(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101650670 B
(45)授权公告日 2013.01.09

(21)申请号 200810303758.7

(22)申请日 2008.08.14

(73)专利权人 鸿富锦精密工业(深圳)有限公司
地址 518109 广东省深圳市宝安区龙华镇油松第十工业区东环二路 2 号
专利权人 鸿海精密工业股份有限公司

(72)发明人 杜耀宏

(51)Int.Cl.
G06F 9/46(2006.01)
H04L 29/06(2006.01)

(56)对比文件
US 2001/0004746 A1,2001.06.21, 全文 .
CN 101059763 A,2007.10.24, 全文 .
CN 101038545 A,2007.09.19, 全文 .

审查员 何明伦

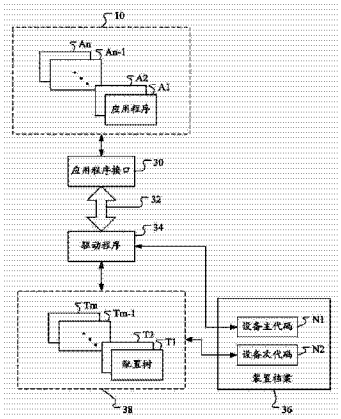
权利要求书 2 页 说明书 6 页 附图 9 页

(54)发明名称

可共享应用程序配置参数的电子系统及其方法

(57)摘要

一种可共享应用程序配置参数的电子系统，包括内存、接口模块及装置驱动程序。多个应用程序及文件储存于所述内存中的用户空间，所述文件记录有多个配置参数。所述接口模块响应所述应用程序的请求以解析所述文件，以及取得所述配置参数。所述装置驱动程序建立所述配置参数的数据结构于所述内存中的内核空间，让所述电子系统的多个应用程序经由所述装置驱动程序从所述数据结构共同存取所述配置参数，以执行所述电子系统的不同功能。



CN 101650670 B

1. 一种可共享应用程序配置参数的系统，应用于电子系统中，所述电子系统包括内存及处理器，

所述内存包括用户空间及内核空间，所述用户空间存储多个应用程序及文件，所述文件记录多个配置参数，其中所述配置参数由第一应用程序使用，

所述电子系统执行接口模块及装置驱动程序以执行所述可共享应用程序配置参数的，其特征在于，所述可共享应用程序配置参数的系统包括：

配置参数取得装置，用于通过所述接口模块以响应所述第一应用程序的请求以解析包括所述配置参数的所述文件，以及取得所述配置参数；

数据结构建立装置，用于通过所述装置驱动程序建立第一应用程序使用的所述配置参数的数据结构于所述内存的内核空间；及

功能执行装置，用于通过所述多个应用程序从所述数据结构存取所述配置参数，以执行所述电子系统的不同功能。

2. 如权利要求 1 所述的可共享应用程序配置参数的系统，其特征在于，所述内存还包括操作系统的内核程序，所述内核程序经由所述装置驱动程序从所述数据结构存取所述第一应用程序的所述配置参数。

3. 如权利要求 1 所述的可共享应用程序配置参数的系统，其特征在于，所述装置驱动程序关联装置档案，用以表示所述装置驱动程序所控制的对象，使所述装置驱动程序所控制的对象经由所述电子系统提供的系统呼叫以及所述装置档案的文件名而被存取，所述装置驱动程序所控制的对象为包括所述第一应用程序的配置参数的所述数据结构的集合。

4. 如权利要求 3 所述的可共享应用程序配置参数的系统，其特征在于，所述装置档案包括设备主代码及设备次代码，所述设备主代码代表所述装置驱动程序，若所述第一应用程序使用所述配置参数的多个数据结构，每一笔设备次代码用以表示所述多个数据结构中的一个。

5. 如权利要求 4 所述的可共享应用程序配置参数的系统，其特征在于，所述配置参数的数据结构为树状结构，所述装置档案为字符装置档案。

6. 如权利要求 1 所述的可共享应用程序配置参数的系统，其特征在于，所述第一应用程序用以取得所述电子系统的网络通讯协议地址，并记录于所述数据结构中，所述电子系统的第二应用程序读取所述网络通讯协议地址以执行网络地址变换。

7. 如权利要求 1 所述的可共享应用程序配置参数的系统，其特征在于，所述配置参数的数据结构为具有多个节点的树状结构，所述第一应用程序用以在所述数据结构中的一个节点中设定配置参数值，以决定所述电子系统中的多个程序记录其相关事件的事件类别，所述电子系统的内存中的所述用户空间中具有第二应用程序，所述第二应用程序读取所述节点的配置参数值以决定记录于所述记录文件的事件类别。

8. 如权利要求 1 所述的可共享应用程序配置参数的系统，其特征在于，所述第一应用程序于所述数据结构中设定从远程访问电子系统的用户的至少一远程装置的网络通讯地址，所述电子系统的第二应用程序读取并根据所述至少一远程装置的网络通讯地址以容许或拒绝所述至少一远程装置连接至电子系统。

9. 一种可共享应用程序配置参数的方法，执行于电子系统中，所述电子系统包括内存及处理器，所述内存存储有应用程序及装置驱动程序，介于所述应用程序及装置驱动程序

之间还存在接口模块，所述接口模块包括多个用于接收功能请求的软件功能模块，所述处理器用于执行程序，所述电子系统执行所述接口模块，所述应用程序及所述装置驱动程序以执行所述可共享应用程序配置参数的方法，其特征在于，所述方法包括：

所述应用程序请求所述接口模块以使用所述应用程序的配置参数，其中所述应用程序储存于所述电子系统的所述内存中的用户空间，并且所述应用程序的配置参数供所述电子系统的至少二个应用程序使用；

所述接口模块响应所述应用程序的请求以解析包括所述应用程序的配置参数的文件，以及取得所述配置参数；及

所述装置驱动程序建立所述配置参数的数据结构于所述内存中的内核空间，让所述至少二个应用程序经由所述装置驱动程序从所述数据结构共同存取所述配置参数，以执行所述电子系统的不同功能。

10. 如权利要求 9 所述的可共享应用程序配置参数的方法，其特征在于，所述应用程序用以取得所述电子系统的网络通讯协议地址，并记录于所述数据结构中，所述电子系统的第二应用程序读取所述网络通讯协议地址以执行网络地址变换。

