(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 109451472 B
(45)授权公告日 2021.04.06

(21)申请号 201811168738.3

(22)申请日 2018.10.08

(65)同一申请的已公布的文献号
申请公布号 CN 109451472 A

(43)申请公布日 2019.03.08

(73)专利权人 四川长虹电器股份有限公司
地址 621000 四川省绵阳市高新区绵兴东路35号

(72)发明人 汪文羿

(74)专利代理机构 四川省成都市天策商标专利事务所 51213
代理人 刘兴亮

(51)Int.Cl.
H04W 4/80(2018.01)

(56)对比文件
CN 104598960 A,2015.05.06
US 2016234696 A1,2016.08.11
CN 105893038 A,2016.08.24

审查员 刘丹

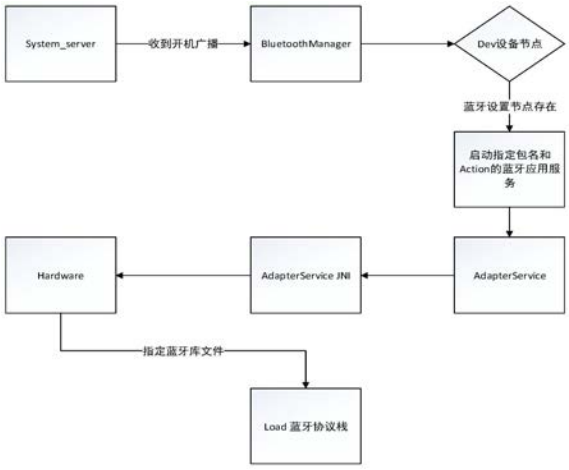
权利要求书1页 说明书3页 附图1页

(54)发明名称

Android智能设备上多蓝牙芯片的动态管理方法

(57)摘要

本发明公开了Android智能设备上多蓝牙芯片的动态管理方法,包括如下步骤:步骤一、蓝牙芯片型号的获取;步骤二、不同蓝牙模块的加载:首先预置蓝牙芯片对应的蓝牙应用和协议栈。其次,在系统核心进程启动完毕后,通过判断前面在init中设置的标志,指定跨进程启动对应蓝牙芯片的蓝牙应用服务。最后,通过JNI去获取硬件抽象层的ID,在获取硬件抽象层接口中,根据init中设置的标志,查找对应蓝牙协议栈的库文件,并把库文件的路径和名字传入Load方法,这样就完成了蓝牙芯片对应蓝牙协议栈的加载。本发明可以实现一套Android整机镜像版本兼容多个蓝牙芯片,提升了用户体验。



CN 109451472 B

1.Android智能设备上多蓝牙芯片的动态管理方法,其特征在于包括如下步骤:

步骤一、蓝牙芯片型号的获取

Android系统的内核启动完成以后,在insmod蓝牙设备驱动之前;首先通过Linux内核提供的标准方法,获取当前内核识别到的USB设备的Product ID和Vendor ID,然后在init进程中通过对PID和VID的判断,来加载当前内核识别到的蓝牙芯片的设备驱动文件;

程序加载完成之后,通过Android提供的系统属性服务,设置一个系统属性作为蓝牙芯片型号的标志,以便后续的程序可以简单的通过这个标志来获取当前接入的蓝牙芯片型号;

步骤二、不同蓝牙模块的加载

首先为了运行加载不同的蓝牙应用服务和蓝牙协议栈,需要预置蓝牙芯片对应的蓝牙应用和协议栈;由于系统只支持运行一个蓝牙进程,所以必须对不同的蓝牙应用和协议栈进行区分,蓝牙应用采用定义不同的匹配动作Action,让上层应用获取蓝牙服务主服务的时候通过定义的字符串启动对应蓝牙应用,其中涉及的其他服务的名字不变;蓝牙协议栈通过在Android编译文件中指定不同的名字,然后把指定的名字通过前面的系统属性服务存储在系统中,这样就可以在硬件抽象层加载协议栈的时候根据系统属性进行区别;

其次,在系统核心进程启动完毕后,通过判断前面在init中设置的标志,指定跨进程启动对应蓝牙芯片的蓝牙应用服务,这样上层的应用接口就和当前接入芯片的蓝牙应用服务实现了对应;

