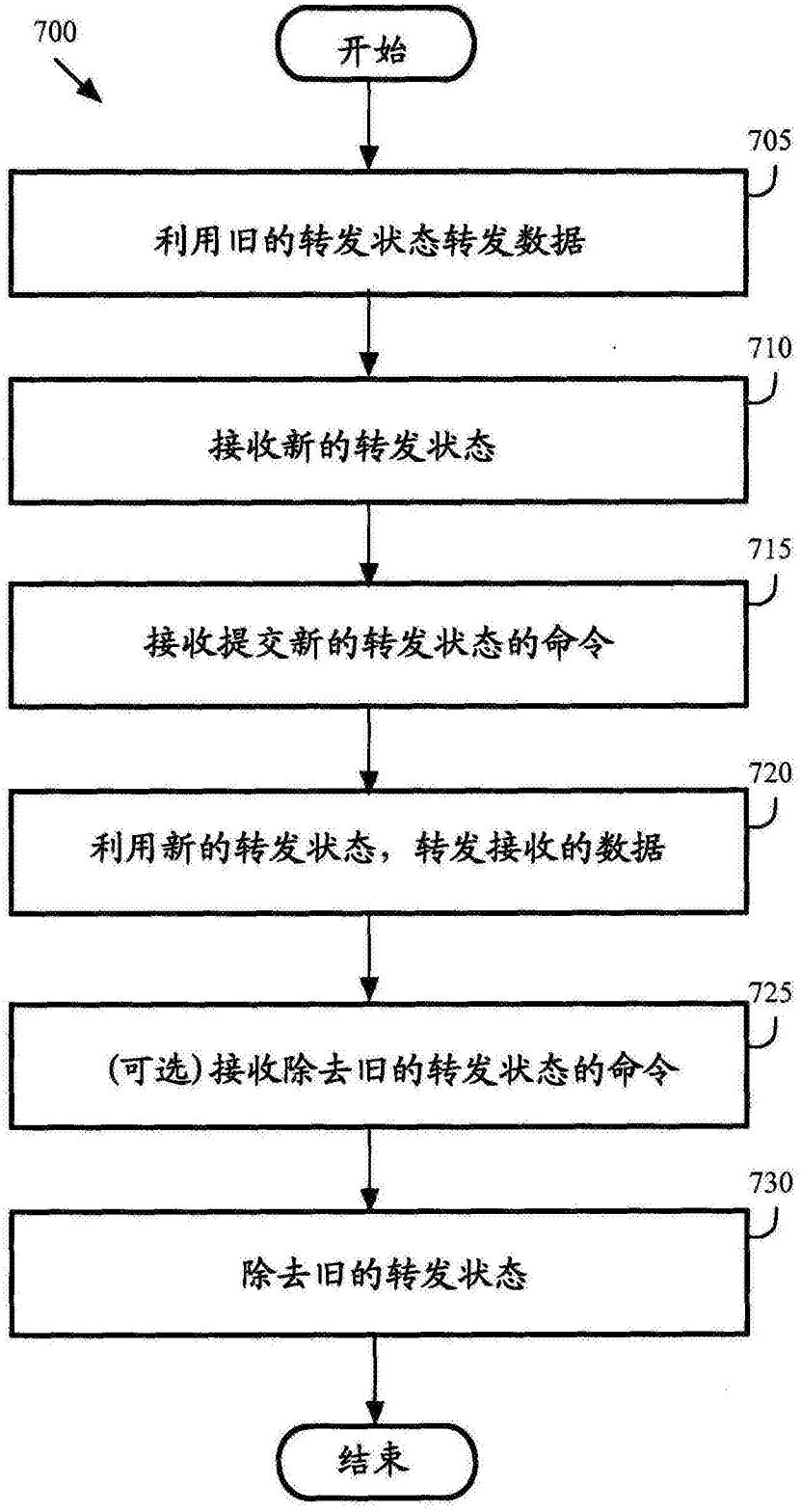

700
开始
705
利用旧的转发状态转发数据
710
接收新的转发状态
715
接收提交新的转发状态的命令
720
利用新的转发状态，转发接收的数据
725
(可选)接收除去旧的转发状态的命令
730
除去旧的转发状态
结束

