[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 12/28
H04L 12/46
H04L 12/64
H04L 29/06
G06F 9/46
G06F 13/38

[12] 发明专利申请公开说明书

[21] 申请号 03807535.0

[43] 公开日 2005 年 7 月 27 日

[11] 公开号 CN 1647455A

[22] 申请日 2003.4.9 [21] 申请号 03807535.0
[30] 优先权
[32] 2002. 4. 9 [33] EP [31] 02290890.9
[86] 国际申请 PCT/EP2003/004694 2003.4.9
[87] 国际公布 WO2003/085892 英 2003.10.16
[85] 进入国家阶段日期 2004.9.29
[71] 申请人 汤姆森许可贸易公司
地址 法国布洛里
[72] 发明人 让－巴蒂斯特·亨利 纪尧姆·比绍 若埃尔·西罗

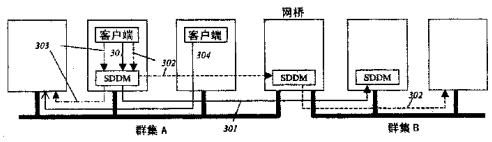
[74] 专利代理机构 中科专利商标代理有限责任公司
代理人 戎志敏

权利要求书 6 页 说明书 53 页 附图 21 页

[54] 发明名称 在多群集网络中进行通信的方法、连接设备及网桥

[57] 摘要

一种网桥设备，包括至少两个接口，用于对网络中的各个网络设备群集进行接口，其中所述网桥设备包括用于连接群集的至少两个接口入口。 所述网桥设备，针对每个入口，包括第一软件组件(SDDM)，用于从内部客户端接收对至少一个网络设备的设备描述配置存储器数据(SDD)的请求，所述第一软件组件适用于通过对其他设备中的类似软件的功能调用，从其他设备中得到设备描述数据。 本发明还涉及一种多群集网络中的设备，所述设备包括上述软件组件，以及一种设备发现方法和一种用于建立设备之间的连接的方法。



ISSN1008-4274

知识产权出版社出版

1、 一种网桥设备，包括至少两个接口，用于对网络中的各个网络设备群集进行接口，其中所述网桥设备包括用于连接群集的至少两个接口入口，

其特征在于所述网桥设备，针对每个入口，包括第一软件组件（SDDM），用于从内部客户端接收对至少一个网络设备的设备描述配置存储数据（SDD）的请求，所述第一软件组件适用于通过对其他设备中的类似软件的功能调用，从其他设备中得到设备描述数据。

2、 根据权利要求 1 所述的网桥设备，其特征在于所述第一软件组件适用于通过对去往远程群集设备的路径上的网桥设备的类似软件组件的功能调用，获得针对不具有类似软件组件的远程群集设备的数据。

3、 根据权利要求 1 或 2 所述的网桥设备，其特征在于所述第一软件组件适用于通过向所述设备发布介质相关请求消息，获得针对在与其自身相同的群集上不具有类似软件组件的设备的数据。

4、 根据权利要求 1 到 3 之一所述的网桥设备，其特征在于所述第一软件组件适用于保持下述列表中的至少一个：

a. 所述网络上的其他设备的第一软件组件的标识符列表；

b. 不具有类似第一软件组件的设备的列表，与去往所述列表中的设备的路径上的最近入口的相应标识符相关联。

5、 根据权利要求 1 到 4 之一所述的网桥设备，其特征在于所述第一软件组件适用于在其入口本地群集上监控不具有第一软件组件的设备的设备描述数据的变化，并在与所述网桥设备的其他入口相连的群集上产生相应的设备描述数据改变事件。

6、 根据前述权利要求之一所述的网桥设备，其特征在于所述网桥设备，针对每个入口，还包括第二软件组件（CMM），用于使各个入口的入口的其他软件组件与入口群集的通信介质进行接口，所述第二软件组件包括应用程序可编程接口，其中至少特定的方法能够由所述

网络的其他设备的软件组件全局地访问，以便远程访问所述通信介质。

7、 根据权利要求 6 所述的网桥设备，其特征在于所述全局可访问的方法包括写入、读取、锁定、登记、丢弃、指示中的至少一个。

8、 根据前述权利要求之一所述的网桥设备，其特征在于所述网桥设备，针对每个入口，还包括第三软件组件（NM），用于保持所述网络的所有群集上的所有设备的列表。

9、 根据权利要求 8 所述的网桥设备，其特征在于所述第三软件组件适用于在检测到所述网络的任意群集上的变化时，产生第一事件，将所述变化的本质通知给其入口的软件组件。

10、 根据权利要求 8 或 9 所述的网桥设备，其特征在于所述第三软件组件适用于产生第二事件，用于仅将事件发布入口的远程设备列表的状态通知给其他入口的第三软件组件。

11、 根据权利要求 10 所述的网桥设备，其特征在于所述第二事件包括相对于事件发布入口的远程设备，即，通过事件发布入口的共同入口能够到达的设备的可能不完全列表。

12、 根据权利要求 8 到 11 之一所述的网桥设备，其特征在于所述第三软件组件适用于产生第三事件，用于向所述群集上的所有设备的第三软件组件通知主入口的远程设备列表是稳定的。

13、 根据权利要求 12 所述的网桥设备，其特征在于所述第三事件包括相对于事件发布入口的远程设备，即，通过事件发布入口的共同入口能够到达的设备的完全列表。

14、 根据前述权利要求之一所述的网桥设备，其特征在于每个入口包括第四软件组件（EM），用于向共同入口转发在入口的本地群集上检测到的事件消息。

15、 根据权利要求 1 到 14 之一所述的网桥设备，其特征在于每个入口包括：第五软件组件（Reg），用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在其他群集上的第五软件组件转发所述请求，将初始请求者的标识符用作源地址，并将对此请求的非级联响应转发回初始请求设备。

16、 根据权利要求 1 到 14 之一所述的网桥设备，其特征在于每

个入口包括：第五软件组件（Reg），用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在其他群集上的第五软件组件转发所述请求，其中转发入口包含转发入口的地址，作为参数，用于接收和级联对所述转发请求的响应，并用于将对此请求的级联响应转发回初始请求设备。

17、 根据权利要求 16 所述的网桥设备，其特征在于用于转发所述请求的所述装置适用于使用第一消息类型向网桥设备的第五软件组件转发请求，以及使用第二消息类型向非网桥设备的第五软件组件转发请求，其中所述转发入口的标识符是所述第一消息中的参数，而不是所述第二消息中的参数。

18、 根据权利要求 1 到 14 之一所述的网桥设备，其特征在于每个入口包括：第五软件组件（Reg），用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在其他群集上的第五软件组件转发所述请求，将初始请求者的标识符用作源地址，用于截取对此转发请求的响应，用于级联这些响应的内容，并将对所述初始请求的单一级联响应发送回所述初始请求设备。

19、 根据前述权利要求之一所述的网桥设备，其特征在于所述网桥设备还包括装置，用于转换其群集的通信介质之间的分组的传送类型。

20、 根据前述权利要求之一所述的网桥设备，其特征在于每个入口包括第六软件组件（SM），用于在接收到来自另一设备的第六软件组件的连接建立请求时，针对跨越所述网桥的连接，建立本地群集上的连接段。

21、 根据权利要求 20 所述的网桥设备，其特征在于入口的所述第六软件组件适用于建立其本地群集上的连接，并通知其本地群集的下一入口，以执行去往连接末端设备的路径上的下一段建立。

22、 一种多群集网络中的群集连接设备，其中群集通过网桥设备相连，每个网桥设备包括至少两个群集接口，其中将每个接口看作其相应群集上的网络设备，其特征在于

所述网络设备包括第一软件组件（SDDM），用于从内部客户端接

收对至少第二设备的设备描述配置存储数据（SDD）的请求，所述第一软件组件适用于通过对至少一个其他设备中的类似软件的功能调用，从至少一个其他设备中得到设备描述数据。

23、 根据权利要求 22 所述的设备，其特征在于所述第一软件组件适用于通过对去往远程群集设备的路径上的网桥设备的类似软件组件的功能调用，获得针对不具有类似软件组件的远程群集设备的数据。

24、 根据权利要求 22 或 23 所述的设备，其特征在于所述第一软件组件适用于通过向位于与其相同的群集上的、不具有类似软件组件的第二设备发布介质相关请求消息，获得针对所述第二设备的数据。

25、 根据权利要求 22 到 24 之一所述的设备，其特征在于所述第一软件组件适用于保持下述列表中的至少一个：

- 所述网络上的其他设备的第一软件组件的标识符列表；

- 不具有类似第一软件组件的设备的列表，与去往所述列表中的设备的路径上的最近入口的相应标识符相关联。

26、 根据权利要求 22 到 25 之一所述的设备，其特征在于还包括第三软件组件（NM），用于保持所述网络的所有群集上的所有设备的列表，其中所述第三软件组件包括装置，用于从与其本地群集相连的入口获得远程设备列表，以及用于将远程设备列表与本地群集设备列表相级联。

27、 根据权利要求 26 所述的设备，其特征在于所述第三软件组件还适用于在网络设备列表中保持在针对相对于设备自身的本地群集的远程设备的路径上的最近入口的指示。

28、 根据权利要求 25 到 27 之一所述的设备，其特征在于所述第三软件组件适用于在检测到所述网络的任意群集上的变化时，产生第一事件，将所述变化的本质通知给其本地设备的组件。

29、 根据权利要求 22 到 28 之一所述的设备，其特征在于所述设备包括第五软件组件（Reg），用于从本地客户端接收对远程软件组件的列表的请求，以及用于仅向所述本地群集的设备的第五软件组件转发所述请求。

30、 根据权利要求 22 到 29 之一所述的设备，其特征在于所述设

备包括第六软件组件（SM），所述第六软件组件（SM）包括针对相同设备的客户端的应用程序可编程接口，适用于接收用于建立宿设备和源设备之间的连接的请求，所述第六软件元件适用于在源和宿设备之间的路径上，确定在去往宿设备的路径上距源设备最近的入口，并向该入口发送适当的请求，以建立其本地群集上的连接，并用于向该路径上的其他适当的入口传播此请求。

31、一种用于发现网络中的设备的方法，所述网络包括至少两个设备群集和至少一个网桥，其中至少两个群集通过网桥相连，每个网桥包括至少两个接口入口，用于连接相应群集，所述方法包括以下步骤：

- 使每个入口获得其本地群集的设备的标识符（GUID）列表；

- 使每个入口请求来自与其相同的群集的每个入口的远程设备列表；

- 使每个入口通过从其共同入口请求其本地设备和通过共同入口能够达到的远程设备的列表，构建其自身的远程设备列表。

32、根据前一权利要求所述的方法，其特征在于还包括以下步骤：如果网桥位于到给定设备的最短路径上，则使所述网桥通过以所述给定设备为目的地的消息。

33、根据前一权利要求所述的方法，其特征在于所述最短路径是具有最少数目的要跨越入口的路径。

34、一种建立网络中的源设备和宿设备之间的连接的方法，所述网络包括通过网桥设备相连的多个设备群集，其中每个网桥设备包括用于连接群集的接口入口，所述方法的特征在于以下步骤：

（a）在网桥的入口中以及网络的其他网桥感知设备中设置流管理器软件组件；

（b）在设备的流管理器软件组件级，从来自本地客户端的连接接收请求；

（c）在宿设备和源设备之间的路径上，标识距源设备最近的入口，并向该入口发送连接请求；

（d）使接收到所述连接请求的入口建立源设备和入口的网桥之

间的连接段；

（e）使接收到所述连接请求的入口让其网桥的下一入口在去往所述源设备的路径上建立下一入口的本地群集上的下一连接段；

（f）如果存在，标识去往宿的路径上的下一网桥，指示位于去往宿设备的路径上的下一网桥的远程入口建立适当的连接段；

（g）返回步骤（e），直到已经建立了去往宿设备的连接段为止。

在多群集网络中进行通信的方法、连接设备及网桥

技术领域

本发明涉及在多群集网络中进行通信的方法，例如，基于但不局限于 HAVi 群集，以及这种网络中的设备和用于连接群集的网桥设备。

背景技术

HAVi-表示家庭音频/视频交互，目前定义在 IEEE 1394 总线（通过 2000 版本增强的 1995 版本）上（2001 年 5 月 15 日发布的版本 1.1），并因而继承了对 IEEE 1394 的限制。一个限制是使用单群集网络。

这种 HAVi 网络难以遍布整个住宅，尽管家庭网络应当典型地连接家庭中的所有设备。需要的是连接几个不同的 HAVi 群集。

2002 年 11 月 21 日以汤姆森许可贸易公司的名义递交的 PCT 专利申请 EP02/013175 和 EP02/13179 涉及一种利用 GUID 代理技术将 HAVi 网络与 UPnP 网络相连的网关。

发明内容

本申请描述了网桥设备和网络设备，尤其是在这些设备中实现的软件组件及其在多群集环境下的相互作用。

应当注意，不同的软件组件自身构成了独立的实体和发明，并可以针对每一个单独地要求保护。

本发明涉及一种网桥设备，包括至少两个接口，用于对网络中的各个网络设备群集进行接口，其中所述网桥设备包括用于连接群集的至少两个接口入口，其特征在于所述网桥设备，针对每个入口，包括第一软件组件，用于从内部客户端接收对至少一个网络设备的设备描述配置存储器数据的请求，所述第一软件组件适用于通过对其他设备中的类似软件的功能调用，从其他设备中得到设备描述数据。

根据本发明的实施例，所述第一软件组件适用于通过对去往远程群集设备的路径上的网桥设备的类似软件组件的功能调用，获得针对不具有类似软件组件的远程群集设备的数据。

根据本发明的实施例，所述第一软件组件适用于通过向位于与其相同的群集上的、不具有类似软件组件的设备发布介质相关请求消息，获得针对所述设备的数据。

根据本发明的实施例，所述第一软件组件适用于保持下述列表中的至少一个：

a）所述网络上的其他设备的第一软件组件的标识符列表；

b）不具有类似第一软件组件的设备的列表，与去往所述列表中的设备的路径上的最近入口的相应标识符相关联。

根据本发明的实施例，所述第一软件组件适用于在其入口本地群集上监控不具有第一软件组件的设备的设备描述数据的变化，并在与所述网桥设备的其他入口相连的群集上产生相应的设备描述数据改变事件。

根据本发明的实施例，所述网桥设备，针对每个入口，还包括第二软件组件，用于使各个入口的入口的其他软件组件与入口群集的通信介质进行接口，所述第二软件组件包括应用程序可编程接口，其中至少特定的方法能够全局地访问所述网络的其他设备的软件组件，以便远程访问所述通信介质。

根据本发明的实施例，所述全局可访问的方法包括写入、读取、锁定、登记、丢弃、指示中的至少一个。

根据本发明的实施例，所述网桥设备，针对每个入口，还包括第三软件组件，用于保持所述网络的所有群集上的所有设备的列表。

根据本发明的实施例，所述第三软件组件适用于在检测到所述网络的任意群集上的变化时，产生第一事件，将所述变化的本质通知给其入口的软件组件。

根据本发明的实施例，所述第三软件组件适用于产生第二事件，用于只将事件发布入口的远程设备列表的状态通知给其他入口的第三软件组件。

根据本发明的实施例，所述第二事件包括好比是事件发布入口的远程设备，即，通过事件发布入口的共同入口能够到达的设备的潜在不完全列表。

根据本发明的实施例，所述第三软件组件适用于产生第三事件，用于通知所述群集上的所有设备的第三软件组件，主入口的远程设备列表稳定。

根据本发明的实施例，所述第三事件包括好比是事件发布入口的远程设备，即，通过事件发布入口的共同入口能够到达的设备的完全列表。

根据本发明的实施例，每个入口包括第四软件组件，用于向共同入口转发在入口的本地群集上检测到的事件消息。

根据本发明的实施例，每个入口包括：第五软件组件，用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在它的其他群集上的第五软件组件转发所述请求，将初始请求者的标识符用作源地址，并将对此请求的非级联响应转发回初始请求设备。

根据本发明的实施例，每个入口包括：第五软件组件，用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在它的其他群集上的第五软件组件转发所述请求，其中转发入口将转发入口的源地址作为参数添加到转发请求上，用于接收和级联对所述转发请求的响应，并用于将对此请求的级联响应转发回初始请求设备。

根据本发明的实施例，用于转发所述请求的所述装置适用于使用第一消息类型向网桥设备的第五软件组件转发请求，以及使用第二消息类型向非网桥设备的第五软件组件转发请求，其中所述转发入口的标识符是所述第一消息中的参数，而不是所述第二消息中的参数。

根据本发明的优选实施例，每个入口包括：第五软件组件，用于在网桥的群集之一上，接收来自另一设备的第五软件组件的请求；以及装置，用于向在它的其他群集上的第五软件组件转发所述请求，将初始请求者的标识符用作源地址，用于截取对此转发请求的响应，用

于级联这些响应的内容，并将对所述初始请求的单一级联响应发送回所述初始请求设备。

根据本发明的实施例，所述网桥设备包括装置，用于转换其群集的通信介质之间的分组的传送类型。

根据本发明的实施例，每个入口包括第六软件组件，用于在接收到来自另一设备的第六软件组件的连接建立请求时，针对跨越所述网桥的连接，建立本地群集上的连接段。

根据本发明的实施例，入口的所述第六软件组件适用于建立其本地群集上的连接，并通知其本地群集的下一入口，以执行去往连接端设备的路径上的下一段建立。

本发明的另一目的是一种多群集网络中的群集连接设备，其中群集通过网桥设备相连，每个网桥设备包括至少两个群集接口，其中将每个接口看作其相应群集上的网络设备，其特征在于所述网络设备包括第一软件组件，用于从内部客户端接收对至少一个网络设备的设备描述配置存储器数据的请求，所述第一软件组件适用于通过对至少一个其他设备中的类似软件的功能调用，从至少一个其他设备中得到设备描述数据。

根据本发明的实施例，所述第一软件组件适用于通过对去往远程群集设备的路径上的网桥设备的类似软件组件的功能调用，获得针对不具有类似软件组件的远程群集设备的数据。

根据本发明的实施例，所述第一软件组件适用于通过向位于与其相同的群集上的、不具有类似软件组件的第二设备发布介质相关请求消息，获得针对所述第二设备的数据。

根据本发明的实施例，所述第一软件组件适用于保持下述列表中的至少一个：

a）所述网络上的其他设备的第一软件组件的标识符列表；

b）不具有类似第一软件组件的设备的列表，与去往所述列表中的设备的路径上的最近入口的相应标识符相关联。

根据本发明的实施例，所述设备还包括第三软件组件，用于保持所述网络的所有群集上的所有设备的列表，其中所述第三软件组件包

括装置，用于从与其本地群集相连的入口获得远程设备列表，以及用于将远程设备列表与本地群集设备列表相级联。

根据本发明的实施例，所述第三软件组件还适用于在网络设备列表中保持在好比是设备自身的本地群集的远程设备的路径上的最近入口的指示。

根据本发明的实施例，所述设备包括第五软件组件，用于从本地客户端接收对远程软件组件的列表的请求，以及用于只向所述本地群集的设备的第五软件组件转发所述请求。

根据本发明的实施例，所述设备包括第六软件组件，所述第六软件组件包括针对相同设备的客户端的应用程序可编程接口，适用于接收用于建立宿设备和源设备之间的连接的请求，所述第六软件元件适用于在源和宿设备之间的路径上，确定在去往宿设备的路径上距源设备最近的入口，并向该入口发送适当的请求，以建立其本地群集上的连接，并用于向该路径上的其他适当的入口传播此请求。

应当注意，网桥的入口也是其本地群集上的设备。

本发明的另一目的是一种用于发现网络中的设备的方法，所述网络包括至少两个设备群集和至少一个网桥，其中至少两个群集通过网桥相连，每个网桥包括至少两个接口入口，用于连接相应群集，所述方法包括以下步骤：

- 使每个入口获得其本地群集的设备的标识符列表；

- 使每个入口请求来自与其相同的群集的每个入口的远程设备列表；

- 使每个入口通过从其共同入口请求其本地设备和通过共同入口能够达到的远程设备的列表，构建其自身的远程设备列表。

根据本发明的实施例，所述方法还包括以下步骤：如果网桥位于到给定设备的最短路径上，则使所述网桥通过以所述给定设备为目的地的消息。

根据本发明的实施例，所述最短路径是具有最少数目的要跨越入口的路径。

本发明的另一目的是一种建立网络中的源设备和宿设备之间的

连接的方法，所述网络包括通过网桥设备相连的多个设备群集，其中每个网桥设备包括用于连接群集的接口入口，所述方法的特征在于以下步骤：

（a）在网桥的入口中以及网络的其他网桥感知设备中设置流管理器软件组件；

（b）在设备的流管理器软件组件级，从来自本地客户端的连接接收请求；

（c）在宿设备和源设备之间的路径上，标识距源设备最近的入口，并向该入口发送连接请求；

（d）使接收到所述连接请求的入口建立源设备和入口的网桥之间的连接段；

（e）使接收到所述连接请求的入口使其网桥的下一入口在去往所述源设备的路径上建立入口的本地群集上的下一连接段；

（f）如果存在，标识去往宿的路径上的下一网桥，指示位于去往宿设备的路径上的下一网桥的远程入口建立适当的连接段；

（g）返回步骤（e），直到建立去往宿的连接段。

附图说明

在对本发明的优选实施例的描述中，本发明的其他特征将显而易见。本发明并不局限于实施例。将借助于附图对实施例进行描述。

图 1 是多群集 HAVi 网络的示意图；

图 2 是 HAVi-HAVi 网桥的示意图；

图 3 是示出了使用 SddManager 的不同情况的网络示意图；

图 4 是示出了使用 CMM 的示例的多群集 HAVi 网络的示意图；

图 5 是示出了通过网桥发送消息的示意图；

图 6 是表示事件发布算法的示意图；

图 7 是表示针对登记处的业务改善的示意图；

图 8 是表示针对网桥登记处的请求/响应算法的示意图；

图 9 是现有技术的 HAVi 流连接模型的示意图；

图 10 是示出了多群集流构建的示意图；

图 11 是示出了针对数据的可选路由的多群集网络示意图；

图 12 是 HAVi 网桥感知设备的软件结构的示意图；

图 13 是示出了本地发现方法的多群集网络示意图；

图 14 是示出了叶节点网桥上的第一远程列表更新的多群集网络示意图；

图 15 是示出了网桥网络管理器相互作用的多群集网络示意图；

图 16 是示出了用于构建网络管理全局列表的方法的示意图；

图 17 是示出了对网络管理器的客户端调用的多群集网络示意图；

图 18 是示出了远程列表概念的多群集网络示意图；

图 19 是示出了新设备的连接的示意图；

图 20 是示出了更新远程列表的传播的示意图；

图 21 是具有环路的网络中的本地发现方法的示意图；

图 22 是示出了入口根据其请求远程列表的方法的示意图；

图 23 是示出了以不完全远程列表进行入口更新的方法的示意图；

图 24 是示出了利用事件发送不完全远程列表的示意图；

图 25 是包括两个 HAVi 群集和利用 GUID 代理技术的网桥的网络的示意图。

具体实施方式

一种用于桥接两个 HAVi 群集的方法基于软件组件代理方案。图 25 是由通过网桥设备链接的两个 HAVi 群集形成的网络的示例。以分别被称为设备控制模块（DCM）和功能组件模块（FCM）的软件组件表示设备和子设备或功能。

HAVi 设备发现方法基于 IEEE 1394 总线上的‘GUID’识别。GUID 表示全局惟一标识符。GUID 惟一地标识了 IEEE 1394 设备。

网桥一侧的设备将不会由网桥另一侧的设备识别，因为其在 IEEE 1394 级是不可见的。一侧的控制器将不能使用另一侧的目标。网桥设备构建一侧 DCM 和 FCM 的代表，以便将其作为 DCM 和 FCM 暴露给另一侧，作为其所代表的真实软件组件的代理组件。

在图 25 中，以其 SEID（软件组件 ID）表示真实的 DCM 和 FCM。

SEID 是 GUID（在每个设备的底部示出了示例）和该设备内部惟一的号码的组合。

将这些 DCM 和 FCM 通过代理 SE（软件组件）表示在网桥的另一侧。以虚线示出，以便将其与真实的 SE 区别。针对每个真实的 DCM 和 FCM，有一个代理 SE。控制应用程序可以通过其代理 SE 控制网桥后面的真实目标设备。