最后,在蓝牙应用服务初始化的过程中,会通过JNI去获取硬件抽象层的ID,需要在获取硬件抽象层接口中,根据init中设置的标志,查找系统Hardware目录下的对应蓝牙协议栈的库文件,找到对应的蓝牙库文件后,把库文件的路径和名字传入Load方法,这样就完成了蓝牙芯片对应蓝牙协议栈的加载。

2.根据权利要求1所述Android智能设备上多蓝牙芯片的动态管理方法,其特征在于:

在步骤一之前,需要预先将不同蓝牙芯片的驱动程序、蓝牙协议栈,以及蓝牙应用服务程序预置在系统中,具体包括:

为不同蓝牙芯片的蓝牙设备驱动程序和蓝牙协议栈定制不同的名字,然后再对驱动程序和蓝牙协议栈进行命名,具体是在Makefile中按照btusb_xxx.ko的格式命名驱动程序,在Android.mk中按照bluetooth,xxx.so的格式命名蓝牙协议栈,其中xxx代表厂商型号;

对蓝牙应用服务程序进行编译,编译出包名与Action完全不一样的应用,从而使得系统bind指定的蓝牙应用服务程序。

Android智能设备上多蓝牙芯片的动态管理方法

技术领域

[0001] 本发明涉及一种蓝牙芯片的管理方法,具体涉及一种Android智能电视上多蓝牙芯片的动态管理方法,属于Android智能设备技术领域。

背景技术

[0002] 当前Android智能电视基本标配了蓝牙功能,因为蓝牙在低功耗,数据传输方面有一定优势。但是当前支持Android智能电视的蓝牙芯片有不同的厂商,其采用的蓝牙应用服务,蓝牙协议栈以及蓝牙设备驱动都有芯片厂商自己的特殊定制,无法实现一套蓝牙代码兼容不同蓝牙厂商的芯片,导致Android系统编译生成的镜像文件只能与某一个特定的蓝牙芯片绑定。

[0003] 现有的连接方式存在如下问题:一是无法实现蓝牙芯片的更改,如用户希望使用其他的蓝牙接收器就无法实现。二是大大增加了Android整机系统镜像文件的维护工作量,如采用一个新的蓝牙芯片,就需要针对这个芯片重新维护一套Android整机系统镜像文件。

[0004] 本发明就是针对在Android智能电视上蓝牙芯片无法动态管理加载的问题,提出了一个有效的解决方法。

发明内容

[0005] 本发明的目的就是解决现有技术中Android智能电视无法动态兼容多个蓝牙芯片的问题,提供一种Android智能设备上多蓝牙芯片的动态管理方法。

[0006] 本发明主要是通过在Android智能电视上,动态识别接入的蓝牙芯片,然后根据蓝牙芯片的型号,在Android运行的不同阶段,选择合适的时机运行加载蓝牙芯片对应的蓝牙设备驱动、蓝牙协议栈以及蓝牙应用服务,从而实现对多蓝牙芯片的动态管理。

[0007] 在具体介绍本发明的技术方案之前,对本发明涉及到的相关技术内容进行说明如下:

[0008] Android智能电视的蓝牙启动机制:

[0009] Android系统的内核启动完成后,首先会创建init进程,在此进程中Android系统会通过 insmod的方式将指定蓝牙设备驱动程序载入内核,完成蓝牙芯片相关初始化工作。然后在 Android系统核心进程启动完毕后,在进程System_server中通过跨进程的通讯方式启动指定蓝牙应用服务。最后在蓝牙应用服务初始化的过程中,通过Android Hardware动态获取蓝牙的Module ID,把蓝牙协议栈加载起来并完成相关初始化工作,这样蓝牙功能就成功打开,可以提供相关的蓝牙服务。

[0010] 通过上面的描述,可以知道如果想要做到能够根据不同蓝牙芯片动态加载对应的蓝牙模块,关键就在于能够在加载运行蓝牙设备驱动、蓝牙协议栈以及蓝牙应用服务之前,能够通过合适的方法识别不同的蓝牙芯片型号,并且根据型号的不同,采取合适的方法加载不同的代码模块。

[0011] 本发明具体是这样实现的:

