[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 3/12 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410032321.6

[45] 授权公告日 2007 年 6 月 20 日

[11] 授权公告号 CN 1322411C

[22] 申请日 2004.3.26
[21] 申请号 200410032321.6
[30] 优先权
[32] 2003.3.31 [33] US [31] 10/403，788
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 R·E·奥莱斯 E·G·安托诺夫
M·P·郝洛尔 T·J·莱特尔
T·罗斯 A·L·汀
[56] 参考文献
US6160796A 2000.12.12
US6460069B1 2002.10.1
审查员 陈晓华

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 陈 斌

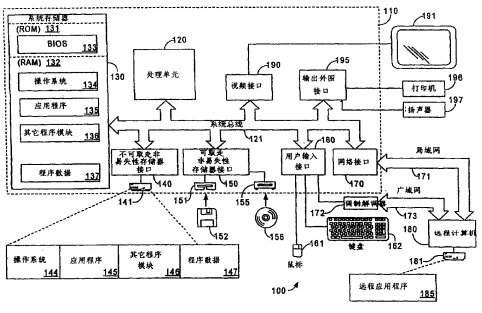
权利要求书 3 页 说明书 29 页 附图 10 页

[54] 发明名称

维护网络外围设备驱动程序组件的方法和系统

[57] 摘要

揭示了网络外围设备驱动程序维护架构和对应的方法。 该架构包括驱动程序版本识别层，它包括一组驱动程序描述。 每个驱动程序版本包括一组构成每个特定的驱动程序版本的组件版本。 在驱动程序版本识别层下的驱动程序组件层包括单独识别的组件版本(容件)，后者包括组成由驱动程序版本识别的组件版本的驱动程序文件的组。 揭示了用于保持多个同时活动的连网外围设备驱动程序的方法。该方法包括在机器上以包括由机器完成的下列步骤的方式存储新的驱动程序。 最初，机器在目录结构中建立一容件，用于存储组成驱动程序组件的版本的一组驱动程序文件。 对该新的容件赋予单独的识别符名。 一组与组件版本相关的驱动程序文件被插入该容件。



1.用于在连网的机器上维持多个同步活动的连网的外围设备驱动程序的组件的方法，其中同时活动的连网的外围设备驱动程序的不同组件可能包含带有同一名字的不同的外围设备驱动程序文件，该方法包括：

建立用于存储组成连网的外围设备驱动程序组件的版本的一组驱动程序文件的容件；

对此容件赋予单独的识别符名；和

将该组驱动程序文件插入该容件。

2.如权利要求 1 的方法，其特征在于，该容件是文件系统子目录。

3.如权利要求 1 的方法，其特征在于，单独的识别符名包括该设备驱动程序组件的版本识别。

4.如权利要求 3 的方法，其特征在于，版本包括日期。

5.如权利要求 3 的方法，其特征在于，版本包括一编码的多段的值。

6.如权利要求 1 的方法，其特征在于，还包括：

对由连网的外围设备的服务器保持的外围设备队列，请求驱动程序识别信息；和

响应请求的步骤，接收规定连网外围设备驱动程序的特定版本的驱动程序的识别。

7. 如权利要求 6 的方法，其特征在于，包括：

将该驱动程序识别与当前保持在该目录结构中的一组驱动程序版本比较。

8.如权利要求 7 的方法，其特征在于，还包括：

判定该组驱动程序版本未包括对应于该驱动程序识别的驱动程序版本；和

上载该驱动程序版本的拷贝。

9. 如权利要求 8 的方法，其特征在于，该驱动程序识别还规定驱动程序包的识别，它包括对应于包含连网外围设备驱动程序的版本的驱动程序包版本的版本信息，该方法还包括：

将驱动程序包识别与一组可得到的包版本进行比较。

10.如权利要求 9 的方法，其特征在于，还包括：

判定该驱动程序包识别不对应于在包版本组中的一个包版本；和

发出包括该驱动程序包识别的查询到连网的驱动程序包的目录。

11.如权利要求 8 的方法，其特征在于，上载步骤包括上载包含该驱动程序的拷贝的驱动程序包的版本的完整拷贝。

12.如权利要求 1 的方法，其特征在于，还包括提供该容件的活动的驱动程序组件存储。

13.用于在连网的机器上维持多个同时活动的连网外围设备驱动程序的系统，其中不同的同时活动的连网外围设备驱动程序可能包含带有同一名字的不同的外围设备驱动程序文件，该系统包括：

建立用于存储组成连网的外围设备驱动程序组件的版本的一组驱动程序文件的容件的装置；

对此容件赋予单独的识别符名的装置；和

将该组件驱动程序文件插入该容件的装置。

14.如权利要求 13 的系统，其特征在于，该容件是文件系统子目录。

15.如权利要求 13 的系统，其特征在于，单独的识别符名包括该设备驱动程序组件的版本识别。

16.如权利要求 15 的系统，其特征在于，版本包括日期。

17.如权利要求 15 的系统，其特征在于，版本包括一编码的多段的值。

18.如权利要求 13 的系统，其特征在于，还包括：

对由连网的外围设备的服务器保持的外围设备队列，请求驱动程序识别信息的装置；和

响应用于请求的装置，接收规定连网外围设备驱动程序的特定版本的驱动程序的识别的装置。

19.如权利要求 18 的系统，其特征在于，还包括：

将该驱动程序识别与当前保持在该目录结构中的一组驱动程序版本作比较的装置。

20.如权利要求 19 的系统，其特征在于，还包括：

判定该驱动程序版本未包括对应于该驱动程序识别的驱动程序版本的装置；和

上载该驱动程序版本的一个拷贝的装置。

21.如权利要求 20 的系统，其特征在于，该驱动程序识别还规定驱动程序包的识别，它包括对应于包含连网外围设备驱动程序的版本的驱动程序包版本的版本信息，系统还包括：

将驱动程序包括识别与一组可得到的包版本进行比较的装置。

22.如权利要求 21 的系统，其特征在于，还包括：

判定该驱动程序包识别不对应于包版本组中的一个包版本的装置；和

发出包括该驱动程序包识别的查询到连网的驱动程序包的目录的装置。

23.如权利要求 20 的系统，其特征在于，用于上载的装载获得包含该驱动程序的拷贝的驱动程序的版本的完整拷贝。

24.如权利要求 13 的系统，其特征在于，还包括提供保持该容件的活动的驱动程序组件存储的装置。

25.用于在网络中保持驱动程序包的多个版本的方法，其中不同的驱动程序可能带有同一名字，该方法包括：

在目录结构中建立用于存储包括一组文件的驱动程序包版本的新的容件，该组文件包括至少一个包括驱动程序文件的文件；

将一个强名赋给单独识别驱动程序包的版本的新的容件；和

将该组驱动程序包版本的文件插入新的驱动程序包的容件。

维护网络外围设备驱动程序组件的方法和系统

技术领域

本发明一般涉及网络计算机系统。本发明尤其涉及在客户机共享外围设备（如打印机、扫描仪、多功能外围设备等）的网络环境中管理连网的外围设备客户的方法和计算机系统机制。网络客户依靠本地加载到客户机的驱动程序和有关的驱动程序组件与共享的外围设备通讯。

背景技术

在包括在网络上互相连结的多个分别的用户/计算机从企业中利用打印机及其它外围设备的流行和经济的方法，是在多个分别的用户/计算机之中共享连网的外围设备。不是对每台客户机提供专用的外围设备，网络提供连网客户机和服务器的特有的机器间通讯的能力，以在许多用户中共享的外围设备。那样的共享通常改善了外围设备的利用程度并降低了为连网的用户提供设备服务的开销。

由多个连网客户共享外围设备给出为每个客户提供对多个外围设备的范例（如打印机）的访问的机会，且客户通过驱动程序与支持各种外围设备的服务器通讯。外围设备驱动程序通常包括组合起来提供完整驱动程序的多个组件（如文件或文件的组合，也称为“部件（assembly）”）。驱动程序支持为外围设备连结，在客户与服务器之间交互的各种方面。

在包括由数以百计的用户共享的多个外围设备的大型网络中，管理外围设备驱动程序包括在客户机上查找，存储，安装（激活），和更新驱动程序。在那样的环境中，驱动程序的管理由管理员具体地走到每台客户机并具体地加载新的驱动程序，这不是容易实现的。以前在众知的“定位打印（point－and－print）”的打印机驱动程序管理布局中，打印机驱动程序在“一个文件接着文件”的基础上从源传送到目标客户机。那样的更新机制引出一系列问题，包括：对给定设备安装不完整的软件包；在文件更新期间不能对特定的外围设备保留合适的驱动程序文件的版本；在加载到外围设备服务器和客户的驱动程序之间通常不能保持一致性/兼容性。

当外围设备在客户/服务器连网的外围设备环境中使用时，外围设备驱动程序的不匹配是一个问题。在运行同一版本驱动程序时，客户和服务器操作得最好。然而，若一个客户只能同时运行一个驱动程序版本，当客户与运行同一驱动程序的不同版本的两个服务器交互时，会引起驱动程序的不匹配。当在客户机上安装设备驱动程序时，只有具有同一名字的一个版本的驱动程序文件被安装在客户机上。在文件名在两个不同驱动程序版本之间交叠的情况，客户机/用户只存储两个同名文件之一（如文件版本带有更当前的文件创建日期）。结果，完全的客户－服务器驱动程序的兼容性只能对客户具有外围设备连结的，运行不同驱动程序版本的，多个活动的服务器中的一个得到保证。这对于执行一个设备驱动程序版本，它的一个或多个组件的版本已被活动的驱动程序目录中其它驱动程序组件版本替代，的外围设备服务器又引起不协调的功能。在客户机上，活动的驱动程序的替代能在多个层次的任一个上发生，包括：（1）在驱动程序层次上，其中一个驱动程序版本不同于第二个驱动程序版本；（2）在驱动程序组件层次上，其中组件的内容在版本之间不同；（3）在驱动程序文件层次上，其中一个文件的两个同名版本的内容不同。

从改写驱动程序/组件/文件的版本引起的不兼容/不协调的例子包括不支持的配置特征，操作，和用户界面。

发明内容

本发明寻求着手解决关于在连网的外围设备环境中维护可能不同的驱动程序版本和它们在机器上相关的组件版本方面以前技术的缺点。揭示了维护多个同时活动的连网外围设备驱动程序的方法。例如，此方法由连网机器实现，以保证在下列情况下与所有连网的外围设备/服务器的兼容性，其中不同的同时活动的连网外围设备驱动程序可能包含带有同一名字的不同的外围设备驱动程序组件和/或文件。

本方法包括在机器上以包括由机器完成的下列步骤的方式存储新的驱动程序。最初，机器建立一容件，用于存储构成连网外围设备驱动程序的至少一个组件的一个版本的一组驱动程序文件。然后对该容件赋给单独的识别符名。随后将这组驱动程序文件插入该容件。

按本发明的另外方面，机器将活动的驱动程序保持在多层活动的网络设备驱动程序访问/存储架构内。该架构包括包含一组单独识别的驱动程序版本的

驱动程序版本识别层。在第二层，每个单独识别的驱动程序版本参考一组构成连网外围设备驱动程序的版本的驱动程序组件。在本发明的一个实施例中，组件包括文件组，组成驱动程序组件版本的文件存储在单独识别的分配给组件版本的子目录中。那样驱动程序组件的单独识别便于防止在后续的组建版本被加载到连网的系统的活动驱动程序存储时，驱动程序文件的第一版本改写带有同一名字的驱动程序文件的第二版本。

附图说明

虽然附后的权利要求详细列出了本发明的特征，从下面结合附图的详细描述能更好的理解本发明及其优点。附图是：

图 1 是描述用于实现本发明的实施例的示例性计算机系统的方框图；

图 2 是描述很好地加入本发明的代表性网络环境的高层原理图；

图 3 是图示地描述在本发明的示例性网络实施例中，保存在服务器和客户机中各种有关驱动程序的软件和信息的原理图；

图 4a 是描述在本发明的实施例中单个驱动程序包的示例性组件的方框图；

图 4b 是示例性分层树图，描述用于在驱动程序和从中提取的它们的程序包的版本之间进行组织和区分的多个层次；

图 5a 是示例性活动的驱动程序目录组织结构，包括一个清单和一组对应于每个安装在活动的驱动程序目录中的驱动程序的子目录；

图 5b 图示了示例性网络外围设备布局，其中单个客户能同时支持与运行同一驱动程序的不同版本的两个服务器的兼容性；

图 6 是用于元数据存储的示例性数据库；

图 7 是由对体现本发明的客户的打印纸脱机程序支持的示例性方法/功能组；

图 8 是流程图，概述了用于在驱动程序包存储器中安装软件包的一组步骤；

图 9 是流程图，概述了用于在对特定服务的客户机上维护合适的驱动程序版本的一组步骤；和

图 10 是流程图，概述了客户在与服务器的交互过程中如何访问驱动程序版本。

具体实施方式

通过例子，本发明体现在包括客户、服务器、和连网外围设备（如打印机、扫描仪等）的网络计算机系统环境中，例如，本发明在包括集中的驱动程序包存储的网络中实现，存储作为用于将外围设备驱动程序安装到请求的客户的定点源操作。集中的驱动程序包存储保存完整的驱动程序包的诸版本，包括由客户使用的可能的多驱动程序/版本连网以便与网外围设备的服务器通讯。

