

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

H04L 29/06 (2006.01)

H04L 29/10 (2006.01)



[12] 发明专利申请公布说明书

[21] 申请号 200610068144.6

[43] 公开日 2007 年 1 月 10 日

[11] 公开号 CN 1893434A

[22] 申请日 2006.3.14

[21] 申请号 200610068144.6

[30] 优先权

[32] 2005.3.15 [33] US [31] 11/080,897

[71] 申请人 微软公司

地址 美国华盛顿州

[72] 发明人 A·古普塔 J·L·迪维

P·C·埃耶 R·T·拉奥

T·R·马尼恩

[74] 专利代理机构 上海专利商标事务所有限公司

代理人 张政权

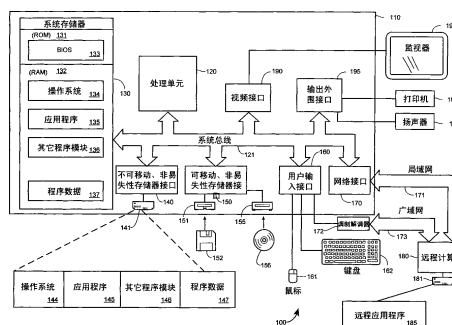
权利要求书 3 页 说明书 21 页 附图 17 页

[54] 发明名称

构建对等消息通信应用程序的 API

[57] 摘要

所要求保护的方法和系统是被设置成支持用于在面向服务的框架，尤其是面向服务的消息通信系统上实现的组播编程模型的 API。



1. 在启用对等联网的计算机中为实现一组设备服务的软件程序提供一种应用程序编程接口,所述计算机适用于作为通过具有对等联网协议的实现的对等联网协议来互操作的计算设备的网络的一部分,所述提供应用程序编程接口包括:

服务对要向对等网络上的其它计算设备通告的最大数量服务的请求;
服务确定消息是否应当被传播到对等网络上的其它计算设备的请求;
服务用于联系对等网络上的计算设备的端口号的请求;
服务对用于联系对等网络上的计算设备的传输方法的请求; 以及
服务关于要使用哪一解析器的请求。

2. 如权利要求 1 所述的应用程序编程接口,其特征在于,所述解析器使用 `pnrp` 将网孔 id 转换成网络地址。

3. 如权利要求 1 所述的应用程序编程接口,其特征在于,还包括允许设置要向对等网络上的其它计算设备通告的最大数量服务。

4. 如权利要求 1 所述的应用程序编程接口,其特征在于,还包括允许设置过滤器来确定消息是否应当被传播到对等网络上的其它计算设备。

5. 如权利要求 1 所述的应用程序编程接口,其特征在于,还包括允许设置用于联系对等网络上的计算设备的端口号。

6. 如权利要求 1 所述的应用程序编程接口,其特征在于,还包括允许设置用于联系对等网络上的计算设备的传输方法。

7. 如权利要求 1 所述的应用程序编程接口,其特征在于,还包括允许设置要使用的解析器。

8. 一种具有计算机可执行指令的计算机可读介质,所述计算机指令在启用对等联网的计算机中使用,所述计算机适用于作为通过具有对等联网协议的实现的对等联网协议来互操作的计算设备的网络的一部分,所述计算机指令为实现一组设备服务的软件程序提供一种应用程序编程接口,包括:

用于服务对要向对等网络上的其它计算设备通告的最大数量服务的请求的计算机可执行指令;

用于服务确定消息是否应当被传播到对等网络上的其它计算设备的请求的计算机可执行指令;

用于服务用于联系对等网络上的计算设备的端口号的请求的计算机可执行指令；

用于服务对用于联系对等网络上的计算设备的传输方法的请求的计算机可执行指令；以及

用于服务关于要使用哪一解析器的请求的计算机可执行指令。

9. 如权利要求 8 所述的计算机可读介质，其特征在于，所述解析器使用 `pnrp` 将网孔 `id` 转换成网络地址。

10. 如权利要求 8 所述的计算机可读介质，其特征在于，还包括用于允许设置要向对等网络上的其它计算设备通告的最大数量服务的计算机可执行指令。

11. 如权利要求 8 所述的计算机可读介质，其特征在于，还包括用于允许设置过滤器来确定消息是否应当被传播到对等网络上的其它计算设备的计算机可执行指令。

12. 如权利要求 8 所述的计算机可读介质，其特征在于，还包括用于允许设置用于联系对等网络上的计算设备的端口号的计算机可执行指令。

13. 如权利要求 8 所述的计算机可读介质，其特征在于，还包括用于允许设置用于联系对等网络上的计算设备的传输方法的计算机可执行指令。

14. 如权利要求 8 所述的计算机可读介质，其特征在于，还包括用于允许设置要使用的解析器的计算机可执行指令。

15. 一种计算装置，包括：

能够生成视频图像的显示单元；

输入设备；

操作上耦合到所述显示单元和所述输入设备的处理装置，所述处理装置包括处理器和操作上耦合到所述处理器的存储器；

连接到网络以及所述处理装置的网络接口；

所述处理装置被编程为，在启用对等联网的计算机中为实现一组设备服务的软件程序提供一种应用程序编程接口，所述计算机适用于作为通过具有对等联网协议的实现的对等联网协议来互操作的计算设备的网络的一部分，所述提供应用程序编程接口包括：

服务对要向对等网络上的其它计算设备通告的最大数量服务的请求；

服务确定消息是否应当被传播到对等网络上的其它计算设备的请求；

服务用于联系对等网络上的计算设备的端口号的请求；

服务对用于联系对等网络上的计算设备的传输方法的请求；以及
服务关于要使用哪一解析器的请求；

其中，所述解析器使用 `pnrp` 将网孔 `id` 转换成网络地址。

16. 如权利要求 15 所述的计算装置，其特征在于，还包括被编程为允许设置要向对等网络上的其它计算设备通告的最大数量服务的处理装置。

17. 如权利要求 15 所述的计算装置，其特征在于，还包括被编程为允许设置过滤器来确定消息是否应当被传播到对等网络上的其它计算设备的处理装置。

18. 如权利要求 15 所述的计算装置，其特征在于，还包括被编程为允许设置用于联系对等网络上的计算设备的端口号的处理装置。

19. 如权利要求 15 所述的计算装置，其特征在于，还包括被编程为允许设置用于联系对等网络上的计算设备的传输方法的处理装置。

20. 如权利要求 15 所述的计算装置，其特征在于，还包括被编程为允许设置要使用的解析器的处理装置。

构建对等消息通信应用程序的 API

背景技术

因特网上的组通信技术允许具有共同兴趣的用户协作、共享文件、彼此聊天、组播音频和视频用于演示和分组会议、以及参与多玩家游戏。实际上，在自组织(ad hoc)的基础上形成组的能力提供了允许具有共同兴趣的用户在与一般的因特网总体隔离的虚拟区域或组中聚集的重大优点，从而便于在诸如此类的有意向个人之间的协作中的有用讨论。然而，当前大多数组通信和形成在以服务器为中心的环境中发生，由此，所有通信都流向或流过个人可连接到其上来加入和参与组的大型中央服务器。

对等技术使得用户能够在无服务器的环境中彼此联系，从而不受基于服务器的因特网通信的约束。在基于对等的系统中，可以维护用户的自由性(anonymity)和私密性，因为通信直接在网络内的对等体之间发生。然而，尽管个人通信和文件共享在对等网络中是相对良好地建立的，在组对等环境中建立、发现、加入、维护和共享信息不是良好建立的。然而，个人越来越习惯于由以服务器为中心的环境中这样的分组技术所提供的好处。

发明内容

所要求保护的方法和系统是一种支持用于在面向服务的框架上，尤其在面向服务的消息通信系统实现的组播编程模型的 API 集。它提供了应用层组播能力，而无需要求一 IP 组播基础结构。所要求保护的系统使得面向服务的应用程序能够简易且有效地实现组播通信。此外，所要求保护的权利要求中的方法和系统提供了用于为可增强组播过程的对等应用程序创建和维护网孔拓扑结构的管理过程。这些过程中的某一些可包括：信道监视和传递服务；洪泛和过滤服务；通告服务；连接维护服务；跨领域因特网范围网孔能力；以及无服务器对等解决方案服务。

附图说明

图 1 是可依照本发明操作的计算系统的框图；

图 2 示出了开放系统互连模型；
图 3 示出了其中可构建所要求保护的系统的可能的面向服务的消息通信系统；
图 4 示出了使用图 3 的消息通信系统的消息通信；
图 5 示出了组播网孔拓扑结构；
图 6 示出了所要求保护的系统的逻辑网孔实施例；
图 7 示出了用于在本发明的实施例中实现网孔的通用对象模型；
图 8 示出了用于在本发明的实施例中创建网孔的通用流程图；
图 9 示出了本发明的实施例中的近邻连接过程；
图 10 可示出本发明的实施例中的同步器过程；
图 11 可示出一对近邻之间的各种同步相关消息的交换，其中发起者是新节点；
图 12 可示出网孔中一对现有节点之间的同步过程；
图 13 可示出既有节点和新节点之间的同步过程，其中既有节点是发起者；
图 14 可示出本发明的实施例中的通用洪泛过程；
图 15 示出了本发明的实施例中的消息处理过程；
图 16 示出了本发明的实施例中的网孔管理过程；以及
图 17 是依照本发明的方法的流程图。

具体实施方式

尽管以下文本陈述了众多不同实施例的详细描述，然而应当理解，该描述的合法范围是由本专利所附的权利要求书中的文字来定义的。该详细描述被解释为仅为示例性的，且不描述每一可能的实施例，因为描述每一可能的实施例即使不是不可能的也是不切实际的。可实现众多替换实施例，使用其它现有技术或在本专利的提交日期之后开发的技术，这些都落入权利要求书的范围之内。

也应当理解，除非在本专利中使用句子“如此处所使用的，术语‘_____’在此被定义为意味着……”或类似的句子来明确定义术语，否则没有意图要明确或隐含地限制该术语的意义超出其简单或普通意义之外，且这样的术语不应被解释为基于本专利任何章节中所作出的任何语句（除权利要求书中的语言之外）而在范围上有限制。在本专利所附的权利要求书中所述的任何术语在本专利中以与单个意义相一致的方式来引用意义上，这是为简明起见，仅仅为了不使读者感到迷惑，且这样的权利要求术语不旨在隐含地或另外限于该单个意义。最后，除非权利要求元素是通过叙述单词“装置”和功能而没有叙述任何结构来定义的，否则任何权利要求

元素的范围不旨在基于 35 U.S.C. §112 第 6 段的应用来解释。

图 1 示出了其中可实现所要求保护的方法和装置的块的系统的合适的计算系统环境 100 的示例。计算系统环境 100 仅是合适的计算环境的一个示例，且不旨在对本发明的方法和装置的使用范围或功能提出任何局限。也不应将计算环境 100 解释为对示例性操作环境 100 中所示的任何一个组件或组件的组合具有任何依赖性要求。

所要求保护的方法和装置的块可以使用众多其它通用或专用计算系统环境或配置来操作。适用于本发明的众所周知的计算系统、环境和/或配置包括但不限于，个人计算机、服务器计算机、手持式或膝上设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费者电子设备、网络 PC、小型机、大型机、包括任一上述系统或设备的分布式计算环境等等。

所要求保护的方法和装置的块可在诸如程序模块等由计算机执行的计算机可执行指令的一般上下文环境中描述。一般而言，程序模块包括例程、程序、对象、组件、数据结构等等，它们执行特定的任务或实现特定的抽象数据类型。该方法和装置也可以在分布式计算环境中实践，其中，任务由通过通信网络连接的远程处理设备来执行。在分布式计算环境中，程序模块可以位于包括存储器存储设备的本地和远程计算机存储介质中。

参考图 1，用于实现所要求保护的方法和装置的块的示例性系统包括计算机 110 形式的通用计算设备。计算机 110 的组件可包括，但不限于，处理单元 120、系统存储器 130 以及将包括系统存储器的各类系统组件耦合至处理单元 120 的系统总线 121。系统总线 121 可以是若干种总线结构类型的任一种，包括存储器总线或存储器控制器、外围总线以及使用各类总线体系结构的局部总线。作为示例而非局限，这类体系结构包括工业标准体系结构（ISA）总线、微通道体系结构（MCA）总线、增强 ISA（EISA）总线、视频电子技术标准协会（VESA）局部总线以及外围部件互连（PCI）总线，也称为 Mezzanine 总线。

计算机 110 通常包括各种计算机可读介质。计算机可读介质可以是可由计算机 110 访问的任一可用介质，包括易失性和非易失性介质、可移动和不可移动介质。作为示例而非局限，计算机可读介质包括计算机存储介质和通信介质。计算机存储介质包括以用于储存诸如计算机可读指令、数据结构、程序模块或其它数据等信息的任一方法或技术实现的易失性和非易失性，可移动和不可移动介质。计算机存储介质包括但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、

数字多功能盘（DVD）或其它光盘存储、磁盒、磁带、磁盘存储或其它磁存储设备、或可以用来储存所期望的信息并可由计算机 110 访问的任一其它介质。通信介质通常具体化为诸如载波或其它传输机制的已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据，并包括任一信息传送介质。术语“已调制数据信号”指以对信号中的信息进行编码的方式设置或改变其一个或多个特征的信号。作为示例而非局限，通信介质包括有线介质，如有线网络或直接连线连接，以及无线介质，如声学、RF、红外和其它无线介质。上述任一的组合也应当包括在计算机可读介质的范围之内。

系统存储器 130 包括以易失性和/或非易失性存储器形式的计算机存储介质，如只读存储器（ROM）131 和随机存取存储器（RAM）132。基本输入/输出系统 133（BIOS）包括如在启动时帮助在计算机 110 内的元件之间传输信息的基本例程，通常储存在 ROM 131 中。RAM 132 通常包含处理单元 120 立即可访问或者当前正在操作的数据和/或程序模块。作为示例而非局限，图 1 示出了操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137。

计算机 110 也可包括其它可移动/不可移动、易失性/非易失性计算机存储介质。仅作示例，图 1 示出了对不可移动、非易失性磁介质进行读写的硬盘驱动器 141、对可移动、非易失性磁盘 152 进行读写的磁盘驱动器 151 以及对可移动、非易失性光盘 156，如 CD ROM 或其它光介质进行读写的光盘驱动器 155。可以在示例性操作环境中使用的其它可移动/不可移动、易失性/非易失性计算机存储介质包括但不限于，磁带盒、闪存卡、数字多功能盘、数字录像带、固态 RAM、固态 ROM 等等。硬盘驱动器 141 通常通过不可移动存储器接口，如接口 140 连接到系统总线 121，磁盘驱动器 151 和光盘驱动器 155 通常通过可移动存储器接口，如接口 150 连接到系统总线 121。

