第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·网络与通信· 文章编号：1000—3428(2007)10—0111—03 文献标识码：A 中图分类号：TP393

一种新的TOE体系结构的研究与原型实现

刘天华[1,2]，朱宏峰[1,2]，谭振华[1]，常桂然[1]
(1. 东北大学信息科学与工程学院，沈阳 110004；2. 沈阳师范大学软件学院，沈阳 110034)

摘 要：将处理器从繁重的通信任务中解脱出来，解决通信系统中的瓶颈部分，适应高速通信网络。采用Linux平台，结合ML403开发版的功能进行分析与设计，编写或修改内核代码，硬件部分采用VHDL设计，用LKM机制加载调试。提出了一套基于ML403的实现TOE的体系结构与设计方法，搭建系统实验平台并得出部分仿真结果。实验初步成功表明了该方案设计方法的正确性和高效性，这将为提高整体通信系统性能、普及TOE网卡做出一定的贡献。

关键词：TOE；ML403；TCP/IP

Research and Prototype Implementation of New TOE Architecture

LIU Tianhua[1,2], ZHU Hongfeng[1,2], TAN Zhenhua[1], CHANG Guiran[1]
(1. School of Information Science and Engineering, Northeastern University, Shenyang 110004;
2. School of Software, Shenyang Normal University, Shenyang 110034)

【Abstract】In order to remove CPU from heavy services of communication and alleviate the bottlenecks of communication system to suit for the high speed network, this paper adopts Linux platform and combines ML403 development board, codes or modifies portion of kernel code and designs hardware section with VHDL. It finally tests with LKM mechanism. It puts forward a ML403-based solution for TOE architecture design, builds a system experiment platform with some simulation results. Initial success of test indicates that this solution is correct and effective, which will make contribution to the performance improvement of whole communication system and popularization of the TOE network card.

【Key words】TOE; ML403; TCP/IP

目前通信延迟主要分为两个部分：处理消息的延迟和网络延迟。现代高速互联网络已经达到很高的传输速度，因此通信中的瓶颈就从以前的互联网络转移到处理消息收发任务的软件上。对此问题主要有两个方向的解决方案：

(1)软件方向解决方案。找出软件优化中的对象，即主要的瓶颈部分是什么，然后针对操作系统内核和驱动程序中的软件代码进行优化。

(2)硬件方向解决方案。软件和硬件的实现功能相同，不同的是速度和价格，可以利用硬件的速度优势解决通信瓶颈。但硬件的设计和价格相对成本较高，本文针对此方法进行了设计与分析，结合各种可能的手段设计出一套成本较低、实现较方便、可扩展性好的TOE解决方案。

TOE(TCP/IP Offload Engine)：TCP/IP卸载引擎，所谓卸载是指将 CPU 上的计算或处理转移到专门的处理单元上进行。与传统方式对比，TCP/IP卸载引擎引入了一种新的网络接口体系结构。它将TCP/IP协议栈的处理工作从服务器上卸载下来，交给网卡来处理。在具有CPU卸载引擎的网卡上有专门的处理器或硬件来完成 TCP/IP 协议处理，简化了整个TCP/IP的处理路径，从而减轻了CPU和服务器I/O系统的处理负担，消除了服务器的网络瓶颈。

1 相关工作

对于TCP/IP软件解决方向，文献[1,2]分析其主要的协议开销部分，文献[3]提出了对应方案并在一定程度上减少了协议开销。随着问题的不断深入和硬件成本的不断降低，以TOE为代表的硬件解决方向被提上日程，很多大学及公司研发部门开始关心并进行研究，实现了TOE中的部分卸载[4]。相对于已经实现的方案，本文主要关注设计的可扩展性、适应性以及可配置性3大方面：即注重未来扩展、价格以及可根据需要进行配置，经过论证可行。

2 TOE体系结构

TOE是架设在服务器上的，需要和操作系统进行结合。由于操作系统本身自带TCP/IP协议栈软件模块，如何使TOE和原有协议栈结合是设计过程中必须要考虑的问题。TOE的体系结构如图1所示。