而且在本发明的实施例中，安装在驱动程序包存储器的软件包的完整性被保证。驱动程序完整性是通过在该软件包安装在集中的驱动程序存储或任何其它机器，为以后分配到请求的客户机或从该软件包安装驱动程序所用同时，拷贝整组与驱动程序包相关的文件组而得以保证。不试图消除冗余的文件。在安装的包中完整地不加选择地存储所有与驱动程序有关的文件，保证了当客户机请求安装驱动程序时，与特定请求的驱动程序相关的文件组将是既兼容又完整。

这里揭示的驱动程序存储和管理方案的又一方面包括在系统中保存活动的驱动程序的新方式。在本发明的一个实施例中，构成单独的驱动程序版本的至少一部分有关文件组保存在具有单独识别符名的目录密件中。这里提到的唯一识别符名提供了高度的保证，使具有同一识别符名的容件确实是同一个。通过组合在单独的识别符名下构成驱动版本的至少一部分的文件，在客户机上能同时激活不同的驱动程序的版本。

单独识别符名能以若干不同的格式出现。那样单独的识别符名的例子是在NET 部件的上下文中使用的强名（Strong name）。另选地，通过使用 GUID 版本组合单独地识别驱动程序的某个版本，能得到合适的功能。类似地，赋给驱动程序每个版本的 GUID 能本身用作单独的识别符名。指定驱动程序/文件的版本的 GUID 不便于容易地判断，两个不同的驱动程序/文件仅是同一驱动程序文件的两个版本，因而从驱动程序管理的观点，至少结合版本识别符名使用一个驱动程序/产品 GUID 通常更有利。在又一个实施例中，安全性/可靠性部分（如为其密钥的散列值）加到单独的识别符名，以便于认证特定驱动程序或其它组件/文件的源。

按这里揭示的实施例，使用单独的识别符名识别包含不同组成的驱动程序文件的不同版本的驱动程序，使客户机能同时支持与执行同一外围设备驱动程序的不同版本的服务器相关的外围设备（如打印机）连结。在两个不同服

务器上运行的那样不同版本的例子是两个服务器使用具有同一名字的驱动程序 DLL 文件的不同版本，不是改写两个驱动程序 DLL 文件之一，本发明意在存储两个不同 DLL 文件的每一个，到它自己单独识别的目录中。

在使用.NET 部件约定的本发明的示例性实施例中，当对共享的外围设备的驱动器版本开始存入集中的驱动程序存储时，对驱动程序的特定版本确定强名。例如，强名由驱动程序作者提供。另选地，在不知道这里描述特定的驱动程序存储器底层结构/方法编著驱动程序的情况，在存储驱动程序时，根据从与特定驱动程序版本相关的一个或多个文件提取的元数据（如版本号，时间标记，校验组等）产生合适的单独的识别符名。强名具有很高的似然率区分两个不同文件的情况一从而保证高度的可靠性，两个具有同一强名的文件确实是同一个。赋给各种版本容件的每一个的强名使集中的驱动程序存储和驱动程序的接收方（如打印机客户）从可能已存储在机器中的其它版本中区分出一个驱动程序组件（如部件）版本。

在本发明的特定实施例中，元数据存储以压缩的格式保存有关安装的包和驱动程序的信息。压缩的格式（如数据库）便于在以后当客户请求通过任何合适的外围设备驱动程序安装方法/激活驱动程序时，容易地查找。

在揭示的本发明的实施例中，外围设备驱动程序客户存储至少驱动程序的部分（或组件）到驱动程序专门版本的容件。当特定的驱动程序被放在客户机的活动程序存储时，它被赋予单独的识别符名（如强名）。如上解释，强名保证，驱动程序的特地告版本很容易从当时由客户保存的同一驱动程序的其它版本中区分出来（版本信息的内容还便于如根据哪个驱动程序更加新而作出决定）。一旦在接收的客户机上激活了特定驱动程序版本，赋给特定驱动程序版本的强名便于从在客户机上同一驱动程序的另外激活版本中区分出该驱动程序版本。

按客户端驱动程序/存储方案，在驱动程序安装期间当具有同样名字的驱动程序文件的以前版本被拷贝到驱动程序目录时，它可能被改写。与此方案相反，实施本发明的客户端驱动程序管理/存储方案时，客户驱动程序管理工具程序能够区分驱动程序机器相关组件/文件的各种版本。例如，当驱动程序的较新的版本从集中的驱动程序文件存活促安装到客户机时，客户在其驱动程序文件存储器中创建单独识别的容件（如文件系统目录），至少方便于与较新的驱动器版本有关的新组件。在本发明的一个实施例中，不是在新目录下

存储整个驱动程序组件的组，只有新的组件版本被存储在具有单独识别符名的新的容件中，以便区分同一驱动程序组件的各个版本。同一驱动程序组件的以前安装的老版本在其自己单独识别的容件中保持不变。结果，连网的外围设备（如打印机）客户能同时支持对同时运行连网打印机驱动程序的两个不同版本的两个独立的连网设备服务器的访问。

转向附图，图 1 画出合适的操作环境 100 的例子，用于在加入本发明的共享的网络外围设备环境中落实集中的驱动程序包存储器，外围设备客户和服务器。操作环境 100 仅是合适的操作环境的一个例子，不试图对对本发明的使用或功能的范围提出任何限止。适用于本发明的其它众知的计算系统，环境包括个人计算机，服务器计算器，膝上/便携计算设备，手持计算设备，多处理系统，基于微处理器的系统，网络 PC。小型机，大型主机，包括任何上述系统和设备的分布式计算机环境等，但不限于这些。

本发明用如程序模块那样拟由计算机执行的计算机可执行指令完成的一组步骤和过程的一般上下文来描述。通常，程序模块包括执行特地告任务或实现特定抽象数据类型的例行程序，对象，组件，数据结构等。虽然参考在单台计算机系统上本地执行的过程描述示例性实施例，本发明可能加入到在分布式计算环境操作的网络节点中，在那里任务由通过通讯网络连接的远程处理设备完成。在分布式计算环境中，程序模块通常位于包括存储器设备的本地和远程的计算机存储介质中。

继续参考图 1，实现本发明的示例性系统包括拟以计算机 110 形式的通用计算设备。计算机 110 的组件包括处理单元 120，系统存活促其 130，和将包括系统存储器的各种组件连接到处理单元 120 的系统，但不限于这些。系统总线 121 能是若干总线结构的任一种，包括存储器总线或存储控制器，外围总线，和使用各种总线提心结构的任一种的局部总线。例如，那样的体系结构包括工业标准体系结构（ISA）总线，图形加速端口（AGP），微通道体系结构（MCA）总线，增强的 ISA（EISA）总线，视频电子技术标准协会（VESA）局部总线，和外设部件互连（PCI）总线，后者也称 Mezzanine 总线。

计算机 110 通常包括各种计算机可读介质。计算机可读介质能是任何由计算机 110 访问的可得到的介质，并包括易失和非易失介质，可取走和不可取走介质。例如，计算机可读介质可包括计算机存储介质和通讯介质，但不限于这些。计算机存储介质包括易任何存储的方法和技术实现的易失和非易失，

可取走和不可取走介质，用于存储如计算机的可读指令，数据结构，程序模块或其它数据那样的信息，计算机存储介质包括 RAM，ROM，EEPROM，闪存或其它存储技术，CD－ROM 数字多功能盘（DVD）或其它光盘存储设备，或任何其它能存储希望的信息并能由计算机 110 访问的介质，但不限于这些。通讯介质一般体现在计算机可读指令名数据结构，或以如载波或其它传输机制的调制数据信号的数据，并包括任何信息提交介质。术语“调制数据信号”指的是具有以如在信号中编码信息的方式设置或改变的一个或多个特征的信号。例如，通讯介质包括如有线网络或直线连结的有线介质，如声波，RF，红外线或其它无线介质的无线介质，但不限于这些。任何上面的组合也包括在计算机可读介质的范围中。

系统存储器 130 包括以易失和/或非易失存储器方式的计算机存储介质，如只读存储器（ROM）131 和随机存储器 132。包含如在起动期间帮助在计算机 110 的各单元之间传输信息的基本例行程序的基本输入/输出系统 133（BIOS）有时存储在 ROM131。RAM132 通常包含由处理单元 120 立即访问和/或当时操作的数据和/或程序模块。例如图 1 示出操作系统 134，应用程序 135，其它程序模块 136，和程序数据 137，但不限于这些。

计算机 110 还能包括其它可取走/不可取走，易失/非易失计算机存储介质。仅作为例子，图 1 示出读写不可取走，非易失磁介质的硬盘驱动器 140，读写可取走，非易失磁盘 152 的磁盘驱动器 151，读写可取走，非易失光盘 156，如 CD－ROM 或其它光介质的光盘驱动器 155。其它在示例性操作的环境可使用的可取走/不可取走，易失./非易失计算机存储介质包括盒式磁带，闪存卡，数字多功能盘，数字视频带，固态 RAM，固态 ROM 等，但不限于这些。硬盘驱动器 141 通常经过如接口 140 那样不可取走存储器接口连接系统总线 121，而磁盘驱动器 151 和光盘驱动器 155 通常通过如接口那样的可取走存储器接口连接系统总线 121。

上述在图 1 中示出的驱动器及其相关的计算机存储介质提供对计算机 110 的计算机可读指令，数据结构，程序模块和其它数据的存储。例如在图 1 中，硬盘 141 示作为存储操作系统 144，应用程序 145，其它程序模块 146，和程序数据 147。注意，这些组件能与操作系统 134，应用程序 135，其它程序模块 136，和程序数据 137 相同或不相同。在这里对操作系统 144，应用程序 145，其它程序模块 146，和程序数据给出不同序号以表明至少它们是不同的拷贝。

用户能通过如键盘和常称为鼠标的定为设备，跟踪球或接触垫那样的输入设备输入命令和信息到计算机 110。其它输入设备（未示出）能包括麦克风，操作杆，游戏垫，卫星碟，扫描仪等。这些和其它输入设备常常通过连接系统总线的用户输入接口 160 连接到处理单元 120，但也能经过如并行口，游戏。或通用串行总线 USB 那样的其它接口和总线结构连接。监视器 191 和其它类型的显示设备也能经过如视频接口 190 那样的接口连接系统总线 121。除监视器以外，计算机也能包括通过输出外围接口 190 连接的其它外围输出设备，如扬声器 197 和打印机 196。

计算机 110 可能使用到如远程计算机 180 那样的一台或多台远程计算机的逻辑连接在网络管理环境中操作。远程计算机 180 能是个人计算机，服务器，路由器，网络 PC，对等设备或其它公共网络节点，并通常包括上述有关计算机 110 的许多或所有单元，虽然在图 1 中只示出存储器设备 181。在图 1 中画的逻辑连接包括局域网（LAN）171 和广域网（WAN）173，但还能包括其它网络。那样的网络环境在办公室，企业范围计算机网络，内联网和因特网是常见的。

在 LAN 网络环境使用时，计算机通过网络接口或适配器 170 连接到 LAN171。在 WAN 网络环境使用时，计算机 110 通常包括调制解调器 172 或其它建立经过因特网那样的 WAN173 通讯的装置。内置或外接的调制解调器能经过用户输入接口 160 或其它合适的机制连接系统总线 121。在网络环境中，关于计算机 110 画出的程序模块或其部分能存储在远程存储设备中。例如，图 1 示出驻留在存储器设备 181 的远程应用程序 185。可以理解，示出的网络连接时示例性的，能使用实现所述外围设备驱动程序管理方案的其它装置。

图 2 画出示例性网络，包括参与按本发明的示例实施例的连网外围设备（如打印机）驱动程序管理安排的一组网络实体。示例性计算机网络环境 200 包括一组为登录的用户执行应用程序的客户机 202（1－n）。客户机的数量（n）从几台计算机到成千上百是无所谓的。那样的网络设想包括广域网链路，它向实际上任何数量的另外客户机开放网络。客户机 202 通过网络打印机驱动程序配置打印特征（如方向，分解率，打印方式等），并经过网络链路 210 移交请求到由打印服务器 204 和 205 保存的打印队列。打印服务器 204 和 205 使用对应于由客户机使用的打印机驱动程序的打印机驱动程序处理打印请求，提交这些请求到特定的打印队列。打印服务器 204 和 205 使用与特定队

列相关的打印机驱动程序，移交用于合适打印机打印作业，该打印机相应于打印作业所占据的打印队列。随后，打印服务器 204 和 205 为客户机 202 通过网络链路 210 将移交的打印作业送到一组连网打印机 206 和 208 的一个（按客户的原始打印请求）。随后，打印机 206 和 208 接收从打印服务器 204 和 205 移交的打印作业，且打印机 206 和 208 产生打印文档输出。

