(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102662910 A
(43)申请公布日 2012.09.12

(21)申请号 201210081358.2

(22)申请日 2012.03.23

(71)申请人 浙江大学
地址 310027 浙江省杭州市西湖区浙大路38号

(72)发明人 王总辉 陈文智 黄大鹏

(74)专利代理机构 杭州天勤知识产权代理有限公司 33224
代理人 胡红娟

(51)Int.Cl.
G06F 15/173(2006.01)

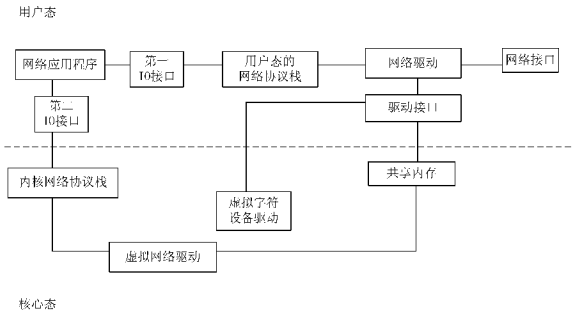
权利要求书 2 页 说明书 8 页 附图 2 页

(54)发明名称

基于嵌入式系统的网络交互体系及网络交互方法

(57)摘要

本发明公开了一种基于嵌入式系统的网络交互体系及网络交互方法，所述的网络交互体系包括：用户态中的网络接口、网络驱动、用户态网络协议栈以及第一IO接口；第二IO接口以及驱动接口；核心态中的内核网络协议栈、虚拟网络驱动以及虚拟字符设备驱动；以及，共享内存。所述的网络交互方法包括以下步骤：初始化；以及：当消息携带由用户态网络协议栈执行的通用命令时，消息的接收与发送；当消息携带由内核网络协议栈执行的通用命令时，消息的接收与发送。本发明将用户态的网络驱动与内核网络协议栈进行衔接，实现了传统网络应用程序也能在用户态网络驱动下使用，优化了网络通讯的性能。



CN 102662910 A

1. 一种基于嵌入式系统的网络交互体系，所述的嵌入式系统能够通过系统调用进入用户态以及内核态，其特征在于，所述的网络交互体系包括：

用户态中的网络接口、网络驱动、用户态网络协议栈以及第一 IO 接口；第二 IO 接口以及驱动接口；

核心态中的内核网络协议栈、虚拟网络驱动以及虚拟字符设备驱动；以及，

共享内存；

所述的第一 IO 接口以及第二接口用于从网络应用程序接收消息以及向网络应用程序传递消息；其中，通过第一 IO 接口进行交互的消息携带由用户态网络协议栈的特定网络协议层执行的通用命令，通过第二 IO 接口进行交互的消息携带由内核网络协议栈的特定网络协议层执行的通用命令；

