[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 12/00
H04L 29/00

[12] 发明专利申请公开说明书

[21] 申请号 200510009490.2

[43] 公开日 2005 年 9 月 14 日

[11] 公开号 CN 1668010A

[22] 申请日 2005.2.16
[21] 申请号 200510009490.2
[30] 优先权
[32] 2004. 3.12 [33] US [31] 10/799,440
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 A·H·艾弗布克 D·C·马尔
D·B·德根 D·P·门齐斯
J·R·弗舍尔 M·舍帕德
康成国

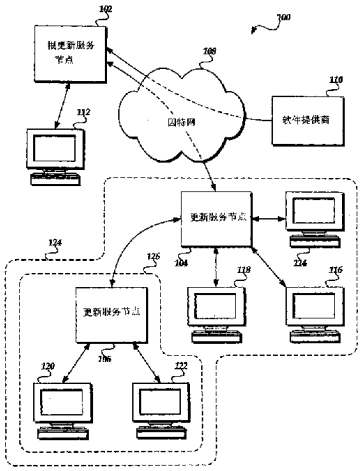
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 张政权

权利要求书 3 页 说明书 18 页 附图 10 页

[54] 发明名称 用来在更新分发系统中分发更新元数据的基于标记模式

[57] 摘要

提出了一种用来把软件更新元数据信息传送到客户计算机和更新服务节点的基于标记结构。 更新元数据文件包括：包括单独地标识软件更新的软件更新标识符的标识符标记；带有关于软件更新的一般属性信息的零个或多个一般属性标记；带有根据语言组织的本地化属性信息的零个或多个本地化属性标记；标识如更新元数据中所述当前软件更新与其它软件更新的依从关系的零个或多个关系标记；带有用来确定软件更新对客户计算机的可应用性的信息的零个或多个应用规则标记；带有关于软件更新的有效载荷文件的信息的零个或多个文件标记；以及带有导向用来安装软件更新的软件处理器的信息的处理器专用数据标记。



ISSN1008-4274

1. 存储组织在一基于标记数据结构中的计算机可读数据的一种计算机可读介质，所述数据结构用来把对应于一软件更新的更新元数据传送到更新服务节点或客户计算机，其特征在于，所述数据结构包括：

一 UpdateIdentity 元素，用来单单独地标识所述软件更新；

零个或多个 Properties 元素，用来存储关于所述软件更新的一般属性；

零个或多个 LocalizedPropertiesCollection 元素，用来存储导向与所述软件更新相关的计算机用户的内容；

零个或多个 Relationships 元素，用来存储所述软件更新与其它软件更新的关系；

零个或多个 ApplicabilityRules 元素，用来存储确定所述软件更新对客户计算机的可应用性的规则；

零个或多个 Files 元素，用来存储描述关于所述软件更新的有效负载信息的信息；

零个或多个 HandlerSpecificData 标记，用来存储安装所述软件更新专用类型的更新处理器的信息。

2. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述基于标记数据结构是 XML 数据结构。

3. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述基于标记数据结构的所述元素，如果出现，按照上述顺序包括在所述更新元数据中。

4. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 UpdateIdentity 元素包括单单独地标识所述软件更新的一单单独标识符子元素、以及标识关联于所述软件更新的修正号的一修正号子元素。

5. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述基于标记数据结构中的所述 Relationships 元素包括标识必须在安装所述软件更新之前安装的第二软件更新的一先决条件子元素。

6. 如权利要求 5 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识多个软件更新的多个先决条件子元素，这些子元素可用布尔运算符连接成一逻辑语句，使得对所述逻辑语句的运算可确定所述软件更新安装在客户计算

机上的适应性。

7. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识必须共延展地（coextensively）安装的多个软件更新的一捆绑子元素。

8. 如权利要求 7 所述的基于标记数据结构，其特征在于，所述捆绑子元素的多个软件更新可用布尔运算符连接成一逻辑语句，使得对所述逻辑语句的运算可确定所述软件更新安装在客户计算机上的适应性。

9. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识由所述软件更新取代的至少一个其它软件更新的一取代子元素。

10. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识在所述软件更新安装之前必须安装的其它软件更新的任意数量先决条件子元素、标识必须共延展地安装的多个软件更新的捆绑子元素、以及标识由所述软件更新取代的至少一个其它软件更新的取代子元素。

11. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来给客户计算机上现有文件打补丁的所述软件更新的有效负载的信息。

12. 如权利要求 1 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来替代客户计算机上现有文件的所述软件更新的有效负载的信息。

13. 如权利要求 12 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来给客户计算机上现有文件打补丁，以及替代客户计算机上现有文件的所述软件更新的有效负载的信息。

14. 存储组织成一基于标记数据结构的计算机可读数据的一种计算机可读介质，所述数据结构用来把对应于一软件更新的更新元数据传送到更新服务节点或客户计算机，其特征在于，所述数据结构包括：

一 UpdateIdentity 元素，用来单单独地标识所述软件更新；

零个或多个 Relationships 元素，用来存储所述软件更新与其它软件更新的关系；

零个或多个 Files 元素，用来存储描述关于所述软件更新的有效负载信息的信息。

15. 如权利要求 14 所述的基于标记数据结构，其特征在于，所述基于标记数据结构是 XML 数据结构。

16. 如权利要求 14 所述的基于标记数据结构，其特征在于，所述 UpdateIdentity 元素包括单单独地标识所述软件更新的一单单独标识符子元素、以及标识关联于所

述软件更新的修正号的一修正号子元素。

17. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识必须在安装所述软件更新之前安装的第二软件更新的一先决条件子元素。

18. 如权利要求 17 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识多个软件更新的多个先决子元素，这些子元素可用布尔运算符连接成一逻辑语句，使得对所述逻辑语句的运算可确定所述软件更新安装在客户计算机上的适应性。

19. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识必须共延展地（coextensively）安装的多个软件更新的一捆绑子元素。

20. 如权利要求 19 所述的基于标记数据结构，其特征在于，所述捆绑子元素的多个软件更新可用布尔运算符连接成一逻辑语句，使得对所述逻辑语句的运算可确定所述软件更新安装在客户计算机上的适应性。

21. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识由所述软件更新取代的至少一个其它软件更新的一取代子元素。

22. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Relationships 元素包括标识在所述软件更新安装之前其它软件更新的任意数量先决子元素、标识必须共延展地安装的多个软件更新的捆绑子元素、以及标识由所述软件更新取代的至少一个其它软件更新的取代子元素。

23. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来给客户计算机上现有文件打补丁的所述软件更新的有效负载的信息。

24. 如权利要求 16 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来替代客户计算机上现有文件的所述软件更新的有效负载的信息。

25. 如权利要求 24 所述的基于标记数据结构，其特征在于，所述 Files 元素包括标识用来给客户计算机上现有文件打补丁，以及替代客户计算机上现有文件的所述软件更新的有效负载的信息。

用来在更新分发系统中分发更新元数据的基于标记模式

技术领域

本发明涉及软件和计算机网络，尤其涉及用来在更新分发系统中分发更新元数据的基于标记模式。

背景技术

几乎所有的可购买软件产品都要经受持续的修正过程，以修理和更新软件的特征。软件产品的每一次修正常常需要添加新文件、用较新修正替换现有文件、删除过时文件、或这些动作的各种各样的组合。替换较旧文件、添加新文件、并删除软件产品的过时文件的这个过程此后将被称为“更新产品”，并且在更新产品中使用的数据集，包括二进制文件、数据文件、更新指令、元数据等等，此后将被简称为“更新”。

一旦软件供应商已为软件产品创建了用以排除问题、提高安全或添加新特征的更新，软件供应商将希望该更新对其客户库广泛可用。常常，诸如当更新涉及更正产品缺陷或解决重大安全问题时，软件供应商将希望该更新能尽快地安装在客户的计算机上。确实，大多数软件供应商有着把软件更新尽快并尽量无困难地分发给其客户的商业动机。

计算机行业已在连接到网络特别是因特网的计算机数量方面经历了爆炸性的增长。由于该爆炸性增长，并由于经与因特网连接可用的通信能力，因特网已变成软件供应商向其客户分发更新的重要且不可或缺的通道。实际上，因特网已变成许多软件供应商向其客户提供软件更新的主要分发通道。软件供应商最感兴趣的常常是在因特网上分发软件更新，因为在因特网上进行电子更新分发降低了他们的总成本，并使客户在一旦软件更新可用时就可获得它们。越来越频繁地，这些软件更新无任何用户干涉就在因特网上进行。

尽管现在通常因特网被用作从软件供应商分发软件更新的渠道，还是有若干问题会频繁产生。两个这种问题包括(1)关于更新分发基础设施/资源的功效，以及(2)软件更新在分发和安装上的管理控制。

关于分发资源的功效，包括因特网的网络仅拥有常被称为带宽的有限量的通

信资源。有限量的通信带宽常导致瓶颈，特别是对于普及软件产品的软件更新，诸如微软公司的 Windows®系列的操作系统和相关产品。即使当在因特网上分布的多个下载点上都有软件更新时也存在这种瓶颈。发生这种瓶颈的一个原因是因特网提供的非结构化访问模型。例如，如果计算机 A 上的第一个用户请求下载最新的软件产品，该下载经过第一个用户的独立服务提供商（ISP）。此外，该请求被视为单个独立的访问，意思是该请求被视为独立于或无关于所有其它网络通信和/或请求。由此，如果恰巧有相同 ISP 的计算机 B 上的第二个用户请求与第一个用户同样的下载，该第二个用户的请求也被视为单个独立的访问。在此例中，由于每个请求都被单独处理，相同的下载要在相同的基础设施上传送两次。显然，如果用户数量有了实质性的增长，有限的通信带宽将变成瓶颈。在颇为常见的此例中，如果下载能高速缓存在一本地区域中，且每个用户请求从本地高速缓存中得到满足，这将会有效得多。

关于分发的控制，许多机构特别是大型机构要控制更新向其计算机的分发是有合理原因的。例如，不幸地某些更新具有或者会引入常称为错误（bug）的缺陷，它们会“破坏”软件产品的特征。这些破坏的特征也许并不重要，但是常常它们能瓦解某企业的关键关键任务（mission-critical）特征。因为企业不能承受失去其关键任务特征，负责的企业在把更新发送到其它计算机上之前，首先在受控环境中评估并测试每个软件更新一段时间。该评估阶段允许机构去证实更新是否将负面地影响关键任务特征。只有在已经令人满意地确定了更新将不会使任何关键任务特征崩溃之后，才允许向机构剩下的计算机分发更新。显然，大多数机构必须对软件更新在其计算机上的安装进行控制。

企业或机构常需要控制软件更新的分发的另一个原因是确保机构中计算机间的连续性。对信息服务部门而言，拥有所有计算机在其上操作的标准化目标平台是非常重要的，不管它是字处理器的平台还是操作系统的平台。如果没有标准，软件和计算机维护会变得不必要的复杂和困难。

本地控制重要的又一原因是为了发账单。在大机构中，单独在计算机上安装软件，或对机构中每台计算机的特定软件产品单独地许可证维持许可证，常常是低效的。相反，单节点许可证允许机构在多台计算机上运行软件产品。因而，机构可能需要报告在节点许可证下运行产品的计算机数量，或需要限制节点许可证下运行产品的计算机数量。所有这些原因常需要对软件更新分发的本地控制。

