基于代理的设备虚拟化技术及其应用

杨 霞，刘维飞，郭文生，廖士钞，孙海泳

(电子科技大学信息与软件工程学院 成都 610054)

【摘要】随着智能手机逐步成为人们工作和生活不可或缺的随身设备，其安全性和用户隐私问题也越来越突出。为解决该问题，提出一种基于Linux容器技术的方法，在同一台手机设备上建立多个虚拟手机系统，使多个相互隔离的Android系统同时运行。为了使多个子系统共享同一台设备资源，提出了基于代理的设备虚拟化技术，并以手机的Radio设备为例实现了该技术。然后，在一个实现了的原型系统上的测试结果表明每个虚拟手机系统均可正常使用所有物理设备，说明该方法可有效地实现设备的虚拟化。最后，通过对多系统的数据隔离、系统资源开销的测试验证了该方法的可行性。由于各虚拟手机系统之间相互隔离、互不影响，因而很好地保护了用户数据的隐私。此外，即使某一个Android子系统出现运行故障或者被恶意攻击，其他子系统照样可以正常运行，从而提高了整个系统的安全性。同时，各子系统的应用和功能可以根据不同的应用场景和需求特别定制，可满足用户的个性化需求。

关 键 词 Android操作系统； 设备虚拟化技术； Linux容器； 智能手机

中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2017.06.016

Devices Virtualization Technology Based on the Proxy

YANG Xia, LIU Wei-fei, GUO Wen-sheng, LIAO Shi-chao, and SUN Hai-yong

(School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract As the mobile phone plays a more important role in our life, the problem about security and privacy of smart phone become more prominent. To solve this problem, this paper presents an approach to implement more virtual phone (VP), which is based on Linux container technology, running two or more Android systems in a single smart phone device. In order to allow each VP uses devices concurrently, we present a virtualization technology based on device proxy, and realize it on radio device. A prototype system is implemented on Nubia Z7max smartphone and the functionality of virtualization, data isolation etc. are tested. The experiment results show that our approach is useful and feasible. Each VP can simultaneously shares devices, user's data about devices and applications are isolated between isolated Android systems, and the system cost is kept in allowable range. Due to the VPs are isolated each other, there are three benefits: this approach can protect the user's privacy effectively, even one VP is corrupted the whole system can still work, and it also can satisfy user's personalized demand because the VP's functions can be customized by application scenarios and user's requirements.

Key words Android OS; device virtualization technology; Linux container; smartphones

随着Android系统在智能手机、平板等嵌入式设备上的广泛使用，Android系统所面临的安全问题和隐私问题也日益严峻。采用虚拟化技术在同一硬件平台上运行多个操作系统，将不同安全等级应用隔离运行，可以提高Android系统的安全性并保护用户隐私。目前，Android系统主要采用加密方法对高安全性应用或者私密数据进行保护，并且高安全性或高秘密程度的应用与普通应用程序运行于相同的Android系统中，因此无法满足用户对安全性和私密性的需求。

由于Android操作系统基于Linux内核，这使容器技术(Linux container, LXC)可以较为方便地部署在Android系统上实现内核级的隔离。然而，容器技术对Android平台的设备虚拟化支持尚不完善，不便于多个容器对同一个硬件设备的复用。为了解决该问题，本文采用LXC技术实现了多个虚拟手机系统在同一个平台上的隔离运行，同时提出一种基于代理的设备虚拟化技术，有效地解决了多Android平台上某些物理设备不能复用的问题，由此，每个虚拟手机系统以为自己独占系统资源。每个虚拟手机的

收稿日期：2016－06－14；修回日期： 2016－09－12

基金项目：中央高校基金(A03017023701169, ZYGX2015J066)

作者简介：杨霞(1978－)，女，博士，副教授，主要从事嵌入式虚拟化技术、嵌入式安全系统方面的研究.

