零拷贝技术在网络分析工具中的应用①

刘小威[1]，陈蜀宇[2]，卢 尧[1]，林铭炜[1]

[1](重庆大学 计算机学院，重庆 400030)
[2](重庆大学 软件学院，重庆 400030)

摘 要：针对目前网络分析工具的报文捕获机制及优缺点，实现了一种新的专用于报文捕获的网络协议簇—PF_ZEROCOPY。该协议簇基于零拷贝思想，借助内存共享技术，将网络报文直接 DMA 传输到用户空间的缓存区，绕开 Linux 网络协议栈、减少了内存拷贝次数；使用 DMA 缓存描述符环，实现了网卡和用户程序无冲突访问共享缓冲区；通过封装成内核网络协议簇，PF_ZEROCOPY 具有易于应用和移植的特点。实验结果分析表明，该方法对随机长度的报文捕获速率可达 900Mb/s 以上，与 libpcap 相比较为明显地改善了报文捕获的能力。

关键词：PF_ZEROCOPY；零拷贝；DMA；内存共享；描述符环

Application of Zero-Copy Technology to Network Analysis System

LIU Xiao-Wei[1], CHEN Shu-Yu[2], LU Yao[1], LIN Ming-Wei[1]

[1](School of Computer Science and Technology, Chongqing University, Chongqing 400030, China)
[2](School of Software, Chongqing University, Chongqing 400030, China)

Abstract: Aiming at the performance of packet capture, this paper realizes a new protocol family dedicated to packet capture called PF_ZEROCOPY. Based on Zero-Copy and memory sharing, PF_ZEROCOPY transfers the packet from network card to the memory that user program can access directly by DMA, so that bypasses network protocols, and reduces the times of data copy. By using the descriptor ring of DMA buffer, network card and user program can work without collision to access the shared memory. PF_ZEROCOPY is easy to use and transplant by packaged into protocol family. Experimental results indicate that the throughput of PF_ZEROCOPY for random size messages is above 900Mb/s, and PF_ZEROCOPY surpasses libpcap's in performance.

Key words: PF_ZEROCOPY; zero-copy; DMA; memory sharing; descriptor ring

为了提供网络监视、安全、流量分析和预警等服务，网络分析工具需要对网络数据进行实时的抓包存取。在大流量网络数据的情况下，不恰当的报文捕获技术会造成数据包延迟，甚至丢包现象，大大影响网络分析工具的性能。因此，报文捕获技术往往成为网络分析工具的性能瓶颈，是各种网络分析工具中的关键技术。

1 现有报文捕获机制概述

1.1 传统的报文捕获机制

在 Linux 系统中，传统的网络报文捕获过程是，当报文到达网卡时，网卡通过 DMA 方式将其传送到网卡驱动的缓冲环中，并在完成 DMA 传送时产生一个硬件中断。中断处理程序将接收到的数据包封装成 sk_buff 结构，然后由 netif_rx 传递给网络协议层进行进一步处理。在内核中经过协议层和 Socket 层处理后，用户进程通过系统调用 read 或 recvfrom 从内核空间读取数据包。

由于内核缓冲了绝大多数 I/O 操作，内核空间缓冲区在一定程度上分割了用户空间和物理设备，有利于应用层程序的设计实现。但是，内核缓冲区的使用却增加了数据拷贝、中断和 CPU 参与等系统开销，从

① 基金项目:重庆市自然科学基金(CSTC)(2008BB2307)
收稿时间:2011-07-22;收到修改稿时间:2011-09-05

而影响数据包捕获的性能。

1.2 基于零拷贝的报文捕获机制

为了提高报文捕获效率、减少系统开销，目前大多数网络分析工具也都摒弃了传统的报文捕获机制，而是采用零拷贝技术实现。零拷贝的基本思想，是节点在收发报文时，在网络设备和用户程序的应用缓冲区之间直接进行数据传输，绕开网络协议层，避免内存拷贝，并减少 CPU 参与的次数。