在本发明的实施例中，客户机的客户计算机在首次安装和激活与管理打印请求的打印服务器（如 204 或 205）相关的合适的连网打印机驱动程序组件之后，发出打印请求。在本发明的实施例中，那样的组件最初以驱动程序包的形式（下面参考图 4a 和 4b 讨论）保存在网络 200 上集中的驱动程序存储 212。集中的驱动程序存储 212 从该软件包移交驱动程序（包括解压缩的组件）和元数据。元数据用下列两者之间充分详细的区别识别软件包和软件包中的各驱动程序（每个可能包括许多组件/文件）：（1）同一软件包/驱动程序的不同版本，或(2)对应不同设备驱动程序的同一命名的文件。

在本发明的实施例中，客户机 202 区分驱动程序和驱动程序的组成文件一存储在由每个客户机 202 维持的活动的驱动程序存储 302。与从集中的驱动程序存储 212 安装网络设备驱动器相关（或间接的通过包含与所需驱动程序相关的组件/文件的完整组的任何其它网络设备一如打印服务器 204 或 205 之一），客户机 202 对每个各别的驱动程序创建分别的容件（如目录）。从与该驱动调程序相关的签署信息（至少部分地）移交赋给每个驱动程序容件的各别的标号。在本发明的另外实施例中，前述的信息具有各种形式。

上述网络环境列出了较好的加入本发明的示例性计算机网络环境。本专业行家理解，本发明可应用到各种情况，在那里连网外围设备驱动程序的客户希望（若不是必需）保存具有可能重叠的组成的驱动程序文件名的多个活动的驱动程序（包括同一驱动程序的多个版本），在描述了那样的示例性网络环境后，外围设备驱动程序包的内容，处理程序包，并分配及维持从该包提取的驱动程序，将在下面按本发明图示的实施例予以讨论。

现转向图 3，设备在驱动程序存储安排中各驱动程序和/或驱动程序信息的三个表示。驱动程序包存储 300 以如由制造商发货的形式存储驱动程序包的完整拷贝。通常预期，软件包将包括支持多个计算机环境的多个驱动程序，甚至基本上同一驱动程序的不同版本（如包括增加/修改的特征和/或错误排除的同一驱动程序的顺序的发行）。在本发明的实施例中，驱动程序包存储 300

位于集中的驱动程序存储 212。然而，驱动程序包存储 300 能位于客户机或服务器。在本发明中的两种情况，驱动程序包存储 300 放在分布式文件系统的公共目录（文件共享）中。为节省空间，驱动程序包的内容被压缩。下面参考图 4a 和 4b 详细描述驱动程序包存储 300 的示例版本的组件。

在本发明的实施例中，驱动程序文件从驱动程序包存储 300 中提取并放在活动的驱动程序存储 302。通常在网络中的服务器和客户机上找到的活动的驱动程序存储 302 是用于保持加载的，解压缩的驱动程序文件的目录结构。在本发明的实施例中，活动的驱动程序存储 302 的驱动程序文件保存在由网络上其它客户和服务器机器可访问的分布式文件系统部分。在那样的实施例中，在客户或服务器机器上的活动的驱动程序存储 302 与在机器存储空间中其它的，非公共的组件隔离。下面参考图 5a 和 5b 描述示例的驱动程序存储 302 的形式和功能。

在网络外围设备驱动程序维护架构的实施例中，元数据存储 304 包括支持基于密钥的安排，并搜索加载到以及可用于本地机器的驱动程序的数据库。在本发明的实施例中，元数据存储 304 还包括对每个驱动程序版本的清单，它列出一组构成特定驱动程序版本的组件。例如，密钥可搜索的字段的例子包括：环境，制造商，驱动程序，和程序包。下面参考图 6 描述示例的数据库的方案。

转向图 4a，在驱动程序包存储 300 中的驱动程序包能采取各种形式。然而，在示例实施例中，驱动程序包包括三个主要部分：INF 文件 400，CAT 文件 410，和经压缩并放入 CAB 文件 420 的驱动程序源文件。INF 文件 400 描述驱动程序包的内容。在本发明的实施例中，INF 文件 400 包括由包 ID 和版本包的单独的 ID－便于在同一驱动程序的各版本之间作出区分。例如，版本由插入 INF 文件 400 的驱动程序版本字段的时间标记（年/月/日）指定。另选地，对特定程序包版本的单独的 ID 从散列操作在 CAT 文件 410 或甚至 CAB 文件 420 中的签署信息而产生。在又一个范例中，版本是包括识别较大/较小的版本并排除错误的发行的多个段的规格化的版本码。由识别信息的合适版本对方数量与多样性证明，本发明致力于与包相关的数据/元数据，它们提供高度的可靠性，使得两个具有同一版本值的包确实是同一个包。那样的识别在这里称为“强名－Strong name）。

INF400 包括到访问驱动程序包中 CAT 文件 410 的安装程序的指向。CAT

文件 410 包括认证该包的单独的签署。例如，CAT 文件 410 还包括识别签署成为有效的平台的元数据，和用于创建该签署的认证，但不限于这些。除了识别 CAT 文 410 以外，INF 文件 400 描述如何安排在驱动程序包中的文件，以移交一个或多个驱动程序部件一构成特定驱动程序版本的一组有关的文件。INF 还包括即插即用（Plug－and plug）ID。在本发明的实施例中，在 CAB（“柜－Cabinet”）文件 420 中以压缩的形式提供实际的驱动程序文件。

下面是 INF400 的简单例子。

[Version]

Package＝{package GUID};识别包的内容

Driver＝mm/dd/yyyy：识别修订版（从包的其它版本区别此版本）

[Manufacture]

%FooCorp%＝Foo，NT.5.1，NT.6.0；这些是 Target OS Version Ids

[Foo]

;Min 2k 段

Copy Files＝@unidriv.dll ，

[Foo.NT.5.1]

<XP（NT5.1）段，不必要 butonhyuW2K，若与 WK 无区别，则能忽略>

[Foo.NT.6.0]

;NT6.0 段，被以前的操作系统版本所忽略

Install Package＝Infname；相对于当前 dir 参考的 INF

Manifest＝foo.manifest

示例性 INF 包括对三个各别的操作系统的指向。第三个（Foo.NT.6.0）利用有关从包加载文件的增加的机制。“Install Package 识别对于应与当前的包一起安装的另外包的 INF。组件（NET 部件是它的特定例子）是驱动程序的积木，并能包括多个文件，甚至其它组件。清单包含驱动程序的强名，和驱动程序的组件（如.NET 部件）的强名。

在本发明的实施例中，支持只包含 Unidrv 的包，且 INF 输送那个信息。在本发明的实施例中，这些需求由包括“CompName＝CompGUID”的段标签[PrintSoftware Components]实现。每个组件以段名 CompGuid 来描述。

例如：

[compGuid]

copyFiles＝@Unidrv.dll,etc.

DriveVer=mm/dd/yyyys 可选，若不同于[Version]段，则需要

包被存入在可配置位置的驱动程序包存储 300。默认地，驱动程序包存储驻留在本地机器（如集中的驱动程序存储 212）。然而在本发明的实施例中，驱动程序包存储 300 的内容使得能对其它连网机器可访问。例如，那样的可访问性通过将程序包存储 300 放在网络上其它机器发布的分布式文件系统共享部分上而达到。

转向图 4b，分层的树型图画出了如由 INF 文件 400 规定的在驱动程序包存储 300 中驱动程序容件（如程序包。驱动程序、部件）和文件的分层的安排/联系。注意到，在一个文件在不同的驱动程序部件使用多次的情况，在包内存储单个拷贝，且在包中存储到单个驱动程序拷贝的诸连接口，以节省存储文件的完整拷贝。

在共享驱动程序包存储 300 的最高层，驱动程序包容件 450 对应于共享目录的根部。在下面一层，一组包 GUID一限定的包容件 460，462 和 464 可能保持单独地识别的包（由对每个包的分别的 GUID 区分）的多个版本。在示例性实施例中，包容件保持两个包 470 和 472一每个在图 4b 中由赋给每个特定包版本的强名如程序包的日期）代表的目录结构中单独地识别。在本发明的实施例中。其中使用 GUID＋版本方法而不是强名，在图 4b 中画出的分层的树对应于下面目录安排：

Root{Package GUID1}\Package Data1\ ： 包含第一程序包的一个版本

Root\{Package GUID1}\Package Data2\ ： 包含第一程序包的新的版本

Root\{Package GUID2}\Package Data3\ ： 包含第二程序包的版本

……

Root{Package GUIDn}\Package Data4\ ： 包含第 n 程序包的版本

在本发明的实施例中，包数据格式是 yyyymmdd。在又一个示例性实施例中，为识别/区分驱动程序包版本使用强名，或甚至只用 GUID 值。

此外如图 4b 所示，除了包含如规定构成特定的驱动程序版本（由单独的识别符名识别）的组件的驱动程序清单那样非驱动程序组件与有的文件外，包 472 分别对驱动程序 A 和驱动程序 B 规定驱动程序组件专有的容件 480 和 482。另选地。给予每个驱动程序它自己的容件（但是在组件对包中的许多不

同驱动程序是公共的情况这是不希望的）。在本发明的实施例中，驱动程序组件专有的容件 480 和 482 由模型－和－提供者（model－and－Provide）GUIDMPG 的组合指定，后者区分如由不同驱动程序软件提供者分配的具有同一名字的驱动程序组件。驱动程序组件专有的容件 480（驱动程序 A）又包括对应于驱动程序组件 A 的版本 1 和 2 的两个版本容件（如部件）490 和 492。在本发明的实施例中，部件包含组合起来完成驱动程序功能的一个或多个文件的组。

在本发明的示例性实施例中，从其它驱动程序版本区分出驱动程序版本的驱动程序版本的至少一部分（如一组件）被存入分别的各自加标号的容件。此安排使连网外围设备客户能支持一个驱动程序的多个不同的版本。在本发明的特定实施例中，构成驱动程序组件的特定版本的文件以部件形式组合在一起。每个部件被赋予加不同标号的容件（如目录服务的子目录），使得与同一驱动程序的不同版本相关的部件被赋予不同的标号。

驱动程序组件（如部件）的版本通过用单独的识别符名对每个驱动程序部件容件加标号而区分，识别符名是任何合适的识别符，它在区分特定驱动程序的两个不同版本方面达到高的似然率。在特定示例性实施例中，单独的识别符名是.NET 强名。强名包括赋给构成特定驱动程序版本的至少一部分的有关的驱动程序文件（如一个部件）的组的版本日期。另选地，以用 a.b.c.d 格式显示的 64 位值的格式提供版本，其中 a.b 代表特征版本号，而 c.d 代表错误排除号。在又一个实施例中，强名是基于在从其它驱动程序版本区分出该驱动程序版本的信息上完成的校验和或散列操作。

转向图 5a，通常在网络中的服务器和客户上找到的活动的驱动程序存储 302 是保存加载的，解压缩的驱动程序文件 502 和清单信息 504 的目录结构。在本发明的实施例中使用两类清单：（1）组件清单描述组件的内容（如识别构成一个组件和任何其它需要的组件的文件），和（2）驱动程序清单描述驱动程序的组件（只是需要的组件，不受任何文件信息，因为驱动程序本身不具有不是组件的部分的任何文件）。在本发明的实施例中，每个驱动程序组件的每个版本在活动的驱动程序存储 302 中被分配，它自己的单独识别的容件（如子目录）。而且，活动的驱动程序存储 302 包括按组件的清单信息，它描述/列举保存在活动的驱动程序存储 302 中与每个特定的驱动程序组件版本（容件）有关的文件的组。

在本发明的实施例中，清单信息 504 还包括在由单独的识别符（对每个驱动程序版本）加标号的文件中的列表，识别符识别在活动的驱动程序存储 302 中构成驱动程序版本的组件。向每个特定的驱动程序版本提供它自己的单独识别的容件（如一文件），它只包含列举驱动程序版本的组件的驱动程序清单（不是其它文件）。在本发明的实施例中，驱动程序清单容件（文件）位于活动的驱动程序存储 302 中，或在另选的实施例中，位于某个另选合适的位置（包括在元数据存储 304 中的驱动程序版本表）。例如，对每个驱动程序/组件的容件的单独识别符名是一单独的名，它根据从驱动程序/组件的版本可提取的特征数据的任何组合创建，以保证同一驱动程序/组件的两个不同版本具有不同的容件名。如上提到，那样单独的识别符名的例子是：（1）强名。和（2）模型一和一提供者 GUID（MPG）和版本组合。

在本发明的实施例中，包含用于设置对特定驱动器的上下的版本信息的组件描述，在将特定的驱动程序组件版本存入活动的驱动程序存储 302 的时刻被存入清单信息 504 中。然后，使用特定驱动程序的服务或应用在加载驱动程序组件文件之前，根据清单信息 504 设置上下文。该上下文（如“激活上下文”）是在包含那样信息的存储器中的数据结构，系统使用该信息重定向应用，以加载特定的文件版本/部件。在此情况，它从驱动程序的清单文件创建。该上下文确定，对特定的驱动程序版本应使用可能的文件（如 D4）的许多版本的哪一个。例如该上下文能被管理员取代。