所述的用户态网络协议栈，处理用于执行的通用命令，依据通用命令生成由网络驱动执行的特定命令；

所述的内核网络协议栈，处理用于执行的通用命令，依据通用命令生成由虚拟网络驱动执行的特定命令；

虚拟网络驱动，执行内核网络协议栈的特定命令，访问内核网络协议栈并实现共享内存与内核网络协议栈之间的消息交互；

驱动接口，访问共享内存并实现网络驱动与共享内存之间的消息交互；以及，

虚拟字符设备驱动，用于向驱动接口提供内核态的控制接口以及将所述的共享内存映射至用户态的地址空间。

2. 如权利要求 1 所述的网络交互体系，其特征在于，所述的第二 IO 接口为套接字接口。

3. 如权利要求 1 所述的网络交互体系，其特征在于，所述的共享内存被组织成循环缓冲区，并采用无锁的方式进行读写。

4. 如权利要求 1 所述的网络交互体系，其特征在于，所述的驱动接口通过调用 vn_put 函数以及 vn_get 函数读取共享内存中的数据。

5. 如权利要求 1 所述的网络交互体系，其特征在于，虚拟字符设备驱动的控制接口包括 ioctl 接口以及 mmap 接口。

6. 一种利用如权利要求 1 所述网络交互体系实现的网络交互方法，其特征在于，包括以下步骤：

(1) 对网络交互体系的初始化，加载虚拟络驱动以及虚拟字符设备驱动，建立共享内存；

(2) 当所述的消息携带由用户态网络协议栈执行的通用命令，消息的接收或发送包括步骤：

在网络接口或第一 IO 接口中载入消息；

读取所述的消息；

基于所述的消息，对网络驱动以及用户态网络协议栈发起操作；

向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第一 IO 接口中载入响应；以及，

读取所述的响应；

(3) 当所述的消息携带由内核网络协议栈执行的通用命令，消息的接收或发送包括步

骤：

在网络接口或第二 IO 接口中载入消息；

读取所述的消息；

基于所述的消息，对网络驱动、驱动接口、共享内存、虚拟网络驱动以及内核网络协议栈发起操作；

向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第二 IO 接口中载入响应；以及，

读取所述的响应。

7. 如权利要求 6 所述的网络交互方法，其特征在于，在步骤 (1) 中，所述的建立共享内存，包括步骤：

(1.1) 申请两段各 512 个页面大小的空间，计算出每个页的页框号，将这些页全设置成为保留页；

(1.2) 网络驱动通过驱动接口控制所述的虚拟字符设备驱动，将所述两段空间映射到用户态的地址空间。

8. 如权利要求 6 所述的网络交互方法，其特征在于，所述的共享内存包括发送缓冲区以及接收缓冲区，在步骤 (3) 中，

所述的向网络应用程序发送消息，包括步骤：

网络驱动调用驱动接口检查共享内存的接收缓冲区是否满，若满则丢包，不满则通过驱动接口将消息发送至接收缓冲区；以及，

内核网络协议栈调用虚拟网络驱动轮询接收缓冲区，通过虚拟网络驱动读取接收缓冲区内的消息并与网络应用程序进行消息交互；

所述的从网络应用程序传送消息，包括步骤：

内核网络协议栈调用虚拟网络驱动检查共享内存的发送缓冲区是否满，若满则丢包，不满则通过虚拟网络驱动将消息放入共享内存的发送缓冲区；以及，

网络驱动调用驱动接口轮询发送缓冲区，通过驱动接口读取发送缓冲区内的消息并与网络接口进行消息交互。

9. 如权利要求 8 所述的网络交互方法，其特征在于，所述的轮询包括步骤：

a、在共享内存上建立工作队列；

b、将轮询任务放入到工作队列；

c、检查共享内存的缓冲区是否为空，如果未空则一直读取消息；如果共享内存的缓冲区为空，则将轮询任务放入工作队列等待指定的时间。

基于嵌入式系统的网络交互体系及网络交互方法

技术领域

[0001] 本发明涉及嵌入式系统，特别涉及一种基于嵌入式系统的网络交互体系及网络交互方法。

背景技术

[0002] 当前计算机系统发展的趋势是体积的减小，于是便出现了掌上电脑和与之适应的嵌入式系统。嵌入式系统为一种计算机操作系统，当其运行在如掌上电脑一类的计算机上，往往具有实时系统的特征，但嵌入式系统在体积、内存容量以及电源等方面受到限制。这就要求操作系统和应用程序必须有效地对内存进行管理。由于嵌入式系统一般不使用虚拟储存技术，开发人员往往仅能对有限的物理内存做适应开发。

[0003] 现有技术的处理器或处理器核能够在两种模式下运行，即内核态和用户态。当嵌入式系统通过系统调用进入内核态时，处理器执行内核代码。完成嵌入式系统的一些功能。当嵌入式系统进入内核态时，应用程序的上下文将从当前执行的处理器或处理内核上切换出去，即现有技术的嵌入式系统采用了“上下文切换”来支持内核态与用户态的切换。这种上下文切换机制带来了许多物理内存上的开销，尤其是对于一些操作系统密集型应用，若频繁进行上下文切换对性能有较大的影响，降低嵌入式系统的运行速度。

[0004] 为了在嵌入式系统的网络交互方面克服上述问题，当前的许多嵌入式系统已将网络驱动及网络协议栈从内核态实现变为在用户态实现，一般的网络应用程序可通过修改其内部代码以支持网络协议栈，其基于的网络交互体系包括：

[0005] 网络接口；

[0006] 实现于用户态的网络驱动，将消息读取至网络接口；

[0007] 实现于用户态的网络协议栈，包括多个特定的网络协议层；

[0008] IO 接口，被安排用于从网络应用程序接收消息与向网络应用程序传递消息；所述的消息为携带为由特定网络协议层执行的通用命令。

[0009] 这种将网络驱动以及网络协议栈实现在用户态的嵌入式系统能够在很大程度上减少了内核态与用户态之间的上下文切换，缩短了数据的传递路径，并且提高了网络交互性能，嵌入式系统的内存也得到了有效的利用。

[0010] 但此种嵌入式系统的网络驱动无法与内核网络协议栈衔接。由于一些网络应用程序（包括网络调试工具）无法修改其内部代码，故仅支持内核协议栈，网络驱动无法直接与这些网络应用程序进行数据交换。因此，这种将网络驱动以及网络协议栈实现在用户态的嵌入式系统会导致部分网络应用程序无法使用。

发明内容

[0011] 本发明的目的在于提供一种基于嵌入式系统的网络交互体系及网络交互方法，能够对用户态网络驱动与内核网络协议栈进行衔接，实现了仅支持内核网络协议栈的网络应用程序也能在用户态网络驱动场景下使用，该类型的嵌入式系统得到了普适。

[0012] 一种基于嵌入式系统的网络交互体系，所述的嵌入式系统能够通过系统调用进入用户态以及内核态，所述的网络交互体系包括：

[0013] 用户态中的网络接口、网络驱动、用户态网络协议栈以及第一 IO 接口；第二 IO 接口以及驱动接口；

[0014] 核心态中的内核网络协议栈、虚拟网络驱动以及虚拟字符设备驱动；以及，

[0015] 共享内存；

[0016] 所述的第一 IO 接口以及第二接口用于从网络应用程序接收消息以及向网络应用程序传递消息；其中，通过第一 IO 接口进行交互的消息携带由用户态网络协议栈的特定网络协议层执行的通用命令，通过第二 IO 接口进行交互的消息携带由内核网络协议栈的特定网络协议层执行的通用命令；

