里约热内卢：一个可以在移动系统之间共享I/O的系统解决方案

技术报告2013-12-17日，莱斯大学

阿达兰，林洪云，林中米大学

摘要

移动系统配备了多种I/O设备，包括摄像头、麦克风、传感器和调制解调器。有许多新的用例允许一个移动系统上的应用程序利用另一个系统的I/O设备。本文介绍了里约热内卢，一个支持未修改的应用程序，并公开了一个可供共享的I/O设备的所有功能。里约热内卢的设计在许多类型的I/O设备中都是常见的，因此大大减少了支持新的I/O设备的工程效率。我们在安卓系统上实现的里约热内卢总共有6700行
代码和支持四个I/O类，使用少于450个类特殊代码行。里约热内卢还支持不同形式因素的移动系统之间的I/O共享，包括智能手机和平板电脑。我们表明，里约热内卢在音频、传感器和调制解调器方面实现了接近本地I/O的性能，但由于两个系统之间的网络吞吐量限制，摄像头的性能明显下降，这可能会通过新兴的无线标准得到缓解。

1.介绍

如今，用户拥有各种各样的移动系统，包括智能手机、平板电脑、智能眼镜和智能手表，每一种设备都配备了大量的I/O设备，如摄像头、扬声器、麦克风、传感器和蜂窝调制解调器。有许多有趣的用例中，运行在一个移动系统上的应用程序访问另一个系统上的I/O，原因有三个基本原因。(i)移动系统可以处于不同的物理位置或方向上。例如，人们可以通过平板电脑相机应用程序来控制智能手机的高分辨率摄像头，从而更容易地捕捉到自画像。（ii）移动系统可以为不同的用户提供服务。例如，如果一个人的智能手机可以访问另一个设备的扬声器，他就可以为另一个用户播放音乐。（iii）某些移动系统具有独特的I/O设备，因为其独特的形式因素和目标用例。例如，用户可以使用智能手机中的调制解调器和SIM卡从平板电脑上打电话。

不出所料，目前存在着共享各种I/O设备的解决方案，如相机[1]、扬声器[2]、调制解调器（SMS）[3]和图形[4]。然而，这些解决方案有三个基本限制。(i)它们不支持未经修改的应用程序。例如，IP网络摄像头[1]和MightyText [3]不允许现有的应用程序远程使用摄像头或调制解调器；它们只支持自己的自定义应用程序。（ii）它们不公开I/O设备的所有功能以进行共享。例如，IP网络摄像头不支持远程调整各种摄像头参数，如对焦深度、曝光率、分辨率和白平衡等。强大文本支持来自其他设备的短信和彩信，但不支持电话。Wi-Fi扬声器[2]不支持扬声器上的固化均衡器功能。（iii）它们是I/O类专业的，需要重要的工程支持来支持新的I/O设备。例如，IP网络摄像头[1]可以共享摄像头，但不能共享调制解调器或传感器。

在本文中，我们介绍了里约热内卢（Remote I/O），这是一种针对移动系统的I/O共享解决方案，克服了上述三个限制。里约热内卢采用了一种拆分堆栈I/O共享模型，其中I/O堆栈，i。e.，从应用程序到I/O设备的所有软件层，都在一定的边界上在两个移动系统之间分割。所有跨越此边界的通信都在托管应用程序的移动系统上被拦截，并通过I/O设备转发到移动系统，在那里它们由I/O堆栈的其余部分提供服务。里约热内卢使用设备障碍作为其选择的边界。设备ile被用于类似于unix的操作系统中，如Android和iOS，来抽象许多类的I/O设备，提供一个与I/O类无关的边界。设备ile边界支持未修改的应用程序的I/O共享，因为它对应用程序层是透明的。它还通过允许一个系统中的进程与另一个系统中的设备驱动程序直接通信，向其他移动系统公开了每个I/O设备的全部功能。里约热内卢不是第一个利用设备边界边界的系统；我们之前的工作[5]使用设备边界作为单个系统内I/O虚拟化的边界。然而，在两个物理上独立的系统之间共享I/O设备会给如何正确地利用这个边界带来一系列不同的挑战，如下所述。

里约热内卢的设计和实现必须解决以下来自I/O堆栈跨两个系统拆分的基本挑战。(i)一个支持

成功可能会在设备文件上发出操作，要求驱动程序对进程的内存进行操作。然而，通过I/O共享，进程和驱动程序驻留在两个具有不同的物理内存的移动系统中。在里约热内卢中，我们使用分布式共享内存（DSM）设计支持跨系统内存映射，该设计支持进程、驱动程序和I/O设备（通过DMA）对共享页面的访问。我们还支持来自两个系统的协作的跨系统内存复制。（ii）移动系统通常通过无线连接进行通信，与同一系统内的进程和驱动程序之间的延迟相比，后者具有较高的往返延迟。为了解决这一挑战，我们减少了由于ile操作、内存操作或DSM一致性消息而导致的系统之间的往返次数。（iii）移动系统之间的连接可能由于移动性或可靠性问题而随时中断。这可能会在所有相关系统的操作系统中造成不可取的侧面缺陷。我们可以解决这个问题，可以在断开时正确清理远程I/O连接的残差，切换到同一类的本地I/O设备，如果存在，或者不存在，向应用程序返回适当的错误消息。

我们提出了一个针对安卓系统的里约热内卢的原型实现。我们的实现支持四个重要的I/O类：摄像头、音频设备，如扬声器和麦克风、传感器，如加速计，和蜂窝调制解调器（用于电话和短信）。它由大约6700行代码（LoC）组成，其中少于450行是I/O类的专业代码。它还支持异构移动系统之间的I/O共享，包括平板电脑和智能手机。参见[6]以获得里约热内卢的视频演示。

我们对Galaxy nexus智能手机上的里约热内卢进行了评估，并表明它(i)支持现有的应用程序，（ii）允许远程访问所有I/O设备功能，（iii）需要低工程efort来支持不同的I/O设备，（iv）为音频设备、传感器和调制解调器实现接近本地I/O的性能，但由于我们的设置和测试系统中Wi-Fi吞吐量的限制，摄像头共享的性能明显下降。随着新兴的无线标准支持更高的吞吐量，我们假设这种退化很可能在不久的将来消失。

2. 设计

在本节中，我们将描述里约热内卢的设计，包括它的架构和它为使用它的移动系统提供的保证。

.12分裂堆栈架构

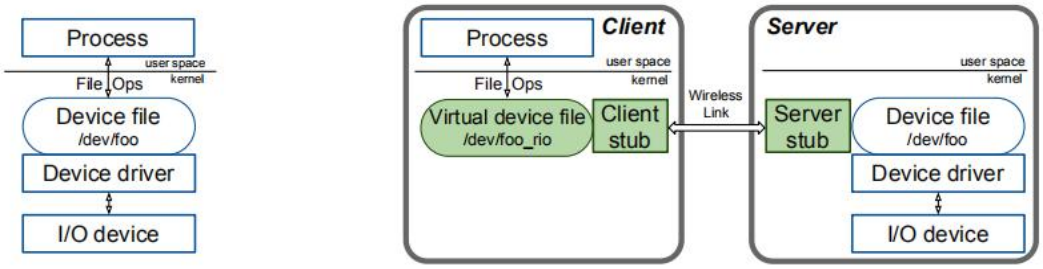
里约热内卢采用分栈模型进行移动系统之间的I/O共享。它在一个移动设备上的I/O堆栈中的设备文件边界处拦截通信系统，并将它们转发到另一个系统，由I/O堆栈的其余部分执行。

类似于unix的操作系统，如Android和iOS，使用设备iles来抽象许多类的I/O设备。图1(a)显示了类似于unix的操作系统中的典型的I/O堆栈。设备驱动程序在内核中运行，管理设备，并通过设备iles将I/O设备功能导出到用户空间进程。用户空间中的进程然后在设备文件上发出ile操作，以便与设备驱动程序交互。使用设备文件作为I/O共享边界的主要优点是，许多I/O设备都常见，减少了支持各种I/O类所需的工程效率。此外，这样的边界对应用程序层是透明的，并且立即支持现有的应用程序。它还允许进程直接与驱动程序通信，从而将所有I/O设备功能暴露给其他移动系统。