可共享应用程序配置参数的电子系统及其方法

技术领域

[0001] 本发明涉及一种计算机技术，尤其涉及一种可共享应用程序配置参数的电子系统。

背景技术

[0002] 在程序开发中经常使用可扩展标记语言 (eXtensible Markup Language,XML) 来存贮配置参数 (configuration)，应用程序调用 XML 解析库或利用其解析函数，来解析自已的 XML 格式的配置文件，以生成配置树。这方法既简单又直观，因此已被广泛采用。

[0003] 但是在实践中，应用程序 (application) 彼此之间、或应用程序与操作系统 (operating system) 内核 (kernel) 之间在需要共享配置时，用现有的方法难以完成。详细情形请参照图 1，内存 100 中包括了用户空间 (user space) 及内核空间 (kernel space)。应用程序群组 10a ～ 10b 及记录配置参数的文件 12 皆设置于用户空间；而内核程序位于内核空间。任何的应用程序要存取其配置参数时，会从解析文件 12 的 XML 内容以取得配置参数，并依据 XML 格式的内容中各配置参数的结构关系以建立配置参数的关系树，或称为配置树，例如图 1 中的配置树 11a ～ 11c。而配置参数关系树中的每一个节点记录配置参数的值。应用程序会将配置参数关系树设置于内存 100 中其应用程序本身可以存取的位置，而其它的应用程序则不知道其位置，也没有权限可以存取。这样会造成各配置树内的参数值难以同步，也难在程序之间共享。

[0004] 然而有些参数需要在程序之间共享。举例来说，在非对称式数字用户线 (Asymmetric Digital Subscriber Line，简称 ADSL) 调制解调器中，在利用点对点协议 (Point-To-Point Protocol,PPP) 或动态主机组态协议 (dynamic host configuration protocol，简称 DHCP) 取得网络通讯协议 (Internet protocol，简称 IP) 地址后，所述调制解调器中的其它程序例如网络地址变换 (network address translation，NAT) 或防火墙 (Firewall) 程序都需要取用所述 IP 地址。需要共享并同步处理的参数在所述的配置树目前架构下难以达成其目的。这样导致的结果是程序之间无法共享配置参数，操作系统的内核程序 13 也很难使用配置树或 XML 配置参数文件。

发明内容

[0005] 本发明提供一种可共享应用程序配置参数的电子系统，包括内存、接口模块及装置驱动程序。所述内存，包括用户空间及内核空间，所述用户空间存储多个应用程序及文件，所述文件记录多个配置参数，其中所述配置参数由第一应用程序使用。所述接口模块响应所述第一应用程序的请求以解析包括所述配置参数的所述文件，以及取得所述配置参数。所述装置驱动程序建立所述配置参数的数据结构于所述内存中的内核空间，让所述电子系统的多个应用程序经由所述装置驱动程序从所述数据结构共同存取所述配置参数，以执行所述电子系统的不同功能。

[0006] 本发明还提供一种可共享应用程序配置参数的方法，执行于电子系统中。所述电

子系统包括应用程序、接口模块及装置驱动程序。所述应用程序请求所述接口模块以使用所述应用程序的配置参数，其中所述应用程序储存于所述电子系统的内存中的用户空间。所述接口模块响应所述应用程序的请求以解析包括所述应用程序的配置参数的文件，以及取得所述配置参数。所述装置驱动程序建立所述配置参数的数据结构于所述内存中的内核空间，让所述电子系统的多个应用程序经由所述装置驱动程序从所述数据结构共同存取所述配置参数，以执行所述电子系统的不同功能。

[0007] ADSL 调制解调器的例子中，利用本发明，在 PPP 或 DHCP 程序取得 IP 地址后，所述调制解调器中的 NAT 或防火墙程序都可以通过装置驱动程序取用所述 IP 地址以执行其 NAT 或防火墙功能。

附图说明

[0008] 图 1 为配置树在内存中的传统设置示意图。

[0009] 图 2 为电子系统的结构方块图。

[0010] 图 3 为配置树在内存中的新设置示意图。

[0011] 图 4 为所述电子系统的模块示意图。

[0012] 图 5 显示配置参数文件的实例示意图。

[0013] 图 6 显示配置参数文件中的区块 B1 转换成配置树的部分的示意图。

[0014] 图 7 显示电子系统内的配置树的建立流程图。

[0015] 图 8 显示电子系统内的配置树的修改流程图。

[0016] 图 9 显示电子系统内的配置树中特定节点的读取流程图。

具体实施方式

[0017] 本发明可以执行于各种电子装置或系统，例如路由器、非对称式数字用户线 (Asymmetric Digital Subscriber Line，简称 ADSL) 装置、缆线调制解调器 (cable modem)、以及机上盒 (Set Top Box)。

[0018] 参阅图 2，电子系统 200 包括非挥发性内存 (Nonvolatile Memory)210、处理器 220、主存储器 230 及通讯单元 240。通讯单元 240 可以包括通讯端口及通讯相关的组件，例如无线通讯的控制器及天线、ADSL 装置中的数字讯号处理器 (Digital Signal Processor) 及模拟至数字转换器 (Analog-to-Digital Converter)、缆线调制解调器中的调谐器 (tuner)、以太网络 (Ethernet) 控制器、通用串行总线 (Universal Serial Bus，USB) 控制器及 / 或外围组件互联标准 (Peripheral Component Interconnect，PCI) 控制器。处理器 220 可以由集成电路 (Integrated Circuit，简构 IC) 组成，用以处理数据及执行程序。处理器 10 可以是由单颗封装的 IC 所组成，或连接多颗封装的 IC 而组成。举例来说，处理器 10 可以仅包括中央处理器 (Central Processing Unit，简称 CPU)，或者是 CPU、通讯控制器的组合。所述的通讯控制器用以控制电子系统 200 中的各组件的通讯，或电子系统 200 与外部装置的通讯。在不同的实施方式中，所述各种通讯组件可以整合在通讯单元 240 或处理器 220 之中。

[0019] 主存储器 230 可以包括各种随机存取内存 (Random Access Memory，简称 RAM)；非挥发性内存 210 的实例如电子可抹除可程序化只读存储器 (Electrically Erasable

Programmable ROM, 简称 EEPROM) 或闪存 (Flash Memory)。非挥发性内存 210 包括电子系统 200 的操作系统及应用程序。非挥发性内存 210 中的程序及数据可以数据压缩的形式储存，待需执行或使用时再解压缩至主存储器 230。

[0020] 请参照图 3，电子系统 200 可以包括内存 100A。内存 100A 可以是虚拟内存，实际上可以映像至主存储器 230 及 / 或非挥发性内存 210。内存 100A 包括用户空间及内核空间。配置参数文件 12 具有应用程序 10c 的多个配置参数，其中所述配置参数至少有二个程序使用。