功能可以根据用户需要而特殊定制，如可以建立工作和生活两个虚拟手机，将工作和生活的应用分开执行，保护用户隐私。

1 相关研究

目前，为使多个Android系统在同一设备上同时运行，常采用的方法主要有：传统的虚拟化技术和容器技术。在设备虚拟化方面，根据采用的虚拟化方法的不同，所采用的设备虚拟化技术也各有差异。

传统虚拟化技术如XEN和KVM等，通过Hypervisor，使用半虚拟化或者全虚拟化技术为客户机虚拟出多个独立的运行平台，因此隔离性较好。传统虚拟化方法常采用的设备虚拟化技术大致分为两类：半虚拟化驱动，如Xen实现的前后端半虚拟化驱动模型[1]和KVM的Virtio[2]；设备模拟，在支持全虚拟化的平台上，Xen与KVM均支持利用QEMU为虚拟机模拟设备[3-5]。但是设备模拟方法需要CPU支持虚拟化功能，并且在进行设备I/O操作时，处理器所处的特权等级会发生多次切换，开销较大，导致I/O效率较低。随着嵌入式处理器对虚拟化功能的支持，XEN、KVM均面向ARM处理器推出了解决方案[6-8]，使多个操作系统运行于嵌入式平台成为可能。从隔离性来看，传统虚拟化隔离度最高、最安全，但是从资源利用率和I/O效率来看，传统虚拟化技术资源利用率较低，难以运用于嵌入式设备。

容器[9]作为一种轻量级的虚拟化技术，除提供资源隔离外，最大的优点是客户机共享同一个内核提供的资源，虚拟化开销小，资源利用率较高。基于容器的多Android系统分为应用级和系统级容器。文献[10]提出一款基于Android框架层的应用级安全容器Knox，客户机运行于Android宿主机之上。宿主Android的存在，使客户机能直接调用宿主机的设备服务，满足自身对设备操作的需求，在设备虚拟化方面几乎没有开销。但由于客户机的安全性和稳定性严重依赖于宿主Android系统的安全性和稳定性，因此即使采取了很多加密手段，Knox的可靠性依然不高。

LBE公司最近推出了平行空间技术[11]，基于Android的多用户环境实现了虚拟应用系统，可以在一个手机上安装多个应用程序，如微信。该方法在用户空间对应用程序的数据进行隔离，实现更轻量级，但安全性、隔离性比上述各方案更差。

文献[12]提出了系统级容器技术，在Android系统的Linux内核支持下实现多个容器，每个容器中运行一个Android系统。由于多个容器之间仅共享Linux内核，因此系统级容器的安全性取决于Linux内核，在借助于现有的Linux安全机制的基础上，系统级容器的安全性将会优于Knox。另外，由于客户机之间仅仅共享内核，隔离性也较Knox好，但比传统虚拟化技术安全性略差。

本文通过系统级容器的方法实现在同一个硬件平台上的运行多个虚拟手机(VP)。为了使多个虚拟手机独立使用手机的硬件设备，需要重点解决设备复用问题，虽然文献[12]已经实现了在同一个手机上同时运行多个Android系统，但是在设备复用方面还需要进一步的研究。本文基于此工作，提出了一套设备复用机制，使多个虚拟手机复用同一套硬件设备资源，每个虚拟手机中的Android子系统都自以为独占这些设备和资源。

2 基于容器技术的多虚拟手机框架

文献[12]已经提出了采用容器技术的多Android框架，基于Linux内核所提供的命名空间Namespace和Cgroups机制，实现了多个Android在同一个Linux内核上同时运行的方案，本文基于此框架进一步研究了设备虚拟化技术，以实现多个虚拟手机系统在同一个硬件平台的运行。基于容器的多虚拟手机系统框架如图1所示。