考虑到以上说明的涉及软件更新分发的各种问题，所需要的是一种可扩展软

件更新分发体系结构，它用来提供对软件更新分发的控制并增加其分发功效。本发明处理在现有技术中发现的这些和其它问题。

发明内容

根据本发明的诸方面，提出了一种用来把更新元数据从更新服务节点传送到子更新服务节点或客户计算机的基于标记的结构。该基于标记的结构包括：包括唯一地标识软件更新的软件更新标识符的标识符标记；带有关于软件更新的一般属性信息的零个或多个一般属性标记；带有根据语言组织的本地化属性信息的零个或多个本地化属性标记；标识如在更新元数据中所述的当前软件更新与其它软件更新的依从关系的零个或多个关系标记；带有用来确定软件更新对客户计算机的可应用性的信息的零个或多个可应用性规则标记；带有涉及软件更新的有效载荷文件的信息的零个或多个文件标记；以及带有涉及用来安装软件更新的软件处理器的信息的处理器专用数据标记。

根据本发明的其它方面，提出了一种用来把更新元数据从更新服务节点传送到子更新服务节点或客计算机的基于标记的结构。该基于标记的结构包括：包括唯一地标识软件更新的软件更新标识符的标识符标记；标识如更新元数据中所述当前软件更新与其它软件更新的依从关系的零个或多个关系标记；和带有涉及软件更新的有效载荷文件的信息的零个或多个文件标记。

附图说明

结合附图参阅以下详细说明，本发明的前述诸方面和许多附带优点将变得更容易领会，也变得更好理解。

图 1 是根据本发明诸方面形成的示例性更新分发系统的示图；

图 2 是示出根据本发明诸方面形成的更新服务节点的示例性逻辑组件的框图；

图 3 是示出根据本发明诸方面形成的更新服务根节点的示例性逻辑组件的框图；

图 4A 和 4B 是示出根据本发明诸方面，在从父更新服务节点向子更新服务节点提供软件更新中，在父更新服务节点和子更新服务节点之间的示例性交换的框图；

图 5 是示出示例性例程的流程图，该例程在子更新服务节点上执行，以周期性地从父更新服务节点获取更新；

图 6 是示出示例性子例程的流程图，该子例程适合于在图 5 的示例性例程中使用，来从父更新服务节点获取更新目录；

图 7 是示出示例性子例程的流程图，该子例程适合于在图 5 的示例性例程中使用，用来从父更新服务节点获取软件更新；

图 8 是示出示例性例程的流程图，该例程用来处理来自子更新服务节点的更新请求；以及

图9是示出确定更新元数据文件内容的部分示例性基于XML更新元数据模式的框图。

具体实施方式

根据本发明的诸方面，提出了以分层方式组织的用来分发软件更新的更新分发系统。图 1 是根据本发明诸方面形成的示例性更新分发系统 100 的示图。根据本发明，在诸如所示更新分发系统 100 的更新分发系统“顶部”，是更新服务根节点 102。诸如软件供应商 110 的软件供应商，通过向更新服务根节点 102 提交更新，经更新分发系统 100 分发其软件更新。根据本发明的诸方面，诸如软件供应商 110 的软件供应商可经诸如因特网 108 的网络向更新服务根节点 102 提交软件更新。

诸如示例性更新分发系统 100 的分层更新分发系统，除更新服务根节点 102 外可能将包括至少一个其它更新服务节点。如图 1 所示，示例性更新分发系统 100 包括更新服务根节点 102 和两个其它更新服务节点：更新服务节点 104 和更新服务节点 106。根据本发明，每个分层更新分发系统都在更新服务根节点 102 下以树状结构形式组织。换言之，更新分发系统中的每个更新服务节点有零个或多个子更新服务节点。因而，尽管示例性更新分发系统 100 示出每个父更新服务节点，即更新服务根节点 102 和更新服务节点 104，都仅有一个子节点，这仅仅是为了说明而不应被解释为限制本发明。此外，除了更新服务根节点 102 以外，更新分发系统中的每个更新服务节点都有一个父更新服务节点。因而，如图 1 所示，更新服务节点 104 是更新服务根节点 102 的子节点，而更新服务节点 106 是更新服务节点 104 的子节点。由此可见，每个更新服务节点，除了更新服务根节点 102，都有父更新服务节点和子更新服务节点。

如在示例性更新分发系统 100 中所示，更新服务根节点 102 经因特网 108 与更新服务节点 104 通信。然而可以理解，这仅是说明性的而不应被解释为限制本发明。更新分发系统中的每个更新服务节点仅需要能够经某些通信网络与其父节点和

/或子节点通信。因而，当更新服务节点 104 经因特网 108 与其父节点-更新服务根节点通信时，它也可经局域网 124 可选地与其子更新服务节点，诸如更新服务节点 106 通信。

还如图 1 所示，更新服务节点 106 驻留于局域网 124 的子网络 136 中。作为示例，局域网 124 可对应于机构的一般公司网络，且更新服务节点 104 代表了通过与其父更新服务节点 102 的连接到更新分布系统 100 的公司链接。此外，子网络 126 可对应于公司网络中的可标识计算机组，诸如测试/评估组、远程办公室、或关键任务组。如下面将要更详细描述的，根据本发明诸方面，更新服务节点 104 的管理员能够控制向更新服务节点 106，最终向客户计算机分发更新。

可以理解，包括更新服务根节点 102、更新服务节点 104 和 106 的每个更新服务节点被配置为向子更新服务节点以及客户计算机分发软件更新。如图 1 所示，示例性更新分发系统 100 包括客户计算机 112-122。包括更新服务根节点 102 的每个更新服务节点根据本地配置信息向子更新服务节点和客户计算机分发更新。根据一实施例，管理员确定各组，并使更新分发规则与那些组相关联。每个更新服务节点具有至少一个分发组。

作为要说明更新分发系统如何操作的示例，假设局域网 124 对应于机构的公司网络。根据本方面的一实施例，更新服务节点 104 的管理员可为公司网络 124 确定多个分发组，包括对应于包括更新服务节点 106、客户计算机 120 和 122 的评估组，用来评估更新对一般公司网络以及包括更新服务节点 104、客户计算机 114-118 的一般公司组的适应性。

对于评估组，管理员把更新服务节点 106 作为其成员，并使规则与该组相关联，从而当更新可用时就能立即分发给评估组的成员。或者，对于一般公司组，管理员添加客户计算机 114-118 并关联规则，使得如果由管理员特别授权只会把更新分发给一般公司组。还假设子更新服务节点 106 的管理员创建包括评估子网络 126 中客户计算机 120 和 122 的缺省组，可立即向该组分发任何新的软件更新。

继续上例，软件供应商 110 向更新服务根节点 102 提交软件更新。根据在更新服务根节点 102 上建立的规则，最终将向公司更新服务节点 104 分发更新。在接收更新之后，根据由管理员建立的规则，公司更新服务节点 104 向评估组成员（仅定义为子更新服务节点 106）分发更新，但在特别授权向该组分发更新之前扣留来自一般公司组的更新。

继续上例，接收更新之后，评估更新服务节点 106 对每个定义组处理更新。

在此例中，评估更新服务节点 106 仅有一个组。然而如前所述，在真正的实现中，可能会有多个确定的组，每个组都有唯一的相关联分发规则集。对于该例，评估更新服务节点 106 立即提供可用于向客户计算机 120 和 122 分发的更新。客户计算机 120 和 122 现在可得到更新，并可开始评估阶段/过程。

仍继续上例，当公司更新服务节点 104 的管理员对更新适于在整个公司网络 124 上分发足够满意时，管理员明确地授权更新向一般公司组的成员分发。公司更新服务节点 104 相应地提供可用于客户计算机 114-118 的更新。应当理解，也可将评估更新服务节点 106 包括在一般公司组中。然而，因为已经更新了评估更新服务节点 106，不需要其它更新相关动作来把更新分发到评估子网络 126。

由上例可见，本发明在本地分发和下载效率方面提供了重要优点。除了本地分发控制的上述方面，还实现了通信带宽的显著节约。例如，尽管图 1 所示的示例性公司网络 124 包括了五台客户计算机，却只进行了一次将软件供应商的更新从更新服务根节点 102 下载到公司更新服务节点 104。明显地，当由更新服务节点服务的客户计算机数量增加时，在父更新服务节点和更新服务客户机节点之间的通信带宽的使用保持常量，因此极大地减少了要使用的通信带宽。另外，更新分发系统是可扩展并可升级的。更新分发系统可用至少两种方法扩展：可向父更新服务节点添加任意数量的子更新服务节点，且子更新服务节点也可以是父更新服务节点。因此可裁剪更新方法系统的每个子树来满足个别需求。

图 2 是示出根据本发明诸方面形成的更新服务节点 200，诸如公司更新服务节点 104（图 1）或评估更新服务节点 106（图 1），的示例性逻辑组件的框图。如图 2 所示，更新服务节点 200 包括更新网络服务 202、客户机更新模块 204、子更新模块 206 和报告模块 208。示例性更新服务节点 200 还包括认证/授权模块 210、管理应用编程接口(API) 212、更新内容存储 214、管理用户接口 218、以及更新信息存储 216。

更新网络服务 202 提供网络服务的通用集，通过它客户计算机、子更新服务节点和父更新服务节点可与更新服务节点通信。例如，参照图 1，为了使子/评估更新服务节点 106 从父/公司更新服务节点 104 获取软件更新，客户机可通过父节点的更新网络服务 202 通信。类似地，当诸如更新服务根节点 102 的父更新服务节点具有包括更新的信息要传送到其子更新服务节点 104 时，父更新服务节点通过子节点的更新网络服务 202 通信。

关于存储在更新服务节点上的更新和更新信息，客户机更新模块 204 处理客

户计算机和更新服务节点200之间的通信。更新相关通信包括，但不限于，响应于客户机请求分发更新，并向客户计算机提供可用软件产品及相关联更新的列表。客户机更新模块 204 还负责确定是否授权客户计算机根据相关联分发规则获取特定更新，并用客户计算机被授权访问的更新相关信息对客户计算机作出响应。

子更新模块 206 处理父更新服务节点与其子更新服务节点之间的更新相关通信。更新相关通信包括，但不限于，标识子更新服务节点可用的软件产品及相关联更新的列表，以及响应来自子更新服务节点的更新请求。下游更新模块206负责确定是否授权子更新服务节点根据相关联分发规则获取特定更新，并用子更新服务节点被授权访问的更新相关信息对子更新服务节点作出响应。

报告模块 208 产生更新相关报告，诸如哪个组已收到或未收到特定更新、哪个客户计算机已下载/安装或未下载/安装更新、什么更新在更新服务节点上可用等等。这些报告可诸如由管理员内部使用，并也可通过父节点的更新服务接口 202 提交给父更新服务节点。如上所述，对公司而言诸如为了发账单或为了维护而确定哪台客户计算机安装有特定更新常常是重要的。由报告模块 208 产生的信息/报告可以是这些报告的基础。

认证/授权模块210负责认证，即确定特定客户计算机或子更新服务节点的身份，并确定客户计算机或子更新服务节点是否被授权访问更新服务节点200上的可得到的更新。对于经认证并被授权访问更新服务节点上更新的那些客户计算机和子更新服务节点，认证/授权模块 210 发出必须结合获取更新使用的授权令牌（authorization token）。授权令牌的签发和使用将参照图 4A 和 4B 在后面进行更详细的描述。