[0021] 为了达到配置参数共享及同步化的目的，电子系统 200 的其中一个应用程序 (例如图 3 中的应用程序 10c) 启动对配置参数文件 12 的解析程序后，经由内核程序 13 根据配置参数文件 12 中各配置参数的关系，以建立配置参数文件 12 中各配置参数的配置树 (例如配置树 11d) 于内存 100A 的内核空间。所有要使用到配置树 11d 中的参数的应用程序皆需通过内核程序 13 来执行读取或修改。虽然在此实施方式中，配置参数的数据结构为树状结构，但是并非限定于此，也可以利用数组、链接串行 (linked list) 等数据结构。应用程序 10a ～ 10c 皆可以通过内核程序 13 读取或修改配置树 11d 内的参数，藉此达到配置参数共享与同步化的目的。其具体实施方式请参照图 4。

[0022] 在图 4 中，电子系统 200 具有应用程序群组 10，其中包括多个应用程序 A1 ～ An，其中 n 为大于 1 的正整数。应用程序 A1 ～ An 的实例可包括图 3 中的应用程序群组 10a ～ 10c。电子系统 200 还具有应用程序接口 30、内核程序 13 提供的系统呼叫 32、驱动程序 34、及装置档案 (device file 或 device special file)36。

[0023] 应用程序接口 30 包括软件功能模块库 (library)，其中包括多个用以解析配置参数文件 (例如配置参数文件 12) 的软件功能模块 (function)，根据所述解析结果以建立对应配置树的软件功能模块、以及修改及读取配置树及其中参数值的软件功能模块。由于配置树在内核空间中，应用程序接口 30 的软件功能模块可以利用内核程序 13 提供的系统呼叫 (system call, 例如“ioctl”)，以驱使内核程序 13 致动驱动程序 34 以完成建立或修改配置树群组 38 中的配置树的动作。配置树群组 38 包括多个配置树 T1 ～ Tm，其中 m 为大于 1 的正整数。举例来说，配置树群组 38 包括图 3 中的配置树 11d。

[0024] 装置档案在 Unix、Linux 等以及相类似的操作系统中用以代表特定装置驱动程序及其对应的装置，例如储存于 /dev 的目录下，使应用程序可以通过标准的输出 / 输入系统呼叫来与所述装置档案代表的驱动程序互动。在本实施方式中，装置档案 36 的文件名为“xmlconf”，并且利用装置档案 36 对应配置树群组 38，以代表中的多个配置树。图 4 中，装置档案 36 中的设备主代码 (major number)N1 为代表驱动程序 34 的一个号码，而设备次代码 (minor number)N2 包括了代表配置树 T1 ～ Tm 的多个笔号码，每一笔设备次代码代表一个配置树。应用程序群组 10 中的应用程序、应用程序接口 30 中的软件功能模块或驱动程序 34 皆能够通过装置档案 36 以指定配置树群组 38 中的特定配置树，并予以读取或修改。

[0025] 应用程序开启设备文文件 /dev/xmlconf 就可以找到设备主代码 N1，通过设备号就可以请求内核程序 13 找到并利用驱动程序 34。驱动程序 34 利用一笔设备次代码就可以找到其对应的特定配置树，并根据所述应用程序的请求以修改所述配置树或读取并回传配置参数值给所述应用程序。

[0026] 装置档案的实例包括字符装置档案 (character special file 或 character

device) 及区块装置档案 (block special file 或 block device)。字符装置档案对应的字符装置与操作系统以字符为单位来交换数据，而区块装置与操作系统则以较大的数据区块为单位来交换数据。字符装置的实例包括调制解调器 (modem) 及电话通讯相关的装置。区块装置的实例包括硬式磁盘驱动器及光驱。驱动程序 34 每次读取配置树上一个节点的配置参数值，所以装置档案 36 可以是字符装置档案。

[0027] 实例：

[0028] 图 5 显示配置参数文件 12 的实例，其对应的配置树的一部分显示于图 6。配置参数文件 12a，其中的配置参数包括在 <ConfigTree> 与 </ConfigTree> 之间。<AtmCfg> 与 </AtmCfg> 中间包括异步传输模式 (asynchronous transfer mode，简称 ATM) 相关的设置。<AtmCfgTd> 与 </AtmCfgTd> 之间包括关于 ATM 连接类型 (Link Type) 的设置，比如 " CBR" 表示用恒定的速率连接。<AtmCfgVcc> 与 </AtmCfgVcc> 之间主要包括 ATM 层的持久虚拟线路 (permanent virtual circuit，简称 PVC) 参数。<SecCfg> 与 </SecCfg> 之间主要包括网络安全的相关设置。<WirelessCfg> 与 </WirelessCfg> 之间是 Wi-Fi(IEEE 802.11) 相关的设置。<RouteCfg> 与 </RouteCfg> 之间主要设置电子系统 200 的静态路由表。<PMapCfg> 与 </PMapCfg> 之间主要是做埠对应 (Port Map) 用的设置。<SNTPCfg> 与 </SNTPCfg> 之间是设置时间服务器的参数，即与简单网络时间协议 (Simple Network Time Protocol，SNTP) 服务器相关的参数。<Voice> 与 </Voice> 之间是网络电话 (Voice Over Internet Protocol，简称 VoIP) 相关的设置。<pppsrv_8_35> 与 </pppsrv_8_35> 之间是点对点协议 (Point-To-Point Protocol，PPP) 连接 PPP 服务器时的相关设置。<wan_8_35> 与 </wan_8_35> 之间的配置参数主要用于建立广域网络 (Wide Area Network，简称 WAN) 端的接口有关，它和 ATM PVC 是紧密相关的。

[0029] 电子系统 200 中的内核程序 13 或应用程序皆可以启动配置参数文件 12a 转换成配置树的工作。举例来说，内核程序 13 在电子系统 200 开机时启动配置树的建立。或者，应用程序 Ai (其中 i 为正整数，且 1 ≤ i ≤ n) 启动配置树的建立。电子系统 200 内执行的内核程序 13 或应用程序 A1 ～ An 可以利用应用程序接口 30 将配置参数文件 12a 转换成特定的数据格式，再交给驱动程序 34 以建立配置参数文件 12a 的对应配置树。