近年来，国内外学者提出了多种数据包捕获的方案，如 PF_RING[1]、HPPCP[2]、UPC[3]等。这些的报文捕获机制中，有的并没有采用零拷贝技术，有的实现了零拷贝技术，却需要针对具体的网卡，修改对应的网卡驱动，在不同的硬件平台和驱动环境下就无法简便地利用零拷贝技术，使得易用性和可移植性较差。

出于提升报文捕获性能和可移植性的考虑，本文实现了一种新的零拷贝报文捕获机制，并将其封装成网络协议簇——PF_ZEROCOPY，可方便地应用在网络分析工具中。

2 PF_ZEROCOPY零拷贝技术的方法原理

要避免数据在内核空间与用户空间的拷贝，最理想的方式是让网卡直接将报文传送到用户空间的应用缓冲区。DMA 允许数据在内存和 I/O 设备之间直接传输[4]，在接收数据包时，将数据报文直接从网卡 DMA 到用户进程事先申请的缓冲区，从而减少数据拷贝和 CPU 参与的次数。

网卡 DMA 操作需要知道报文缓存区的物理地址，我们将网卡的 DMA 缓存地址设置成用户缓存区的物理地址，这样网卡每次 DMA 传送报文时，都直接传送到用户进程事先分配好的缓存区。为了协调 DMA 与用户进程对缓冲区的无冲突访问，在内核态中定义了描述符环，用户进程通过 mmap 将描述符环映射到自己的进程空间，内核态和用户进程以内存共享的方式访问描述符环。采用 PF_ZEROCOPY 零拷贝方法，收发网络报文的流程如图 1 所示。

3 关键技术及实现

PF_ZEROCOPY 协议簇主要实现了以下功能：

① 初始化时，向网卡驱动和用户程序传递参数，申请描述符环内存空间；

② 将用户进程的虚拟地址转换成物理地址；

③ 协调内核态与用户进程无冲突访问共享缓冲区。

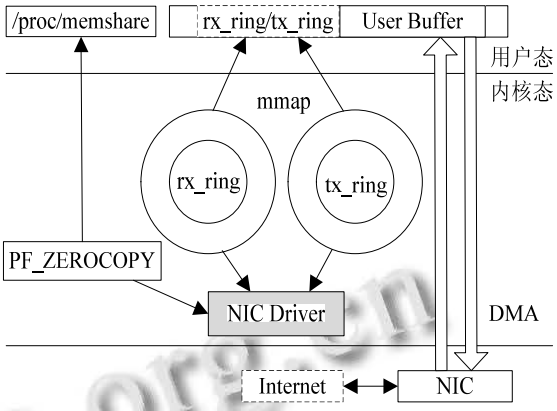

图 1　PF_ZEROCOPY 零拷贝的报文收发流程

3.1 主要数据结构及内存分配

3.1.1 User Buffer 的内存结构

User Buffer 是在用户进程中申请的，分为 rx_user_buf 和 tx_user_buf，分别用于存放 DMA 接收报文缓存和发送报文缓存。Linux 系统内存页面大小为 4kB，存放报文的一个数据块大小需要大于一个 MTU（1514 字节）。我们用一个内存页面存放 2 个数据报文，这样网卡进行 DMA 传送报文时，每个数据块的物理地址都是连续的，不会跨越两个页面[4,5]。为防止申请大的、物理地址连续的内存空间[6]，我们申请多个不连续的内存页组成 User Buffer。User Buffer 的报文数据块包含有数据报文和报文长度信息，其结构定义如下：

```
struct su_buff {
char packet[1514];
unsigned short len;
}
```

3.1.2 描述符和描述符环

描述符是网卡与 User Buffer 进行 DMA 传输的纽带，其中包含有 DMA 缓冲区块的物理地址和数据报文的信息参数，描述符结构定义如下：

```
typedef struct dma_desc {
unsigned long buf_addr;
unsigned long buf_va;
unsigned short buf_len;
unsigned short flag_access;
} dma_desc_t;
```

buf_addr 为该描述符所指向的缓冲区物理地址，buf_va 为物理地址 buf_addr 对应的应用进程的虚拟地址，buf_len 为对应的报文缓冲区的长度，flag_access 为该描述符指向的数据块的访问状态，有 3 种：可读（READABLE）、可写（WRITABLE）和未初始化（UNINITIALIZED）。

