RTLinux 下基于半轮询驱动的用户级报文传输机制*

田志宏+, 方滨兴, 云晓春

(哈尔滨工业大学 计算机网络与信息安全技术研究中心,黑龙江 哈尔滨 150001)

User-Level Message Passing Mechanism Based on Semi-Polling Driven in RTLinux

TIAN Zhi-Hong+, FANG Bin-Xing, YUN Xiao-Chun

(Research Center of Computer Network and Information Security Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: Phn: +86-451-86418273, E-mail: tianzhihong@hit.edu.cn, http://pact518.hit.edu.cn

Received 2003-09-22; Accepted 2004-03-02

Tian ZH, Fang BX, Yun XC. User-Level message passing mechanism based on semi-polling driven in RTLinux. *Journal of Software*, 2004,15(6):834~841.

http://www.jos.org.cn/1000-9825/15/834.htm

Abstract: Software overhead in interconnection network communication has currently become the bottleneck of a cluster system. To reduce it, a user-level communication software UMPS based on real-time OS RTLinux is designed and implemented, which is comfortable with VIA. A new concept of semi-polling driven is presented. With the semi-polling driven mechanism, the interrupts frequency is lowered and the processing performance for short message is significantly ameliorated. By means of the address translation and buffer managing algorithm based on the resource-mapping graph, applications bypass OS and interact with network interface directly using asynchronous DMA. So the overhead and latency in communication are efficiently reduced. Experimental results indicate that the throughputs of UMPS for 64 byte and 1500 byte messages are 394 Mbps and 895 Mbps respectively, and the performance of UMPS surpasses that of other mechanisms.

Key words: RTLinux; asynchronous DMA; latency; semi-polling driven; interrupt

摘 要: 网络通信软件的处理开销已成为影响机群系统的性能瓶颈,为了提高机群系统的网络性能,在实时操作系统 RTLinux 下,设计并实现了一个符合 VIA 规范的用户级通信软件 UMPS,提出了半轮询驱动的概念,利用半轮询驱动机制降低了系统中断频率,明显提高了短报文的处理能力.通过更为高效的地址翻译和基于资源映射图的缓冲区管理算法,应用程序旁路操作系统,依靠异步 DMA 直接与通信设备进行交互,有效地降低了网络通信的延迟与开销.通过性能的分析比较表明,UMPS 接收 64 byte 与 1500 byte 的报文时吞吐量分别达到 394 Mbps 和 895 Mbps,与现有的报文传输机制相比,UMPS 的性能有了较为显著的提高.

* Supported by the National High-Tech Research and Development Plan of China under Grant Nos.863-104-02-02, 2002AA142020 (国家高技术研究发展计划(863))

作者简介: 田志宏(1978－),男,黑龙江哈尔滨人,博士,主要研究领域为计算机网络与信息安全;方滨兴(1960－),男,教授,博士生导师,主要研究领域为信息安全,计算机网络,并行计算;云晓春(1971－),男,博士,教授,博士生导师,主要研究领域为计算机网络,信息安全.

关键词: RTLinux;异步 DMA;延迟;半轮询驱动;中断
中图法分类号: TP316 文献标识码: A

随着计算机网络的迅猛发展以及一些高速度、基于交换的诸如 Fibre Channel,High-Performance Parallel Interface(Hippi)和 Asynchronous Transfer Mode(ATM)等网络技术的出现,基于网络并行计算的机群系统以其高效率、低消耗等优点受到越来越多的重视[1].在机群系统中,各节点通过相互之间的消息传递来实现数据的交换和同步,共同协作完成统一的任务,节点间通信速度的高低成为影响计算性能的关键因素.虽然高速局域网络技术的迅速提高,提供了高带宽、低差错的物理链路,使得通信系统的性能得到了大幅度的提高,已经达到 Gb 级带宽,微秒级延迟的水平.但相对于计算元件,通信系统的发展依然滞后,例如,在每秒 G 比特的网络流量下,单节点机最大可以处理的数据量却仅为 110Mbps 左右,这种原始带宽和被传输到应用层的最大吞吐量之间巨大差异的存在,造成了用户在升级网络设备的同时却没有得到相应的计算性能的提高[2].其主要原因在于,常用的操作系统如 Linux 等,出于通用性的考虑,在协议栈的实现上效率往往很低,使得在关键传输路径上软件开销占整个报文传递延迟的绝大多数.由此可见,局域网中的瓶颈从原来的物理信道转移到系统软件协议处理上[3].设法降低通信开销对于充分利用最新计算资源、解决挑战性的技术问题都有重要意义.