图1(b)描述了里约热内卢如何分割I/O堆栈。它显示了两个移动系统：服务器和客户端。服务器系统有一个客户端系统希望使用的I/O设备。里约热内卢在客户端创建与服务器中实际设备文件相对应的虚拟设备麻烦。虚拟设备文件对客户端进程产生I/O设备在客户端本地存在的错觉。要使用远程I/O设备，客户端中的进程将对虚拟设备文件执行ile操作。这些ile操作被客户端存根模块拦截，该模块将每个ile操作的参数打包到一个数据包中，并将其转发给服务器存根模块。服务器存根解包参数，并对实际的设备文件执行ile操作。然后，它将ile操作的返回值发送给客户机存根，后者将它们返回给进程。注意，图1(b)只显示了使用单个服务器的单个I/O设备的客户端。然而，里约热内卢的设计允许客户端使用来自多个服务器的多个I/O设备。它还允许多个客户端使用来自服务器的I/O设备。

在里约热内卢中，客户端进程始终是与服务器驱动程序进行通信的启动器。这是因为进程和驱动程序之间的通信总是由进程通过ile操作启动的。当I/O设备需要通知事件进程时，将使用轮询文件操作完成通知。要等待事件，进程发出一个阻塞轮询文件操作，该操作阻塞内核（因此在里约热内卢中的服务器内核中），直到事件发生。或者，它定期发布非阻塞轮询，以检查事件的发生。

一些ile操作，如读、写、ioctl和mmap，要求驱动程序对进程内存进行操作。mmap要求驱动程序将一些内存页面映射到进程地址空间中。为此，里约热内卢使用DSM设计，支持通过方法访问共享页面



类Unix系统中的(a) I/O堆栈(b)里约热内卢在设备文件边界处分割I/O堆栈

图1：里约热内卢在设备边界处分割I/O堆栈。远程使用I/O设备的进程位于客户端系统中，并与虚拟设备le进行交互。实际的设备设备、设备驱动程序和I/O设备都驻留在服务器系统中。里约热内卢在客户机和服务器之间转发所有操作。无线链路可以通过AP或设备到设备的连接。

客户端进程以及服务器驱动程序和设备（通过DMA）(? 3.1).其他三个ile操作要求驱动程序将数据复制到或从进程内存中复制数据。服务器存根拦截驱动程序对这些副本的请求，并通过客户端存根的协作(? 3.2).

.22保证

在设备文件边界上远程使用I/O会影响文件操作的三种预期行为：连接的可靠性、延迟和信任模型。也就是说，远程I/O引入了进程和驱动程序之间断开的可能性，由于无线往返，增加了每个ile操作的显著延迟，并允许不同信任域中的进程和驱动程序进行通信。因此，里约热内卢为客户端和服务器提供了以下保证。

首先，为了避免意外断开导致客户客户端和服务器中不希望的副作用，里约热内卢在检测到断开后在两个系统中触发清理。里约热内卢保证服务器的断开行为类似于杀死使用I/O设备的本地进程。里约热内卢还保证客户端将透明地切换到同一类别的本地I/O设备；否则，里约热内卢将向应用程序返回适当的错误消息(? 5).

其次，里约热内卢减少了由于文件或内存操作和DSM一致性消息(? 4)是为了减少延迟和提高性能。此外，它还保证了ile操作的额外延迟只影响I/O设备的性能，而不是影响其正确性。里约热内卢可以提供这种保证，因为大多数ile操作没有超时阈值，而只是阻塞，直到设备驱动程序处理它们。轮询是进程可以为其设置超时时间的唯一ile操作。在6.3，我们解释了我们目前支持的Android中的I/O设备使用的轮询操作不使用轮询超时机制。我们也解释了里约热内卢可以如何处理

如果使用，则显示投票超时。

最后，进程通常信任它们使用设备文件接口与之交互的设备驱动程序。为了保持这种信任，我们打算目前的里约热内卢设计仅用于可信移动系统之间的I/O共享。在10、我们将讨论如何增强当前的设计，以维护这一保证，同时支持在不可信的移动系统之间的I/O共享。

3. 跨系统存储器支持

为了处理ile操作，设备驱动程序通常需要通过执行内存操作来操作进程内存。然而，这些操作对里约热内卢带来了挑战，因为进程和驱动程序驻留在具有不同的物理内存的不同移动系统中。在本节中，我们将介绍我们的解决方案。

内存操作有三种类型。第一个是map_page，驱动程序使用它将系统或设备内存页面映射到进程地址空间。此内存操作用于处理mmap ile操作及其支持的page_falt。请注意，内核本身执行相应的unmap_page内存操作，而不是驱动程序。另外两种类型的内存操作分别是复制到用户和复制到用户[1]，驱动程序使用它将一个bufer从内核复制到进程内存中，反之亦然。这两个内存操作通常用于处理读、写和ioctl文件操作。

.13跨系统内存图

里约热内卢中的跨系统内存映射支持跨两个移动系统使用分布式共享内存（DSM）在它们之间进行的map_page内存操作[7-12]。里约热内卢的DSM的核心是一个简单的写-

[1] 我们使用现有的Linux函数的名称来引用这些操作。

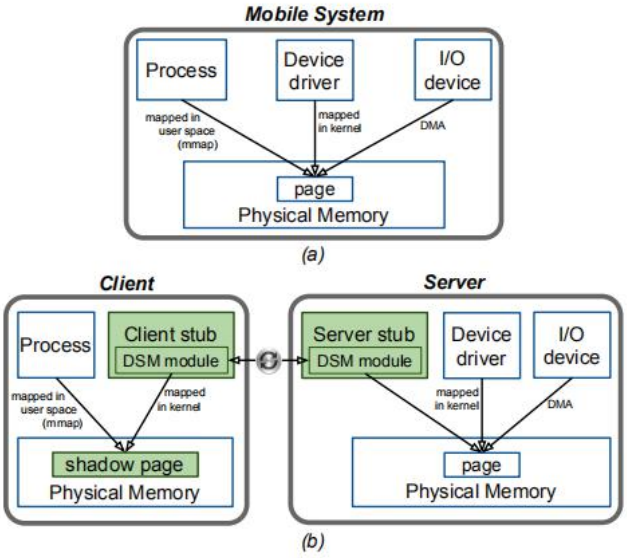


图2：本地I/O设备的(a)内存贴图。(b)跨系统内存映射在里约热内卢。

无效协议，类似于[11]。里约热内卢中的DSM的新颖之处在于，它不仅可以通过进程，还可以通过内核代码，如驱动程序，也可以通过设备（通过DMA）来支持对分布式共享内存页面的访问。

图2说明了里约热内卢中的跨系统内存映射。当从服务器驱动程序拦截map_page操作时，服务器存根通知客户端存根，然后客户端存根在客户端中创建一个影子内存页面（对应于服务器中的实际内存页面），并将该卷影页面映射到客户端进程地址空间。这两个存根中的DSM模块保证了进程、驱动程序和设备具有对这些页面的一致视图。也就是说，对实际页面和影子页面的更新都始终可用于其他移动系统。

我们在里约热内卢的DSM中选择了一个写无效的协议。与主动将更新传播到其他系统[10]的更新协议相比，只有在其他系统上需要更新数据时，使协议才会无效。这将使客户端和服务器之间传输的数据量最小化，从而减少资源消耗，e。g.，能量，在这两个系统中。对于无效的协议，每个内存页面可以处于三种可能的状态之一：读写、只读或无效。虽然无效协议是里约热内卢中的默认协议，但我们也可以使用更新协议，如果它的性能不佳；吗？6.2解释了这样的一个场景。

我们使用4KB的页面（小页面）作为一致性单元，因为它是map_page内存操作的单元，这意味着驱动程序可以将小到一个小页面的内存映射到进程地址空间中。

当许多页面一起更新时，我们将它们进行批处理以提高性能(？4.3)。

为了管理客户端进程对（影子）页面的访问，我们使用页表权限位，类似于一些现有的DSM解决方案[7]。当影子页处于读写状态时，页表将授予该进程对该页面的完全访问权限，并且该进程的所有读写指令都是本机执行的，没有额外的开销。在只读状态下，只写入这些页面会导致页面错误，而读写都会导致页面错误处于无效状态。在页面出现故障时，客户机存根会触发适当的一致性消息。对于读取错误，它将从服务器获取页面并重试进程的操作。对于写错误，如果处于无效状态，它首先获取页面，然后向服务器发送无效消息。