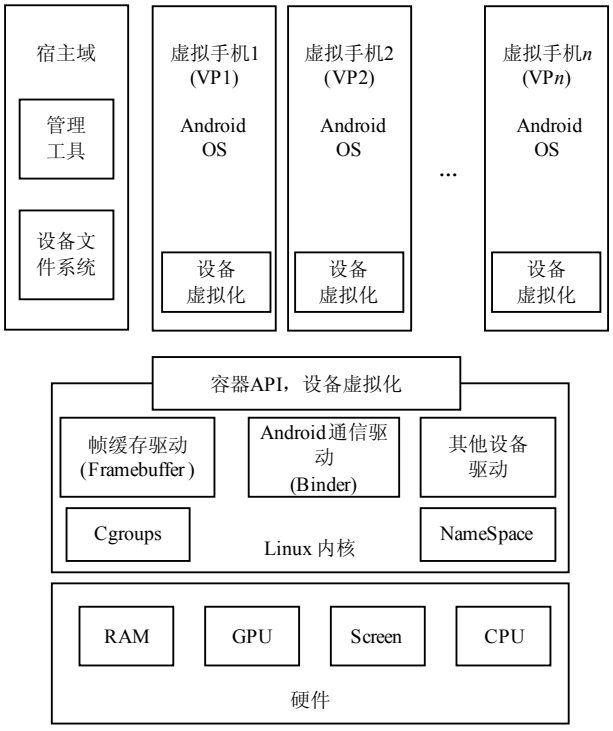

图1 基于容器的多虚拟手机系统框架

该框架分为3层，自底向上分别为硬件层、内核

层、Android框架层。在内核层提供3个重要的功能：

1) 通过利用Linux内核的命名空间和Cgroups机制，在Linux内核之上创建并运行多个相互隔离、互不影响的运行空间，即容器。并在每个容器中运行一个独立的不包含Linux内核的Android系统。2) 内核层的容器API为容器管理功能提供接口，容器管理模块中的管理工具通过调用相应的API来对容器进行管理。如容器的创建和销毁、启动、停止以及容器的修改等功能。3) 硬件设备的虚拟化，由于Linux内核中的命名空间机制并不能完全支持Android系统一些设备，通过在内核中添加一个新的Namespace机制来实现对这些内核设备的虚拟化，如Framebuffer(帧缓存驱动)、Binder(Android通信驱动)等。此外，本文通过在容器内部添加设备虚拟化层来实现对闭源设备的复用，如Radio设备。

3　采用代理的设备虚拟化技术设计

3.1　Android系统的设备驱动模型介绍

Android系统架构如图2所示，从上到下，分别是Android的框架层、系统服务层、硬件抽象层(hardware abstraction layer, HAL)、以及Linux内核。Android框架层向应用开发人员提供应用相关服务和编程接口；系统服务层则向上层提供底层硬件设备相关服务，如多媒体服务、搜索服务、提醒服务等；硬件抽象层则为Android提供了上层服务操作设备的标准接口；Linux内核作为Android平台的基石，提供任务调度、设备资源管理等服务。

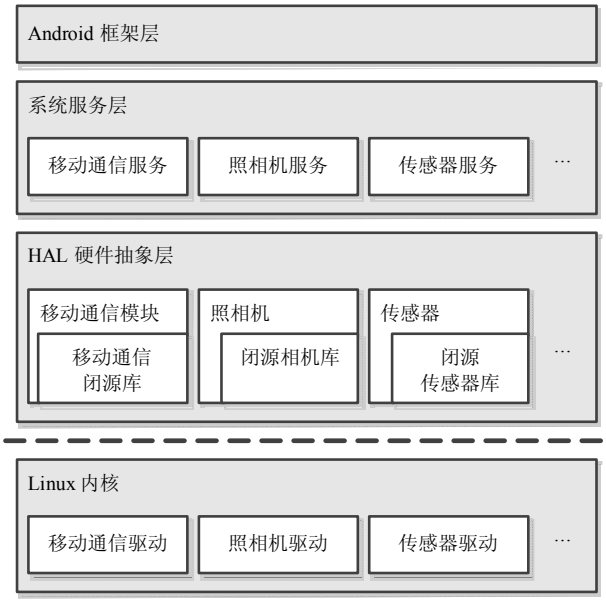

图2　Android系统框架

硬件抽象层是连接Android系统上层设备操作与Linux内核驱动程序的桥梁[13]，为Android平台定义了一个标准的硬件操作接口，该接口由硬件供应商来实现，以闭源动态库的形式存在，源码不公开。Android在启动时会加载并初始化动态库，以便于应用程序通过调用动态库中的方法来操作硬件设备。