管理 API 212 代表的的应用程序接口，通过它进行更新服务节点 200 的控制，并最终存储并分发更新。当更新网络服务 202 接收来自客户计算机和子更新服务节点的各种各样更新相关请求时，这些请求最终被分解成对管理 API 212 的直接调用，或经客户机更新模块 204 和子更新模块 206 的间接调用。结合管理用户接口218或安装在更新服务节点200上被适当配置为使用管理API 212的某些其它程序，管理员最终控制对该更新服务节点、以及任何子更新服务节点和客户计算机的更新过程的所有方面。

通过管理用户接口 218，管理员可经管理 API 212 配置并维护更新服务节点200。因而，通过管理用户接口 218，管理员创建、修改、并删除组以及每个组的相关联规则。此外，使用管理用户接口218，管理员建立客户计算机或子更新服务

节点所属的组。通过管理用户接口 218，管理员还可明确授权更新向客户计算机或子更新服务节点的分发、配置更新服务节点 200 周期性地向其父更新服务节点查询新更新、配置报告参数并查看内部报告等等。如上所述，尽管管理用户接口 218 允许管理员对更新服务节点 200 诸方面进行控制，可替代管理用户接口 218 使用驻留于更新服务节点 200 上的适当地用管理 API 212 来操作的另一应用。

如上所述，根据本发明的一实施例，更新服务节点 200 包括更新内容存储 214 和更新信息存储 216。更新内容存储 214 存储代表软件更新的真正文件，诸如二进制文件和补丁文件。相反，更新信息存储 216 存储对应于更新服务节点 200 上可得到的更新的信息和元数据，包括存储在更新内容存储 214 上的更新文件。根据一实施例，更新内容存储 214 和更新信息存储 216 都是关系数据库。尽管示例性更新服务节点 200 被示为有两个数据存储，不应如此限制本发明。在一另外实施例中，可把更新内容存储 214 和更新信息存储 216 组合成一个信息存储。

根据本发明的诸方面，即使未在更新内容存储 214 中具体地存储更新，软件更新可表示成对客户计算机和子更新服务节点在更新服务节点 200 上“可用”。尤其，可在更新服务节点上存储指向父更新服务节点或其它地方上更新文件的链接，而不是立即下载和存储在更新服务节点 200 上的真正更新文件。因而，如果客户计算机请求更新或子更新服务节点请求真正的更新，就从父更新服务节点取下更新并将其存储在更新内容存储 214 中，准备将其向客户计算机或子更新服务节点传送。本领域技术人员将理解，该类更新访问被称为及时下载（just-in-time downloading）。用此方法，在真正请求“可用”更新之前不需要在各种网络信道上分发更新。根据本发明的诸方面，更新服务节点 200 的管理员可有选择地确定，是否要以及时方式获取软件更新。

尽管图 2 的以上描述说明了示例性更新服务模块 200 的各种组件，可以理解也可存在更新服务模块的其它组件。此外，应当理解上述组件是逻辑组件而不必要是真实组件。在一真实实现中，根据实现确定可把以上标识的组件组合在一起和/或与其它组件组合。另外，可以理解尽管更新服务节点 200 可视为网络上的服务器计算机，在实际实现中，可在任意多类计算装置上实现更新服务节点。例如，可在单台独立的计算机系统或者包括多个计算装置的分布式计算系统上实现和/或安装每个更新服务节点 200。

图 3 是示出根据本发明诸方面形成的更新服务根节点 300，诸如图 1 所示更新服务根节点 102 的示例性逻辑组件的框图。类似于更新服务节点 200（图 2）的逻

辑组件，更新服务根节点 300 包括更新网络服务 202、子更新模块 206 和认证/授权模块 210。另外，示例性更新服务根节点 300 还包括管理 API 212、更新内容存储 214、以及更新信息存储 216。可选地，更新服务根节点 300 还可包括客户机更新模块 204、报告模块 208、以及管理用户接口 218。

客户机更新模块 204 是更新服务根节点 300 的可选组件，取决于该更新服务根节点是否直接向客户计算机提供软件更新。例如，参照图 1，更新服务根节点 102 应包括可选的客户机更新模块 204 作为直接服务客户计算机 112 的更新服务根节点。然而，如果更新服务根节点 300 不直接服务客户计算机，则可省略客户机更新模块 204。

因为更新服务根节点 300 没有要向其提供更新报告的父更新服务节点，报告模块 208 对更新服务根节点 300 是可选的。然而，在更新服务根节点管理员想要更新报告时，可选地包括报告模块 208。

除了包括更新服务节点 200（图 2）所包括的逻辑组件外，更新服务根节点 300 还包括软件供应商接口 302。软件供应商接口 302 提供通信接口，通过这个接口软件供应商 110（图 1）直接向更新服务根节点 300 提交软件更新，并间接向示例性更新分发系统 100 提交软件更新。

类似于图 2 的更新服务节点 200，图 3 的以上描述示出了示例性更新服务根模块 300 的各种组件。然而，应当理解也存在更新服务根模块 300 的其它组件。此外，应当理解上述组件是逻辑组件而不必是真实组件。在一真实实现中，根据实现确定可把以上标识的组件组合在一起和/或与其它组件组合。另外，可以理解尽管更新服务节点 300 可视为网络上的服务器计算机，在真实实现中可在任意多台计算装置上实现更新服务节点。例如，可在单台独立的计算机系统或者包括多个计算装置的分布式计算系统上实现和/或安装更新服务节点 300。

为了更好地理解如何穿过更新分发系统 100 从更新服务根节点分发更新，示出了在父更新服务节点和子更新服务节点之间示例性交换是恰当的。图 4 是示出根据本发明诸方面，在从父更新服务节点 402 向子更新服务节点 404 传播软件更新中，父更新服务节点和子更新服务节点之间的示例性交换 400 的框图。由此可见，示图 400 被分成两半，左边部分对应父更新服务节点 402 的动作和事件，而右边部分对应子更新服务节点 404 的动作和事件。

为了参照图 4 进行讨论，还应当理解父更新服务节点 402 可以是，也可以不是更新分发系统 100 中的更新服务根节点。另外，为了这个讨论，假设已由管理员

对父更新服务节点 402 进行了配置，使得除非由管理员明确授权，子更新服务节点 404 不能接收软件更新。

如示例性交换 400 所示，从事件 406 开始，父更新服务节点 402 直接地（如果父更新服务节点是更新服务根节点 102）或间接地经更新分发系统 100 从软件供应商 110 处接收软件更新。在父更新服务节点 402 从软件供应商 110 处接收软件更新之后的某个时间，子更新服务节点 404 开始从父更新服务节点获取软件更新的过程。

根据一实施例，可配置子更新服务节点 404 周期性地自动获取从父更新服务节点 402 可得到的软件更新。更特别地，通过管理用户接口 218，管理员能可选地配置子更新服务节点 404，周期性地自动获取在父更新服务节点 402 可得到的最新软件更新。作为一个示例，管理员可配置子更新服务节点 404 按天或按小时地自动获取来自父更新服务节点 402 的最新软件更新，并指定自动更新过程要开始的日期时间。也可利用其它周期性的调度和标准。类似地，管理员可经管理用户接口手动启动更新过程。

为了开始更新过程，在事件 408 子更新服务节点 404 向父更新服务节点 402 认证并授权它自己。向父更新服务节点 402 的认证和授权提供对软件更新分发的控制元素，限制了向经授权更新服务节点的更新分发。认证和授权技术是本领域技术人员众所周知的，可采用任意数量的这种技术向父更新服务节点 402 认证并授权子更新服务节点 404。本发明不限于任意一种技术。

在恰当地向父更新服务节点 402 认证并授权之后，在 410 框父更新服务节点 402 向子更新服务节点 404 返回授权令牌。根据一实施例，授权令牌是时间敏感令牌，它向父更新服务节点提供子更新服务节点 404 的授权以在有限时间内进一步进行更新动作。因而如果子更新服务节点 404 未向父更新服务节点 402 认证并授权，则无授权令牌返回，且子更新服务节点不能执行除认证和授权外的任何其它更新相关动作。类似地，当更新令牌过期后，子更新服务节点 404 不能向父更新服务节点 402 执行除认证和授权外的任何其它更新相关动作。

在接收授权令牌后，在事件 412 子更新服务节点 404 向父更新服务节点提交对产品更新目录以及授权令牌的请求。产品更新目录代表父更新服务节点 402 为之分发软件更新的软件产品内容的列表或表格。

根据本发明的诸方面，不要求子更新服务节点 404 传播父更新服务节点 402 提供的所有软件更新。例如，参照图 1 的示例性更新分发系统，公司更新服务节点

104 可具有对更新服务根节点 102 所提供软件产品一部分的节点许可证。相应地，对于公司更新服务节点104而言也不必获取更新服务根节点102所提供的所有软件产品，因为大多数软件是从来不会使用的。因而，更新服务节点的管理员可有选择地建立将在更新服务节点上提供的那些软件产品更新。

根据本发明的一方面，无论子更新服务节点 404 是否被配置为分发每种产品的更新，从父更新服务节点 402 获取的更新产品目录标识了所有可提供的更新的软件产品。然而，根据本发明的一另选方面，从父更新服务节点 402 获取的更新产品目录仅标识请求子更新服务节点被配置成拟分发的更新的那些软件产品。例如，可根据子更新服务节点 404 所属的组或多个组，来确定哪些软件产品在产品更新目录中列出的限制。

在事件 414，父更新服务节点 402 向子更新服务节点 404 返回产品更新目录。在事件 416，子更新服务节点 404 从产品更新目录中选择提供当前所需最新更新的那些产品。应当注意，尽管产品更新目录仅列出子更新服务节点 404 分发的那些软件产品，可配置子更新服务节点在不同时间或按照不同周期性时间表获取不同软件产品的更新。

在事件 418，子更新服务节点 404 提交更新同步请求以及授权令牌，它标识了提供子更新服务节点目前搜索的更新的选定产品。在同步请求中包括的是标识在子更新服务节点 404 上产品所提供最新更新的信息。标识产品最新更新的信息此后被称为“更新锚点”（update anchor）。通常每个软件产品的更新锚点都存储在更新信息存储 216 中（图 2）。在一实施例中，更新锚点包括修正号以及关联于该修正号的日期。

响应于更新同步请求，在事件 420，父更新服务节点 402 确定哪些更新（如果有的话）对子更新服务节点 402 可得到。如上所述，该确定基于与特定软件更新相关联的特定规则、子更新服务节点 404 是其成员的组或多个组、以及更新锚点。对于此例，如前所述，未对子更新服务节点 404 明确授权先前接收的软件更新。由此，在事件 406 接收的软件更新未被确定为对子更新服务节点 404“可用”。因而，在事件 422 向子更新服务节点 404 返回一个没有标识在事件 406 接收的软件更新的更新列表。根据本发明的诸方面，更新列表根据同步请求标识了所有在父更新服务节点 402 上“可用”的更新。在一实施例中，更新列表由关联于一个更新的单单独标识符标识每个“可用”的更新信息。