[0030] 请参照图 7，当电子系统 200 内的程序需要建立配置树时，请求执行应用程序接口 30 中用来建立配置树的软件功能模块 (步骤 700)。其中所述请求中包对应驱动程序 34 的装置档案 36 的名称。所述软件功能模块响应所述程序的请求以分析配置参数文件 12a 来产生代表配置参数文件 12a 的结构及其内配置参数值的数据结构 (步骤 702)。<protocol> 卷标中的 autoScan 属性值可以在步骤 702 的数据结构中以“protocol.autoScan=enable”来表示。所述软件功能模块再通过电子系统 200 的操作系统的系统呼叫 32 以传递所述装置档案 36 的文件名、所述数据结构及其它数据至内核程序 13(步骤 704)。内核程序 13 根据所述文件名取得装置档案 36 中的设备主代码 N1，再根据设备主代码 N1 取得驱动程序 34，并将所述数据结构交给驱动程序 34，使驱动程序 34 开始建立所述数据结构对应的配置树 (步骤 706)。驱动程序 34 新增设备次代码 N2 以作为将要建立的配置树的代码 (步骤 708)，并根据所述数据结构以建立配置参数文件 12a 对应的配置树 Tj(其中 j 为正整数，且 1 ≤ j ≤ m)(步骤 710)。

[0031] 图 6 显示配置参数文件 12a 中的区块 B1 转换成配置树的示意图。区块 B1 中

<SystemInfo>与</SystemInfo>标签以“SystemInfo”为名，其间包括了以“protocol”、“sysLog”及“sysUserName”为名的标签。因此，在被程序请求的情况下，驱动程序 34 会在配置参数文件 12a 对应的配置树中建立“SystemInfo”、“protocol”、“sysLog”及“sysUserName”对应的节点，且“SystemInfo”为“protocol”、“sysLog”及“sysUserName”等节点的父节点。“protocol”标签包括“autoScan”、“upnp”、“igmpSnp”、“igmpMode”、macFilterPolicy”、“encodePassword”及“enetwan”的属性，相应地，驱动程序 34 会建立“autoScan”、“upnp”、“igmpSnp”、“igmpMode”、“macFilterPolicy”、“encodePassword”及“enetwan”等节点作为“protocol”节点的子节点，且每个属性对应的节点中包括其属性值。同理，驱动程序 34 也会建立“sysLog”及“sysUserName”节点的子节点，以及所述配置树的其它节点。

[0032] 在配置树 Tj 建立之后，其它的程序就可以读取或修改配置树 Tj 的节点内的配置参数值。请参照图 8，当电子系统 200 内的程序需要修改配置树中的特定节点（以下称为目标节点）时，请求执行应用程序接口 30 中用来修改配置树的软件功能模块（步骤 800）。其中所述请求中包括对应驱动程序 34 的装置档案 36 的名称、新的配置参数值及其它数据。举例来说，执行 PPP 的程序或动态主机组态协议 (Dynamic Host Configuration Protocol，简称 DHCP) 的程序从 WAN 取得电子系统 200 的新的 IP 地址作为新的配置参数值，随后更新配置树 Tj 中用来记录电子系统 200 的 IP 地址的目标节点中的 IP 地址。应用程序可以在步骤 800 的所述请求中指明所述目标节点的名称及从配置树 Tj 至所述目标节点的路径，其中所述路径包括目标节点的父节点及祖先节点的名称。

[0033] 所述软件功能模块响应所述程序的请求，通过电子系统 200 的操作系统的系统呼叫将所述请求中的文件名、目标节点名称及新的配置参数值传递给内核程序 13（步骤 802）。内核程序 13 根据所述文件名取得装置档案 36 中的设备主代码 N1，再根据设备主代码 N1 取得驱动程序 34，并将所述新的配置参数值交给驱动程序 34，使驱动程序 34 更新配置树 Tj 中目标节点内的配置参数值（步骤 804）。驱动程序 34 取得配置树 Tj 中的所述目标节点（步骤 806），并以所述新的配置参数值来更新目标节点内的配置参数值（步骤 808）。步骤 806 中，驱动程序 34 可以根据所述目标节点的路径以取得所述目标节点，或搜寻配置树 Tj 以取得。

[0034] 在图 6 中配置树 Tj 的例子上，电子系统 200 的 IP 地址是从区块 B2 中的“entry1”标签的 address=“192.168.1.1”的属性取得的。在所述 IP 地址更新后，电子系统 200 中的防火墙程序可以从配置树 Tj 读取电子系统 200 的 IP 地址，并根据所述节点中新的 IP 地址以执行其任务。电子系统 200 的网络地址变换 (network address translation，NAT) 程序根据所述节点中新的 IP 地址以执行 NAT。

[0035] 请参照图 9，当电子系统 200 内的程序需要读取配置树中的特定节点（以下称为目标节点）时，请求执行应用程序接口 30 中用来读取配置树的软件功能模块（步骤 900）。其中所述请求中包对应驱动程序 34 的装置档案 36 的名称及目标节点的名称。应用程序可以在步骤 900 的所述请求中指明从配置树 Tj 至所述目标节点的路径，其中所述路径包括目标节点的父节点及祖先节点的名称。

[0036] 所述软件功能模块响应所述程序的请求，通过电子系统 200 的操作系统的系统呼叫将所述请求中的文件名及目标节点的名称传递给内核程序 13（步骤 902）。内核程序 13

根据所述文件名取得装置档案 36 中的设备主代码 N1，再根据设备主代码 N1 取得驱动程序 34，并将目标节点的名称交给驱动程序 34，使驱动程序 34 取得配置树 Tj 中所述目标节点内的配置参数值（步骤 904）。驱动程序 34 根据目标节点的名称取得配置树 Tj 中的所述目标节点（步骤 906），并将目标节点内的配置参数值回传给内核程序 13（步骤 908）。内核程序 13 再将目标节点内的配置参数值回传给请求读取的程序（步骤 910）。步骤 906 中，驱动程序 34 可以根据所述目标节点的路径以取得所述目标节点，或搜寻配置树 Tj 以取得。

[0037] 利用所述系统结构及方法，电子系统 200 内的应用程序 A1 ～ An 及内核程序 13 可以共享并同步配置参数。其它的实例如，电子系统 200 包括系统设定程序。系统设定程序允许的用户名称记录在图 6 中“sysUserName”节点的子节点“value”中，用户名称为“admin”。系统设定程序提供使用者接口让使用者设定图 6 中配置树的配置参数值。电子系统 200 中的每个程序记录其执行时期相关的事件于记录文件。电子系统 200 可以将所述记录文件发送至外部服务器供分析之用。电子系统 200 中的每个程序根据图 6 中 logLevel 节点的配置参数值以决定记录于所述记录文件的事件类别。换言之，logLevel 的配置参数值决定电子系统 200 中的多个程序记录事件的详细程度。“sysUserName”及“logLevel”节点的配置参数值可以利用所述方法被修改及读取。