[0017] 所述的用户态网络协议栈，处理用于执行的通用命令，依据通用命令生成由网络驱动执行的特定命令；

[0018] 所述的内核网络协议栈，处理用于执行的通用命令，依据通用命令生成由虚拟网络驱动执行的特定命令；

[0019] 虚拟网络驱动，执行内核网络协议栈的特定命令，访问内核网络协议栈并实现共享内存与内核网络协议栈之间的消息交互；

[0020] 驱动接口，访问共享内存并实现网络驱动与共享内存之间的消息交互；以及，

[0021] 虚拟字符设备驱动，用于向驱动接口提供内核态的控制接口以及将所述的共享内存映射至用户态的地址空间。

[0022] 下面介绍本发明的优选技术方案。

[0023] 具体地，所述的第二 IO 接口为套接字接口。

[0024] 进一步地，所述的共享内存被组织成循环缓冲区，并采用无锁的方式进行读写。

[0025] 进一步地，所述的驱动接口通过调用 vn_put 函数以及 vn_get 函数读取共享内存中的数据。

[0026] 进一步地，虚拟字符设备驱动的控制接口包括 ioctl 接口以及 mmap 接口。

[0027] 一种利用如上所述网络交互体系实现的网络交互方法，包括以下步骤：

[0028] (1) 对网络交互体系的初始化，加载虚拟络驱动以及虚拟字符设备驱动，建立共享内存；

[0029] (2) 当所述的消息携带由用户态网络协议栈执行的通用命令，消息的接收或发送包括步骤：

[0030] 在网络接口或第一 IO 接口中载入消息；

[0031] 读取所述的消息；

[0032] 基于所述的消息，对网络驱动以及用户态网络协议栈发起操作；

[0033] 向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第一 IO 接口中载入响应；以及，

[0034] 读取所述的响应；

[0035] (3) 当所述的消息携带由内核网络协议栈执行的通用命令，消息的接收或发送包括步骤：

[0036] 在网络接口或第二 IO 接口中载入消息；

[0037] 读取所述的消息；

[0038] 基于所述的消息，对网络驱动、驱动接口、共享内存、虚拟网络驱动以及内核网络协议栈发起操作；

[0039] 向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第二 IO 接口中载入响应；以及，

[0040] 读取所述的响应。

[0041] 在步骤 (1) 中，所述的建立共享内存，包括步骤：

[0042] (1.1) 申请两段各 512 个页面大小的空间，计算出每个页的页框号，将这些页全设置成为保留页；

[0043] (1.2) 网络驱动通过驱动接口控制所述的虚拟字符设备驱动，将所述两段空间映射到用户态的地址空间。

[0044] 进一步地，所述的共享内存包括发送缓冲区以及接收缓冲区，在步骤 (3) 中，

[0045] 所述的向网络应用程序发送消息，包括步骤：

[0046] 网络驱动调用驱动接口检查共享内存的接收缓冲区是否满，若满则丢包，不满则通过驱动接口将消息发送至接收缓冲区；以及，

[0047] 内核网络协议栈调用虚拟网络驱动轮询接收缓冲区，通过虚拟网络驱动读取接收缓冲区内的消息并与网络应用程序进行消息交互；

[0048] 所述的从网络应用程序传送消息，包括步骤：

[0049] 内核网络协议栈调用虚拟网络驱动检查共享内存的发送缓冲区是否满，若满则丢包，不满则通过虚拟网络驱动将消息放入共享内存的发送缓冲区；以及，

[0050] 网络驱动调用驱动接口轮询发送缓冲区，通过驱动接口读取发送缓冲区内的消息并与网络接口进行消息交互。

[0051] 更进一步地，所述的轮询包括步骤：

[0052] a、在共享内存上建立工作队列；

[0053] b、将轮询任务放入到工作队列；

[0054] c、检查共享内存的缓冲区是否为空，如果未空则一直读取消息；如果共享内存的缓冲区为空，则将轮询任务放入工作队列等待指定的时间。

[0055] 为了将网络驱动收到的消息传递给内核网络协议栈（即消息的接收）或将内核上层网络协议栈需要发送的网络消息包传递给用户态网络驱动程序（即消息的发送），首先得有一种内核态与用户态的通讯方式。

[0056] 由于传统的内核态与用户态通讯方式是系统调用，频繁地系统调用会带来频繁的上下文切换，带来较大的性能损失，另外由于内核态和用户态地址空间的不同，通过系统调用的方式在两态间传递消息必须进行消息拷贝，这又会带来性能损失，为了避免这些性能损失，采用建立共享内存的方式实现内核态与用户态的通讯，大大减少了系统调用的次数与消息的拷贝次数。同时，为了优化共享内存，可将所述的共享内存组织为一循环缓冲区，并采用无锁方式进行读写，能够避免使用互斥锁带来的开销。但这种读写方式必须使用轮询方式来查看缓冲区是否为满或为空。