在事件 424，由于更新列表为空，即在父更新服务节点 402 上无更新当前“可

用”，子更新服务节点 404 的更新过程延迟或休眠预定时间量。根据当前示例，在该延迟期间，在事件 426，父更新服务节点 402 的管理员授权在事件 406 接收的软件更新 404 分发到子更新服务节点。

在事件 428（图 4B），子更新服务节点 404 通过向父更新服务节点 402 认证及授权自己再次开始自动更新过程。在事件 430，作为响应父更新服务节点 402 向子更新服务节点 404 返回一授权令牌。

在事件 432，子更新服务节点 404 向父更新服务节点 402 提交一请求及授权令牌，要求产品更新目录。在事件 434，父更新服务节点 402 向子更新服务节点 404 返回产品更新目录。在事件 436，子更新服务节点 404 对更新目录选择希望更新的产品。在事件 438，子更新服务节点 404 提交用授权令牌所标识的那些选中产品的更新同步请求。

由于子更新服务节点 404 已授权获取先前在事件 406 接收的软件更新，在事件 404 父更新服务节点 402 确定，对该子更新服务节点该软件更新是“可用”的，并在更新列表中包括对应的更新信息。随后在事件 442，父更新服务节点 402 向子更新服务节点 404 返回现标识在事件 406 中接收的软件更新的更新列表。

有了标识父更新服务节点 402 上“可用”更新的更新列表，子更新服务节点 404 现在有了获取软件更新所必需的信息。根据本发明的一实施例，子更新服务节点 404 从父更新服务节点 402 分两部分获取软件更新：获取更新元数据，并获取更新内容或文件（此后称为更新有效载荷）。如在后面将要更详细描述的，更新元数据描述了软件更新的有关方面，包括但不限于：唯一标识更新的更新标识符、关联于软件更新的修正号信息、是否应当考虑软件更新有优先级、语言专用信息、与其它软件更新的关系、用来下载的更新有效负载的位置、安装处理器例程等等。

把整个软件更新分两部分下载，即更新元数据和更新有效负载，常常是有益的部分原因是，更新有效负载常常比更新元数据大得多，而且即使需要更新有效负载（即需要用来在客户计算机上安装。）也并非总是立即需要的。因而，根据本发明的一实施例，更新有效负载与更新元数据分开下载，且仅在需要时下载。本领域技术人员将认识到这种下载技术是惰性下载（lazy downloading），或者是及时下载。根据本发明的诸方面，管理员可配置更新服务节点以及时方式获取更新有效负载，或在获取更新元数据之后立即下载。此外，在一另外实施例中，可联合下载更新元数据和更新有效负载。

如图 4B 所示，在事件 444，更新在更新列表中被标识后，子更新服务节点 404

根据其在更新列表中的单单独标识符请求“可用”软件更新的更新元数据。象与父更新服务节点 402 的大多数其它通信交换一样，更新请求与授权令牌一起提交。应当注意，尽管在示例中所有更新元数据在一次访问中下载，根据本发明的另外方面（未示出），更新元数据可分多次访问下载。例如，在第一次访问中，首先下载确定软件更新是否可应用和/或是需要的所必需的更新元数据元素，诸如可应用度规则和对其它软件更新的依从性。然后，在确定更新是可应用和/或所需的之后，可获取剩下的更新元数据。作为响应，在事件 446 父更新服务节点 402 向子更新服务节点 404 返回软件更新的更新元数据，而子节点把更新元数据存储到更新信息存储 216 中。

可选地，在事件 418，子更新服务节点 404 提交从父更新服务节点 402 下载更新有效负载的请求。作为响应，在事件 450 父更新服务节点 402 向子更新服务节点 404 返回更新有效负载，而子节点把它存储在更新内容存储 214 中。在另外的实施例中，子更新服务节点 404 从可能不是父更新服务节点 402，而是在更新元数据中指定的存储位置上下载更新有效负载。

因为更新动作已在子更新服务节点 404 上发生，在事件 452，子更新服务节点产生勾划出最近发生更新动作的更新报告，并将其提交给父更新服务节点 402。然后，子更新服务节点 404 再次延迟，直到安排的更新过程的下一次运行（未示出）。

本领域技术人员将理解，上述事件是为了进行说明，且反映了事件和环境的一个特定示例性集。明显地，根据特定细节和环境也会发生导致上述事件产生某些变化的其它事件。另外，应当理解尽管子更新服务节点 404 从父更新服务节点 402 获取最新的“可用”软件更新，子更新服务节点可同时处理来自其子更新服务节点的多个更新请求。因此，应当视事件的以上顺序是说明性的，而非对本发明的限制。

图 5 是示出示例性例程 500 的流程图，该例程在子更新服务节点，诸如图 1 的公司服务节点 104 上执行，用来周期性地从父更新服务节点获取更新。从 502 方框开始，子更新服务节点从父更新服务节点获取“可用”更新的经同步处理更新列表。从父更新服务节点获取“可用”更新的经同步处理更新列表，将参照图 6 如下描述。

图 6 是示出示例性子例程 600 的流程图，该子例程适于在图 5 的示例性例程 500 中使用，用来从父更新服务节点获取“可用”更新的经同步处理更新列表。从方框 602 开始，如前参照图 4A 和 4B 所述，子更新服务节点向父更新服务节点认证并授权它自己，并响应于适当的认证和授权接收授权令牌。在方框 604，结合授

权令牌，子更新服务节点建立了与父更新服务节点的通信参数。建立通信参数使得父更新服务节点和子节点能适当地建立父子节点都能理解的共同基础。通信参数包括但不限于：通信更新协议或版本；产品分组；等等。

建立与父更新服务节点的通信参数之后，在方框 606，子更新服务节点获取描述父更新服务节点用来提供/分发更新的软件产品的产品更新目录。在方框 608，子更新服务节点选择从中当前搜索更新的那些软件产品更新。在方框 610，子更新服务节点向父更新服务节点提交更新同步请求，包括授权令牌和标识已在子更新服务节点上的当前修正和更新的与选中软件产品相关联的“锚点”。

响应于更新同步请求，在方框 612，子更新服务节点从父更新服务节点获取更新列表，该列表根据在父更新服务节点上“可用”的软件更新以及当前存储在子更新服务节点上的软件更新进行同步操作。如上所述，更新列表通过单单独的标识符标识对子更新服务节点“可用”的父更新服务节点上的那些软件更新。此后，示例性子例程 600 结束。

再次参照图 5，从在父更新服务节点获取了经同步操作的更新列表之后，在判定框 504，可确定当前是否有任何软件更新“可用”于从父更新服务节点下载。该确定根据在经同步操作的更新列表中是否列有任何更新标识符。如果当前没有“可用”的软件更新可下载，示例性例程 500 进行到延迟框 510，在那里示例性例程延迟/休眠直到下一更新阶段。可选地，如果有从父更新服务节点下载的更新“可用”，在框 506，子更新服务节点从父更新服务节点获取更新。从父更新服务节点获取“可用”更新参照图 7 如下描述。

图 7 是示出示例性子例程 700 图的流程图，该子例程适于在图 5 的示例性例程 500 中使用，用来从父更新服务节点获取“可用”软件更新。从框 702 开始，选择更新列表中的第一个更新标识符。在框 704，子更新服务节点从父更新服务节点获取对应于选定更新标识符的更新元数据，并将其存储在更新信息存储 216 中。

根据一实施例，在框 706，子更新服务节点从父更新服务节点获取对应于选定更新标识符的更新有效负载，并将其存储在更新信息存储 212 中。可选地，不需要立即把更新内容下载到子更新服务节点。如前所述，可有选择地配置子更新服务节点来以及时方式从父更新服务节点下载更新。根据如图 7 所示的该可选方法，示例性子例程 700 可选地从方框 704 进行到判定框 708。

在判定框 708，在获得选定的更新标识符的更新元数据以及可选的更新有效负载之后，可确定在更新列表中是否有任何另外的更新标识符。如果有另外的更新标

识符，在方框 710，选择更新列表中下一个更新标识符。例程 700 继续直到在判定框 708，确定更新列表中不再有更新标识符，于是示例性子例程 700 结束。

再参看图 5，在从父更新服务节点获得“可用”更新之后，在方框 508，子更新服务节点向父更新服务节点报告更新动作。然后，在延迟框 510，示例性例程 500 延迟/休眠预定时间量直到下一更新阶段，并随后进行到方框 502 重复上述更新过程。

如图 5 所示，在判定框 504，即使当父更新服务节点上没有更新“可用”，能可选地配置子更新服务节点向父更新服务节点报告其更新动作。根据该另选配置，当没有更新可用时，示例性例程 500 可进行到方框 508 以报告更新动作。

图 8 是在父更新服务节点上实现的示例性例程 800 的流程图，该例程用来响应于子更新服务节点的更新同步请求产生标识“可用”更新的经同步更新列表。从方框 802 开始，父更新服务节点从子更新服务节点接收对标识“可用”更新的更新列表的更新同步请求。在方框 804，选择在更新同步请求中标识的第一个软件产品。

在判定框 806，确定对被标识软件产品是否有任何可用更新。该确定根据存储在更新信息存储 216 中软件产品的元数据、由子更新服务节点提供的更新锚点、以及关联于子更新服务节点所属组的分发规则而作出。根据该确定，如果有更新“可用”，在方框 808，将关联于“可用”更新的单单独更新标识符写入更新列表。在将 “可用”更新的单独更新标识符写入更新列表之后，在判定框 810，确定是否还有任何在更新同步请求中标识的另外的软件产品。如果在更新同步请求中存在另外的软件产品，在方框 814，父更新服务节点选择在更新同步请求中标识的下个软件产品，并返回到判定框 806，以确定是否有选定软件产品的“可用”更新。另外，如果没有在更新同步请求中标识的另外的软件产品，在方框 814，向子更新服务节点返回更新列表。然后，示例性子例程 800 结束。

如前所述，更新元数据描述了软件更新的相关方面。实际上，许多这些方面对软件产品而言都象更新有效负载一样重要。例如，如果第一软件更新的更新元数据表示必须安装先前的软件更新作为先决条件，而且根据管理员判定目前并未安装先前的软件更新，那么下载第一软件更新的更新有效负载就没有价值了。因此，通过不下载第一软件更新的更新有效负载，可节约大量的通信带宽。当然，本领域技术人员将理解这仅仅是更新元数据如何提供涉及软件更新的重要和相关信息的一个示例。

尽管更新元数据可用任何格式的文件进行传送，根据本发明的诸方面，对应

于软件更新的更新元数据被描述并包含在基于标记文件中，诸如可扩展标记语言（XML）文件。可使用任何适当的基于标记文件格式，这些格式允许描述和包含数据的可定制的标记。用来描述和包含信息的诸如基于 XML 或超文本标记语言（HTML）文件的基于标记文件是本领域众所周知的。如本领域所知，常常可根据称为模式的定义文件来确定基于标记 XML 文件的内容。更新元数据可由子更新服务节点和客户计算机两者使用。