上文讨论并在图 1 示出的驱动器及其关联的计算机存储介质为计算机 110 提供了计算机可读指令、数据结构、程序模块和其它数据的存储。例如，在图 1 中，示出硬盘驱动器 141 储存操作系统 144、应用程序 145、其它程序模块 146 和程序数据 147。注意，这些组件可以与操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137 相同，也可以与它们不同。这里对操作系统 144、应用程序 145、其它程序模块 146 和程序数据 147 给予不同的标号来说明至少它们是不同的副本。用户可以通过输入设备，如键盘 162 和定位设备 161（通常指鼠标、跟踪球或触摸板）向计算机 110 输入命令和信息。其它输入设备（未示出）可包括麦克风、操纵

杆、游戏垫、圆盘式卫星天线、扫描仪等等。这些和其它输入设备通常通过耦合至系统总线的用户输入接口 160 连接至处理单元 120，但是也可以通过其它接口和总线结构连接，如并行端口、游戏端口或通用串行总线（USB）。监视器 191 或其它类型的显示设备也通过接口，如视频接口 190 连接至系统总线 121。除监视器之外，计算机也可包括其它外围输出设备，如扬声器 197 和打印机 196，它们通过输出外围接口 195 连接。

计算机 110 可以使用到一个或多个远程计算机，如远程计算机 180 的逻辑连接在网络化环境中操作。远程计算机 180 可以是个人计算机、服务器、路由器、网络 PC、对等设备或其它常见的网络节点，并通常包括许多或所有相对于计算机 110 所描述的元件，尽管在图 1 中仅示出了存储器存储设备 181。图 1 描述的逻辑连接包括局域网（LAN）171 和广域网（WAN）173，这里示出作为示例而非局限。这类网络环境常见于办公室、企业范围计算机网络、内联网以及因特网。

当在 LAN 网络环境中使用时，计算机 110 通过网络接口或适配器 170 连接至 LAN 171。当在 WAN 网络环境中使用时，计算机 110 通常包括调制解调器 172 或用于通过 WAN 173，如因特网建立通信的其它装置。调制解调器 172 可以是内置或外置的，它通过用户输入接口 160 连接至系统总线 121。在网络化环境中，相对于计算机 110 所描述的模块或其部分可储存在远程存储器存储设备中。可以理解，示出的网络连接是示例性的，也可以使用在计算机之间建立通信链路的其它装置。

图 2 示出了由国际标准化组织（“ISO”）开发的开放系统互连（“OSI”）模型，它可用于描述本发明的方法和系统的各方面。OSI 是一种对于如何在电信网络中的两点之间传送消息的标准描述。

依照 OSI 模型，电信网络中两个端点之间的通信过程可被分为层，每一层添加一组其自己的特殊的、相关的功能。通信计算机可配备这七层功能。当在用户之间传递给定消息时，数据流通过一端的每一层，往下通过该计算机中的各层，且在另一端，当消息到达时，另一数据流往上通过接收计算机中的各层并最终到达最终用户或程序。提供这七层功能的实际的编程和硬件通常是计算机操作系统、应用程序、传输控制协议（“TCP”）/网际协议（“IP”）或替换的传输和网络协议、以及使得信号能够被施加在附加到计算机的线之一上的软件和硬件的组合。

OSI 将通信划分成七层。这些层分为两组。较高层 209 是在消息从用户传递或传递到用户时使用的，而较低层 208 是在任何消息通过主计算机时使用的。针对该

计算机的消息通过较高层 209。目的地为某一其它主机的消息不被向上传递到较高层，而是被转发到另一主机。该七层现在简要描述如下。

物理层 201 定义了接口的物理特征，诸如机械组件和连接器、诸如表示二进制值的电压电平等电气方面、以及诸如设置、维护和移去物理链路的功能方面。用于数据通信的公知的物理层接口包括串行接口、并行接口以及用于诸如以太网和令牌环等 LAN 的物理规范。

数据链路层 202 定义了用于跨物理连接在两个系统之间发送和接收信息的规则。数据链路通常是网络段（不是网间）和点对点链路。数据被打包成帧，用于跨底层物理网络传输。可使用某些可靠性功能，诸如接收数据的确认。

网络层 203 提供了跨多个网络传递数据的网间服务。网间寻址方案向每一网络和每一节点分配一唯一的地址。网络层支持多个数据链路连接。在因特网协议套件中，IP 是网络层网际协议。

传输层 204 提供了端对端通信服务，并确保数据在这些最终系统之间可靠地传递。两个最终系统建立连接，且参与对话以跟踪跨网络的分组传递。该协议也调节分组流以容纳较慢的接收器，并确保如果在链路中发生了中断，则传输不会完全被停止。传输控制协议（“TCP”）可以是一种传输层协议。

会话层 205 通过使用常规技术或对话协调系统之间的信息交换。对话不总是需要的，但是某些应用程序可能需要一种如果连接临时丢失则知道何处重新开始数据传输的方式，或可能需要周期对话以指示一个数据集的末端和一个新的数据集的始端。

表示层 206 包含作为用户在工作站上运行的操作系统和应用程序的一部分的协议。在该层中，信息被格式化以供显示或打印。数据内的代码被解释，且在该层中也处理数据加密和转换。

应用层 207 提供了用于访问底层网络服务的定义过程。应用层用于定义各种用户应用程序的范围。

尽管组播技术是为 OSI 模型的较低层 208，即传输/网络层开发的，然而所要求保护的系统集中于提供用于在 OSI 模型的较高层 209 上实现组播的应用程序级编程模型。由此，使用本发明的方法和系统，开发者可实现组播功能，而无需 IP 组播基础结构。此外，如下所述，所要求保护的系统可改进面向服务的编程模型。

面向服务的框架和编程模型

面向服务与面向对象的主要区别在于面向服务的框架如何定义“应用程序”。面向对象的开发可集中于从独立的类库构建的应用程序,而面向服务的开发可集中于从一组自治服务构建的系统。服务可以简单地是通过消息交换来与其交互的程序。然而,不像现有的分布式对象技术,服务可仅通过良好定义的 XML 接口来与其客户机交互。诸如跨服务边界传递完整类、方法和调用等行为可能不被允许。由此,应用程序开发者和系统继承者可能不需要所集成的软件的底层对象模型、类型系统和协议的具体知识。

应用程序可展示一组其它应用程序可利用的“服务”,而不是如分布式对象系统中那样通过直接对象激活来集成完全不同的应用程序。利用服务框架的应用程序可使用基于标准的协议和可动态获取的策略定义来展示其服务。构成连接系统的应用程序的这种去耦合可允许更简单的集成,在对系统进行自适应以随时间改变时更灵活,并且也可允许更可靠的操作。

现有的面向服务的编程模型可实现现有的较高 OSI 层消息通信服务,诸如简单对象访问协议(“SOAP”)。SOAP 可以作为为消息提供方案的较高层 OSI 服务来操作。利用 SOAP 协议的消息通信系统的一个可能的实施例在图 3 中示出。所示的消息体系结构(后文称为“消息总线”)可被视为用于简化 SOAP 节点的开发和维护的应用程序编程接口(“API”)。所要求保护的系统和方法可在提供基本的较高层消息处理的任何面向服务的模型上实现,但是为演示起见,所要求保护的方法和系统将使用图 3 中所示的消息总线来描述。

如图 3 所示,消息总线网络层 301 和传输 302 可由已存在的典型传输层/网络层协议的任一种构成。TCP/IP 是一种流行的传输层/网络层协议,且可以无需启用 IP 组播,这是使用本发明的应用层组播系统的一个优点。传输层可封装诸如 IP 套接字或命名等联网机制,或用于传递数据的任何机制。消息总线传输 303 可将消息转换成要通过数据链路层和物理层在线缆上传递的实际字节。

向上移至表示/会话层 307,端口对象 305 可表示用于消息的可配置的、逻辑管道。每一端口对象 305 具有该端口用于将逻辑消息转换成实际网络通信的传输集合 303。端口内部地被实现为一对管道对象:发送管道 306 和接收管道 307。每一管道可支持可插式消息处理程序(未示出)、组成端口扩展,它可检查和操纵沿该管道行进的消息。使用消息总线的应用中的典型节点可仅例示单个端口对象,尽管可能创建一个以上对象。本说明书的剩余部分将引用端口以协助用户概念化本说明书,但是对端口的使用在现代消息总线体系结构中可能是不必要的。

在应用层 309，应用程序节点 310 可通过信道对象 308 向端口对象发送消息。信道对象在各种实施例中存在，它可被缩小成两组：发送信道和接收信道。单个信道类也可实现发送和接收信道。

消息总线可支持各种信道实现，以允许各种消息交换模式。例如，用于单向数据包样式的消息的简单信道可以仅表示最终接收器。另一方面，更复杂的信道可支持相关的双向消息通信、可靠消息通信或跨应用程序执行的消息持续性。信道可以是其中由应用程序节点创建的消息对象可被传递给基础结构节点（例如，IP）的第一点，而从应用程序节点的观点来看，信道是消息总线体系结构中的可扩展性的第一点。

使用消息总线框架发送和接收消息的通用过程如下。

1. 由初始发送者创建消息对象。
2. 通过将对消息对象的引用传递到信道来发送消息。
3. 信道将消息传递到与该信道相关联的端口对象。
4. 端口将消息注入到端口的发送管道中，其中消息由向管道注册的每一消息处理程序顺序地处理。
5. 当消息到达发送管道的末端时，端口将消息引用传递到向该端口注册的传输对象。该传输对象是基于消息的下一中继段的地址来选择的。
6. 传输层将消息格式化字节流，并将字节推到网络上。

此时，消息离开了 SOAP 节点，且进入网络云状框。在消息到达其行程中的下一 SOAP 节点之前，消息不再服从消息总线或 SOAP。相反，消息可以通过诸如 TCP/IP 或 HTTP 等网络协议的字节流。

在某一点，消息字节到达网络上作为 SOAP 节点的监听节点。以下可在监听节点处发生。

1. 传输导出的对象接收字节流，从字节流中它在存储器中创建消息对象。消息头部被全部读入存储器中，然而，消息内容保留在传输中，且在需要之前可能保留在线缆上。
2. 传输然后将消息引用传递到传输对象向其注册的端口的接收管道。
3. 消息由向接收管道注册的每一消息处理程序顺序地处理。如果消息处理程序选择，它可将消息从接收管道中拉出，并将其传递到与该消息处理程序相关联的信道。
4. 如果消息使其到达端口上接收管道的末端，则端口将消息传递到向该端

口的主接收信道注册的应用程序定义的消息处理程序。

5. 应用程序代码具有对消息对象的引用，使用该引用，它可以完成它感到满意的任何事情。通常，检查消息的动作头部，且处理消息的内容。
6. 一旦应用程序代码完成了该消息，消息总线读取并丢弃消息内容中剩余的任何未读的字节，以清除传输，且将消息对象留给无用信息收集。
7. 如果消息在其从初始发送者到最终接收者的路线上通过 SOAP 中介，则该中介也可使用如上列出的接收步骤以及发送步骤来处理消息。中介是同时作为单个消息的发送者和接收者的节点。SOAP 路由器是中介的一个常见示例。

图 4 示出了消息从初始发送者 401 通过两个中介 403、404 到最终接收者 402 可采取的路径。该图示出了仅发送管道 405 可由初始发送者 401 使用，且仅接收管道 406 可由最终接收者使用。同时，对于中介 403、404。发送管道 407、408 和接收管道 409、410 都可用于处理消息。注意，初始发送者和最终接收者的角色可在响应消息的行程中交换。

尽管消息总线框架提供了用于实现使用面向服务的框架的服务的基本机制，但是没有规定要启用组播功能。所要求保护的方法和系统在这样一种面向服务的消息总线体系结构上构建，以向开发者提供在面向服务的模型上构建以向其应用程序添加组播功能的能力。

组播网孔系统

所要求保护的系统和方法可以实现被称为网孔的特定组播拓扑结构。图 5 示出了一种可能的网孔拓扑。一般而言，组播组的成员可被称为节点 501，而组可被认为是网孔 500 且由使用通信信道（例如，509）来彼此互连的一组节点 501-508 来表示。网孔中的每一节点 501-508 能够通过传播过程向网孔中的每一其它节点传递消息，在该传播过程中，由一个节点（例如，节点 501）发送的每一消息传递到相邻的节点 502-505，相邻节点继续将消息传递到其相邻节点，直到每一节点接收到该消息。当节点具有多个连接时，它可接收消息并制作该消息的副本以沿连接到它的每一信道发送。由于传播的方向一般从消息发送者流走，且类似于物理洪泛，因此使用组播的消息传播一般被称为洪泛。组播网孔中的每一节点可接收，且可知道在网络上洪泛的任何消息，即一个节点通过洪泛的消息知道什么，网孔中的每一节点也知道什么。因此，有益的是具有一种合理大小或关于何时应继续洪泛消息的

规则的网孔。

使用面向服务的消息通信系统的组播网孔环境

图 6 示出了所要求保护的组播网孔系统的通用实施例。在该实施例中，对等节点 601 是连接到网孔 602 的节点。网孔可以是一组命名的对等节点 601-605，它们由相邻信道 607-611 多重连接以形成节点的耦合网络，用于以消息或点对点数据流的形式来传播数据。网孔是基于到相邻节点的连接来构建的，其中网孔中的相邻节点是离开一个通信链路（通过通信信道直接连接）的对等节点。

对等节点可以与耦合到网孔 602 的应用程序 612 的实例相关联。对等信道 613、614 可被认为是对于对等节点的特定信道，用于在应用程序 612 或服务（以面向服务的语言）和网孔 602 之间通过网孔的对等节点 601 来通信。任何应用程序可创建节点，但是节点不可创建应用程序。应用程序可创建多个节点，且每一节点可主存多个服务。例如，考虑聊天应用程序。聊天应用程序可在称为 ChatMesh（聊天网孔）的网孔中创建节点。它可创建两个不同的服务，即 VideoService（视频服务）和 TextService（文本服务）。所发送的任何文本消息可被传递到 TextService，且所发送的任何视频消息可被传递到 VideoService。

当最初连接到网孔时，对等节点 601 可连接到已经连接到该网孔的另一对等节点，例如 603。连接对等节点 601 可确定连接到该网孔的相邻节点 603 的地址，并向相邻节点发送加入网孔的请求。在连接到网孔之后，新对等节点 601 可创建到网孔中的其它近邻，例如 604、605 的连接 608、610。