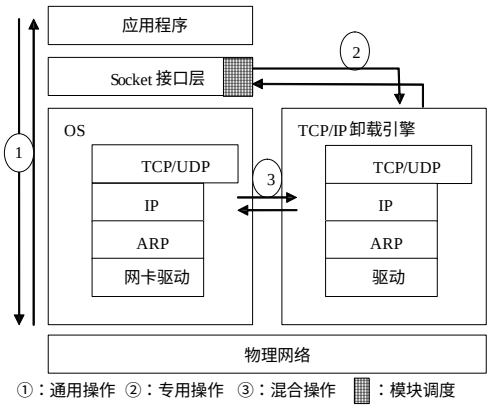

图1 TCP/IP卸载引擎体系结构

基金项目：辽宁省自然科学基金资助项目(20042042)；辽宁省教育厅科研基金资助项目(05L420)

作者简介：刘天华(1966－)，男，博士生，主研方向：计算机网络及协议；朱宏峰，博士生；谭振华，硕士生；常桂然，博士、博导

收稿日期：2006-05-25 **E-mail：**liutianhua@sina.com

图 1 所示的体系结构是根据操作系统的网络体系结构基础上进行扩展而成。通过图中的几个箭头来说明 TOE 设计中的几个关键问题。

(1)在①处的箭头代表正常的网络交互行为，即该体系结构可以在普通网卡的基础上进行正常运作，这直接保证系统的正常运转。说明该结构具有通用性。

(2)在②处的箭头表明，系统调度直接转向卸载引擎。说明 TCP/IP 协议栈处理流程全部由卸载引擎来处理，操作系统可以通过串口模块转到卸载引擎，从而释放 CPU 资源。这种特性说明了该结构具有专用性。

(3)在③处的箭头表明了另外一种情况，即将协议栈的部分操作装载在卸载引擎中，而部分操作仍然保留在 OS 的协议栈中，利于 TCP/IP 协议栈的进一步扩展。这说明了该体系结构具有可扩展性。

图中两种协议栈的处理流程在原则上是一致的。其中左侧的协议栈处理过程基于 OS 和 CPU，右侧的协议栈是基于卸载引擎的。二者最大区别在于操作系统里的网卡驱动是针对普通的以太网卡，而卸载引擎里的驱动则是针对实现了该引擎的专用网卡。

3 TOE 解决方案设计

在分解 TCP 协议栈的基础上用 ASIC 芯片的方案完全实现或部分实现 TOE，并进行对 TOE 优化的研究。根据 TOE 体系结构，总结出在设计 TOE 专用网卡的时候有 3 个关键的方面必须完成，即协议栈的处理、ML403 的规划以及与操作系统的结合。

3.1 TOE 中 TCP/IP 协议栈的设计

需要一种和操作系统分离开来的 TCP/IP 协议栈实现到 TOE 专用网卡里。协议栈要求至少有 TCP、UDP、IP、ARP 4 个模块。由于用硬件实现，因此这些协议最终必须用硬件描述语言来描述，实现在 FPGA 里。采用两种方法来实现协议栈：

(1)编写协议栈，对协议栈进行精简分析后形成，这种方法的优点可以使得整个设计过程非常清晰和可控；缺点是耗时教长，且很可能由于测试量不够而导致整个协议栈存在很多的错误。

(2)使用开源的、精简的、稳定的协议栈，其最大的特点是稳定性好，但其实现内容不一定能满足要求。

瑞士计算机科学院开发一套精简协议栈LWIP[5]结合这两种方法，包括了TCP、UDP、ICMP、IP等基本协议，本文对LWIP进行进一步的修改，同时增加ARP模块。

3.2 硬件开发板 ML403 的规划设计

由于 TOE 最终是实现成 TOE 专用网卡，因此必须有硬件开发板来做电路设计和规划。硬件开发板完成硬件电路的设计和实现，选择使用 ML403 来做这个过程的设计和分析。