[0057] 但一般的轮询由于一直检测缓冲区会导致 CPU 负荷过重，造成很大的性能损失。因此，本发明在传统的轮询方式上进行优化，在缓冲区的读取端读端若有消息则移植读取，

直到缓冲区空，即将读取操作放入至一个工作队列等候，并在指定的时间间隔后重启读取操作，这时缓冲区若有新的消息则开始读消息；若此时缓冲区再空，则再次将读取操作放入工作队列中等待指定的时间，这样就避免了持续读取操作带来的 CPU 性能开销。

[0058] 实现内核态与用户态之间的通讯需要将消息与上层网络协议栈衔接起来，通过虚拟网络驱动能够实现内核网络协议栈与用户态网络驱动的消息交互：虚拟网络驱动可按网络驱动模型编写，实现消息的收发。

[0059] 虚拟字符设备驱动实现了对系统控制，包括建立共享内存，用户态网络驱动可通过操作该虚拟字符设备驱动对应的设备文件对系统进行控制，如 ioctl 以及 mmap，建立的共享内存申请分配也是通过该字符设备驱动的 mmap 方法实现。

[0060] 本发明能够在网络驱动下实现内核网络功能，基于传统套接字的网络应用程序能够在用户态网络驱动下执行，同时保证很高的性能；本发明将用户态的网络接口与内核网络协议栈进行衔接，实现了传统网络应用程序也能在用户态网络驱动下使用，并且优化了网络通讯的性能。

附图说明

[0061] 图 1 是本发明网络交互体系的结构示意图；

[0062] 图 2 是本发明网络交互方法的流程示意图；

[0063] 图 3 是本发明网络交互方法中消息接收流程示意图；

[0064] 图 4 是本发明网络交互方法中消息发送流程示意图。

具体实施方式

[0065] 下面结合附图详细介绍本发明的具体实施方式。

[0066] 一种基于嵌入式系统的网络交互体系，如图 1 所示，包括系统用户态中的网络接口、网络驱动、第一 IO 接口、第二 IO 接口、驱动接口以及用户态网络协议栈，系统内核态中的虚拟字符设备驱动、虚拟网络驱动以及内核网络协议栈，共享内存。

[0067] 所述的用户态网络协议栈以及内核网络协议栈均包括多个特定的网络协议层。

[0068] 由于绝大部分的网络应用程序（下称一类网络应用程序）支持用户态网络协议栈，且实现于用户态的网络驱动能直接访问用户态网络协议栈，因此，所述的网络交互体系在一类网络应用程序的网络交互中能够在用户态直接实现网络应用程序的网络交互过程，具体地，网络交互体系包括：

[0069] 第一 IO 接口，用于从一类网络应用程序接收消息以及向一类网络应用程序传递消息；所述的消息携带由用户态网络协议栈的特定网络协议层执行的通用命令；

[0070] 用户态网络协议栈，处理用于执行的通用命令，依据通用命令生成由网络驱动执行的特定命令；

[0071] 网络驱动，执行用户态网络协议栈的特定命令，访问用户态网络协议栈并实现网络接口与网络协议栈之间的消息交互。

[0072] 在用户态的网络应用程序中，存在部分的网络应用程序（下称二类网络应用程序）仅支持内核网络协议栈，且实现于用户态的网络驱动不能直接访问内核网络协议栈，因此，所述的网络交互体系在二类网络应用程序的网络交互中须在内核态实现网络应用程

序的网络交互过程。具体地，网络交互体系包括：

[0073] 第二 IO 接口，用于从二类网络应用程序接收消息以及向二类网络应用程序传递消息；所述的消息携带由内核网络协议栈的特定网络协议层执行的通用命令；

[0074] 内核网络协议栈，处理用于执行的通用命令，依据通用命令生成由虚拟网络驱动执行的特定命令；

[0075] 虚拟网络驱动，执行内核网络协议栈的特定命令，访问内核网络协议栈并实现共享内存与内核网络协议栈之间的消息交互；

[0076] 驱动接口，访问共享内存并实现网络驱动与共享内存之间的消息交互；以及，

[0077] 虚拟字符设备驱动，用于向驱动接口提供内核态的控制接口以及将所述的共享内存映射至用户态的地址空间。

[0078] 所述的共享内存由两块内存构成，一块用于作为发送缓冲区，另一块用于作为接收缓冲区。两块缓冲区均被组织成循环缓冲区，并采用无锁的方式进行读写，这样能够避免使用互斥锁带来的开销。以共享内存的方式实现内核态与用户态的通信，能够大大减少系统调用的次数与数据的拷贝次数。

[0079] 所述的第二 IO 接口为套接字接口。为了实现网络应用程序与内核网络协议栈进行消息交互，创建一个套接字接口，并设置套接字接口的设置通讯类型（如 TCP、UDP、广播等）。在网络应用程序与内核网络协议栈进行消息交互时，调用 send、recv 等用户态套接字库函数进行消息交互。

[0080] 在本实施例中，由于用户态的网络驱动无法对网络交互体系内核态的部分进行控制，也无法直接对共享内存进行访问，因此本体系首先设置了驱动接口，能够实现网络驱动对共享内存中数据的读写。