网孔的目的可以是创建用于将消息洪泛到所有网孔节点的逻辑拓扑结构。为自动实现用于洪泛的最优拓扑结构，网孔可维护签名、一组触点节点，并评估重复洪泛的消息以确定哪些近邻更有效。随着时间的推移，连接可以被创建和移除，使得网孔收敛于对洪泛最优的拓扑结构。

当节点从网孔断开时，它可能在网孔中创建称为网孔分区的间断。对等节点使用网孔签名（下文描述）以及网孔触点信息来检测和修复图分区。

对等信道 613、614 可以是在应用程序 612 和对等节点 601 之间发送和接收的通信信道。它们可向应用程序表现为组播信道，而无需启用 IP 的组播基础结构。每一对等信道 613、614 可与对等节点 601 相关联。对等节点 601 可以负责实现网孔算法（例如，优化、分区检测和修复）以及贯穿网孔洪泛消息。每一对等节点可提供其自己的服务，称为节点服务，它使用可靠消息总线信道与一组相邻节点服务

交互。该节点服务例如可在服务框架上实现，并使用 TCP 和 HTTP 传输上的可靠信道。

在对等信道上发送的消息的目的地可以是由网孔中的另一节点提供的特定服务。网孔中的每一节点可接收网孔上发送的每一消息的副本。当节点接收消息时，它可将其传递到实现该消息的预期服务的任何相关联的对等信道。为将消息传播到网孔中的每一节点，始发节点可将消息发送到其近邻的每一个。这些近邻可将消息发送到其近邻，依此类推。实际的洪泛算法描述如下。

应用程序可从一个相邻对等节点 601 通过隧道到另一个对等节点 602，例如，通过在定址到节点的近邻的另一外部消息中封装应用程序消息或内部消息。新外部消息可由每一近邻创建。以此方式，原始消息，包括其目的地服务地址，可在节点之间保持不变。所有节点可从所有其它节点到达。在这不为真的意义上（例如，节点在不可遍历的防火墙之后），网孔的弹力降低。

对象模型环境

图 7 示出了可实现所要求保护的系统的实施例的对象。所要求保护的组播系统的对象可由对等信道工厂（PeerChannelFactory）701、对等信道（PeerChannel）702、对等监听器工厂（PeerListenerFactory）703、对等监听器（PeerListener）704、对等节点（PeerNode）705、同步器（Synchronizer）706、维护器（Maintainer）707、连接器（Connector）708、洪泛器（Flooder）709、近邻管理器（NeighborManager）710 以及近邻（Neighbor）711 构成。逻辑上，上述逻辑对等节点可以被实现为由对等节点 705、同步器 706、连接器 708、维护器 707、洪泛器 709 和近邻管理器 710 构成的一组对象。

对等信道工厂 701 可负责产生对等信道 702。对等信道工厂 701 也可维护它所创建的活动信道的集合。对等信道工厂 701 可以不直接与对等节点 705 相关联，但是在需要时查找或创建它们。

对等监听器工厂 703 对象可负责产生对等监听器 704。对等监听器工厂 703 管理活动监听器，并维护监听器所产生的活动信道的集合。对等监听器工厂 703 可以仅与一个对等节点 705 相关联。

对等节点 705 可表示网孔中的一个节点，并包含其相关联的配置。对等节点 705 可以是实现该节点的所有内部组件的所有者。该类也可包含用于查找现有节点的静态方法。

近邻 711 可包含底层的相邻信道，并跟踪该信道的状态。近邻可展示用于与相邻节点通信的方法。

近邻管理器 710 可包含近邻 711 的集合。近邻管理器 710 可展示作为所有近邻 711 的聚集的事件（例如，**MessageAvailable**（消息可用））。该对象可提供对等节点服务的实现。

连接器 708 对象可处理发送和接收连接消息（**CONNECT**（连接）、**WELCOME**（欢迎）、**REFUSE**（拒绝）、**DISCONNECT**（断开））。

维护器 707 对象可实现网孔维护算法。维护器 707 可包含网孔签名、已知网孔节点的集合以及当前网孔触点的集合。

洪泛器 709 可实现洪泛逻辑。洪泛器可包含近来看见的消息的表。

同步器 706 可协调相邻信道上的轻量级触点/签名同步协议。

网孔构造

网孔可按名称或网孔 ID 来标识。网孔 ID 是可被解析成 IP 地址（例如，123.12.123.12）的有效主机名（例如，soccerpals）。网孔上的服务的地址可由统一资源标识符（“URI”）来指定，且使用对等信道方案前缀（**net.p2p://**）、网孔 ID、路径以及服务名称（例如，**net.p2p://soccerpals/path/service**）来指定。**EndpointAddress**（端点地址）结构可用于引用网孔中的节点，且可部分地由至少一个 URI 构成。

图 8 示出了使用本发明的对象模型实施例来创建网孔的过程。（以下描述将使用其枚举的名称，例如对等节点、对等信道来参考上述所要求保护的系统的对象）。

一般而言，在服务或客户机期望与网孔通信的任何时刻，可要求该服务或客户机通过网孔的对等节点来通信。具体地，服务可与对等监听器工厂相关联，而客户机可与对等信道工厂相关联。这一过程可涉及首先创建（801）和打开（802）对等信道工厂的应用程序。对等信道工厂可被例示为应用程序中的对象，且因此可与该应用程序相关联。接着，可通过调用对等信道工厂上的方法（称为 **CreateChannel**（创建信道））来创建（803）对等信道。对等信道然后可检查是否有相关联的对等节点（804）。如果有相关联的对等节点，则对等信道可连接到该对等节点，并打开对等信道（805），且开始接收和发送消息（806）。否则，如果没有现有的对等节点，对等信道可创建新的对等节点（807），打开对等信道（808）（例如，通过调用对等信道上的 **OPEN** 函数并传入 **EndpointAddress**）并打开新的对等节点

(809)。

在打开新对等节点时，对等节点可调用解析器(Resolver)对象，它负责提供名称解析服务。存在众多解析器方法，包括 RegisterMeshID(注册网孔 ID)、UnregisterMeshID(未注册网孔 ID)等。解析器方法可实现自定义名称解析服务，或可使用由消息总线提供的解析服务，诸如对等名称解析协议(“PNRP”)。新创建的对等节点可调用解析器来注册节点服务(810)，该节点服务是由每一对等节点为内部实现目的而唯一提供的服务。该节点服务可以与发起新对等节点的创建或发起到现有对等节点的连接的应用程序服务不同。通过向解析器注册，其它节点可通过其节点服务引用获得对新创建的对等节点的引用。

除注册其节点服务之外，对等节点可向解析器查询网孔中的其它节点(811)。如果不调用解析器的 ResolveMeshId(解析网孔 ID)方法，节点可能不知道在网孔中是否有其它节点(除非其它节点之一发起了到该节点的连接)。如果没有其它节点，则对等节点可向对等信道引发指示，该指示向应用程序引发警告。如果有其它节点，则对等节点可调用其维护器对象以将网孔 ID 解析成远程节点服务的 EndpointAddress。一旦获得了 EndpointAddress，近邻管理器可用于创建到相邻节点的相邻信道(811)。一旦创建了相邻信道，可通过发送 CONNECT 消息来尝试连接(812)。如果接收到 WELCOME 消息，则对等节点可被打开且在线，且同步过程开始。如果接收到 REFUSE 消息，则对等节点可尝试其它连接。如果对等节点未找到或连接到任何其它节点，则对等节点可将其自身设为离线状态(813)，否则对等节点可以在线(814)，并可接收和发送消息(815)。

一般而言，上述过程可以类似于对等监听器过程，不同之处在于初始信道创建框。当处理仅对在由目标为该服务的消息请求之后与网格通信感兴趣的服务时，该服务可创建并打开对等监听器工厂而非对等信道工厂。对等信道然后可用于创建对等监听器工厂。当由对等节点接收到匹配过滤器准则的第一消息时，对等监听器可产生对等信道。一旦创建了对等信道，可进行类似于上述过程的双向通信。然而，对等监听器也可用消息过滤器来匹配，使得它仅向应用程序通知匹配过滤器准则的消息。

当应用程序希望洪泛消息时，它可通过其对等信道与对等节点通信。只要消息对其可用，对等信道就通知应用程序。或者，如果实现使用对等监听器，对等监听器可生成对等信道，并通知应用程序已为其接收消息。一旦创建了对等信道，应用程序可接收并处理它。

每一对等节点可具有单个相关联的近邻管理器。近邻管理器可被限于传递和接收角色，并可被赋予维护所有其近邻，即直接连接到它的对等节点的列表，以及收集定向到它的消息和发送来自属于对等节点的对等信道的消息的任务。

近邻接收连接请求

图9示出了处理相邻连接的情形。当第一节点发起到第二节点的连接时(901)，可通知第二节点的近邻管理器，并可创建近邻对象以跟踪远程第一节点(902)。近邻管理器可通知其对等节点(903)，该对等节点表示第二相邻节点，并进而通知连接器。第一节点的连接器通过发送 CONNECT 消息启动与第二节点的连接协议。响应的连接器然后可依照以下向发起节点发送响应：

如果当前近邻数大于或等于最大值(在下文“网孔维护”一节中进一步描述)，则可向该近邻发回具有短近邻列表和/或其它已知节点的 REFUSE 消息(905)，并关闭信道(906)。否则，可发送具有短近邻列表和/或已知节点的 WELCOME 消息(907)。

如果信道不被上述过程关闭，则节点可交换包含每一节点上通告的服务(908)的 ADVERTISE (通告)消息。

同步节点

当近邻连接到网孔上的另一节点时，可以要求该网孔的当前状态同步以确保所有节点共享该网孔的同一视图。该状态包括网孔触点节点的列表以及网孔签名(用于如下描述的网孔维护功能)。

对等同步器(Peer Synchronizer)可以是对等节点的一个组件，它负责在一对相邻节点之间运行同步协议。该同步过程可涉及图10中示出的以下步骤。在框1000，当近邻转移到已连接状态时，发起同步器可通过发送 SyncRequest (同步请求)消息来启动同步过程。SyncRequest 消息可包含维护器的触点高速缓存中的触点抽象列表以及签名。每一触点抽象可包含触点节点的节点 ID 以及触点的版本号。对于新节点，该列表可以为空。

在框1005，在接收 SyncRequest 消息之后，响应的同步器可将抽象列表移交给维护器。维护器可实现网孔维护算法，并可包含网孔签名、已知网孔节点的集合以及当前网孔触点的集合。在框1010，维护器进而可将抽象列表与其触点高速缓存的内容进行比较。在框1015，维护器可构建存在于其高速缓存中但不存在于抽

象列表中的触点的列表（即，发起节点不具有这些触点），并添加比抽象列表中的触点更新的触点。在框 1020，创建经更新的抽象。不在响应者的高速缓存中的任何触点被添加，或者如果响应者具有较旧的版本，则维护器可将它们添加到新的、经更新的抽象列表中。在框 1025，可将触点列表、签名和新抽象列表返回给同步器。

在框 1030，如果由维护器返回的抽象列表非空，则控制可传递到框 1035，其中响应的同步器发送具有该抽象列表和签名的其自己的 SyncRequest 消息。在框 1040，同步器可发送具有触点的一系列 SyncData（同步数据）消息。最终的触点块和签名可以在 SyncEnd（同步结束）消息中发送。如果在框 1030，由维护器返回的抽象列表为空，则同步器可以不发送其自己的 SyncRequest，且控制可传递到框 1040。

在框 1043，在发起方，可确定发起方是否发送其自己的 SyncRequest。如果发起者未发送 SyncRequest，则当发送了所有的 SyncData 和 SyncEnd 消息之后，可在框 1045 发消息通知同步过程的结束。如果发起者发送了其自己的 SyncRequest，则同步过程可在框 1050 处接收到 SyncEnd 消息时结束。

在框 1055，在响应方，可确定响应方是否发送了其自己的 SyncRequest。如果响应者未发送 SyncRequest，则当发送了所有的 SyncData 和 SyncEnd 消息之后可在框 1060 发信号通知同步过程的结束。如果响应者发送了其自己的 SyncRequest，则同步过程可在框 1065 接收到 SyncEnd 消息时结束。

图 11 可示出一对近邻之间的各种同步相关消息的交换，其中发起者是新节点。图 12 可示出网孔中一对现有节点之间的同步过程。图 13 可示出现有节点和新节点之间的同步过程，其中现有节点是发起者。

发送/洪泛消息

图 14 示出了洪泛消息的情形。当应用程序希望发送消息时（1401），它可调用对等信道方法的 SEND（发送）方法（1402），对等信道可调用对等节点（1403），对等节点可调用洪泛器来发送消息（1404）、内部消息的 ID 可被添加到洪泛器的消息表（1405），且洪泛器可将内部消息封装到外部隧道消息中（1406），并将消息作为隧道消息发送到所有近邻（1407）。外部隧道消息可仅用于相邻信道上的通信，而不会改变内部应用程序消息。可以完成隧道以维持内部消息的完整性，以仅供内部对等节点组件处理和传递，同时提供封装头部和传递机制以供相邻信道处

理。这可有效地隔离两条路径之间的传输机制。

接收消息

图 15 示出了接收消息的情形。当在与对等节点相关联的相邻信道上接收消息时 (1501)，近邻管理器可通知对等节点 (1502)，对等节点可调用其洪泛器 (1503) 以依照其类型来处理消息。

对于所有洪泛的消息，内部消息可从外部隧道消息中提取 (1504)。对等节点可调用洪泛器来确定内部消息之前是否已被发送过 (1505)。如果是，则可关闭该消息，且处理完成 (1506)。如果内部消息未完成通过网孔传播 (即，消息应当被进一步洪泛)，则可发生进一步的处理。

如果内部消息是用于处理网孔功能的内部消息 (1507)，则对等节点可将该消息转发到其正确的内部节点对象 (例如，维护器) (1508)。这些组件如下所述处理消息。对于签名消息，维护器可确定该消息应当是否停止进一步洪泛 (1509)，此时消息被关闭且处理完成 (1506)。否则，内部消息的 ID 可被添加到洪泛器的消息表 (1516)，且内部消息可由洪泛器重新封装到另一外部隧道消息并被发送到除从其接收该原始消息的近邻之外的所有近邻 (1517)。

如果内部消息是应用程序消息 (1510)，则对等节点可搜索与该消息的目的地服务相关联的信道对象 (例如，对等信道或对等监听器) (1511)。如果找到匹配的对等信道，则消息可被发送到与对等信道相关联的应用程序/服务 (1512)。否则，如果找到匹配的对等监听器，则可通过调用 `AcceptChannel` (接受信道) 或 `BeginAcceptChannel` (开始接受信道) 来向应用程序/服务创建新对等信道 (1513)，且消息可被发送到对应的应用程序。