3.2　设备复用问题分析

在如图1所示的系统框架中，多个虚拟手机使用同一个Linux内核，共享内核资源。通过对内核驱动进行虚拟化扩展可以在内核驱动层为每个虚拟手机系统虚拟出一套设备，并且可以避免对虚拟手机中的Android系统进行过多的修改。然而，对于部分将设备功能实现抽取并以闭源动态库的形式放置于HAL层的设备，上述的虚拟化技术便不再有效，如电话设备，本文将此类设备称为闭源设备。

由于HAL层接口的实现由设备供应商提供，并且以闭源动态库的形式发布，因此无法修改其源代码以实现设备的虚拟化功能扩展。此外，这些动态库中可能还保存了设备运行的状态信息，如果不做处理，多个虚拟手机中的设备信息将不一致，导致设备运行异常甚至系统崩溃。由于以上原因，只有在HAL接口或者以上层次做设备虚拟化扩展，才能有效地绕过设备供应商的闭源动态库，使多个虚拟手机共享这些硬件设备。文献[12]提出了基于容器的虚拟化技术，实现了开源设备的虚拟化，但无法解决电话、Wifi、传感器等闭源设备模块的复用问题。本文通过在HAL层添加设备代理，实现了闭源设备的虚拟化扩展。

3.3　采用设备代理的框架设计

基于上述分析，本文提出了一种设备代理的方法，在HAL层实现闭源硬件设备的虚拟化技术，解决了对闭源设备无法虚拟化的难题。在Android HAL层中，每一类硬件设备均有与之对应的HAL子系统，Android上层调用相应的设备子系统接口来操作设备。本文提出的设备代理技术适用于手机平台的绝大多数闭源设备，如音频、传感器等。本文将以Radio(移动通信设备，包含通话和移动网络)为例，对设备代理技术进行描述，Radio设备代理框架如图3所示。

基于代理的设备虚拟化框架主要包含两大部分：虚拟设备客户端和虚拟设备服务端。位于每个VP子系统中的虚拟设备客户端主要作用是获取应用程序发给设备的操作请求，然后将其转发到虚拟设备服务端。虚拟设备服务端位于宿主域，负责为客户端提供设备虚拟化的支持，包含设备操作请求过滤模块、设备事件响应过滤模块，它们和设备供

应商提供的闭源动态库协同工作，共同为虚拟设备客户端的设备操作请求服务。

基于代理的虚拟化设备工作流程如下：

1) VP系统中，应用程序通过系统编程接口发出对设备操作的服务请求；2) HAL子系统的虚拟设备客户端获取该请求，然后将其转发给位于宿主域的虚拟设备服务端；3) 虚拟设备服务端接受此请求后，结合当前各VP子系统的虚拟设备状态，将操作请求分类，并根据状态分别使用设立的设备虚拟操作接口进行处理。