[0038] 电子系统 200 更包括用户认证程序。系统设定程序提供使用者接口让使用者设定可以从远程访问电子系统 200 的用户的远程装置的网络通讯协议（Internet Protocol，简称 IP）地址，并记录于配置参数文件 12a 中的区块 B3，即 <TelnetAcl> 与 </TelnetAcl> 之间。区块 B3 中的配置参数转换成配置树 Tj 中的节点后，系统设定程序可以增加、删除或修改所述节点中的 IP 地址。用户认证程序利用配置树 Tj 的所述节点（未图标）中的 IP 地址以容许或拒绝用户使用的远程装置连接至电子系统 200。

[0039] 本发明提供的方法可以克服现有方法的不足。利用所述系统结构及方法，电子系统内的应用程序及内核程序可以共享并同步配置参数。所述方法可以广泛应用于 Linux/Unix 环境下或其它的操作系统环境下的程序开发。

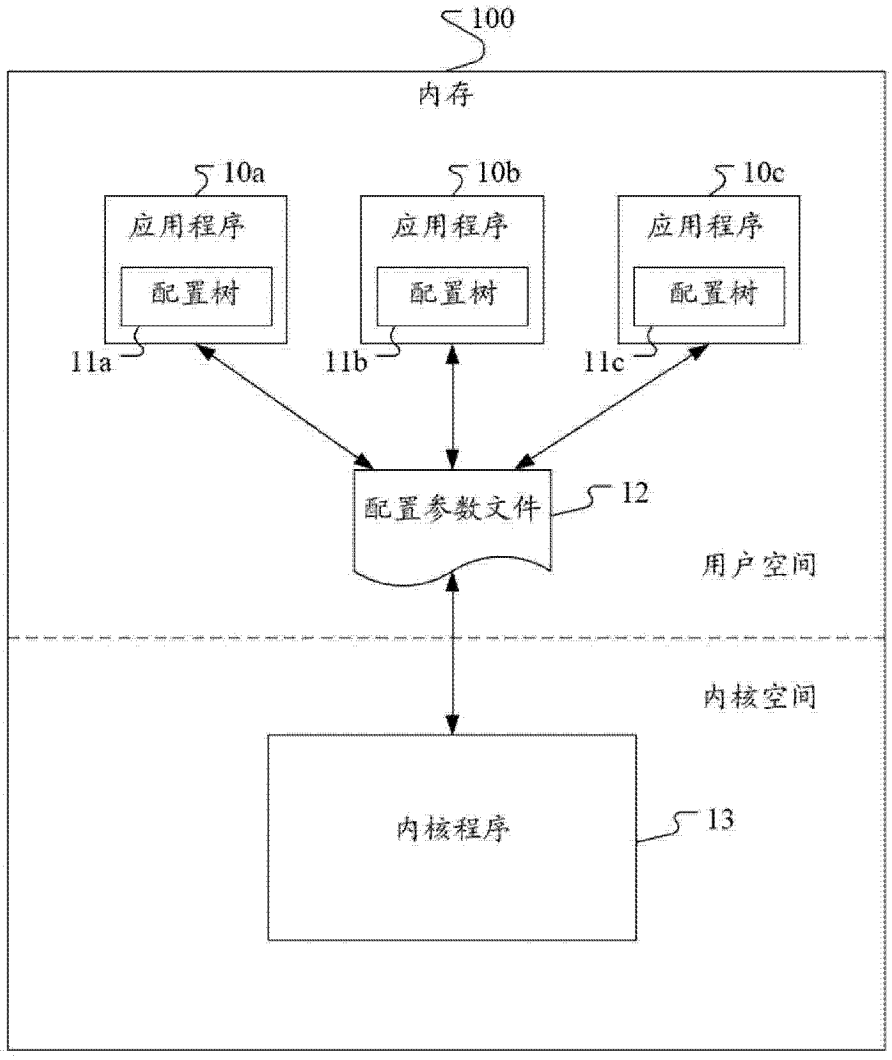

100
内存
10a
应用程序
配置树
11a
10b
应用程序
配置树
11b
10c
应用程序
配置树
11c
配置参数文件
12
用户空间
内核空间
内核程序
13