图7

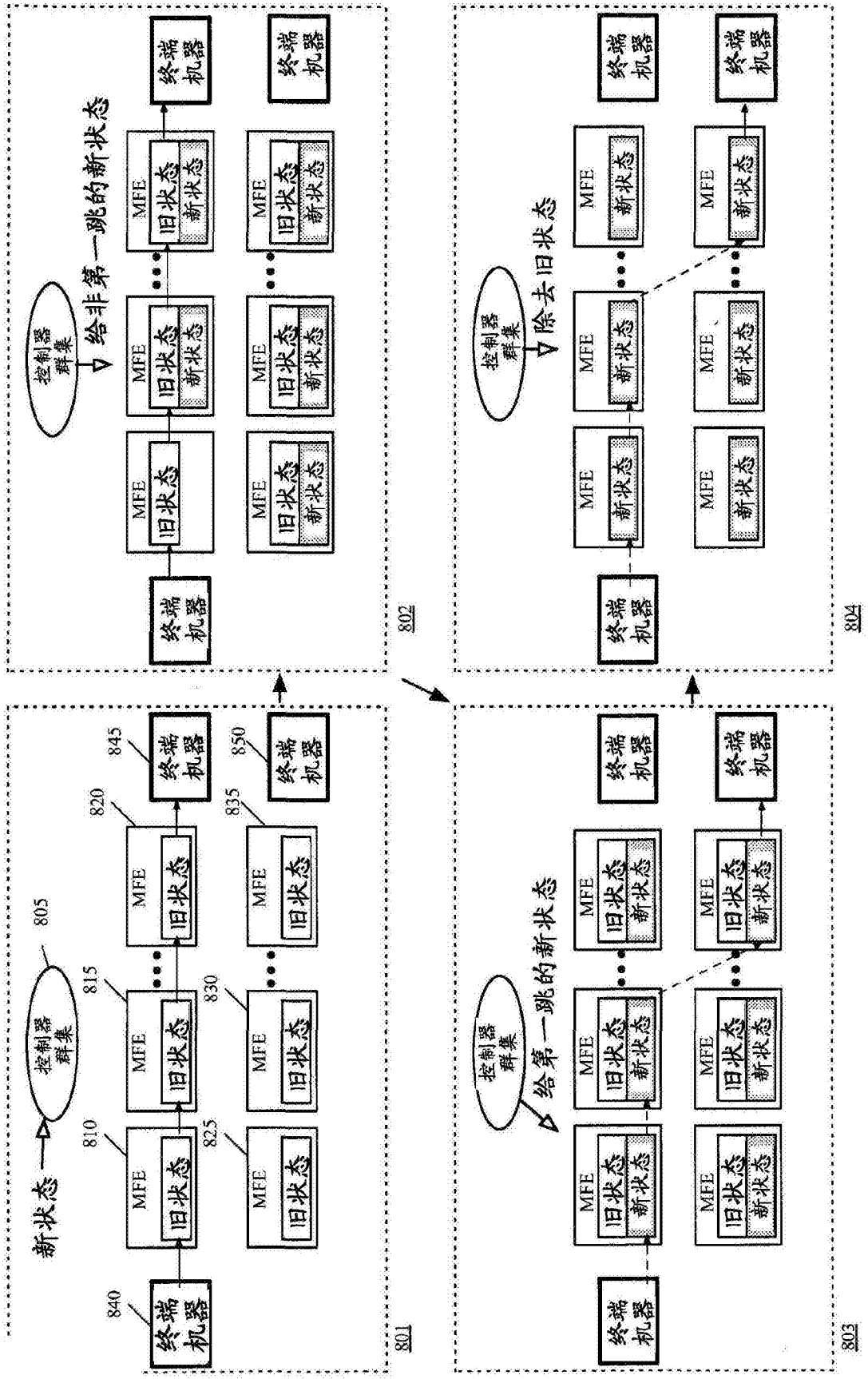

控制器群集
新状态
805
给非第一跳的新状态
除去旧状态
给第一跳的新状态
MFE
旧状态
新状态
终端机器
801
802
803
804
810
815
820
825
830
835
840
845
850