FPGA 里实现 ARP、IP、ICMP、TCP 模块。Compact Flash 里主要存放软件代码以及需要保存的参数。DDR SDRAM 用来做系统运行时的代码和数据存储。网络接口芯片 RJ45 用来做网络连接，已实现了 MAC 层的主要功能。所有需要 CPU 总线处理的数据都由 PowerPC 来处理。与网络进行交互通过 RJ45 口，而与主机进行交互是通过 RS232 串行通信口。

3.3 服务器与 ML403 之间的串口编程

服务器与 ML403 之间是通过 RS232 串口来进行连接，因此在服务器端需要对串口进行编程。正常情况下，驱动的主要作用是将网卡获取的数据通过软中断进行系统调用，从而进入协议栈的处理。在 TOE 方案里，协议栈主要由硬件实现，因此服务器与 ML403 之间的连接必须是对 RS232 串行通信端口传来的数据进行汇总，再通过中断进行系统调用，执行其他的操作。

要访问一个串口，在 Linux 中只需打开相应的设备文件，然后向这个文件读写数据就可以完成数据的接收和发送。因此，实现服务器与 ML403 之间进行通信功能的串口编程可以放置在一个专门的 Linux 串口模块程序(SerialPortModule)里，该模块初始状态是在用户态下，通过 Linux 的 LKM 机制(loadable kernel module)加载将其切换到内核态。

3.4 基于 ML403 的 TOE 解决方案

综合上面的各个模块将基于 ML403 的 TOE 开发的解决方案规划出来。图 2 是基于 ML403 的解决方案。

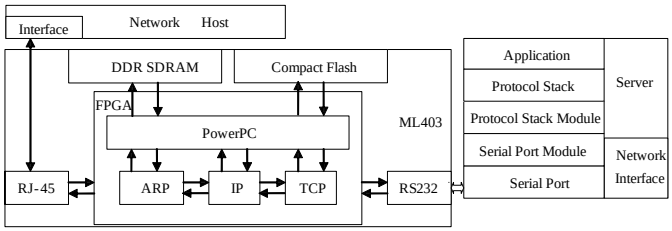

图 2 基于 ML403 的 TOE 解决方案

从数据流程的角度来分析网络主机通过网线和 ML403 的 RJ-45 连接，服务器通过串行通信口与 ML403 的 RS232 串口连接。当服务器发送数据时，用户数据正常从 Socket 层往下传输。在协议栈已经放置了模块钩子，当用户模块已经通过命令装载到内核中时，hook 生效。那么用户数据将如本文所设计的通过串口到达 ML403。ML403 在 PowerPC 的总调度下，通过 FPGA 的各个模块进行数据的传递，直到通过 RJ45 口发送到网络中。

当服务器接收数据时，帧数据到达 ML403 的 RJ45 口，芯片中的 MAC 层会将数据进行初步的分析，并将数据传递给上层协议如 IP，此时进入 FPGA，最后通过串口到达主机。串口模块 SerialPortModule 将数据转给协议栈模块 TOE SocketModule，TOESocketModule 再将数据交给 Socket 层。本方案主要用来做开发和测试，针对 FPGA 的电路设计部分以及如何和协议栈连接的问题。

4 实现模块划分

在模块设计部分，将整个流程划分为 4 大模块：主机部分分为主机应用部分和 TOESocketModule 部分(为描述方便，将 SerialPortModule 作为 TOESocketModule 的一部分)；ML403 分为 ML403 内核模块和 ML403 应用模块。其具体功能如下：

(1)主机应用。发送原始命令及各种参数信息，如地址、数据长度等信息，是数据传输的发起者和结束者。

(2)主机 TOESocketModule。为改写的 Linux 驱动模块，使用 LKM 机制加载后实现。主要功能是在主机与 ML403 之间传递各种信息、接口控制及传输信息。