下面表示对赋给在活动的驱动程序存储 302 的驱动程序文件 502 中两个不同的子目录容件的特定驱动程序组件/部件（包括两个相关文件）的两个版本的两个清单信息 504 的输入项。

Assembly 1：Foo.dll(vx.y)

Fooi.dll(va.b)

Assembly 1’：Foo.dll(vx.y’)

Fooui.dll(va.b)

部件 1’具有 Foo.dll 和 Fooui.dll 两者的不同版本。在本发明的实施例中。在特定的驱动程序组件的两组文件之间的任何差别导致建立具有对应于驱动程序文件 502 中的子目录的它们自己的清单的不同的组件（部件）。

在本发明的实施例中，活动的驱动程序存储 302 的驱动程序文件 502 被保存在由网络上其它客户和服务器机器可访问的分布式文件系统共享部分中。

在那样的实施例中，在客户或服务器机器上的活动的驱动程序存储 302 与在机器的存储空间中其它非公共的组件隔离。在分布式文件系统共享部分中驱动程序文件 502 的安排对应于由清单信息 504 识别的部件。

转向图 5b，图示了示例性范例，其中本发明特别有用一即那时客户 540 保持到运行同一驱动程序（“Foo”）的两个不同版本（“x”和“y”）的第一服务器 550 和第二服务器 560 的两个不同的连结。在此图示情况，客户 540 利用其能力同时保持驱动程序 Foo555 的版本 x 和驱动程序 Foo565 的版本 y 活动，以达到在处理与对应的连网外围设备 570 和 580 有关的外围设备设置时与服务器 550（运行版本 x556）和服务器 560（运行版本 y566）两者的完全同时的兼容性。那样的设置分别继续存在于服务器 550 和 560 中的设置 557 和 567。继续存在的设置 557 和 567 被分别拷贝到保存在客户 540 的设置 558 和 568 之中。保持完全的驱动程序版本兼容性的失败引起客户 540 在解释服务器 550 和 560 上的设置 558 和 568 方面可能的无能。

转向图 6，数据库方案画出了示例的驱动管理工具软件的元数据存储 304 的示例安排。上面提到，元数据存储 304 保存在至少一台客户机上，并以有序方式概括了机器上加载的活动的驱动程序。元数据存储 304 不需要实现本发明。但是，它的存在提供了管理在网络环境中大量外围设备驱动程序的方便的方法。在本发明的实施例中，元数据存储 304 由假脱机程序服务管理。

在图 6 中概括的示例的数据库的方案如下地安排/访问。在包的表 600 在的输入项由 Compacklookup 函数 610 通过提交“PackagID”来引用。对 Package（包）表 600 中的输入项提供的子字段包括每个被识别的包的 PackageGUID，版本，和 INFName。

ComPackLookup 函数 610 还通过“ComponentID”访问在组件表 620 中的输入项。当列举组件时，它们包括 EnvID，ComponentGUID，ComponentName 和 Version（版本）。EnvID 字段提供能用作在 Environments（环境）表 630 上搜索关键字的值。Environments 表 630 的输入项包括对识别的环境（如 WindowsNT×86，WindowsNTIA64 等）的对应的 EnvName。

另外预期的表输入项搜索函数是 PhysDriverComponentLookup 函数 640。PhysDriverComponentLookup 函数 640 通过 ComponentID 搜索关键字访问 Component 表 620 中的输入项。PhysDriverComponentLookup 函数 640 还通过 PhysDriverID 搜索关键字访问 PhysicalDriver 表 650。在列举 PhysicalDriver 表

650 中具体的驱动程序输入项时，它们包括逻辑 driverID，EnvID，和 languageID（语言 ID）。EnvID 字段提供用于搜索 Environment 表 630 的值。此外，逻辑 driverID 字段提供用于搜索 logicalDrivers 表 660 的值。在本发明的实施例中，PhysicalDrivers 表 650 还存储对每个单独识别的具体的驱动程序/版本组合的驱动程序清单表。

因此，在本发明的实施例中，逻辑驱动程序指的是从对特定平台编辑的驱动程序提取的或限定于特定用户语言（如英语）的驱动程序。具体的驱动程序附属于对其已进行编译并可能对其已限定语言的特定识别的环境（如“WindowsNT×86”，“WindowsNTIA64”）。被识别的具体的驱动程序可能包括若干组件（如 HPLaserJet4 Customization，sRGB Profile and Unidrv）。注意，不是所有组件具有对特定环境（如 Image Color Management（ICM）profiles，or GPD（一般的打印机描述））的依赖性。

LogicalDriver 表 660 的输入项包括驱动程序名，制造商 ID，plug－n－play ID，版本，和 providerID。制造商 ID 转而又参考制造商（Manufacturers）表 670 的输入项。例如制造商表 670 的字段包括名字字段和 URL 字段。

在本发明的实施例中，元数据存储 304 数据库在提供驱动程序存储的服务的首次自举时被创建。在本发明的实施例中，打印机假脱机程序提供那样的服务。在另外范例中使用专用的驱动程序管理服务。元数据存储 304 数据库从与操作系统的拷贝一起提供的 INF 内容填入。另选地，将预填入的数据库加载到该构造中一以复杂化该构造的处理为代价（因为在数据库中的信息对在 INF 中的信息是冗余的，必须有新的构造后的步骤）。

关于如上解释的包的表，PachageID（由包的 INF 提供）不能可靠地使用来区分基本上同一个包的两个版本，因为作为规则，PackageID 不随整个包的每次改变而改变（只要 INF 不改变）。相反，一旦包被创建，由 INF 设置的 PackageID 不（在版本之间）改变。因此，存储在包中的 DriverVer 输入项被用于区分具有同一 PackageID 值的包的两个版本。

讨论了由网络环境中客户使用的外围设备驱动程序维护工具软件的底层组织之后，注意力转向使用这里描述的底层架构实现的系统和示例方法的功能性界面。开始转向图 7，对加载/安装的包和驱动程序识别一组示例的（打印机驱动程序有关的）应用程序界面（API）。使用如远程过程调用帮助程序库实现 API。结果，各功能不需由所有外围设备提供者实现。

Upload Printer Driver Package（）API700

Upload Printer Driver PackageAPI700 便于上载打印机驱动程序包。Upload Printer Driver PackageAPI700 是异步 API。下面是 Upload Printer Driver PackageAPI700 的示例格式。

```
HRESULT
Upload Printer Driver Package（
IN LPCTSTR          psz Server，
IN LPCTSTR          psz INFPath，
IN DWORD            dw Flags，
IN HANDLE           hEvent，
OUT HRESULT*        pError，
OUT DWORD*          pdw CancelId，
OUT PDDRIVER_PACKAGE_ID    pPackageId
）；
```

其中下面说明引用的参数：

psz Server 是加载该包的服务器的全民。若服务器是集群的，则此参数以集群名开始。

Psz INFPath 是全路径和 INF 文件名，后者被处理以产生打印机驱动程序包。INF 的处理发生在客户端以避免若 INF 文件不能从服务器访问引起的问题，如当 INF 文件在 CD 上，而 CD 在客户机上未被共享出来。分析 INF 文件并准备所有参考的的文件的表。每个文件以全路径出现在表中。该表被送到服务器，而在那里上载对应的文件。通过读出下面在服务器上注册关键字的 DriverStorageShareName 值，找出在服务器上的上载包的位置：

SYSTEM\\Current Control Set\\Control\\Print\\Driver Storage Root一它能如通过用户界面由打印服务员管理员配置。如上解释，每个包被赋予单独加标号的子目录。在本发明的示例实施性中，对上载的包产生 GUID，且使用包的 GUID 在 DriverStrorgeShareName 下创建一目录名。在该目录下创建子目录，它具有下述格式：YYYYMMDD，其中 YYYY,MM，和 DD 是包的当前版本的年，月和日期。所有包文件在该子目录下被拷贝。由包的 INF 文件提供的结构管理包的目录的树结构。

DwFlags 提供用于修改 API 的行为的机制。例如，若 dwFlags 的值等于 0

×01，则若该包已存在于服务器上，在上载期间它将被改写/替代。Dwflags 参数还对该包设置平台。

hEvent 是由调用者提供的事件处理程序。当上载作业完成时，通知对应的事件。若 hEvent 是空，则调用被转换成同步的。

pError 是到 HRESULT 的指针，后者是写从异步调用返回的值的地方。在 hEvent 被通知以后该值有效。

pdwCancelId 是到 DWORD 值的指针，它能用于通过对 UploadCancel API（下面讨论）的调用删除异步的 Upload Printer Driver Pachage。在返回 Upload Printer Driver Package 之后，pdwCancelId 的值立即生效。

PpackageId 是到下面类型的 PackageId 结构的指针：

```
Typedef struct_DRIVER_PACKAGE_ID
{
DWORD        dw Size;
GUID          DpackageGuid;
SYSTEMTIME    DpackageTime;
}DRIVER_PACKAGE_ID,*PDDRIVER_PACKAGE_ID;
```

当异步调用完成，它指向的结构将包含该包的 GUID 和时间。使用接收的 DRIVER_PACKAGE_ID 给包的子目录加标号。

InstallDriverFromPackage（）API710

InstallDriverFromPackage（）API710 安装来自安装的包的驱动程序。标志参数规定拟安装的驱动程序的类型（打印机驱动程序，端口监视器，打印处理器等）。安装方法更新了元数据存储 304 和清单信息 504 以反映新安装的驱动程序。就安装的驱动器的组件尚未出现在活动的驱动程序存储 302 的情况而言，新的子目录被加入，新的组件被存入新组件的容件中。下面是 InstallDriverFromPackage API710 的示例格式。

```
H RESULT
WINAPI
InstallPrinterDriverFromPackage（
  IN LPCTSTR          psz Server，
  IN DRIVER_PACKAG_ID      PackageID，
  IN LPCSTR           psz Model Name，
```

```
    IN LPCSTR        psz Evironment
    IN DWORD         dw Version,
    IN DWORD         dw Flags
);
```

在上面概括的 API 中：

PszServer 规定一服务器名。使用 NUL1 值从本地的包存储安装驱动程序。注意在本发明实施例中，驱动程序包已经用 Upload Printer Driver Package API700 安装在由识别的 pszServer 定义的机器上。

PackageID 定义了安装的包，我们从该包安装打印机驱动程序。对 PackageID 的值由 Upload Printer Driver Package API700 返回。

PszModelName 是打印机驱动程序的名。被识别的打印机驱动程序必须存在于由 Package 定义的包中。

PszEvironment 是拟安装驱动程序的环境。在本发明的实施例中使用字符串名。那样字符串名的例子包括“WindowsNT×86”，“WindowsIA64”和“WindowsNT AMD64”。

DwVersion 规定拟安装的驱动程序的版本。

DwFlags 便于规定对 API 默认行为的客户化。示例的标志能从众知的 AddPrinterDriverAPIcounterpart 推导出来，并包括：

IDFP_STRICT_UPGRADE

IDFP_STRICT_DOWNGRADE

IDFP_COPY_ALL_FILES

IDFP_COPY_NEW_FILES

IDFP_DON'T_COPY_FILES_TO_CLUSTER

IDFP_COPY_TO_ALL_SPOOLERS

IDFP_NO_UI

IDFP_INSTALL_WARNED_DRIVER

在 AddPrinterDriver 中的某些标志的 InstallDriverFromPackage API710 中没有付本。这些是：

APD_COPY_FROM_DIRECTORY－因为我们没有传送－DRIVER_INFO_6 结构，那是不需要的。

APD_RETURN_BLOKING_STATUS_CODE－由于没有低层客户调用此函

数，此 API 总是返回成块码。

APD_DON'T_SET_CHECKPOINT－因为若试图安装未签署的驱动程序，此 API 总是设置校验点。

在本发明的实施例中，标志组包括给予调用者关于从 INF 中安装哪些组件的更大的控制的那些。这些标志包括：

IDFP_INSTALL_PRINTER_DRIVER

INFP_INSTALL_LANG_MONITOR

INDP_INSTALL_PRINT_PROC

IDFP_INSTALL_ICM_PROFILE

IDFP_INSTALL_ASP_PAGES

未签署的驱动程序由 InstallDriverFromPackage API 710 如下地处理：

• 若安装远程地发生，它失败。

• 若用户不是管理员，它失败。

• 若用户是管理员，类似于 PnP 用服务器方/客户方安装的那样，在用户的上下文重试安装。

GetDriverStorageRoot720

调用 GetDriverStorageRoot 函数寻找服务器存储驱动程序包存储 300 的驱动程序包的根位置。在本发明实施例中，包存储的根的位置通常由管理员设置。例如，GetDriverStorageRoot720 函数的格式如下：

```
HRESULT
GetDriverStorageRoot（
IN LPCTSTR     psz Server，
OUT LPTSTR*    ppDslrorageRootBuffer
）；
```

pszServer 是驱动程序包存储 300 驻留的服务器的全民。若服务器是集群，则全民用集群名开始。

PpDStorageRootBuffer 是指向拟写缓冲器地址的指针的指针，该缓冲器包含到驱动程序存储的根的路径。在本发明的实施例中，路径是零结尾的。对该缓冲器的存储器由 API 分配。调用者通过调用 Local FreeAPI 响应空出缓冲器。