以Radio设备为例，通过上述设备代理流程，可以使多个VP通过各自的SIM卡同时使用Radio设备。

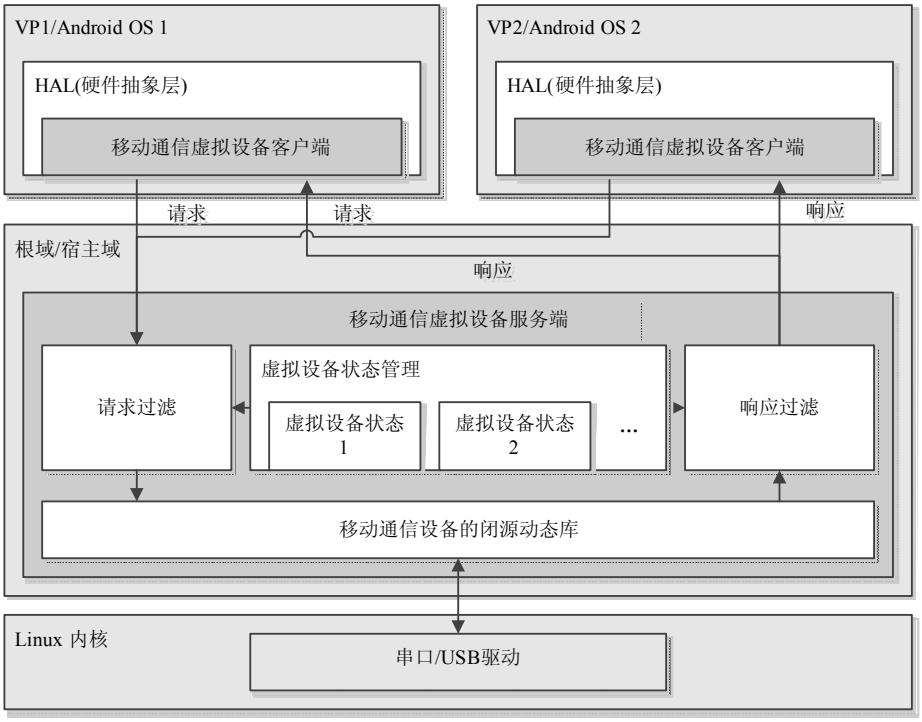

图3 设备代理的系统框架设计

4 设备虚拟化技术的实现

本文以Radio设备为例，详细阐述基于代理的设备虚拟化技术的实现过程。按实现位置的不同，Radio设备的HAL接口可分为供应商动态库函数接口和Android Radio服务RILD的函数接口。虚拟设备客户端和服务端分别实现这两部分接口。下面分别对客户端和服务端的的实现进行详细地阐述。

4.1 Radio虚拟设备客户端的实现

Radio虚拟设备客户端的作用与供应商的Radio动态库类似，主要作用是获取应用程序对设备的操作请求并进行处理。为此，在客户端需要实现虚拟设备客户端的动态库函数接口和设备操作代理函数。Radio虚拟设备客户端必须提供与设备供应商提供的Radio闭源动态库libril-vendor一致的动态库接口供Android调用，为此需要实现的接口函数包括模块初始化函数RIL_Init和操作Radio设备的虚拟函数接口，如onRequest、onStateRequest等。获取应用程序的设备操作并转发到虚拟设备服务端的代理工作是虚拟设备客户端的最主要功能。本文通过修改HAL接口函数的实现，将设备操作转发到虚拟设备服务端。虚拟设备客户端和虚拟设备服务端必须保证收发的设备操作和参数在两端均是正确有效的。

4.2 Radio虚拟设备服务端的实现

为了向虚拟设备客户端提供代理服务，虚拟设备服务端自身必须能够执行对设备的操作。因此，虚拟设备服务端必须实现RILD中的Radio HAL接口，包括操作闭源动态库的函数接口，如初始化闭源动态库，以及与动态库通信的回调函数接口，下文称作虚拟回调函数接口。

此外，虚拟设备服务端还包括以下3个模块：虚拟设备状态组、请求过滤模块、响应过滤模块。

4.2.1 虚拟回调函数接口的实现

虚拟回调函数接口用于接收动态库返回的设备操作结果以及设备递交的异步事件。Radio虚拟设备服务端必须实现虚拟回调函数接口，才能接收来自设备动态库的设备操作结果或者异步事件消息，需要实现的回调函数包括结构体RIL_Env中的所有函数指针。

4.2.2　虚拟设备状态管理模块的实现

虚拟设备服务端为每一个虚拟设备客户端维护一个虚拟设备状态结构体，对虚拟设备状态跟踪，这些状态与真实硬件设备运行过程中所具有状态信息相对应，以Radio设备为例，虚拟设备状态结构体包含的信息如下：

```
typedef struct {
    int radioState;
    int screenOnOff;
    int dataOnOff;
    //...
} RadioStateInfo;
```

其中，radioState对应Radio的开、关状态(即虚拟设备客户端状态)，screenOnOff对应手机屏幕的开、关状态，dataOnOff对应数据连接的开、关状态等。

4.2.3　请求过滤模块的实现

请求过滤模块用于对来自不同虚拟设备客户端的设备操作请求进行过滤分流。根据各个虚拟设备客户端所处状态的不同，某个虚拟设备客户端会被判定为可以操作硬件设备，或者只能操作虚拟设备。对于后者，请求过滤模块会为其进行设备操作模拟，并返回正确的设备操作响应。