本发明的本实施例将基于使用 GUID 代理的入口和网桥。但是，本发明并不局限于此特定的情况。此外，尽管 HAVi 1.1 是基于 IEEE 1394 的，但本实施例的群集可以基于其他网络技术，尤其是基于因特网协议（IP）或无线技术（IEEE 802.11、Hiperlan 2…）。在本实施例中，作为示例，这些灵活性通过使用 GUID 代理技术实现。在本发明的优先权日可用的最新的 HAVi 版本是版本 1.1。HAVi 1.1 并未描述网桥，所以如果 HAVi 1.1 设备与多群集网络相连，其将不能感知任何网桥。

本申请首先描述了 HAVi 网桥设备，然后，描述 HAVi 网桥感知设备，即，能够利用网桥设备的资源并与之通信的设备。由于网桥对于 HAVi 1.1 设备不是透明的，可能会需要这种设备。

I]网桥设备

图 1 示出了包括通过两个网桥 104 和 105 按照串联方式互连的三个群集 101 到 103 的网络。群集 101 基于 IEEE 1394，群集 102 基于 IP 技术，而群集 103 是基于如 IEEE 802.11 的无线网络。例如，群集 102 上的设备可以是 HAVi 设备，或是网桥针对其管理 HAVi 代理表示的 UPnP 设备。

根据本实施例的 GUID 代理解决方案的原理在于，在本地群集上，通告位于本地群集外部的设备的 GUID，从而本地 HAVi 设备得知它们的存在。一旦知道了远程设备的远程 GUID，则此设备就可以通过 HAVi 软件组件进行寻址，因为消息发送系统在其内部表中知道必须向哪个设备发送 HAVi 消息。当向远程设备发送 HAVi 消息时，其目的地地址为代理 GUID 的地址。网桥适当地传递来自基于被代理 GUID 的 HAVi

中间件和 HAVi 适用模块（DCM、FCM、应用程序）的消息。

在图 2 中示出了 HAVi-HAVi 网桥的软件体系结构。尽管多于两个入口也是可能的，其由根据本实施例的两个入口构成。每个入口与 HAVi 群集（例如，IEEE 1394-1995 总线）相连，并为其群集上可寻址的实体。完全的 HAVi 栈涉及网桥。HAVi 模块是否仿真其自身的两种实例或者两个独立的模块是否针对相同的功能性同时运行是与实现相关的。尽管从功能上将每个入口存在一个消息发送系统，在图 2 中，只示出了一个消息发送系统。利用驻留在其中的节点的 GUID 寻址软件组件——且消息发送系统是软件组件，并且每个入口拥有其自己的 GUID。

a）SddManager

自描述设备数据（SDD）是 HAVi 设备向其他设备提供与其自己相关的信息（设备类型、容量、版本等）的手段。在 HAVi-1.1 中，SDD 是 IEEE 1394 HAVi 设备的配置 ROM（包含如 GUID 等其他信息）的一部分，并由其他设备利用直接 IEEE 1394 读取事务进行读取。

这对于单一 IEEE 1394 群集较好，但当 HAVi 网络是多群集的，且构建在不同的介质技术上时，这种读取事务是不足够的。于是，所需的是读取网络上的任意 HAVi 设备的 SDD 数据的手段。这可以通过 HAVi 消息来实现。根据本实施例，软件组件可以利用消息发送系统来访问 HAVi 设备的 SDD 数据。为了向任意群集上的任意客户端提供所请求的 SDD 数据，在 HAVi 栈中定义了应用程序专用接口（API）。

SddManager 是一种新的系统软件组件，其在本地处理对 SDD 数据的请求和收集来自远距离 SddManager 的响应方面，具有与登记处的相似性，而与登记处之间的不同在于登记处存在于具有中间功能性（IAV）和完全功能性（FAV）的所有设备上，而在任意根据本实施例的网桥感知 HAVi 设备上实现 SddManager。网桥感知设备是 FAV 或 IAV 类型的（完全 A/V 设备或中间 A/V 设备）。在 HAVi 1.1 或更低版本的设备上不存在 SddManager。因此，具有 SddManager 的设备将与不具有 SddManager 的设备共同存在于相同的群集上。这意味着 HAVi 设备中

的客户端应用程序或软件组件将优选地调用其本地 SddManager 用于所有请求，并且本地 SddManager 将负责收集所有信息（向其他 SddManager 发送查询和/或进行本地低级操作）。如果在设备上没有本地 SddManager，则客户端将不得不通过其他手段获得信息。在后一种情况下，客户端运行在不具有网桥知识的 HAVi-1.1 设备上。然后，其只能访问本地 IEEE 1394 群集设备。

根据本实施例，客户端执行以下处理，以获得 SDD 数据：

```
        if(local SddManager exists)
(301,302,303)     call the local SddManager
        else    //(i.e. the device is not a bridge aware device)
(304)         use local API (e.g. CMM1394) to retrieve the SDD data
            of the local cluster device    // it can be only the local cluster
here
```

换句话说，客户端应用程序优选地适用于在具有 SddManager 的设备上和不具有 SddManager 的设备上进行操作。

根据本实施例，SddManager 对其从由其他 SddManager 通知的事件中获得的 SDD 数据信息进行高速缓存。这允许减少网络上的业务量，并减少做出请求时从 SddManager 到客户端的响应时间。高速缓存集中在 SddManager 中，并不必由相同设备上的几个客户端冗余地进行。

在 SddManager 级，执行以下处理：

```
        if(query from a client concerns local data only (i.e. in the same
device))
            send response
        else
            if(remote SddManager exists for the target device to be queried)
(301)             call the remote SddManager
            else
                if(target device is on the local cluster)
(303)                 use local API (e.g. CMM1394) to retrieve the SDD data
                else //i.e. the target device is not bridge aware and not
on the local cluster
```

(302) call the SddManager of the bridge exit portal

换句话说，如果从客户端接收到的查询并不是只涉及本地可用数据，则 SddManager 首先检查要获得针对其的 SDD 数据的目标设备是否包含 SddManager，并在包含的情况下，调用目标的 SddManager。否则（即，目标设备不具有 SddManager），其检查目标设备是否在本地群集上，并使用如通信介质管理器等本地 API 获得数据（例如，用于 IEEE 1394 的 CMM 1394）。对于远程非网桥感知目标设备，向本地群集的退出入口的 SddManager 转发请求。

图 3 示出了针对以上述两个处理表示的不同情况的消息序列。参考数字对应于在上述两种处理中所给出的步骤。

优选地，SddManager 存储（本地和远程）网络上所有其他 SddManager 的列表，并针对不具有 SddManager 的设备，存储最近入口的 GUID（如下所述，由网络管理器提供），以向其发送查询。

SddManager 提供以下服务：

服务	通信类型	局部性	访问
SddManager::GetSddData	M	全局	全部
SddDataChanged	E	全局	SddManager (全部)

在本实施例中，SddManager 具有以下数据结构：

(a) DeviceProfile

定义

```
struct DeviceProfile {
    DeviceClass   deviceClass;
    boolean       withDcmManager;
    boolean       withStreamManager;
    boolean       withResourceManager;
    boolean       withDisplayCapability;
    boolean       deviceActive;
```

```
    boolean        bridge;
};
```

描述

此结构在 HAVi_Device_Profile 类下存储在 IEEE 1394 配置 ROM 中找到的值（HAVi-1.1 规范，458 页）。

deviceClass 参数给出了设备的类型（FAV、IAV、…）。如果该模块出现在该设备中，则 withXxxManager 布尔值为真。withDisplayCapability 表示对于 IAV，DDI 控制器是否出现，以及对于 FAV，DDI 控制器和级 2 UI（用户接口）是否出现。如果该设备有效，则 deviceActive 布尔值为真。bridge 参数表示该设备是否为网桥。

(b) Vendor

定义

```
struct Vendor {
    VendorId          vendorId;     //defined      in
HAVi-1.1 p110
    DeviceManufacturer vendorText;  //defined   in
HAVi-1.1 p149
};
```

描述

与制造商有关的信息。最大字符数为 50，以 UNICODE UTF-16 在 2 个字节上进行编码，所以最大尺寸为 100 字节。

(c) Model

定义

```
struct Model {
    ModelId     modelId;      //defined in HAVi-1.1
p200
    DeviceModelmodelText;   //defined in HAVi-1.1
```

```
p149
    };
```

描述

此结构给出了与模型有关的信息。最大字符数为 50，以 UNICODE UTF-16 在 2 个字节上进行编码，所以最大尺寸为 100 字节。

(d) DcmProfile

定义

```
struct DcmProfile {
    uint     transferredDcmCodeUnitSize;
    uint     installedDcmCodeSpace;
    uint     installedDcmWorkingSpace;
    Version    MessageVersion;      //defined  in
HAVi-1.1 p110
    };
```

描述

此结构包含与 DCU 有关的信息。这些字段与 HAVi-1.1 规范在第 9.10.7 节 460 页中定义的相同。

(e) SddData

定义

```
struct SddData {
    Version          version;              //defined
in HAVi-1.1 p110
    DeviceProfile      deviceProfile;
    Vendor         vendor;
    Model              model;
    UserPreferredName  userPreferredName;
//defined in HAVi-1.1 p150
    DeviceIcon     deviceIcon;          //defined
```

```
in HAVi-1.1 p158
        DcmProfile        dcmProfile;
        sequence<octet>    dcmReference;
    };
```

描述

此结构提供了与 HAVi 设备有关的信息。其基本上是与 HAVi-1.1 规范的 IEEE 1394 配置 ROM SDD 部分中相同的信息。对于与这些字段有关的详细信息，参见 HAVi-1.1 规范第 9 节。注意，在设备配置文件（profile）中添加了一比特，网桥比特，由 deviceProfile 结构体中的网桥布尔值表示。此数据块用于表示 HAVi 设备是否为网桥。还应当注意，dcmProfile 和 dcmReference 仅针对 BAV 设备是有效字段。

SddManager 的应用程序可编程接口（API）如下：

SddManager::GetSddData

原型

```
Status SddManager::GetSddData(in GUID guid, out SddData sddData)
```

参数

- guid - 客户端想要获得针对其的 SDD 数据的 GUID。
- sddData - 指定 GUID 的 SDD 数据。

描述

此方法获得针对由其 GUID 指定的给定 HAVi 设备的 SDD 数据。如果 GUID 是本地设备（客户端的主机）的 GUID，则本地 SddManager 与相应的 SDD 数据一起发送响应。如果 GUID 是远程设备的 GUID，则本地 SddManager 负责获得远程 SDD 数据。根据上面已经展示的处理来完成。

错误代码

- SddManager::EUNKNOWN_GUID - GUID 未知。
- SddManager::ENOT_READY - 当前正在更新 SDD 数据。客户端可以稍后再试。
- SddManager::ELAV - 指定的 GUID 是 LAV 设备，所以不具有 SDD

数据。

SddManager 使用以下事件：

SddDataChanged

原型

```
void SddDataChanged(in GUID guid, in SddData sddData)
```

参数

- guid - 已经改变了 SDD 数据的设备的 GUID
- sddData - 新 SDD 数据。

描述

本事件用于将由 GUID 指定的设备的 SDD 数据的变化通知给网络上的设备。作为 SddManager 的主机的设备可以提供针对其本地 SDD 数据的这种事件。

网桥设备可以通过检测到具有改变后的 SDD 的设备的本地入口上的 SDD 数据变化（例如，通过针对 IP 群集的组播消息）并向其他入口的 SddManager 传输信息，来提供针对不具有 SddManager 的远程设备的 SDD 数据的事件。

在这种情况下（当网桥传播针对不具有 SDD 管理器的设备的事件时），包括不具有 SddManager 的设备的群集的所有入口将向其自身的共同入口传输该事件，而其共同入口将依次远程地转发该事件，根据已知的 SDD 处理（针对 IEEE 1394 网络的总线复位和配置 ROM 读取，针对 IP 网络的组播分组的发送）更新本地群集。例如，不具有 SDDManager 的设备可以是如基本（BAV）非 IEEE 1394 设备的 HAVi 1.1 设备。对于遗留设备（LAV），并不存在问题，因为其不具有 SDD 数据。

如下执行 IEEE 1394 配置 ROM 增强：

为了与 SddManager 结构体的定义相一致，在 IEEE 1394 HAVi 设备的配置 ROM 中增加新的字段，如下。

此 HAVi_Device_Profile 是 IEEE 1394 配置 ROM 的 24 比特立即值（如 IEEE 1212 所规定的那样）字段，已经包括：

- HAVi_Device_Class [bit 0...3]

- HAVi_DCM_Manager [bit 4]
- HAVi_Stream_Manager [bit 5]
- HAVi_Resource_Manager [bit 6]
- HAVi_Display_Capability [bit 7]
- HAVi_Device_Status [bit 8]
- Bits 9...23 保留

在比特 9 中增加新的字段：

HAVi_Bridge 是 1 比特立即值字段，针对 IAV/FAV 设备规定了此设备是否为 HAVi 网桥。对于 BAV，此比特应当为 0。

HAVi_Bridge 值	含义
0	不是网桥
1	是网桥

b）通信介质管理器

现在，对网桥入口的修改后的通信介质管理器（CMM）进行描述：

根据本实施例，网桥的 CMM（事实上为每个入口的 CMM，由于每个网桥有几个 CMM）的 API，对于其 API/方法中的至少一些是全局可访问的，代替了只能由主机设备的元件组件进行访问（即，本地可访问性）。这对于每类 CMM 都是有效的。此后，将描述针对基于 IEEE 1394 的设备的 CMM（CMM1394）和针对基于 IP 设备的 CMM（CMMIP），由于其存在于网桥设备中。

CMM1394 API 为：

服务	通信类型	局部性	由（目的地）访问/针对事件：由（目的地）发送
Cmm1394::GetGuidList	M	本地	全部
Cmm1394::Write	M	本地 网桥中全局	信任

PC041008

Cmm1394::Read	M	本地 网桥中全局	信任
Cmm1394::Lock	M	本地 网桥中全局	信任
Cmm1394::EnrollIndication	M	本地 网桥中全局	信任
Cmm1394::DropIndication	M	本地 网桥中全局	信任
<Client>::Cmm1394Indicatio n	MB	本地 网桥中全局	CMM1394 (信任)
NewDevices	E	本地	CMM1394 (全部)
GoneDevices	E	本地	CMM1394 (全部)
NetworkReset	E	本地	CMM1394 (全部)
GuidListReady	E	本地	CMM1394 (全部)

CMMIP API 如下：

服务	通信类型	局部性	访问
Cmmip::GetGuidList	M	本地	全部
Cmmip::GetIpAddress	M	本地 网桥中全局	信任
Cmmip::GetGuid	M	本地 网桥中全局	信任
Cmmip::Send	M	本地 网桥中全局	信任
Cmmip::EnrollIndication	M	本地 网桥中全局	信任
Cmmip::DropIndication	M	本地 网桥中全局	信任
<Client>::CmmipIndication	MB	本地 网桥中全局	CMMIP (信任)

NewDevices	E	本地	CMMIP (全部)
GoneDevices	E	本地	CMMIP (全部)
ChangedDevices	E	本地	CMMIP (全部)
GuidListReady	E	本地	CMMIP (全部)
ProxyGuidCreated	E	全局	CMMIP (CMMIP)

enroll 和 drop API 允许远程 HAVi 软件组件从 CMM 的入口的网络本地的设备接收低级消息。'send' API（针对 IP 的 send，针对 IEEE 1394 的 read-write-lock）允许向远程群集上的特定设备发送消息。例如，这些装置可以由通过远程安装的设备控制模块（DCM）使用，以便通过一个或多个网桥，与其受控设备进行通信。

想要使用远程 CMM（即，不位于与其相同的设备中）的 HAVi SE 必须知道由远程 CMM 使用的链接技术，即，其必须知道上述 API。

由 HAVi SE 使用的处理是：

• 发现远程 CMM 技术，通过以软件组件类型属性值 0x00000001（即，通信介质管理器）查询本地登记处（其向远程登记处发送查询），并获得远程 CMM 的 SEID。根据本实施例，此 SEID 的软件句柄（swHandle）部分给出 CMM 的类型（针对 CMM1394 的 0x0001、针对 CMMIP 的 0x0009…

• 如果 SE 知道如何处理相应的链接技术，则其可以使用 CMM。例如，其必须能够根据该技术，规定要发送给远程设备的消息的内容。

图 4 示出了其 GUID 等于 1 的设备的软件组件指示网桥的远程 CMMIP 向 GUID 等于 3 的远程 IP 设备发送 IP 消息的示例。

总之，至少使网桥入口的 CMM 的特定的功能能够由除了 CMM 自身设备本地的客户端以外的其他客户端访问，尤其是，为了允许这些客户端使用 CMM 的 API，以便直接在不同的网络技术上进行通信，例如，发送低级消息等。

c）设备发现/网络管理器

根据本实施例，创建网络管理器软件组件，以提供与连接于整个

HAVi 网络的所有设备有关的信息。CMM 提供与其本地群集相连的 GUID 列表。网络管理器能够给出整个多群集网络的 GUID 列表，包括本地群集。每个入口中都存在网络管理器。优选地，网络管理器也出现在网桥感知设备中。

由网络管理器提供以下服务：

服务	通信类型 (消息或事件)	局部性访问/对于事件：发送方 (接收方)	
NetworkManager::GetNetDeviceList	M	本地	全部
NetworkManager::GetNetDeviceInfo	M	全局	全部
NetworkManager::NetworkUpdated	E	本地	网络管理器(全部)
NetworkManager::GetRemoteDeviceList	M	全局	网络管理器
NetworkManager::RemoteNetworkChanged	E	全局	网桥中的网络管理器 (网桥中的网络管理器)
NetworkManager::RemoteNetworkUpdated	E	全局	网络管理器 (向所有网络管理器发送)

表 7

网络管理器数据结构如下：

(a) ClusterType

定义

```
enum ClusterType {IEEE1394,IP};
```

描述

ClusterType 提供了与特定群集上所使用的技术有关的信息。根据本实施例，定义了两种群集技术：1394(HAVi-1.1)和 IP，但也可以容易地添加其他技术。

(b) NetDeviceInfo

定义

```
struct NetDeviceInfo {
   GUID        deviceGuid;
   uint        hops;
   GUID        nearestPortalGuid;
   ClusterType   clusterType;
};
```

描述

NetDeviceInfo 结构提供了与设备有关的信息，而无论其在网络中的位置，即其提供 GUID 本身、到达其的跳数（由网络管理器使用来解决环路问题，如稍后所述）、到达此设备 1 的最近入口的 GUID 及其与之相连的群集的类型。最后两项允许客户端到达其最近入口上的 CMMXXX，以便访问远程群集的低级特征，并向远程设备发送低级消息。

(c) RemoteNetworkState

定义

```
enum   RemoteNetworkState   {STABLE,   CHANGING,
FINAL};
```

描述

RemoteNetworkState 提供了与入口后面的远程网络（即，位于包括网络管理器的入口后面的群集）的状态有关的信息。STABLE 表示入口的远程群集设备列表是稳定的，且其他网络管理器可以依赖其。CHANGING 表示远程列表仍在进行修改。FINAL 表示远程列表应当是稳定的，但需要群集的其他入口的确认（详见，行为发现处理）。

网络管理器 API 如下：

(a) NetworkManager::GetNetDeviceList

原型

```
Status NetworkManager::GetNetDeviceList(
   out uint updateId,
   out                    sequence<NetDeviceInfo>
```

```
activeNetDeviceList,
        out                          sequence<NetDeviceInfo>
nonactiveNetDeviceList)
```

参数

* updateId – 网络的更新号码。每次改变列表时，此号码递增 1，其可以由客户端使用来检查网络是否保持相同。

* activeNetDeviceList – 网络上的所有有效设备（active device）的列表。第一项应当为本地设备。

* nonactiveNetDeviceList – 网络上的所有非有效设备（non-active）的列表。

描述

此 API 返回整个网络上的所有设备的列表，分为有效设备列表和无效设备列表。在 NetDeviceInfo 结构中包含与每个设备有关的信息。这给出了设备的 GUID、到达其最近的入口的 GUID 和其所依附的群集的类型。例如，在图 4 中，SE 获得 GUID 3，得知其群集是基于 IP 的，且到达其的最近入口是 GUID 6，所以其可以向其 GUID 等于 6 的设备的 CMMIP 发送 CmmIp::Send HAVi 消息。

错误代码

* NetworkManager::ENOT_READY – 列表仍不可用，系统可能正在对其进行更新。这是一种瞬时错误，客户端软件组件可以重试或使用 NetworkUpdated 事件。

(b) NetworkManager::GetRemoteDeviceList

原型

```
Status NetworkManager::GetRemoteDeviceList(
    out uint updateId,
    out                          sequence<NetDeviceInfo>
activeRemoteDeviceList,
    out                          sequence<NetDeviceInfo>
nonactiveRemoteDeviceList)
```

参数

▪ updateId – 网络的更新号码。每次改变列表时，此号码递增 1，其可以由客户端使用来检查网络是否保持相同。

▪ activeNetDeviceList – 此网桥后面的网络上的所有有效设备的列表。

▪ nonactiveNetDeviceList – 此网桥后面的网络上的所有非有效设备的列表。

描述

此 API 返回在包括网络管理器的网桥后面的网络上的所有可到达设备的列表，分为有效设备列表和无效设备列表（有效设备为准备好接收消息的设备，如设备的 SDD 数据所规定的那样，针对 IAV 和 FAV 的 HAVi，专用于 BAV）。在 NetDeviceInfo 结构中包含与每个设备有关的信息。这给出了设备的 GUID、到达其的退出入口的 GUID、跳数、最近入口和其所依附的群集的类型。在本实施例中，将对此 API 的访问限制于网络管理器。由设备（网桥或非网桥）的网络管理器使用，查询网桥设备的远程设备列表，并因而构建其内部表，并解决环路问题。根据变体，使 API 对于其他软件组件也是可用的。

错误代码

▪ NetworkManager::ENOT_READY – 列表仍不可用，系统可能正在对其进行更新。这是一种瞬时错误，客户端软件组件可以重试或使用 RemoteNetworkChanged 事件(只针对网桥设备的网络管理器)和 RemoteNetworkUpdated 事件(针对所有网络管理器)。

(c) NetworkManager::GetNetDeviceInfo

原型

```
Status NetworkManager::GetNetDeviceInfo(
   in GUID guid,
   out NetDeviceInfo deviceInfo)
```

参数

▪ guid – 客户端想要与其有关的一些信息的 GUID。

• deviceInfo – 与此设备有关的信息，即，相应的NetDeviceInfo 结构体。

描述

此 API 提供了与给定网络设备有关的完整信息。网络管理器返回 NetDeviceInfo 结构。

错误代码

• NetworkManager::EUNKNOWN_GUID – GUID 未知。

根据本实施例，网络管理器事件如下：

(a) NetworkUpdated

原型

```
void NetworkUpdated(
   in uint updateId,
   in                    sequence<NetDeviceInfo>
activeNetDeviceList,
   in                    sequence<NetDeviceInfo>
nonactiveNetDeviceList,
   in sequence<GUID> changedDevices,
   in sequence<GUID> goneDevices,
   in sequence<GUID> newDevices)
```

参数

• updateId – 网络的更新号码。每次改变列表时，此号码递增 1，其可以由客户端使用来检查网络是否保持相同。

• activeNetDeviceList – 整个 HAVi 网络上的有效设备的列表。第一项应当为本地设备。

• nonactiveNetDeviceList – 整个HAVi 网络上的非有效设备的列表。

• changedDevices – 改变了跳数和最近入口的 GUID 的列表。

• goneDevices – 离开网络的 GUID 的列表。

• newDevices – 加入网络的 GUID 的列表。

描述

NetworkUpdated 是本地事件，发送给作为网络管理器的主机的设备的软件组件。当在 HAVi 网络上某处存在变化（无论什么群集）时，即，在网络管理器从与其本地群集相连的网桥接收到一个或多个 RemoteNetworkUpdated 事件之后（远程变化），或在其本地群集上的变化之后，产生此事件。在重新配置期间，网络管理器可以返回针对 NetworkManager::GetNetDeviceList API 的 NetworkManager::ENOT_READY，直到 NetworkUpdated。activeNetDeviceList 和 nonactiveNetDeviceList 内容的定义与 NetworkManager::GetNetDeviceList API 中所定义的相同。changedDevices, goneDevices 和 newDevices 字段只提供 GUID，因为在旧设备列表中，离开的设备是已知的，而在新设备列表中为新的改变后的设备提供了完整的信息。