由此,本文提出了在实时操作系统 RTLinux 下,基于半轮询驱动的用户级报文传输机制——UMPS(user-level message passing based on semi-polling driven),该传输机制提高了报文获取进程的优先级,降低了中断的频率,并旁路操作系统(去除报文传输关键路径中操作系统的干预),从而极大地提高了系统报文的峰值吞吐量,有效地减少了网络通信的延迟.

本文第 1 节分析了相关的研究工作及系统的构建目标.第 2 节描述系统的整体结构与实现.第 3 节详细讨论了所采用的各项优化技术.实验测试结果以及性能分析将在第 4 节给出.最后是总结.

1 相关的工作与目标

在机群系统中,通信速度的高低是影响计算性能的关键因素,如何利用现有高速网络的成熟技术,充分发挥底层网络的物理带宽,成为机群能否实现高效机间通信的首要问题.

一直以来,端节点的报文获取大多采用基于传统TCP/IP协议栈的函数库Libpcap[4],但操作系统中低效的网络协议构架导致报文传输过程中存在频繁的系统调用、数据拷贝以及系统上下文切换开销,因此,其低下的通信性能无法适应大规模网络环境的需求.

基于内核可加载模块的 KLMP[5]绕过系统的 TCP/IP 协议栈中协议层和套接字层,减少了系统调用和内存拷贝的次数,但是其本质的用户区——内核分级结构并没有完全将操作系统从报文传输的关键路径中去除,性能瓶颈依然存在.

为了避免报文在主机传递路径中的冗余开销[6],消除操作系统的性能瓶颈,VMMC[7],U-Net/MM[8]和 MMUC[9]采用了用户区——设备扁平结构的用户级网络通信构架,完全旁路操作系统.出于对通用性的考虑,VMMC 和 U-Net/MM 通过网卡缓存一部分动态锁定的用户缓冲区物理地址,并借助操作系统内核虚存管理系统完成缓存未命中时的额外操作,在增加了网卡固件复杂度的同时还产生了由频繁的锁定内存和请求调页操作带来的巨额开销;而 MMUC 则通过通信区在系统核心与用户态之间高效地进行数据交换,实现了真正的零拷贝,但其纯中断驱动的报文接收机制使得短报文的处理性能效果不佳.

为了避免短报文处理过程中频繁中断所产生的延迟,PM 系统[10]采用轮询来驱动报文的接收,机制虽然简单,但是容易造成较低报文输入速率下 CPU 的巨额开销.

综合考虑上述不足,报文传输机制 UMPS 具有以上框架所不具备的优点:1) 可扩展的、模块化的系统架构;2) 符合虚拟接口结构(virtual interface architecture,简称 VIA)规范,平衡 VIA 各层次实现的功能,缩短通信关键路径,实现通信性能的最大化;3) 基于状态的调度方式在中断与轮询之间作了很好的平衡,降低系统产生中断的频率,显著提高了短报文的处理性能;4) 简单、高效的地址翻译机制,消除了频繁的锁定内存和请求调页操

作,减少内存地址虚实转换开销,降低网卡固件复杂度;5) 基于资源映射图的缓冲区管理算法,提供了一个轻量级的报文缓存机制,在降低存储开销的同时提高了内存空间的利用率.

我们要达到的目标是:

- 降低系统的中断频率;
- 低延迟、低开销和高带宽的短报文处理性能;
- 高效的地址翻译及缓冲区管理机制.