为了管理服务器驱动程序对页面的访问，我们使用页面表权限位作为内核内存，因为驱动程序在内核中运行。然而，与使用小的4KB页面的进程内存不同，内核内存的某些区域，例如Linux中的身份映射区域，使用更大的页面，例如ARM [13]中的1MB页面，以获得更好的TLB和内存容量。当驱动程序请求将大内核页面的一部分映射到进程地址空间时，服务器存根通过销毁旧页面表条目和创建新条目，动态地将大内核页面分解为多个小页面。通过这种技术，服务器存根可以对对每个小页面的内核访问执行不同的保护模式，而不是在大页面的粒度上执行保护。为了最小化在内核中使用小页面的副作用，例如，更高的TLB争用，当页面被进程未映射时，服务器存根立即将小页面打包回单个大页面。

为了管理服务器I/O设备通过DMA对页面的访问，服务器存根为每个页面维护一个显式的状态变量，拦截驱动程序对设备的DMA请求，相应地更新状态变量，并触发适当的一致性消息。请注意，设备不能使用页面表权限位来访问页面，因为设备的DMA操作位会绕过页面表。

里约热内卢支持顺序一致性有两个原因。首先，DSM模块在页面故障和DMA完成时立即触发一致性消息，并在每个系统中维护这些消息的顺序。其次，进程和驱动程序使用ile操作来协调它们自己对映射页面的访问。

3.2跨系统复制

里约热内卢中的跨系统内存复制支持两个移动系统之间的复制从用户和复制到用户内存操作。我们通过服务器和客户机存根之间的协作来实现这一点。当

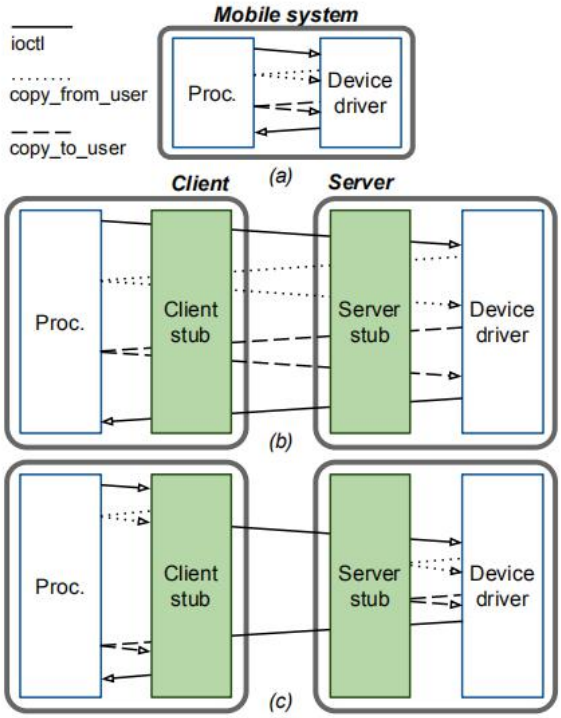


图3：典型的执行(a)操作为(a)的本地I/O设备、(b)和未优化里约热内卢（？3.2）的远程I/O设备、(c)和优化里约热内卢（？4.1）的远程I/O设备。如gure所示，优化已将往返次数从3次减少到1次。如果le操作需要更多的内存操作，那么减少可能会更明显。

服务器存根从驱动程序拦截复制从用户或拷贝到用户操作，服务器存根向客户端存根发送请求以执行该操作。在copy_from_user的情况下，客户端存根从进程内存中复制数据，并将其发送回服务器存根，服务器存根将其复制到由驱动程序确定的内核调试程序中。在copy_to_user的情况下，服务器存根复制数据并发送到客户端存根，然后客户机存根将其复制到相应的进程内存存根。图3 (b)说明了一个典型的ioctl ile操作的跨系统副本。

当处理单个ile操作时，驱动程序可能会执行多个内存复制操作，从而在移动系统之间造成尽可能多的往返操作，因为服务器存根必须为每个副本发送一个单独的请求。大量的往返行程会显著降低I/O性能。在4.1，我们解释了如何通过预取客户端存根中的复制数据来复制从用户操作，以及批处理服务器存根中的拷贝到用户操作的数据，将这些往返减少为每次文件只有一个操作。

4. 缓解高延迟

客户端和服务器之间的连接通常具有很高的延迟。例如，Wi-Fi和蓝牙最多有1-2 ms的往返延迟[14]，这明显高于进程和设备驱动程序之间典型的本地通信的几纳秒延迟(e。g.，系统调用）。在本节中，我们将讨论如此高延迟带来的挑战，并提出我们的解决方案，通过减少复制内存操作、文件操作和DSM一致性消息导致的往返次数来降低其对I/O性能的影响。

.14份因复制品而造成的往返行程

由于复制从用户和复制到用户内存操作导致的往返对里约热内卢的性能造成严重问题，因为单个ile操作可能连续执行多个复制内存操作。例如，Linux的PCM音频驱动程序中的一个ioctl可以执行四个复制_fror_用户操作。为了解决这个问题，我们使用了以下两种技术。(i)在客户端存根中，我们确定并预取服务器驱动程序所需的所有数据，并将其与ile操作一起传输。使用这种技术，来自驱动程序的所有copy_from用户请求都在服务器内部本地服务。（ii）在服务器存根中，我们推断出驱动程序打算复制到进程内存中的所有数据，并将其与ile操作的返回值一起传输到客户机。使用这种技术，所有copy_to_user操作都可以在客户端本地执行。图3 (c)说明了这些技术。

预取驱动程序复制用户请求的数据需要客户机存根模块提前确定驱动程序所需的进程内存数据程序的地址和大小。这对于读写文件操作来说是微不足道的，因为这些信息被嵌入到它们的输入参数中。然而，对于ioctl，这样做并不简单，因为ioctl的输入参数并不总是具有足够的描述性。正如在[5]中提到的，许多编写良好的驱动程序在ioctl输入参数之一i中嵌入了关于一些简单的驱动程序内存操作的信息。e.，ioctl命令编号。在这种情况下，我们通过解析客户端存根中的命令号来推断内存操作。但是，有些情况下，命令号并不包含所有必要的信息。对于这些情况，我们使用以前工作中的静态分析工具，该工具分析驱动程序的源代码来提取驱动程序代码的一小部分，然后可以在客户端存根中执行
推断驱动程序内存操作的参数。[5]提供了关于我们的静态分析工具的更多细节。

为了维护驱动程序的进程内存的一致视图，必须在发送copy_to_时更新服务器存根中的预取数据

用户数据，如果内存位置重叠。下面的代码段显示了一个来自Linux PCM音频驱动程序的相关示例。驱动程序在将整个数据结构从进程内存复制到内核之前，首先更新进程内存中的数据结构（使用put_user（）函数，本质上是一个copy_ to_user内存操作）。在处理copy_to_user时更新预取数据可以确保copy_frob_user数据不会来自陈旧的预取数据，从而保证一致性。

```
结构snd_xferi xferi；
结构化snd_xferi__用户*_xferi = arg；
. . .
if（put_user（0，&_xferi->结果））
        返回-EFAULT；
如果(copy_from_user(&xferi，_xferi，
                            Sizeof（xferi））
        返回-EFAULT；
```

.24.由于文件操作而导致的往返行程

文件操作由每个进程线程同步执行，因此，每个文件操作都需要一次循环。为了优化性能，流程应该发出尽可能少的ile操作数。更改ile操作的数量并不总是可能的，或者可能需要对流程源代码进行大量更改，例如，Android中的I/O服务代码，这不利于里约热内卢减少工程效率的目标。然而，对过程代码的最小更改偶尔会导致文件操作发布的显著减少，从而证明工程效率是合理的。?6.3解释了安卓音频设备的一个例子。

.34.由于DSM的一致性而造成的往返行程

如前所述？3.1，我们使用4 KB的小页面作为里约热内卢中的DSM相干单元。然而，当同时更新几个页面时，这样一个相对较小的连贯单元会导致对所有数据进行多次往返。在这种情况下，在一次往返旅行中一起传输所有更新后的页面会更加有趣。ffi?.26解释了安卓摄像头的一个例子。