[0081] 更为具体地，由于网络驱动需对体系数据传递的控制，驱动接口具体需实现如下功能：

[0082] 1、为网络驱动提供操作共享内存的方法，包括通过该驱动接口执行 vn_put 函数以及 vn_get 函数，其中，vn_put 函数实现了将数据写入接收缓冲区的功能，vn_get 函数实现了将数据从发送缓冲区读取的功能；

[0083] 2、系统初始化，即在网络驱动运行时，对内核态中虚拟网络驱动以及虚拟字符设备驱动的程序加载；包括通过该驱动接口执行 vn_init 函数，其中，vn_init 函数实现了系统的初始化，配置共享内存的在用户空间的 mac 地址，即将共享内存映射至用户态的地址，并发送开始收发命令字。

[0084] 3、封装屏蔽共享内存的内部细节。

[0085] 由于驱动接口无法直接调用内核态中的文件，为了实现驱动接口的上述功能并进一步完善上述功能，本体系还于内核态设置了虚拟字符设备驱动。

[0086] 虚拟字符设备驱动用于向驱动接口提供内核态的控制接口以及将所述的共享内存映射至用户态的地址空间；通过驱动接口以及虚拟字符设备驱动，实现于用户态的网络驱动能够对内核态中相应的文件进行读写特定操作并能够在内核态中触发执行表 1 所示函数功能。

[0087] 表 1 函数及其执行功能

[0088]

函数名	功能
vn_cdev_open(open)	赋值私用数据指针
vn_cdev_exit(release)	无
vn_cdev_ioctl(ioctl)	提供 ioctl 功能
vn_cdev_mmap(mmap)	提供内存映射功能

[0089] 其中，对驱动接口提供的控制接口，包括 ioctl 接口以及 mmap 接口，分别实现对 vn_cdev_ioctl 函数的调用以及 vn_cdev_mmap 函数的调用；实现的功能如下：

[0090] 一、vn_cdev_ioctl 函数，在此处实现了 3 个控制命令字，即：

[0091] VN_IOC_START：开始数据收发；

[0092] VN_IOC_STOP：暂停数据收发；以及，

[0093] VN_IOC_MAPDIR：选择 mmap 函数指定的内存块。

[0094] 二、mmap 函数，调用 remap_pfn_range 函数将之前初始化时申请的内存空间映射到用户态的地址空间中，从而实现共享内存。

[0095] 虚拟网络驱动主要负责将内核网络协议与共享内存衔接起来，执行内核网络协议栈的特定命令，访问内核网络协议栈并实现共享内存与内核网络协议栈之间的消息交互，使得网络驱动能够间接地访问内核网络协议栈，实现一个虚拟的网络接口。通过用户态网络配置工具 ifconfig、vconfig 等可以对该虚拟网络驱动进行配置，该虚拟驱动按照传统网络设备模型编写，具体实现函数如表 2 所示。

[0096] 表 2 虚拟网络驱动的实现函数

[0097]

函数名	功能
vn_ndev_init	初始化网络设备驱动
vn_ndev_exit	释放资源
vn_ndev_open (open)	打开设备，开始收发数据
vn_ndev_stop (stop)	停止数据收发
vn_ndev_setmacaddr (set_mac_address)	设置 MAC 地址
vn_ndev_tx (hard_start_xmit)	发送数据
vn_ndev_poll	轮询
vn_ndev_do_poll	处理轮询任务