2 UMPS 的结构与实现

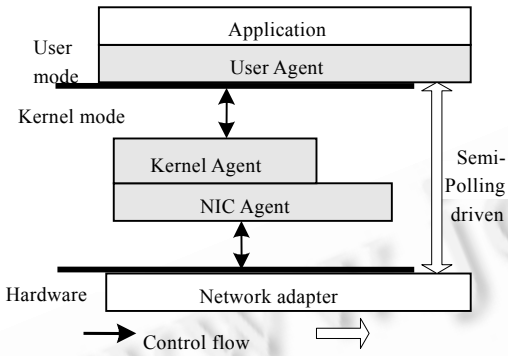

Fig.1 Function structure of UMPS

图 1 UMPS 的功能结构图

UMPS 的总体框架如图 1 所示,主要包括 3 个组件:User Agent(UA),Kernel Agent(KA)和 NIC Agent(NA).其中 UA 位于系统的用户态,是一系列用户层函数库,为应用程序提供直接访问网络设备的编程接口;其他两部分位于系统核心态,内核逻辑设备 KA 维护用户空间的虚拟地址——物理地址转换和缓冲区管理;专门编写的驱动程序 NA 作为软件和硬件之间的抽象层,负责网卡硬件的初始化、与 KA 交互获取 DMA 操作所需的物理地址,以及在半轮询机制的控制下利用异步 DMA 进行高效的用户级报文传输.

表 1 为 UA 所提供的 API 函数,通过调用下列 API,应用程序得以灵活、方便地访问底层通信网络,进行实时、高效的数据传输.

Table 1 Description of application programming interface

表 1 应用编程接口描述

API	Description
UMPS_Initialize	Register network interface, complete initialization before communication, and allocate resource including buffer area, port etc
UMPS_Release	After communication, release all the resources by this function
UMPS_Loop	Take out data repeatedly from buffer and send them to Callback function for processing
UMPS_ GetDataBlock	Get a packet from user buffer area
UMPS_ FreeDataBlock	Free the data buffer that has been processed
UMPS_ SendBuff	Send packets by this function
UMPS_ ShowState	Return information of current states to user, e.g. number of receiving packets

UMPS 包括初始化和报文传输两个阶段,与 VIA 规范基本相符;3 个代理之间层次分明,各自拥有完善的独立接口,保证了 UMPS 的模块化和高可扩展性;UA 提供的编程接口屏蔽了底层硬件的实现细节,使得 UMPS 框架可以直接在一些通用硬件上工作,具有良好的平台无关性.

3 UMPS 的性能优化技术

3.1 基于状态的半轮询驱动机制

中断的代价是昂贵的,Pentium Pro 的 CPU 派发中断到内核中断处理程序大约需要 10μs,在网络数据流量巨大的情况下,在中断级别上处理网络请求的操作系统将自陷在中断响应这一环节,产生所谓的 receive livelock[11],从而导致报文吞吐率的急剧下降.若带宽占用率相同,在全部传输短报文(长度<128 byte)的极端情况下,由于中断最为频繁,系统的峰值处理能力将达到最低限度.

为了解决中断驱动模式带来的负面影响,PM 系统采用完全轮询驱动的模式,有效地提高了短报文的处理能力,但是一旦报文频率远小于轮询频率,过多的轮询就会加重 CPU 的负载,在资源的利用率上,这种方法不是最优的.

在低报文频率情况下,远大于报文频率的轮询操作加重 CPU 负载;反之,中断产生的消息处理开销影响系统的性能.单独使用轮询或中断都不是改善报文吞吐率的有效途径.通过多次实验,我们发现,在网络负载较低的情况下,报文到达时间是不可预测的,此时利用中断处理机制能够很好地避免由于报文到达的随机性而产生的延迟;而在网络负载较高时,报文的输入速率基本达到一个稳定状态,这时轮询机制就能够充分发挥作用,保证系统的高吞吐率.根据上述结论,一种比较好的实现方法就是采用半轮询驱动进行传输控制.

图 2 是根据半轮询驱动机制得到的有限状态机模型.

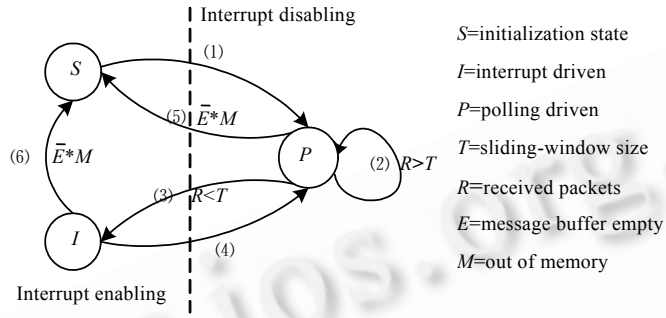

Fig.2 The state machine of semi-polling driven

图 2 半轮询驱动有限状态机模型

(1) 第 1 个报文到达,系统由开中断状态下的初始状态 *S* 转变为关中断状态下轮询机制 *P*,并通过可配置的滑动窗口尺寸 *T* 控制每次轮询所接收的报文个数 *R*;