.44.处理投票超时的问题

轮询是发布进程可以为其设置超时的唯一ile操作。由于里约热内卢为每个ile操作增加了明显的延迟，因此如果使用一个相对较小的超时阈值，它可以打破轮询的语义。到目前为止，在我们的Android实现中，我们支持的所有I/O类都不使用轮询超时（即，该过程要么独立地阻塞，直到事件准备好，要么使用非阻塞的轮询）。如果轮询与超时一起使用，则应针对远程I/O设备调整超时值。这可以在处理程序中完成

与投票相关的系统尺度，如选择。使用心跳往返时间（？5)，客户端存根可以提供系统调用处理程序需要添加到请求的超时值的附加延迟的最佳估计。进程通常依赖于内核来强制执行所请求的轮询超时；因此，这种方法保证了进程函数在面对高延迟时不会中断。在极少数情况下，进程使用外部计时器来验证其请求的超时，必须修改进程以适应远程I/O设备的额外延迟。

5.处理断开连接

由于移动性，客户端和服务器之间的连接可能在任何时候丢失。如果处理不当，断开可能会导致以下问题：使驱动程序不可用，导致阻塞客户端进程独立，或泄漏资源，e。g.，内存，在客户端和服务器操作系统中。当遇到断开连接时，服务器和客户机存根会采取适当的操作，如下所述。

我们使用一个超时机制来检测断开连接。客户端存根定期将心跳消息发送到服务器存根，服务器存根立即发送回确认。如果客户端存根在某个阈值之前没有收到确认，或者服务器没有收到客户端的消息，那么它们都会触发断开事件。我们不使用在光照下的ile操作作为心跳，因为ile操作可能需要不可预测的时间来在驱动程序中完成。确定最佳心跳间隔和超时阈值，以实现开销和检测精度之间可接受的交易，这是未来工作的一部分。

处理服务器中的断开：从服务器的角度来看，网络断开相当于杀死正在与驱动程序通信的本地进程。因此，就像操作系统清除已终止进程的残差一样，服务器存根也会在断开连接后清理客户端进程的残差。对于每个映射区域和每个打开的ile描述符，服务器存根分别调用驱动程序的close_map处理程序和释放ile操作处理程序，以便驱动程序释放所分配的资源。最后，它将关闭ile描述符，并发布它自己的簿记数据结构。

处理客户端中的断开：在断开时，我们在客户端中采取两个操作。首先，我们清理客户机存根中已断开连接的远程I/O的残差，类似于服务器中的清理过程。接下来，我们尽量使断开连接对应用程序尽可能透明。如果客户端拥有同一类别的本地I/O设备，则我们将在断开连接后透明地切换到该本地I/O设备。如果没有可比的I/O设备，我们重新-

类型	总循环流量	组件	LoC
类的	6291	服务器存根 客户端存根 存根之间共享 DSM 支持Linux内核代码	2801 1651 647 1192 327
级谱	431	照相机 - HAL - DMA 音频设备 传感器 蜂窝调制解调器	 36 134 64 128 69

表1：里约热内卢代码的分解。

打开API支持的适当的错误消息。这些行动需要阶级专业的发展，那么呢？6.3解释了我们如何为传感器实现这一点。对于我们目前支持的三种I/O类，可以切换到本地I/O，包括摄像头、音频和传感器，如加速度计。对于调制解调器，断开连接意味着电话将被取消或未启动，或将不会发送短信；所有这些都是现有应用程序可以理解的行为。

6. 机器人实现

我们已经为Android操作系统和ARM架构实现了里约热内卢。该实现目前支持四类I/O设备：传感器（包括加速度计）、音频设备（包括麦克风和扬声器）、摄像头和蜂窝调制解调器（用于电话和短信）。它由大约6700个LoC组成，其中不到450个是I/O类特殊的，如表1所示。我们已经在运行Linux内核3.0的安卓4.2.2的Galaxy Nexus安卓智能手机，以及在运行Linux内核的安卓4.2.2的三星Galaxy Tab 10.1平板电脑上测试了这个实现。该实现可以在不同形式因素的系统之间共享I/O：我们已经演示了这一点，可以在智能手机和平板电脑之间共享传感器。

图4显示了一个安卓系统内的里约热内卢的体系结构。在安卓系统中，应用程序进程并不直接使用设备文件与驱动程序进行交互。相反，它们通过类特定的api与类特定的I/O服务进程进行通信。I/O服务进程会加载一个硬件抽象层（HAL）库，以便使用该设备文件与该设备驱动程序进行交互。里约热内卢的设备ile边界位于I/O服务进程的下方，并将其ile操作转发到服务器。正如我们将在本节的其余部分中解释的那样，我们需要对HAL或I/O服务流程进行小的修改，但不需要对应用程序进行修改。

.16个客户端和服务器存根

客户机和服务器存根是里约热内卢的两个主要组件，构成了里约热内卢实现的一个大部分。每个存根都有三个模块。最不重要的

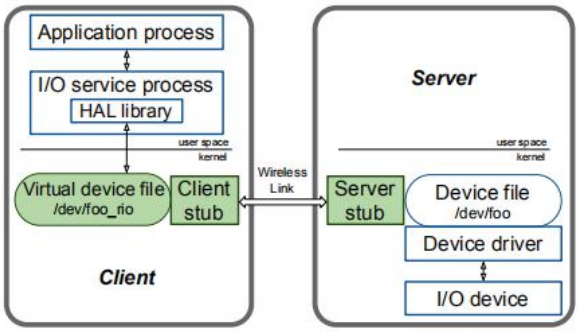


图4：里约热内卢在安卓系统内的架构。里约热内卢通过HAL将I/O服务进程发出的le操作转发给服务器。里约热内卢支持未修改的应用程序，但需要对类规格c I/O服务进程和/或HAL进行小的更改。

模块支持与应用程序和设备驱动程序的交互。在客户机存根中，此模块拦截ile操作并将其参数打包到数据结构中；在服务器存根中，它从数据结构中解包参数，并调用设备驱动程序的ile操作。第二个模块通过将数据结构序列化成数据包并将它们传输到另一端来实现与另一个存根的通信。最后，第三个模块实现了里约热内卢的DSM，并进一步解释了在？6.2.

我们使用内核TCP套接字用于客户端和服务器存根[15]之间的通信。我们使用TCP来确保成功接收到所有数据包，否则设备、驱动程序或应用程序可能会损坏。

为了处理跨系统内存操作，服务器存根拦截驱动程序的内核函数调用内存操作，并将它们转发给客户机存根。这包括拦截7个复制到user和复制到用户的内核函数，以及map_page的3个内核函数。拦截内核函数支持来自未经修改的驱动程序的内存操作。

.26 DSM模块

里约热内卢的DSM模块在客户端和服务器之间共享。它实现了DSM协议的逻辑，例如，触发和处理一致性消息。DSM模块在两种情况下被调用：页面故障和DMA。当出现页面故障时，我们会使用内核故障处理程序来调用DSM模块。此外，DSM模块必须处理设备DMA到受DSM保护的页面。我们监控驱动程序对设备的DMA请求，并在DMA完成时调用DSM模块。

为了监控驱动程序对设备的DMA请求，我们仪器了相应的内核功能。这些函数通常是I/O总线专用的，e。g.，I^2C和

将应用于使用该I/O总线的所有I/O设备。如果驱动程序使用非标准接口，则需要专门的仪器仪表。例如，Galaxy Nexus智能手机中TI OMAP4 SoC上的摄像头使用驱动程序和成像子系统（ISS）组件之间的自定义信息，其中摄像头硬件位于[16]。我们安装了负责与国际空间站通信的驾驶员来监控
DMA请求，只有134个LoC。

当我们收到内存应用程序的DMA完成通知时，我们可以使用DSM更新协议立即将更新的启动程序推送给客户端一个可选的优化。此外，我们在一次往返中更新了整个程序。这些优化可以提高性能，因为它们最小化了移动系统之间的往返次数(？？4.3)，因此我们用它们来制作相机帧。

如中所述？3.1，内核地址空间的某些区域，即身份映射区域，使用在ARM体系结构中称为secc的大型1MB页面。为了将这些1MB的部分分割成更小的4KB页面，用于我们的DSM模块，我们首先使用现有的页面表，以获得对该部分的第一级描述符（PGD条目）目的引用。然后，我们分配一个新页面，其中包含512个二级页面表条目（PTEs），每个页面对应一个；总共，这512个PTEs引用了两个1MB的虚拟部分
记忆力我们用来自原始部分的正确的页帧号和权限位填充每个二级PTE。最后，我们将一级描述符条目更改为指向我们新的二级pte表，并填充相应的缓存和TLB条目。