GetPrinterPackageInfo 730

GetPrinterInfo730 函数使客户能请求列举包信息，下面是 GetPrinterPackageInfo 730 函数的示例格式：

```
HRESULT
GetPrinterPackageInfo(
[in,string,unique]LPCTSRT     psz ServerNme,
[in]   PQUERY_CONTAINER   pQuery Container,
[out]   DWORD              *pc GUIDs//GUID count
[out]   GUID               **pp Guid Array,
[out]   SYSTEMTIME         **pp Version Array
);
```

Upload Cancel 740

UploadCancel 740 函数取消以前引用的对 UploadPrinterDriverPackageAPI700 的异步调用。下面是 UploadCancel740 函数的示例格式：

```
HRESULT
WINAPI
Uploadcancel（
In DWORD     dwCanceld
);
```

dwCancelId 是由拟被取消的 UploadPrinterDriverPackageAPI700 的调用返回的识别符。DwCancelId 参数的有效值是正的。若异步调用能被取消，该 Upload Cancel740 函数返回 S_OK。在此情况，Upload Printer Driver Package API700 清除，且随后通知对应的事件。若异步 API 不能被取消或若 dwCancelId 的值无效，则 UploadCancel740 函数返回 E_FAIL。

IprintSetup750

在本发明的实施例中，IPrintSetup750 界面在如从假脱机驱动程序（如 Winspool.drv）使用的单个 COM 界面中展示所有新的界面。Iprintsetup750 界面是所有新函数的封装处理程序。

OpenDriver（）760

OpenDriver( )760 函数对每个驱动程序建立标识值。下面是 OpenDriver()760

函数的示例格式。开始提供数据结构。然后概括示例方法格式。

```
Struct LOGICAL_DRIVER_ID
{
GUID      Model Prov Guid;
HYPER   Version;//HYPER 是类型 64 位有符号整数
}
enum eEvironment
EnvX86,
EnvIA64,
EnvAMD64,
EnvAlpha,
EnvWin9x
}
Struct PHYSICAL_DRIVER_ID
{
LOGICAL_DRIVER_ID     logDriver;
eEvironment              Env;
DWORD              cVersion;
LANGID              LanguageId;
}
struct DRIVER_OPTIONS
{
DWORD        cbSize;
DWORD        dwFlags;
}
HRESULT
OpenDriver(
IN            PCTSTR     pszServerName,
IN            PHYSICAL_DRIVER_ID        *pDriver,
IN   OPTIONAL DRIVER_OPTION          *pOptin,
OUT        HANDLE                      *pHandle
```

);

SetDriver()770

SetDriver()770 函数便于设置与驱动程序有关的特定参数。示例的 SetDriver（）770 函数具有如下格式：

```
HRESULT
SetDriverData(
[in] HANDLE   hDriver,
[in]PTSTR   psz DataName,
[in]DWORD Bufsize
[in]VOID    *pBuf
);
```

GetDriverData()780

GetDriverData()780 函数使能检索与驱动器有关的特定参数。示例的 GetDriverData（）780 函数具有下列格式：

```
HRESULT
GetDriverData(
[in]HANDLE          hDriver.
[inPTSTR           pszDataName,
[in]DWORD          BufSize,
[out]VOID          *pBuf,
[out]DWORD         *pReqBufSize
);
```

CloseDriver()790

CloseDriver（）790 函数使能释放以前打开的驱动器，从而使安装驱动程序的特定客户上的客户不活动。示例的 CloseDriver（）790 函数具有下列格式：

```
HRESULT
CloseDriver(
[in]HANDLE    hDriver
).
```

转向图 8，概括一组步骤，用于安装一个包（如打印机驱动程序包）到具有如图 4 画出的逻辑结构类型的计算机系统的驱动程序包存储 300。如上解释，

在安装包时，不是只安装包的选择部分，而是在特定的包容件上安装整个包。最初在步骤 800，保存驱动程序包存储 300 的计算机系统接收安装完整的目标包到驱动程序包存储 300 的调用（如 UploadPrinterDriverPackage700）。作为响应，在步骤 810 期间从拟安装的包提取区分信息，在本发明的实施例中，在步骤 810 期间安装程序将来自拟安装的包中的 INF 文件的包 GUID 和版本读到驱动程序包存储 300。

接着在步骤 820 期间，安装程序访问包容件 450，并判断在步骤 810 提取的包 GUID 是否作为在驱动程序包容件 450 中的一个包 GUID 容件名出现。若提取的包 GUID 未出现，则控制转到步骤 830，在那里创造新的包 GUID 容件并赋予提取的包 GUID。于是，在本发明的示例实施例中，包 GUID 容件保持具有同一包 GUID 的所有包的版本。控制然后转到步骤 840。

在步骤 840 创建一版本容件，以保持驱动程序包的特定版本。在本发明的实施例中，版本容件按在步骤 810 提取的版本加标号。例如，版本是赋给驱动程序包的日期，或者更规格化的版本名包括指定主发行，子发行，和错误排除发行的多个级联的数字字符值。在特定实施例中，版本包括 4 字节的粒度（granularity）。

在创建了保存包的新的版本容件后，在步骤 850 期间包被存入新的版本容件。存储的包的格式随选择的本发明的实施例而变。然而在特定实施例中，在步骤 850 期间驱动程序文件从安装的原始包的 CAB 文件提取。至少在某些范例中，驱动程序的那样的提取包括解压缩该文件。按照由 INF 文件对安装的包定义的部件创建容件/子目录，它在文件系统（如子目录）位置存储提取的驱动程序，此位置在第一层由模型及提供者的 GUID（MPG）确定，随后在第二层由版本确定。例如，版本由日期或分段在字符序列指定－以类似于存储包的版本的方式。另选地，驱动程序被存储在由强名识别的容件中。

如图 8 的步骤 860 表示，在存储文件到驱动程序包存储 300 的目录结果之后，描述包括其驱动程序的存储的包的元数据被加到元数据存储 304 数据库。保存那样的数据库便于相当快地搜索集中的驱动程序存储 300 的文件内容。注意，步骤 860 实际上没有依赖人能和以前特定的步骤，因而能发生在安装驱动程序包的任何时刻。然而通过一直等到包确实安装在驱动程序包 300，元数据存储的内容的准确性被保证。控制随后转到结束（End）。

回到步骤 820，若提取的包 GUID 在驱动程序包容件 450 中被识别（表明

对现有的包的 GUID 容件的匹配），则控制转到步骤 870。在步骤 870，包版本（强名）与现有的包版本比较。若提取的包版本不匹配在包的 GUID 容件中识别的包版本，则此识别的包特定的版本尚未存在于驱动程序包存储 300 中，控制转到 840。另一方面，若提取的包版本匹配在包的 GUID 容件中识别的包版本，则该包的特定的版本已存在于驱动程序包存储 300 中，控制转到 880。

在步骤 880，为安装程序提供改写该驱动程序的当前的，认为是同一版本发选项。若希望改写，则控制转到步骤 850。在匹配包版本容件标号下的以前现存的包的版本被替代。若不希望改写，控制转到结束。在结束时注意，上述包装方法是以便于区分包的版本和包中驱动程序的版本的方式存储包的过程的示例。本专业的行家容易理解，看到这里揭示内容后，诸步骤能修改及重新安排而同时保持区分安装的版本的能力。

转向图 9，概括了一组步骤，为客户和服务器建立对连网外围设备的可兼容驱动程序。在示例步骤中，客户以使得驱动程序的不同版本被分别保存的方式，从驱动程序包存储 300 中，将驱动程序安装到客户计算机系统的活动的驱动器存储 302 中。每个驱动程序版本与单独识别的存储在活动的驱动程序存储 302 中的组件的表（如清单）相关。对每个单独识别的组件，一个清单存储在活动的驱动程序存储 302 中。此外，每个驱动程序清单/表被存储在对应于保存的客户上的元数据存储 304 的 Physical Drivers 表 650 中特定驱动程序版本的一个输入项中。

存在若干方法，预期能用于同时支持包括具有同一名字的不同文件的同一驱动程序的多个分别的版本。在本发明的实施例中通过将每个活动的驱动程序版本的所有组件/文件存储到单独识别的文件系统容件（如子目录），达到那样的隔离。然而，许多驱动程序使用特定的驱动程序组件（如 unidriver），而对每个驱动程序/版本组合保存拷贝将浪费存储空间。因此，本发明的另外实施例对在单独识别的子目录容件中的特定的驱动程序存储不完整的文件组。替代地，从存储活动驱动程序存储 302 的多个子目录中的组件创建完整的驱动程序。感觉对在元数据存储 304 中的驱动程序的清单的内容构造完整的驱动程序。

按本发明的各个实施例区分在特定子目录中合并文件/组件的层。在一个实施例中，每个包括一个或多个文件的组件版本被赋予在活动的驱动程序存储 302 中的一个子目录。在另一个实施例中，对每个驱动程序版本建立单独识别

的子目录，而该子目录包含与那个特定驱动程序相关的非共享组件/文件。驱动程序版本的共享组件包含在活动的驱动程序存储 302 的其它位置。

在本发明的特定实施例中，保含驱动程序文件的子目录被存储在推广的目录下，后者存储包括应用软件的有关软件模块的组。随后通过对特定驱动程序版本存储在清单信息 504 或元数据存储 304 清单的组件清单，向依赖于对特定驱动程序版本的驱动程序文件的客户软件提供赋给每个子目录的单独的识别符名（如强名）。而且在本发明的实施例中，包含各种驱动程序组的子目录使得其它机器能从客户机读出或下载驱动程序一称为“共享出来”的目录。下面（对连网打印机驱动程序）描述的一组步骤假设，客户和该客户寻求建立连接的服务器均支持上述包和驱动程序版本区分的能力。

在步骤 900 中，客户机请求从打印服务器请求识别特定的设备驱动程序。例如，此请求是 GetPrinterDriver 或 GetDriverData 调用。接着在步骤 910，打印服务器向请求的客户返回足以在由驱动程序包存储 300 保存的驱动程序文件系统结构中定位特定驱动程序版本的驱动程序识别信息。在本发明的特定实施例中，打印服务器返回包 GUID 和版本，和驱动程序 MPG 和版本。在本发明的示例实施例中，此信息足以在元数据存储 304，活动的驱动程序 302，或驱动程序包存储 300 文件系统中规定一驱动程序版本。而且 MPG 和版本足以使请求的客户能通过访问元数据存储 304 或活动的驱动程序存储 302（它也包含驱动程序版本和相关的清单）判断，所需要的打印驱动程序是否已经在客户机的活动的驱动程序存储 302 中。

接收了识别信息后，在步骤 920 中客户机判断，识别的驱动程序版本（如 MPG 和版本）是否已经在其活动的驱动程序存储 302 中。若在服务的响应中识别的驱动程序版本在客户的活动的驱动程序存储 302 中，则客户具有为完成与服务器的全兼容互相作用所必需的驱动程序组件，因而控制转到结束。然而若识别的驱动程序未出现，控制从步骤 920 转到步骤 930。在那样客户机判断，识别的包 GUID 和版本是否在已知位置可得到（本地的或在另外机器上某个已知的共享出来的位置，如中央驱动程序存储 212）。若识别的包 GUID 和版本不对应于客户可到的包，则控制转到步骤 940。

在步骤 940 客户机确定在网络中识别的包和版本的位置。在本发明的示例实施例中，客户机向驱动程序包存储 300 发出请求。例如，驱动程序包存储 300 通过打印服务器（或中央驱动程序存储 212）的服务提供，后者保存所有驱动

程序包和它们相应位置（由共享名识别）的表。因此在步骤940，驱动程序包存储300返回如\\server（服务器）\share name（共享名），它由客户使用，请求在步骤910中识别的包版本的拷贝。应注意，本例子利用包版本的识别而不是驱动程序版本的识别。然而在驱动程序被共享出来的本发明另选的实施例中，客户能够请求对特定的驱动器版本的位置。在接收了请求的网络位置后，控制转到步骤950。

在步骤950中，客户请求从包（或者驱动程序）的版本的已知位置上载已识别的该包的版本。例如，上载的版本包括服务器名，共享名，packageGUID和版本。在接收了请求的驱动程序包版本后，客户机按照如参考图8描述的步骤在其本地的驱动程序包存储300中安装该包。控制随后转到步骤960。