(2) 若 *R*>*T*,说明报文输入率接近或达到系统饱和状态,则继续保持轮询驱动模式;

(3) 反之,为避免轮询开销,重新打开中断,直到下一个报文到达,并重复(4)(2)(3);

(4) 开中断状态 *I* 下,接收到一个新报文,则重新改变为关中断状态下轮询机制 *P*;

(5) 如接收队列未满,而内存资源耗尽,则重新回到初始状态 *S*,以释放所有资源;

(6) 同(5).

半轮询驱动机制的主体(即轮询操作)在内核的软中断过程中被调用,而软中断则通过 NA 的中断处理函数激活.这样,在低报文频率情况下,半轮询驱动基本上蜕化为传统的中断机制来处理大部分报文,以降低延迟.由此可见,该机制将中断与轮询紧凑结合,保证了在系统高负载情况下使用轮询操作,而在低负载情况下使用中断模式,在充分发挥二者优点的同时避免了各自的缺陷,在中断延迟与系统峰值吞吐量之间做到了很好的平衡.另一方面,定制的实时操作系统 RTLinux 内核保证了轮询进程的高优先级别,确保了对网卡轮询的精确性,提高了网卡接收报文的速度.测试数据表明,采用半轮询机制后,减少中断次数的效果非常明显,带宽性能被极大地提高.

3.2 静态快表地址翻译技术

在基于零拷贝思想的用户级报文传输机制中,网络接口(NI)的 DMA 引擎需直接与用户缓冲区进行交互.然而,异步 DMA 方式只能对锁定的物理地址操作,因此必须解决用于 DMA 传输的物理地址和用户缓冲区的虚拟地址之间的翻译问题.

为此,文献[12]提出 NI 和 CPU 共享内核页表 TLB 的方法,但是这种不安全的操作使得这种方式并不可靠;VMMC 和 U-Net/MM 利用 NI 缓存一部分动态锁定的用户缓冲区地址,并借助操作系统内核虚存管理系统完成缓存未命中时的额外操作,但每个报文传输都包括页面锁定/解锁操作,这些代价昂贵的原语操作导致报文传输带宽降低.

在 UMPS 中,通过查找 Linux 内核中的三级页表,逻辑设备 KA 获得用户缓冲区中每个虚拟页面的物理地址,并将该页面锁定于内存.转换后的物理地址缓存在一个静态地址快表 ATB(address translation buffer)中,所有用于报文传输的用户缓冲区物理地址被 ATB 完全覆盖.ATB 快表和事先被一次锁定的内存页面组成一套高效

的地址翻译机制.

静态 ATB 快表的技术优势主要体现在:

① 实现简单,相关开销小;

② 在内核空间实现,禁止用户层访问,安全性高;

③ 避免了频繁的内存页面锁定和缺页异常处理操作,提高了性能.

3.3 基于资源映射图的缓冲区管理算法

用户缓冲区用来缓存网络报文,是连接网络层界面和用户层应用的桥梁.传统的缓冲区动态管理算法出于对通用性的考虑,在每次报文传输时都进行 malloc/free 操作,这无疑会影响系统性能.为此,在 UMPS 中我们设计了基于资源映射图的缓冲区管理算法(MG).所谓资源映射图是一系列用于描述用户缓冲区的二元组的集合.算法的基本思想,是在用户空间预先分配一大块静态共享内存池,逻辑上划分为 2K 的存储单元(unit),每个 Unit 的相对偏移量和其所存放的数据报文的长度位于相应的二元组⟨*offset*,*size*⟩中,通过资源映射图统一管理存储单元,有效地避免了 malloc/free 操作.DMA 控制器通过索引资源映射图获得当前可用存储单元,并与之交互,从而来完成以太帧格式的报文传输.