6.2.1支持缓冲区共享使用安卓离子离子

安卓使用ION内存管理框架来为多媒体应用程序分配和共享内存程序，比如使用GPU、摄像头和音频[17]的应用程序。离子层的共享给里约热内卢带来了独特的挑战，如下面的例子所示。

相机HAL使用ION分配程序，并将ION程序句柄传递给内核驱动程序，这将它们转换为这些层次的物理地址，并要求相机向它们DMA新帧。一次
写入帧，通知HAL，并将离子处理句柄转发到图形框架进行渲染。现在，想象一下在里约热内卢中使用远程摄像头。客户端摄像机HAL使用的相同离子图像处理需要被服务器的内核驱动程序和客户端图形框架使用，因为来自服务器的摄像机帧是在客户端显示上呈现的。

为了解决这个问题，我们提供了对全局离子推断的支持，可以在客户端和客户端内部使用
服务器我们通过在服务器中分配一个具有类似属性的离子器(e。g.，大小)到分配的客户端；我们使用DSM模块来保持两个层次的一致性。

6.3特定于类的开发项目

里约热内卢的大部分实现都是与I/O类无关的；我们目前的实现只需要不到450个类特殊的LoC。

解析命名标准：如果客户端具有相同类的I/O设备，其使用的设备文件与服务器中使用的名称相似，则虚拟设备文件必须采用不同的名称(e。g.，/开发/foovs。/dev/foo_rio在图1(b))。但是，设备文件名通常是在HAL中硬编码的，因此需要进行少量修改才能使用重命名的虚拟设备文件来进行远程I/O。

优化性能：如在？4，有时对HAL的小的更改可以通过减少ile操作的数量来显著提高远程I/O性能。例如，音频HAL与驱动程序我们的ioctls交换错误的音频段。HAL决定了每个ioctl的音频段的大小。对于本地设备（延迟非常低），这些错误的片段每个包含大约3 ms的音频，少于里约热内卢的往返时间。因此，我们修改了HAL，以便为远程音频设备使用更大的数据传输段。虽然这增加了音频延迟，但它提高了远程设备的音频速率。?8提供了量化这种交易的测量方法。这种修改只需要大约30个LoC。

支持热插拔和断开连接：远程I/O设备可以随时来来去去；在这个意义上，它们的行为类似于热插拔/删除本地I/O设备。可能需要对I/O服务层进行一些小的更改，以支持远程I/O设备的热插拔和断开连接。例如，Android传感器服务层在电话初始化过程中打开传感器设备的ile（通过HAL库），并且只使用这些ile描述符来读取传感器值。为了支持远程传感器一组传感器的热插拔，我们修改了传感器服务层，以打开虚拟设备文件，并在远程传感器存在时使用它们的文件脱脚器。在断开连接后，我们将切换回使用本地传感器来提供一个应用程序透明的机制。

避免重复的I/O初始化：一些HAL库，包括传感器和蜂窝调制解调器，在系统启动时执行I/O设备的初始化。但是，由于I/O设备已经在服务器中初始化，因此客户端HAL不应该尝试初始化I/O设备。这不仅可以破坏I/O设备，它还可以破坏HAL，因为服务器设备驱动程序拒绝基于初始化的ile操作。

因此，必须修改HAL，以避免两次初始化一个设备。对于传感器HAL来说，实现这一点是微不足道的：我们只需要注释一个LoC。然而，由于调制解调器的HAL不是开源的，我们必须采用一种解决方案，在客户端中的第二个SIM卡来初始化其调制解调器的HAL。我们正在开发一个小扩展的调制解调器内核设备驱动程序（这是开源的）为了伪造SIM卡的存在，并允许客户端HAL在没有第二张SIM卡的情况下初始化。

.46.异构系统之间的共享

因为设备文件边界在所有安卓系统中都很常见，里约热内卢的设计很容易支持异构系统之间的共享，例如，智能手机和平板电脑之间的共享。但是，实现必须正确地处理共享I/O的HAL库，因为它可能是I/O设备或系统中使用的SoC所特有的。我们的解决方案是将服务器中使用的HAL库移植到客户端。这样的端口很容易实现，但有两个原因。首先，每个I/O类的安卓到HAL的界面在不同不同形式因素的安卓系统中是相同的。其次，所有的安卓系统都使用Linux内核，而且大部分都附带ARM处理器。例如，我们通过在平板电脑源代码树中编译智能手机HAL，成功地将Galaxy Nexus智能手机传感器HAL库移植到三星Galaxy Tab 10.1平板电脑上。

7.用例

在本节中，我们将介绍两组里约热内卢的用例：第一组使用远程I/O设备利用未修改的应用程序，并且可以立即用里约热内卢演示（参见[6]的视频演示）；第二组需要新的或修改的应用程序，可能需要升级到OS I/O堆栈组件，e。g., Android I/O服务进程(? 6).

.1用里约热内卢演示了7个用例

多系统摄影：在里约热内卢中，人们可以使用一个移动系统上的相机应用程序，通过另一个系统上的相机来拍照。这种功能在许多不同的场景中都很方便，特别是在拍摄自画像时，因为它可以将相机硬件与相机卷视器、捕捉按钮和设置解耦。一些现有的应用程序试图帮助用户使用语音识别、音频引导或人脸检测[18]进行自画像。然而，里约热内卢的优势在于，用户可以(i)近距离查看相机，（ii）舒适地保持相机设置，（iii）随时准备好按下捕捉按钮，即使物理相机在几十英尺远。或者，你可以使用智能手机和平板电脑上的前置摄像头来捕捉自画像，但前置摄像头可以捕捉照片其分辨率明显低于后置摄像头。

多系统游戏：更大的移动系统，如平板电脑，提供了一个优越的游戏屏幕，但比口袋大小的智能手机更笨重。此外，使用传感器作为输入的游戏，e。g., 一个赛车游戏，需要倾斜的移动系统，使其更难集中精力在显示器的内容。然而，与里约热内卢，第二个移动系统，e。g., 这款智能手机可以用作手机游戏控制器，而较大的平板电脑屏幕则保持静止。

一张SIM卡，许多系统：尽管[19]做了很多努力，用户仍然绑定到一个SIM卡来打和接收电话或短信，主要是因为SIM卡与一个唯一的号码相关联。通过里约热内卢的I/O共享，用户可以使用智能手机中的调制解调器和SIM卡，从任何移动系统打和接收电话和短信。例如，如果用户在家里忘记了智能手机，她仍然可以在工作时接听平板电脑上的电话。

音乐分享：用户可能想让朋友通过智能手机上的音乐订阅应用程序来听一些音乐。使用里约热内卢，用户可以简单地在她朋友的智能手机扬声器上用任何现有的音乐播放应用程序播放音乐。

多系统视频会议：当用户在平板电脑上进行视频会议时，她可以使用智能手机上的扬声器或麦克风，使它们更靠近嘴部，从而在嘈杂的环境中获得更好的音频质量。在一个相关的场景中，她可以使用智能眼镜的摄像头作为平板电脑的外部摄像头，以提供不同的视角。

.27个里约热内卢的未来用例

使用里约热内卢，可以开发新的应用程序来使用另一个系统上可用的I/O设备。

多用户游戏：在前一小节中解释的多系统游戏用例，结合对应用程序的修改，可以实现跨移动系统的多用户游戏的新形式。例如，两名玩家可以使用他们的智能手机在他们面前的一个平板电脑上无线控制赛车游戏。智能手机的显示器甚至可以显示游戏内的上下文菜单或游戏控制器键，类似于游戏机上的那些，为用户提供了一个熟悉的和传统的游戏体验。

音乐共享：如果由音频服务流程和HAL支持，用户可以同时在她和她朋友的智能手机上同时播放相同的音乐。通过适当的应用程序支持，用户甚至可以同时在这两个系统上播放两个不同的音轨。