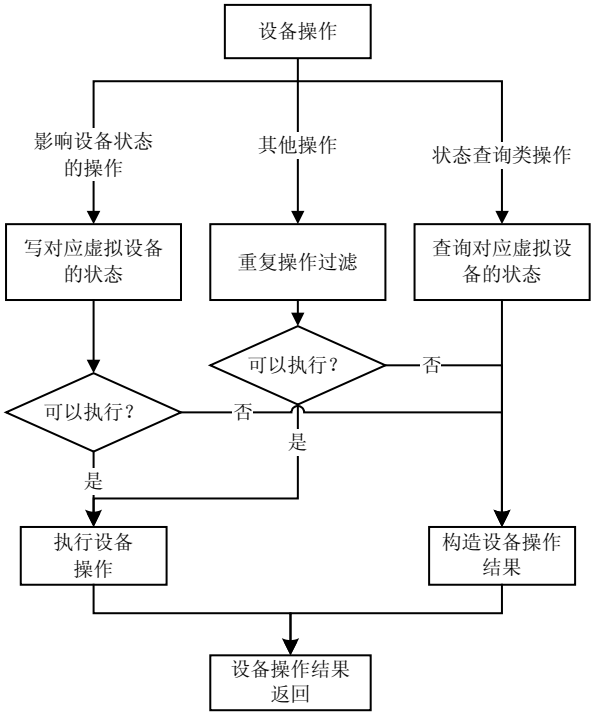

图4　设备操作请求过滤过程

本文进一步按照设备操作是否影响设备运行状态，以及是否是对设备运行状态进行查询，将设备操作划分为3类：影响设备运行状态的设备操作、设备运行状态查询类设备操作、以及其他设备操作，其操作过程如图4所示。

1) 对于影响设备运行状态的设备操作，如Radio设备的打开、关闭操作，首先更新虚拟设备的状态；其次，根据相关虚拟设备状态以及全局设备状态，决定该设备操作是否需要被执行。如对于Radio设备，设备的关闭操作必须在所有虚拟设备均处于关闭状态时，才能被执行，否则只更新虚拟设备的状态，并返回设备操作结果。2) 对于查询类的设备操作，如查询Radio运行状态的操作，虚拟设备服务端读取对应虚拟设备的状态，构造并返回设备操作结果。3) 对于其他设备操作，虚拟设备服务端先进行重复操作过滤处理，对于剩下的设备操作进行代理执行。

4.2.4　响应过滤模块的实现

本文将异步设备操作结果和设备递交的异步事件均称为动态库的响应。响应过滤模块用于对来自动态库的响应进行处理，包括对设备操作结果和对设备递交的异步事件进行处理，此外，响应过滤模块也负责更新虚拟设备状态和全局设备状态。

对于异步设备操作结果的处理主要是根据操作类型，决定是否需要更新虚拟设备状态和全局设备状态。对全局设备状态的更新保证了虚拟设备服务端和物理设备状态的一致性；对虚拟设备状态的更新则保证了虚拟设备服务端能正确地记录每个虚拟设备客户端的状态。对异步事件的过滤主要是为虚拟设备过滤状态冲突的异步事件，使虚拟设备之间不会互相干扰。如，报告Radio设备处于开启状态的异步事件通知不能发送给那些虚拟Radio设备处于关闭状态的虚拟设备客户端。

5　实　验

为了验证本文提出的设备虚拟化技术的可行性，选择了中兴Z7max手机作为实验硬件平台，该手机CPU为高通骁龙801，RAM大小2 GB。Android版本5.1，Linux内核版本3.4。SIM卡槽1插入移动公司的电话卡，卡槽2插入联通公司的电话卡。

在实验手机平台上创建两个容器，分别运行一个VP子系统。位于根域的容器管理服务加载虚拟设备服务端，启动两个客户Android子系统，分别称为VP1和VP2，它们各自加载虚拟设备客户端。本文的基于代理的设备虚拟化技术适合于手机平台上所有驱动程序闭源的设备。此处仅以Radio设备为例对该技术的可行性和有效性进行验证。