根据变体实施例，如果相同群集上的其他设备不具有网络管理器，则使此事件能够由驻留在这些设备中的软件组件访问。

(b) RemoteNetworkUpdated

原型

```
void RemoteNetworkUpdated(
    in uint updateId,
    in sequence<NetDeviceInfo> activeRemoteDeviceList,
    in sequence<NetDeviceInfo> nonactiveRemoteDeviceList,
    in sequence<GUID> changedDevices,
    in sequence<GUID> goneDevices,
    in sequence<GUID> newDevices)
```

参数

- updateId – 网络的更新号码。每次改变列表时，此号码递增 1，其可以由客户端使用来检查网络是否保持相同。

▪ activeNetDeviceList – 整个 HAVi 网络上的有效设备的列表。

▪ nonactiveNetDeviceList – 整个 HAVi 网络上的非有效设备的列表。

▪ changedDevices – 改变了跳数和最近入口的 GUID 的列表。

▪ goneDevices – 离开网络的 GUID 的列表。

▪ newDevices – 加入网络的 GUID 的列表。

描述

RemoteNetworkUpdated 是只针对网络管理器的全局事件。当网络管理器已经检测到其正在针对其本地群集而代理的部分网络中的变化时，以及当网络被认为是稳定的时，产生此事件（与下一事件相反，仅向网桥网络管理器发送，并在列表的状态还未稳定时使用）。因为网络管理器共同入口群集上的改变或因为由与其共同入口相连网桥转发的改变，此事件可能发生。在重新配置期间，网络管理器可以返回针对 NetworkManager::GetNetDeviceList API 的 NetworkManager::ENOT_READY，直到 RemoteNetworkUpdated。activeRemoteDeviceList 和 nonactiveRemoteDeviceList 内容的定义与 NetworkManager::GetRemoteDeviceList API 中所定义的相同。changedDevices, goneDevices 和 newDevices 字段只提供 GUID，因为在旧设备列表中，离开的设备是已知的，而在新设备列表中为新的改变后的设备提供了完整的信息。

(c) RemoteNetworkChanged

原型

```
void RemoteNetworkChanged(
    in RemoteNetworkState state,
    in uint updateId,
    in                    sequence<NetDeviceInfo>
activeRemoteDeviceList,
    in                    sequence<NetDeviceInfo>
```

```
nonactiveRemoteDeviceList,
        in sequence<GUID> changedDevices,
        in sequence<GUID> goneDevices,
        in sequence<GUID> newDevices)
```

参数

- state – 远程网络的状态，这对于以远程列表迭代来解决环路问题是有用的。
- updateId – 远程网络的更新号码。每次改变列表时，此号码递增1，其可以由客户端使用来检查远程网络是否保持相同。
- activeNetDeviceList – 整个HAVi网络上的有效设备的列表。
- nonactiveNetDeviceList – 整个HAVi网络上的非有效设备的列表。
- changedDevices – 改变了跳数和最近入口的GUID的列表。
- goneDevices – 离开网络的GUID的列表。
- newDevices – 加入网络的GUID的列表。

描述

RemoteNetworkChanged 是只以网桥设备的网络管理器为目的地的全局事件。其与 RemoteNetworkUpdated 事件相同，但在其重新配置步骤期间，此事件由网桥设备中的网络管理器使用。在这些步骤期间，在达到稳定的网络状态（尤其是，如果出现环路）之前，可以产生几个事件。这避免了因为网络仍未稳定而向网桥感知设备发送不会被使用的消息。此字段的含义与针对 RemoteNetworkUpdated 事件的含义相同。

根据以上描述，在根据本实施例的网桥之间的发现处理如下：

目的在于发现所有与 HAVi 网络相连的设备。一旦完成，每个入口中的‘远程’设备列表给出与通过其自身能够达到哪些设备有关的信息。尽管 IEEE 1394.1 拓扑通过屏蔽（mute）网桥而打开环路，但本实施例中，与环路有关的行为却不同。根据本实施例，网桥可能对于特定的路径是可通过的，而对其其他路径则不行，所以当环路存在时，并不将网桥全部屏蔽，而是如果其位于去往由 GUID 标识的设备的

特定路径上，其将提供其远程列表中的这些 GUID。根据以下所解释的特定标准来决定是否通过网桥。在本实施例中，跳数反映出为了到达目的地而跨越的入口数，而不是网桥数。由于可以通过其共同入口而不是通过其群集到达入口，所以做出这种选择（即，尽管对于群集上的非入口设备，将不得不完全跨越网桥，但对于去往入口的消息也不必跨越整个网桥）。

基本发现处理如下：

每个本地群集执行其自身的本地发现处理。其自身所知的针对 IEEE 1394 群集的发现处理是基于 IEEE 1394 总线复位的，包括利用‘selfID’分组的拓扑信息传播。

一旦此阶段完成，CMM1394 读取所有节点的配置 ROM，以获得其 GUID。（如果出现）读取 SDD 数据以获得与相连设备有关的 HAVi 定义消息。

其自身所知的针对 IP 群集的发现处理是基于组播通告分组的。

根据本实施例的 CMMIP 根据这些通告，构建其 GUID 列表，例如，当新设备与群集相连时，出现这些通告。在这些分组中也包含 SDD 数据。

在这一点上，本地 GUID 列表和 SDD 数据对于两种群集类型都是已知的，因而群集的网络管理器知道出现在其本地群集上的网桥设备。

为了构建完整的网络设备列表，网络管理器开始彼此查询。

根据本实施例的处理为：

（1）网络管理器查找其群集上的网桥。

（2）网桥设备的网络管理器调用与其本地群集相连的其他网络的入口的 NetworkManager::GetRemoteDeviceList API。这些被查询入口的共同入口应当构建了其网络侧的设备的列表。

（3）网桥设备的网络管理器（通过 HAVi 消息，或者根据优选实施例，通过网桥内部信息共享等）从其共同入口获得将成为其自身的远程设备列表的列表。当共同入口已经调用了与其自身的群集相连的其他网桥设备的 NetworkManager::GetRemoteDeviceList API 时，共同入口能够提供此列表。

（4）网桥感知（BA）设备的网络管理器调用与其本地群集相连的网桥设备的 NetworkManager::GetRemoteDeviceList API。这样构建了其完整网络 GUID 列表。

HAVi SE 调用其本地网络管理器的 NetworkManager::GetNetDeviceList API。

如果在所请求的信息可用之前进行此步骤，则可以返回 ENOT_READY 错误，并且客户端将必须等待。根据本实施例，步骤 1 和 2 实际上并行进行，所以提出了避免死锁问题的机制。设备列表构建处理从拓扑的叶节点群集进行到根节点群集（至少针对不具有环路的网络）。

发现规则如下：

将以下规则应用于发现处理。根据远程网络状态，即‘改变中’、‘最终’或‘稳定’对其进行分类。此外，存在一些普通规则，无论任何状态。因此，将规则分类为“G”、“C”、“F”和“S”类。

G1	在第一本地发现之后，远程列表为空（如果接收到相应的请求，则发送响应 ENOT_READY），向共同入口请求远程列表（即使是不完全的，但至少为共同入口的本地列表），然后请求来自可能存在的群集的其他入口的远程列表。
G2	在重新配置本地群集和对新/离开设备的本地发现之后，检查群集上的新网桥。
G2.1	没有新网桥 => 保持先前的状态，并且如果是稳定状态，则更新共同入口。
G2.2	新网桥 => 检查新本地列表与旧远程列表之间的重复 GUID（例如，当新网桥允许比旧拓扑更短的路由时可能发生）。
G2.2.1	重复 GUID => 进入变化中状态，并请求新入口的远程列表。
G2.2.2	没有重复 GUID => 进入最终状态，并请求新入口的远程列表。
G3	重复管理：具有最小跳字段的入口胜出（即，将其选为处于去往设备的路径上），并在跳数相等的情况下，具有最高反向 GUID 的入口胜出（当然也可以使用解决这种平局的其他标准）。

C1	当接收到来自群集的入口的 ENOT_READY 错误响应时，如果自从最后一次更新以来，发生了一些变化，则以不完全列表（其至少为本地群集的列表）更新共同入口，如果没有变化，则 S3。针对来自入口的事件，等待特定的时间量，并且如果未接收到事件，再次请求远程列表。
C2	当从群集的另一入口接收改变中事件或响应时，检查重复 GUID。
C2.1	如果重复 GUID => 进入最终状态，去除重复 GUID（跳到规则 G3），并在群集上发送最终事件。
C2.2	如果没有重复 GUID => 如果不是 S3，将共同入口更新为改变中状态。
C3	来自改变中状态的共同入口的更新 => 检查重复 GUID。
C3.1	重复 GUID => 进入最终状态，去除重复 GUID（跳到规则 G3），并在群集上发送最终事件。
C3.2	没有重复 GUID => 如果不是 S4，在群集上发送改变中事件。
F1	接收来自群集的另一入口的最终事件 => 检查重复 GUID。
F1.1	重复 GUID：对重复进行管理（跳到规则 G3）
F1.1.1	所有都有利于他(F1 的其他入口) => 发送稳定事件进行确认。
F1.1.2	至少一个有利于我(执行处理的当前接口) => 与更新列表一起发送最终事件。
F1.2	没有重复 GUID =>发送稳定事件进行确认。
S1	接收来自群集的所有其他入口的稳定列表 => 如果稳定列表与所发送的先前的稳定列表不同，则以此列表更新共同入口。
S2	来自共同入口的稳定更新 => 检查重复 GUID。
S2.1	重复 GUID => 已知并进行了管理吗？
S2.1.1	是 => 如果最后一个事件不是稳定事件，或者如果是来自最后一次发送的稳定事件的新/离开 GUID，则发送稳定事件。
S2.1.2	否 =>进入最终状态，去除重复 GUID（跳到规则 G3），并在群集上发送最终事件。
S2.2	没有重复 GUID => 在群集上发送稳定事件。
S3	当远程列表处于稳定状态时，在从群集的其他入口接收到不稳定列表时，不进行对共同入口的更新。
S4	如果群集的所有其他入口是稳定的，针对来自不稳定列表的共同入口的更新，不在群集上发送事件。

表 8

上述处理包括确保直通的步骤，如果需要，包括适当地解决冗余路径冲突的迭代处理。为此，定义了三种可能的状态——改变中、稳定和最终。利用RemoteNetworkChanged事件中的RemoteNetworkState数据结构，传播与这些状态有关的信息。

根据变体实施例，在此发现处理中实现超时处理，以便避免使较慢的网桥在能够应答之前被来自其他网桥的大量事件冲击。

(d)消息发送

根据HAVi规范，从一个软件组件向另一个软件组件发送HAVi消息。通过SEID（软件组件标识符）来标识软件组件。此SEID由软件组件驻留在其中的设备的GUID和该设备内惟一的swHandle构成。HAVi消息的报头包含目的地SEID和源SEID。

在本实施例中，网桥设备并不修改HAVi消息（这里所称的HAVi消息不包含TAM报头）。目的地SEID、源SEID、协议类型、消息类型、消息号码、消息长度和消息体字段保持不变。但是，消息发送系统将消息路由到目的地。为此，当网桥中的消息发送系统在群集上接收到HAVi消息时，其查看目的地SEID，更精确地，查看包含在此SEID中的GUID。如果此GUID是其自身的GUID，则此消息是针对内部SE的，并传递该消息。如果此GUID出现在其远程GUID列表中，则其将此消息转发到其共同入口（或者适当的共同入口，如果存在多于一个共同入口的话）。然后，共同入口将向相应的目的地设备（考虑其内部表）发送此消息。此设备可以是最终目的地设备或者是路径上的下一个网桥。

对于消息发送系统的普通行为，并未发生任何改变：

- 消息号码仍然遵循HAVi-1.1规范第3.2.1.2.3节第29页中的规则。对于消息的最初发送方和最终接收方也是如此。位于路径上的网桥中的消息发送系统并不关心HAVi消息内部是什么，而只是对其进行转发。
- 只发送‘简单’消息（即，不要求确认），而不要求确认。
- 发送‘可靠’消息，并阻止调用方，直到接收到肯定或否定

确认（Ack 或 Nack）为止。对于最初发送方也是如此，位于路径上的网桥中的消息发送系统只是转发消息（初始消息，Ack 或 Nack 消息）。

返回到图 4 所示的拓扑，当 GUID 为 1 的设备中的 SE 想要向 GUID 为 3 的设备中的另一 SE 发送 HAVi 可靠消息时，GUID 为 1 的设备的消息发送系统向网桥入口（GUID 为 5 的设备）发送消息。网桥的消息发送系统查看目的地 SEID，更精确地，查看包含在此 SEID 中的 GUID，并推导出此消息是针对其远程列表中的设备的。然后，消息发送系统向其共同入口转发 HAVi 消息，其共同入口依次将 HAVi 消息发送给 GUID 为 3 的设备（参见图 5）。然后，从 GUID 为 3 的设备通过网桥向 GUID 为 1 的设备发送确认响应。

如下进行错误处理：

只是将在 HAVi-1.1 规范中已经规定的添加到对 HAVi 消息的错误处理中。事实上，位于路径上的网桥中的消息发送系统只是转发消息。

(e)事件管理器

当 SE 正在发送事件时（利用 EventManager::PostEvent API），事件管理器只将其发布在其本地群集上（利用 EventManager::ForwardEvent API）。接收到来自另一事件管理器的事件的网桥的事件管理器将此事件转发到到其共同入口。该共同入口然后将该事件发送到其群集（利用 EventManager::ForwardEvent API）等。

入口的事件管理器将事件转发到其共同入口与否的规则在于原始发布方的 GUID 是否存在于其共同入口的远程列表中。作为 ForwardEvent 消息中的参数，给出原始发布方的 GUID。这里，提醒一下 ForwardEvent API：

```
Status EventManager::ForwardEvent(
    in SEID posterSeid,
    in EventId eventId,
    in sequence<octet> eventInfo)
```

posterSeid 参数是向其本地事件管理器发布事件的原始 SE 的

SEID。包含在此 SEID 中的 GUID 给出了 SE 所驻留的设备的 GUID。入口使用此 GUID 来确定其是否转发此事件。

当网桥向远程群集转发来自非 BA 设备（可以从远程设备控制这些设备）的事件时，入口的事件管理器并不向其群集的非 BA 设备的事件管理器转发从其共同入口接收到的事件消息（即，远程事件）。

因此，事件的错误处理保持与 HAVi-1.1 中相同（参见事件管理器协议章节 5.4.5，第 144 页）。小的更新在于，每个入口用作位于其后的装置的代理。所以，事件管理器将接收来自其本地群集的事件管理器（其向之发送消息的事件管理器）的响应，且当入口接收到来自其共同入口群集的所有响应时，入口将进行响应，合并和反映这些响应。

图 6 示出了针对多群集网络上的事件发布（post）的基本处理。由 GUID 为 3 的设备的事件管理器向其群集的所有事件管理器转发从 GUID 为 3 的设备中的 SE 发布的事件。然后，一旦发布方的 GUID 在其共同入口的远程列表中，入口就向远程群集转发此事件（这就是为什么入口 6 并不将其转发给入口 5 的原因）。然后，入口将此事件转发给除非 BA 设备以外、位于其群集上的事件管理器（这是为什么设备 2 未接收到此事件的原因）。

根据变体实施例，对 PostEvent API 的“全局”参数进行如下修改。目前，其被定义为表示事件是否为 HAVi 网络本地或 HAVi 网络全局的布尔值。以如下的‘enum’结构体替代此布尔值。

```
enum EventScope {LOCAL, CLUSTER, NETWORK};
```

在优选实施例中，PostEvent API 保持不变。

(f)登记处

在 HAVi 中，由 SE 向登记处发送登记处查询请求（Registry::GetElement 或 Registry::MultipleGetElement）。基本处理是 SE 查询其本地登记处，然后，登记处将向 HAVi 网络上的所有其他登记处转发该查询。一旦登记处从远程节点接收到查询，其在搜索其自身的数据库之后，应答该查询。

这里以网桥保持此概念。接收到来自远程节点的查询的登记处将应答搜索其自身的数据库，除网桥设备中的登记处以外。基本处理保持为 SE 查询其本地登记处，登记处将向网络上的所有登记处转发该请求。稍后，对其进行详细描述。

入口的登记处自然地向其共同入口的登记处转发来自其群集的登记处的请求。但只有在请求的最初发送方的 GUID 出现在其共同入口的远程列表中时(即，其共同入口位于去往最初发送方的反向路径上)，才这样做。与先前一样，这避免了在不同的路径上向相同的目的地发送消息。如果此最初 GUID 不在其共同入口的远程列表中，则将不转发请求。对于拓扑环路，这有可能发生。在这种情况下，其共同入口将通过其群集上代理最初 GUID 的网桥来接收请求（所以，通过另一路由）。此外，网桥的登记处并不转发来自非 BA 设备的登记处的请求。这些设备对远程 GUID 一无所知，因此，将不能向远程 SEID 发送消息（对登记处的基本查询返回包括 GUID 的 SEID)。

在被请求时，则共同入口的登记处可以向其群集上的其他登记处转发请求，包括其他网桥。随即，在整个网络上发送登记处请求。

BA 设备的登记处只向其群集发送其请求，所以由登记处本身控制群集之间的登记处通信（'群集分离'）。在图 7 中，与网络管理器列表（本地和远程）一起示出了三网桥环路网络。

根据变体实施例，基本处理如下：最初登记处（设备 GUID 1）向整个网络上的所有登记处发送查询。于是，在第一群集上发送的 HAVi 消息数为 9 个（由于网络上有 9 个登记处)，每个登记处一个。在另一群集上，此数目减少，由于所有网桥并未转发所有消息。

根据优选实施例，登记处（GUID 1）只向其自身群集的所有登记处发送查询。现在，此第一群集上的 HAVi 消息数为 3 个。然后，网桥的登记处以其共同入口的群集重复此操作(但只有当最初发送方 GUID 1 出现在其共同入口的远程列表中时才如此：这就是为什么 GUID 为 7 的入口并未将其转发给 GUID 为 8 的入口的原因)。

此小示例示出了对查询的改进（以三个消息代替九个)，但对于响应出现相同的现象。利用优选实施例，入口中的登记处通过合并其

共同入口群集登记处的所有响应，创建了单一的响应。此外，在此示例中，每个设备都可以通过一个网桥到达，但当链接几个网桥时，在最初发送方附近的群集中，冗余 HAVi 消息数据变得巨大。

如下执行登记处消息处理：

利用群集分离，最初登记处查询其群集上的所有登记处。这减少了请求的总体业务量。为了使入口能够知道其是否必须向其共同入口转发该请求（根据共同入口的远程列表），HAVi 消息的源 SEID 必须使最初发送方的 SEID（如果改变此源 SEID，则因为针对网络管理器行为而选择的路由管理，环路网络中的查询将不能终止）。但，网络中的所有登记处将响应最初请求方，并且最初请求方将接收到比其发送的查询多的响应，这可能是不能被理解的。这就是为什么根据本实施例，初始请求方只接收来自其本地群集的登记处的响应的原因。

以下变体可以用于解决此问题（以 GetElement 作为示例）：

1. BA 设备的登记处知道其只向其群集发送查询，但其将接收到来自整个网络的所有登记处的响应。只针对请求进行了消息数的减少，而并未针对响应进行消息数的减少。因为并不转发来自非 BA 设备的请求，因而其能够工作。

2. 修改 Register::GetElement API。增加新的参数以包括与最初请求方的 SEID 有关的信息。API 变为：

```
Status Registry::GetElement(
    in   SEID initialRequester,
    in   SimpleQuery query,
    out sequence<SEID> seidList)
```

接收到此消息的网桥的入口根据包含在此最初请求方参数的 SEID 中的 GUID，知道其是否必须向其共同入口转发该查询。针对响应进行业务量改进。当向其远程群集转发请求时，网桥必须向 HAVi 1.1 设备发送 HAVi 1.1 消息，而向 BA 设备转发此新消息（根据每个设备的 SDD 的版本字段）。这些请求是新请求，具有网桥的源 SEID（而不再是最初请求方的 SEID）。入口将收集所有发送给其的响应（因为其是请求的源 SEID），并将其合并为一个将要发送给其最初请求方的 SEID 列表（原始上，入口从其接收到请求的设备）。

3. 在上述变体 2 中，非网桥登记处并不使用最初请求方的标识。此信息只对于网桥设备的登记处是有用的，以便确定是否向远程群集转发请求。另一变体在于：以专用于网桥设备的新方法扩展登记处的 API。网桥感知登记处将使用此针对入口登记处的调用。

```
Status Registry::ForwardGetElement(
    in   SEID initialRequester,
    in   SimpleQuery query,
    out sequence<SEID> seidList)
```

于是，被调用的网桥设备知道最初请求方的标识。非网桥设备接收正常 GetElement 调用。两个调用均包含网桥的源 SEID，而不是最初请求方的 SEID。当网桥接收到来自登记处的所有响应时，其将这些响应合并为一个 SEID 列表，并对其最初接收到的 ForwardGetElement 调用进行响应。

4. 第四变体在于避免修改登记 API。网桥原样转发 GetElement 请求，即不修改 HAVi 消息中的源 SEID。当网桥设备接收到来自其群集上的登记处（另一网桥或非网桥设备）的响应时，其不对该响应进行转发。分析该响应，并提取出 SEID 列表，以便构建其将要发送回其曾经从其接收到查询的请求方的合并 SEID 列表。当其接收到所有响应时，其可以与合并 SEID 列表一起，发送其自身的响应。

下表尝试总结了四个已提出的变体的 pros 和 cons。

1	否	否	低
2	是	是	中
3	是	是	中
4	是	否	高

表 9

优选变体是第三和第四个，因为不需要修改 GetElement API。变体 3 具有能够在网桥之间同步的优点。

图 8 给出了利用第三变体的、针对 GetElement 调用的相互作用

的示例。

最初发送方向其本地登记处发送 GetElement 请求。然后，本地登记处将此 GetElement 请求转发给其本地群集上的其他登记处。当网桥的登记处接收到此请求时，其将此请求转发给共同入口登记处（假设包含在源 SEID 中的 GUID 在此共同入口的远程列表中）。然后，将此看作新请求。向共同入口的群集上的登记处发送此新请求。向非网桥设备发送 GetElement，而向网设备发送 ForwardGetElement。如果此群集中出现其他网桥，则重复此处理。

在每个远程群集上，由网桥设备的登记处发送不同的请求，即，网桥并不是简单地将最初请求放置在远程群集上。网桥设备保持对这些请求的跟踪，以便将其群集的登记处的响应回馈给其共同入口。当网桥设备已经接收到来自其群集的登记处的所有响应时，其将所有响应合并为单一的响应（一个 SEID 列表），并将该单一的响应提供给其共同入口。然后，共同入口可以将增加了其自身的 SEID 列表的这个 SEID 列表发送给请求方登记处。如果请求方登记处在另一网桥中，则此响应将为 ForwardGetElement 响应，或者为针对与最初请求方相连的网桥的 GetElement 响应。

在图 8 的特定示例中，通过网桥设备合并 SEID 列表：

• GUID 为 6 的入口将来自 GUID 为 7 的设备的列表（E）及其自身的列表（D）回送给 GUID 为 5 的共同入口。GUID 为 5 的入口获得此列表，并加入其自身的列表（C）。结果为（C，D，E）。

• GUID 为 10 的入口将来自 GUID 为 8 的设备的列表（空）、来自 GUID 为 3 的设备的列表（空）、来自 GUID 为 4 的设备的列表（F）及其自身的列表（空）回送给 GUID 为 9 的共同入口。结果为（F）。

• GUID 为 1 的设备的登记处接收来自 GUID 为 2 的设备的响应(B)、来自 GUID 为 5 的设备的响应（C，D，E）和来自 GUID 为 9 的设备的响应（F）。其加入其自身的列表（A），并将应答回送给 SE。最终结果为（A，B，C，D，E，F）。

针对 GetElement 方法所提及的也可以应用于 MultipleGetElement 方法。以下为专用于网桥登记处的新 API：

```
Status Registry::ForwardMultipleGetElement(
    in   SEID initialRequester,
    in   ComplexQuery query,
    out sequence<SEID> seidList)
```