多系统视频会议：一个视频会议应用程序可以被扩展，以显示来自智能眼镜和平板电脑的并排视频流

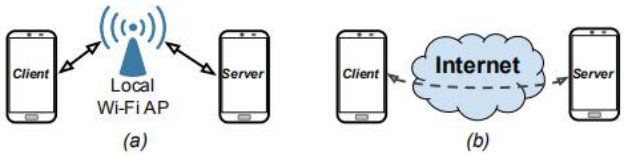


图5：评估设置。

同时地通过这种方式，用户不仅可以和她的朋友分享她的面部视频流，而且她还可以同时分享她面前的场景流。

多相机摄影：通过使用多个移动系统上的相机，可以实现各种计算摄影技术[20]。例如，用户可以同时使用她的智能手机和智能眼镜的相机来捕捉不同曝光时间的照片，以消除运动模糊[21]，或者通过交错/合并两个相机[22,23]的帧来使视频的时间/空间分辨率增加一倍。人们甚至可以把智能手机作为智能眼镜相机的外部捆绑器。

8. 评价

我们对里约热内卢进行了评估，并表明它(i)支持传统应用程序，（ii）允许访问所有I/O设备功能，（iii）需要低工程efort来支持不同的I/O设备，（iv）实现了与音频设备、传感器和调制解调器接近本地I/O的性能，但由于吞吐量限制，相机的性能下降。我们将进一步讨论，未来的无线标准将消除这个性能问题。

所有的实验都是在两款Galaxy Nexus智能手机上进行的。我们在实验中使用了手机之间的两个不同延迟的连接。第一次连接（图5(a)）是通过彼此接近的移动系统之间的无线局域网。g.，两者都是由用户携带的或在同一房间内携带的。我们将这两部手机连接到同一个Wi-Fi接入点，其中位数和平均延迟分别为4.4 ms、13.8 ms和14.3 Mbps。第二个连接（图5b）是在不同地理位置的移动系统之间，一个在家里，一个在20英里外的工作场所。我们通过来自商品互联网供应商的外部ip，通过互联网连接这两部手机。该连接的中位数和平均延迟分别为55.2 ms和56.9 ms和1.2 Mbps的吞吐量。除非另有说明，所有报告的结果都使用第一个局域网连接。

8.1非性能属性

首先，里约热内卢支持现有的未经修改的应用程序。我们已经使用不同类别的I/O设备，用各种默认的和第三方应用程序测试了里约热内卢。

第二，与现有的解决方案不同，里约热内卢公开了所有的func-

远程I/O设备的兼容性。例如，客户端系统可以保持每个相机参数，包括分辨率、曝光率、对焦度和白平衡。类似地，应用程序可以使用不同均衡器的扬声器。

在里约热内卢中支持新的I/O设备需要较低的工程效率。如表1所示，我们只需要128、64、170和69个LoC来分别支持传感器、音频设备（包括扬声器和麦克风）、摄像头和调制解调器（用于电话和SMS）。

.28性能基准

在本小节中，我们测量了里约热内卢中不同的I/O类的性能，并将它们与本地性能进行比较。

音频设备：我们通过测量不同的音频采样大小下的音频采样率来评估扬声器和麦克风的性能，因此，也有不同的音频延迟。使用更大的尺寸减少了与驱动程序的交互，但增加了音频延迟。音频延迟是一个样本从这个过程中到达扬声器所需的平均时间（对于麦克风，反之亦然），并直接由HAL中使用的音量大小决定。

图6显示了当通过本地或通过里约热内卢远程访问扬声器和麦克风的不同的讨论大小（以及因此不同的延迟）的实现速率。我们使用最小3 ms的漫游大小，因为它是最小的Android的低延迟模式。结果表明，如此小的尺寸会降低里约热内卢的音频速率。这主要是因为HAL为每个3 ms音频段发出一个ioctl，但由于网络的高往返时间，ioctl需要超过里约热内卢的时间。然而，igure表明里约热内卢能够在稍大的48 kHz音频速率的6-7 ms。我们相信里约热内卢实现了可接受的低音频延迟，因为安卓为扬声器使用308 ms的高延迟音频模式，并使用22 ms的麦克风。

当移动系统通过上述高延迟连接连接时，我们还用里约热内卢测量音频设备的性能，例如，在工作中使用智能手机远程打电话。我们的测量表明，里约热内卢达到了所需的48 kHz的麦克风使用的尺寸小至85 ms。然而，对于扬声器，里约热内卢只能使用300 ms的触发器实现25 kHz的最大采样率（其他触发器尺寸表现很差）。虽然这是无害的立体声音频（这需要48 kHz），它是足够的单声道音频。

相机：我们测量实时、流媒体相机预览和照片捕捉的性能。在第一种情况下，我们测量相机应用程序可以测量的帧率（每秒帧数）

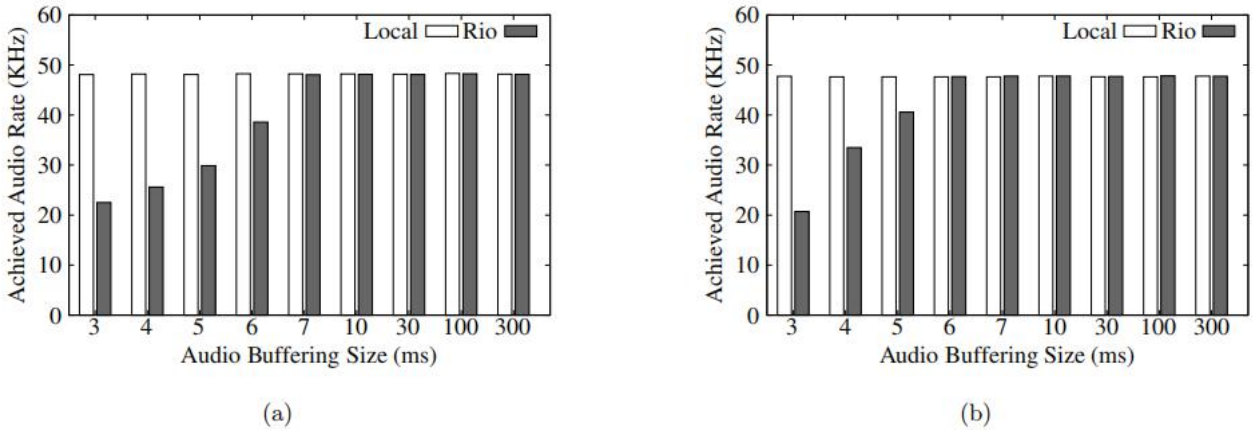


图6：扬声器(a)和麦克风(b).的性能在两个孔中，X轴显示HAL中的钻孔大小。ffff音量越大，播放效果就越平滑，但音频延迟就越大。Y轴表示所获得的音频速率。

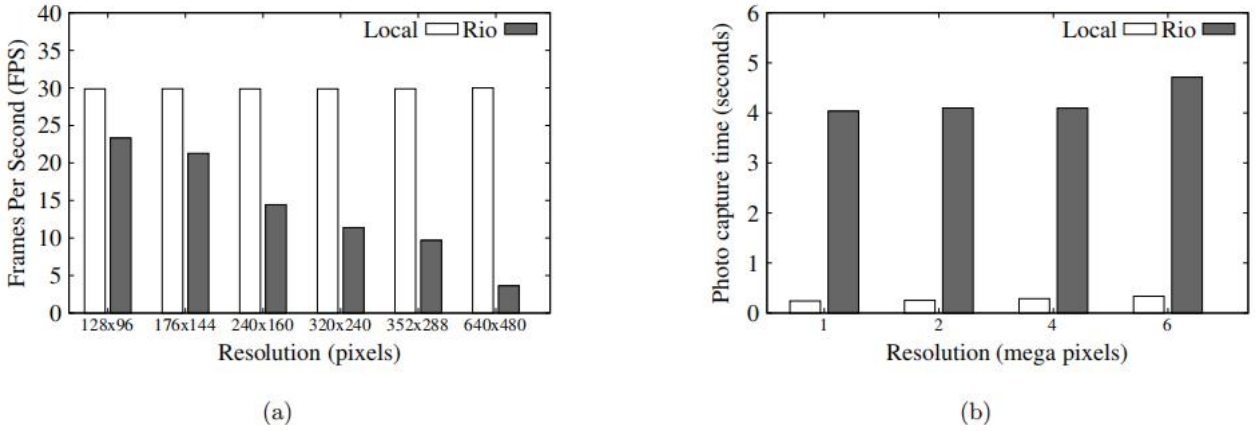


图7：在客户端和服务器之间有14.3 Mbps的无线局域网连接的实时流媒体摄像头预览(a)和照片捕获(b)的性能。未来具有更高吞吐量的无线标准将提高性能，而不需要更改里约热内卢。