5.1 设备虚拟化功能的可行性验证

由于中兴Z7max手机拥有两张SIM卡，因此可以在VP1和VP2启动前，将两张SIM卡分别分配给不同VP(也可以全都分配到同一个VP)。在VP1和VP2均启动完毕后，两个容器的Android系统分别显示SIM卡1和SIM卡2使能，信号强度显示正常，两个容器均可正常拨打/接听电话、发送/收取短信，如图6所示。试验结果表明基于代理的设备虚拟化技术可以有效的实现两个VP对电话设备的复用。

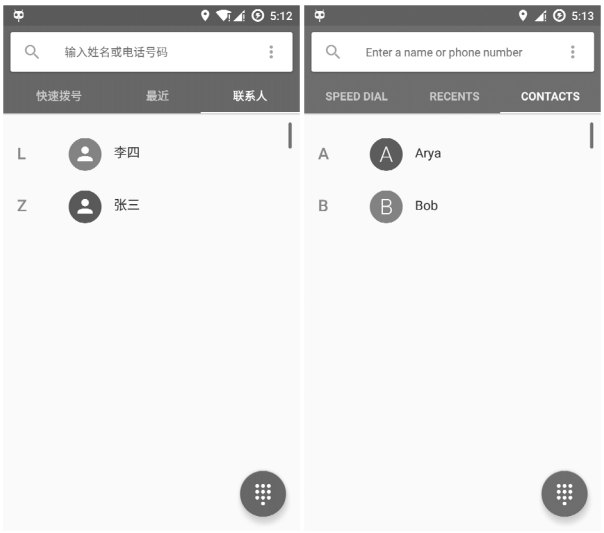

图6 VP1和VP2中的通讯录

5.2 数据隔离性验证

由于在所用实验平台上拥有两张SIM卡，每个VP中通信录、通话记录和信息分开存储，相互不可见，由此保护了用户通话信息的隐私。图6中的VP1和VP2分别使用SIM卡1和SIM卡2时的通讯录是完全隔离的，包括本地和SIM卡上的；图7展示了VP1和VP2之间的通话记录是完全隔开的。

图7 VP1和VP2中的通话记录

5.3 系统资源开销

为了测试性能开销的最大值，实验平台上每个VP中运行完整的相同Android系统镜像，因此，ROM开销增幅较小，测试结果显示仅增加约30%；但是RAM开销较大，大约增加90%。在实际应用环境中，两个VP中的系统可以根据应用需求而特殊定制，从而将会大量的减少内存开销。由于切换到后台的Android系统会处于睡眠状态，因此CPU和电源的开销增幅均不大，分别约为2%和6%。

6 结 束 语

为了提高Android系统的安全性、可定制性，既满足用户的多元化需求，同时还可以保护用户隐私。本文采用Linux的容器技术建立相互独立的执行环境，在同一个手机平台上建立多个虚拟的手机子系统，每个子系统中单独运行Android操作系统，并共享物理设备资源。

然而，容器技术对Android平台上的设备虚拟化支持尚不完善，不便于多个容器内的系统对同一个硬件设备的复用。为了解决该问题，本文提出一种基于代理的设备虚拟化技术，使多个虚拟手机复用同一套硬件设备资源，每个虚拟手机中的Android子系统都自以为独占这些设备和资源。并以手机的Radio设备为例详细阐述了本文技术的设计和实现细节，该方法可以很好地解决闭源设备驱动的复用问题。

为验证本文方法的可行性，实现了一个原型系统，对设备虚拟化功能和数据隔离性进行了验证。此外，对系统的CPU、内存、电量等性能开销还进行了大量的测试。实验和测试数据符合预期，并且说明本文技术是可行的，有效地实现了闭源设备的复用问题，使多个VP均可以使用同一个硬件设备。

借助于本文描述的Radio的虚拟化，可以实现多个虚拟手机拨打和接听电话，以及对手机上的多张SIM卡的使用进行策略控制。

本文的研究工作得到了电子科技大学“一校一带”基金(A03013023001021)的资助，在此表示感谢！

参 考 文 献

[1] BARHAM P, DRAGOVIC B, FRASER K, et al. Xen and the art of virtualization[J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164-177.