(3)ML403 内核。将主机传递的信息进行加工，同时控制 ML403 应用模块。信息的接收采用 DMA 控制，当信息处理完毕后返回一个确认时，它起到连接的作用。实现方法是 ML403 自带的驱动和自己编写程序相结合。

(4)ML403 应用。此模块使用 FPGA 实现的纯硬件处理 TCP/IP 的逻辑电路，速度快，与其配合的缓存也是速度相对较快的静态存储器，其主要功能就是在 ML403 内核模块的控

制下，将缓存中排队的数据进行 TCP/IP 的快速处理，当连接建立时其发出器件准备好的信号。

按照这 4 个模块分别详细说明，流程分为 TOE 连接和 TOE 数据传输来阐述。

图 3、图 4 显示了 TOE 机制如何处理典型的 server/client 模式。从 socket()传递的信息可分为两部分：TCP/IP 连接及其数据传输。下面分别叙述其流程。

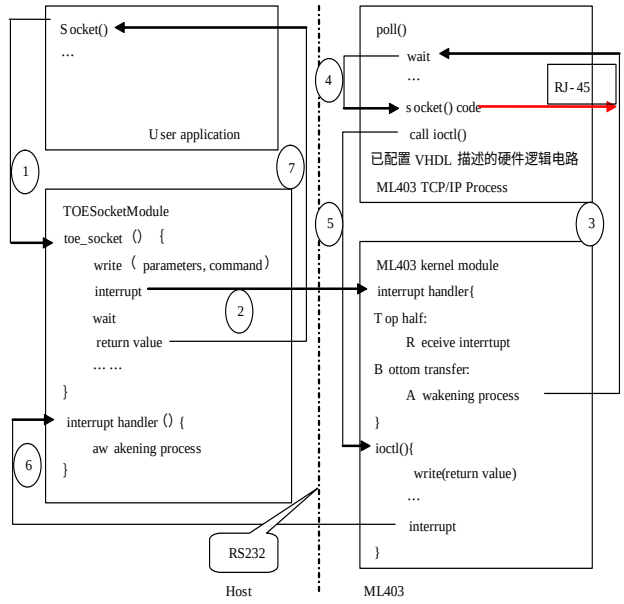

图 3 TOE 进程处理中的连接

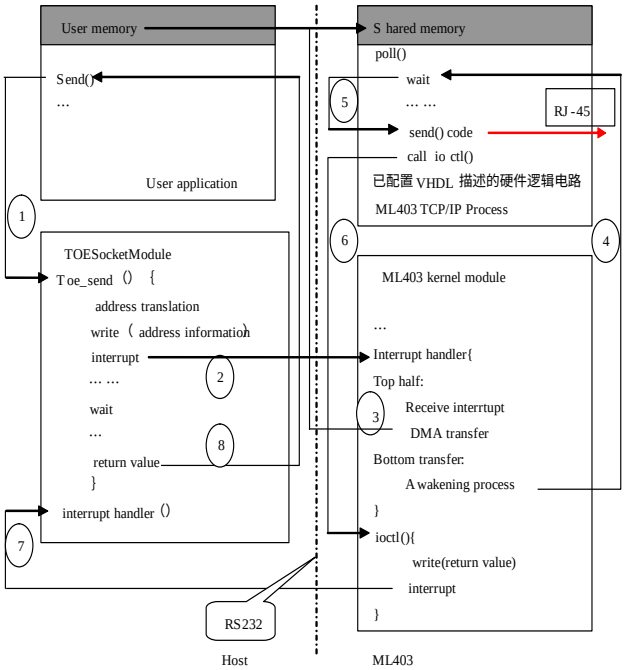

图 4 TOE 进程处理中的数据传输

TCP/IP 连接 如图 3 所示，当主机的用户程序调用 socket()函数时，则 socket()层就会调用 TOESocketModule 模块中的 toe_socket()函数，它在我们改写的驱动程序中实现，运用 LKM 机制加载①，此步需要判断是串口还是网卡，调用相应的驱动处理机制。toe_socket()函数负责将用户参数和命令传递给 ML403，然后驱动程序中断内核模块②，同时 toe_socket()函数等待处理信息的到来。接收到中断后，ML403 内核模块激活 ML403 应用部分③。ML403 应用根据命令执行相关代码④。执行完 socket()代码后，ML403 应用程序通过 ioctl()函数返回一个相关信息的值⑤，此值表示是否成功连接。ML403 内核将此值返回给驱动程序同时中断驱动⑥。中断重新激活 toe_socket()函数。最后 toe_socket()函数将值返回给主机用户应用程序⑦，至此连接完毕。