实现，平均超过1000帧。我们忽略了前50帧，以避免相机初始化的缺陷
在性能上。

图7(a)显示，里约热内卢可以达到可接受的性能(i。e.，>15 FPS)，低分辨率。高分辨率的性能受到客户端和服务器之间的网络吞吐量的瓶颈。ffi里约热内卢的设计大部分时间传输帧而不是文件操作。然而，流媒体摄像机帧是未压缩的，即使是VGA（640480）分辨率，每帧也需要612 KB的数据，这需要大约72 Mbps的吞吐量来保持15 FPS。×

我们认为，里约热内卢支持的低分辨率相机预览是可以接受的，因为里约热内卢支持拍摄最高分辨率的照片。里约热内卢将使用未来的无线标准支持更高的实时摄像头分辨率；例如，802.11n，802.11ac，和一个802.11的ad可以分别实现约200 Mpbs、600 Mbps和7 Gbps的吞吐量[24,25]。这种吞吐量能力可以支持里约热内卢15 FPS的实时摄像头流，分辨率为1280720和19201080，这是Galaxy Nexus支持的最高分辨率。××

为了评估照片捕获，我们测量从捕获请求从应用程序发送到相机HAL到HAL通知应用程序照片已经准备好为止的时间。我们不包括对焦时间，因为它主要依赖于相机硬件和不同的场景。图7(b)显示了使用里约热内卢的本地和远程摄像机的捕获时间。该图像显示了每个分辨率的平均超过10张被捕获的照片。它显示，里约热内卢为捕获时间增加了明显的延迟，主要来自于将原始图像从服务器传输到客户端所花费的时间。但是，用户只需要

非常谨慎地将摄像机对准目标场景（类似于使用本地摄像机时），因为图像将立即在服务器中被捕获。值得注意的是，Galaxy Nexus中的HAL相机使用相同的尺寸大小，无论分辨率如何，因此捕获时间本质上是与分辨率无关的。bufer的大小是8 MB，通过我们的14.3 Mbps连接传输大约需要4.5秒。与实时摄像头流媒体一样，未来的无线标准将消除这种开销，提供与本地相机捕获相当的延迟。

传感器：为了评估遥感器的性能，我们测量了传感器HAL获得一个新的加速度计读数所需的平均时间。我们对10000个样本的测量表明，传感器HAL平均在65 ms获得一个新的本地读数，在71 ms通过里约热内卢获得一个新的远程读数。传感器HAL通过在内核中发出阻塞轮询操作来获得新的读取，该操作直到数据准备就绪，此时HAL发出读取文件操作来读取新值。
在里约热内卢这种情况下，里约热内卢会造成开销，因为阻塞轮询返回，直到读取完成。幸运的是，这种开销在实践中可以忽略不计，也不会影响用户体验。

调制解调器：我们分别测量拨号器和消息传递应用程序启动电话呼叫和发送短信所需的时间。我们测量从用户按下“拨号”或“发送短信”按钮到接收电话上出现通知的时间。我们的测量结果显示，本地和远程调制解调器实现了类似的性能，因为大部分时间都花在了运营商网络上（从T-Mobile到AT&T）。对于本地和远程调制解调器，电话通话大约需要7.8秒和7.9秒，而短信通话大约需要6秒。分别为2秒和5.9秒。

9. 相关工作

移动信息共享对于其他移动和非移动系统的价值。然而，现有的解决方案有局限性：它们不支持未修改的应用程序，不向客户端公开所有的I/O设备功能，或者是I/O类特殊的。

移动系统的I/O共享：现有的移动系统的I/O共享解决方案都来自于上面描述的三个基本限制。例如，IP网络摄像头[1]可以将安卓系统上的摄像头变成IP摄像头，然后可以通过自定义查看器应用程序从另一个移动系统上进行查看。客户端系统不能保留大多数摄像机参数，如果有；所有提示必须在服务器上手动完成。客户端也不能拍照。Wi-Fi扬声器[2]允许人们通过个人电脑在智能手机上播放音乐。它不允许客户端保持扬声器参数，例如，均衡器efects.MightyText [3]允许用户使用其他系统中的SIM卡和调制解调器从PC或移动系统发送短信和彩信信息。它不支持打电话。

屏幕共享：像奇迹[4]这样的应用程序允许一个系统发送屏幕在另一个系统上显示。瘦客户机解决方案还可以显示从客户端上的服务器机器接收到的内容。例如，X窗口系统[26]，THINC [27]，微软远程桌面[28]，VNC [29]，Citrix元帧[30]，和Sun Ray [31]。这些解决方案都没有使用设备边界，它们的边界选择通常是图形特殊的，甚至应用专用的。例如，X设置了应用程序和X服务器之间的边界。因此，这些解决方案将不支持其他类别的I/O设备

其他I/O共享解决方案：远程ile系统[32－34]、网络USB设备[35－39]、无线显示器[40]和IP摄像头[41]也支持I/O共享。这些解决方案也专门适用于一个I/O类，例如，存储。参与式传感系统从已注册的移动系统[42]中收集传感器数据。这些系统使用安装在移动系统上的自定义应用程序，因此比支持各种I/O设备的里约热内卢更有限。

计算o扭oading：有大量关于移动系统[43]的o扭oading计算的文献。g.，网络觅食的[44]，MAUI [45]和彗星[46]。在这项工作中所关注的I/O共享带来了一系列非常不同的研究挑战，并专注于系统支持，而不是编程支持。然而，计算o扭oading和I/O都共享来自分布式系统的已知技术。例如，彗星和里约热内卢都采用了DSM，尽管设计非常不同。

10. 结束语

我们提出了里约热内卢，一种移动系统的I/O共享解决方案，在设备文件边界采用分裂堆栈模型。我们证明了里约热内卢通过支持未修改的应用程序，向客户端公开所有I/O设备功能，以及减少开发速度，克服了现有解决方案的局限性。我们提出了一个针对Android的里约热内卢实现，并证明了它在各种共享场景下实现了足够的性能，并且它支持异构移动系统。接下来，我们将深入了解里约热内卢当前设计和实施的局限性，以及我们克服其中一些问题的计划。

支持更多类的I/O设备：我们当前的实现支持四类I/O设备。我们计划将其扩展到支持图形、触摸屏、GPS和FM广播，因为它们也使用设备ile接口。然而，里约热内卢的设计不能支持两类的I/O：网络和

阻止设备。这是因为这些I/O设备不使用设备ile接口来进行进程和驱动程序之间的通信。网络设备使用套接字和内核网络堆栈，块设备使用文件系统。

与不受信任的系统共享I/O：在本文中，我们假设通过里约热内卢共享I/O的系统彼此信任而不是恶意的(？ 2.2). 支持不受信任的系统给里约热内卢带来了新的挑战，它可以分为两类。(i)保护服务器。正如在[5]中所讨论的，设备驱动程序有漏洞，恶意应用程序可以通过设备ile接口滥用这些漏洞，从而危及驱动程序保护域[47,48]。在里约热内卢中，这意味着客户端中的恶意进程可能会危及服务器，例如，通过特权升级。为了解决这个问题，设备驱动程序和设备需要在服务器的一个保护域中进行沙箱，使用类似于[5]、Nooks [49]和VirtuOS [50]的技术。（ii）保护客户。不受信任的服务器可能会向客户端发出虚假的复制内存操作，以危及客户端。客户端存根可以通过严格检查服务器请求的复制内存操作来简单地防止这种威胁，类似于[5]。请注意，服务器还可以窥探与I/O设备共享的客户端数据，e。g.，音频bufer。由于服务器完全不受信任，因此我们不能为客户机的数据提供任何隔离。这确实是任何I/O共享系统所固有的问题，而不是里约热内卢的特殊问题。

里约热内卢的能源使用：通过无线远程使用I/O设备显然比使用本地设备会消耗更多的能源。里约热内卢使用的设备文件边界相当抽象，因为里约热内卢的大部分能源使用来自于传输I/O数据。由于空间的限制，我们无法详细阐述里约热内卢的能量优化。相反，我们注意到大部分由里约热内卢，e。g.，那些描述在什么地方吗？ffi4会导致更方便的使用无线设备，从而减少能源消耗。我们还注意到，减少延迟的首要任务也排除了使用具有I/O共享的标准802.11省电模式。ffi另一方面，许多已知的技术，用一点延迟换取更多的无线使用，可以支持里约热内卢，例如，数据压缩[51]和uPM [52]。

支持iOS： iOS也使用设备iles，因此可以支持里约热内卢。在iOS系统之间共享I/O设备应该需要类似于本文中报道的工程版本，以便在Android系统之间共享I/O设备。然而，在iOS和Android系统之间共享I/O设备需要潜在的重要工程efort，主要是因为这两个系统有不同的I/O堆栈组件，因此也有不同的I/O设备API。

11. 参考文献

[1]安卓IP网络摄像头应用程序。 https://play.谷歌。com/store/apps/详述id=com.pas.webcam&hl=en.