图8

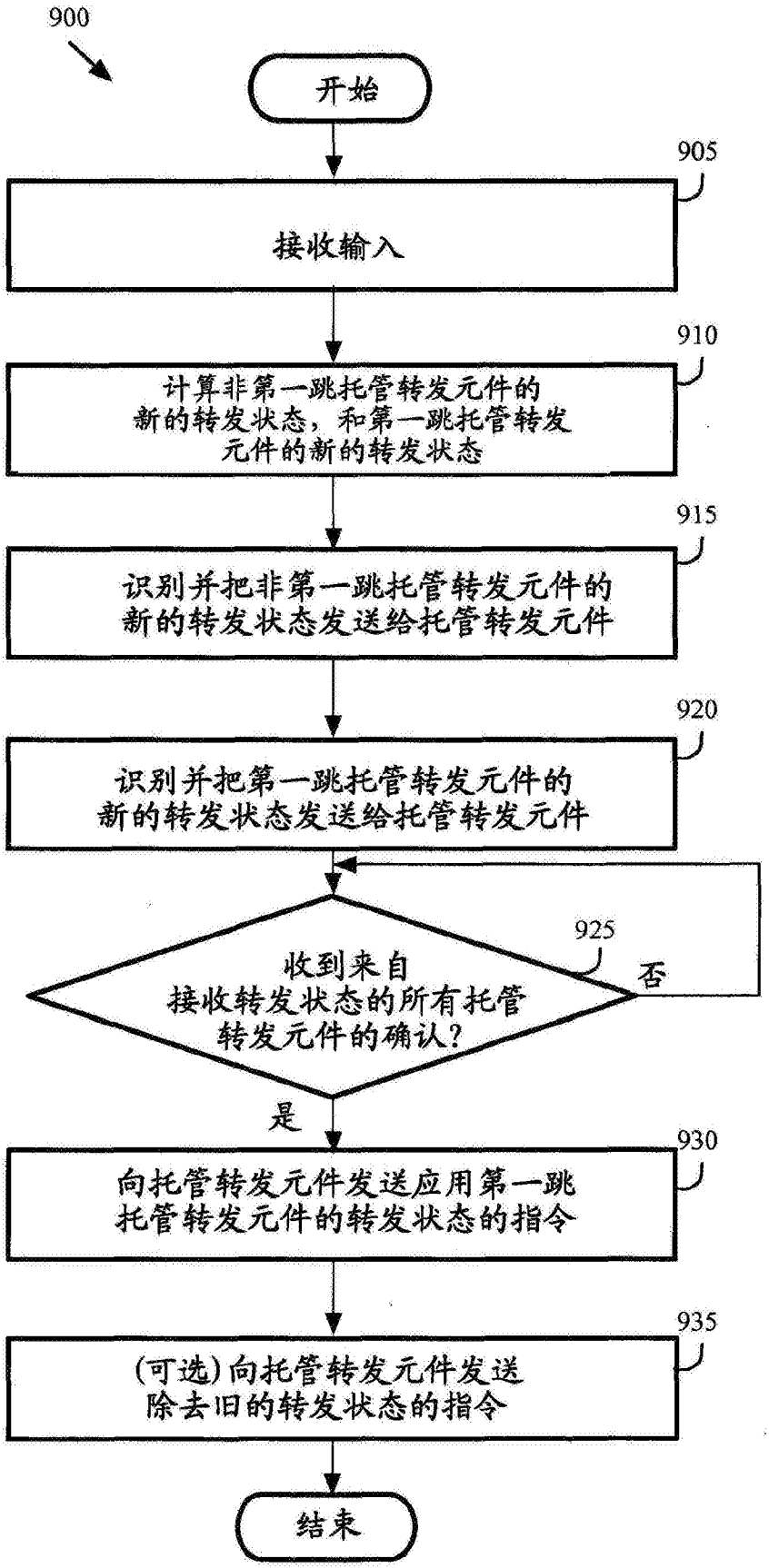

900
开始
905
接收输入
910
计算非第一跳托管转发元件的新的转发状态，和第一跳托管转发元件的新的转发状态
915
识别并把非第一跳托管转发元件的新的转发状态发送给托管转发元件
920
识别并把第一跳托管转发元件的新的转发状态发送给托管转发元件
925
收到来自接收转发状态的所有托管转发元件的确认?
否
是
930
向托管转发元件发送应用第一跳托管转发元件的转发状态的指令
935
(可选)向托管转发元件发送除去旧的转发状态的指令
结束