为了支持并发访问并消除在共用同一存储单元时的互斥锁操作,在实现中采用 4 个环形队列来描述资源映射图:发送队列 txBusyQ、接收队列 rxBusyQ、发送空闲队列 txFreeQ 和接收空闲队列 rxFreeQ.txBusyQ 中保存着待发送数据的二元组;而 rxBusyQ 的每一个二元组则对应着一个接收到的报文,作为优化过程,短报文会直接存于其中.

资源映射图是控制数据收发走向的关键,它们通过 Linux 的系统调用 mmap 由核心空间映射至用户进程空间,为两者所共有,这就保证了无论是核心程序还是应用程序,都可以方便地将其视为自身存储空间的一部分而进行操作.

MG 算法避免了每次报文传输时都进行的 malloc/free 操作以及在共用同一存储单元时的互斥锁操作,改善了主机与网卡间 DMA 相关操作的性能,最大程度地提高了用户缓冲区的空间利用率,从而提供了一个轻量级报文缓存机制,降低了存储开销,实现了全双工的通信方式.

3.4 报文传输过程

通过 UA 提供的虚拟网络接口,应用进程对用户缓冲区进行存取操作,以完成报文的发送与接收.图 3 为 UMPS 数据传输过程,从中可以看出数据报文的走向及其层次关系.

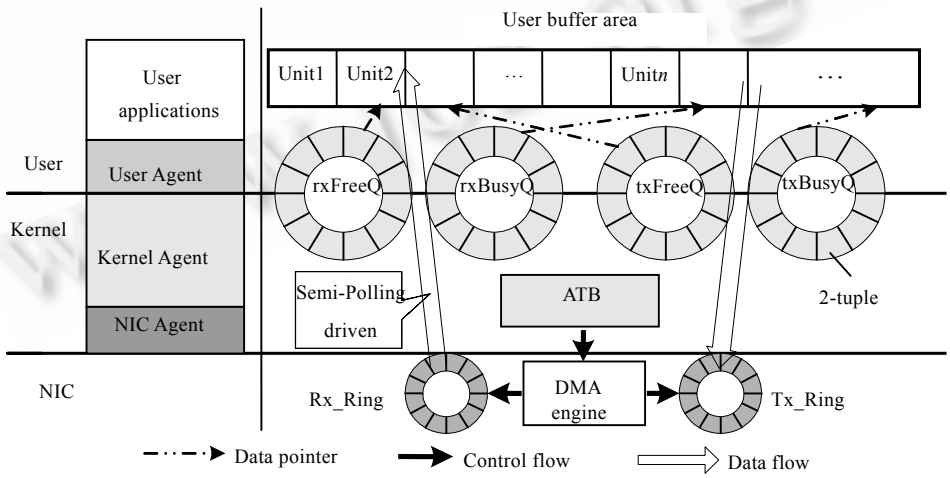

Fig.3 Communication procedure of UMPS

图 3 UMPS 的数据传输过程

结合环形队列的二元组和 ATB 快表,用户缓冲区当前可用存储单元的物理地址 A_p 和虚拟地址 A_v 可以通过

下面的地址翻译公式计算得到(其中,2K 是存储单元长度,*desc* 是队列当前可用二元组,*base* 是 ATB 快表的首地址,*buf* 是用户缓冲区的首地址):

$$A_p = base[desc \rightarrow offset >> 1] + (desc \rightarrow offset \ \& \ 1) \times 2K \quad (1)$$

$$Av = buf + desc \rightarrow offset \times 2K \quad (2)$$

通过半轮询驱动的传输控制,当新的网络数据报文到达时,NA 从 rxFreeQ 队列首部获取一个空闲存储单元的二元组,依据公式 1 获取已被锁定的空闲存储单元的物理地址 A_p,将接收到的网络以太帧通过 DMA 通道直接送到主机内存 A_p 中后,NA 将刚刚被填充数据的存储单元的二元组压入 rxBusyQ 队列尾部.应用进程需要处理报文时,首先从 rxBusyQ 队列首部获取已填充数据的存储单元的二元组,然后根据公式(2)得到虚拟地址 *Av* 索引相应的存储单元中的数据.应用进程处理完报文后将刚刚处理过的存储单元二元组压入 rxFreeQ 队列尾部.