[2]安卓Wi-Fi扬声器应用程序。 https://play.谷歌。com/store/apps/详述id=pixelface.机器人。audio&hl=en.

[3]Mighty文本应用程序。 http://mightytext.网.

[4] Miracast.https://www .wi-fi.org/站点/默认/文件/上传/wp_Miracast_Industry_20120919。pdf.

[5] A.AmiriSaniK。靴子，S。秦和L。钟。设备文件边界处的I/O准虚拟化。要出现在Proc中。ACM片，2014年。

[6]里约热内卢项目主页（包括一个视频演示）。 http://www.ruf.稻米edu/~mobile/rio.html.

[7] Kai李。Ivy：一个用于并行计算的共享虚拟内存系统。在程序中。1988 Int.会议平行处理，第2卷，第94-101页，1988年。

[8]加里·斯科特Delp。建筑和memnet的实现：一个高速共享存储器的计算机通信网络。1988.

[9]：丹尼尔·勒诺斯基、詹姆斯·劳登、库罗什·加拉乔罗、安诺普·古普塔和约翰·轩尼诗。针对DASH多处理器的基于目录的缓存一致性协议，第18卷。ACM，1990年。

[10]约翰B。卡特，约翰K。班尼特和威利·兹韦内波尔。市政会议的实现和性能。在程序中。ACM。

周[11]，斯通，李凯和大卫沃特曼。异构的分布式共享内存。并行和分布式系统，IEEE交易记录，3(5)：540-554,1992年。

[12]，舍伊纳斯，巴巴克·法尔赛，阿尔文·莱贝克，史蒂文·莱因哈特，詹姆斯·拉鲁斯和大卫·伍德。针对分布式共享内存的细粒度访问控制，卷29。ACM，1994年。

[13]臂。体系结构参考手册，ARMv7-A和ARMv7-R版。ARM DDI，0406A，2007。

[14]·瑞安·伍德斯和Manoj Pandey。一个低功耗，低延迟和干扰免疫无线标准。在程序中。IEEE无线通信和网络会议（WCNC），2006年。

[15] Linux套接字。 http://ksocket.sourceforge.net/.

[16]德州仪器。体系结构参考手册，OMAP4430多媒体设备硅版本2。x.SWPU231N，2010.

[17]安卓离子内存分配器。 http://lwn.net/Articles/480055/.

[18]应用程序。

http://giveawaytuesdays.wonderhowto.com/inspiration/10iphone-and-android-appsfor-takingself-portraits-0129658/.
[19] http://www .theverge .com/2011/06/09/google-voiceskypeimessage-and-thedeathof-the-phone-number/.
[20] R.RaskarJ。汤布林，A。莫汉。阿格拉瓦尔和Y。列支敦士登计算摄影。在程序中。STAR欧洲图形，2006。
[21]陆元，吉安孙，权，和Heung-Yeung舒姆。具有模糊/噪声图像对的图像去模糊。ACM图形交易（TOG），26(3)：2007年1月。
[22]贝内特·威尔伯恩，尼尔·乔希，瓦巴夫·瓦什，埃诺-维尔·塔瓦拉，埃米利奥·安图尼斯，亚当·巴斯，安德鲁·亚当斯，马克·霍洛维茨和马克·利沃伊。使用大型照相机阵列的高性能成像。ACM图形事务处理（TOG），24(3)：765-776,2005年。
[23]唱清公园，敏久公园和月亮吉康。超分辨率图像重建：一个技术概述。IEEE信号处理杂志，20(3)：2003年21-36。
[24] 802.11ac：第五代Wi-Fi。在2012年的思科白皮书中。
[25]无线局域网在60 GHz-IEEE802.11广告解释。安捷伦白皮书。
[26] R.W.谢勒和J。格蒂斯X窗口系统。ACM图形事务处理，第5(2)页，1986年。
[27] R.A.Baratto, L.金和J。尼赫。Thinc：一种用于瘦客户端计算的远程显示架构。在程序中。ACM SOSP，2004。
[28] B.C.坎伯兰，G。Carius和A。缪尔地貌名称微软视窗NT服务器4。0，终端服务器版本：技术参考。微软出版社，1999年。
[29] T.理查森，问。斯塔福德-弗雷泽，K。R.木材和A。漏斗虚拟网络计算。IEEE互联网计算，1998。
[30] Citrix元帧。http://www .柠檬酸。com.[31]布莱恩K。施密特，莫妮卡S。Lam和J。杜安诺斯卡特。slim的交互式性能：一个无状态的瘦客户机体系结构。在程序中。ACM，1999年。
[32]网络文件系统。http://etherpad.工具ietf.org/html/rfc3530.
[33] A.P.里夫金，M。P.福布斯，R。L.汉密尔顿M.Sabrio, S.沙阿和K。岳Rfs架构概述。在程序中。用户协会会议，1986年。
[34] P.J.Leach和D。奈克。一个通用的互联网文件系统（cifs/1.0）协议。草案，网络工作组，IETF，1997年。
设备上的[35] Web服务：设备控制在网络上。http://msdn.微软。com/en-us/library/windows/desktop/aa826001(v=vs .85).aspx.
[36] A.哈里，M。是的，Y。J.张和A。弗朗西尼。交换军队智能手机：基于云的USB服务交付。在程序中。ACM莫比赫尔德，2011年。
[37] Digi国际：任何地方的USB。http://www .数字的com/products/usb/anywhereusb.jsp.
[38] USB通过IP。http://usbip .sourceforge .net/.[39]无线USB。http://www.usb.org/wusb/home/.
[40]英特尔WiDi。http://www .英特尔。com/content/架构和技术/集成式无线显示器。html.
[41] Dropcam。https://www .德罗卡姆。com/.
[42] T.达斯，P。莫汉，五。帕德马纳班，R。拉姆吉和A。夏尔马棱镜：使用智能手机的遥感平台。在程序中。ACM MobiSys，2010年。
[43]杰森·弗林。网络觅食：连接移动计算和云计算。关于移动和普适计算的综合讲座，2012年。
[44] Rajesh克里希纳巴兰，达伦格尔，马哈德夫萨蒂亚那拉扬，詹姆斯赫斯勒布。简化了对移动设备的网络搜索。在程序中。ACM MobiSys，2007年。
[45]爱德华多·库尔沃，阿鲁纳·巴拉苏布拉曼尼亚，赵达基，亚历克·沃尔曼，斯特凡·萨罗乌，兰维尔·钱德拉和帕拉姆维尔·巴尔。毛伊岛：用代码o扭oad让智能手机的使用寿命更长。在程序中。ACM MobiSys，2010年。
[46]马克·戈登，贾姆希迪，斯科特·马尔克，毛莫利和徐陈。彗星：通过透明地迁移执行来编写代码o扭oad。在程序中。OSDI，2012.
[47]特权升级使用NVIDIA GPU驱动程序错误。http://www.securelist.com/en/advisories/50085.
[48]特权升级使用DRM/Radeon GPU驱动程序错误。https://lkml.org/lkml/2010/1/18/106.
[49]迈克尔M斯威夫特，布莱恩N贝尔沙德，和亨利M利维。提高商品操作系统的可靠性。在程序中。ACM SOSP，2003。
[50] Ruslan·尼古拉耶夫和戈德玛回来了。VirtuOS：一个具有内核虚拟化的操作系统。在程序中。ACM SOSP，2013。
[51]肯尼斯·巴尔和克斯特·阿萨诺维奇。具有能量感知能力的无损数据压缩。在程序中。ACM MobiSys，2003年。
[52]刘家阳和林钟。.11主动802接口的微电源管理。在程序中。ACM MobiSys，2008年。