[0098] 一种利用了上述网络交互体系的网络交互方法，实现了网络应用程序与网络之间的消息交互过程，如图 2 所示，包括消息的接收以及消息的发送，包括以下步骤：

[0099] (1) 对网络交互体系的初始化，加载虚拟络驱动以及虚拟字符设备驱动，建立共享内存；

[0100] (2) 当所述的消息携带由用户态网络协议栈执行的通用命令，消息的接收或发送包括步骤：

[0101] 在网络接口或第一 IO 接口中载入消息；

[0102] 读取所述的消息；

[0103] 基于所述的消息，对网络驱动以及用户态网络协议栈发起操作；

[0104] 向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第一 IO 接口中载入响应；以及，

[0105] 读取所述的响应；

[0106] (3) 当所述的消息携带由内核网络协议栈执行的通用命令，消息的接收或发送包括步骤：

[0107] 在网络接口或第二 IO 接口中载入消息；

[0108] 读取所述的消息；

[0109] 基于所述的消息，对网络驱动、驱动接口、共享内存、虚拟网络驱动以及内核网络协议栈发起操作；

[0110] 向网络应用程序发送消息或从网络应用程序传送消息，并在完成操作时在网络接口或第二 IO 接口中载入响应；以及，

[0111] 读取所述的响应。

[0112] 进一步地，在步骤 (1) 中，所述的建立共享内存，包括步骤：

[0113] (1.1) 申请两段各 512 个页面大小的空间，计算出每个页的页框号，将这些页全设置成为保留页；

[0114] (1.2) 网络驱动通过驱动接口控制所述的虚拟字符设备驱动，将所述两段空间映射到用户态的地址空间。

[0115] 更为具体地，共享内存的建立即内存分配的过程进一步包括：

[0116] 对网络交互体系的初始化，调用 get_free_pages 申请两段各 512 个页面大小的空间，分别名为 rx_buff 和 tx_buff；

[0117] 调用 virt_to_page 计算出每个页的页框号，对每个页调用 SetPageReserved，将这些页全设置成为保留页；

[0118] 网络驱动通过驱动接口调用 map 函数，继而调用了虚拟字符设备驱动提供的 mmap 函数，将这两段内存映射到用户态的地址空间。

[0119] 进一步地，所述的共享内存包括发送缓冲区以及接收缓冲区，在步骤 (3) 中，

[0120] 所述的向网络应用程序发送消息，包括步骤：

[0121] 网络驱动调用驱动接口检查共享内存的接收缓冲区是否满，若满则丢包，不满则通过驱动接口将消息发送至接收缓冲区；以及，

[0122] 内核网络协议栈调用虚拟网络驱动轮询接收缓冲区，通过虚拟网络驱动读取接收缓冲区内的消息并与网络应用程序进行消息交互；

[0123] 所述的从网络应用程序传送消息，包括步骤：

[0124] 内核网络协议栈调用虚拟网络驱动检查共享内存的发送缓冲区是否满，若满则丢

包，不满则通过虚拟网络驱动将消息放入共享内存的发送缓冲区；以及，

[0125] 网络驱动调用驱动接口轮询发送缓冲区，通过驱动接口读取发送缓冲区内的消息并与网络接口进行消息交互。

[0126] 所述的消息在本实施例即为指定 MAC 地址或者广播包（即一般意义上的数据包）；

[0127] 向网络应用程序发送消息，即消息的接收，采用经优化的轮询方式，当接收缓冲区中有数据，虚拟网络驱动则不断地从接收缓冲区中取出数据，提交给内核网络协议栈，当接收缓冲区空时，将接收例程放入等待队列，等待指定时间后重启接收例程，具体可如图 3 所示，图 3 所述的数据接收流程中，等待的指定时间为 4ms。

[0128] 从网络应用程序传送消息，即消息的发送，进一步包括：在内核网络协议栈需要发送数据时，调用 vn_ndev_tx，该函数将检查发送缓冲区是否已满，不满则将数据拷贝到缓冲区中，满了则丢弃该数据包；当发送缓冲区中有数据，用户态网络驱动则不断地从发送缓冲区中取出数据发送，当接收缓冲区空时，将发送例程放入等待队列，等待指定时间后重启发送例程，具体可如图 4 所示，图 4 所述的数据发送流程中，等待的指定时间为 1ms。

[0129] 更进一步地，轮询的实现是通过将轮询任务放入到工作队列中实现的，包括步骤：

[0130] a、在共享内存上建立工作队列；

[0131] b、将轮询任务放入到工作队列；

[0132] c、检查共享内存的缓冲区是否为空，如果未空则一直读取消息；如果共享内存的缓冲区为空，则将轮询任务放入工作队列等待指定的时间。

[0133] 具体实现方式如下：

[0134] 首先，将轮询任务放入到工作队列；

[0135] 调用 create_singlethread_workqueue 建立一个工作队列 wq，创建一个任务 poll_work，通过 INIT_WORK 函数将该任务与轮询函数 vn_ndev_poll 绑定起来；最后，通过调用 queue_delayed_work 将该任务提交给工作队列 wq，该任务将在指定的 jiffies 之后执行。

[0136] 其次，检查接收缓冲区是否有数据包，如果有则一直读数据包，直到接收缓冲区空，并将轮询任务放入到工作队列等候；如果接收缓冲区为空，则将轮询任务放入工作队列等待指定的时间。

[0137] 在轮询函数 vn_ndev_poll 中，该函数首先检查接收缓冲区是否为空，不为空则调用 vn_ndev_do_poll 接收数据包，然后调用 queue_delayed_work 将自身任务 poll_work 提交给工作队列。这样轮询函数就能按一定的时间间隔不断的被执行，这样也就实现了轮询。