报文发送流程与接收类似,应用进程在需要发送报文时,从 txFreeQ 队列首部获取一个空闲存储单元的二元组,依据公式(2)得到该存储单元的虚拟地址 *Av*,将欲发送的报文存放其中并将对应二元组压入 txBusyQ 对列尾部.NA 从 txBusyQ 队列首部获取欲发送报文的二元组,由公式(1)计算 A_p 并启动 DMA 进行报文发送.DMA 传输结束后,NA 将已发送完毕的存储单元的二元组压入 txFreeQ 队列尾部.

从报文的传输过程不难看出,UMPS 无须加锁/解锁 DMA 页面,分配/释放内存的操作也通过将二元组挂接在不同的资源映射图队列,从而得到简单的解决.

4 实验结果分析

为了对 UMPS 的性能作进一步的分析,我们对所实现的实例进行了性能测试.性能测试是在曙光服务器(CPU 是 PIV2.0GHz×2,内存是 4G),使用千兆快速以太网卡进行的.采取光纤互联的方式与专用硬件发包机 SmartBits-6000(B)相联.

4.1 中断频率

在完全输入短报文(64 byte 和 128 byte 的 IP 包)的情况下,我们针对半轮询驱动机制,在控制中断次数的性能方面进行了实验.在不同的速率和报文长度下,每次发送 100 000 个报文,经过多次测试求取平均值后得到中断产生次数的结果,如图 4 所示.从实际的测量数据可以看出,随着报文输入速率的增加,中断次数不断减少.这与我们的设计相符合:在半轮询驱动机制下,中断的次数由报文输入速率决定,高负载采用轮询机制保证吞吐量,反之蜕化为中断机制节约 CPU 开销.而传统的每包中断机制中断次数仅与输入报文数量有关,在同样的情况下,每包中断机制将产生 100 000 次中断,相比之下,半轮询驱动机制减少中断次数的效果非常明显,这为低延迟、低开销和高带宽的短报文处理能力提供了有效保证.

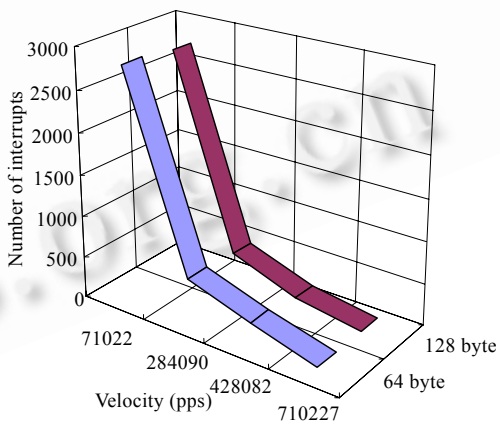

Fig.4 Number of interrupts

图 4 半轮询驱动机制下中断次数

4.2 带宽与延迟

为了测试引入半轮询驱动机制的 UMPS 的通信性能,我们选取用户级通信框架中比较典型的 MMUC、基于内核可加载模块的 KLMP 和传统的 Libpcap 作为主要比较对象.安装了实时操作系统 RTLinux-3.1(在 Red Hat 7.3 上升级安装),峰值吞吐量与接收延迟的测试结果如图 5 所示.图 5 中的接收延迟测试结果是用所测大小的数据报文向接收主机发送,直到其达到饱和状态后求出接收每个报文占用的平均时间得到的.可以看出,需要在内核与应用层之间拷贝数据的 KLMP 与 Libpcap 的接收延迟较 UMPS 和 MMUC 有明显的增加.另一方面,UMPS 的峰值吞吐量接近线速,在报文长度大于约 256 byte 时,由于报文频率降低,半轮询驱动逐渐蜕化为中断机制,因此与 MMUC 的测试结果曲线基本重叠,相差不大,但远远高于 KLMP 与 Libpcap.随着报文长度的增加,UMPS 的

峰值处理带宽不断提高,在处理大小为 1 500 byte 的 IP 数据报文时,达到极限值 895 Mbps,接近 1G,发挥了网卡性能的 89.5%,在流量上的性能提高是非常明显的.其主要原因是:Libpcap 采用传统的 TCP/IP 协议栈,多次数据拷贝、系统调用和复杂的缓冲管理,产生极大系统开销;KLMP 虽然绕过协议栈中协议层和套接字层,但并非真正的用户级传输,操作系统的干预没有完全去除,性能瓶颈依然存在;而 UMPS 则由于高效的地址翻译及缓冲区管理机制,应用程序与网络接口间可以直接传输报文,操作系统被完全旁路,实现了通信性能的最大化.