图 1

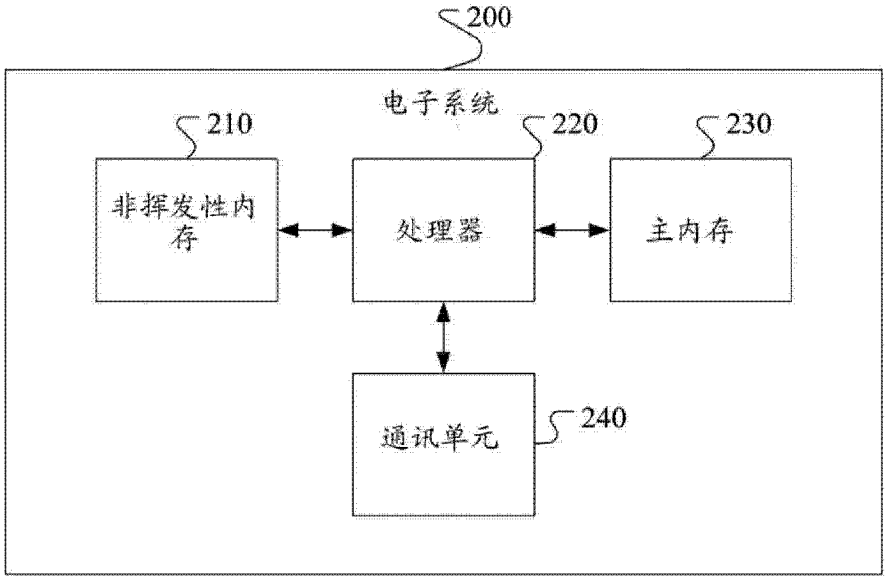


图 2

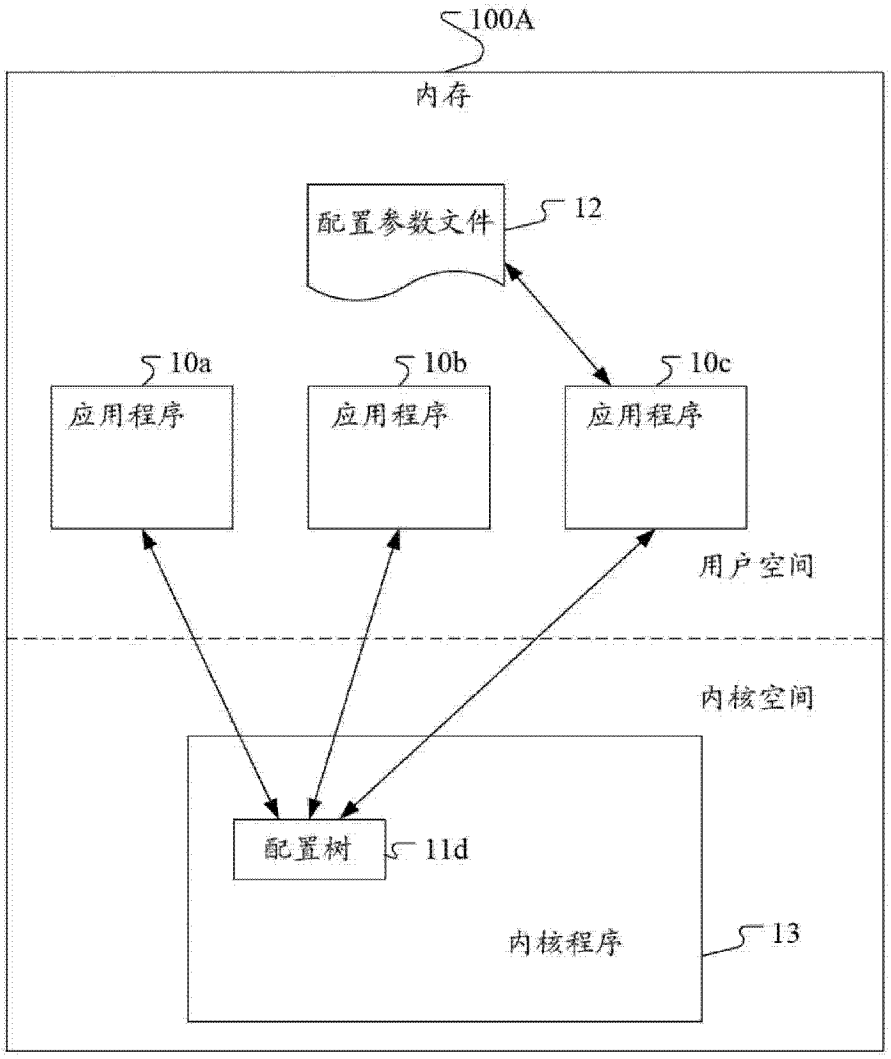

100A
内存
配置参数文件
12
10a
应用程序
10b
应用程序
10c
应用程序
用户空间
内核空间
配置树
11d
内核程序
13

图 3

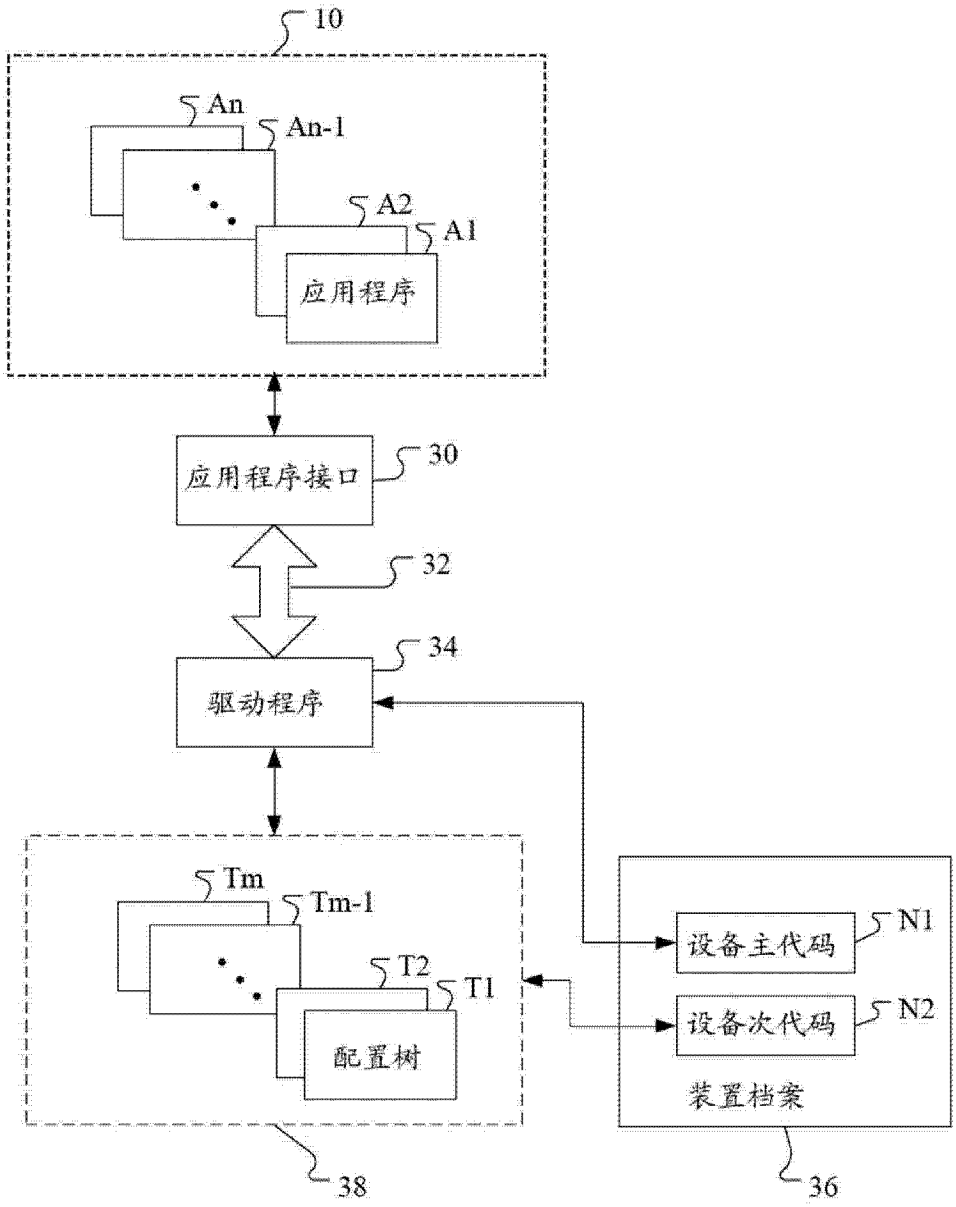

10
An
An-1
A2
A1
应用程序
应用程序接口
30
32
34
驱动程序
Tm
Tm-1
T2
T1
配置树
38
设备主代码
N1
设备次代码
N2
装置档案
36

图 4


12a
-<ConfigTree>
-<SystemInfo>
<protocol autoScan="enable" upnp="enable" igmpSnp="disable" igmpMode="disable" macFilterPolicy="forward" encodePassword="enable" enetwan="disable" />
<sysLog state="disable" displayLevel="ERR" logLevel="DEBUG" option="local" serverIP="0.0.0.0" serverPort="514" />
<sysUserName value="admin" />
</SystemInfo>
B1
-<AtmCfg>... </AtmCfg>
-<AtmCfgTd> ... </AtmCfgTd>
-<AtmCfgVcc>... </AtmCfgVcc>
-<SecCfg>
<srvCtrlList ftp="enable" http="lan" icmp="enable" ssh="enable" telnet="enable" tftp="enable" />
<dmzHost ipAddr="0.0.0.0" />
</SecCfg>
-<Lan>
<entry9999 address="1.1.1.1" mask="255.255.255.0" dhcpServer="disable" leasedTime="0" startAddr="0.0.0.0" endAddr="0.0.0.0" instanceId="1509949446" />
<entry1 address="192.168.1.1" mask="255.255.255.0" dhcpServer="enable" leasedTime="86400" startAddr="192.168.1.2" endAddr="192.168.1.254" instanceId="1509949441" />
</Lan>
B2
-<WirelessCfg> ... </WirelessCfg>