图9

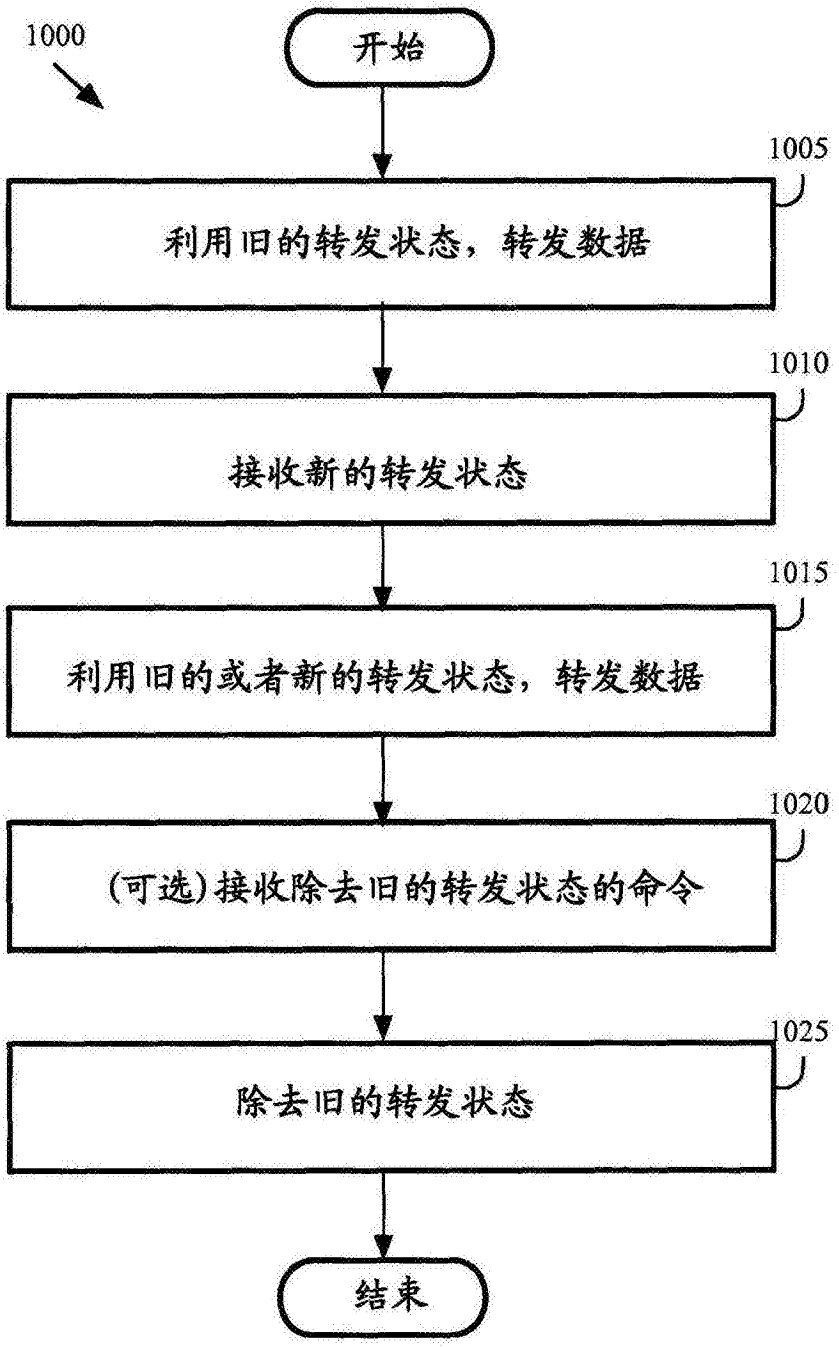

1000
开始
1005
利用旧的转发状态，转发数据
1010
接收新的转发状态
1015
利用旧的或者新的转发状态，转发数据
1020
(可选)接收除去旧的转发状态的命令
1025
除去旧的转发状态
结束