如果内部消息不是应用程序消息 (1510)，则内部消息的 ID 可被添加到洪泛器的消息表 (1516)，且内部消息可由洪泛器重新封装到另一外部隧道消息并发送到除从其接收原始消息的近邻之外的所有近邻 (1517)。附加的洪泛剪除逻辑也可在这一过程中实现 (1514、1515)。例如，某些应用程序可实现与洪泛器相关联的传播过滤器接口对象 (可以被称为 `IPeerMessagePropagationFilter` (对等消息传播过滤器) 接口) (1514)，以确定消息是否应被进一步洪泛。如果洪泛过滤器接口确定不应当有进一步的洪泛，则处理完成 (1515)。

关闭网孔的一部分

以下是可发生的事件，其中可拆散或关闭网孔的一部分。

近邻发起的断开连接：如果近邻干净地断开连接，则近邻可发送 DISCONNECT（断开连接）消息。该消息可由连接器处理，它关闭相邻信道。当相邻信道关闭时（突然或通过正式的断开连接过程），近邻管理器可通知其它内部节点组件。维护器可通过执行网孔维护算法（下文描述）来处理这一事件。对等节点可通过通知应用程序由远程节点通告的服务不再可用来处理这一事件。如果这是唯一的相邻连接，则该节点现在可以离线。

对等节点发起的断开连接：近邻可以被本地（例如，由对等节点）断开连接，这可以由于对等节点正在关闭或由于维护器选择剪除连接。当近邻正在关闭时，可通知本地组件。连接器可通过向近邻发送 DISCONNECT 消息来处理这一事件。一旦关闭了近邻，可通知应用程序，远程节点通告的服务不再可用。

关闭对等信道：当信道被关闭时，它可从拥有工厂的信道集合中移除，且释放对相关节点的对等节点的引用。

通告服务

所要求保护的网孔系统的一个优点是它可允许应用程序使用网孔来定位应用程序然后可消耗的服务。为允许访问由网孔上的各节点提供的服务，应用程序可能需要首先检索关于其近邻的信息。具体地，应用程序可能需要能够枚举在其近邻上通告的服务。

为向相邻节点通告本地服务，应用程序可调用对等节点的 `AddAdvertisedAddress`（添加通告的地址）方法，指定本地应用程序服务的 `EndpointAddress`。这可导致维护器将该服务添加到其当前通告的列表中，并向近邻发送包含经更新的通告服务列表的 `ADVERTISE`（通告）消息。类似地，应用程序可通过调用对等节点的 `RemoveAdvertisedAddress`（移除广告的地址）函数来取消通告服务。

当应用程序通告其服务地址时，该信息由节点高速缓存并被发送到近邻。在接收到该信息时，可引发事件。该事件可向应用程序提供近邻的节点 ID 以及该节点上通告的所有 `EndpointAddress`。

通过通告应用程序服务的地址，应用程序可打开直接到近邻的应用程序服务的信道，由此启用了专用通信信道。为向近邻发送消息（而非向整个网孔发送消息），可要求应用程序检索相邻节点上通告的服务的列表并决定要向哪一服务发送。为找

出这一服务列表，应用程序可调用对等节点的函数以检索通告的服务，该函数可被称为 `GetNeighborAdvertisedAddtesses`（获取近邻通告的地址），它收集高速缓存的服务信息。该方法向相邻节点上通告的服务返回 `EndpointAddress` 的集合以及近邻的节点 ID。

应用程序可选择这些 `EndpointAddress` 之一，并可使用底层消息总线的信道（例如，`http`、`tcp` 等——不是对等通道）来创建到服务的信道。应用程序可使用该信道来直接与近邻的服务通信（离开网孔）。

网孔维护

图 16 示出了本发明的实施例中的网孔维护过程。网孔连接可由三个参数确定，这三个参数被称为 `cIdealNeighbors`（理想近邻）、`cMaxNeighbors`（最大近邻）、`cMinNeighbors`（最小近邻）。节点可维护连接的相邻节点数的计数（1601）。节点可试图维护任一给定时刻到其它节点的近似 `cIdealNeighbors` 连接。节点可以不建立新的外出连接，或者如果当前它具有 `cMaxNeighbors` 可接受新的进入连接（1602），且可向试图连接到它的任何节点发送 `REFUSE` 消息（1603）。如果任何时刻连接数低于 `cMinNeighbors`（1604），则节点可试图建立更多连接。节点可欢迎（`WELCOME`）待决的信道请求（1605）（1606），或者节点可使用由其近邻传递到它（通过 `WELCOME`、`REFUSE` 和 `DISCONNECT` 消息）的已知节点列表（1607）。如果该列表是不足够的，则它可试图将网孔 ID 解析到新节点（例如，通过如上所述打开新信道）。

`cIdealNeighbors`、`cMaxneighbors` 和 `cMinNeighbors` 的值可取决于各种因素，包括网孔大小和网孔通信量。由于为 1 的 `cMinNeighbors` 值可在一个且仅一个连接丢失的情况下导致完全隔离，因此对任何网孔，至少为 2 的 `cMinNeighbors` 值是适当的。`cMaxNeighbors` 的高值可意味着特定的节点正在代表其它节点进行非常不成比例数量的洪泛。在 `cMinNeighbors` 和 `cMaxNeighbors` 之间的 `cIdealNeighbors` 值可以基于良好的冗余度，具有最小的额外开销。

网孔中的每一节点可试图尽可能维护一组最有用的连接，其中连接的有用性可由未看到的其上发送（在任一方向上）的消息的数量来确定。周期性地，当节点具有比 `cIdealNeighbors` 多时（1608），它可断开最不有用的连接（1609）。

每一节点可具有节点 ID，它是随机的 64 位数字。网孔具有签名，签名可被计算为网孔中存在的最低节点 ID。计算可通过令具有最低节点 ID 的节点将其 ID 公

布为签名来完成。然而，节点可能不直接知道它具有最低 ID，因此所有节点可试图公布其 ID。如果网孔签名已低于其 ID，则节点可以不公布。同样，每一节点具有公布之前的补偿周期，它可以与其节点 ID 成指数比例，因此帮助减少不正确的签名公布数。

网孔可具有一组触点节点，触点节点可负责周期性地发布它们所见到的网孔的签名。只要节点的网孔签名视图变得与触点公布的视图不同步（1610），节点就知道网孔已被分区，且可在随机补偿周期之后试图重新连接到不同步的触点之一（1611）。存在该过程可能未检测到分区的少量机会。为缓和这一机会，触点节点可周期性地试图将网孔 ID 解析成新的（随机）节点并形成连接。

API

图 17 是依照本发明的方法的图示。方法 1700 可用于启用对等联网的计算机，该计算机适用于作为通过具有对等网络协议的实现的对等网络协议来互操作的计算设备的网络的一部分。

对等协议的成功可取决于协议在选中实体之间建立有效连接的能力。同样，这一对等网络中组的形成依赖于该能力。然而，本领域的技术人员可以从以下教导中认识到，本发明的方法不限于特定的对等协议。

方法 1700 可在用于实现一组设备服务的软件程序的应用程序编程接口中实施。该方法可以是如先前所解释的设立对等监听器的对等监听工厂的一部分，以及设立对等信道的对等信道工厂的一部分。在框 1710，该方法可服务对最大数量服务的请求，以向对等网络上的其它计算设备通告。默认值可以是例如 25。服务可被认为是计算机程序或用户的计算机体验。组成员可以无需了解可从组中的每一计算设备的获得的所有服务，因为过多的通告会不必要地堵塞网络。要通告的服务的成员可以是任意的，或者可以基于公式，诸如在具有有限网络通信量的较小组中，可通告较多服务，而在具有较多网络通信量的较大网络中，可通告较少服务。另外，可被通告的服务的数目可被设置。数目可由网络的另一成员或网络的管理成员设置。一个示例计算机代码实现如下：

```
public int MaxAdvertisedAddressPerNeighbor {get; set}
```

在框 1720，可服务确定消息是否应当被传播到对等网络上的其它计算设备的请求。确定也可以被认为是过滤器，消息应当通过该过滤器到达网络上的其它计算设备。过滤器可具有无限数目的设置，诸如消息先前是否接收过，消息是否具有接

收计算机识别的预期接收者，网络通信量是否高，消息是最近的还是旧的，等等。默认值可以是无过滤器，这意味着所有消息都被转发。过滤器也可被设置。例如，如果对接收者已知的计算设备离线，且对该已知计算机设备接收到消息，则过滤器可被设置成不转发消息。一个示例计算机代码实现如下：

```
public IPeerMessagePropagationFilter MessagePropagationFilter {get; set;}
```

在框 1730，可服务对用于联系对等网络上的计算设备的端口号的请求。端口可用于将消息传递到网络，且网络上的不同计算设备可使用不同的端口。框 1740 可提供计算设备正用于向网络上的其它成员发送和/或接收消息的端口。默认可以是底层传输（即，TCP、HTTP）选择端口。另外，用于与网络通信的计算设备的端口可被设置。端口可以由网络的另一成员或网络的管理成员来设置。一个示例计算机代码实现如下：

```
public Nullable<int> NeighborServicePort {get; set;}
```

在框 1740，可服务对用于联系对等网络上的计算设备的传输方法的请求。可获得各种传输方法，诸如 HTTP 或 TCP。另外，计算设备的传输方法可被设置。传输方法可由网络的另一成员或网络的管理成员来设置。一个示例计算机代码实现如下：

```
public Transport NeighborServiceTransport {get; set;}
```

在框 1750，可服务关于要使用哪一解析器的请求。由于特定用户可以用各种方式在具有不同地址的各种位置连接到网络，因此可向用户或组分配唯一的身份。该地址然后需要被解析，使得身份通过协议被绑定到一个或多个特定的地址。一个这样的对等名称解析协议是对等名称解析协议（“RNRP”）。例如，解析器可使用 PNRP 将网络地址解析成网孔 id。然而，本领域的技术人员可以认识到，本发明的方法不限于特定的对等协议，而是可以应用于具有同等力度的其它协议。另外，由计算设备使用的解析器可被设置。解析器可由网络的另一成员或网络的管理成员来设置。一个示例计算机代码实现如下：

```
public IPeerResolver Resovler {get; set;}
```

从以上描述中，本发明的方法和系统使得使用面向服务的应用程序框架的应用程序开发者能够使用对等节点对象来简易且方便地集成组播功能。使用该框架，可创建具有组播功能的服务，而无需关注 TCP/IP 或其它底层的低级通信协议，因为对等节点对象使用任何可用的协议来管理消息通信。本系统的一个优点是可创建跨越整个因特网的网孔，而组播 IP 要求某些启用组播 IP 的因特网段（例如，

“Mbone”）来操作。所要求保护的系统也提供了基于应用程序的过滤机制，其中消息流可以使用过滤器接口来智能地控制（例如，消息抑制在使得网孔更有效和安全时是有用的），而 IP 组播解决方案未提供这样的接口。

此外，由于所要求保护的是构建在面向服务的框架之上的，因此应用程序开发者无需使其自身涉及从头开始对组播解决方案编程，它们只需关注它们需要什么组播功能来实现作为其服务的一部分。对等节点对象模型提供了一种基本的基础结构，使得开发者可集中于开发启用组播的服务。所描述的 API 也使得编程更为简易。

尽管以上文本陈述了众多不同实施例的详细描述，然而应当理解，本专利的范围是由本专利所附权利要求书中所陈述的文字来定义的。详细描述应被解释为仅为示例性的，且不描述每一可能的实施例，因为描述每一可能的实施例即使不是不可能的，也是不切实际的。可使用现有技术或在本专利提交日之后开发的技术来实现众多替换实施例，而这些都落入所附权利要求书的范围之内。

由此，可以在此处所描述和示出的技术和结构上作出各种修改和变化，而不脱离所附权利要求书的精神和范围。因此，应当理解，此处所描述的方法和装置仅是说明性的，而非对所附权利要求书有限制。