-<RouteCfg> ... </RouteCfg>
-<PMapCfg>... </PMapCfg>
-<SNTPCfg> ... </SNTPCfg>
-<Voice>... </Voice>
B3
-<TelnetAcl>
<telnetlist1 ipaddress="192.168.0.0" subnetmask="255.255.248.0" />
<telnetlist2 ipaddress="212.94.162.0" subnetmask="255.255.255.128" />
<telnetlist3 ipaddress="80.118.192.0" subnetmask="255.255.248.0" />
<telnetlist4 ipaddress="10.0.32.0" subnetmask="255.255.224.0" />
<telnetlist5 ipaddress="84.96.217.0" subnetmask="255.255.255.0" />
<telnetlist6 ipaddress="172.16.255.0" subnetmask="255.255.255.0" />
</TelnetAcl>
-<pppsrv_8_35>
<ppp_conId1 userName="" password="" serviceName="" idleTimeout="0" ipExt="disable" auth="auto" useStaticIpAddr="0" localIpAddr="0.0.0.0" Debug="disable" />
</pppsrv_8_35>
-<wan_8_35>
<entry1 vccId="1" conId="1" name="pppoe_8_35_1" protocol="PPPOE" encap="LLC" firewall="enable" nat="enable" igmp="enable" vlanId="-1" service="enable" instanceId="1509949442" />
</wan_8_35>
...
</ConfigTree>

图 5

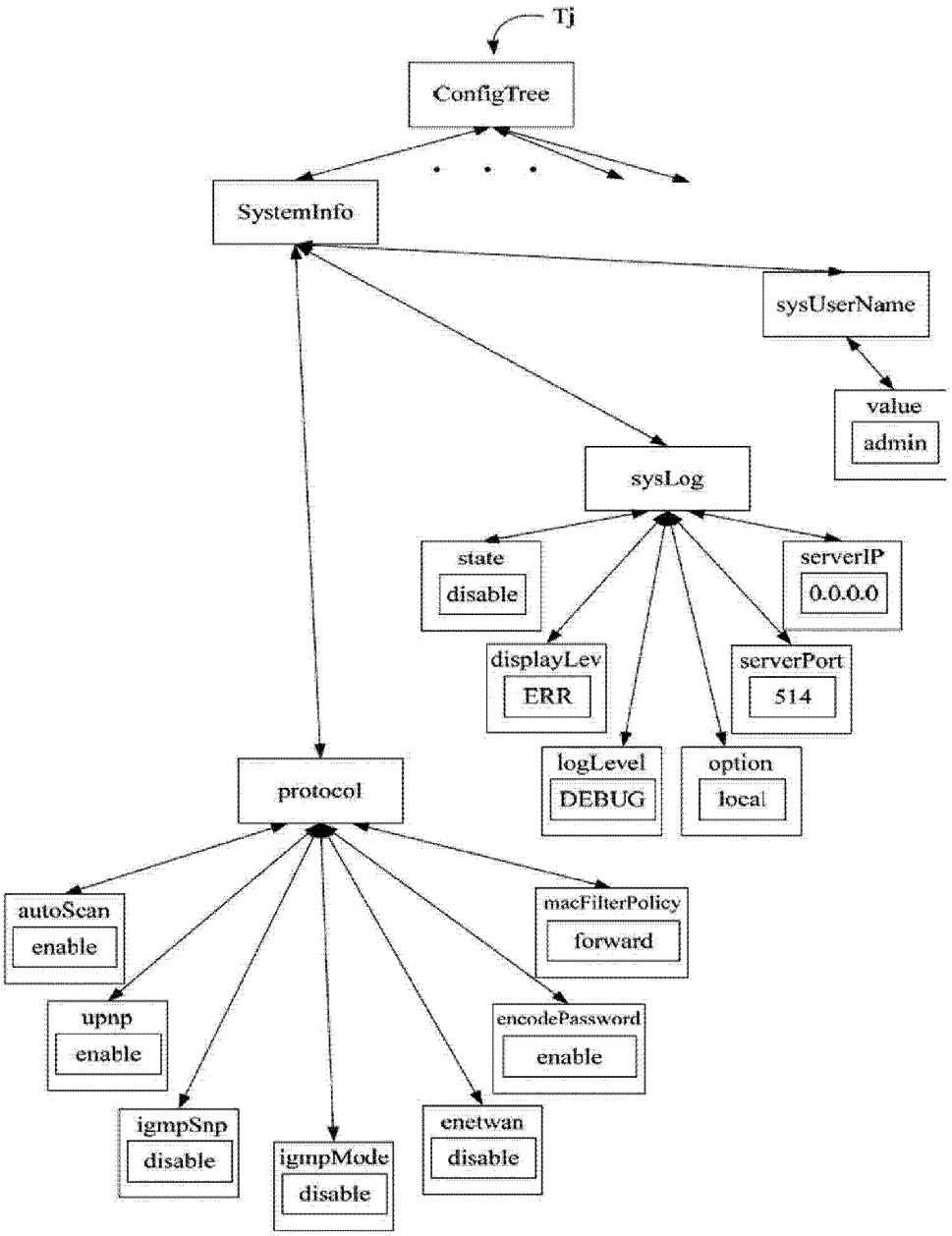

Tj
ConfigTree
SystemInfo
sysUserName
value
admin
sysLog
state
disable
serverIP
0.0.0.0
displayLev
ERR
serverPort
514
logLevel
DEBUG
option
local
protocol
autoScan
enable
macFilterPolicy
forward
upnp
enable
encodePassword
enable
igmpSnp
disable
igmpMode
disable
enetwan
disable