在步骤960中，客户机引用方法/函数，从包存储300安装在步骤910中识别的驱动程序版本到活动驱动程序存储302。在步骤960，在活动的驱动程序存储302中创建一个或多个新的子目录容件，用于存储安装的驱动程序的组件，为新的子目录提供单独的识别符名（如MPG和版本），在步骤910识别的对应于该驱动程序版本的驱动程序/组件版本的驱动程序文件从该包拷贝到新的子目录。随后控制转到步骤970，在那里更新活动的驱动程序存储302的清单信息，使包括与新安装的驱动器版本相关的任何新清单（包括存储列出构成新驱动程序版本的组件的驱动程序清单的新文件），且在本发明的实施例中，在对应于新的活动的驱动程序的元数据存储304中创建新的输入项。新的输入项/清单信息包括通过单独识别符名列出在构成新安装的驱动程序版本的活动的驱动程序存储302中的组件的驱动程序清单。通过如在对每个新的驱动程序版本的元数据存储304中创建新的清单信息/驱动程序输入项，并在单独识别的子目录中存储驱动程序组件/文件的不同版本，上述方法避免了改写包含同一名字但有不同内容（如同一驱动程序文件的不同版本）的任何驱动程序文件。然后控制从步骤970转到结束。

回到步骤930，若识别的包对客户（本地的或在已知的文件共享上）可得到，控制转到步骤980。在步骤980，若包的版本不存储在本地，控制转到步骤950。否则若包本地能得到，控制转到960。且在步骤910中识别的驱动程序版本被安装在客户的活动的驱动程序存储302中。

在讨论了驱动程序版本如何识别并存入活动的驱动程序存储302之后，在图10中概括了一组步骤，使处理来自于活动的驱动程序存储302的特定驱动

程序版本的打印客户，与使用当前存储在客户的活动的驱动程序存储 302 的，已知的驱动程序版本操作的特定的打印服务器交互作用。

最初在步骤 1000，客户应用程序请求使用服务对象，如对于由特定打印服务器支持的打印机的打印队列。对一个打印队列，那样的请求必须是对由打印队列驱动的打印机的可打印区域，此类数据通过封装程序（wrapper）层（这是操作系统及其 API 的部分）从打印驱动程序提供。那样的请求通过如由文字处理应用的打印请求而起动。接着在步骤 1010，操作系统查找对特定打印队列所需要的正确的打印机版本（此打印驱动程序版本当前被与当前的队列有关的打印服务器使用）。这能以若干已知的对一队列确定所需打印机驱动程序的方法的任一种来实现。示例的实施例通过确定驱动程序的版本加上该驱动程序本身的一般识别建立以前已知的版本。

接着在步骤 1020，打印系统从用于运行特定驱动程序版本的驱动程序的清单创建激活的上下文。该激活上下文为操作系统提供搜索路径，以按照单独识别的驱动程序版本访问驱动程序组件的版本。该上下文引导用于查出识别的驱动程序版本的组件的位置的次序。因此，若应用程序（在此例中是操作系统的打印子系统）请求 Unidrv 而未规定版本（换言之，若它未规定强名而只是其一部分），激活的上下文提供引起加载特定版本的信息。提供上下文使得能仅仅通过切换从中导出该上下文的驱动程序清单而从一个驱动程序组件到另一个地切换应用程序（如切换到共享组件更新版本）。总之，激活一上下位对应于激活与特定驱动程序版本相关的特定搜索路径。

在创建激活的上下文之后，在步骤 1030 系统根据激活的上下文加载与特定驱动程序版本有关的驱动程序组件，然后调用该驱动程序来完成与特定驱动程序版本相关的命令/操作。

接着在步骤 1040，驱动程序能加载附加的文件（如配置文件，帮助程序 DLL）。若这样做，按照在驱动程序清单中规定的位置/路径得到那些组件的版本。，在本发明的实施例中，驱动程序的组件也能包括定义附加驱动程序的文件/组件的清单。这些附加文件也由版本信息加上使得在客户机上完成的特定驱动程序版本能匹配服务器机器的版本的 GUID 来识别。这些版本由当前激活的上下文（由驱动程序及其组件的清单驱动）来操纵。

最后在步骤 1050，在完成提交给驱动程序的请求后，打印系统抑制该上下文，控制转到结束。注意，虽然本例子参考网络打印驱动程序给出，本发明

能应于其它共享的连网外围设备，包括传真机，扫描仪等。

本领域的技术人员理解，已经描述了新的有用的方法和系统，用于维护在连网外围设备环境中的外围设备驱动程序和它们的组成组件。看到了能应于本发明的原则的许多可能的环境和设计及完成软件实用程序及工具程序的灵活性，应该认为这里描述的实施例仅是示例性的，不看作对本发明的范围的限止。应用本发明的专业的行家认识到，诸实施例能在安排及细节上加以修改而不背离本发明的精神。因此，这里描述的本发明预计，所有那样的实施例能落入下面权利要求及其等价物的范畴。

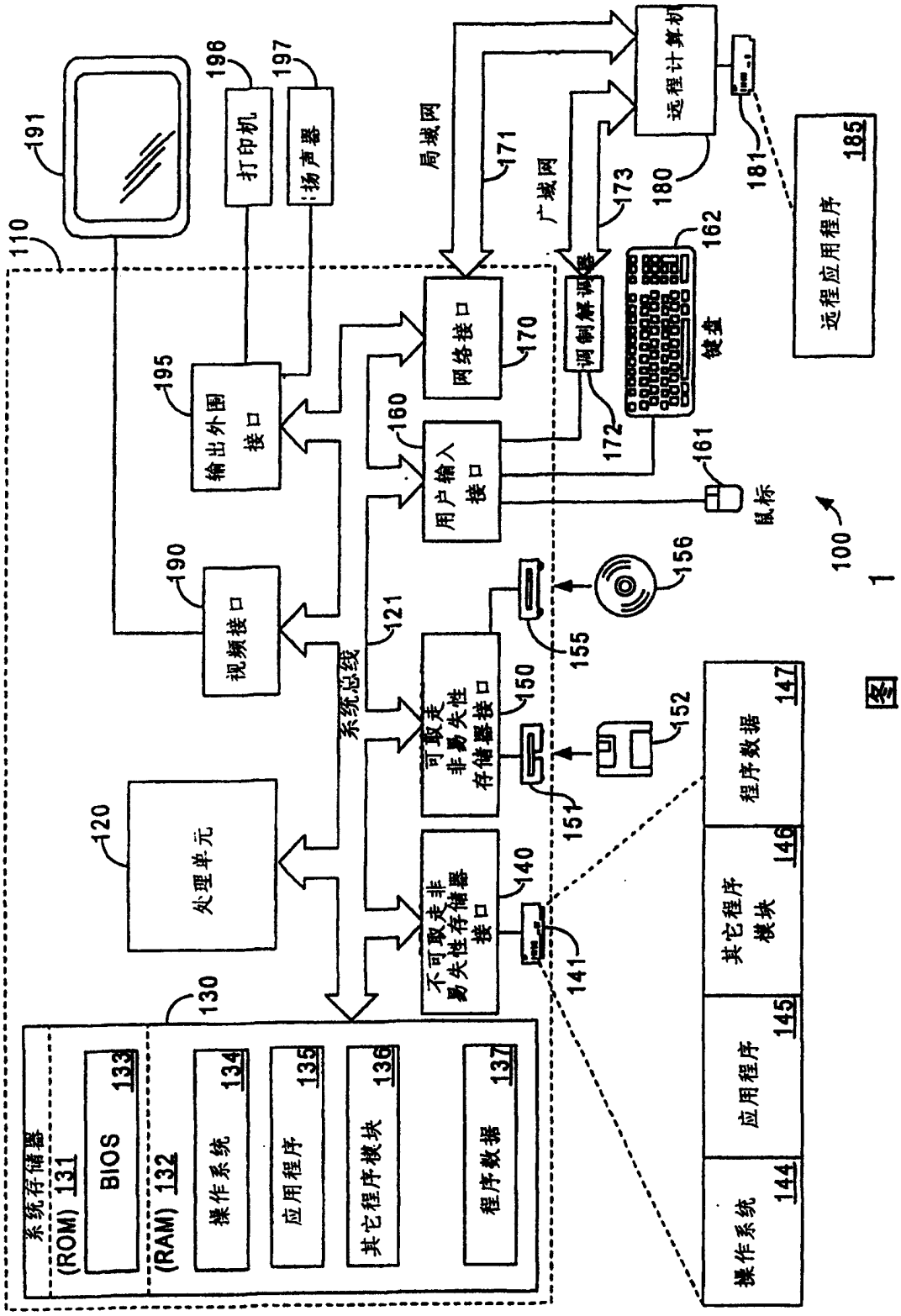

系统存储器
(ROM) 131
BIOS 133
(RAM) 132
操作系统 134
应用程序 135
其它程序模块 136
程序数据 137
处理单元 120
不可取走非易失性存储器接口 140
可取走非易失性存储器接口 150
视频接口 190
输出外围接口 195
网络接口 170
用户输入接口 160
调制解调器 172
系统总线 121
局域网 171
广域网 173
远程计算机 180
远程应用程序 185
打印机 196
扬声器 197
键盘 162
鼠标 161
100
110
130
141
151
155
156
152
181
操作系统 144
应用程序 145
其它程序模块 146
程序数据 147

图 1

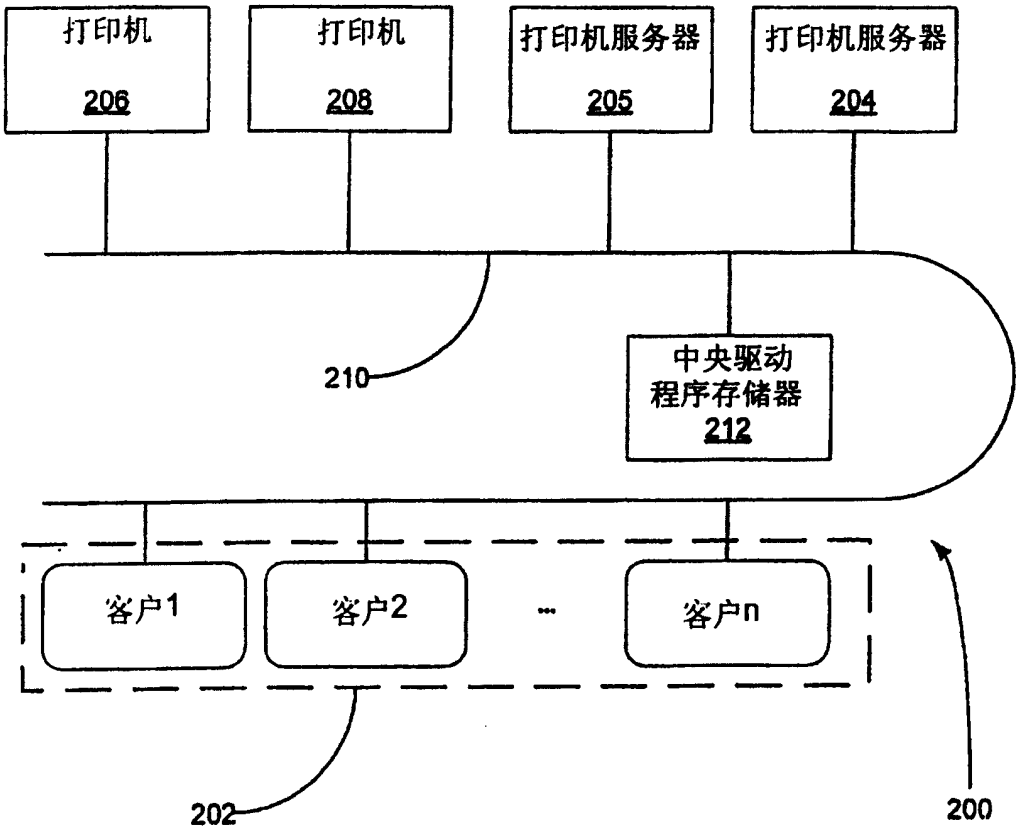

打印机
206
打印机
208
打印机服务器
205
打印机服务器
204
210
中央驱动
程序存储器
212
客户1
客户2
...
客户n
202
200

图 2

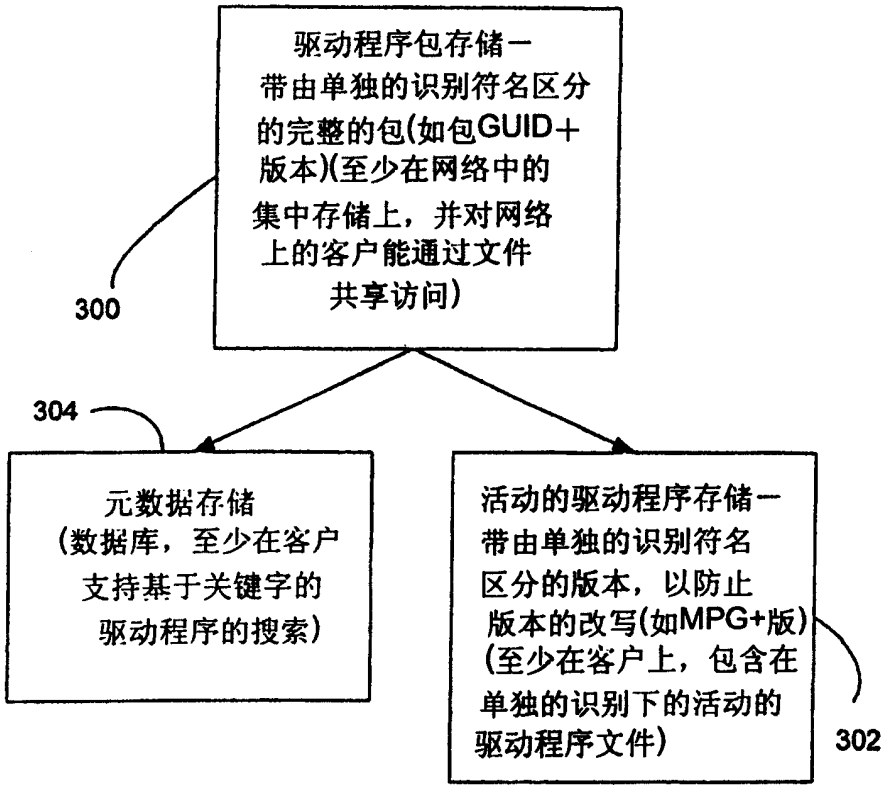

驱动程序包存储一
带由单独的识别符名区分
的完整的包(如包GUID+
版本)(至少在网络中的
集中存储上，并对网络
上的客户能通过文件
共享访问)
300
304
元数据存储
(数据库，至少在客户
支持基于关键字的
驱动程序的搜索)
活动的驱动程序存储一
带由单独的识别符名
区分的版本，以防止
版本的改写(如MPG+版)
(至少在客户上，包含在
单独的识别下的活动的
驱动程序文件)
302