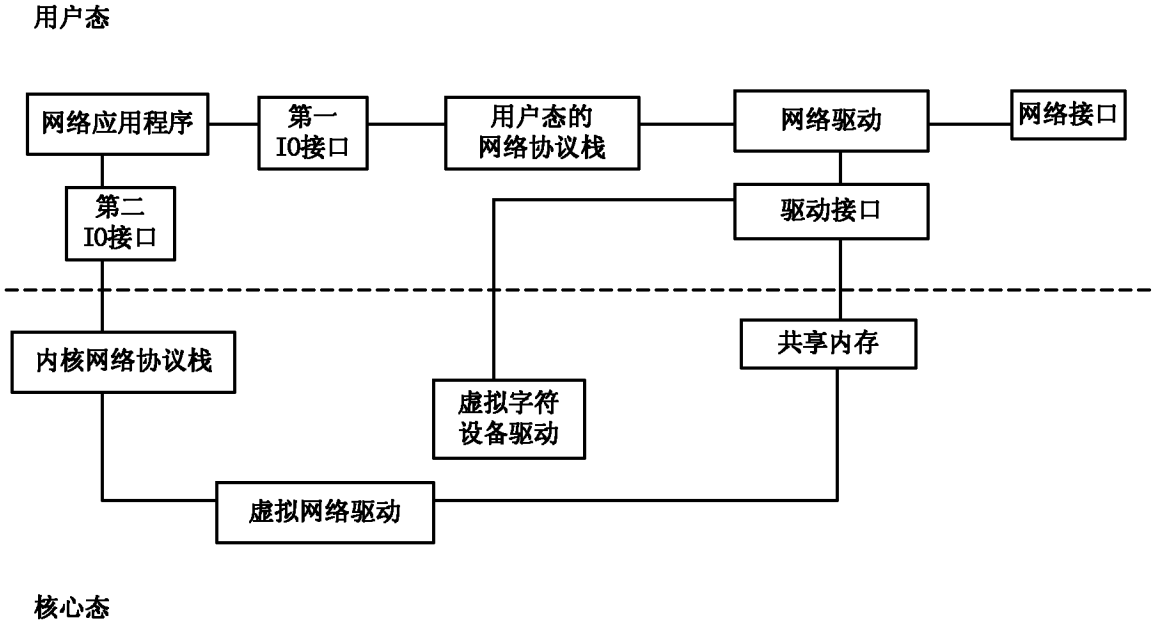

用户态
网络应用程序
第一
IO接口
用户态的
网络协议栈
网络驱动
网络接口
第二
IO接口
驱动接口
内核网络协议栈
虚拟字符
设备驱动
共享内存
虚拟网络驱动
核心态

图 1

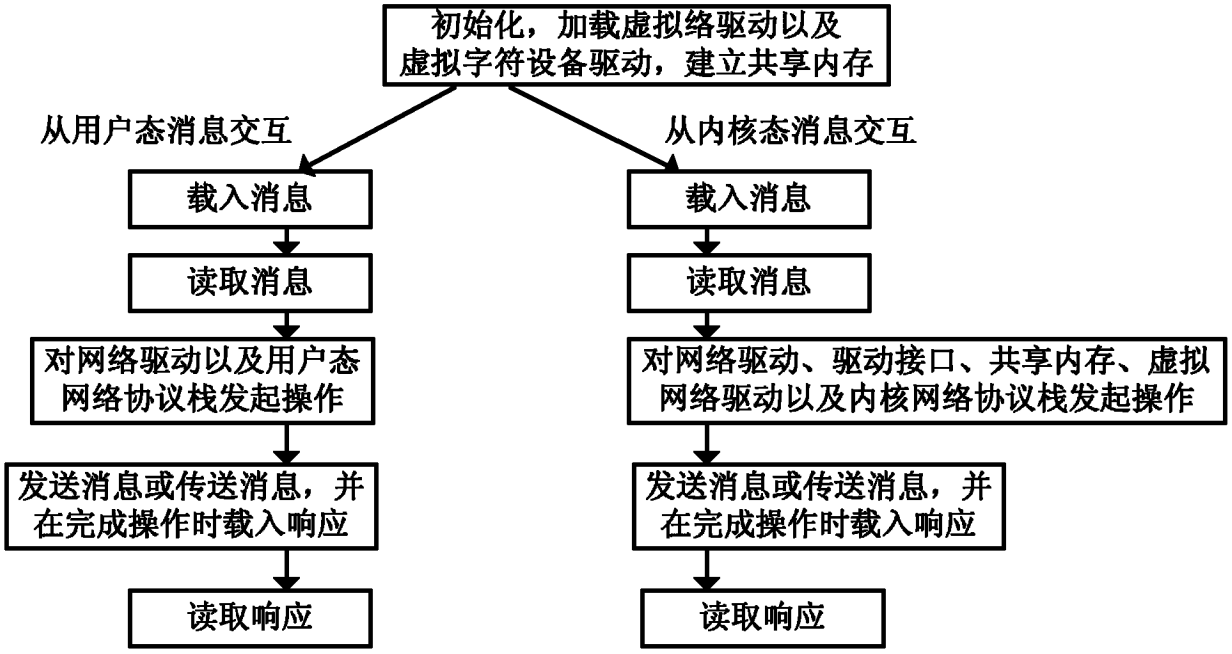

初始化，加载虚拟络驱动以及
虚拟字符设备驱动，建立共享内存
从用户态消息交互
从内核态消息交互
载入消息
读取消息
对网络驱动以及用户态
网络协议栈发起操作
发送消息或传送消息，并
在完成操作时载入响应
读取响应
载入消息
读取消息
对网络驱动、驱动接口、共享内存、虚拟
网络驱动以及内核网络协议栈发起操作
发送消息或传送消息，并
在完成操作时载入响应
读取响应