图 6

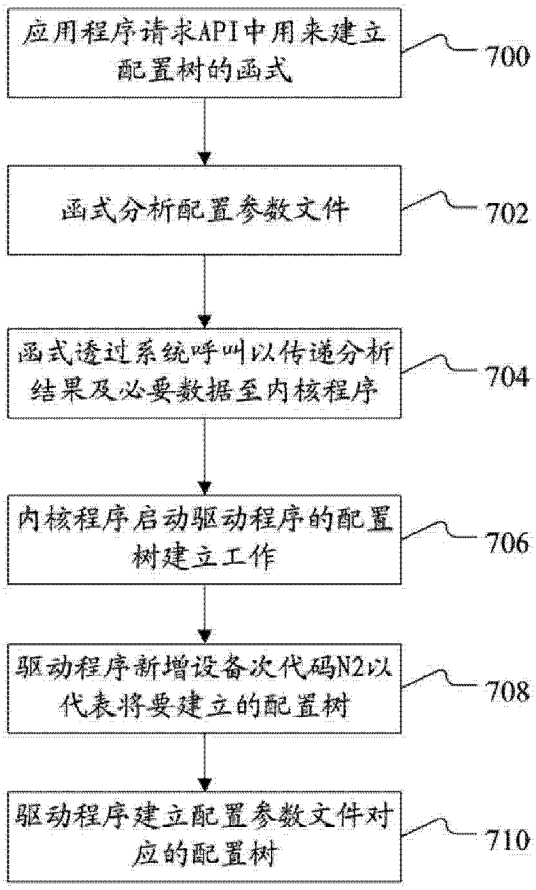

应用程序请求API中用来建立配置树的函式
700
函式分析配置参数文件
702
函式透过系统呼叫以传递分析结果及必要数据至内核程序
704
内核程序启动驱动程序的配置树建立工作
706
驱动程序新增设备次代码N2以代表将要建立的配置树
708
驱动程序建立配置参数文件对应的配置树
710

图 7

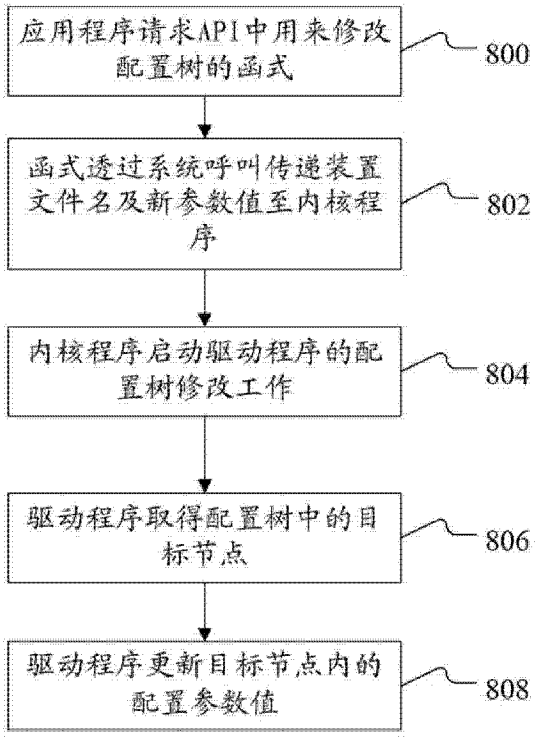

应用程序请求API中用来修改配置树的函式
800
函式透过系统呼叫传递装置文件名及新参数值至内核程序
802
内核程序启动驱动程序的配置树修改工作
804
驱动程序取得配置树中的目标节点
806
驱动程序更新目标节点内的配置参数值
808

图 8

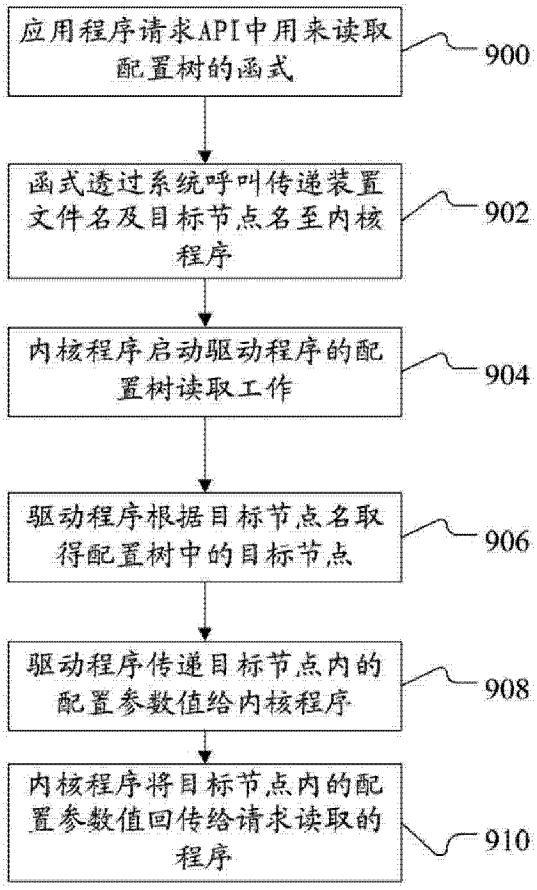

应用程序请求API中用来读取配置树的函式
900
函式透过系统呼叫传递装置文件名及目标节点名至内核程序
902
内核程序启动驱动程序的配置树读取工作
904
驱动程序根据目标节点名取得配置树中的目标节点
906
驱动程序传递目标节点内的配置参数值给内核程序
908
内核程序将目标节点内的配置参数值回传给请求读取的程序
910

图 9