[0012] Android智能设备上多蓝牙芯片的动态管理方法,包括如下步骤:

[0013] 步骤一、蓝牙芯片型号的获取

[0014] Android系统的内核启动完成以后,在insmod蓝牙设备驱动之前。首先通过Linux内核提供的标准方法,获取当前内核识别到的USB设备的Product ID和Vendor ID,然后在init 进程中通过对PID和VID的判断,来加载当前内核识别到的蓝牙芯片的设备驱动文件。

[0015] 程序加载完成之后,通过Android提供的系统属性服务,设置一个系统属性作为蓝牙芯片型号的标志,以便后续的程序可以简单的通过这个标志来获取当前接入的蓝牙芯片型号。

[0016] 步骤二、不同蓝牙模块的加载

[0017] 首先为了运行加载不同的蓝牙应用服务和蓝牙协议栈,需要预置蓝牙芯片对应的蓝牙应用和协议栈。由于系统只支持运行一个蓝牙进程,所以必须对不同的蓝牙应用和协议栈进行区分,蓝牙应用采用定义不同的匹配动作Action,让上层应用获取蓝牙服务主服务的时候通过定义的字符串启动对应蓝牙应用,其中涉及的其他服务的名字不变。蓝牙协议栈通过在 Android编译文件中指定不同的名字,然后把指定的名字通过前面的系统属性服务存储在系统中,这样就可以在硬件抽象层加载协议栈的时候根据系统属性进行区别。

[0018] 其次,在系统核心进程启动完毕后,通过判断前面在init中设置的标志,指定跨进程启动对应蓝牙芯片的蓝牙应用服务,这样上层的应用接口就和当前接入芯片的蓝牙应用服务实现了对应。

[0019] 最后,在蓝牙应用服务初始化的过程中,会通过JNI去获取硬件抽象层的ID,需要在获取硬件抽象层接口中,根据init中设置的标志,查找系统Hardware目录下的对应蓝牙协议栈的库文件,找到对应的蓝牙库文件后,把库文件的路径和名字传入Load方法,这样就完成了蓝牙芯片对应蓝牙协议栈的加载。

[0020] 通过以上的步骤,就可以动态根据接入的蓝牙芯片不同,从而使能对应蓝牙功能。

[0021] 在步骤一之前,需要预先将不同蓝牙芯片的驱动程序、蓝牙协议栈,以及蓝牙应用服务程序预置在系统中,具体包括:

[0022] 为不同蓝牙芯片的蓝牙设备驱动程序和蓝牙协议栈定制不同的名字,然后在对驱动程序和蓝牙协议栈进行命名,比如在Makefile中按照btusb_xxx.ko(xxx代表厂商型号)的格式命名驱动程序,在Android.mk中按照bluetooth,xxx.so的格式命名蓝牙协议栈。

[0023] 对蓝牙应用服务程序进行编译,编译出包名与Action完全不一样的应用,从而使得系统 bind指定的蓝牙应用服务程序。

[0024] 本发明具有如下有益效果:

[0025] 解决了当前Android智能电视无法根据接入蓝牙芯片动态加载使能的问题。采用此方案可以实现一套Android整机镜像版本兼容多个蓝牙芯片,大大减少软件移植的工作量。也可以让用户在蓝牙接收器上有了选择,提升了用户体验。同样,本申请还适用于除了Android 智能电视之外的其他Android智能设备上。

附图说明

[0026] 图1为系统获取蓝牙芯片型号的流程

[0027] 图2为系统动态加载蓝牙各个模块实施流程

具体实施方式

[0028] 下面结合附图和具体实施例对本发明作进一步的说明。

[0029] 本实施实例采用以上的发明方案,应用于Android 6.0智能电视机芯平台具体实施方式是:

[0030] 1、定制不同芯片厂家的蓝牙设备驱动程序、蓝牙协议栈的名字,在Makefile中按照 btusb_xxx.ko(xxx代表厂商型号)的格式命名驱动程序,在Android.mk中按照bluetooth,xxx.so 的格式命名蓝牙协议栈。