TCP/IP 数据传输：图 4 显示了发送数据的过程。当用户应用调用 send()时①，驱动程序不会从用户缓存将数据拷贝到相应的缓存中，驱动程序只计算其物理地址及所用缓存大小。然后驱动就将地址和所需缓存大小成对地交付给 ML403 中的内核处理模块，此时 ML403 内核中产生一个中断②，这时需要判断是串口还是网卡，调用相应的驱动处理机制。ML403 内核将数据地址信息装载到 DMA 控制器，DMA 控制器将用户空间的数据传送到 ML403 的存储器中进行排队，为 ML403 中的硬件逻辑处理 TCP/IP 协议栈提供数据③。然后 ML403 内核模块激活其用 FPGA 实现的硬件处理 TCP/IP 模块④。接着硬件处理模块进行数据的协议处理，然后将处理后的数据发送到缓存排队等待 send()函数处理发送⑤。发送完毕后将返回一个值来确认，其余过程如图 3。

图 3、图 4 显示了从建立连接到发送数据的循环过程，接收数据流程类似。这个过程中利用减少数据复制技术：在 ML403 内核模块中设置有 DMA 传输机制，直接将主机用户空间数据通过计算物理地址和所需缓存大小放置到 ML403 缓存中排队，等待处理。否则至少要经过 3 次数据拷贝：用户缓存－>驱动缓存－>ML403 内核缓存－>ML403 应用缓存。

5 结论及展望

目前为止，按照上述方案已经将平台搭建成功，同时完成 ARP 和 IP 的功能仿真。设计过程从略，仿真部分结果如图 5、图 6。

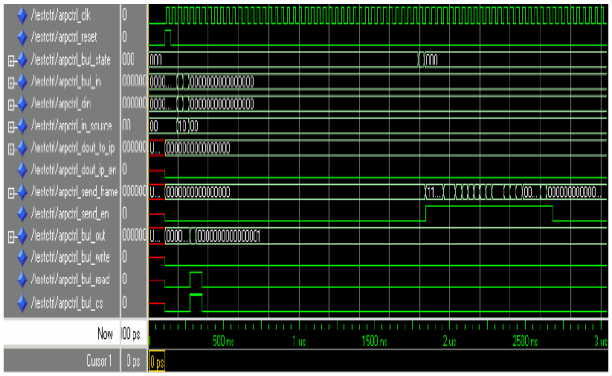

图 5 ARP 控制模块功能仿真（查询与发送报文）

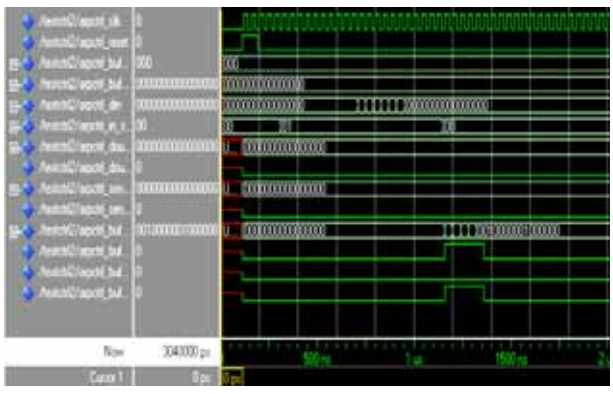

图 6 ARP 控制模块功能仿真（数据帧接收）

(下转第 122 页)