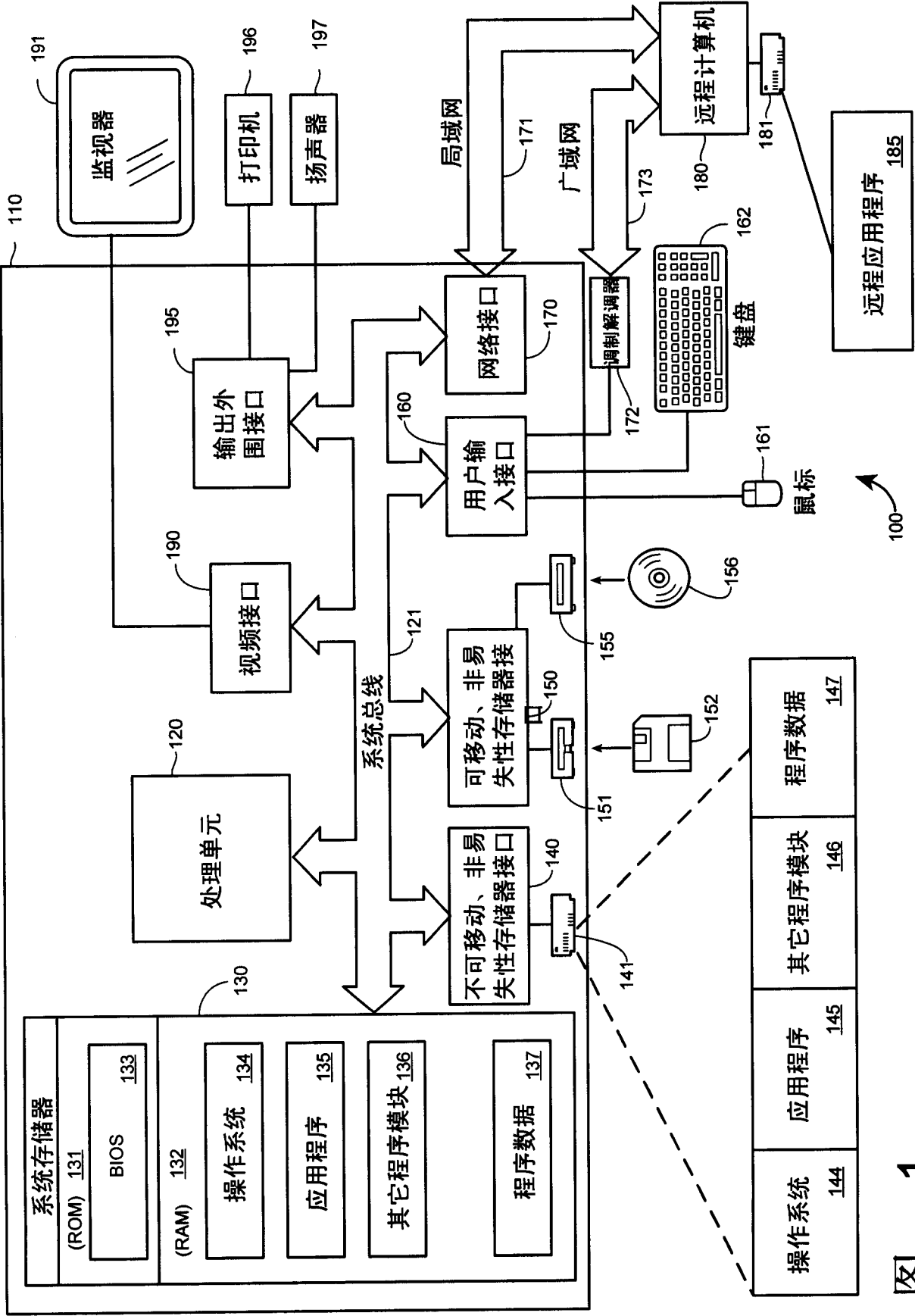
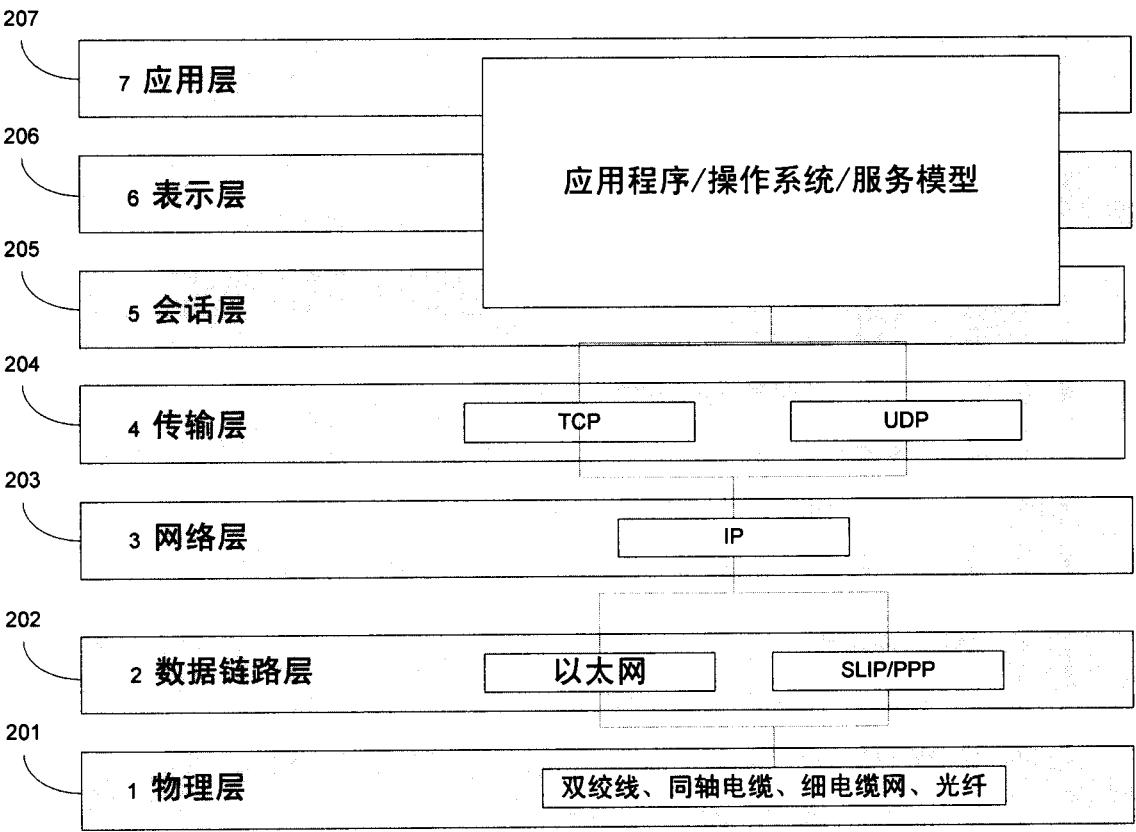
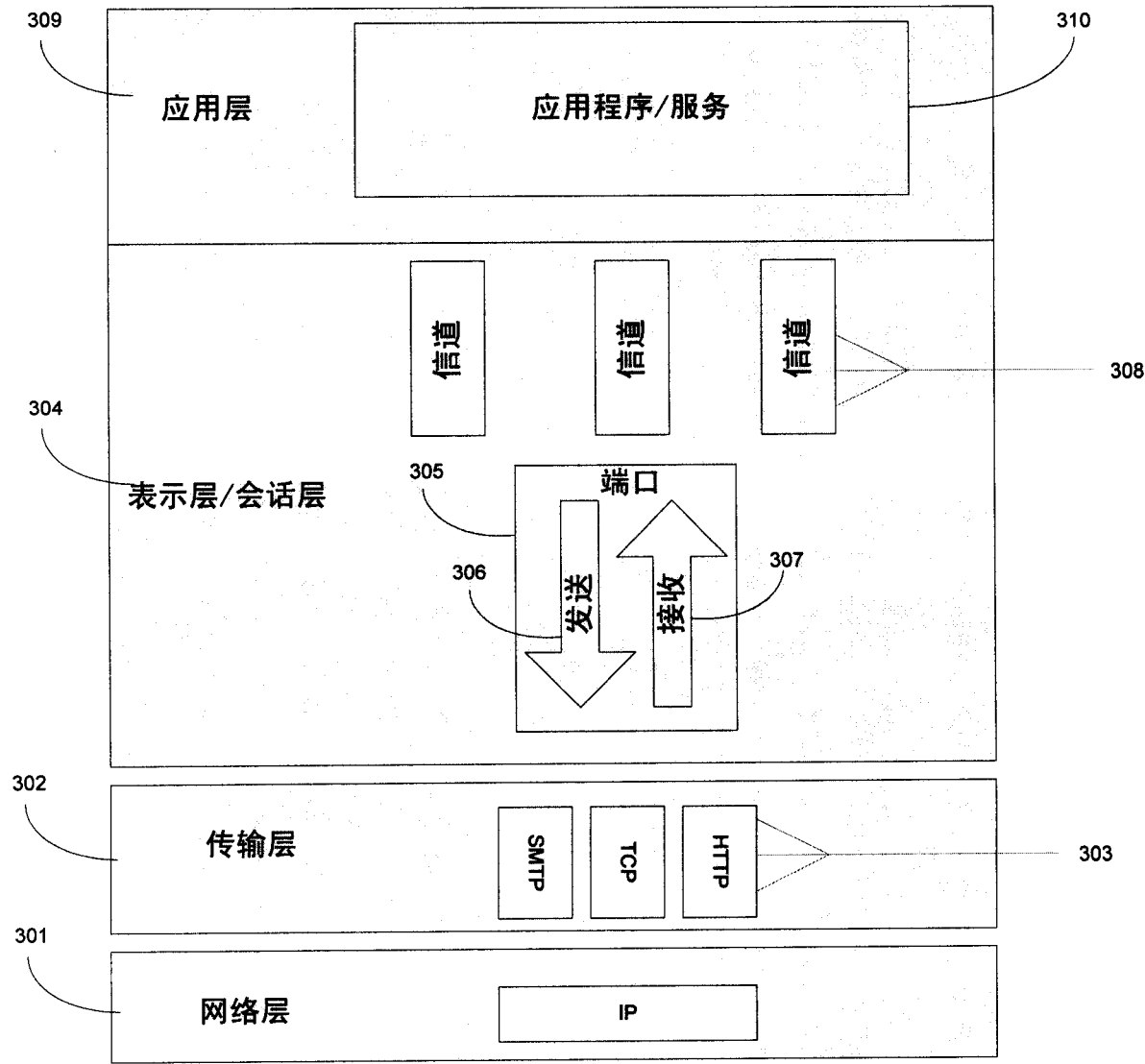