图10

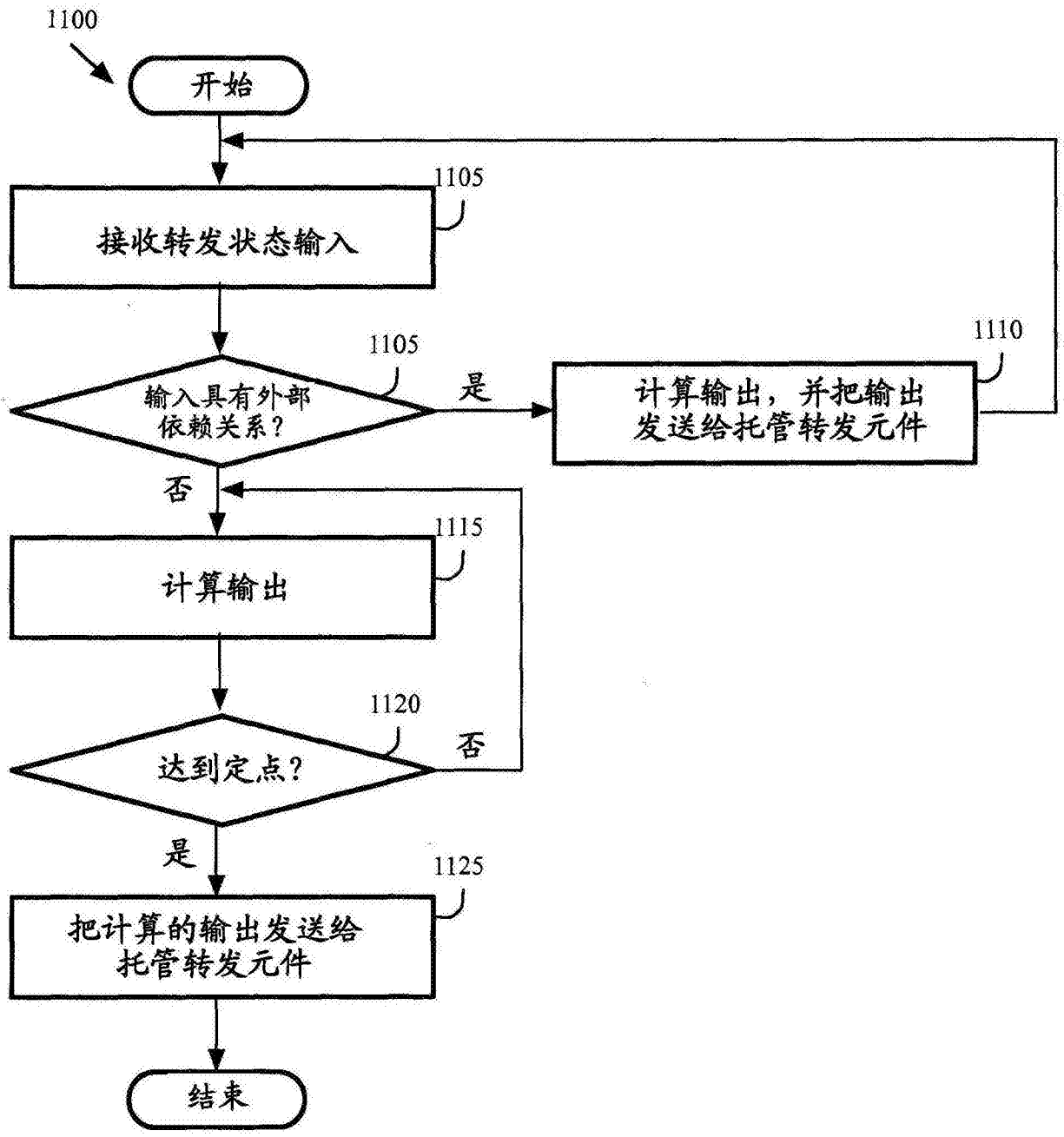

1100
开始
1105
接收转发状态输入
1105
输入具有外部依赖关系？
是
1110
计算输出，并把输出发送给托管转发元件
否
1115
计算输出
1120
达到定点？
否
是
1125
把计算的输出发送给托管转发元件
结束