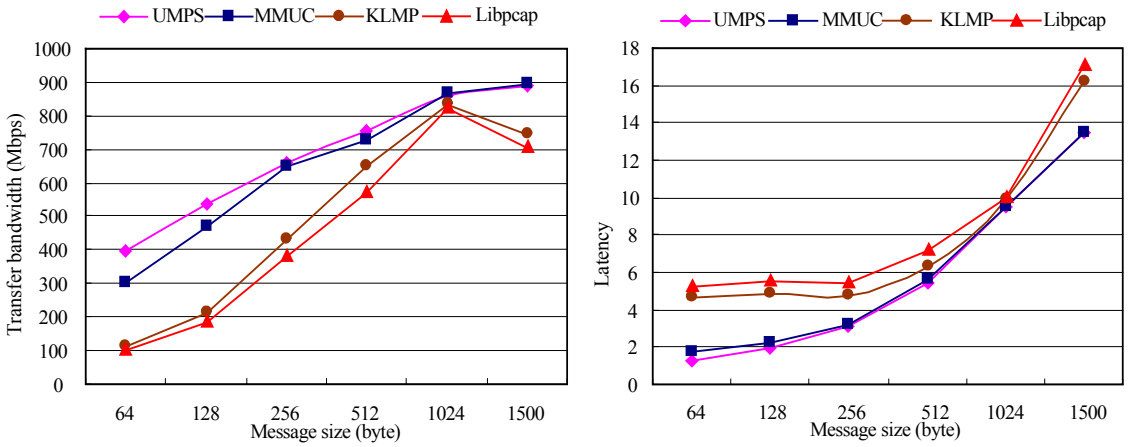

Fig.5 Bandwidth and latency for different platforms

图 5 不同平台下的带宽与延迟

图 6 为 4 种平台短报文处理能力的比较以及在处理 64 byte 短报文时的 CPU 空闲率.在 CPU 利用率上,UMPS 不是最优的,其原因有多方面,主要是图中的数据是在 64 byte 的短报文输入达到饱和状态时测得的平均值,此时的网络负载相当高,而由于半轮询机制产生很少的中断,为保证较高的系统吞吐率,大部分 CPU 时间都被用来对大量的数据报文进行轮询工作.在峰值吞吐量方面,对比 Libpcap,KLMP 和 MMUC 基于半轮询驱动的 UMPS 性能有很大幅度的提高.从图中可以看出,在 64 byte 短报文情况下,UMPS 的峰值吞吐量为 394 Mb/s,而 MMUC 最大可达 300 Mb/s,仅为 UMPS 的 76.1%,其主要原因是,在 MMUC 中频繁的中断使得大部分时间浪费在上下文切换以及中断现场保护上,UMPS 由于基于状态的半轮询机制以及基于资源映射图的缓冲区管理算法等多项优化技术,使得中断的频率和缓冲区管理的开销大幅度下降,充分发挥了网卡的硬件性能以及报文传递的实时性.因此,通过对 4 种平台的测试,表明 UMPS 在短报文的性能改善方面效果是最佳的. (%)