图 3

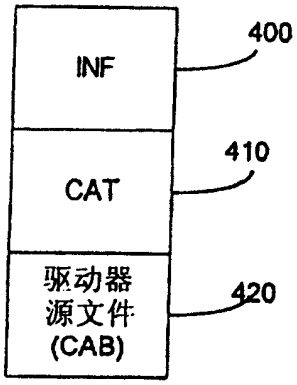

INF
400
CAT
410
驱动器
源文件
(CAB)
420

图 4a

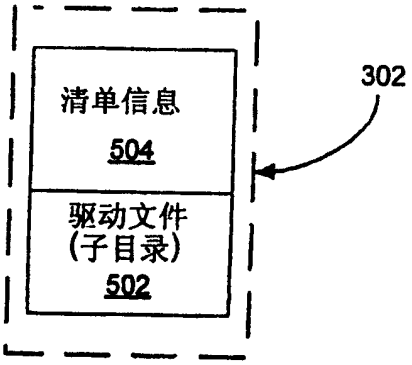

清单信息
504
驱动文件
(子目录)
502
302

图 5a

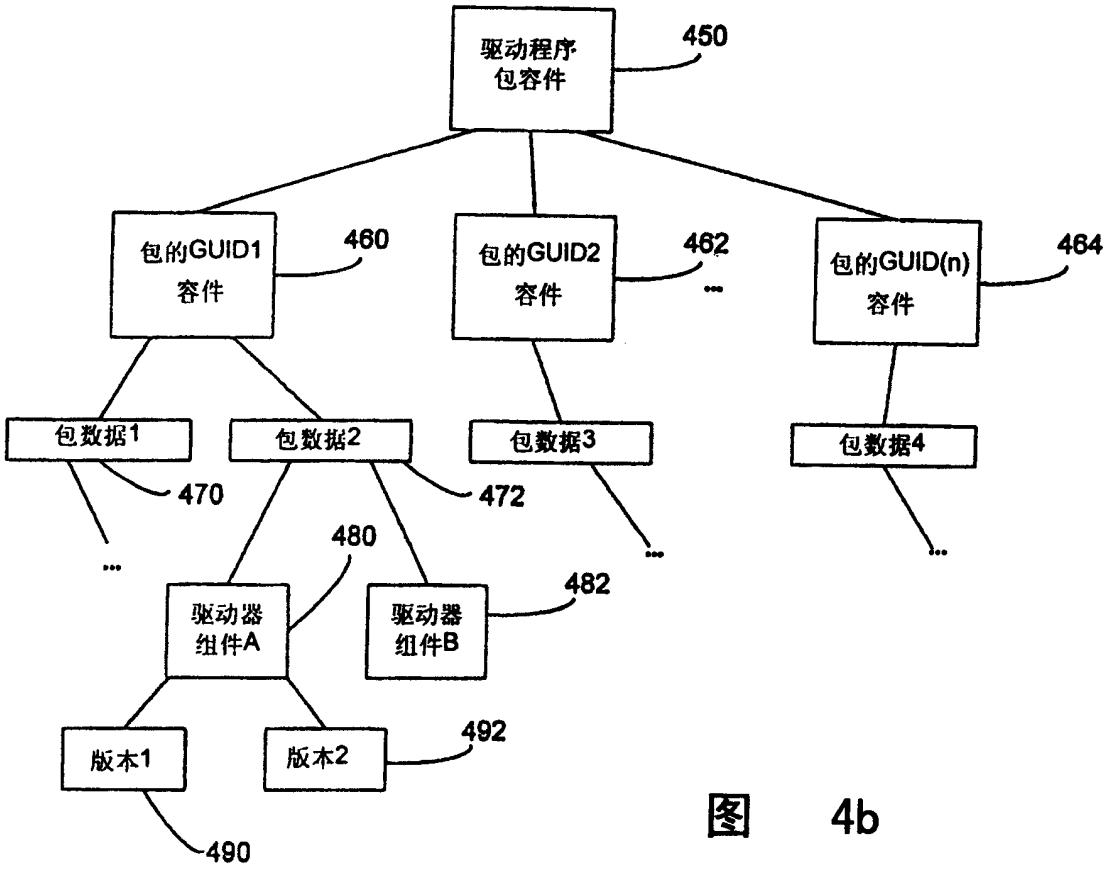

驱动程序
包容件
450
包的GUID1
容件
460
包的GUID2
容件
462
包的GUID(n)
容件
464
包数据1
包数据2
包数据3
包数据4
470
472
480
482
驱动器
组件A
驱动器
组件B
版本1
版本2
492
490

图 4b

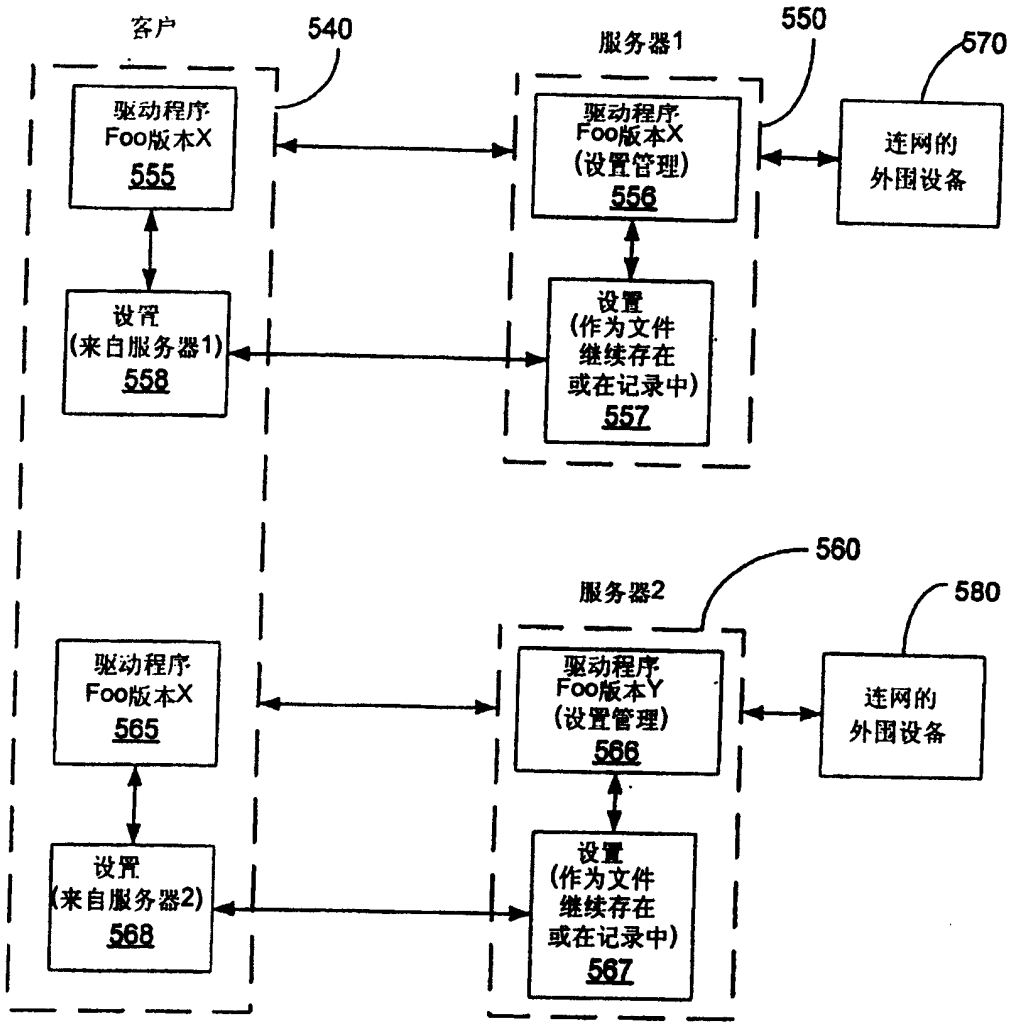

客户
540
服务器1
550
570
驱动程序
Foo版本X
555
驱动程序
Foo版本X
(设置管理)
556
连网的
外围设备
设置
(来自服务器1)
558
设置
(作为文件
继续存在
或在记录中)
557
560
服务器2
580
驱动程序
Foo版本X
565
驱动程序
Foo版本Y
(设置管理)
566
连网的
外围设备
设置
(来自服务器2)
568
设置
(作为文件
继续存在
或在记录中)
567

图 5b

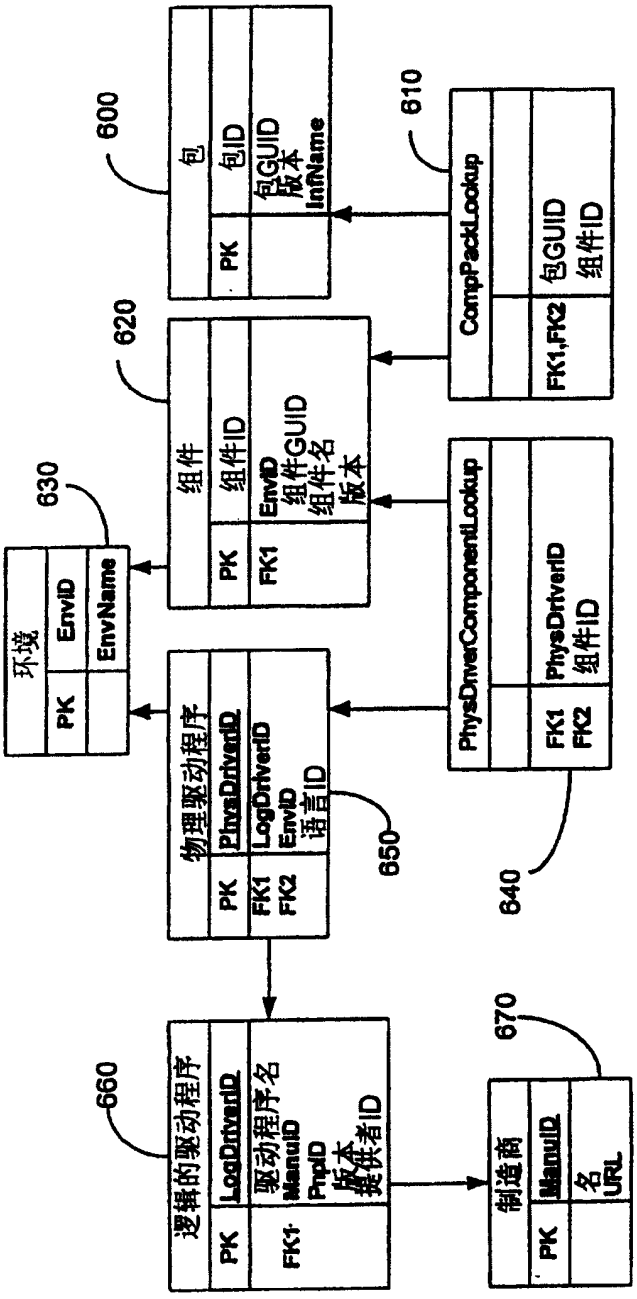

600
包
PK
包ID
包GUID
版本
IntName
610
CompPackLookup
FK1,FK2
包GUID
组件ID
620
组件
PK
FK1
组件ID
EnvID
组件GUID
组件名
版本
630
环境
PK
EnvID
EnvName
物理驱动程序
PK
FK1
FK2
PhysDriverID
LogDriverID
EnvID
语言ID
650
PhysDrvrComponentLookup
FK1
FK2
PhysDriverID
组件ID
840
660
逻辑的驱动程序
PK
FK1
LogDriverID
驱动程序名
ManuID
PrpID
版本
提供者ID
670
制造商
PK
ManuID
名
URL