图 1



参考模型描述了七层相关功能，当消息在网络中从一方发送到另一方时在每一端上需要这些功能

图 2



面向服务的消息通信系统的示例

图 3

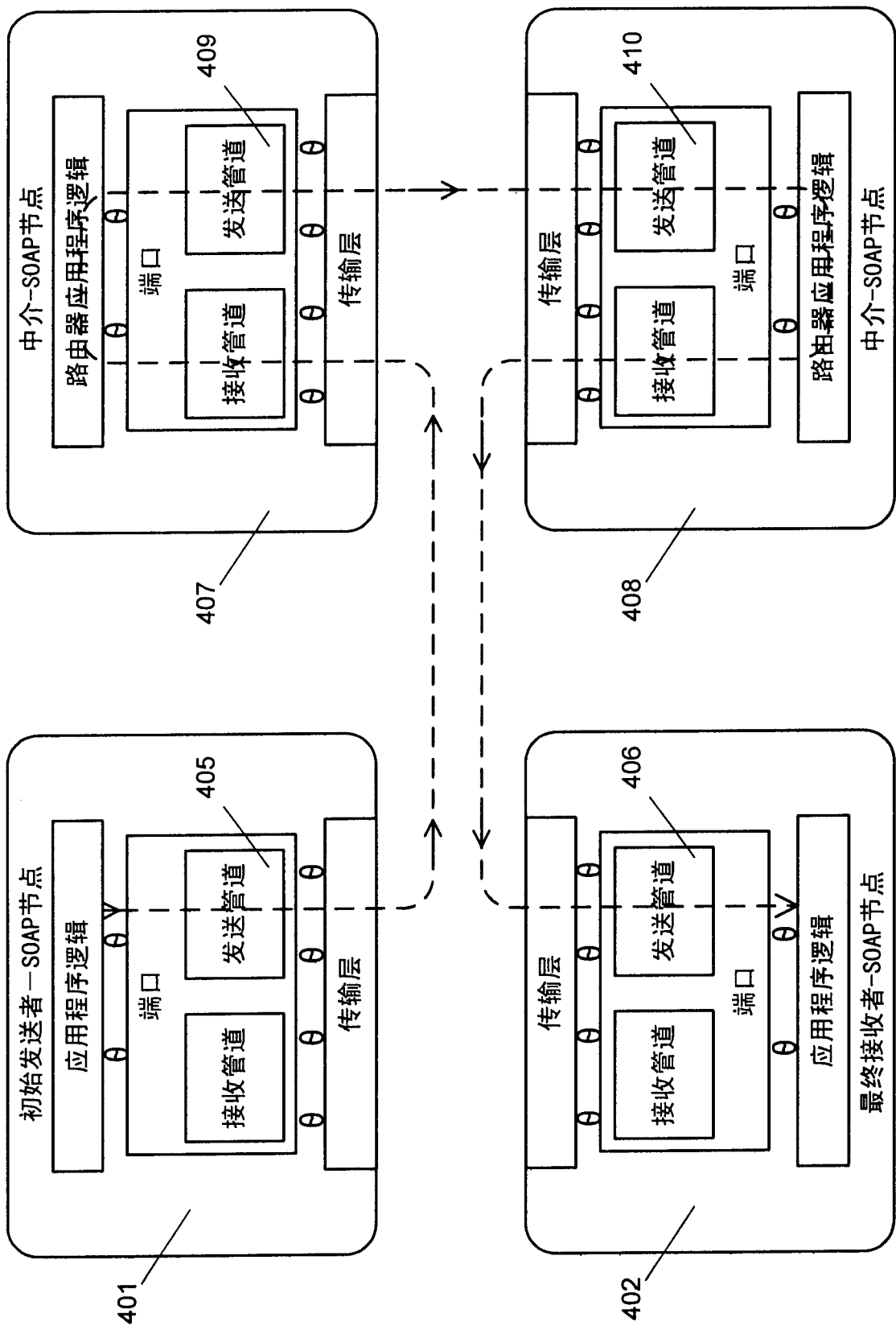


图 4

节点通过发起洪
泛向节点发送消
息，其中节点向
其每一相邻节点
发送消息。相邻
节点进而将消息
发送到其每一相
邻节点

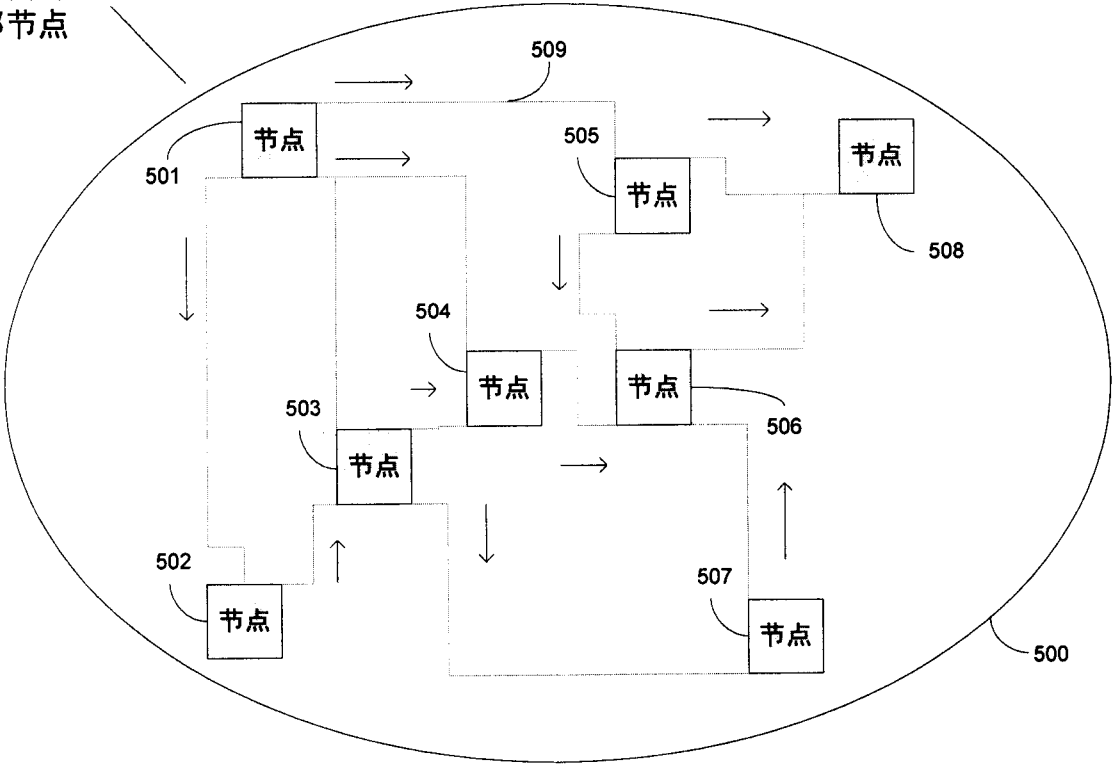
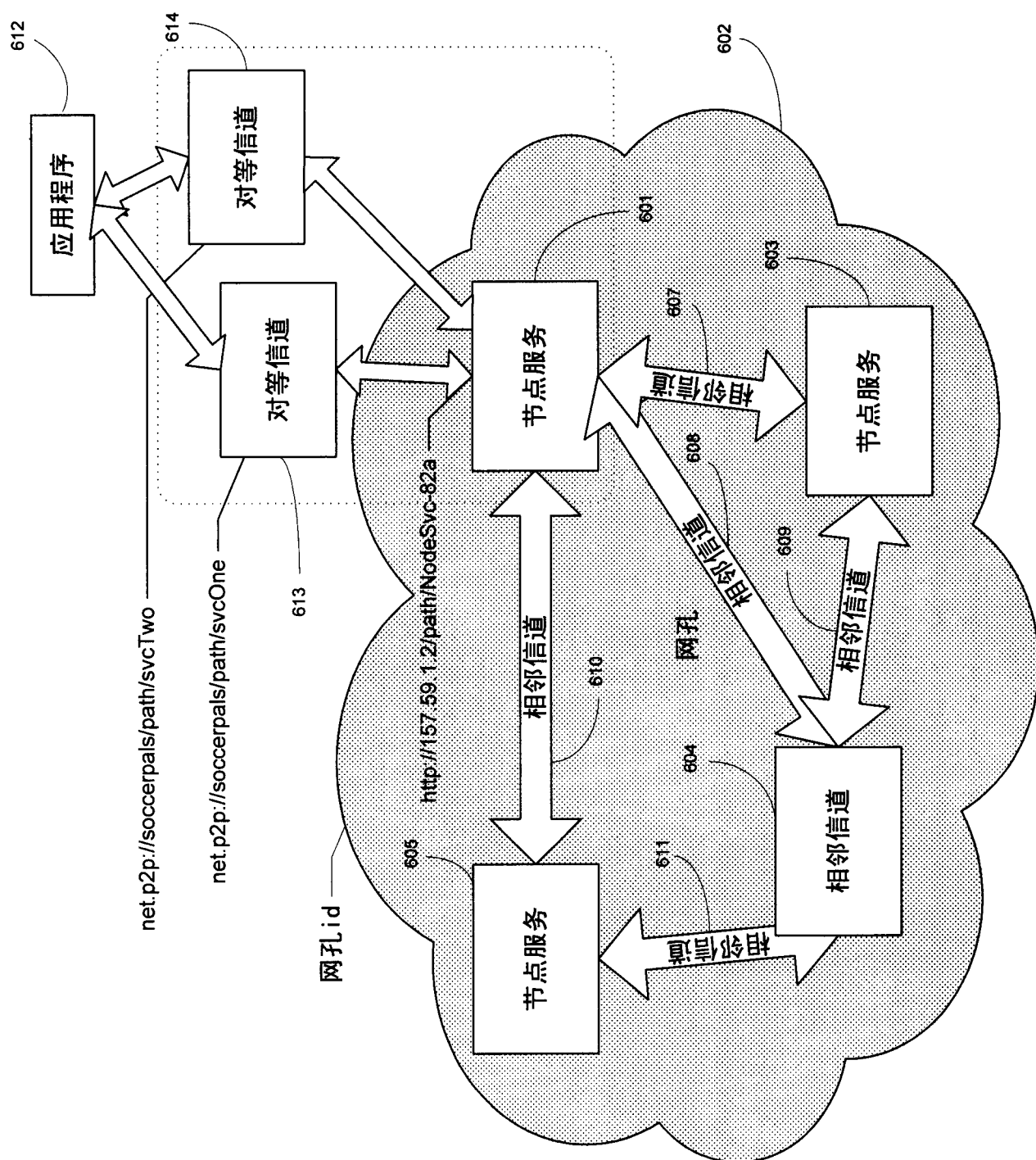