[2] RUSSELL R. Virtio: towards a de-facto standard for virtual I/O devices[J]. ACM SIGOPS Operating Systems Review,

2008, 42(5): 95-103.
[3] DONG Y, LI S, MALLICK A, et al. Extending Xen with intel virtualization technology[J]. Intel Technology Journal, 2006, 10(3): 193-203.
[4] KIVITY A, KAMAY Y, LAOR D, et al. KVM: the Linux virtual machine monitor[C]//Proceedings of the Linux symposium. Ottawa: Proc Linux Symposium, 2007, 1: 225-230.
[5] DALL C, NIEH J. KVM/ARM: the design and implementation of the linux ARM hypervisor[J]. ACM SIGARCH Computer Architecture News, 2014, 42(1): 333-348.
[6] HWANG J Y, SUH S B, HEO S K, et al. Xen on ARM: System virtualization using Xen hypervisor for ARM-based secure mobile phones[C]//Consumer Communications and Networking Conference, 2008, CCNC 2008, 5th IEEE. Las Vegas: IEEE, 2008: 257-261.
[7] LEE S M, SUH S B, JEONG B, et al. Fine-grained i/o access control of the mobile devices based on the xen architecture[C]//Proceedings of the 15th Annual International Conference on Mobile Computing and Networking. Beijing: ACM, 2009: 273-284.
[8] VARANASI P, HEISER G. Hardware-supported virtualization on ARM[C]//Proceedings of the Second Asia-Pacific Workshop on Systems. Shanghai: ACM, 2011: 11.
[9] SOLTESZ S, PÖTZL H, FIUCZYNSKI M E, et al. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors[C]//ACM SIGOPS Operating Systems Review. Lisbon: ACM, 2007, 41(3): 275-287.
[10] Enterprise Mobility Solutions Samsung Electronics Co. Ltd. White paper: an overview of samsung KNOX™[EB/OL]. [2013-06-01]. http://www. samsung. com/global/business/business-images/resource/white-paper/2013/06/Samsung_KNOX_whitepaper_June-0.
[11] LBE Tech. Parallel space[EB/OL]. [2016-06-03]. http://parallel-app.com/.
[12] YANG Xia, SUN Chao-qun. Research and implementation of multiple Android systems based on the container technique[J]. Journal of Chinese Computer Systems, 2016, 37(7): 1422-1427.
[13] Google Inc. Hardware abstraction layer[EB/OL]. [2016-06-03]. http://source.android.com/devices/index. html #Hardware-Abstraction-Layer.

编 辑 漆 蓉

(上接第865页)

[3] JEONG S, JANG Y J, KUM D. Economic analysis of the dynamic charging electric vehicle[J]. IEEE Transactions on Power Electronics, 2015, 30(11): 6368-6377.
[4] THEODOROPOULOS T V, DAMOUSIS I G, AMDITIS A J. Demand side management ICT for dynamic wireless EV charging[J]. IEEE Transactions on Industrial Electronics, 2016, 63(10): 6623-6630.
[5] OTT A L. Experience with PJM market operation, system design, and implementation[J]. IEEE Transactions on Power Systems, 2003, 18(2): 528-534.
[6] DRIESEN J, KATIRAEI F. Design for distributed energy resources[J]. IEEE Power and Energy Magazine, 2008, 6(3): 30-40.
[7] OU C H, LIANG H, ZHUANG W. Investigating wireless charging and mobility of electric vehicles on electricity market[J]. IEEE Transactions on Industrial Electronics, 2015, 62(5): 3123-3133.
[8] IOSIFIDIS G, GAO L, HUANG J, et al. A double-auction mechanism for mobile data-offloading markets[J]. IEEE/ACM Transactions on Networking, 2015, 23(5): 1634-1647.
[9] MAJUMDER B P, FAQIRY M N, DAS S, et al. An efficient iterative double auction for energy trading in microgrids[C]//2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG). [S.l.]: IEEE, 2014: 1-7.

编 辑 漆 蓉