在本发明的一真实实施例中，软件更新的更新元数据被包含在符合更新元数据模式的基于 XML 文件中。图 9 是示出定义更新元数据文件内容的部分示例性基于 XML 更新元数据模式 900 的框图。特别地，示例性更新元数据模式 900 将更新（Update）902 定义为包括以下元素： UpdateIdentity 元素 904； Properties（属性）元素 906；LocalizedPropertiesCollection 元素 908；Relationships（关系）元素 910；ApplicabilityRules 元素 912；Files（文件）元素 914；以及 HandlerSpecificData 元素 916。关于元素 906-916，适当的更新元数据文件可包括这些元素的零个或多个，如通过标记限定词“minOccurs="0"” 918。相反，更新元数据文件必须包括一个 UpdateIdentity 元素 904。根据所示的元数据模式 900，每个更新元数据文件中的更新元素如果出现，必须按照上述顺序显示。

如前所示，每个软件更新都分配有单独的标识符。根据本发明的诸方面，UpdateIdentity 元素 904 包括该单独标识符，以及标识该软件更新的其它信息，诸如也关联于软件更新的修正号。

Properties 元素 906 提供涉及软件更新的信息，包括但不限于：软件更新本地化的语言，即英语、西班牙语、法语等等；提议更新在安装期间可能对计算系统的影响，例如最低影响度、高影响度，以及可能或将需要重新启动计算系统，等等的信息；由软件更新更新的软件类型，诸如系统驱动软件或软件应用；相应于软件更新对于软件供应商的重要性的重要性分级；来自软件供应商的相关安全公告；以及用来标识软件更新的更新处理器的更新处理器标识。由以上所列的各种类型属性可见，在更新元数据文件中可有零个或多个 Properties 元素 906。

LocalizedPropertiesCollection 元素 908 提供涉及软件更新的语言专用信息，诸如但不限于：更新标题；旨在向计算机用户显示的软件更新描述；来自软件供应商的发行注解；终端用户许可证协定及相关信息；用来卸载软件更新的用户指南；等等。根据本发明的诸方面，根据语言组合本地化属性。例如，尽管 LocalizedProperties Collection 元素 908 可包括多条信息，以上信息的英语版本可组合在一起，而西班

牙语版本也可类似地组合。与以上 Properties 元素 906 一样，在更新元数据文件中可有零个或多个 LocalizedPropertiesCollection 元素 908。

Relationships 元素 910 提供关于本软件更新与其它软件更新之间关系的信息。这些关系的示例包括，但不限于，先决条件软件更新、取代软件更新、以及捆绑软件更新。先决条件软件更新关系表明，在安装本软件更新之前，必须在客户计算机系统上安装另一由单独更新标识符标识的软件更新。多个先决条件关系可由诸如逻辑运算符 AND 和 OR 的布尔运算符连接成逻辑语句，使得对逻辑语句的运算可确定软件更新在客户计算机上安装的适应性。

捆绑软件更新标识要一起安装的多个软件更新。作为一个示例，捆绑应用可表示本软件更新与其它软件更新之间的相互依从性，从而如果在计算机系统上安装了其中任何软件更新，则必须安装捆绑中由单独更新标识符标识的所有更新。与以上的先决条件软件更新一样，捆绑软件更新的元素可由布尔运算符连接以形成逻辑语句，用来估算软件更新在客户计算机上安装的适应性。相反，取代软件更新标识已被本软件更新取代的其它软件更新。与以上 Properties 元素 906 一样，在更新元数据文件中可有零个或多个 Relationships 元素 910。

ApplicabilityRules 元素 912 提供规则或测试，用来确定软件更新是否可应用在和/或适于安装在计算系统上。尽管在某些方面与 Relationships 元素 910 的关系相类似，但 ApplicabilityRules 元素 912 测试与另一软件更新可能特别相关或可能不特别相关的计算机系统上的条件。与以上标识的大部分其它元素一样，在更新元数据文件中可有零个或多个 ApplicabilityRules 元素 912。

Files 元素 914 标识关联于软件更新的文件，即更新有效负载，以及与那些文件相关的信息。该另外的信息包括但不限于：在更新有效负载中是否有一个或多个文件；从中可获取文件的位置；文件尺寸；文件名；文件创建日期；等等。本领域技术人员将容易理解单个软件更新可用多种方式安装在客户计算机上。作为一个示例，可通过用补丁修改部分现有文件或者通过用较新版本替换现有文件，来安装相同的软件更新。因而，根据本发明的诸方面，Files 元素 914 的更新有效负载可包括或指向：现有文件的补丁、替换文件、或者补丁和替换文件两者。在更新元数据文件中可有零个或多个 Files 元素 914。

可以各种格式提供更新，诸如描述要用δ补丁信息重写的文件区域的δ补丁、可执行文件、替换文件等等。这些类型更新的每一种需要特定更新处理器，以便在计算系统上执行软件更新。相应地，HandlerSpecificData 元素 916 提供更新元数据

文件中的位置，用来包括处理器专用数据/信息。例如，该信息可包括但不限于，处理器应当在其中执行的目录、对于处理器的命令行变量、如果安装的某些部分失败要采取的动作、如果安装成功要采取的动作、等等。更新元数据文件可包括零个或多个 HandlerSpecificData 元素 916。

尽管已说明和描述了本发明的包括较佳实施例的各种实施例和方面，可以理解能作出各种更改也不背离本发明的精神和范围。例如，尽管本发明已被描述为经通信网络通过更新分发系统传送软件更新，更新元数据以及更新有效负载可在计算机可读介质，诸如压缩盘或软盘上，从更新服务节点传送到客户计算机上。

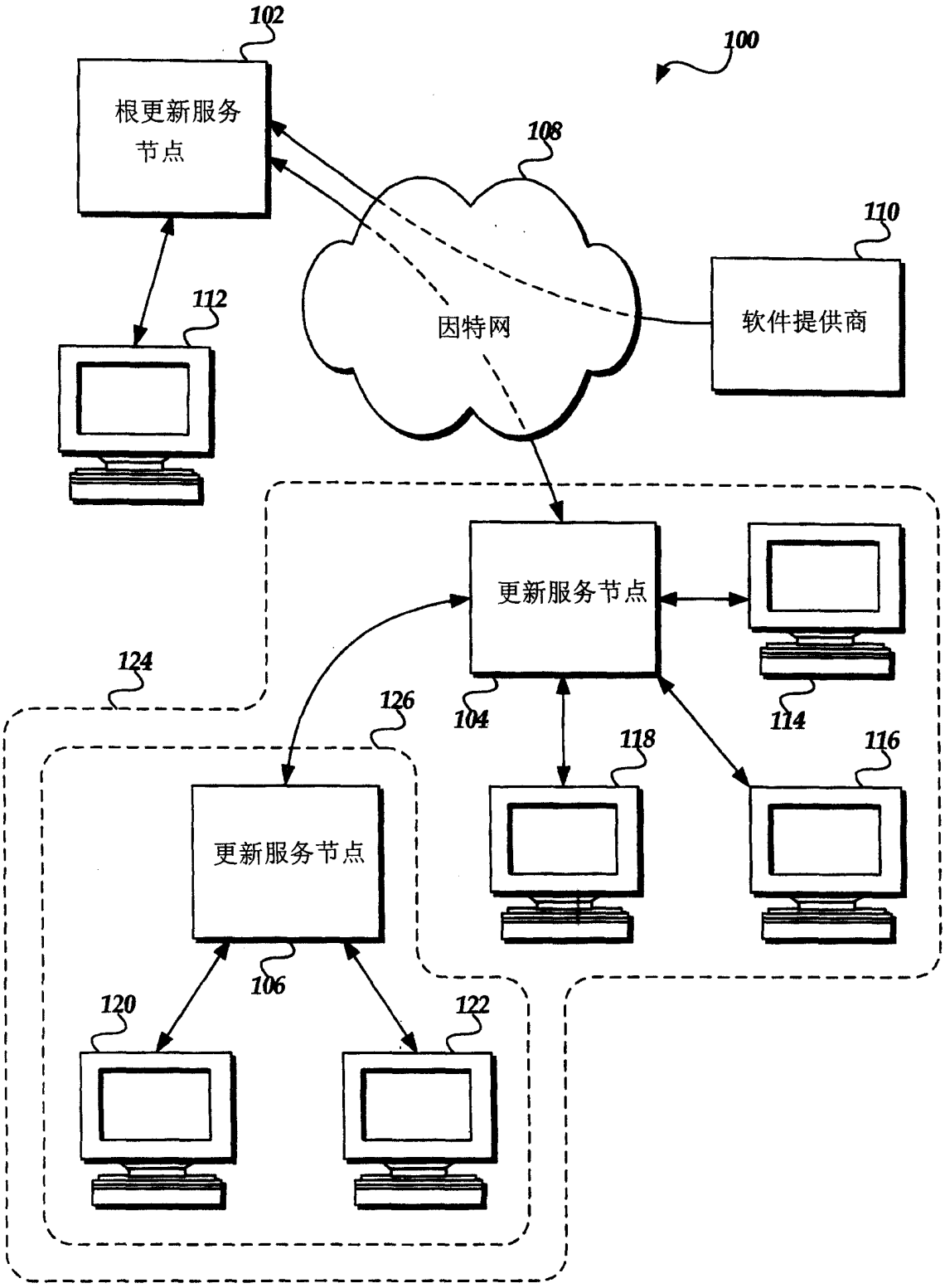

102
100
根更新服务节点
108
110
因特网
软件提供商
112
更新服务节点
124
126
104
114
118
116
更新服务节点
106
120
122