图 6

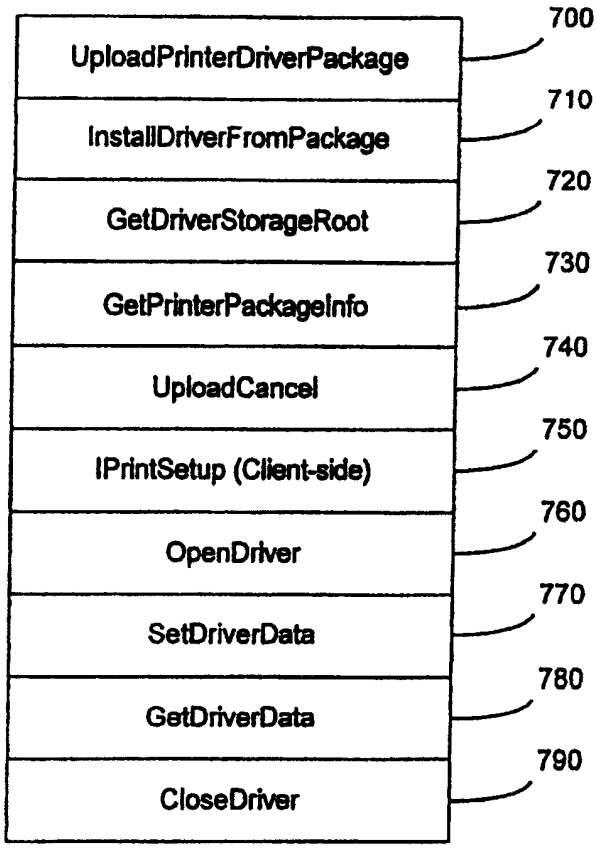

UploadPrinterDriverPackage
700
InstallDriverFromPackage
710
GetDriverStorageRoot
720
GetPrinterPackageInfo
730
UploadCancel
740
IPrintSetup (Client-side)
750
OpenDriver
760
SetDriverData
770
GetDriverData
780
CloseDriver
790

图 7

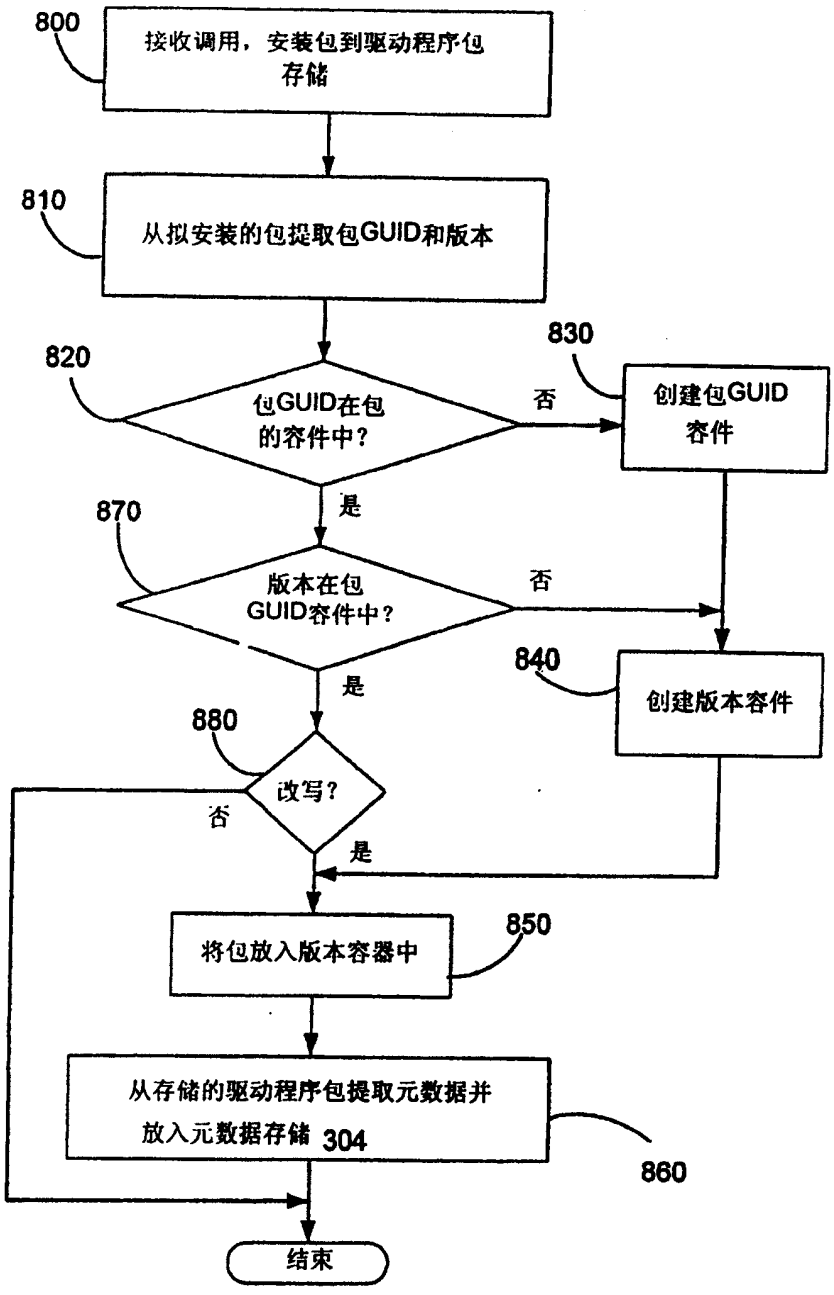

800
接收调用，安装包到驱动程序包存储
810
从拟安装的包提取包GUID和版本
820
包GUID在包的容件中？
否
830
创建包GUID容件
是
870
版本在包GUID容件中？
否
840
创建版本容件
是
880
改写？
否
是
将包放入版本容器中
850
从存储的驱动程序包提取元数据并放入元数据存储 304
860
结束

图 8

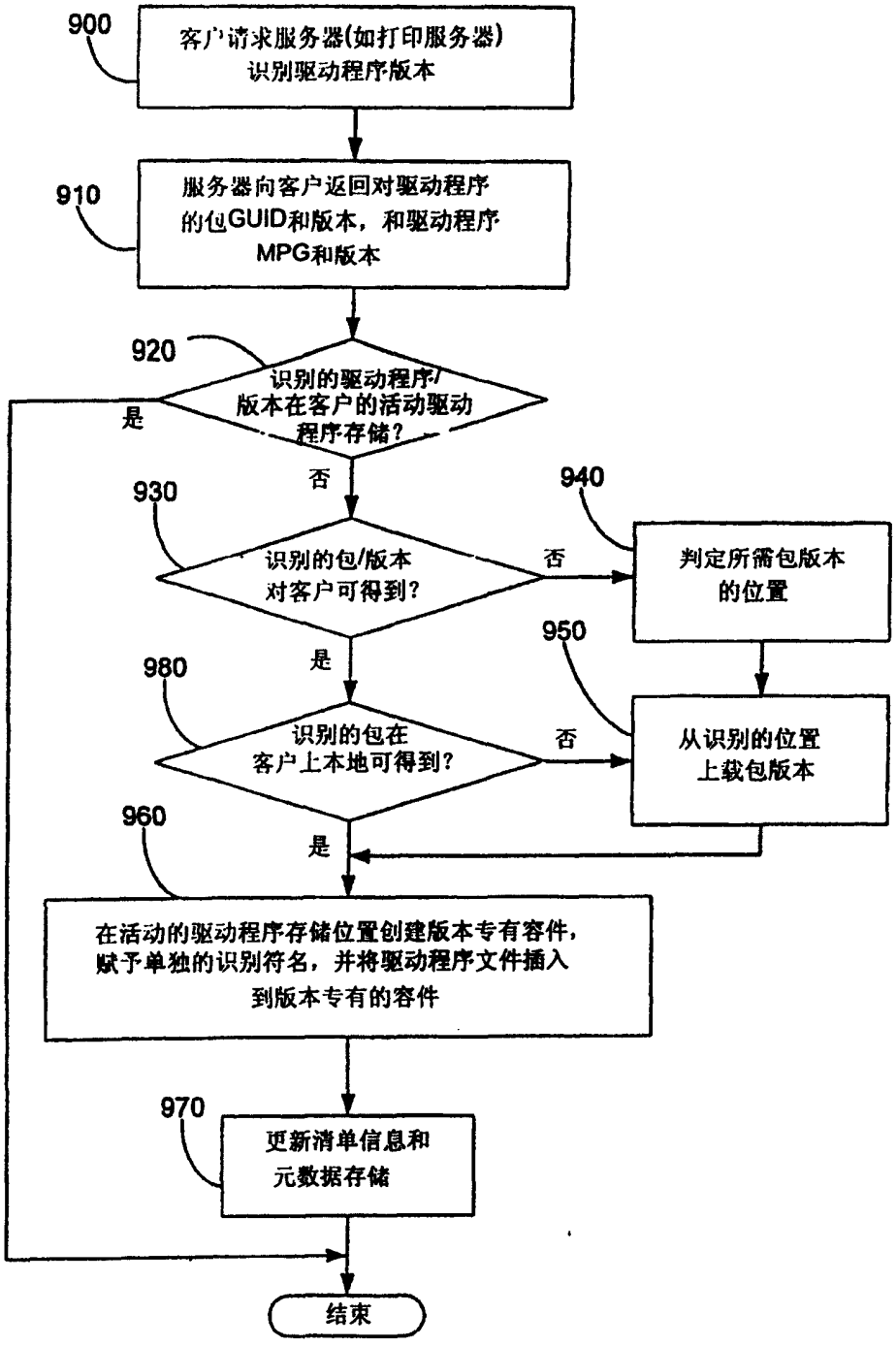

900
客户请求服务器(如打印服务器)
识别驱动程序版本
910
服务器向客户返回对驱动程序
的包GUID和版本，和驱动程序
MPG和版本
920
识别的驱动程序/
版本在客户的活动驱动
程序存储?
是
否
930
识别的包/版本
对客户可得到?
否
940
判定所需包版本
的位置
是
950
980
识别的包在
客户上本地可得到?
否
从识别的位置
上载包版本
960
是
在活动的驱动程序存储位置创建版本专有容件，
赋予单独的识别符名，并将驱动程序文件插入
到版本专有的容件
970
更新清单信息和
元数据存储
结束

图 9

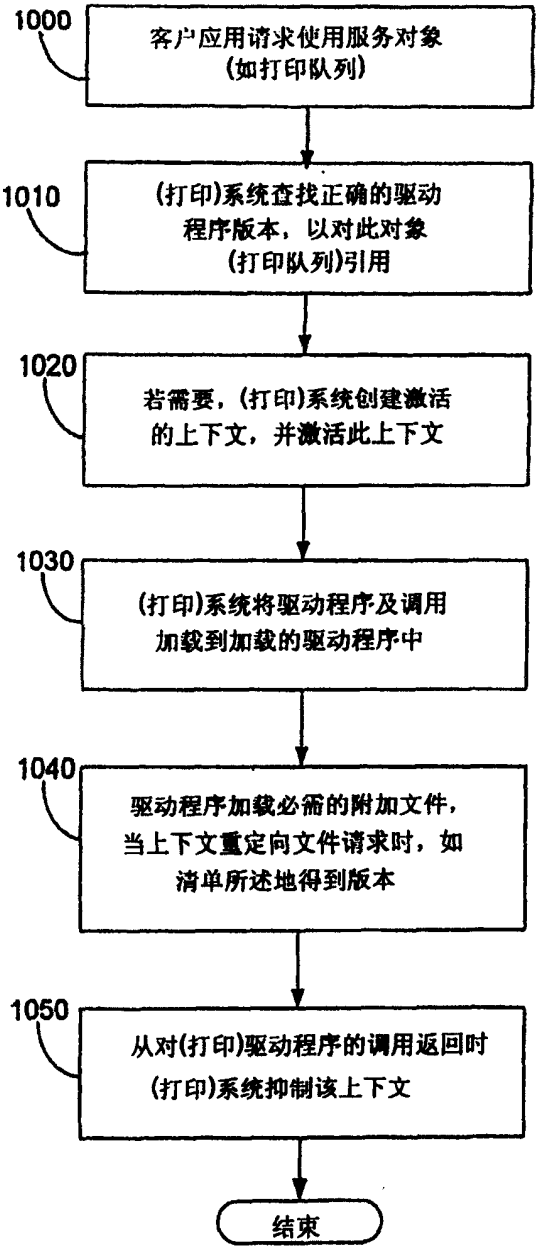

1000
客户应用请求使用服务对象
(如打印队列)
1010
(打印)系统查找正确的驱动
程序版本，以对此对象
(打印队列)引用
1020
若需要，(打印)系统创建激活
的上下文，并激活此上下文
1030
(打印)系统将驱动程序及调用
加载到加载的驱动程序中
1040
驱动程序加载必需的附加文件，
当上下文重定向文件请求时，如
清单所述地得到版本
1050
从对(打印)驱动程序的调用返回时
(打印)系统抑制该上下文
结束

图 10