图 2

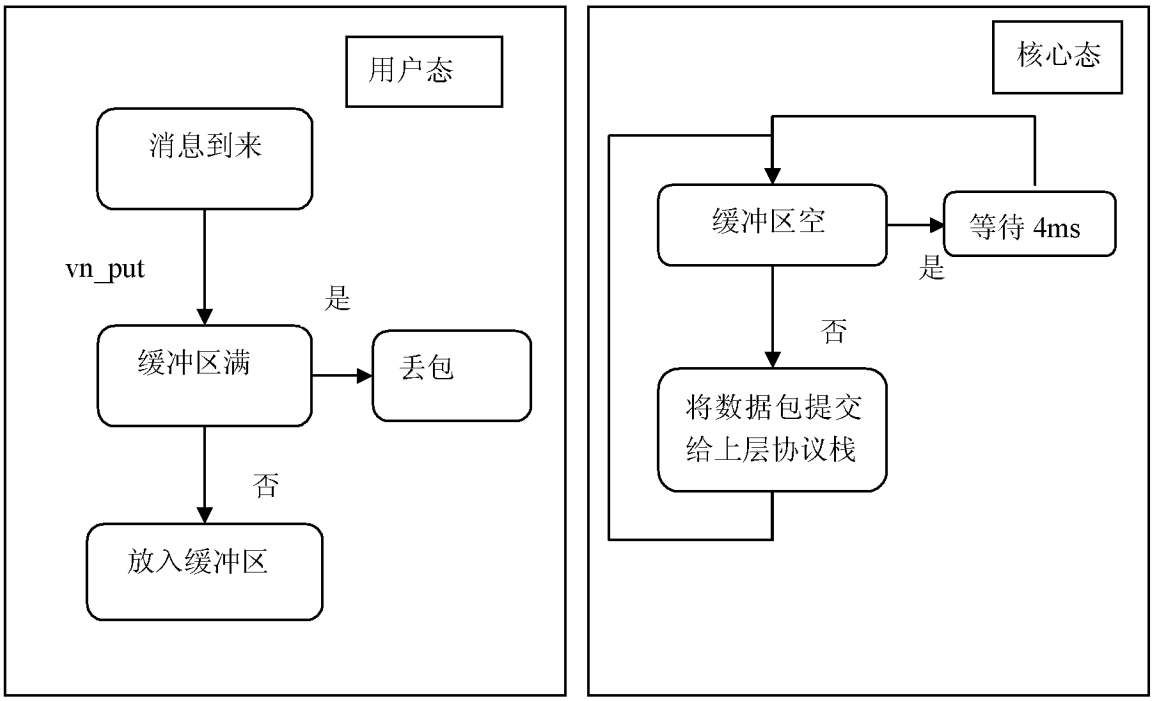

用户态
消息到来
vn_put
是
缓冲区满
丢包
否
放入缓冲区
核心态
缓冲区空
等待 4ms
是
否
将数据包提交
给上层协议栈

图 3

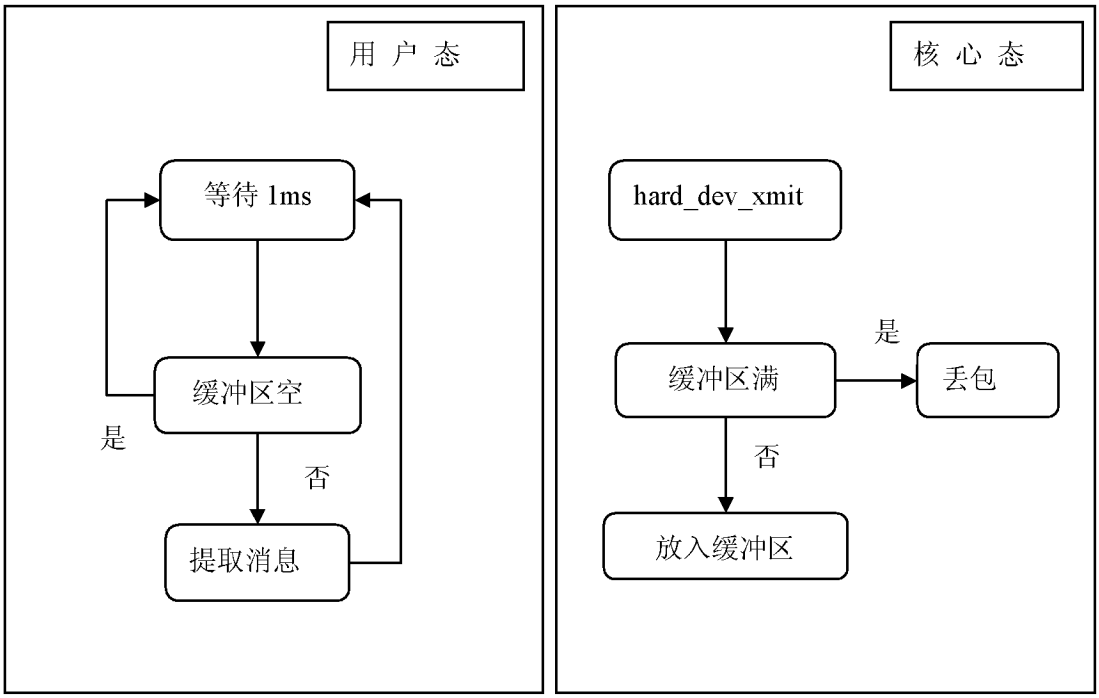

用 户 态
等待 1ms
缓冲区空
是
否
提取消息
核 心 态
hard_dev_xmit
是
缓冲区满
丢包
否
放入缓冲区

图 4