图 1

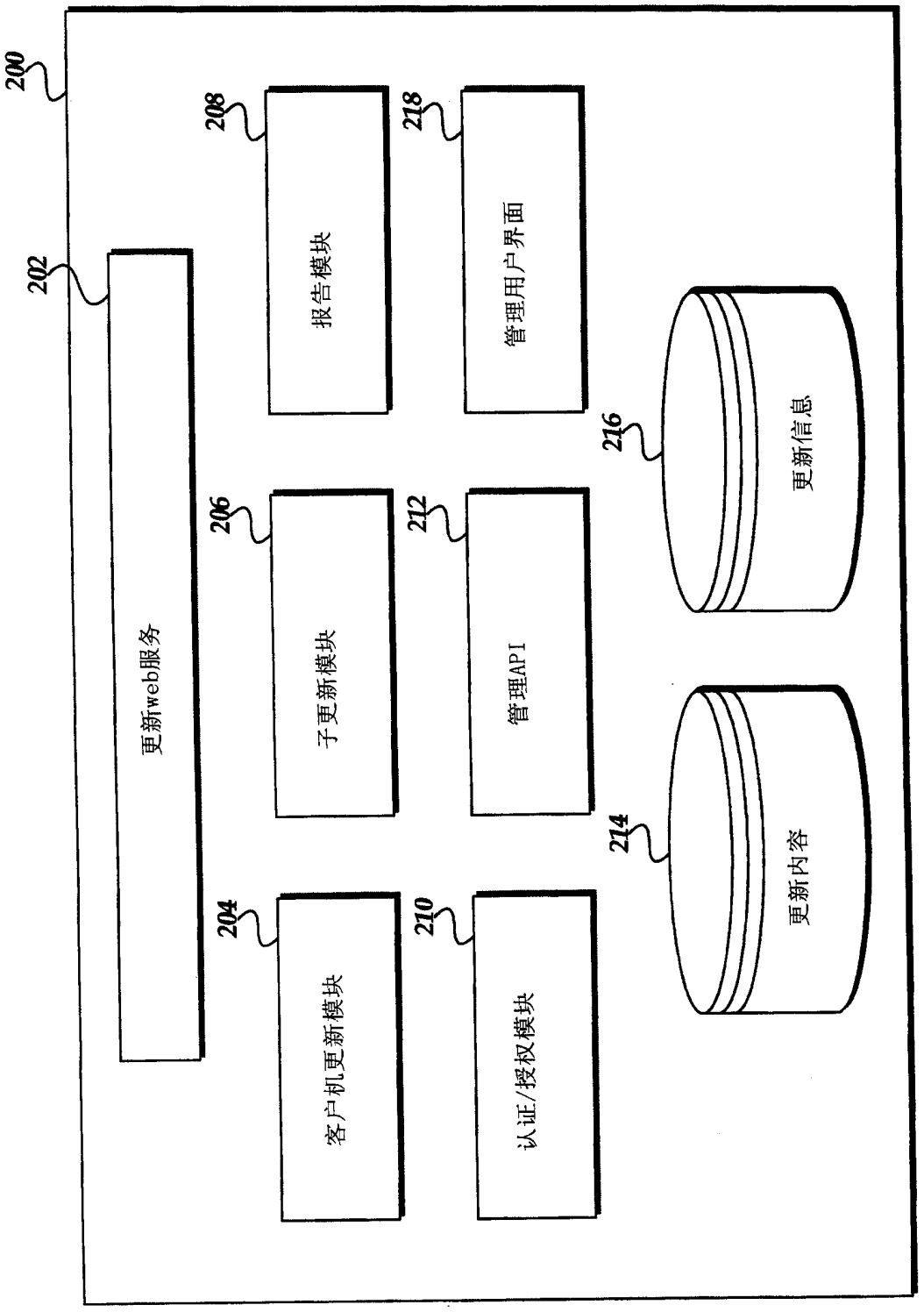

200
202
更新web服务
208
报告模块
218
管理用户界面
206
子更新模块
212
管理API
216
更新信息
204
客户机更新模块
210
认证/授权模块
214
更新内容

图 2

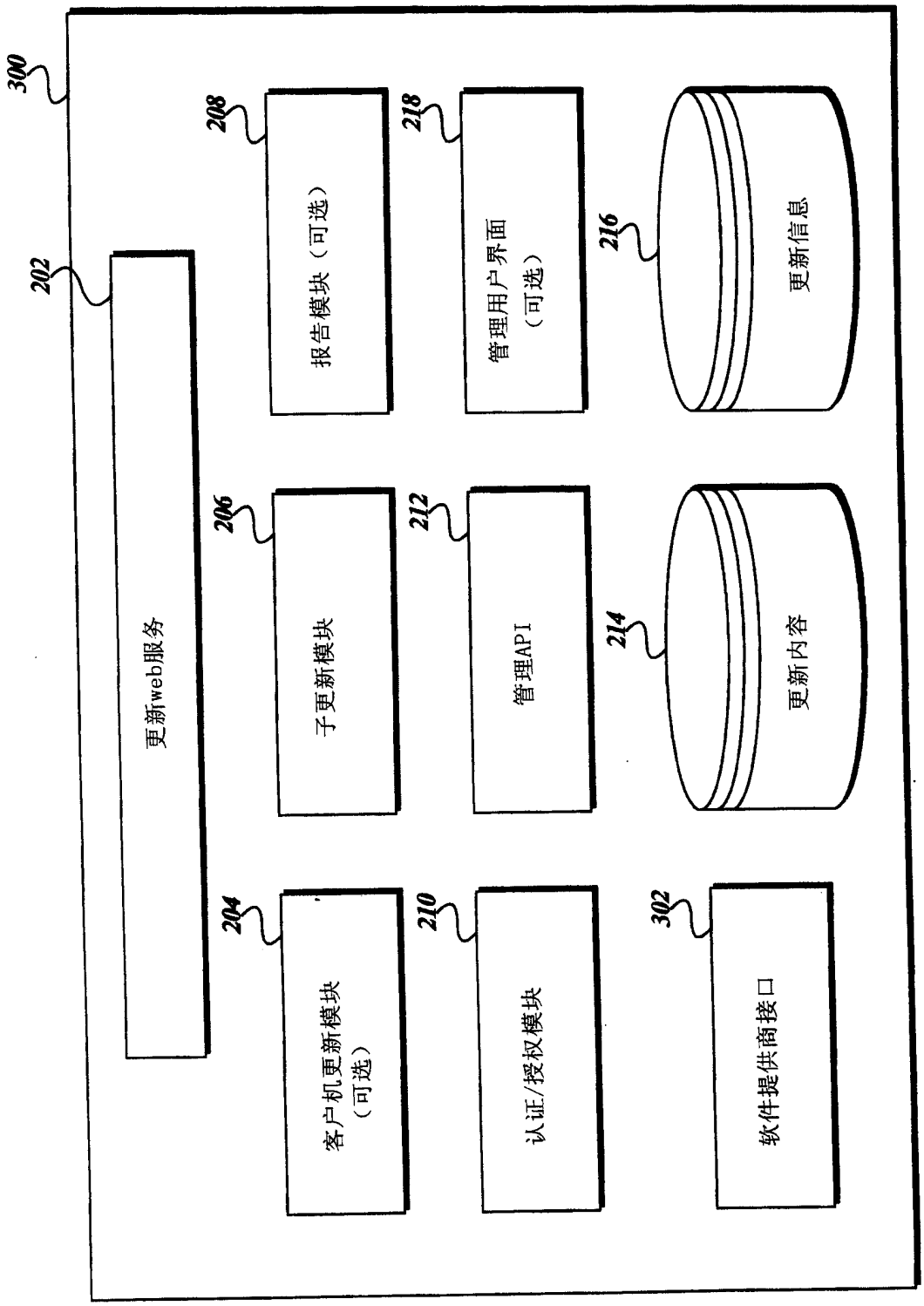

300
202
更新web服务
208
报告模块（可选）
218
管理用户界面
（可选）
216
更新信息
206
子更新模块
212
管理API
214
更新内容
204
客户机更新模块
（可选）
210
认证/授权模块
302
软件提供商接口

图 3

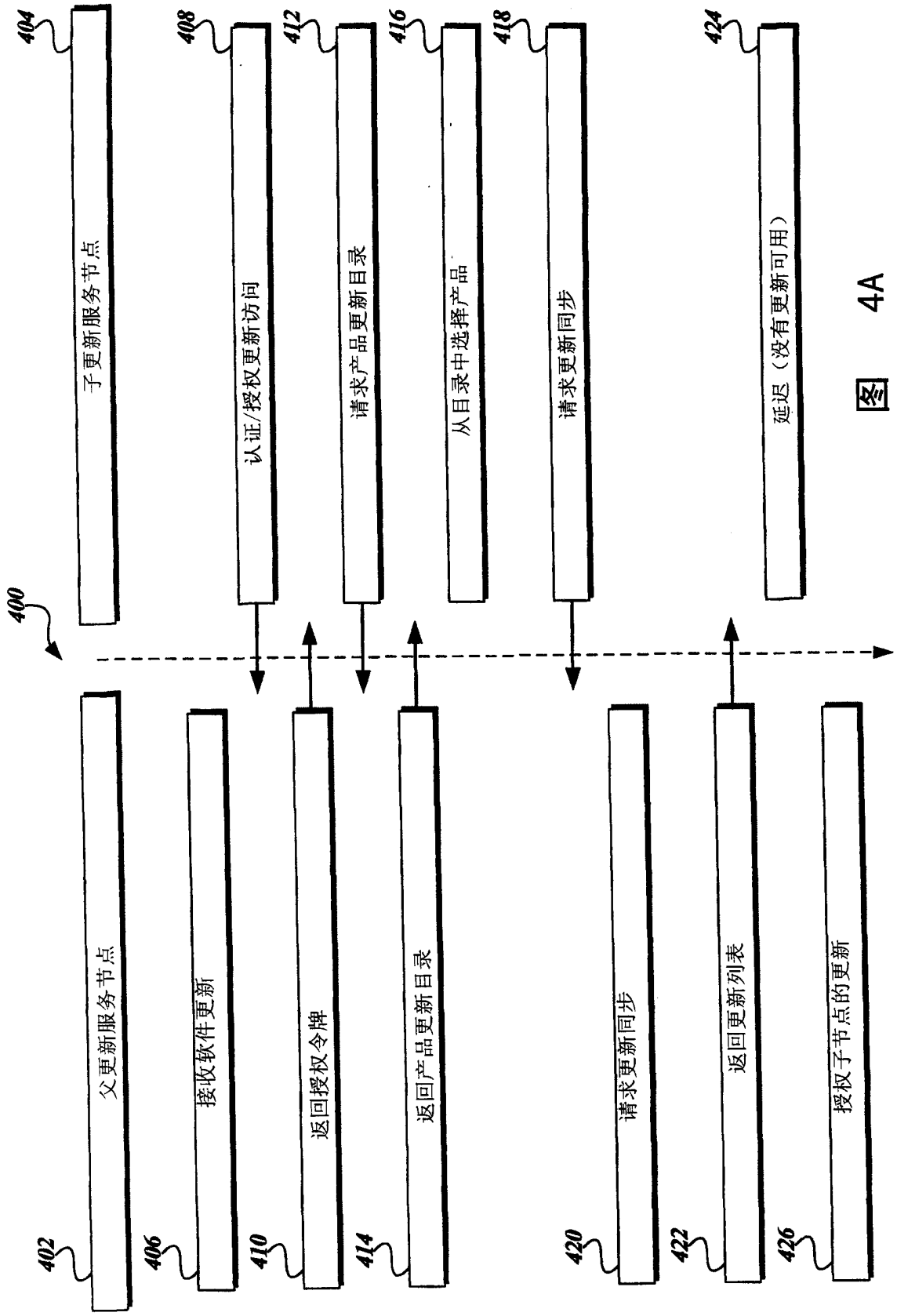

400
402 父更新服务节点
404 子更新服务节点
406 接收软件更新
408 认证/授权更新访问
410 返回授权令牌
412 请求产品更新目录
414 返回产品更新目录
416 从目录中选择产品
418 请求更新同步
420 请求更新同步
422 返回更新列表
424 延迟（没有更新可用）
426 授权子节点的更新