描述符环是描述符类型数组的循环队列，整个描述符环存放在内核空间的一片连续内存中，相邻的描述符元素在内存中处于相邻位置，从而 DMA 控制器可以快速地顺序读取描述符，有助于提供 DMA 传输效率。同时，由于网卡 DMA 和用户进程共享缓冲区，可同时操作 User Buffer，描述符环起着协调网卡驱动和用户进程无冲突访问 User Buffer 的作用。描述符环数据结构定义如下：

```
struct{
dma_desc_t descriptor[BUF_COUNT];
unsigned int read;
unsigned int write;
} rx_ring, tx_ring;
```

descriptor 为描述符数组，BUF_COUNT 为缓存数据块的个数；read 和 write 分别为 rx_ring/tx_ring 上可读标记和可写标记，指明下一个可读和可写报文数据块的描述符。

报文接收描述符环 rx_ring 在模块初始化时建立，用户进程申请到 User Buffer 后即对其进行初始化；而报文发送描述符环 tx_ring 在模块初始化时建立，在用户进程运行过程中，在发送的报文数据块时，对其进行描述符初始化或设置。

3.2 内核态与用户进程内存共享的实现

3.2.1 共享描述符环

描述符环是在内核空间的物理地址连续的内存块。在 PF_ZEROCOPY 模块插入时在 proc 文件系统中新建/proc/memshare 文件，通过/proc/memshare 将描述符环的地址 ring_mem_addr 和大小 ring_mem_size 传递给用户进程。用户进程获得描述符环地址和大小后，调用 mmap 映射到自己的进程空间，即可访问内核空间的描述符环。映射操作如下：

```
int map_fd = open("/dev/mem", O_RDWR);
map_addr = mmap(NULL,ring_mem_size,
PROT_READ|PROT_WRITE,
MAP_SHARED, map_fd,
ring_mem_addr);
```

3.2.2 共享 User Buffer/DMA 缓存

User Buffer/DMA 缓存是在用户空间开辟的，由用户进程和网卡驱动共享。操作系统为每个用户进程分配一段虚拟内存，用户程序进程空间的虚拟地址访问 User Buffer，但是网卡 DMA 对 User Buffer 进行读写操作时，是通过物理地址来寻址的。Linux 引入了虚拟内存的概念，内存管理单元 MMU 会将用户程序的页面在内存与虚拟存储器之间进行换入换出，但是 DMA 操作却要求 User Buffer 的内存页面常驻内存中。为此，在申请 User Buffer 后，即对内存页面的引用计数额外加一 page->count++，使得 User Buffer 的页面不被 MMU 置换出内存，在程序结束释放 User Buffer 之前，再把这些页面额外的引用计数减掉。网卡驱动和用户程序共享 User Buffer，无冲突访问的实现在 3.4 节详细阐述。

3.3 用户空间虚拟地址转换为物理地址

为了网卡 DMA 能直接访问到用户空间的 User Buffer，需要将 User Buffer 的虚拟地址转换为物理地址提供给网卡驱动。在用户进程调用 valloc 申请页对齐的 Buffer，并将自己的进程号 pid 和 Buffer 的虚拟地址 va，传递给 PF_ZEROCOPY，在内核态完成虚拟地址向物理地址的转换。

Linux 系统维护一个三级页表，用于进程虚拟地址到物理内存页面的映射和管理。通常情况下，用户进程申请内存空间时，内核并不立即分配真正的物理内存，而是推迟到要使用时才分配，并在三级页表中建立起虚拟地址到物理地址的映射，但这不影响本文的应用需求。虚拟地址到物理地址的转换实现如下：

```
static unsigned long va_to_phya(unsigned long va,
unsigned pid){
struct task_struct *pcb_task = pid_task(
find_vpid(pid), PIDTYPE_PID));
find_vma(pcb_task ->mm, va);
pgd_t *pgd = pgd_offset(pcb->mm, va);
pud_t *pud = pud_offset(pgd,va);
pmd_t *pmd = pmd_offset(pud,va);
pte_t *pte = pte_offset_kernel(pmd, va);
unsigned long pa = ( pte_val(*pte) & PAGE_
MASK ) | ( va & ~PAGE_MASK );
return pa;
```