图 5



6

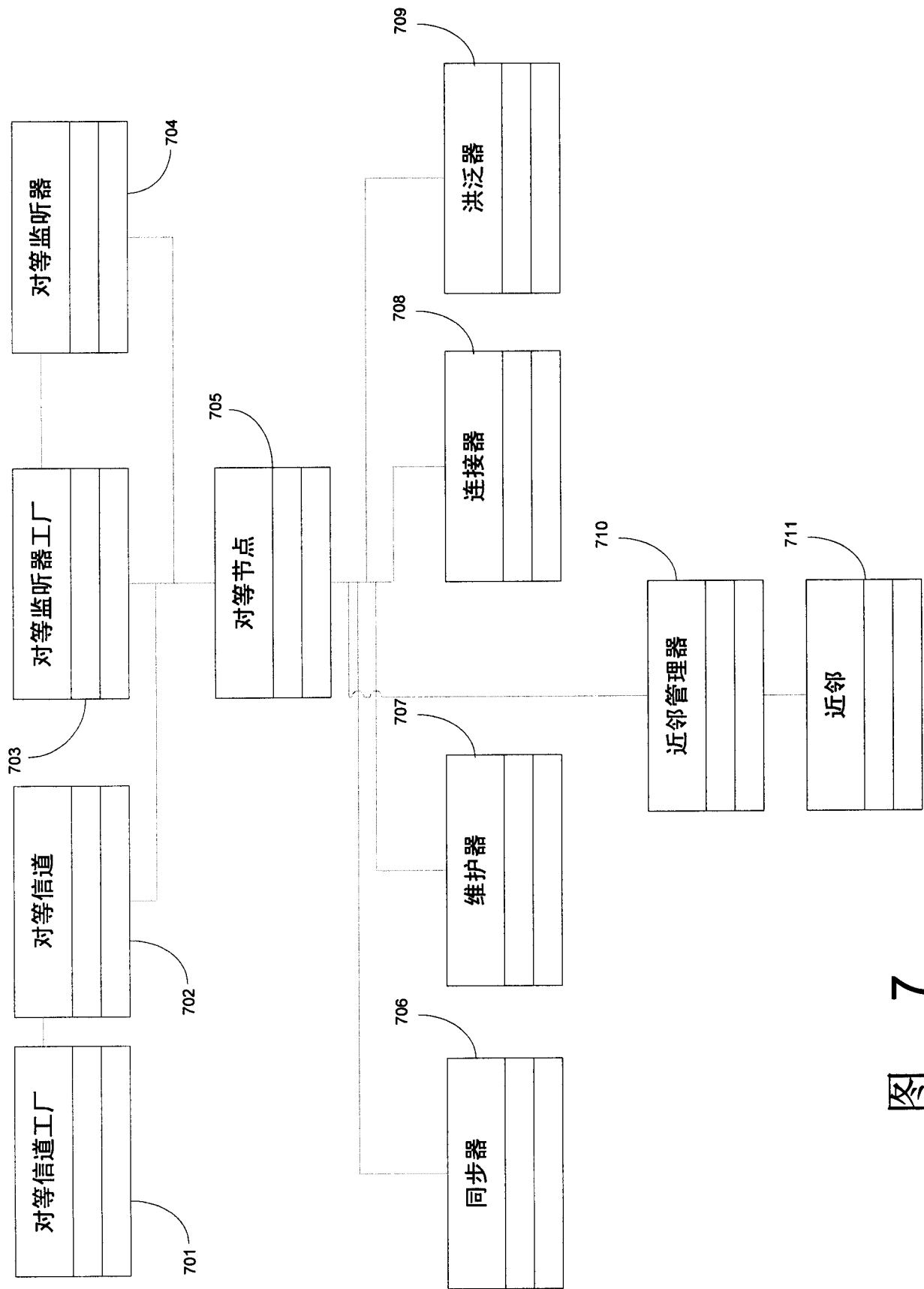


图 7

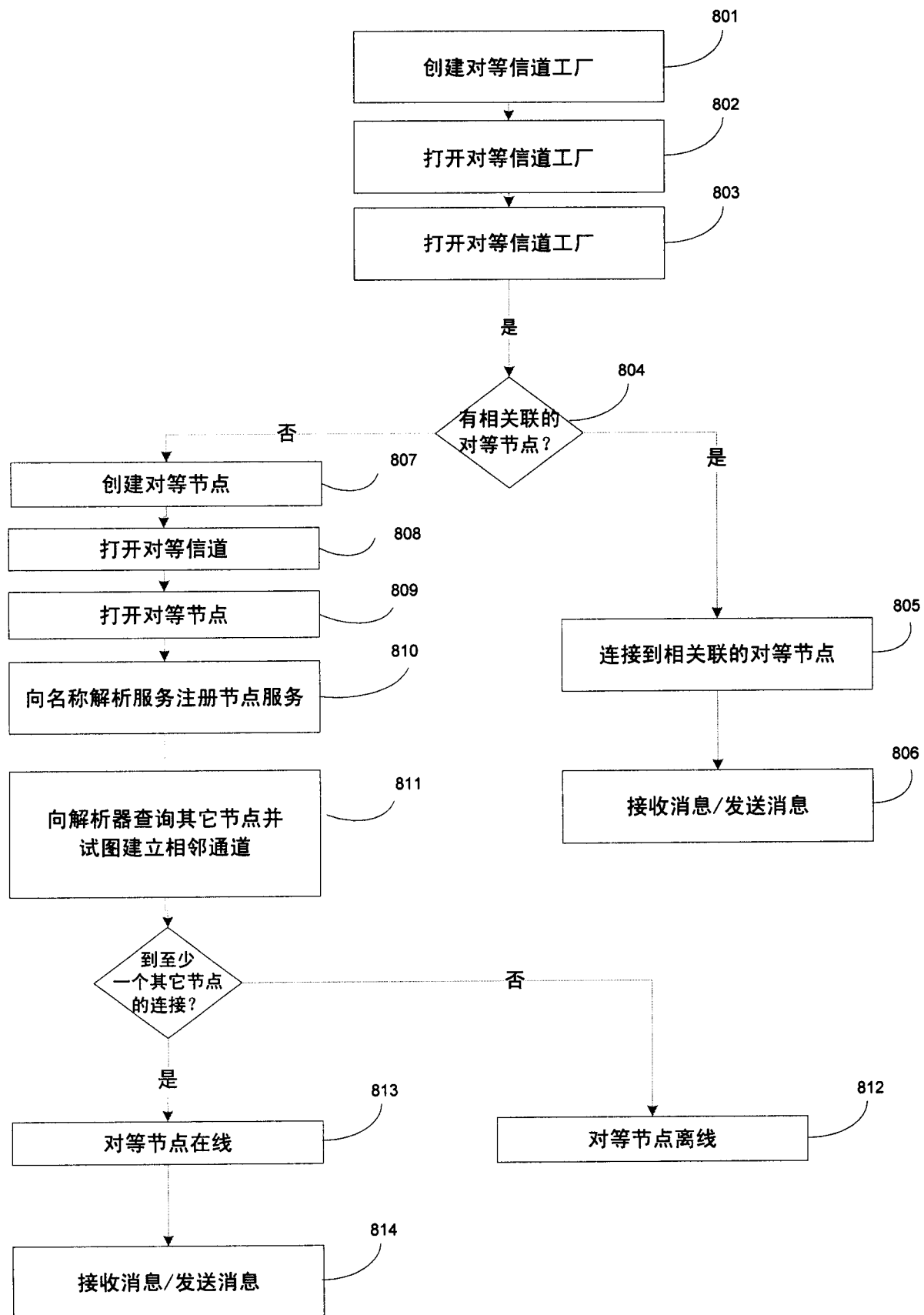


图 8

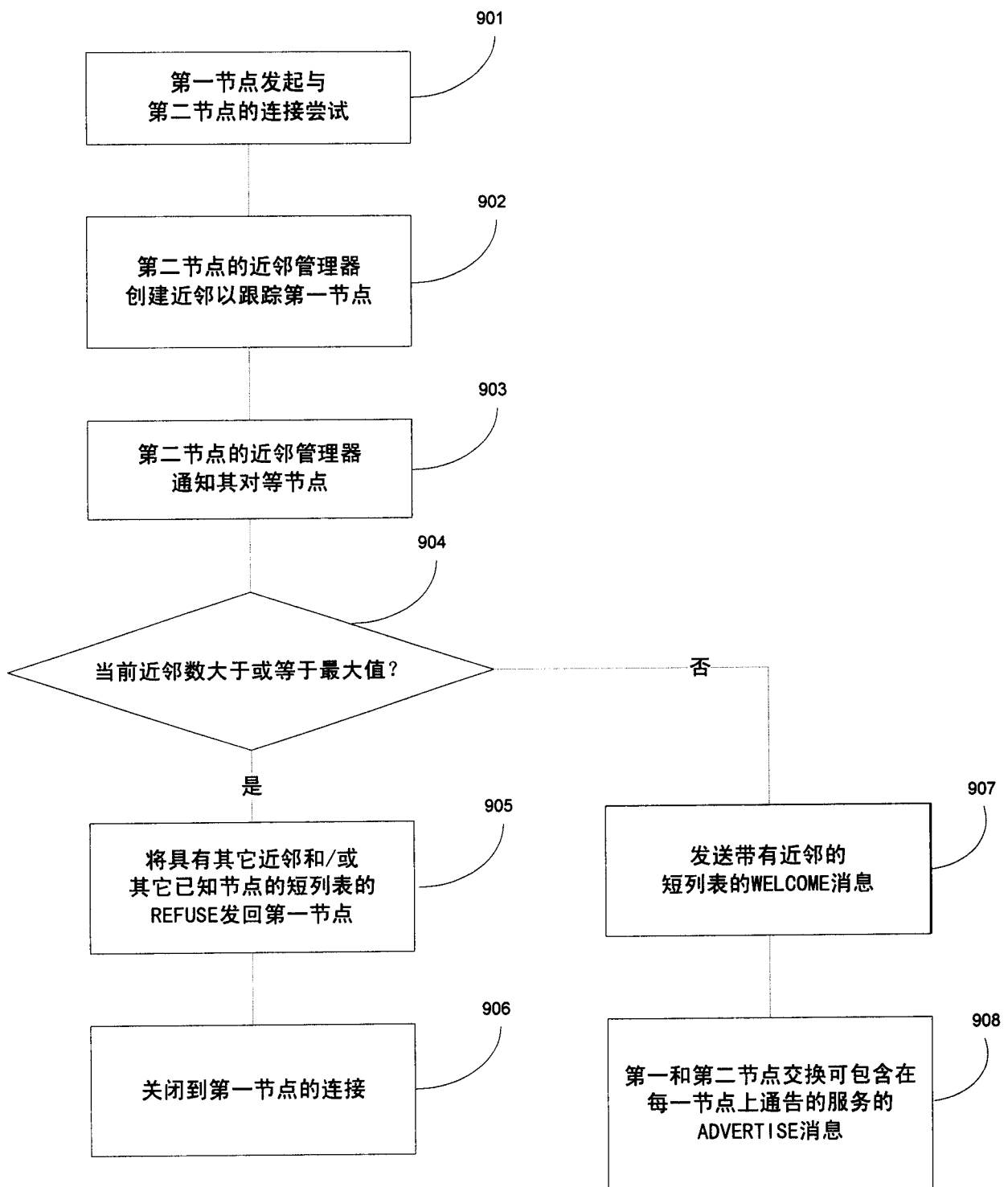


图 9

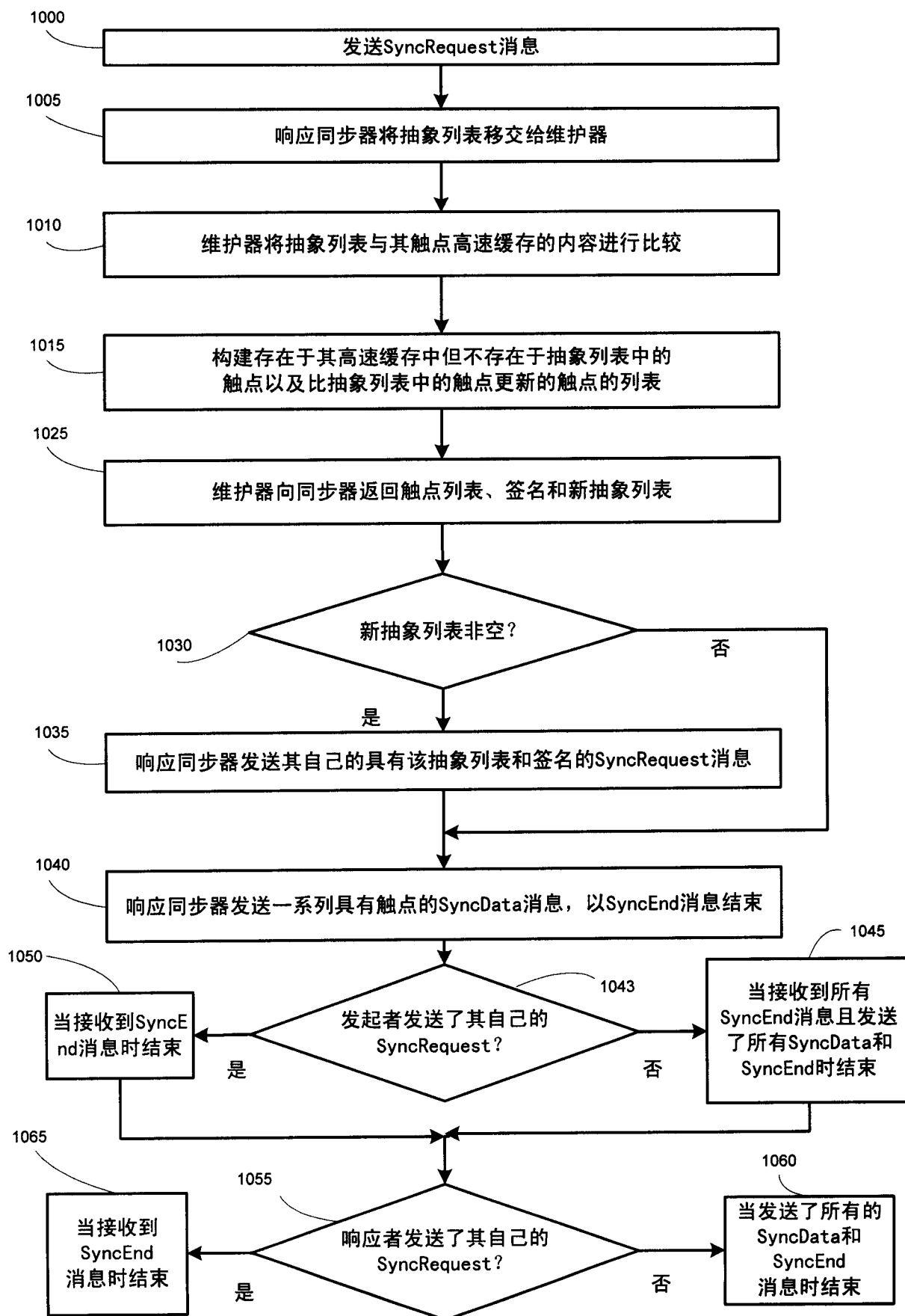


图 10

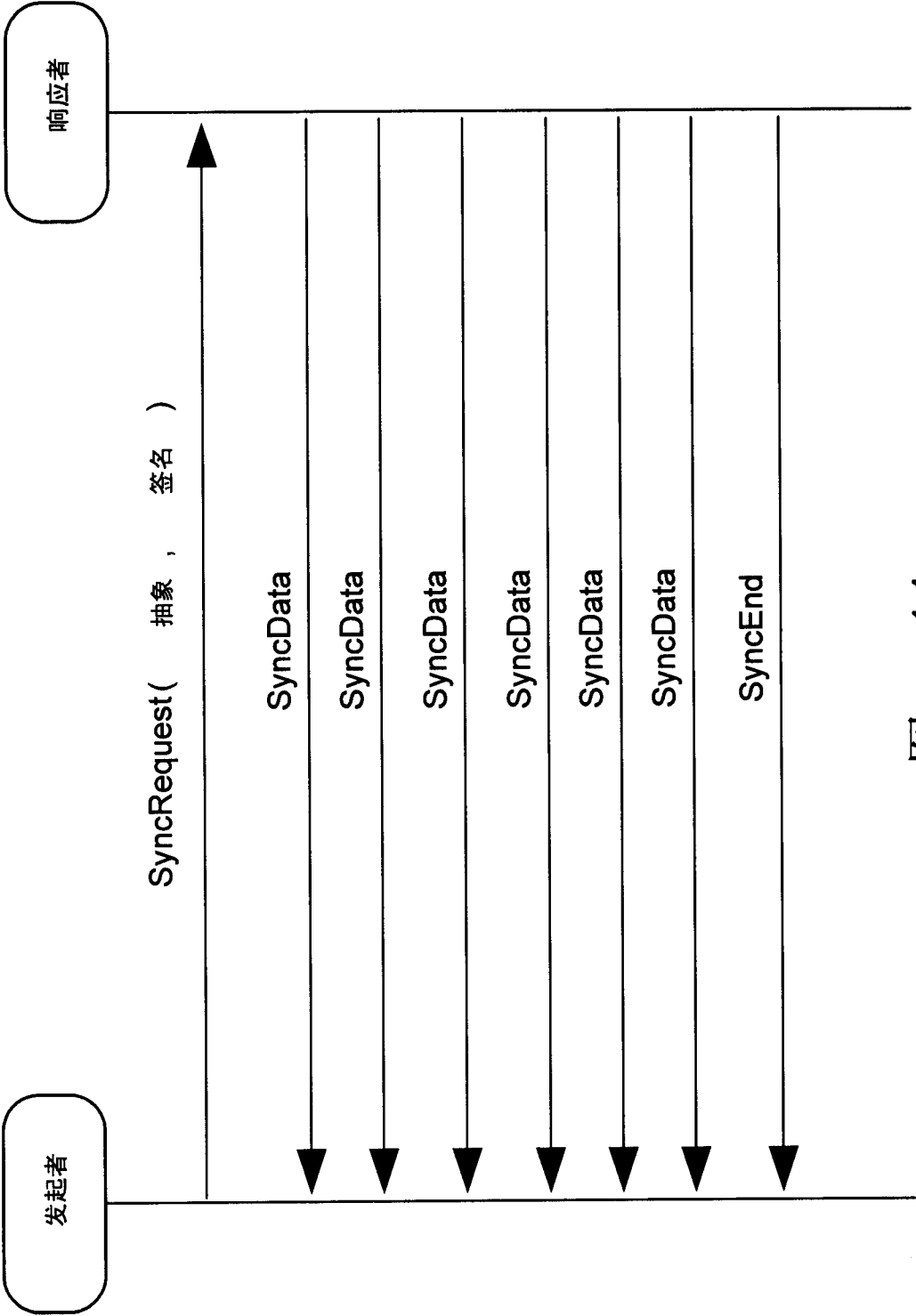


图 11