(g)流

已知的 HAVi 流管理器是允许建立流连接的系统软件组件。流连接使源功能组件和宿功能组件相关联（因而，关联源和宿设备），并确保所需资源的可用性。这些资源可以是信道、带宽等。图 9 示出了由 HAVi 规定的流连接模型。两个功能组件进行互连。在每个关联设备中，已经进行了功能控制模块和设备控制模块(FCM/DCM)之间的内部连接。在关联设备之间进行设备连接（图 9 中的两个完全 A/V 设备）。逻辑连接是 HAVi 级的（即，在软件组件之间），而物理连接涉及真实的设备（在逻辑级，以 DCM/FCM 所表示的那些）。

在已经建立流连接之后，可以在源和宿之间发送流。在 HAVi 中，想要创建流连接的每个应用程序应当使用其本地流管理器（即，位于相同设备中的流管理器）。

根据 HAVi 规范，以 FCM（功能组件模块）在网络中表示功能组件，而以 DCM（设备控制模块）在网络中表示设备。当客户端应用程序从其本地流管理器请求流连接时，其指示源和宿功能组件的标识。在 FcmPlug 结构中组合了提供给流管理器的信息：

- TargetID：功能组件（不是 FCM）所处的设备的 GUID，且是对设备中的组件的索引。
- 插件方向：插入或拔出。
- 插件数量：如果功能组件管理几个插件。

流管理器使用 DCM 的服务来实现内部连接（即，设备内的连接）。为了操作 DCM 模块，流管理器使用 HAVi 消息。因此，建立内部连接的方式并不依赖于介质的技术（例如，IEC61883/IEEE1394 等）。

流管理器使用其链路层的服务（例如，IEC1883 CMP 协议等）来建立设备流连接。

根据本实施例，对多群集流的处理如下：

在单群集 HAVi 网络上，为了建立流，客户端使用其本地流管理器。此本地流管理器完全负责此流。在多群集网络上，客户端本地的流管理器可能并不位于与源和/或宿设备相同的群集上。此外，其可能并不知道源和/或宿设备所使用的介质技术。因此，基本原理在于使路径上的流管理器进行合作。

对于简单的单群集流，客户端能够规定由流管理器使用的传送类型、传输格式、信道和插件。对于多群集流，认为客户端能够选择针对流将要跨越的每个群集的所有这些参数是不现实的（目标是能够具有客户端根本不了解的介质技术）。所以，客户端不得不规定带宽策略（静态或动态）和流类型（对于流惟一的）。然后，流管理器负责所有传送问题。

以流管理器 SprayOut 和 TapIn API 建立 HAVi 中的广播流。

根据本实施例，当流管理器在这些 API 上接收到本地调用，且目标设备是远程的（即，不在本地群集上）时，其将向与目标功能组件（设备）相连的最近入口的流管理器转发此调用。然后，此流管理器将执行广播连接，但此连接将只是远程群集本地的。所以，广播流不会跨越网桥，但能够远程地控制。

现在，将描述所提出的针对点到点流的 API。

为了保持与 HAVi-1.1 设备的后向兼容性，需要针对那些跨越网桥或位于远程群集上的流，定义新的流管理器方法。此后，将对其进行展示。

与已知的流管理器 API 相比，为新方法添加了下划线。

服务	通信类型	局部性	访问/针对事件：发送方（接收方）
StreamManager::FlowTo	M	本地	全部
StreamManager::MultiClusterFlowTo	M	全局	全部
StreamManager::OnThePath	M	全局	网桥中的流管理器
StreamManager::SprayOut	M	本地	全部

StreamManager::TapIn	M	本地	全部
StreamManager::Drop	M	全局	全部
StreamManager::GetLocalConnectionMap	M	全局	全部
StreamManager::GetGlobalConnectionMap	M	全局	全部
StreamManager::ForwardGetGlobalConnectionMap	M	全局	网桥中的流管理器
StreamManager::GetConnection	M	全局	全部
StreamManager::GetStream	M	全局	全部
ConnectionAdded	E	全局	流管理器(全部)
ConnectionDropped	E	全局	流管理器 (全部)
ConnectionChanged	E	全局	流管理器(全部)

表 10

(a) StreamManager::MultiClusterFlowTo

原型

```
Status StreamManager::MultiClusterFlowTo(
    in boolean dynamicBw,
    in FcmPlug source,
    in FcmPlug sink,
    in boolean anyStreamType,
    in StreamType streamType,
    out ConnectionId connId)
```

参数

- dynamicBw – 表示应当设置动态(dynamicBw 为真)还是静态(dynamicBw 为假)带宽分配。
- source – 识别源插件的 FcmPlug 结构体。
- sink – 识别宿插件的 FcmPlug 结构体。
- anyStreamType – 表示流类型是否必须为由客户端规定或由流

管理器选择的流类型。

- streamType – 流类型，如果由客户端规定的话。
- connId – 由 FlowTo 返回的 ConnectionId 值。

描述

此 API 允许客户端请求在多群集 HAVi 网络上创建流。在这种网络上，源设备和宿设备不需要位于相同的介质类型上。对于源和宿惟一必须相同的参数是流类型。可以对流类型进行转换，但是这将在转换器模块（例如，具有一种流类型的输入和另一种不同流类型的输出的转换器 FCM）中进行。由网桥进行传送类型转换。根据本实施例，连接两个不同介质技术的网桥能够将流和消息的传送类型从一种转换为另一种。

因此，根据本实施例，客户端不需要关心此多群集连接所使用的传送类型、传输格式和信道。这些将由位于流的路径上的网桥的流管理器来处理。

错误代码

- StreamManager::ESOURCE_FCM – 由 source 表示的 FCM 不存在。
- StreamManager::ESINK_FCM – 由 sink 表示的 FCM 不存在。
- StreamManager::ESOURCE_PLUG – 由 source 表示的 FCM 不包含所指定的插件。
- StreamManager::ESINK_PLUG – 由 sink 表示的 FCM 不包含所指定的插件。
- StreamManager::EUNSUP_STREAM – 连接要求了不支持的流类型。
- StreamManager::ENO_MATCH_STREAM – 插件不兼容（流类型失配）。
- StreamManager::ENO_MATCH_BW – 源带宽超出了宿所支持的带宽，或者 hint.Stype.maxBW 超出了源/宿 FCM 的支持 Stype.maxBW（带宽失配）。
- StreamManager::ENO_MATCH_SPEED – 源使用了宿不支持的

传输速度。

- StreamManager::ENO_MATCH_DIR – 插件不兼容(方向失配)。
- StreamManager::ESOURCE_BUSY – 源插件是另一个流的成员。
- StreamManager::ESINK_BUSY – 宿插件是另一个流的成员。
- StreamManager::EDEV_BUSY – 分配设备插件失败。
- StreamManager::EINSUFF_BANDWIDTH – 带宽分配失败。
- StreamManager::EINSUFF_CHANNEL – 信道分配识别。
- StreamManager::ESTATICBW – dynamicBw 为假,流类型是可变速率,但不能将源设置为静态带宽分配。
- StreamManager::ERESERVED_SOURCE – 需要建立(不重叠)所需的连接,但由于预留保护而被拒绝。
- StreamManager::ERESERVED_SINK – 保留由 sink 表示的 FCM(而且不是由做出 FlowTo 请求的软件组件保留的)。
- StreamManager::EDEV_CONN – 建立设备连接失败。
- StreamManager::ESHARE – 不能建立连接,因为源插件不可共享(而且拥有方不同于进行 FlowTo 请求的软件组件)。

(b) StreamManager::**OnThePath**

原型

```
Status StreamManager::OnThePath(
    in boolean dynamicBw,
    in FcmPlug source,
    in FcmPlug sink,
    in StreamType streamType,
    in TransportType segmentTransportType,
    in TransmissionFormat segmentTransmissionFormat,
    in Channel segmentChannel,
    in ConnectionId connId)
```

参数

- dynamicBw – 表示应当设置动态(dynamicBw 为真)还是静态(dynamicBw 为假)带宽分配。
- source – 识别源插件的 FcmPlug 结构体。
- sink – 识别宿插件的 FcmPlug 结构体。
- streamType – 连接的流类型。流类型在整个连接上是惟一的，而传送类型、传输格式和信道可以不同（尤其是在跨越不同介质技术时）。
- segmentTransportType, segmentTransmissionFormat, segmentChannel – 当前段的传送类型、传输格式和信道的值，即附属于接收到此调用的群集和入口。
- connId – 由最初流管理器分配的 ConnectionId 值。

描述

此 API 用在网桥中的流管理器之间，以构建跨越至少一个群集的连接。从原始的 MultiClusterFlowTo 方法调用中复制 dynamicBw, source 和 sink 参数。其由入口的流管理器使用，以确定需要向路径上的哪些入口发送所述流。

streamType 参数标识了流的类型。此类型对于整个流是惟一的，其并不受到为了携带所述流而使用的传送的影响。改变其流类型的流（例如，从 DV 到 MPEG2）将通过转换器（例如，FCM 转换器），事实上在转换器处两种流运行，FCM 转换器是第一种流的宿，且是转换后的流的源。

“段”参数（segmentTransportType, segmentTransmissionFormat 和 segmentChannel）标识了用在流的当前段（即，目标流管理器本地）上的参数。这对于接收到此调用的入口的流管理器获得与建立在其段上的连接有关的所有信息，并内部地将其连接到其共同入口是有用的。

由最初流管理器填充 connId 参数，并由流路径上的入口流管理器使用，以将其段流“附加”到多群集流上。

错误代码

- StreamManager::EUNSUP_TRANSPORT – 连接要求了不支持

的流类型。

- StreamManager::EUNSUP_STREAM – 连接要求了不支持的流类型。
- StreamManager::ENO_MATCH_FMT – 插件不兼容(传输格式失配)
- StreamManager::ENO_MATCH_SPEED – 源使用了宿不支持的传输速度。
- StreamManager::ENO_MATCH_TRANSPORT – 插件不兼容(传送类型失配)
- StreamManager::ENO_MATCH_DIR – 插件不兼容(方向失配)
- StreamManager::ESOURCE_BUSY – 源插件是另一个流的成员
- StreamManager::ESINK_BUSY – 宿插件是另一个流的成员
- StreamManager::EDEV_BUSY – 分配设备插件失败
- StreamManager::EINSUFF_BANDWIDTH – 带宽分配失败
- StreamManager::EINSUFF_CHANNEL – 信道分配失败
- StreamManager::EDEV_CONN – 建立设备连接失败

用于建立多群集流连接的处理如下：

由客户端本地并属于客户端的流管理器发起多群集流，与HAVi-1.1中一样（“属于”这里的意思是在其本地连接映射中）。将此流管理器称为“最初”流管理器。其向位于去往宿设备的路径上的目标功能组件（设备）的最近入口的流管理器转发此调用。结果，入口的流管理器能够接收本地调用（通过本地客户端）和远程调用（通过远程流管理器）。

此入口的流管理器负责进行与目标源功能组件在群集上的连接，并且其共同入口的流管理器负责流路径上的下一群集上的连接。如果流跨越其他网桥，则共同入口流管理器将向流路径上的下一网桥的流管理器发送HAVi消息，具有所有必须信息，从而此下一网桥流管理器能够向其共同入口内部转发该连接，所述共同入口将进行其群集上的连接，等等。在每一个段上，适当的流管理器将调用DCM的API来执行传送参数的选择，这些DCM可以是源和宿设备的DCM，但也可以是

该路径上的网桥的 DCM。

所以，处理为：

1. 客户端调用被称为最初流管理器的其本地流管理器的 MultiClusterFlowTo API。

2. 最初流管理器查看源功能组件（设备，而非 FCM），并向去往宿的路径上、与源群集相连的第一入口转发调用。此入口被称为初级入口。有两种可能性，但在行为上没有区别：

o 源功能组件在远程群集上，向与之相连的最近入口的流管理器转发 MultiClusterFlowTo 调用（利用有网络管理器提供的 NetDeviceInfo 结构中的 nearestPortalGuid 得知）。

o 源功能组件在本地群集上，向去往宿功能组件设备的路径上的本地群集入口的流管理器转发 MultiClusterFlowTo 调用（宿设备的 GUID 在此入口的远程列表上，而不在本地群集的任意其他入口的远程列表上）。

3. 初级入口上的流管理器利用流的结束点，对流类型执行所有 DCM 和 FCM HAVi 操作。此外，其对针对此群集的传送执行所有 HAVi 操作。

4. 然后，流管理器负责建立第一段上的流，即源设备和初级入口之间。

5. 然后，转发给其共同入口的流管理器。

6. 此流管理器负责建立第二（或下一）段上的流。其可以去往最终宿设备或去往另一入口。由此流处理器完全确定和处理流在此段上的传送（包括 HAVi 操作和 DCM/FCM 调用）。

7. 如果另一入口在此路径上，流管理器调用驻留在去往宿设备的路径上的下一入口中的流管理器的 StreamManager::OnThePath API。进行到步骤 5。

在特定的群集上，连接的构建可能涉及源和宿端点（入口或设备）的流管理器。

将通过图 10 描述此处理示意图。

对于多群集流连接去除，不需要新的 API。任何想要丢弃运行流

的 SE 将调用拥有流的流管理器的丢弃 API。如果是多群集流，此最初流管理器将把此调用转发给流路径上的第一入口，并且去除处理与构建处理相同，基于每个流管理器内部保存为针对其所负责的群集上的连接的标识符的 ConnectionId。

利用这种解决方案，HAVi-1.1 设备不能丢弃由远程流管理器建立的流，因为它从未看到过该流。其能够丢弃由其群集上的流管理器所拥有的多群集流（即使它并不了解此流的源和/或宿）。

作为变体，可以不从源到宿而是从宿到源地进行连接建立。

在多群集流上，仍然对动态带宽分配进行管理。如果在 MultiClusterFlowTo API 中将 dynamicBw 布尔参数设置为真，则 DCM 源负责重新分配其群集上的资源。然后，其发送 BandwidthRequirementChanged 事件。由负责路径上的下一段的流管理器捕获此事件。如果需要，此流管理器重新分配带宽。如果在 MultiClusterFlowTo API 中将 dynamicBw 布尔参数设置为假，则针对流的所需带宽的变化可能会使流进入故障模式(如 HAVi-1.1 中所述)。

如下进行流连接错误处理：当在构建处理期间，不能在一个段上建立连接时，流管理器将在其 Status 返回值中发送回具有故障原样的 OnThePath 消息。然后，逐段去除连接，直到最初流管理器，最初流管理器警告其客户端，或者采用可能可用的“代替路径”（如下）。

当在一段上切断了现有连接时（因为总线复位，缺少资源等），此段上负责此连接的流管理器发送由最初流管理器捕获的 MultiClusterConnectionDropped 事件，最初流管理器负责丢弃流，或者试图通过“代替路径”保持其有效。可以通过 OnThePath API 的 connId 参数获得最初流管理器。此 connId 参数能够对 mgr 参数进行访问，mgr 参数是最初流管理器的 SEID。

封装对转译

根据本实施例，如果流从基于介质技术 A 的群集跨越基于介质技术 B 的群集并回到基于介质技术 A 的群集，B 型群集上的流管理器决定不转译流的传送类型（例如，IP 上的 1394）。然后，将在群集 B 上

对流进行封装。出于性能的原因，这可能是有用的。但，一旦在群集 B 上对此流增加宿设备，则将转译流，从而使介质技术 B 上的再现方（renderer）能够显示该流。所以，将针对群集 B，进行 A->B 转译，然后，针对目标 A 型群集进行 B->A 转译。

利用所谓的连接映射，流管理器可以提供运行在 HAVi 网络上的所有 HAVi 流列表。利用 GetGlobalConnectionMap API 来完成。其以类似于 Registry::GetElement 的方式工作。与前面一样，由于网络管理器所定义的环路解决处理，需要新的参数将此查询转发给其他群集的流管理器，以便减少业务量。所提出的 API 为：

```
Status StreamManager::ForwardGetGlobalConnectionMap(
        in SEID initialRequester,
        out sequence<Connection> list)
```

initialRequester 参数使入口能够知道其是否必须向其共同入口转发此请求。由入口流管理器收集每个流管理器的本地连接映射，并最终将其发送回最初请求流管理器。

根据本实施例，从其群集上的设备接收 GetLocalConnectionMap 的网桥的流管理器根据从其 SEID 得出的调用方标识而进行不同的操作：

- 调用方不是流管理器。这表示 SE 想要知道其本地连接映射。在应答中只发送本地连接映射。
- 调用方是 HAVi-1.1 设备（即，非网桥感知）中的流管理器。同样，在应答中只发送本地连接映射。
- 调用方是 BA 设备中的流管理器。向共同入口转发请求（如果满足转发规则），并且共同入口流管理器将向其群集的所有流管理器发送 GetLocalConnectionMap，并向与其群集相连的其他入口的流管理器发送 ForwardGetGlobalConnectionMap。

此外，在 Connection 数据结构中进行了小修改，以处理新的多群集连接。在所枚举的 ConnectionType 中增加新条目：

```
enum ConnectionType {FLOW, SPRAY, TAP,
MULTI_CLUSTER_FLOW};
```

并且，在 MULTI_CLUSTER_FLOW 连接类型的情况下，将根据源设备，设置 Connection 结构的 transmissionFormat 和 channel 参数(所以，事实上，只是反映了源和路径上的第一入口之间的第一段上的流)。

将研究以从 connectionId 结构中复制的 segmentId 参数（标识负责特定群集上的流的流管理器的 mgr 字段）来标识每个段上的连接的需要。

根据本实施例，与在环路解决处理中所定义的主路径相比，在特定的条件下提供代替路径。

在由于在路径上的一个群集上缺乏资源而不能建立流，并假定源和宿之间的另一路由能够不通过此群集的情况下，流管理器和网络管理器可以决定将此流重新路由到代替路径上，以避免业务量拥堵的群集。这可以应用于与原始路由具有相同跳数的路由，但也可以应用于具有较高跳数的路由。在这种情况下，网络管理器必须内部保持跟踪，使其能够到达目前处于其他入口的远程列表上的设备，从而能够选择正确的路径。

图 11 示出了使用代替路径的示例。右侧群集 1101 已经有效，利用了已经不幸地几乎预留了该群集的所有资源（带宽或信道或二者皆有）的流。HAVi 应用程序想要构建设备 3 和设备 4 之间的流。利用网络管理器中的基本远程 GUID 列表，这将不能工作，因为将通过群集 1101 来建立流，所以其将失败。一旦得知失败，流管理器决定针对此流使用代替路径，通过具有携带其的资源的左侧群集。甚至可以通过其共同入口 14，将流发送到设备 13 上。

在这种代替路径决定的情况下，使用以下处理：

1. 在构建连接期间，流建立在段（群集）上是不可能的。
2. 警告群集之前的网桥（针对 OnThePath 调用返回的错误）。
3. 网桥检查是否存在另一路径。
4. 如果未找到，将错误返回针对 OnThePath 调用之前的网桥。
5. 转到步骤 3。[3]

(h)资源管理器

资源管理器应当不受网桥的影响。

II]HAVi 网桥感知设备

图 12 示出了根据本实施例的网桥感知设备（在以下的描述中称为 BA 设备）的内部软件体系结构。BA 设备包括 HAVi 1.1 软件组件，加上 Sdd 管理器和网络管理器。

- Sdd 管理器

如已经在专用于网桥设备的章节中所描述的那样，BA 设备的 Sdd 管理器将负责获得整个 HAVi 网络上的任意设备的 SDD 数据。这将通过访问远程 Sdd 管理器或执行本地低级调用来完成。

- CMM

在针对 HAVi-BA 设备的 CMM 中基本上没有变化。CMM 负责实现对低级本地群集的访问。所以，CMM 仍然提供 GetGuidList API，返回本地群集上的所有 GUID 的列表。并且其提供了在此群集上发送/接收低级消息的方式。实际上，在 HAVi-1.1 文献中对 CMM1394 作出了规定。

网络管理器

BA 设备的网络管理器如下进行操作：

- 在群集重新配置之后，其执行本地发现。
- 如果其检测到新网桥，其请求新网桥各自的远程列表。
- 当接收到 ENOT_READY 错误响应时，其等待一些时间，以接收 RemoteNetworkUpdated 事件。
- 如果在特定时间量之后，该事件仍未到来，其可以再次试图获得未作出应答的入口的远程列表。
- 当其得到来自入口的所有响应时，其对列表进行比较，并构建其新列表，并填充改变、离开和新加入字段。
- 当完成其网络设备列表时，其向其客户端发送 NetworkUpdated 事件。
- 在其构建其新列表的时间期间，以 ENOT_READY 错误应答任何

请求获得网络设备列表的客户端。

III]新 HAVi 值

除 HAVi 1.1 中现有的那些以外的新 HAVi 软件组件类型如下。

HAVi 软件组件类型	ATT_SE_TYPE 值	信任	系统元素
SDD_MANAGER	0x0000 0007	是	是
NETWORK_DEVICE_MANAGER	0x0000 0008	是	是

表 11

根据本实施例的 HAVi SEID 如下：

HAVi 软件组件类型	软件组件句柄
SDD_MANAGER	0x0007
NETWORK_DEVICE_MANAGER	0x0008
COMMUNICATION_MEDIA_MANAGER_IP	0x0009

表 12

HAVi API 代码如下。

HAVi API 名称	API 代码
SddManager	0x0017
NetworkManager	0x0018
Cmmlp	0x0019

表 13

根据本实施例，针对登记处、流管理器、Sdd 管理器和 CMMIP 的额外 HAVi 操作代码如下：

HAVi 消息	API 代码	操作 ID
Registry::ForwardGetElement	0x0003	0x05
Registry::ForwardMultipleGetElement	0x0003	0x06
StreamManager::MultiClusterFlowTo	0x0008	0x08
StreamManager::OnThePath	0x0008	0x09
StreamManager::ForwardGetGlobalConnectionMap	0x0008	0x0a
SddManager::GetSddData	0x0017	0x00
NetworkManager::GetNetDeviceList	0x0018	0x00
NetworkManager::GetNetDeviceInfo	0x0018	0x01
NetworkManager::GetRemoteDeviceList	0x0018	0x02
CmmIp::GetGuidList	0x0019	0x00
CmmIp::GetIpAddress	0x0019	0x01
CmmIp::GetGuid	0x0019	0x02
CmmIp::Send	0x0019	0x03
CmmIp::EnrollIndication	0x0019	0x04
CmmIp::DropIndication	0x0019	0x05

表 14

根据本实施例，针对 Sdd 管理器和网络管理器的 HAVi 错误代码如下：

HAVi 错误名称	API 代码	错误代码
SddManager::ENOT_READY	0x0017	0x0080
SddManager::EUNKNOWN_GUID	0x0017	0x0081
SddManager::ELAV	0x0017	0x0082
NetworkManager::ENOT_READY	0x0018	0x0080

NetworkManager::EUNKNOWN_GUID	0x0018	0x0081
Cmmlp::ENOT_READY	0x0019	0x0080
Cmmlp::EUNKNOWN_GUID	0x0019	0x0081
Cmmlp::EUNKNOWN_IP_ADDRESS	0x0019	0x0082
Cmmlp::ESIZE	0x0019	0x0083
Cmmlp::ENOT_FOUND	0x0019	0x0084

表 15

根据本实施例的 HAVi 系统事件类型如下：

HAVi 系统事件类型	由...发布	分布	基本事件号码
MultiClusterConnection Dropped	流管理器	全局	0x001c
SddDataChanged	SDD 管理器	全局	0x0025
NetworkUpdated	网络管理器	本地	0x0026
RemoteNetworkUpdated	网络管理器	全局	0x0027
RemoteNetworkChanged	网络管理器	全局	0x0028
NewDevices	通信介质管理器 IP	本地	0x0029
GoneDevices	通信介质管理器 IP	本地	0x002a
ChangedDevices	通信介质管理器 IP	本地	0x002b
GuidListReady	通信介质管理器 IP	本地	0x002c
ProxyGuidCreated	通信介质管理器 IP	全局	0x002d

IV]发现场景（Discovery scenarios）

（a）不具有环路的网络

图 18 示出了网络管理器在完全网络构建处理期间的相互作用。首先关闭所有设备，然后同时打开，所以针对每个设备，并行地进行

发现。针对网桥网络管理器，对本地和远程列表构成进行描述。

在图 13 中，示出了第一本地发现处理，即 IEEE1394 群集上的拓扑构建和 IP 群集上的组播通告。在第一步结束时，网络管理器在其内部表中具有其本地设备，如图 13 所示。