图11

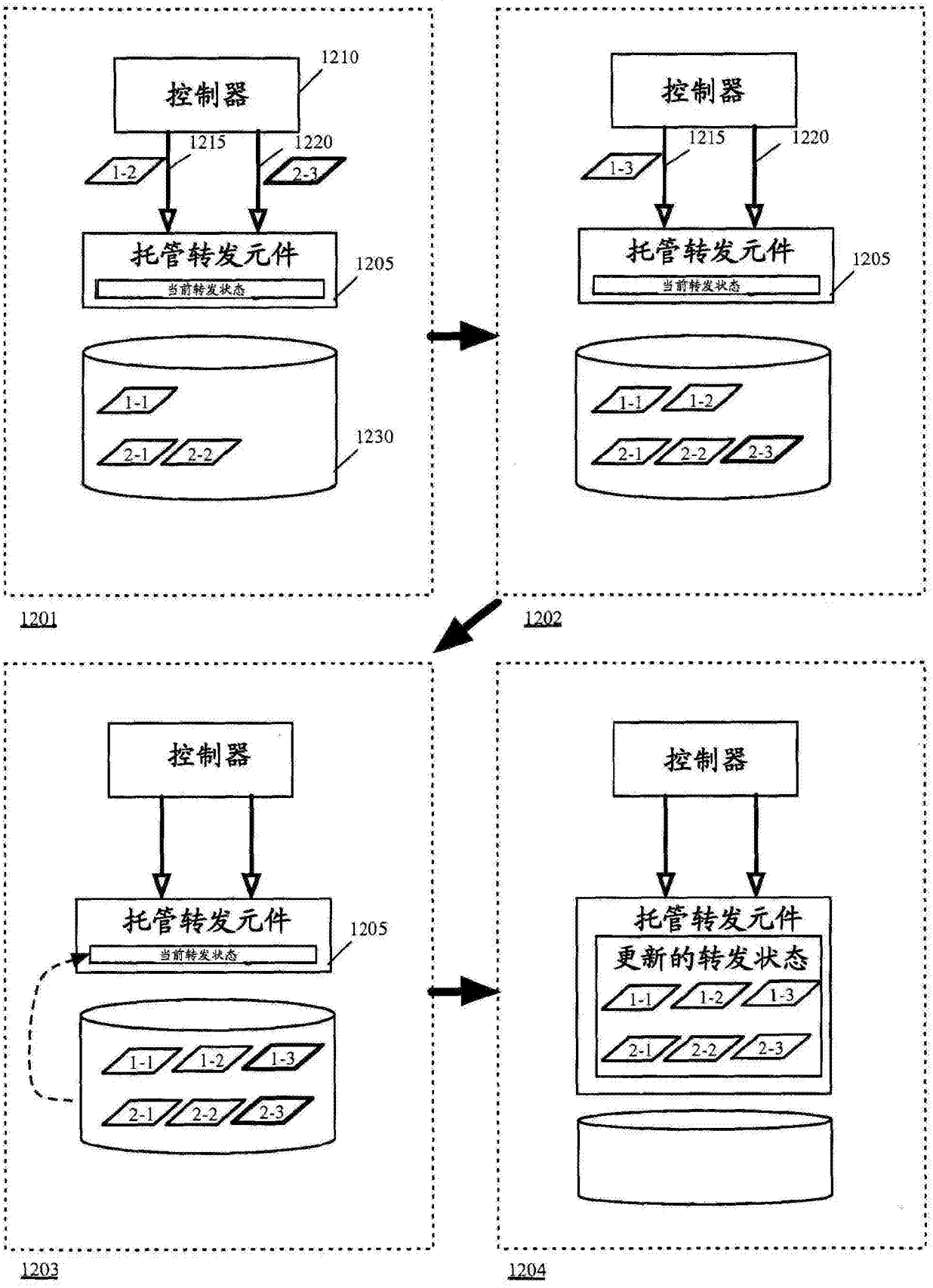

控制器
1210
1215
1220
1-2
2-3
托管转发元件
当前转发状态
1205
1-1
2-1
2-2
1230
1201
控制器
1215
1220
1-3
托管转发元件
当前转发状态
1205
1-1
1-2
2-1
2-2
2-3
1202
控制器
托管转发元件
当前转发状态
1205
1-1
1-2
1-3
2-1
2-2
2-3
1203
控制器
托管转发元件
更新的转发状态
1-1
1-2
1-3
2-1
2-2
2-3
1204