图 4A

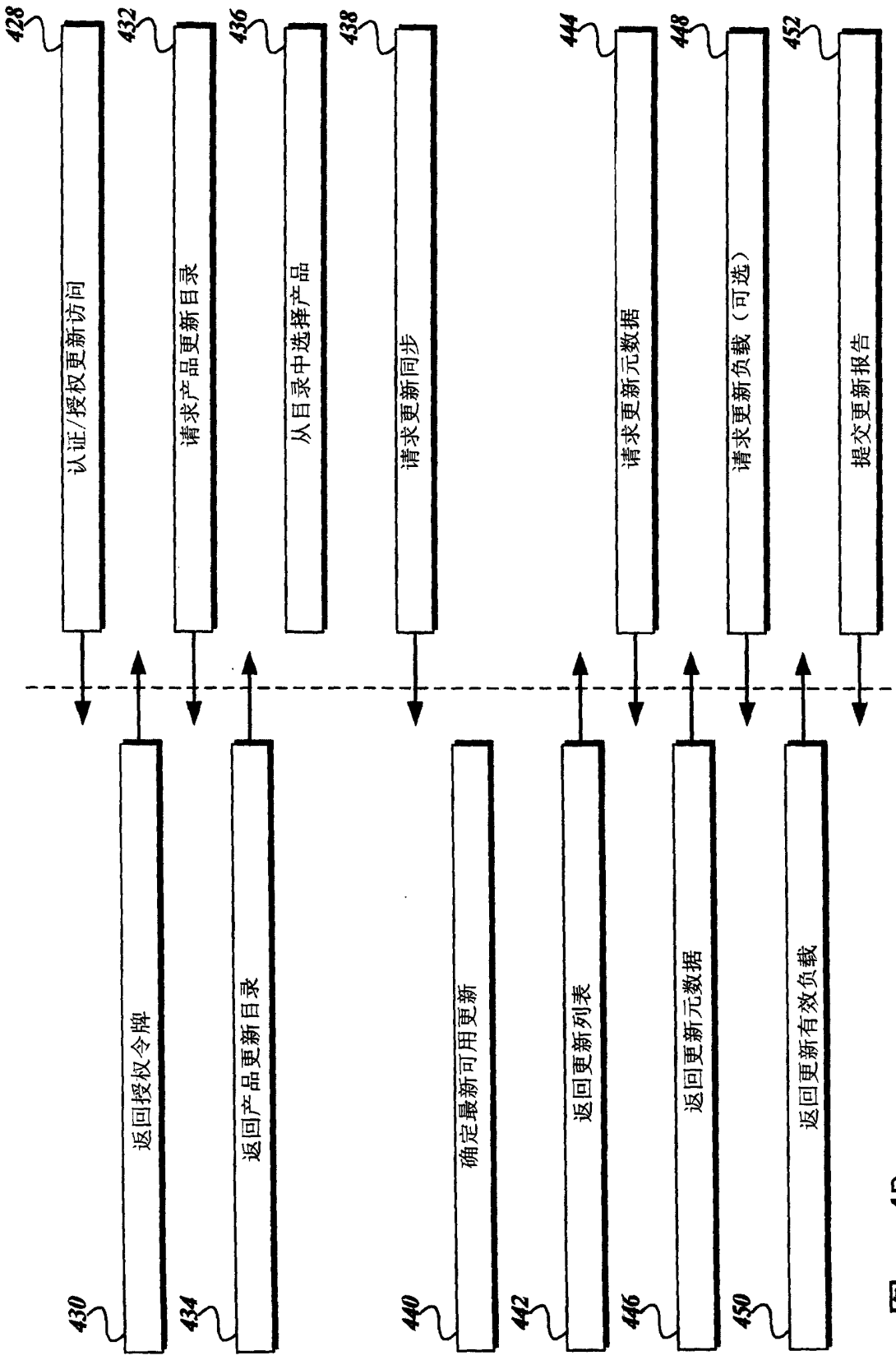

428
认证/授权更新访问
432
请求产品更新目录
436
从目录中选择产品
438
请求更新同步
444
请求更新元数据
448
请求更新负载（可选）
452
提交更新报告
430
返回授权令牌
434
返回产品更新目录
440
确定最新可用更新
442
返回更新列表
446
返回更新元数据
450
返回更新有效负载

图 4B

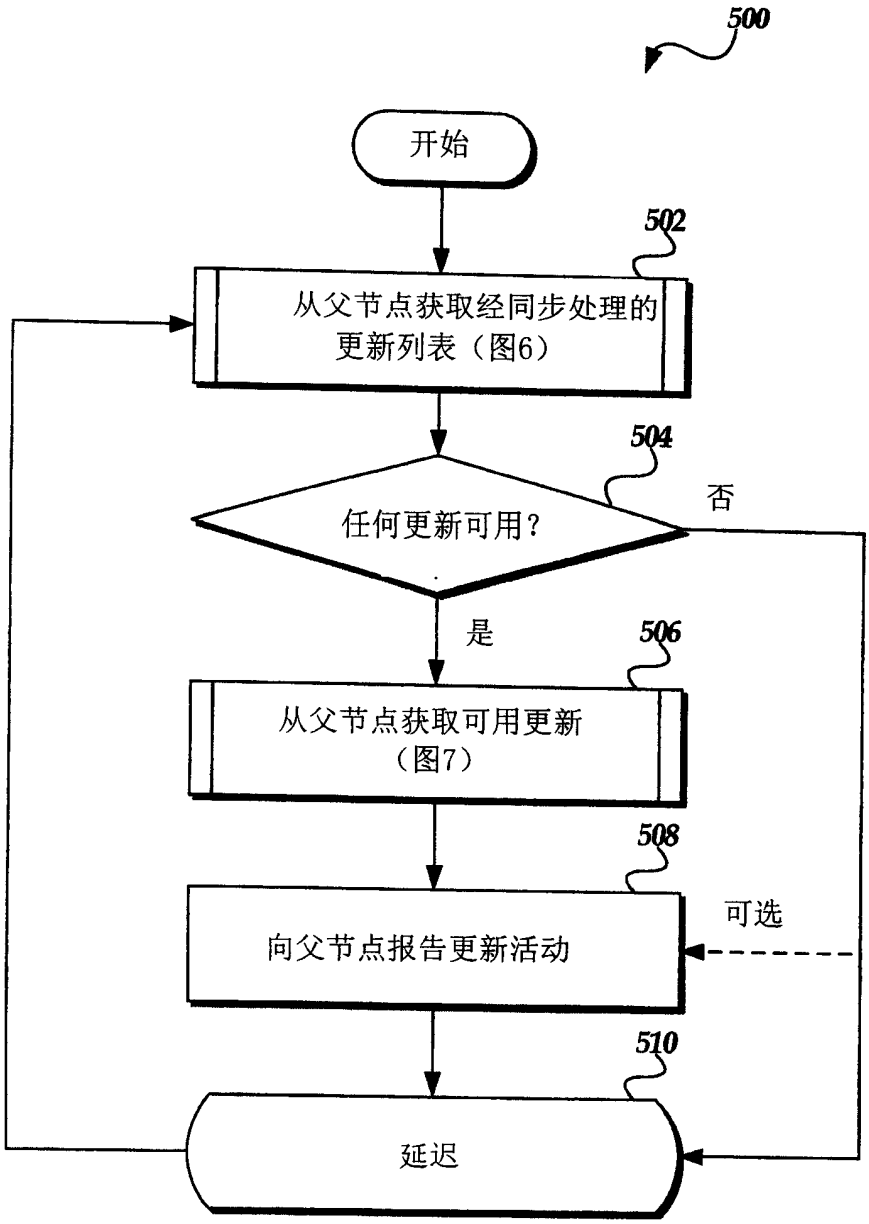

500
开始
502
从父节点获取经同步处理的更新列表（图6）
504
任何更新可用？
否
是
506
从父节点获取可用更新（图7）
508
向父节点报告更新活动
可选
510
延迟

图 5

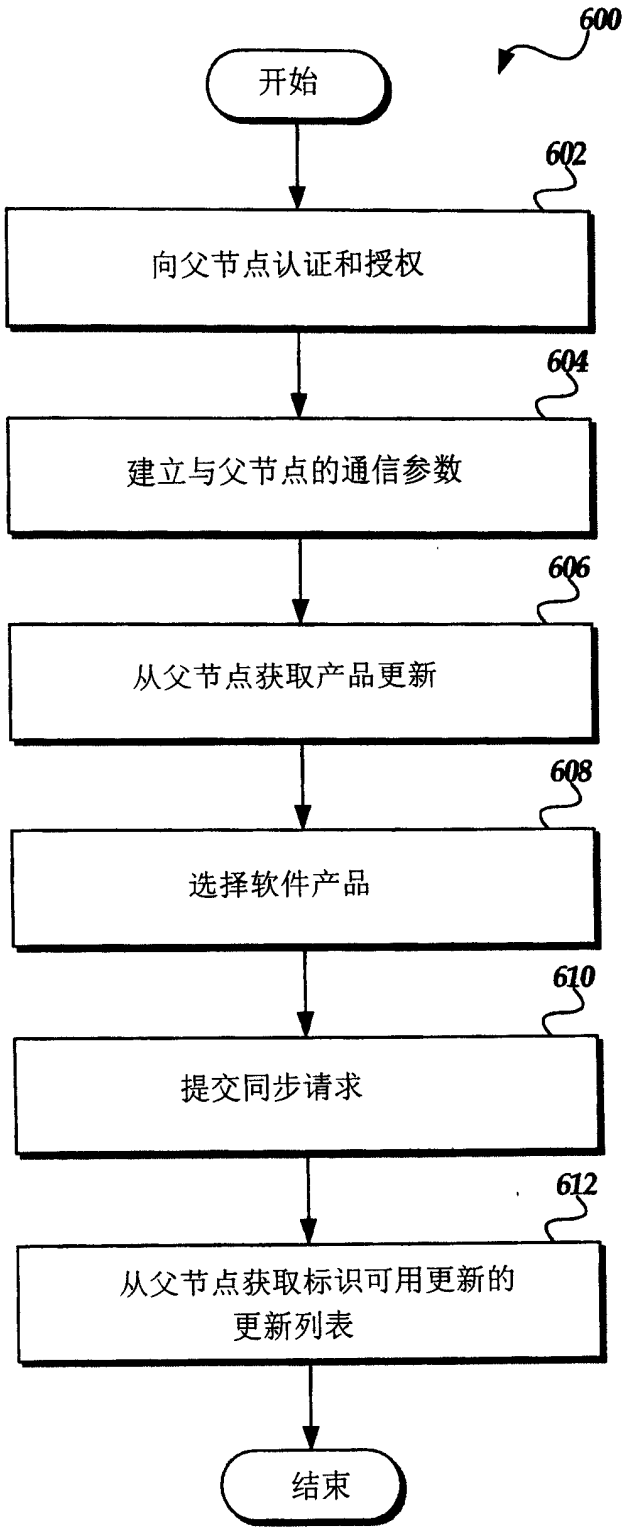

600
开始
602
向父节点认证和授权
604
建立与父节点的通信参数
606
从父节点获取产品更新
608
选择软件产品
610
提交同步请求
612
从父节点获取标识可用更新的
更新列表
结束