然后，网桥设备检查其网络管理器是具有完全非远程列表还是不完全非远程列表。非远程列表包括本地列表加上与群集相连的其他入口的所有远程列表。如果在一个入口中存在这种完全列表，则将其赋予另一侧（共同入口），其将认为其远程列表完全。在图 14 中，网桥 AB 的 GUID 5 的网络管理器发觉其是此设备上的惟一网桥，所以网桥可以更新 GUID 6 的远程列表，使其变为（1，2，5）。对于网桥设备 BC 也是如此，GUID 7 的网络管理器将其远程列表更新为（3，4，8）。

在图 15 中，网桥设备的网络管理器彼此间请求设备的远程列表。网桥设备的网络管理器可能会迭代地更新其远程设备列表。具体地，针对入口 6 和入口 7 的非远程列表是完整的，一旦对请求做出应答，则可以更新入口 5 和 8 的远程列表。

在图 16 中，BA 设备的网络管理器向与其本地群集相连的每个网桥设备请求远程设备列表，并构建其全局网络列表。

在图 17 中，客户端 SE 可以向其本地网络管理器请求整个网络上的设备的全局列表。

（b）在不具有环路的网络中增加新设备

图 18 示出了开始点，即，不具有环路的现有网络。在网桥网络管理器中只示出了远程列表。

添加 GUID 为 9 的新设备。通过本地发现装置（selfID、组播、…），在其所相连的群集上检测此设备。一旦检测到，则在与之相连的网桥的共同入口的网络管理器中更新此 GUID。如图 19 所示，共同入口 7 利用新连接的 GUID 9 更新其远程列表。

然后，更新后的入口向其他网络管理器（在 BA 设备中以及在网桥中）发出 RemoteNetworkUpdated 事件。与此入口相连的网桥捕获此事件，并更新其自身的共同入口远程列表。在图 20 中，入口 6 捕获来

自入口 7 的事件，并更新其共同入口 5。

然后，对整个网络进行更新。现在，GUID 9 在所有群集上均是已知的。

（c）具有环路的网络

如前所述，大约同时打开如图 21 所示的网络的所有设备。第一步仍然是本地发现处理，创建本地列表。

在此结构中，没有入口可以具有将其作为有效远程列表赋予其共同入口的完全非远程列表。所有入口均发觉其在其群集上不是单独的，所以其在更新其自己的共同入口的远程列表之前，首先向其他入口请求它们的远程列表（图 22）。并且由于在此拓扑中没有叶节点，每个入口都将等待其他入口。

由于入口不能应答 GetRemoteDeviceList 查询，其发送 ENOT_READY 响应错误。当入口中的网络管理器接收到这种错误时，其知道与其群集相连的入口并未完成对其远程列表的更新（例如，其正在等待与其共同入口相连的其他入口：对于非常长的单线路网络，这将有可能发生）。

根据本实施例，入口与其共同入口就不完全远程列表进行通信。然后，共同入口以此不完全信息对其远程列表进行更新，如图 23 所示。

于是，入口利用 RemoteNetworkChanged 事件发出此不完全远程列表，如图 24 所示。此事件仅针对网桥的网络管理器，并清楚地指出列表并未处于稳定状态（尽管 RemoteNetworkUpdated 是最终稳定列表）。

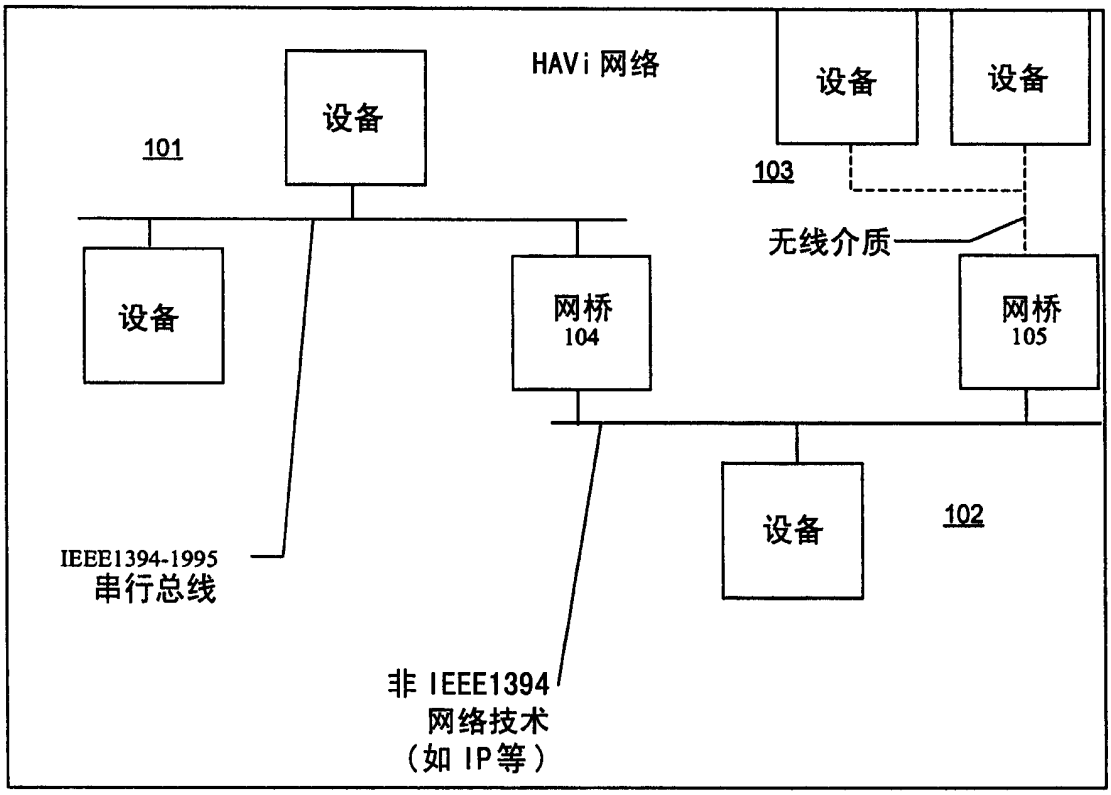

HAVi 网络
设备
设备
设备
101
103
无线介质
设备
网桥
104
网桥
105
设备
102
IEEE1394-1995
串行总线
非 IEEE1394
网络技术
（如 IP等）

图 1

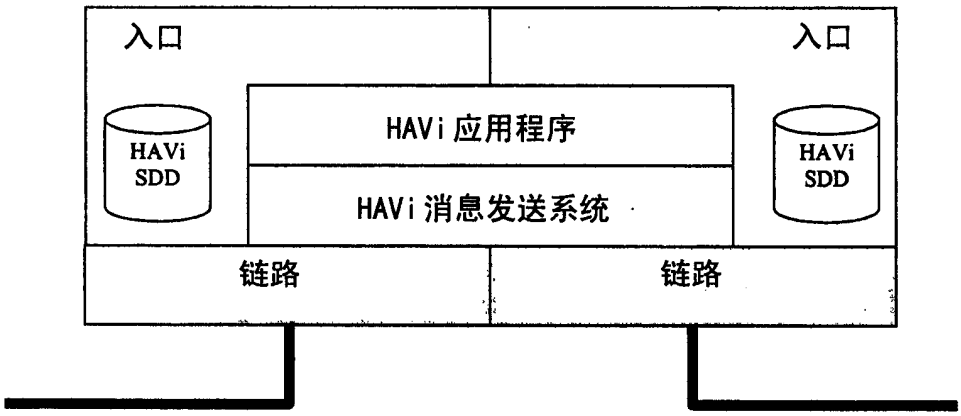

入口
入口
HAVi
SDD
HAVi 应用程序
HAVi 消息发送系统
HAVi
SDD
链路
链路

图 2

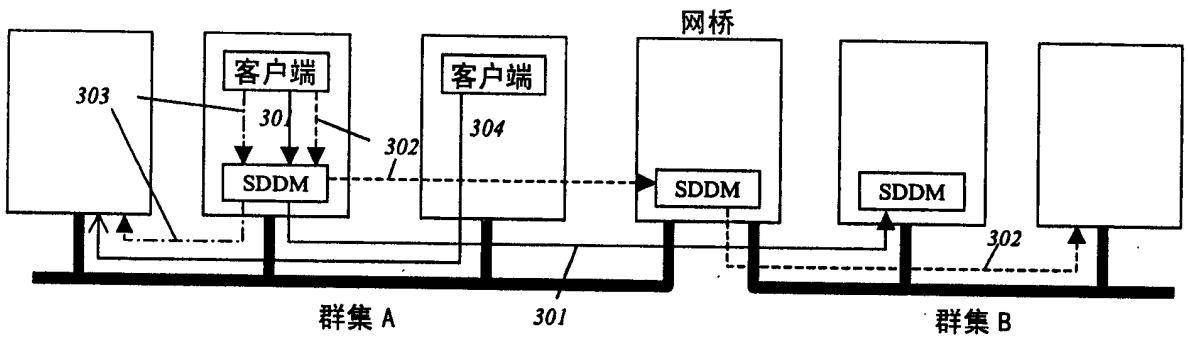

网桥
客户端
客户端
303
301
302
304
SDDM
SDDM
SDDM
302
群集 A
301
群集 B

图 3

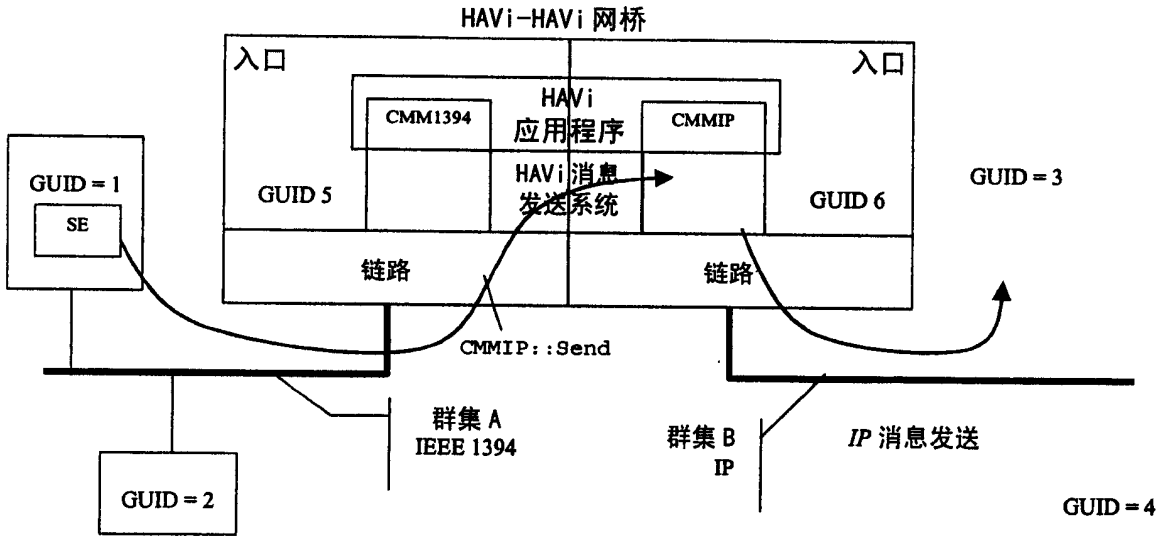

HAVi-HAVi 网桥
入口
入口
CMM1394
HAVi
应用程序
CMMIP
HAVi 消息
发送系统
GUID = 1
SE
GUID 5
GUID 6
GUID = 3
链路
链路
CMMIP::Send
群集 A
IEEE 1394
群集 B
IP
IP 消息发送
GUID = 2
GUID = 4

图 4

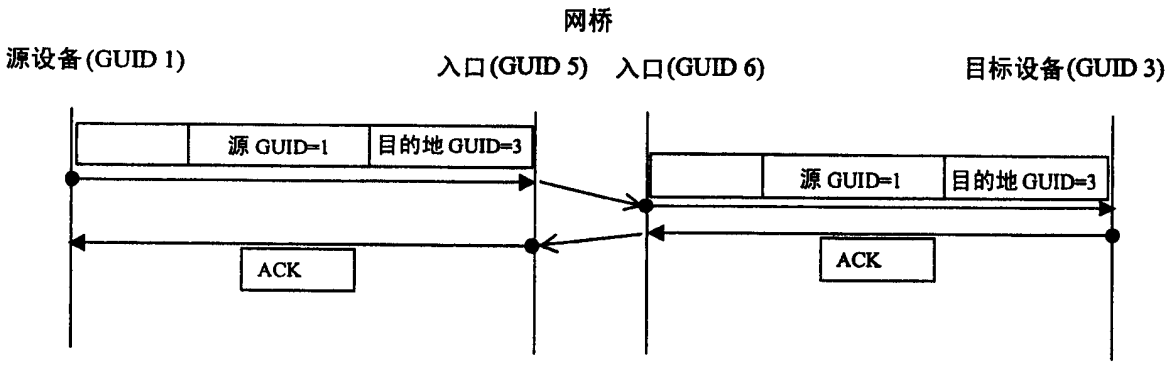

网桥
源设备(GUID 1)
入口(GUID 5)
入口(GUID 6)
目标设备(GUID 3)
源 GUID=1
目的地 GUID=3
源 GUID=1
目的地 GUID=3
ACK
ACK

图 5

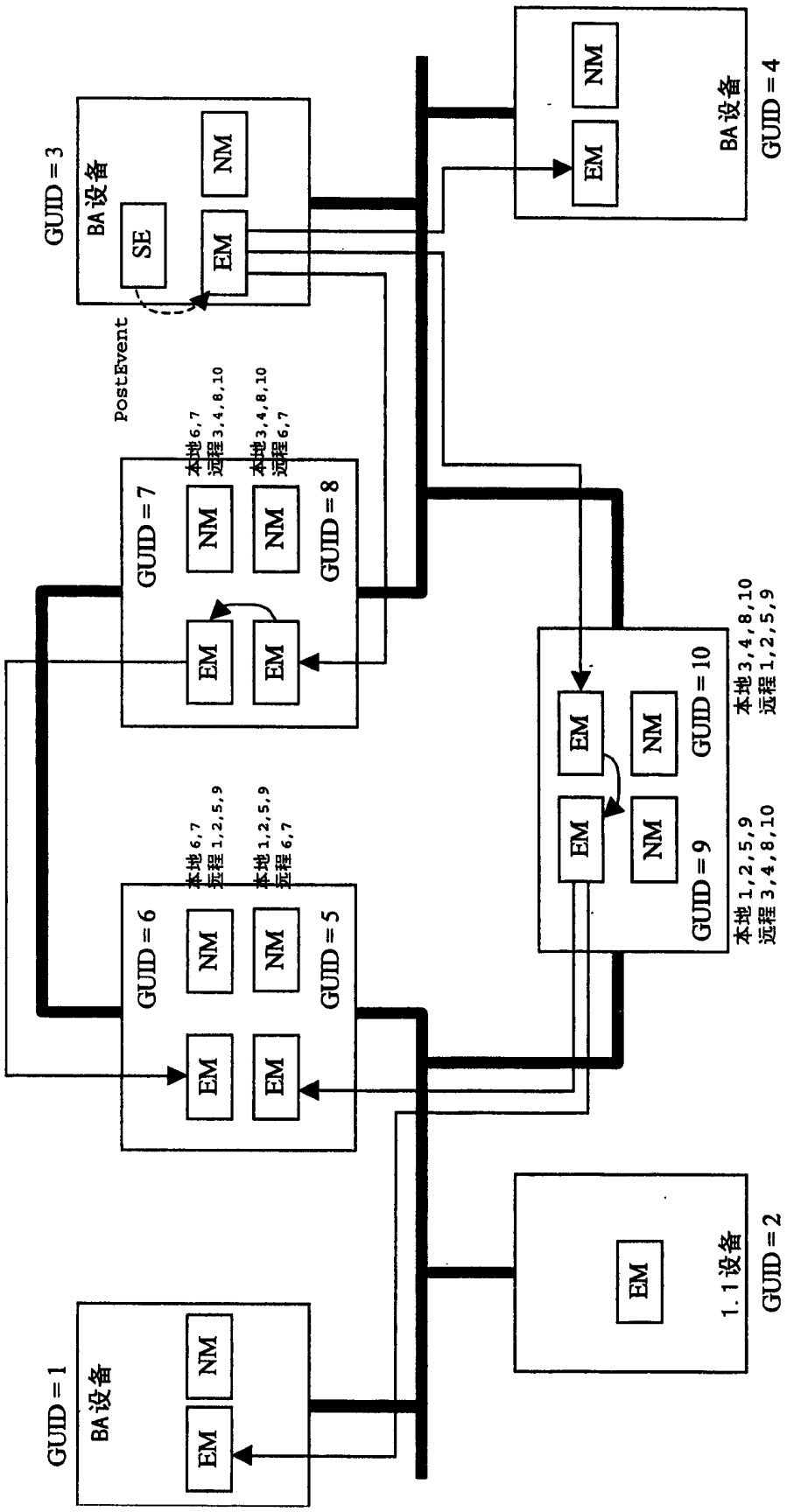

GUID = 1
BA设备
EM
NM
GUID = 2
1.1设备
EM
GUID = 3
BA设备
SE
EM
NM
PostEvent
GUID = 4
BA设备
EM
NM
GUID = 5
GUID = 6
GUID = 7
GUID = 8
GUID = 9
GUID = 10
本地 6,7
远程 1,2,5,9
本地 1,2,5,9
远程 6,7
本地 6,7
远程 3,4,8,10
本地 3,4,8,10
远程 6,7
本地 1,2,5,9
远程 3,4,8,10
本地 3,4,8,10
远程 1,2,5,9
ForwardEvent

图 6

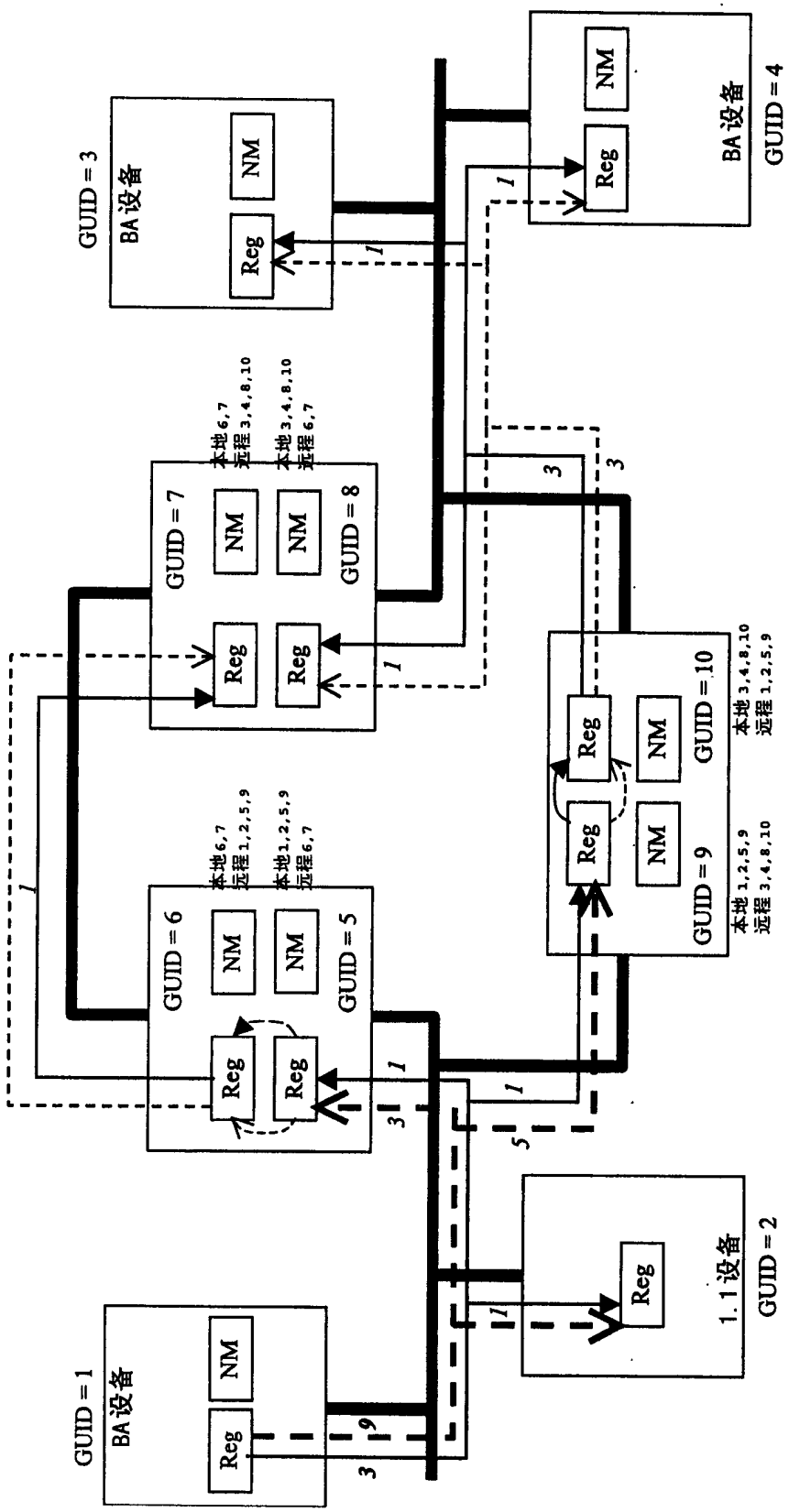

GUID = 1
BA设备
NM
Reg
GUID = 2
1.1设备
Reg
GUID = 3
BA设备
NM
Reg
GUID = 4
BA设备
NM
Reg
GUID = 5
NM
NM
Reg
Reg
本地1,2,5,9
远程6,7
GUID = 6
NM
NM
Reg
Reg
本地6,7
远程1,2,5,9
GUID = 7
NM
NM
Reg
Reg
本地6,7
远程3,4,8,10
GUID = 8
NM
NM
Reg
Reg
本地3,4,8,10
远程6,7
GUID = 9
NM
NM
Reg
Reg
本地1,2,5,9
远程3,4,8,10
GUID = 10
NM
Reg
Reg
本地3,4,8,10
远程1,2,5,9
基本处理
优化处理

图 7

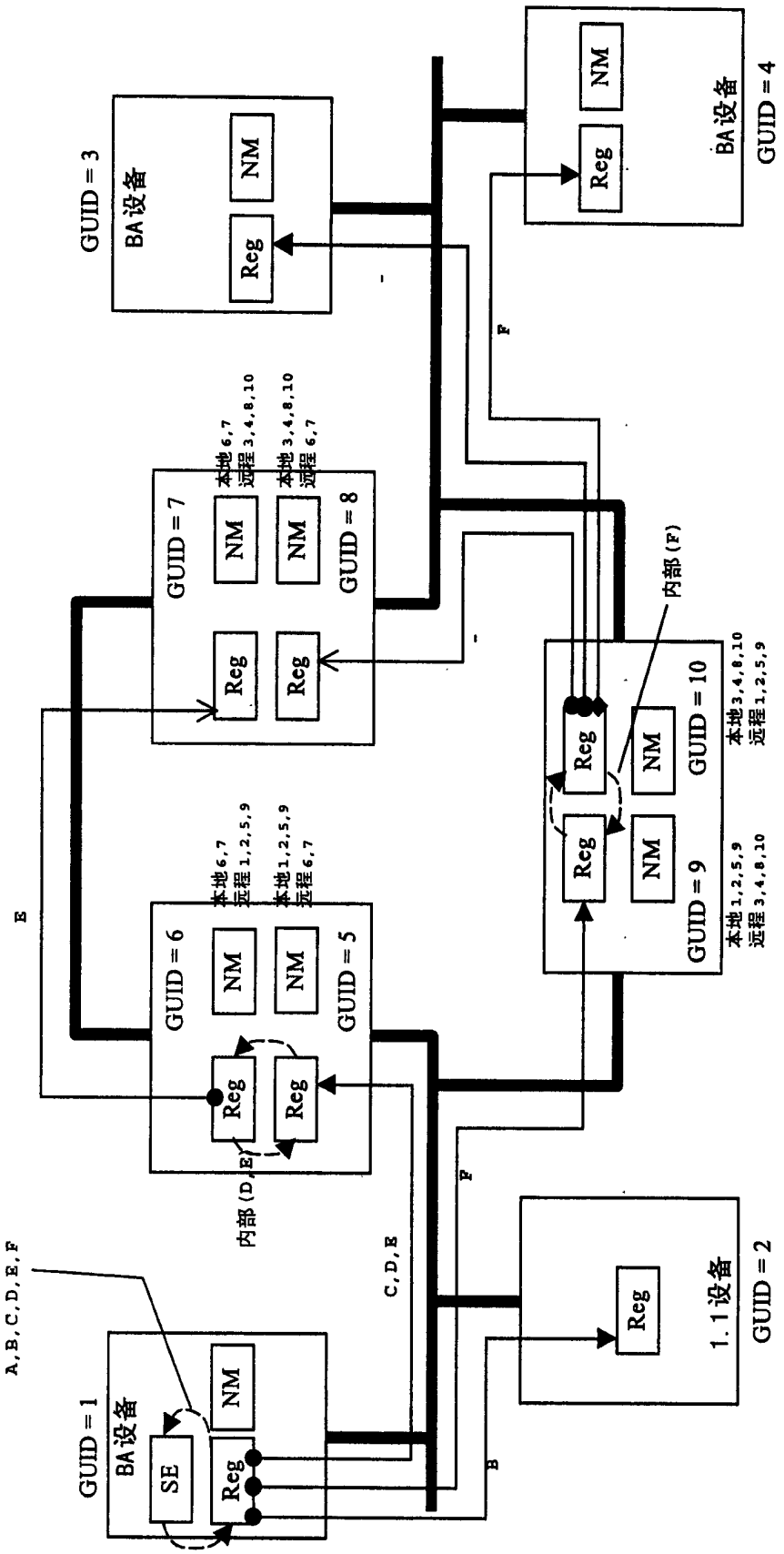

GUID = 1
BA设备
SE
NM
Reg
A,B,C,D,E,F
GUID = 2
1.1设备
Reg
GUID = 3
BA设备
Reg
NM
GUID = 4
BA设备
Reg
NM
GUID = 5
GUID = 6
GUID = 7
GUID = 8
GUID = 9
GUID = 10
本地 6,7
远程 1,2,5,9
本地 1,2,5,9
远程 6,7
本地 6,7
远程 3,4,8,10
本地 3,4,8,10
远程 6,7
本地 1,2,5,9
远程 3,4,8,10
本地 3,4,8,10
远程 1,2,5,9
内部(D,E)
内部(F)
C,D,E
E
F
B
F
X
GetElement: 箭头为请求，圆圈为具有 SEID列表的响应
X
ForwardGetElement: 箭头为请求，圆圈为具有 SEID列表的响应