图12

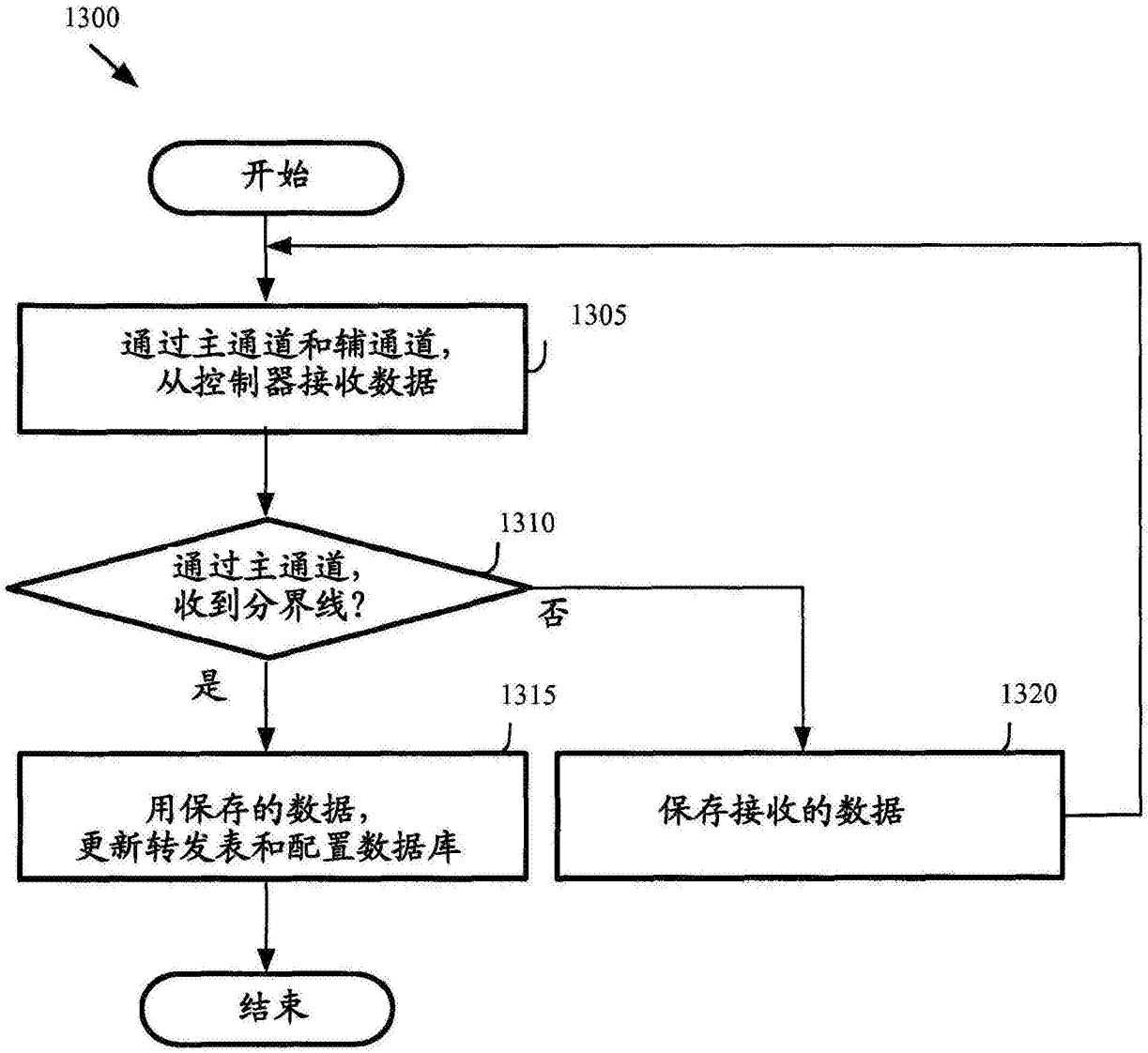

1300
开始
通过主通道和辅通道，
从控制器接收数据
1305
通过主通道，
收到分界线？
1310
否
是
用保存的数据，
更新转发表和配置数据库
1315
保存接收的数据
1320
结束