[0031] 2、由于Android Framework层是通过bindservice启动蓝牙应用服务,而蓝牙应用服务是一个apk文件,蓝牙应用的编译需要依赖Android SDK,然后通过Android标准Makefile文件进行编译。由于系统服务的启动可以通过包名来区别,所以为了让Android的系统启动服务程序可以获取到不同的蓝牙应用服务,所以我们需要编译出两个或多个包名和Action完全不一样的应用,这样才能让系统bind指定的蓝牙应用服务程序。

[0032] 3、Android系统启动init进程之后,在init脚本文件中通过lsusb的方式向系统查询当前所有接入的USB设备,然后在内核返回的查询结果中通过PID和VID进行筛选,找到当前实际接入的蓝牙芯片,然后根据设备驱动程序的名字insmod对应的驱动。最后通过Android系统属性的方式设置一个蓝牙系统属性来表示当前接入设备的型号,具体如附图1所示。

[0033] 4、在系统核心进程System_server中启动BluetoothManagerService服务并完成初始化工作,当收到系统开机完毕的广播后,首先通过判断dev设备节点来确定蓝牙设备驱动程序是否加载正确,然后通过系统属性获取当前蓝牙芯片型号,最后向bindservice方法传入指定的包名和Action,这样就通过跨进程的方式启动了蓝牙芯片对应的蓝牙应用服务。

[0034] 5、上层以bindservice的方式启动蓝牙应用服务,蓝牙应用服务初始化的过程中会通过 JNI的方式调用hw_get_module接口来获取硬件抽象层的ID,在硬件抽象层中我们根据init 设置的系统属性来选择对应的device_id,最后把device_id传入load蓝牙协议栈的方法,系统会根据device_id加载对应的蓝牙库文件。

[0035] 尽管这里参照本发明的解释性实施例对本发明进行了描述,上述实施例仅为本发明较佳的实施方式,本发明的实施方式并不受上述实施例的限制,应该理解,本领域技术人员可以设计出很多其他的修改和实施方式,这些修改和实施方式将落在本申请公开的原则范围和精神之内。