图 8

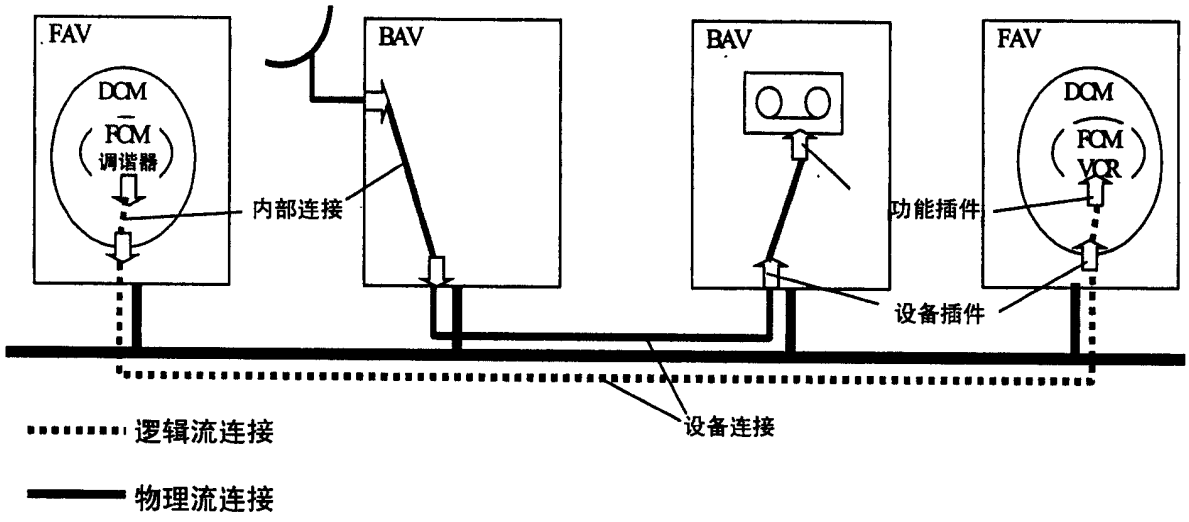

FAV
DCM
FCM
调谐器
BAV
内部连接
BAV
FAV
DCM
FCM
VCR
功能插件
设备插件
设备连接
逻辑流连接
物理流连接

图 9

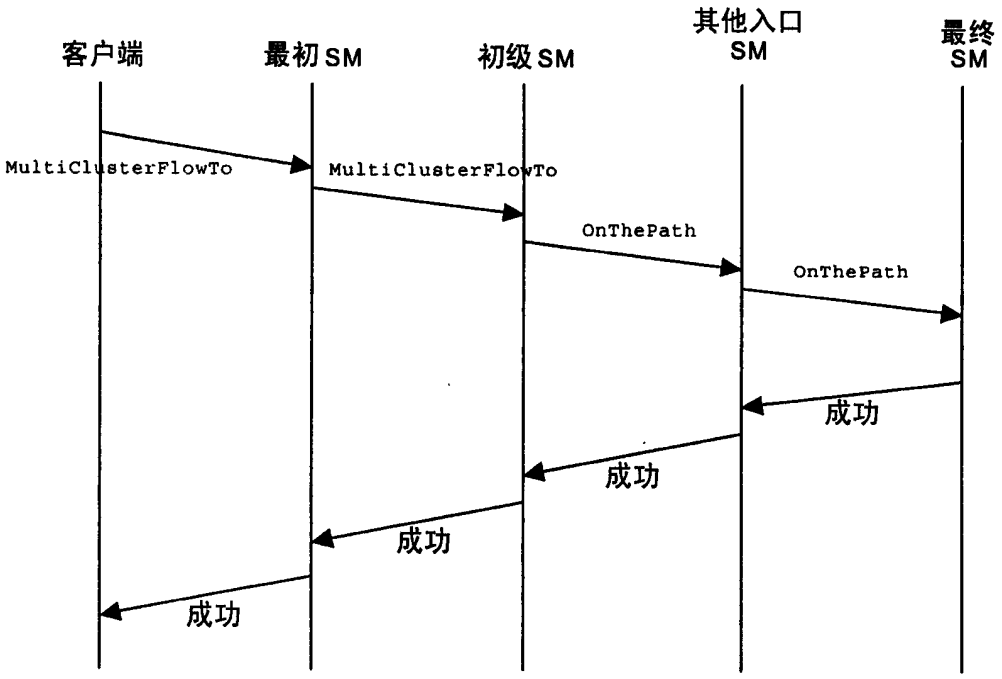

客户端
最初SM
初级SM
其他入口 SM
最终 SM
MultiClusterFlowTo
MultiClusterFlowTo
OnThePath
OnThePath
成功
成功
成功
成功

图 10

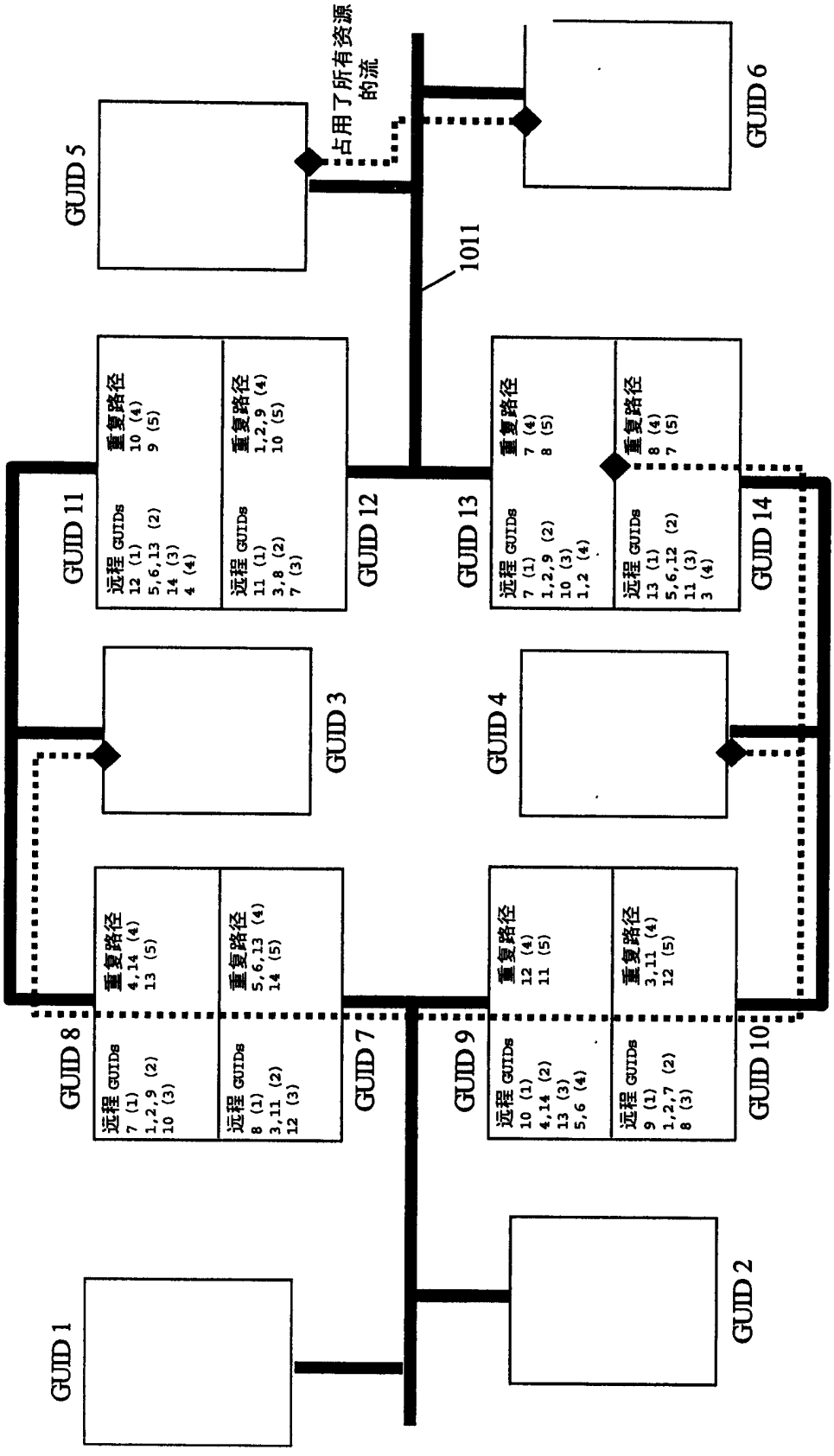

GUID 1
GUID 2
GUID 3
GUID 4
GUID 5
GUID 6
GUID 7
GUID 8
GUID 9
GUID 10
GUID 11
GUID 12
GUID 13
GUID 14
1011
占用了所有资源的流
远程 GUIDs
重复路径

图 11

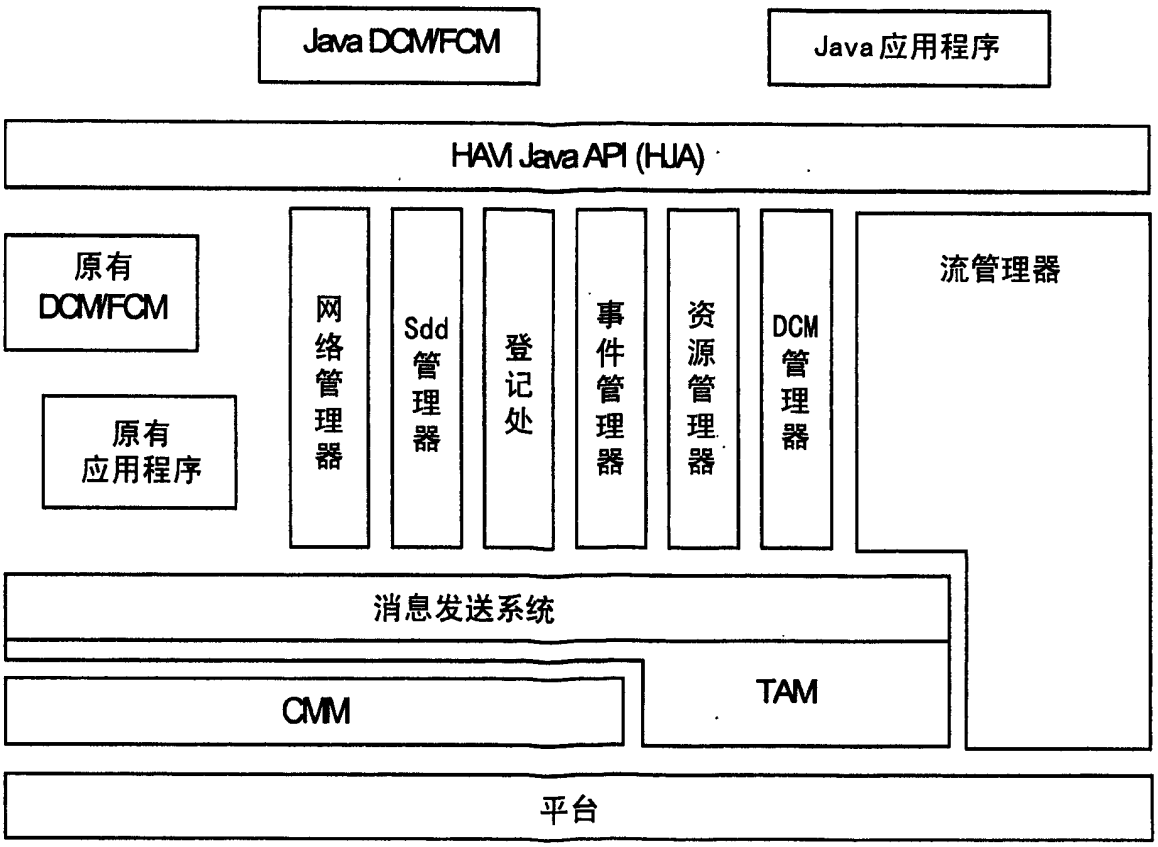

Java DCM/FCM
Java 应用程序
HAVi Java API (HJA)
原有 DCM/FCM
原有 应用程序
网络管理器
Sdd 管理器
登记处
事件管理器
资源管理器
DCM 管理器
流管理器
消息发送系统
CMM
TAM
平台

图 12

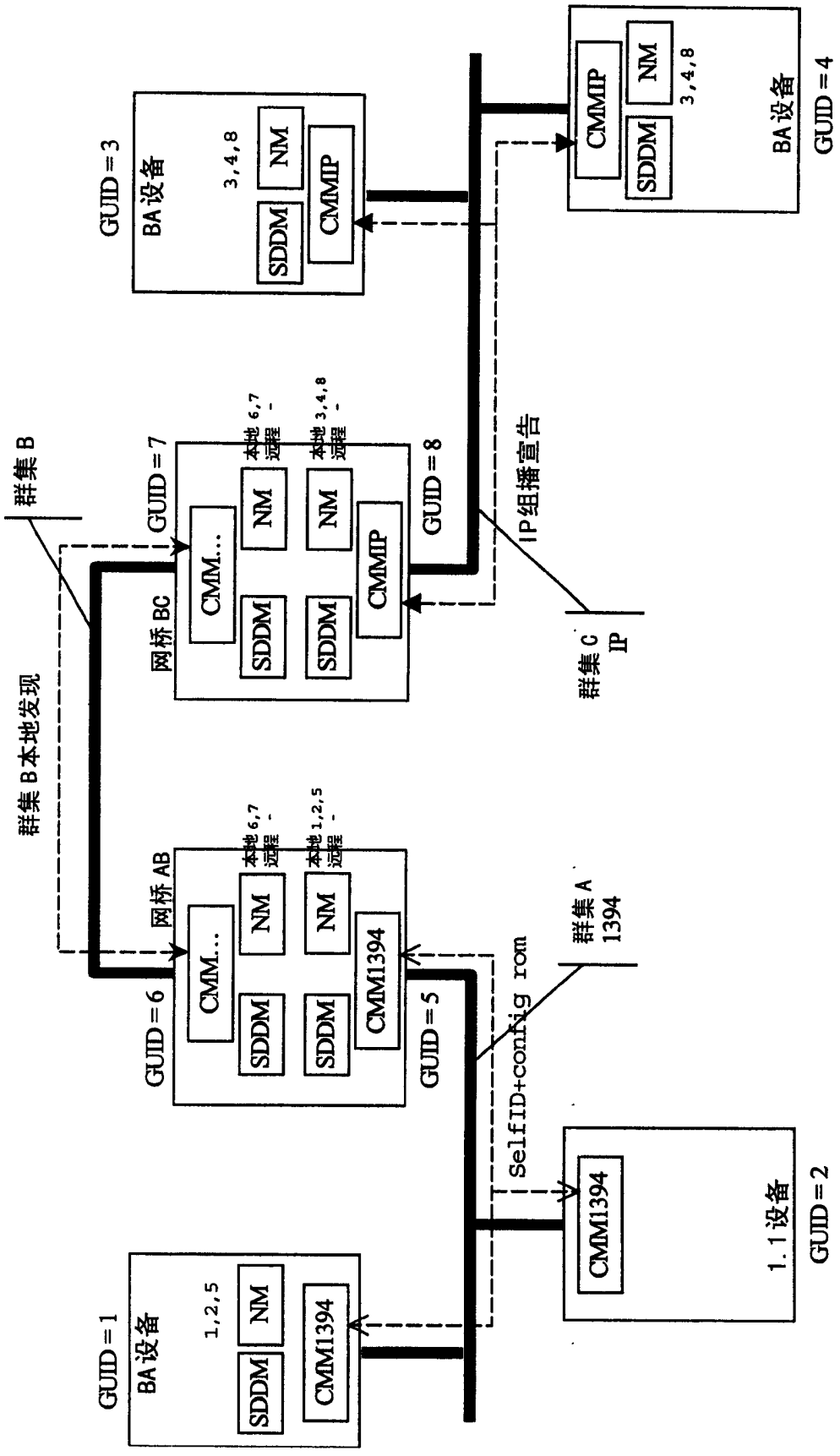

群集 B
群集 B本地发现
网桥 BC
网桥 AB
GUID = 1
BA设备
1,2,5
SDDM
NM
CMM1394
GUID = 3
BA设备
3,4,8
SDDM
NM
CMMIP
GUID = 4
BA设备
3,4,8
SDDM
NM
CMMIP
GUID = 6
CMM...
SDDM
NM
本地 6,7
远程 -
SDDM
NM
本地 1,2,5
远程 -
CMM1394
GUID = 7
CMM...
SDDM
NM
本地 6,7
远程 -
SDDM
NM
本地 3,4,8
远程 -
CMMIP
GUID = 8
GUID = 5
SelfID+config rom
群集 A
1394
IP组播宣告
群集 C
IP
CMM1394
1.1设备
GUID = 2

图 13

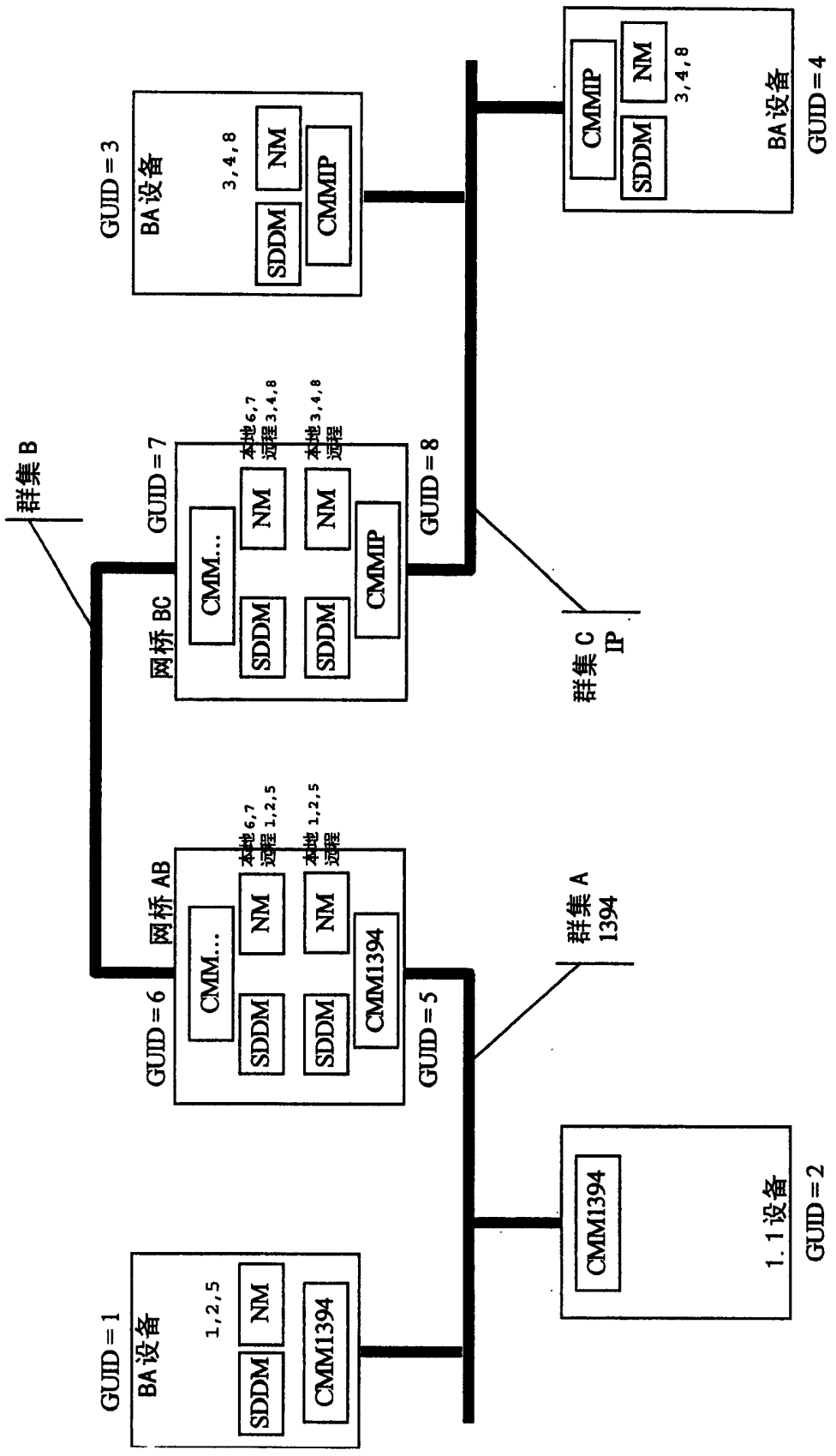

GUID = 1
BA 设备
1,2,5
SDDM
NM
CMM1394
GUID = 2
1.1 设备
CMM1394
GUID = 3
BA 设备
3,4,8
SDDM
NM
CMMIP
GUID = 4
BA 设备
3,4,8
SDDM
NM
CMMIP
GUID = 5
GUID = 6
网桥 AB
CMM...
SDDM
NM
本地 6,7
远程 1,2,5
SDDM
NM
本地 1,2,5
远程
CMM1394
GUID = 7
GUID = 8
网桥 BC
CMM...
SDDM
NM
本地 6,7
远程 3,4,8
SDDM
NM
本地 3,4,8
远程
CMMIP
群集 A
1394
群集 B
群集 C
IP