}

3.4 DMA 和用户进程对 User Buffer 的无冲突访问

用户进程和网卡驱动在访问相同数据块时，就会产生访问冲突。DMA 和用户进程无冲突访问应用缓存，是借助内核空间的描述符环来实现的。我们在描述符环上设置了读写标记 read 和 write，来避免访问冲突。下面以接收报文为例说明无冲突访问的实现，描述符环与报文缓存的关系如图 2 所示：

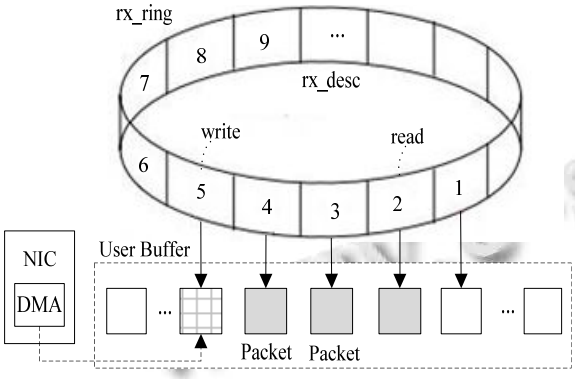

图 2　描述符环与报文缓存的关系结构

模块初始化时，read = write、且均为 0，描述符环上所有描述符的 flag_access 均为 WRITABLE。

① 网卡驱动写接收缓存

网卡驱动依次从 write 标记的描述符中，获得可写缓存块的物理地址，并将报文 DMA 到 rx_user_buf 中。如图 3 所示：

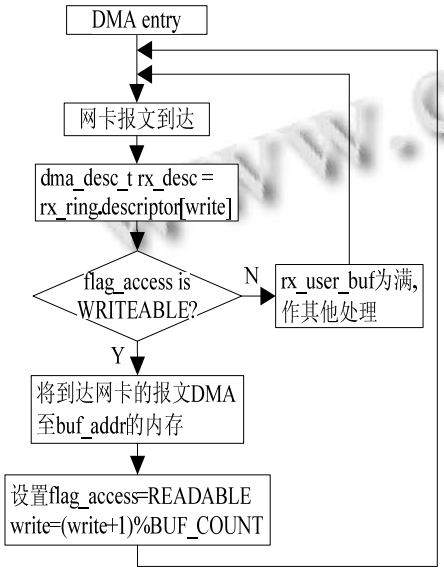

图 3　网卡驱动写 rx_user_buf

② 用户进程读接收缓存

用户进程的 Socket，从接收描述符环上的读标记开始，依次读取虚拟地址内存中的数据。若应用缓冲区为空，用户进程阻塞。如图 4 所示：

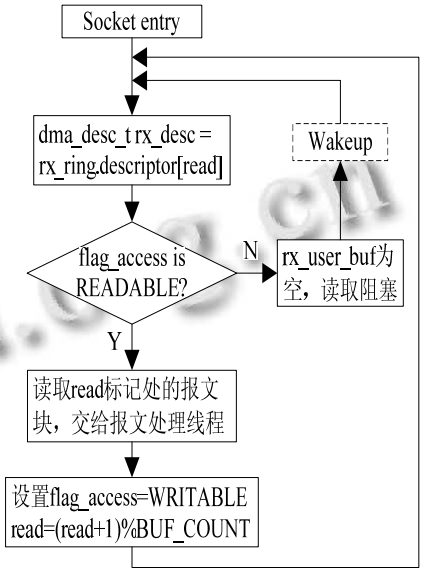

图 4　用户进程读 rx_user_buf

发送报文与接收报文的过程相反，稍有区别的是，用户程序向报文发送缓存 tx_user_buf 填充数据后，再设置发送描述符环 tx_ring 上相应的描述符的虚拟地址和物理地址信息。无冲突访问的控制与接收报文过程相同，这里不再赘述。