图13

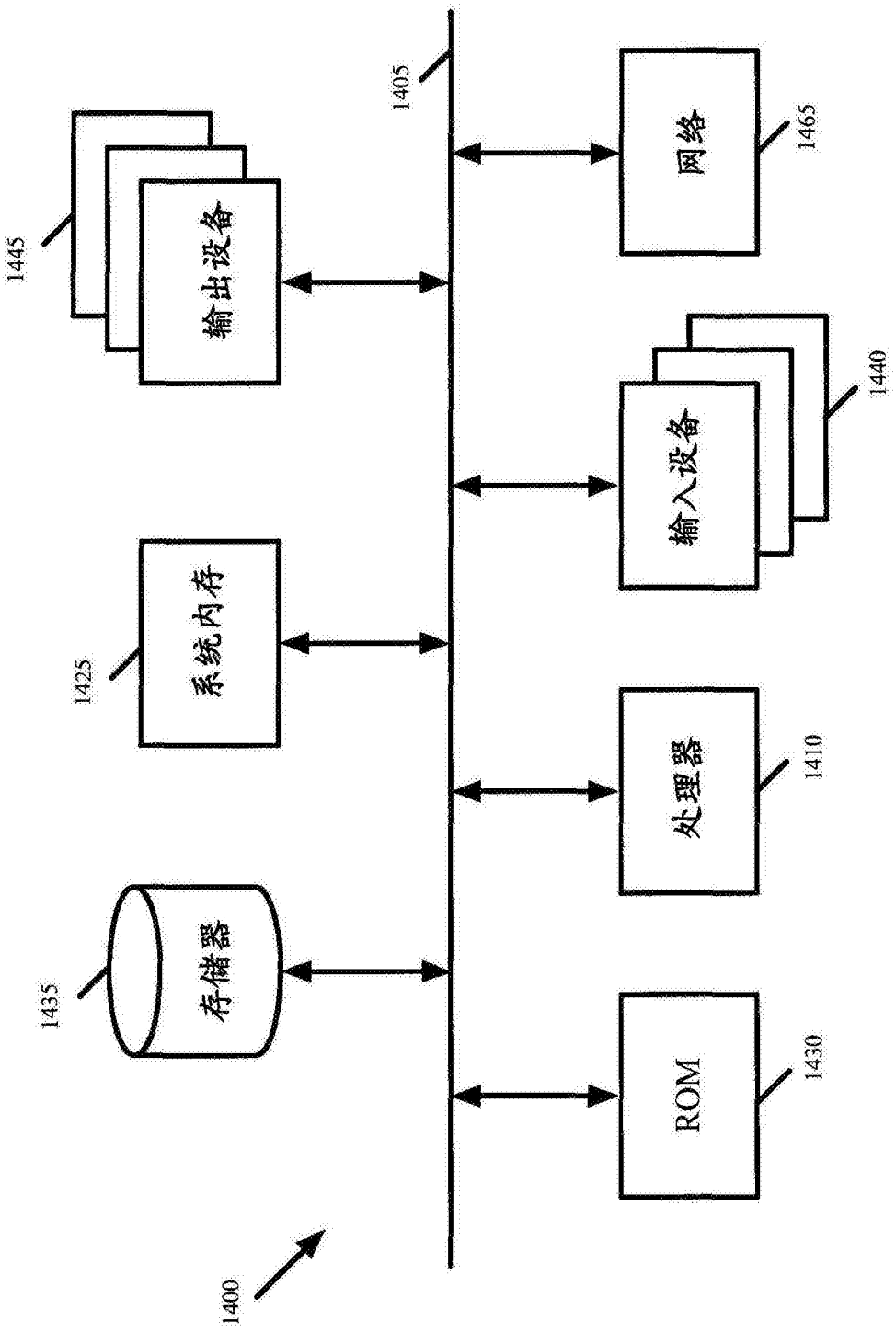

1405
网络
1465
1445
输出设备
输入设备
1440
系统内存
1425
处理器
1410
存储器
1435
ROM
1430
1400

图14