图 6

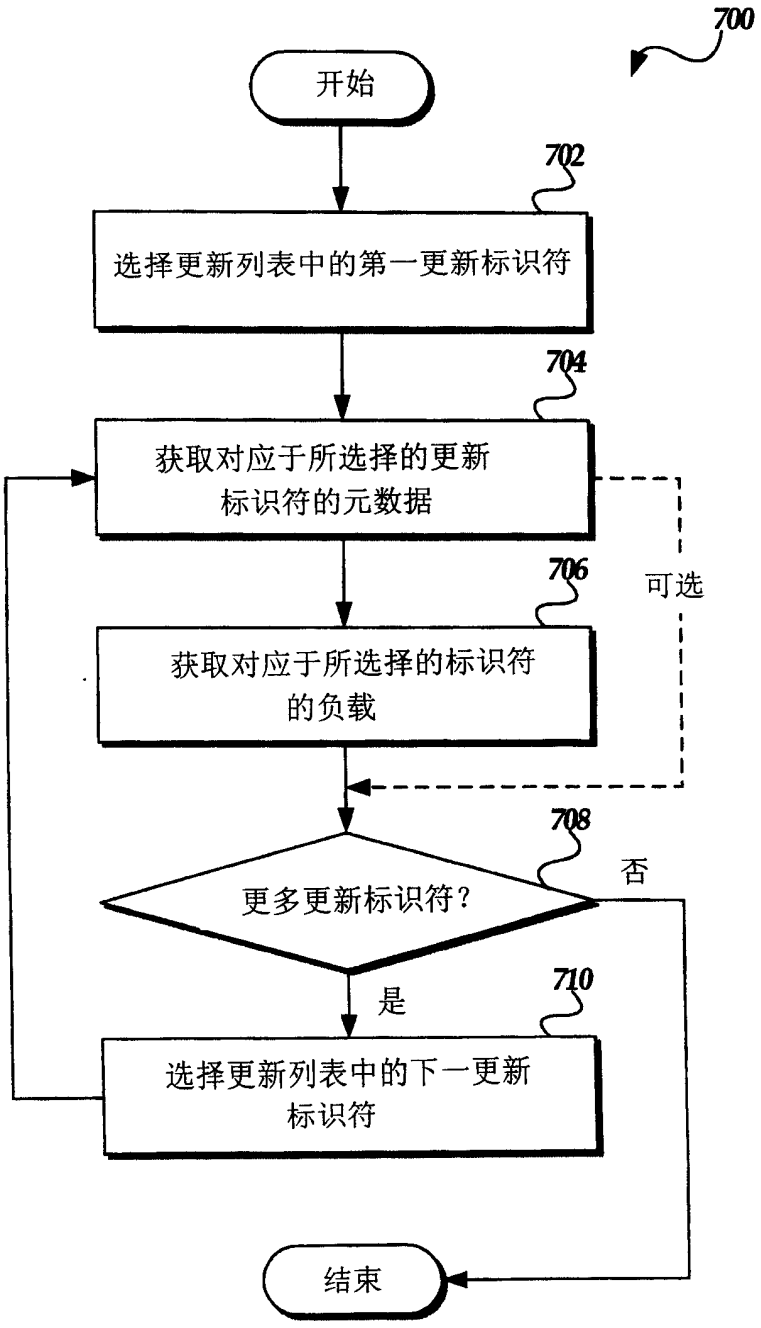

700
开始
702
选择更新列表中的第一更新标识符
704
获取对应于所选择的更新标识符的元数据
706
获取对应于所选择的标识符的负载
可选
708
更多更新标识符?
否
是
710
选择更新列表中的下一更新标识符
结束

图 7

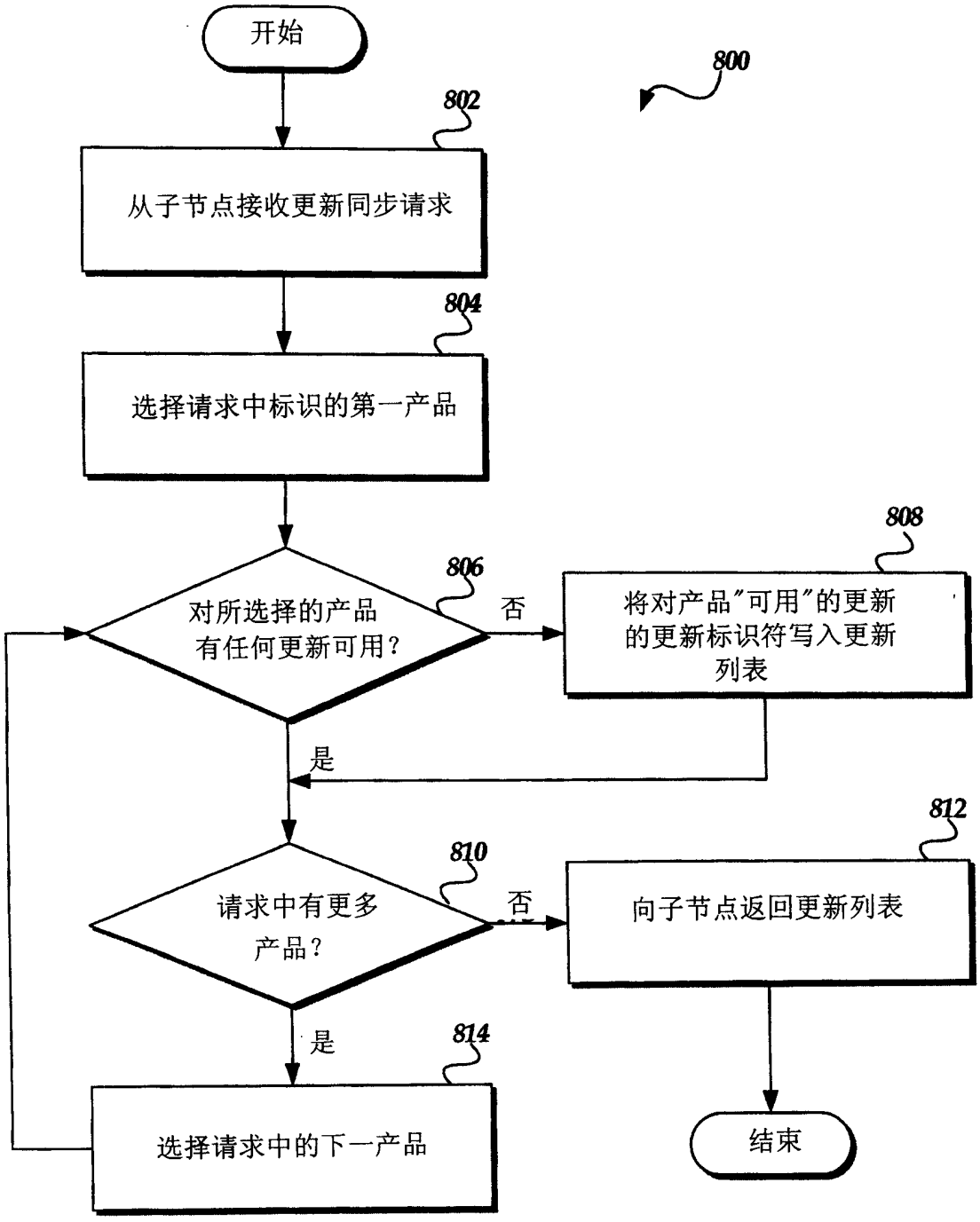

开始
800
802
从子节点接收更新同步请求
804
选择请求中标识的第一产品
808
806
对所选择的产品
有任何更新可用?
否
将对产品"可用"的更新
的更新标识符写入更新
列表
是
812
810
请求中有更多
产品?
否
向子节点返回更新列表
是
814
选择请求中的下一产品
结束

图 8

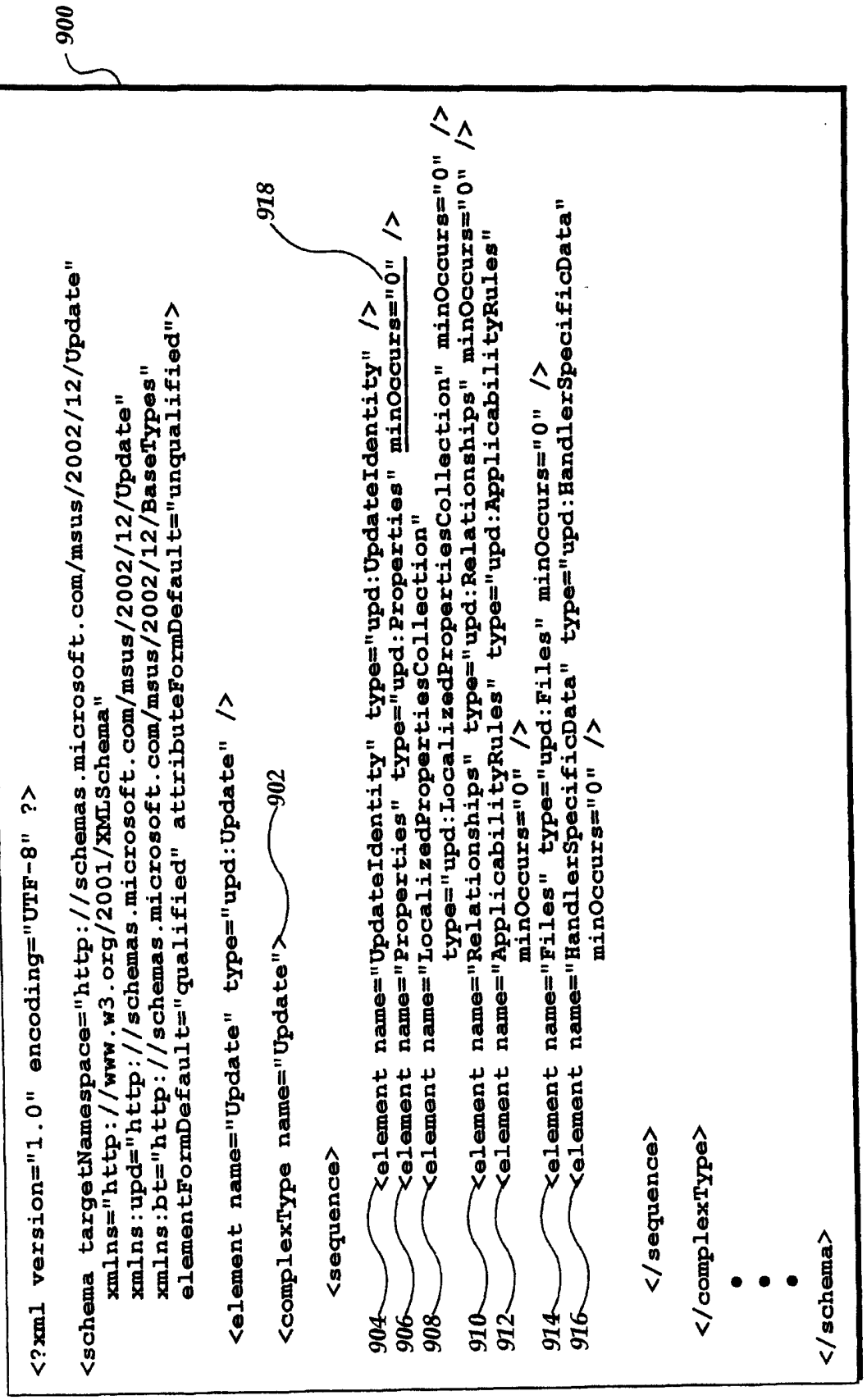

900
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://schemas.microsoft.com/msus/2002/12/Update"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:upd="http://schemas.microsoft.com/msus/2002/12/Update"
xmlns:bt="http://schemas.microsoft.com/msus/2002/12/BaseTypes"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<element name="Update" type="upd:Update" />
<complexType name="Update">
902
<sequence>
918
904 <element name="UpdateIdentity" type="upd:UpdateIdentity" />
906 <element name="Properties" type="upd:Properties" minOccurs="0" />
908 <element name="LocalizedPropertiesCollection"
type="upd:LocalizedPropertiesCollection" minOccurs="0" />
910 <element name="Relationships" type="upd:Relationships" minOccurs="0" />
912 <element name="ApplicabilityRules" type="upd:ApplicabilityRules"
minOccurs="0" />
914 <element name="Files" type="upd:Files" minOccurs="0" />
916 <element name="HandlerSpecificData" type="upd:HandlerSpecificData"
minOccurs="0" />
</sequence>
</complexType>
•
•
•
</schema>

图 9