3.5 PF_ZEROCOPY 协议簇的封装

在 Linux 系统中有许多协议簇，我们将本文的零拷贝技术封装成 PF_ZEROCOPY 协议簇。在自定义的 net_proto_family 协议簇结构体 zerocopy_family_ops 中，注册自定义的 create 函数 zerocopy_create；在 zerocopy_create 中，使 sock->ops 指向自定义的 zerocopy_ops_spkt 结构体，定义如下：

```
static const struct proto_ops zerocopy_ops_spkt = {
.family = PF_ZEROCOPY
.owner = THIS_MODULE;
.sendmsg = zerocopy _sendmsg;
.recvmsg = zerocopy _recvmsg;
}
```

这样，就将基于本文零拷贝技术的报文接收函数 zerocopy_ recvmsg、报文发送函数 zerocopy _ sendmsg，注册进了系统内核。在网络分析工具中，创建基于

PF_ZEROCOPY 协议的 Socket 来收发数据包，即可方便地将本文零拷贝技术应用于报文捕获。

4 测试

为了分析 PF_ZEROCOPY 对报文收发效率的改进，我们在 Linux 3.6.31 平台实现了 PF_ZEROCOPY 协议簇，并对 e1000 网卡驱动和 libpcap 捕包工具作了修改[7,8]。测试环境：Intel Core2 Q9550 2.83GHz，4G RAM，Linux 2.6.31.5，Intel 82576 千兆网卡。测试由 netperf 向被测试主机发送不同大小的数据包，测试对比了 libpcap 与使用 PF_ZEROCOPY 协议簇的 libpcap 接收网络报文的能力。

在不丢包的情况下，两种方案接收报文的最大速率如图 5 所示。当数据包小于 256B 时，PF_ZEROCOPY 收包速率可达 libpcap 的 4 倍以上；在数据包大小大于 256B 和测试包大小随机的情况下时，PF_ZEROCOPY 收包速率均达到 900Mb/s 以上。比较可见，使用 PF_ZEROCOPY 协议簇后，接收报文的能力比 libpcap 有了极大的提高。

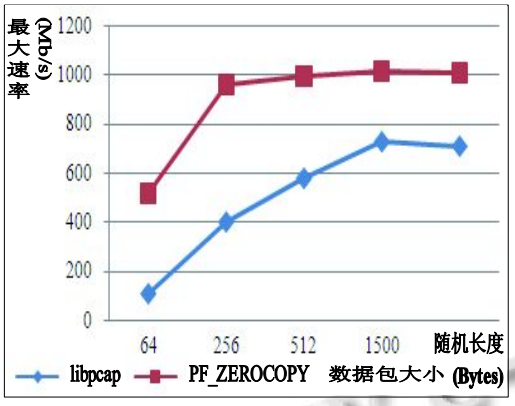

图 5 接收报文速率测试对比

5 结语

本文在现有的多种报文捕获机制的基础上，提出了一种新的实用性和可移植性更好的零拷贝网络传输方法——PF_ZEROCOPY，相比 libpcap 捕包能力有极大提高。随着多核 CPU 和多传输队列网卡的广泛使用，如何协调网卡传输队列与多 CPU 之间的负载平衡，在并行计算平台中进一步提高网络分析工具的报文捕获效率，是值得进一步研究的课题。

参考文献

1 Deri L. Improving Passive Packet Capture:Beyond Device Polling. Proc. of SANE. 2004.

2 王佰玲,方滨兴,云晓春.零拷贝报文捕获平台的研究与实现.计算机学报,2005,28(1):46–52.

3 笱程成,赵荣彩,邰铭.高负载网络下线速包捕获接口的设计与实现.计算机工程与设计,2010,31(10):2203–2205.

4 陈莉君.深入分析 Linux 内核源代码.北京:人民邮电出版社,2002.

5 Rubini A, Corbet J. Linux Device Drivers. 3rd ed. Sebastopol: O'Reiliy, 2005.

6 Araripe LE, Andrade JS, Costa Filho RN. Memory effects on the statistics of fragmentation. Physical Review E, 2005,71(3):036119/1–036119/5.

7 Lin MW, Chen SY, Chang GH, Liu LF, Gong XN. Research on PCA-based anomaly detection system in high-speed network. Journal of Computational Information Systems, 2011,7(7):2315–2321.

8 Lawrence Berkeley National Labs, libpcap, Network Research Group. http://www.tcpdump.org/.