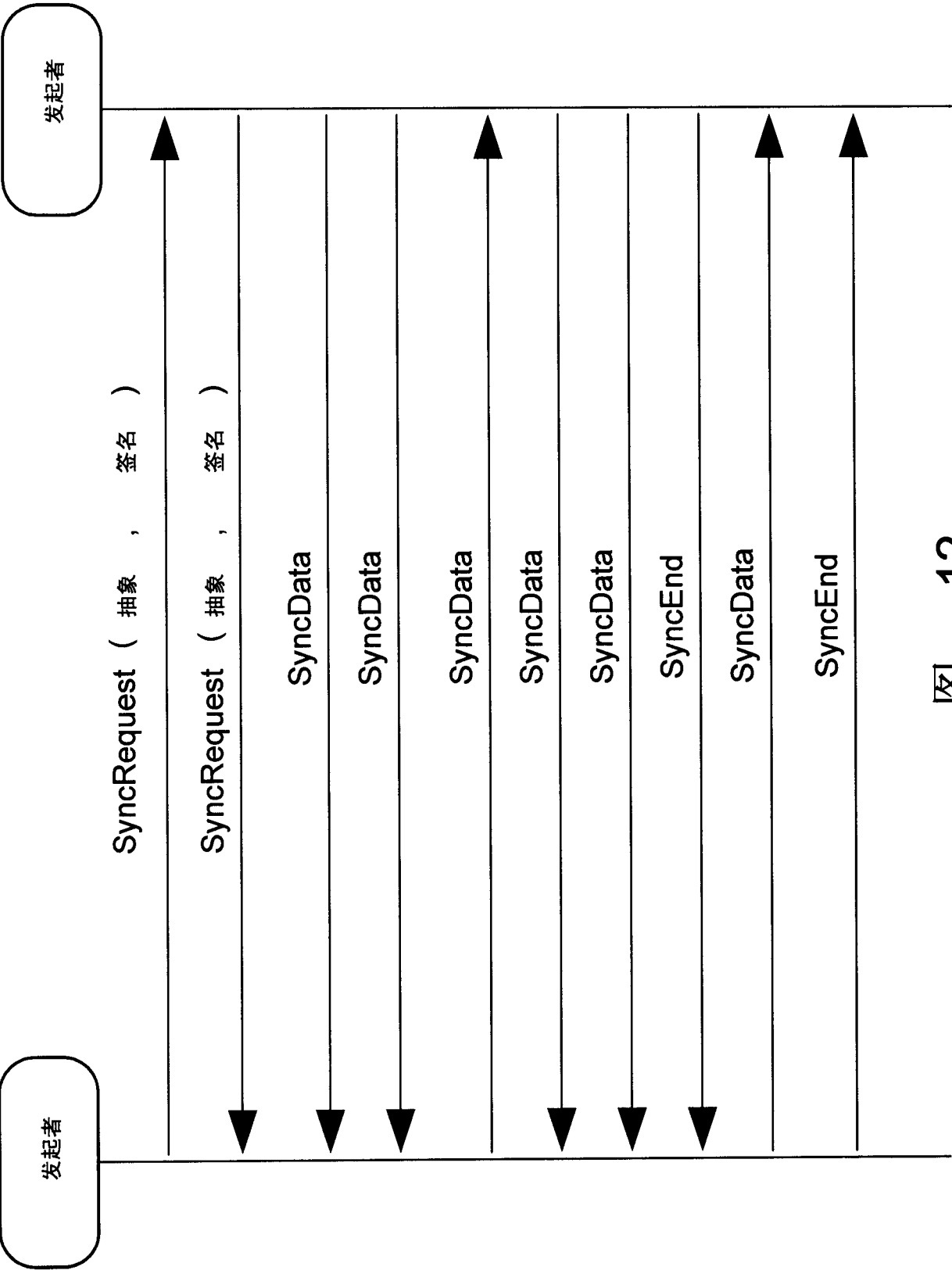


图 12

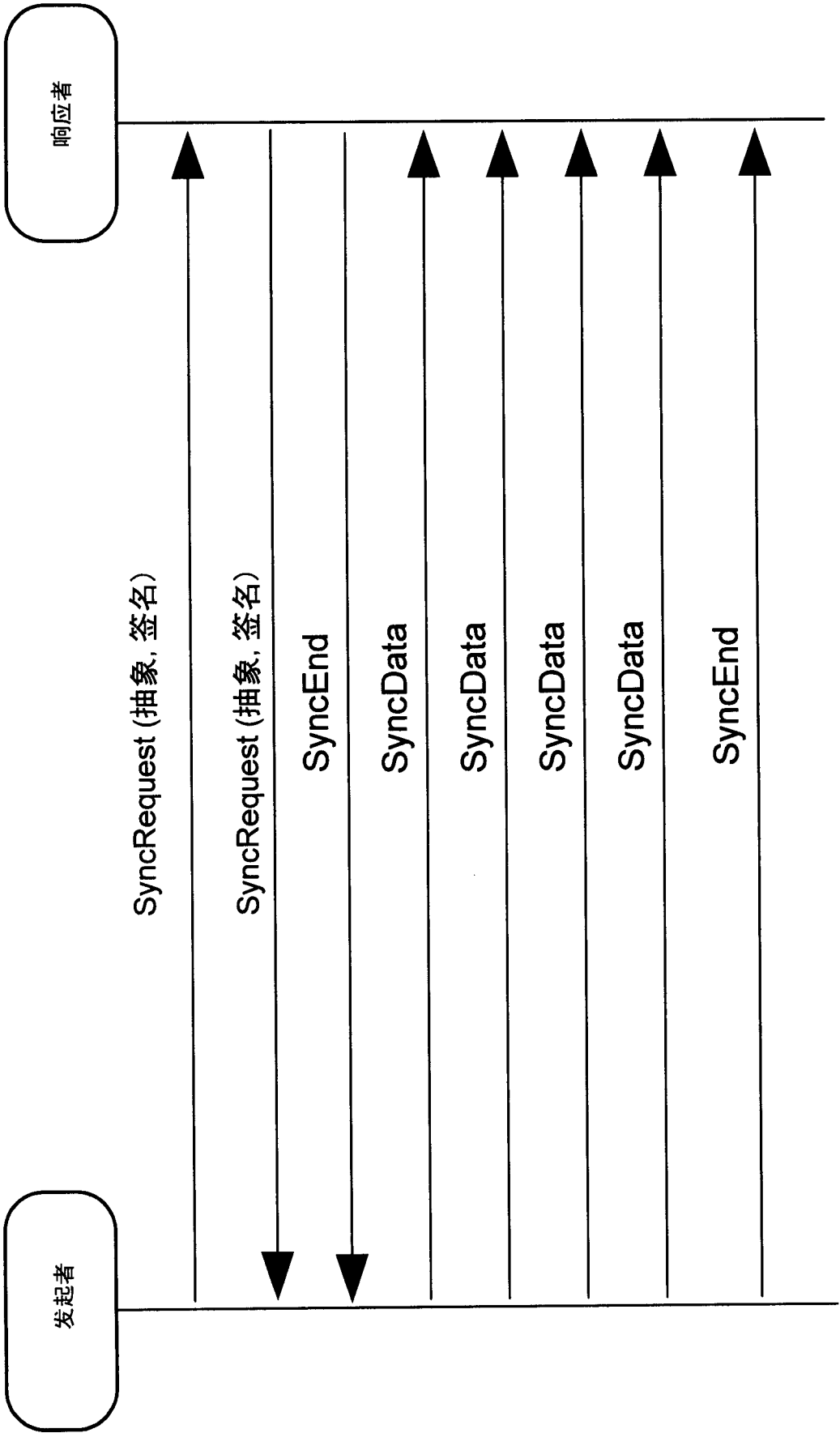


图 13

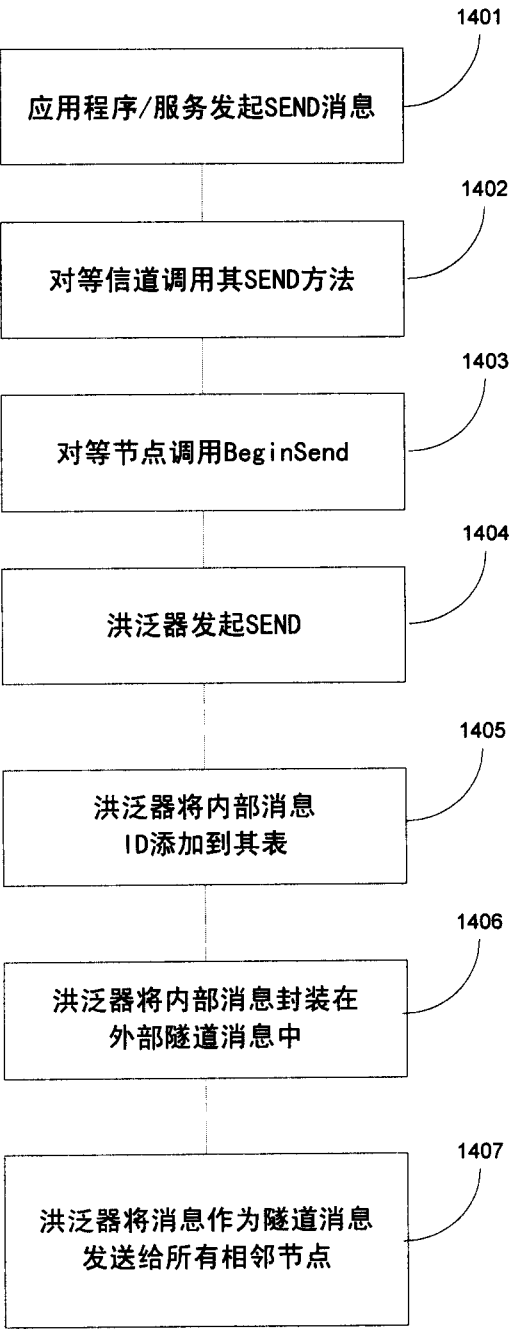


图 14

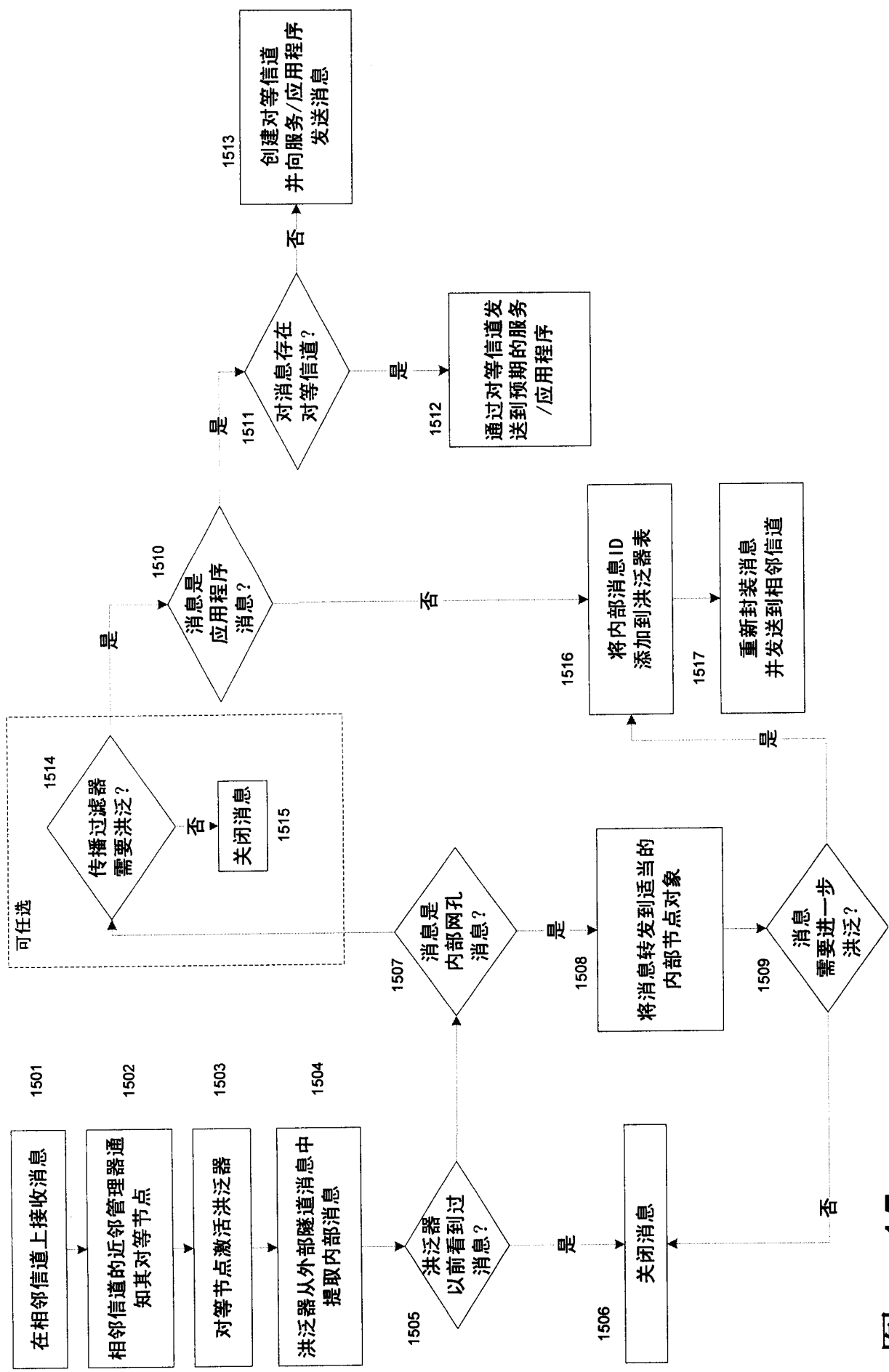


图 15

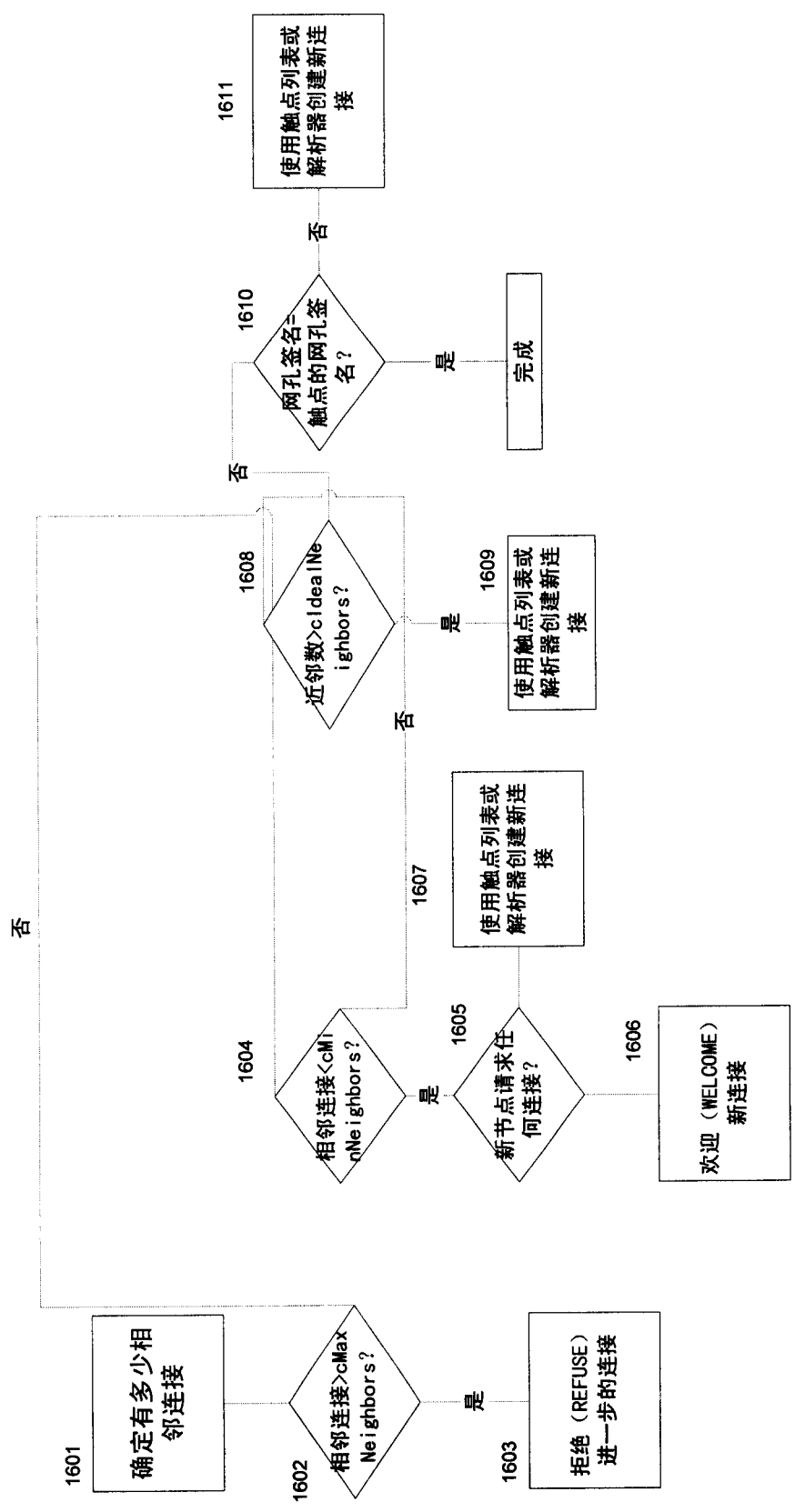


图 16

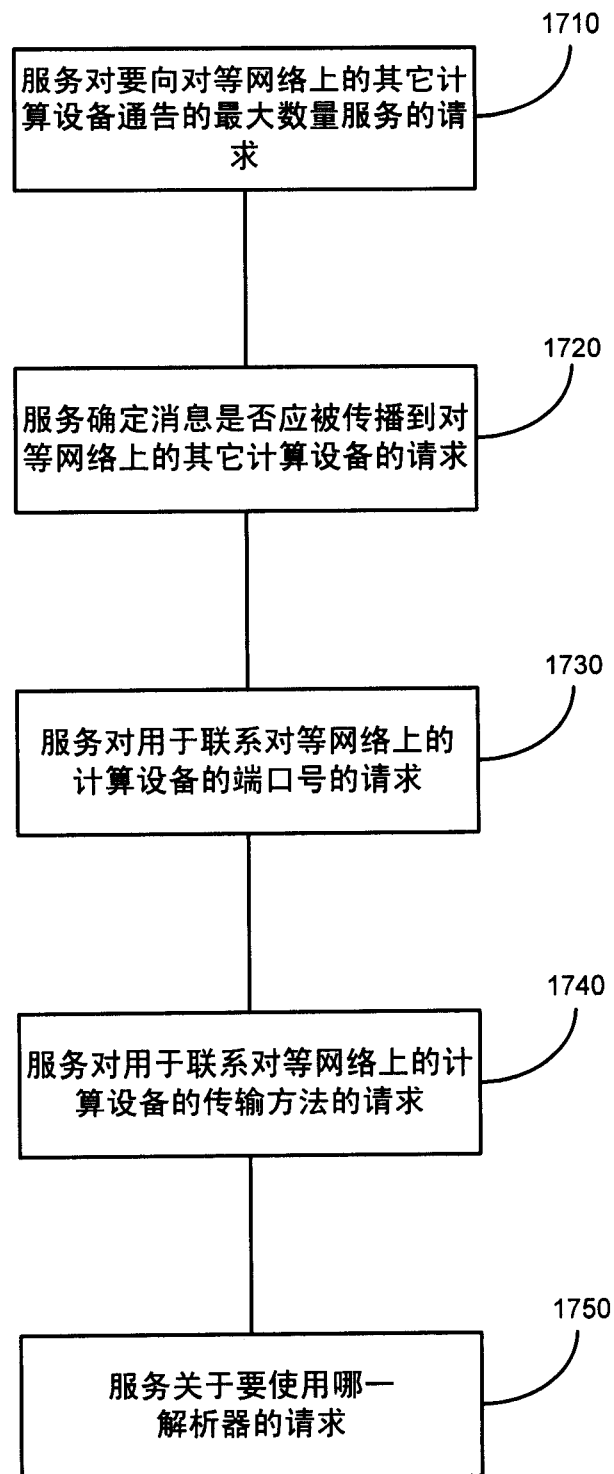


图 17