图 14

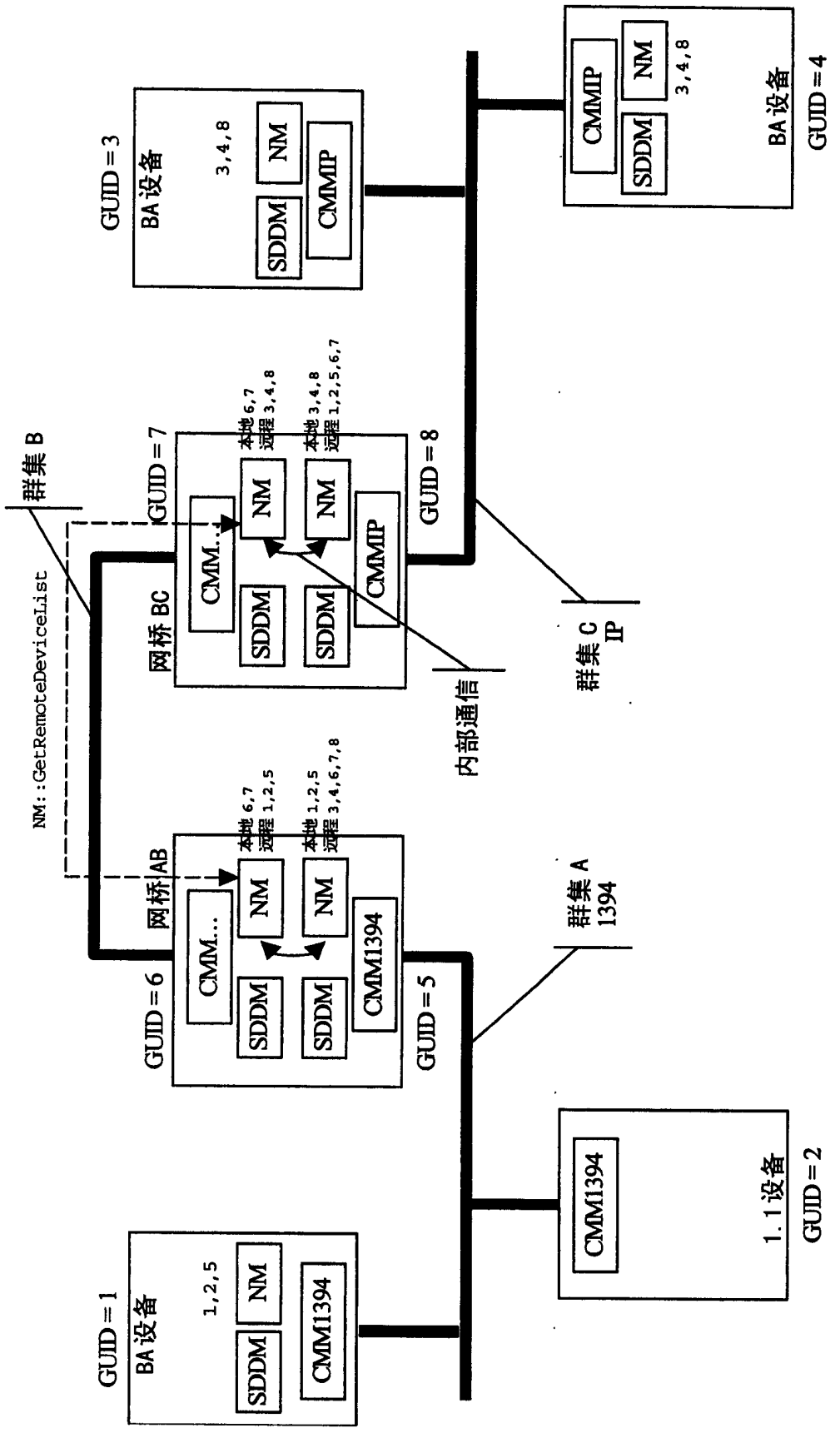

GUID=1
BA设备
1,2,5
SDDM
NM
CMM1394
GUID=2
1.1设备
CMM1394
GUID=3
BA设备
3,4,8
SDDM
NM
CMMIP
GUID=4
BA设备
3,4,8
SDDM
NM
CMMIP
GUID=5
GUID=6
网桥AB
CMM...
SDDM
NM
本地 6,7
远程 1,2,5
SDDM
NM
本地 1,2,5
远程 3,4,6,7,8
CMM1394
GUID=7
网桥BC
CMM...
SDDM
NM
本地 6,7
远程 3,4,8
SDDM
NM
本地 3,4,8
远程 1,2,5,6,7
CMMIP
GUID=8
群集 B
NM::GetRemoteDeviceList
内部通信
群集 C
IP
群集 A
1394

图 15

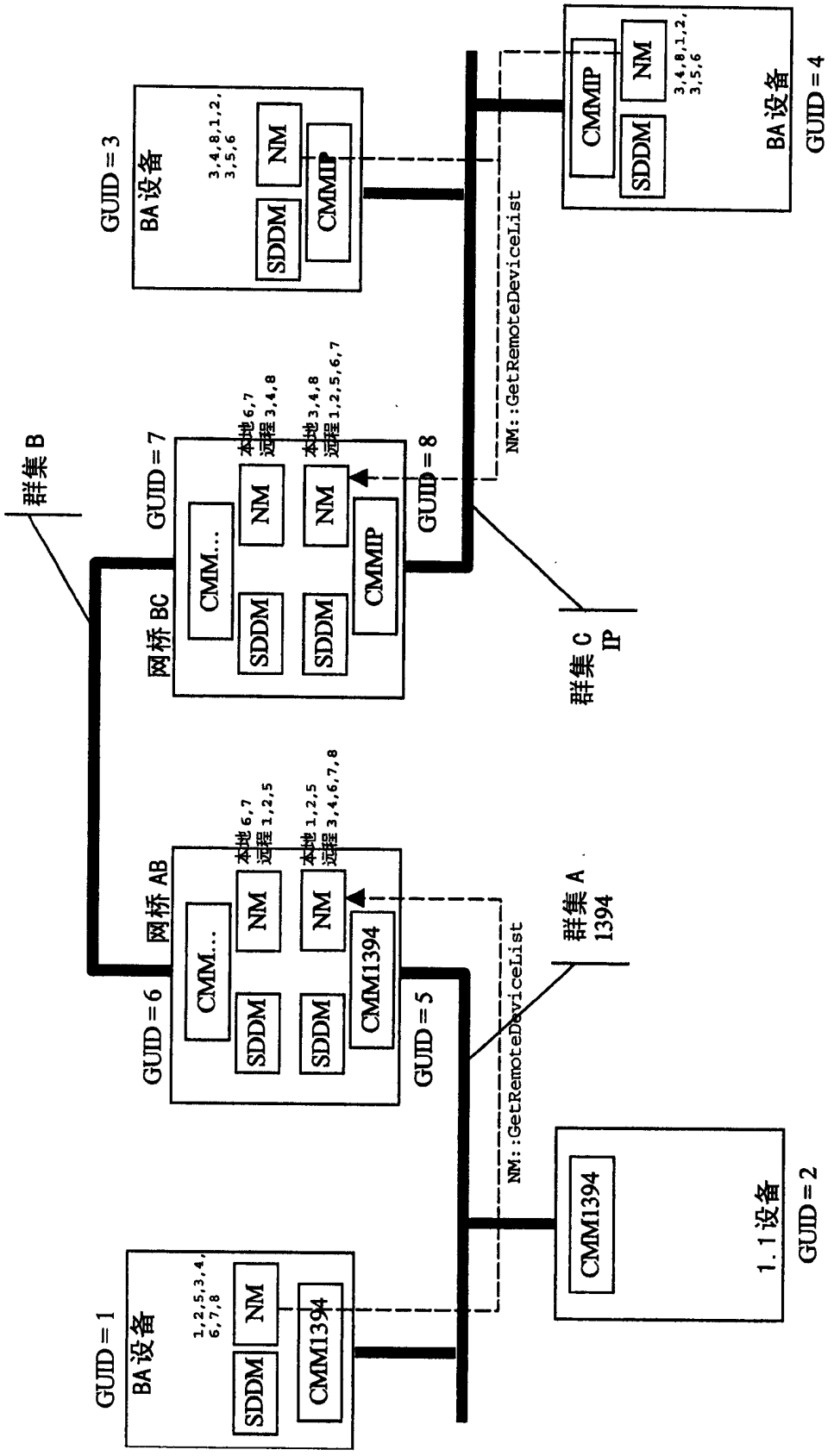

GUID = 1
BA 设备
1,2,5,3,4,
6,7,8
SDDM
NM
CMM1394
GUID = 2
1.1 设备
CMM1394
GUID = 3
BA 设备
3,4,8,1,2,
3,5,6
SDDM
NM
CMMIP
GUID = 4
BA 设备
3,4,8,1,2,
3,5,6
SDDM
NM
CMMIP
GUID = 5
GUID = 6
网桥 AB
CMM...
SDDM
NM
CMM1394
本地 6,7
远程 1,2,5
本地 1,2,5
远程 3,4,6,7,8
GUID = 7
网桥 BC
CMM...
SDDM
NM
本地 6,7
远程 3,4,8
GUID = 8
SDDM
NM
CMMIP
本地 3,4,8
远程 1,2,5,6,7
群集 B
群集 C
IP
群集 A
1394
NM::GetRemoteDeviceList
NM::GetRemoteDeviceList

图 16

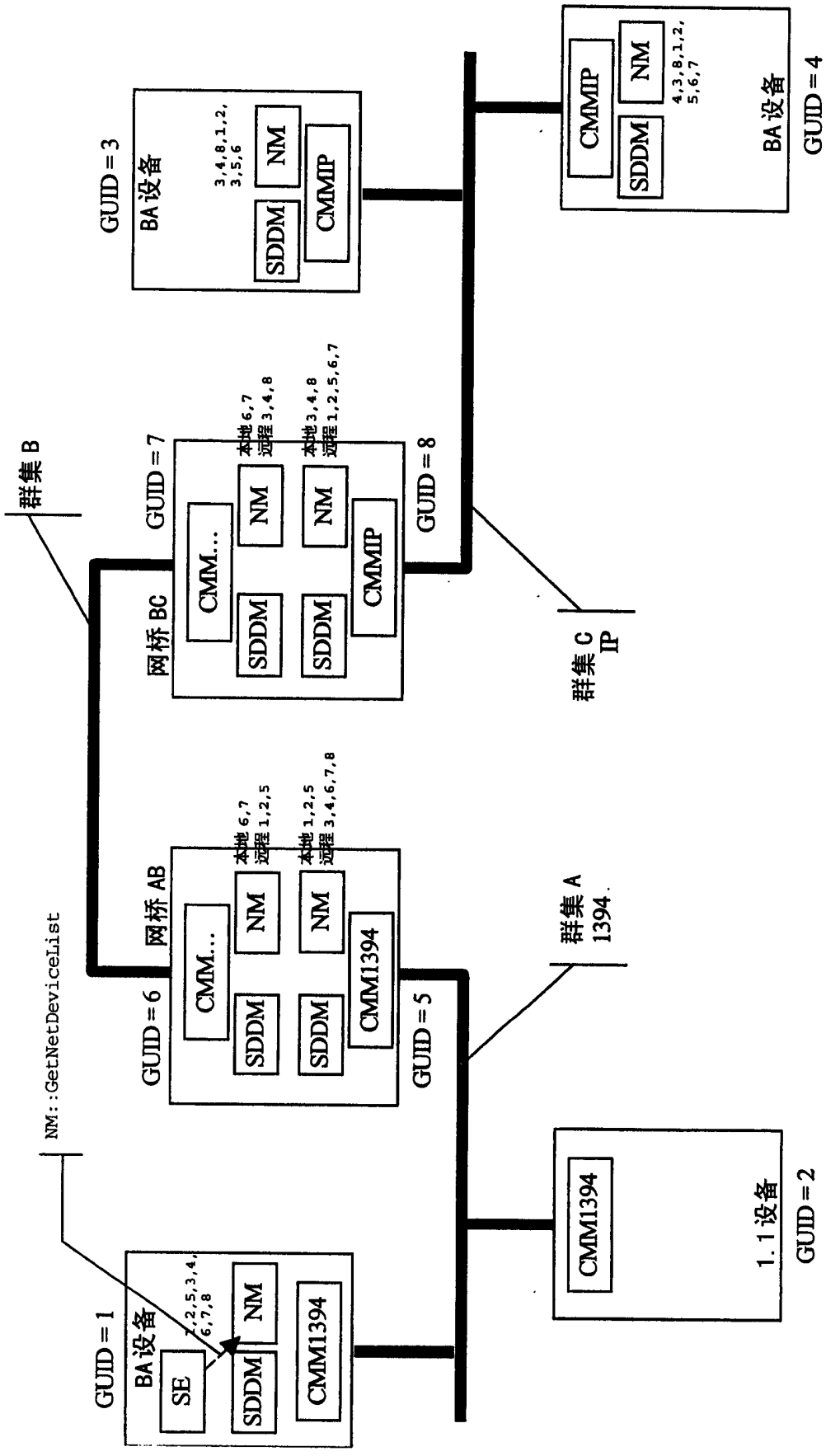

NM::GetNetDeviceList
GUID = 1
BA设备
SE
1,2,5,3,4,
6,7,8
SDDM
NM
CMM1394
群集 A
1394
GUID = 2
1.1设备
CMM1394
GUID = 3
BA设备
3,4,8,1,2,
3,5,6
SDDM
NM
CMMIP
GUID = 4
BA设备
4,3,8,1,2,
5,6,7
SDDM
NM
CMMIP
群集 B
网桥 AB
GUID = 6
CMM...
SDDM
NM
本地 6,7
远程 1,2,5
SDDM
NM
本地 1,2,5
远程 3,4,6,7,8
CMM1394
GUID = 5
网桥 BC
GUID = 7
CMM...
SDDM
NM
本地 6,7
远程 3,4,8
SDDM
NM
本地 3,4,8
远程 1,2,5,6,7
CMMIP
GUID = 8
群集 C
IP

图 17

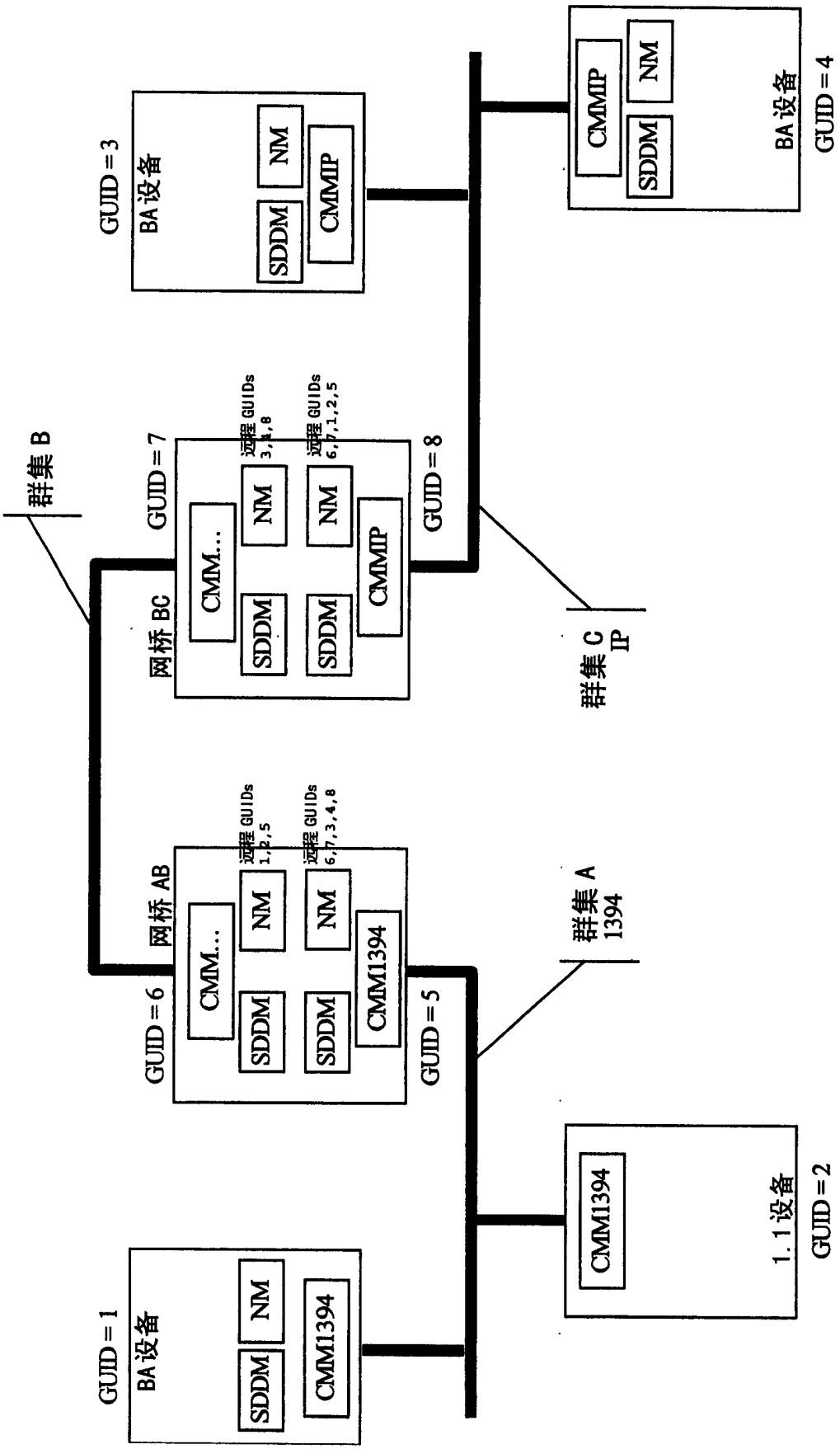

群集 B
GUID = 3
BA 设备
SDDM
NM
CMMIP
GUID = 4
BA 设备
CMMIP
SDDM
NM
网桥 BC
GUID = 7
CMM...
SDDM
NM
远程 GUIDs 3,4,8
SDDM
NM
远程 GUIDs 6,7,1,2,5
CMMIP
GUID = 8
群集 C IP
网桥 AB
GUID = 6
CMM...
SDDM
NM
远程 GUIDs 1,2,5
SDDM
NM
远程 GUIDs 6,7,3,4,8
CMM1394
GUID = 5
群集 A 1394
GUID = 1
BA 设备
SDDM
NM
CMM1394
CMM1394
1.1 设备
GUID = 2

图 18

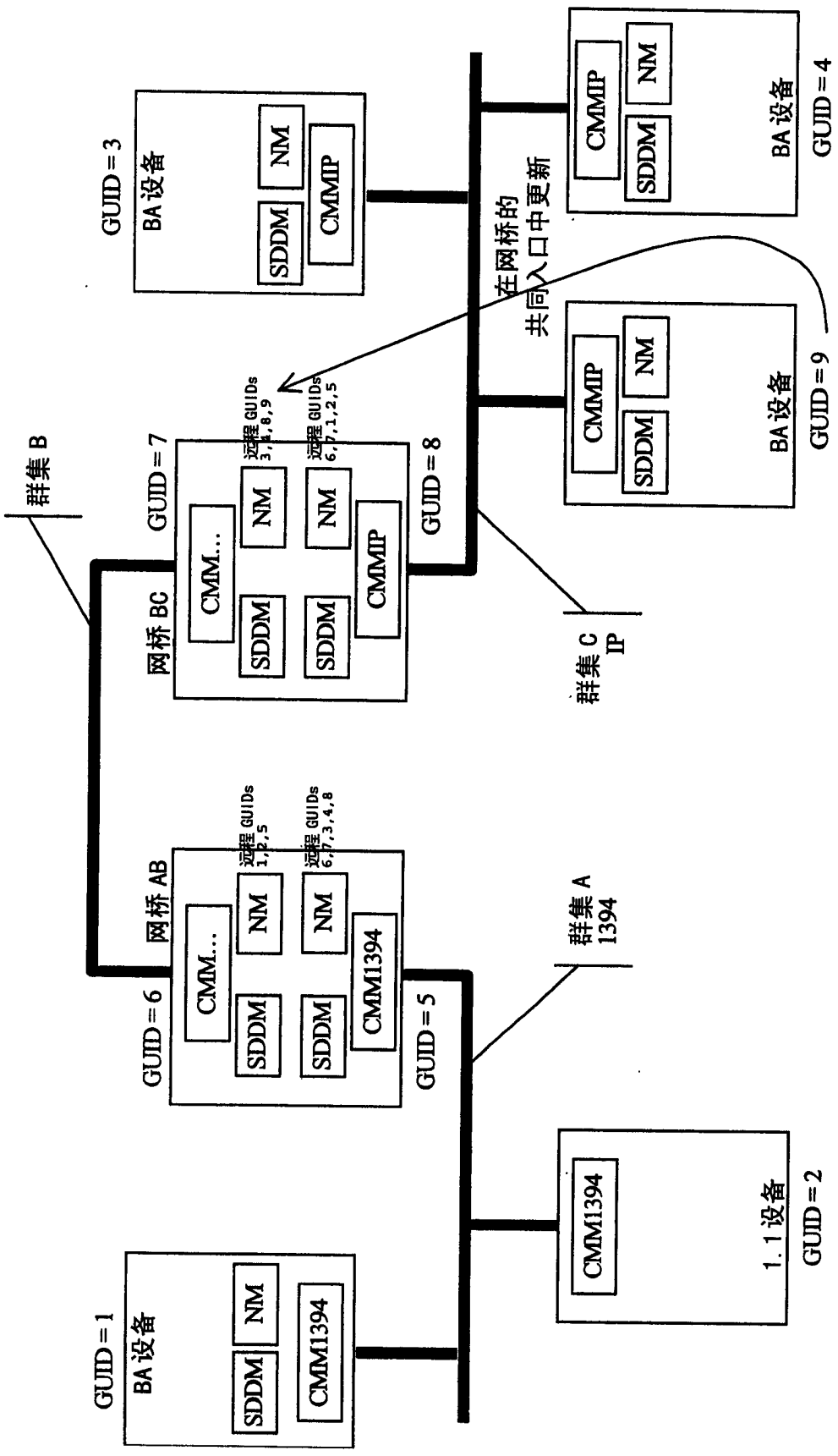

群集 B
网桥 AB
网桥 BC
GUID = 1
BA 设备
SDDM
NM
CMM1394
GUID = 2
1.1 设备
GUID = 3
GUID = 4
GUID = 5
GUID = 6
GUID = 7
GUID = 8
GUID = 9
CMM...
CMMIP
远程 GUIDs 1,2,5
远程 GUIDs 6,7,3,4,8
远程 GUIDs 3,4,8,9
远程 GUIDs 6,7,1,2,5
群集 A 1394
群集 C IP
在网桥的 共同入口中更新

图 19

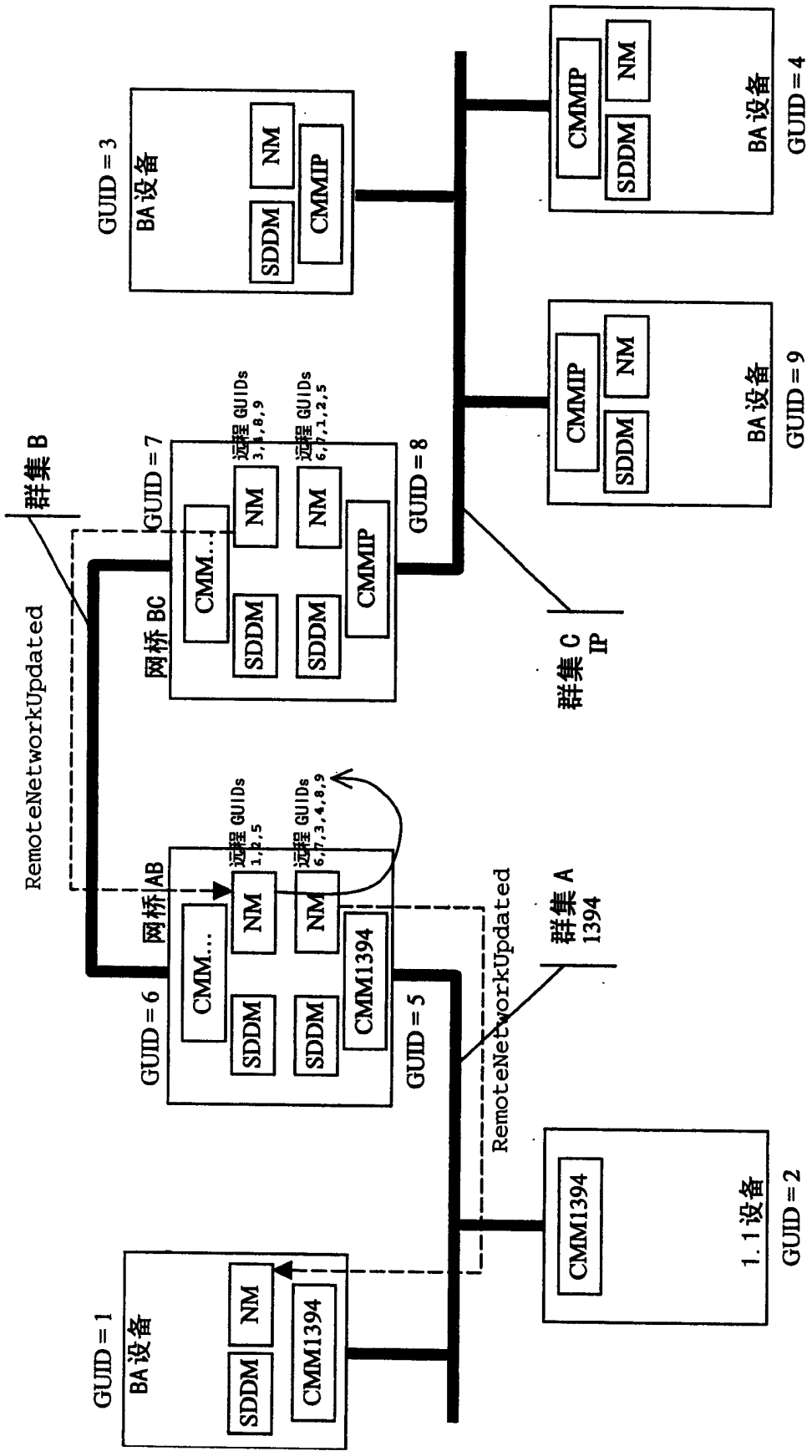

GUID = 1
BA设备
SDDM
NM
CMM1394
GUID = 2
1.1设备
CMM1394
GUID = 3
BA设备
SDDM
NM
CMMIP
GUID = 4
BA设备
CMMIP
SDDM
NM
GUID = 5
GUID = 6
网桥AB
CMM...
SDDM
NM
远程 GUIDs 1,2,5
SDDM
NM
远程 GUIDs 6,7,3,4,8,9
CMM1394
GUID = 7
网桥BC
CMM...
SDDM
NM
远程 GUIDs 3,4,8,9
SDDM
NM
远程 GUIDs 6,7,1,2,5
CMMIP
GUID = 8
GUID = 9
BA设备
CMMIP
SDDM
NM
群集 B
RemoteNetworkUpdated
群集 C
IP
群集 A
1394
RemoteNetworkUpdated