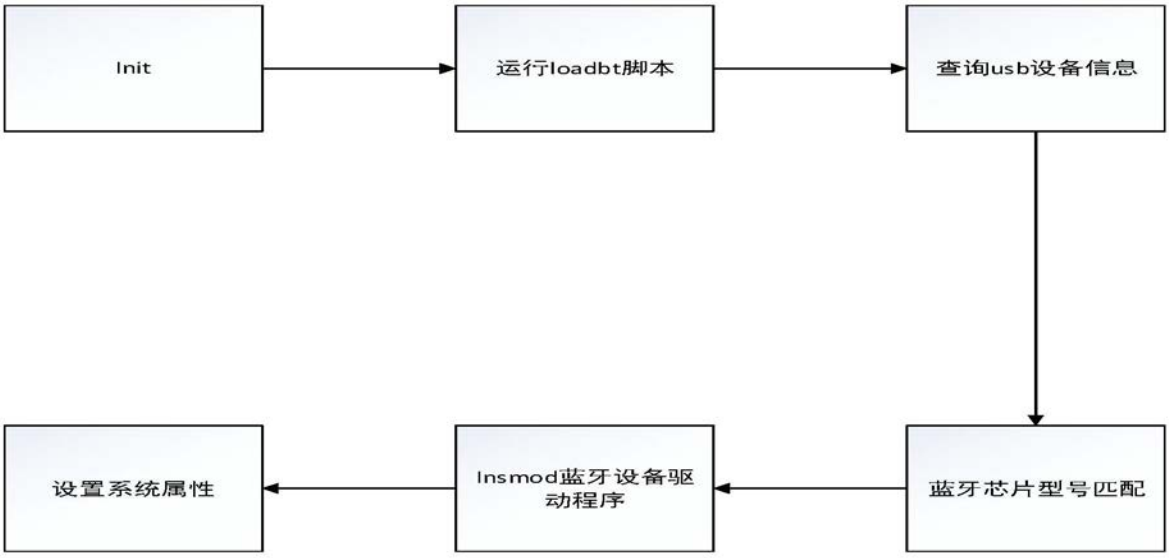

Init
运行loadbt脚本
查询usb设备信息
蓝牙芯片型号匹配
Insmod蓝牙设备驱动程序
设置系统属性

图1

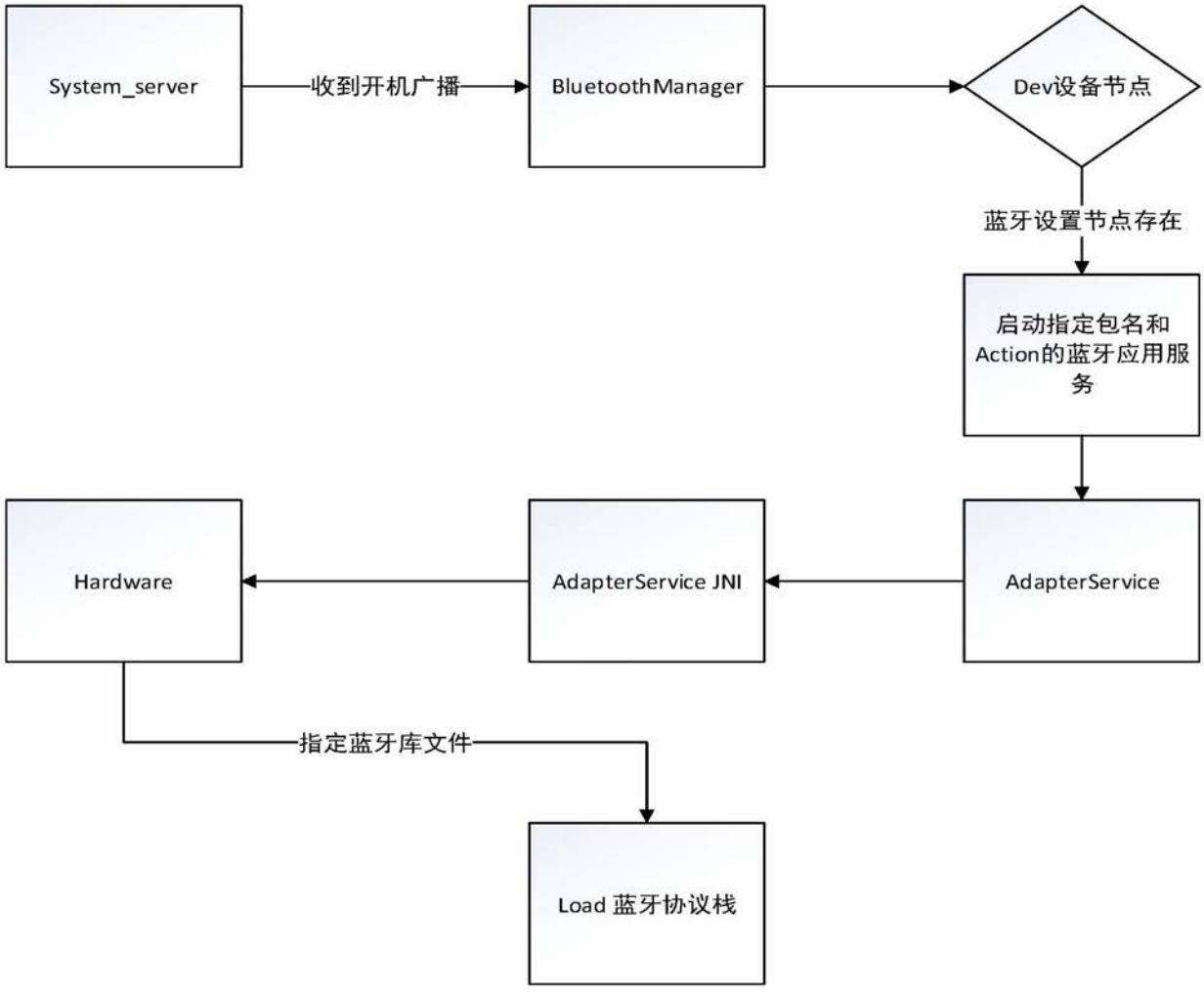

System_server
收到开机广播
BluetoothManager
Dev设备节点
蓝牙设置节点存在
启动指定包名和Action的蓝牙应用服务
AdapterService
AdapterService JNI
Hardware
指定蓝牙库文件
Load 蓝牙协议栈

图2