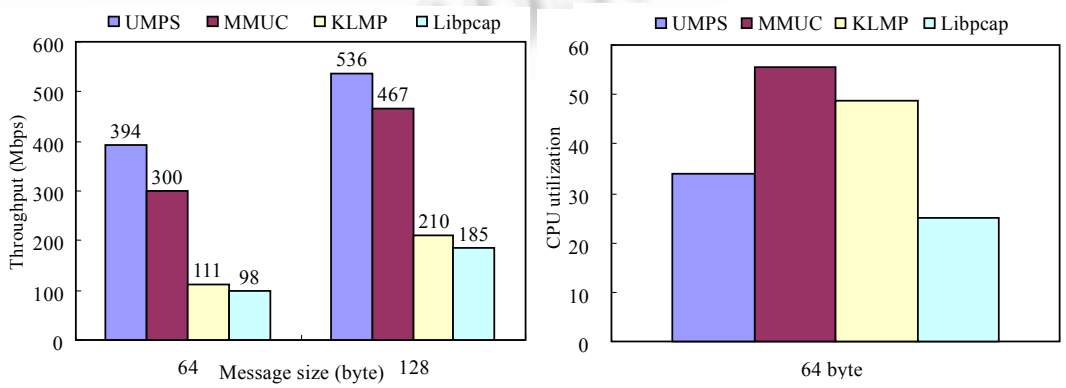

Fig.6 Throughput and CPU utilization for short message

图 6 短报文吞吐量和 CPU 空闲率

5 结 论

本文通过对典型的报文传输框架进行分析比较,设计并实现更为优化的报文传输机制——UMPS,由于引入基于状态半轮询驱动机制、高效的地址翻译和基于资源描述图的缓冲区管理算法,UMPS 充分挖掘了网卡的硬件性能,实现了一个灵活、高效的网络接口,简化了报文传递中的处理环节,极大地消除了系统的通信瓶颈,增强了报文的实时处理能力.实验数据表明,UMPS 有效地降低了系统的中断频率,并具有低延迟、低开销和高带宽的短报文处理能力.目前,虽然 UMPS 的峰值处理能力很高,但仍需要进一步完善和扩展.下一步我们将考虑采用用户级的调度来增加 CPU 的空闲率,以更好地支持上层应用软件.

致谢 在此,特别感谢与我们多次进行有意义探讨的哈尔滨工业大学计算机网络与信息安全技术研究中心的王佰玲、张永铮同学,以及王东滨、邹昕同学在实验过程中的鼎力协助.

References:

[1] Chang S-L, Du DHC, Hsieh J, Lin MJ, Tsang RP. Enhanced PVM communications over a high-Speed Local Area Network. IEEE Parallel and Distributed Technology, 1995,3(3):1405~1431.

[2] Pratt I, Fraser K. Arsenic: A User-Accessible Gigabit Ethernet Interface. Proc. of the INFOCOM, 2001,14(3):64~74.

[3] Pakin S, Karamcheti V, Chien AA. Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors. IEEE Concurrency, 1997,5(2):60~90.

[4] Zhang Y, Zhang DY, Li SL. The Research and Implementation of Network Intrusion Detection System Based on Cooperative Distributed Agent. Chinese Journal of Computer. 2001,24(7):736~741 (in Chinese with English abstract).

[5] Lifeline K. Building into the Linux network layer. Phrack Magazine, 1999,9(55):55~12.

[6] Wolman A, Voelker G, Thekkath CA. Latency analysis of TCP on an ATM network. Proc. of the Winter USENIX Conf. 1994,61(2):125~138.

[7] Blumrich MA, Dubnicki C, Felten EW, Li K, Mesarina MR. Virtual-Memory-Mapped Network Interfaces. IEEE Micro. 1995,15(1):21~28.

[8] Welsh M, Basu A, Eicken TV. Incorporating memory management into user-level network interfaces. In: Proc. of Hot Interconnects V. 1997. 618~628

[9] Liu W, Zheng WM, Shen J, Ju DP. Design and Implementation of memory map mechanism in underlying communication. Journal of Software, 1999,10(1):24~28 (in Chinese with English abstract).

[10] Tezuka H, O'Carroll F, Hori A, Ishikawa Y. Pin-Down cache: A virtual memory management technique for zero-copy communication. Proc. of the Int'l Parallel Processing Symp. 1998,26(1):308~341.

[11] Mogul JC, Ramakrishnan KK. Eliminating receive livelock in an interrupt-driven kernel. ACM Transactions on Computer Systems, 1997,15(3):217~252.

[12] Ang BS, Chiou D, Rudolph L, Arvind. Message passing support on start-voyager. In: Proc. of the 5th Int'l Conf. on High-Performance Computing. New York: IEEE Computer Society, 1998. 246~257.

附中文参考文献:

[4] 张勇,张德运,李胜磊. 基于分布协作式代理的网络入侵检测技术的研究与实现.计算机学报,2001,24(7):736~741

[9] 刘炜,郑纬民,申俊,鞠大鹏.底层通信协议中内存映射机制的设计与实现.软件学报,1999,10(1):24~28.