图 20

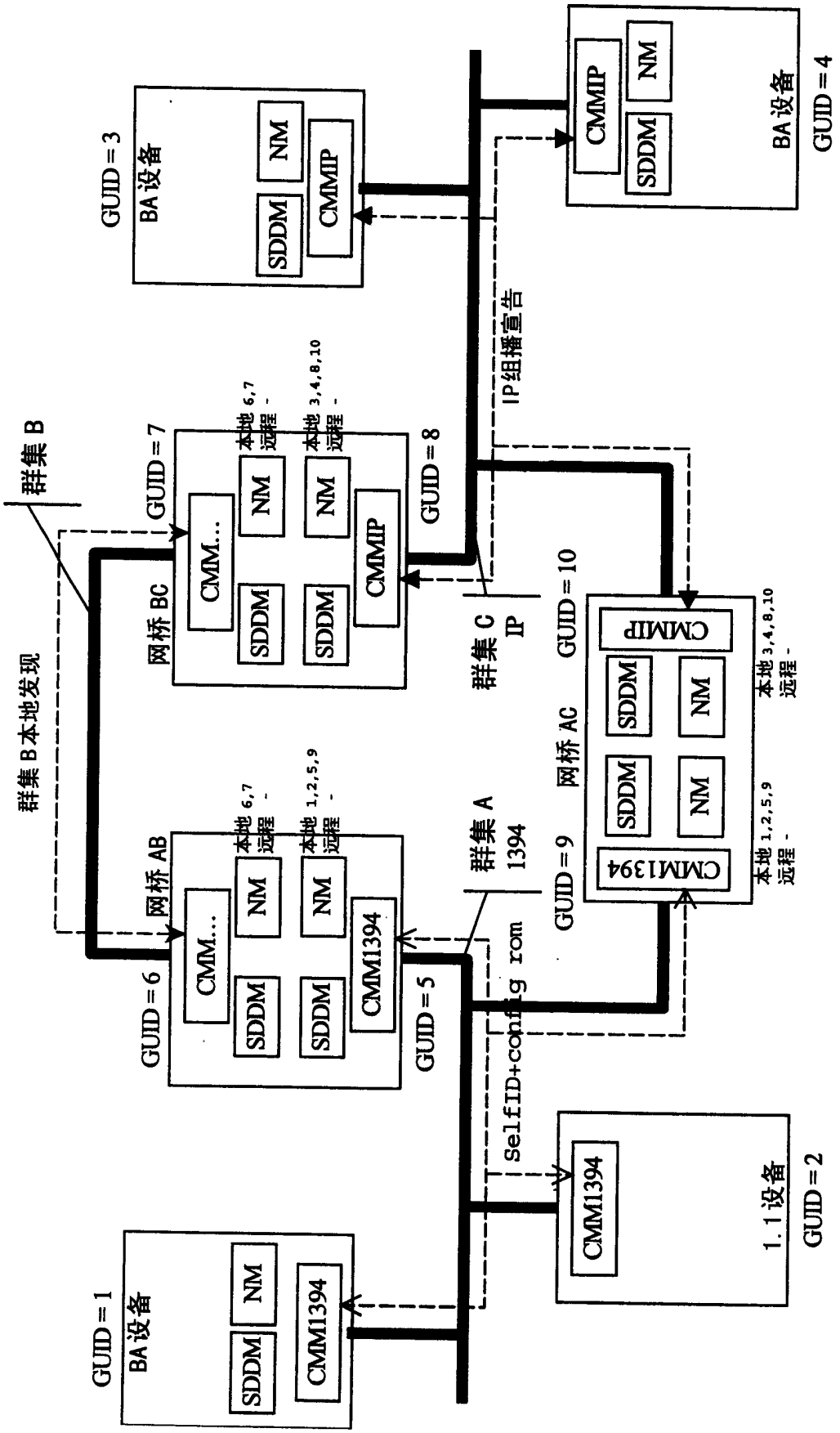

群集 B
群集 B本地发现
GUID = 3
BA设备
SDDM
NM
CMMIP
GUID = 4
BA设备
CMMIP
NM
SDDM
IP组播宣告
GUID = 7
网桥 BC
CMM...
本地 6,7
远程 -
本地 3,4,8,10
远程 -
GUID = 8
群集 C
IP
GUID = 10
网桥 AC
CMMIP
GUID = 6
网桥 AB
CMM...
本地 6,7
远程 -
本地 1,2,5,9
远程 -
CMM1394
群集 A
1394
GUID = 5
GUID = 9
CMM1394
本地 1,2,5,9
远程 -
本地 3,4,8,10
远程 -
SelfID+config rom
GUID = 1
BA设备
CMM1394
1.1设备
CMM1394
GUID = 2

图 21

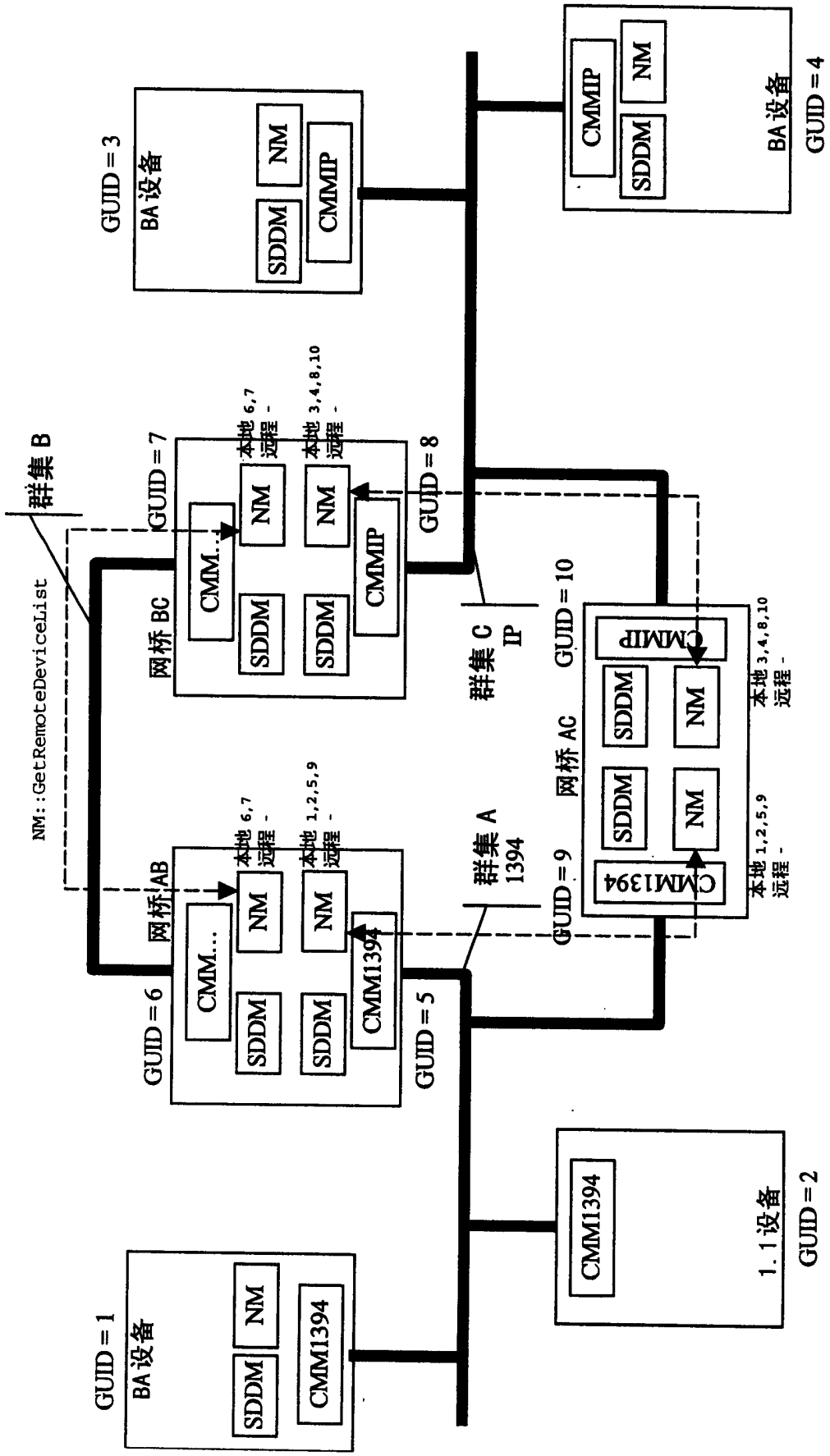

GUID = 1
BA设备
SDDM
NM
CMM1394
GUID = 2
1.1设备
CMM1394
GUID = 3
BA设备
SDDM
NM
CMMIP
GUID = 4
BA设备
CMMIP
SDDM
NM
GUID = 5
GUID = 6
网桥 AB
CMM...
SDDM
NM
CMM1394
本地 6,7
远程 -
本地 1,2,5,9
远程 -
GUID = 7
网桥 BC
CMM...
SDDM
NM
CMMIP
本地 6,7
远程 -
本地 3,4,8,10
远程 -
GUID = 8
GUID = 9
网桥 AC
GUID = 10
CMM1394
CMMIP
SDDM
NM
本地 1,2,5,9
远程 -
本地 3,4,8,10
远程 -
群集 A
1394
群集 B
群集 C
IP
NM::GetRemoteDeviceList

图 22

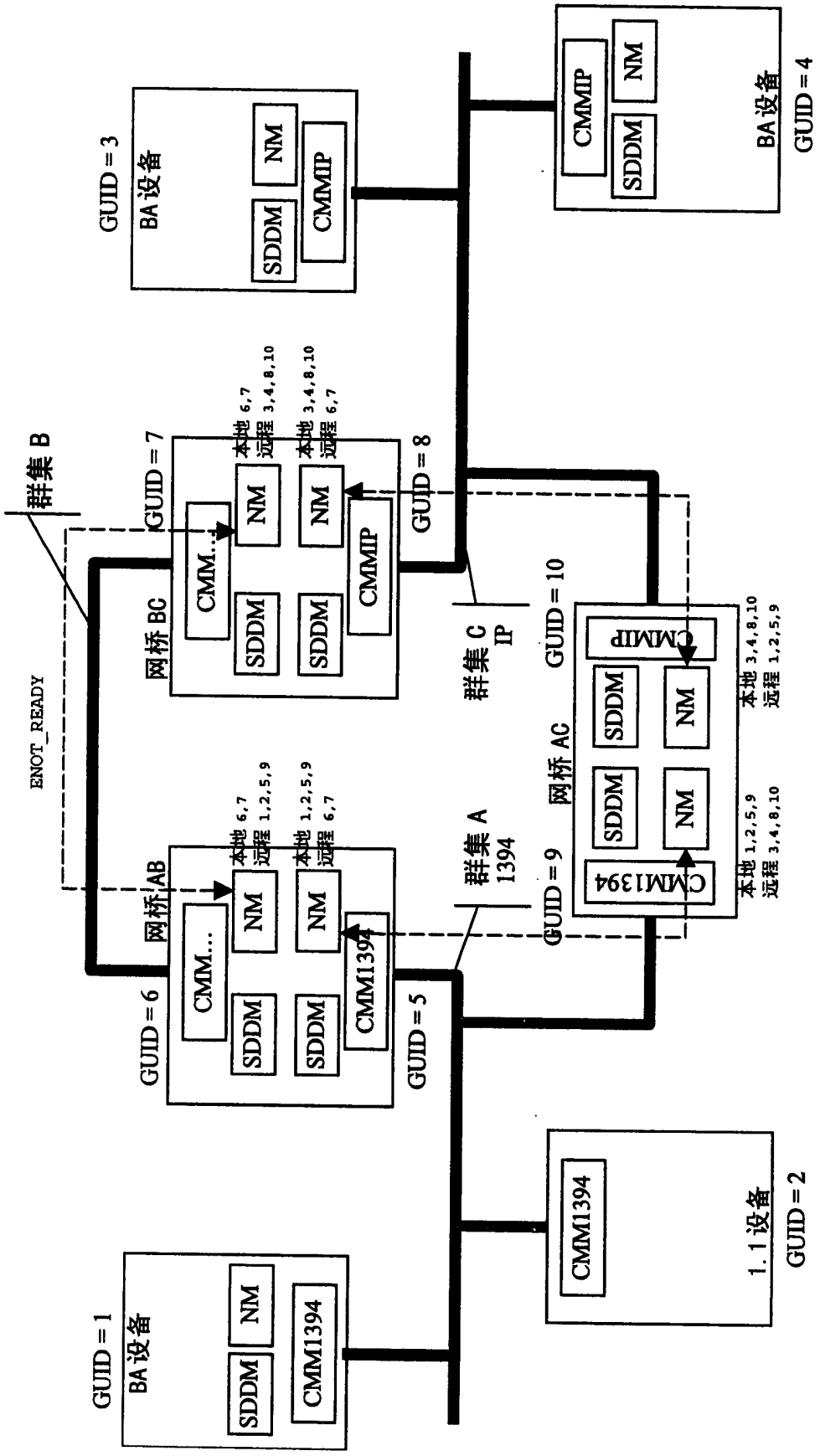

GUID = 1
BA设备
SDDM
NM
CMM1394
GUID = 2
1.1设备
CMM1394
GUID = 3
BA设备
SDDM
NM
CMMIP
GUID = 4
BA设备
CMMIP
SDDM
NM
GUID = 5
GUID = 6
网桥AB
CMM...
SDDM
NM
CMM1394
本地 6,7
远程 1,2,5,9
本地 1,2,5,9
远程 6,7
GUID = 7
网桥BC
CMM...
SDDM
NM
CMMIP
本地 6,7
远程 3,4,8,10
本地 3,4,8,10
远程 6,7
GUID = 8
GUID = 9
网桥AC
CMM1394
SDDM
NM
CMMIP
SDDM
NM
本地 1,2,5,9
远程 3,4,8,10
本地 3,4,8,10
远程 1,2,5,9
GUID = 10
群集 A
1394
群集 B
群集 C
IP
ENOT_READY

图 23

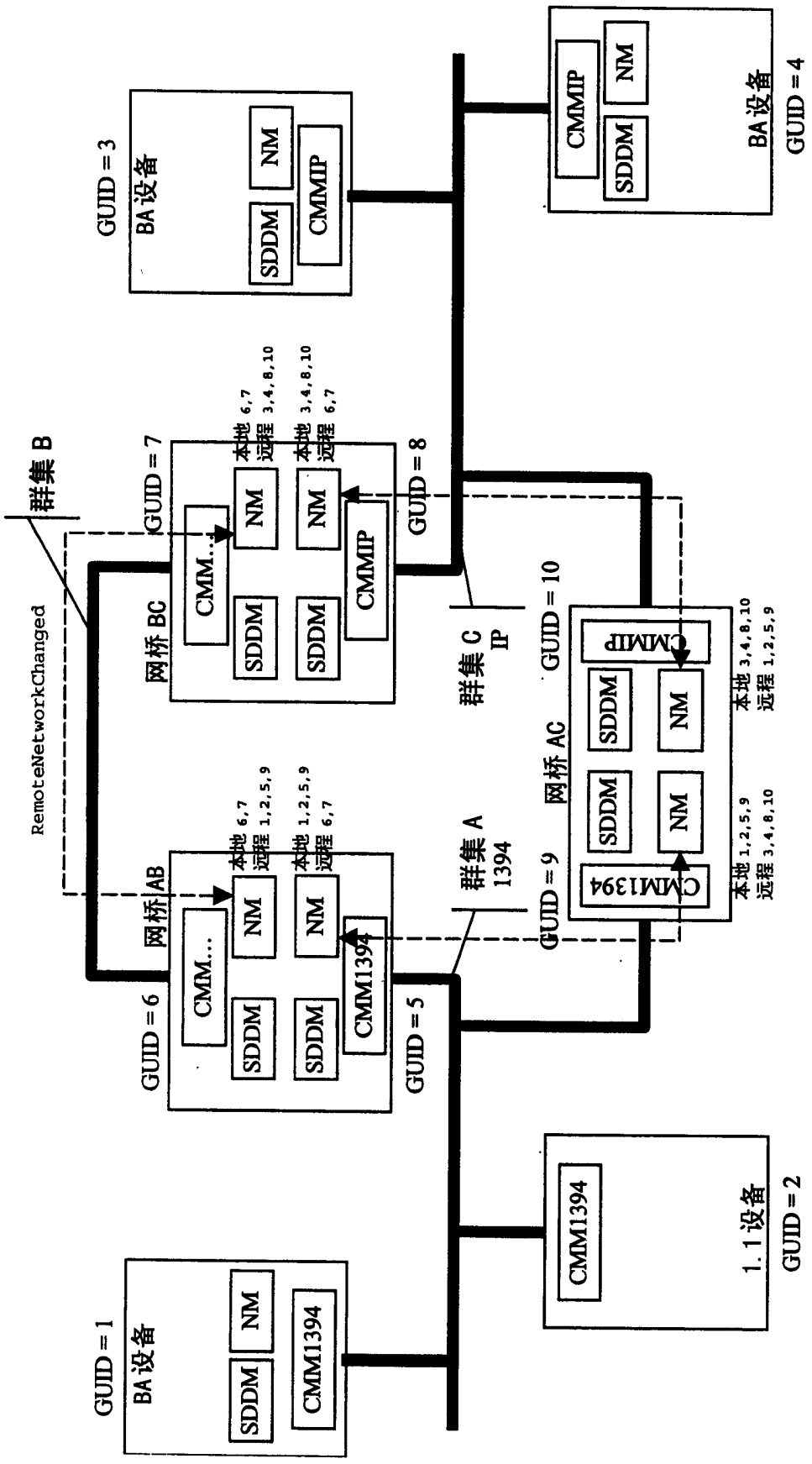

GUID = 1
BA设备
SDDM
NM
CMM1394
GUID = 2
1.1设备
CMM1394
GUID = 3
BA设备
SDDM
NM
CMMIP
GUID = 4
BA设备
CMMIP
SDDM
NM
GUID = 5
GUID = 6
网桥AB
CMM...
SDDM
NM
CMM1394
本地 6,7
远程 1,2,5,9
本地 1,2,5,9
远程 6,7
GUID = 7
网桥BC
CMM...
SDDM
NM
CMMIP
本地 6,7
远程 3,4,8,10
本地 3,4,8,10
远程 6,7
GUID = 8
GUID = 9
网桥AC
CMM1394
SDDM
NM
CMMIP
本地 1,2,5,9
远程 3,4,8,10
本地 3,4,8,10
远程 1,2,5,9
GUID = 10
群集A
1394
群集B
群集C
IP
RemoteNetworkChanged

图 24

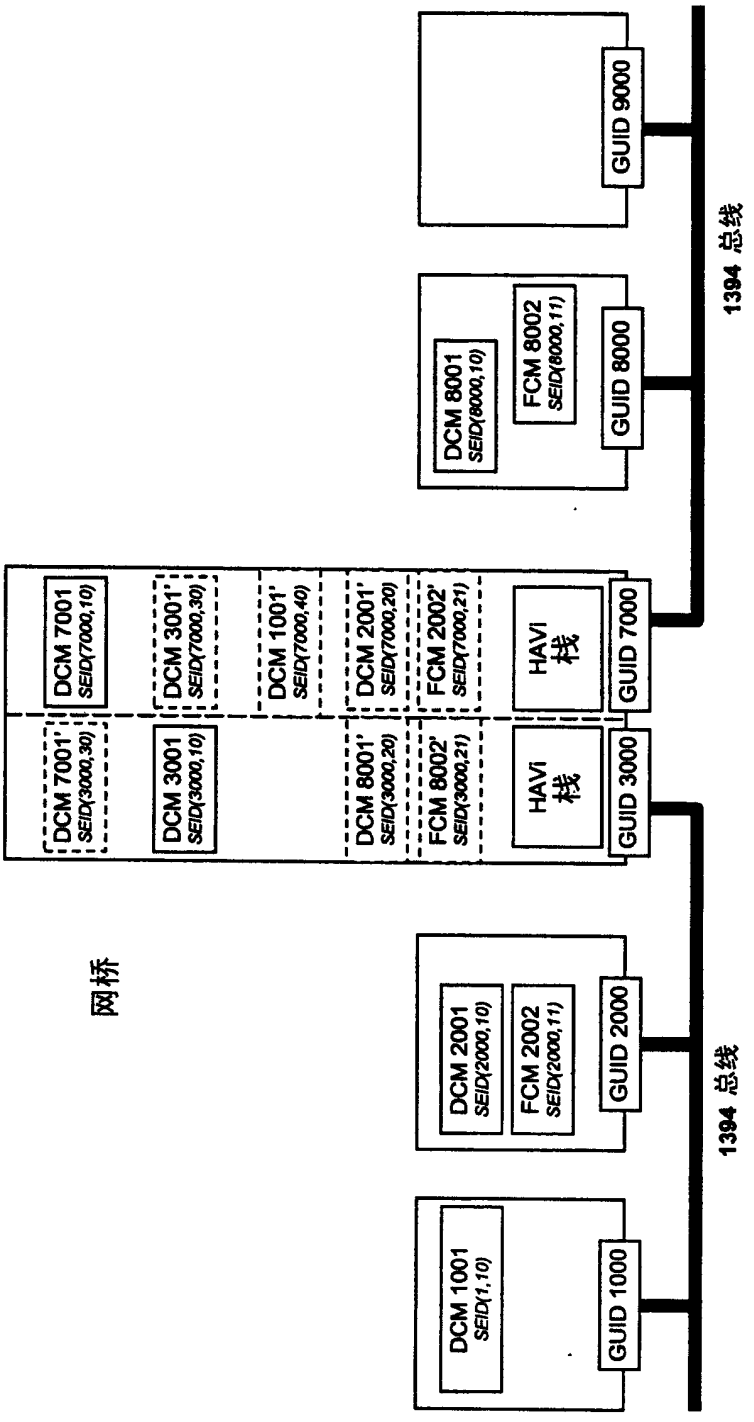

网桥
DCM 7001'
SEID(3000,30)
DCM 7001
SEID(7000,10)
DCM 3001
SEID(3000,10)
DCM 3001'
SEID(7000,30)
DCM 1001'
SEID(7000,40)
DCM 8001'
SEID(3000,20)
DCM 2001'
SEID(7000,20)
FCM 8002'
SEID(3000,21)
FCM 2002'
SEID(7000,21)
HAVi 栈
HAVi 栈
GUID 3000
GUID 7000
1394 总线
DCM 1001
SEID(1,10)
DCM 2001
SEID(2000,10)
FCM 2002
SEID(2000,11)
GUID 1000
GUID 2000
DCM 8001
SEID(8000,10)
FCM 8002
SEID(8000,11)
GUID 8000
GUID 9000
1394 总线

图 25