基于 Linux 系统的报文捕获技术研究

杨　武[1]　方滨兴[2]　云晓春[1]　张宏莉[1]
[1]（哈尔滨工业大学国家计算机内容信息安全重点实验室，哈尔滨 150001）
[2]（国家计算机网络与信息安全管理中心，北京 100031）
E-mail：yangwu@pact518.hit.edu.cn

摘　要　网络带宽的不断增加对报文捕获技术提出了挑战。在对 Linux 系统下传统报文捕获机制深入剖析的基础上，量化分析了报文捕获过程中的性能瓶颈。针对性能瓶颈提出了优化和改进捕包性能的方案。

关键词　报文捕获　TCP/IP 协议栈　性能瓶颈

文章编号 1002-8331-（2003）26-0028-03　文献标识码 A　中图分类号 TP393

Study of Packet Capture Techniques in Linux

Yang Wu[1]　Fang Binxing[2]　Yun Xiaochun[1]　Zhang Hongli[1]
[1]（School of Computer Science and Technology，Harbin Institute of Technology，Harbin 150001）
[2]（National Computer Network and Information System Security Administration Center，Beijing 100031）

Abstract：The increasing network bandwidth challenges the packet capture techniques.Based on the in-depth analysis of the traditional packet capture mechanism in Linux，this paper measures the performance bottleneck in the process of packet capture.The schemes for optimizing and improving the performance of packet capture are proposed.

Keywords：packet capture，TCP/IP protocol，performance bottleneck

1　引言

在网络规模日益庞大的今天，报文捕获和分析技术作为网络管理和网络安全中使用的基本技术正得到广泛的应用。入侵检测系统、防火墙、流量统计以及网络计费等网络管理系统都是基于网络数据包捕获的方式实现的。通过对捕获的数据包进行分析，可以实时了解网络的使用状况（如响应时间、丢包率、某一时刻的网路负载及平均负载），发现网络运行瓶颈以及分析网络数据包中携带的非法内容。

由于网络技术和网络带宽的发展速度非常快，传统的 10Mbps 共享局域网已经迅速被 100Mbps、1000Mbps 的交换网络所取代，网络中的数据流量也有了成倍的增加。在骨干网络上，数据流量已达到每秒钟上 G 个比特。不断增大的网络流量对报文捕获和分析技术提出了挑战。

目前，在 Unix/Linux 操作系统下的网络数据包捕获系统普遍是建立在 Libpcap 捕包平台上的，Libpcap 的英文意思是 Library of Packet Capture，即数据包捕获函数库[1]。该库提供的 C 函数接口可用于需要捕获经过网络接口（只要经过该接口，目标地址不一定为本机）数据包的应用系统中。在大流量网络环境下，基于 Libpcap 的低效报文捕获技术无法捕获足够数量的网络数据包以供上层应用系统使用。

因此为保证各种基于 Linux 的网络应用系统能够在高速网络中有效地运行，有必要对网络报文捕获技术进行深入的研究，分析影响报文捕获性能的因素并在此基础上提出优化和改进方法。

2　Linux 内核协议栈分析与性能评价

在 Linux 系统中，操作系统内核协议栈为用户提供了一种工作在数据链路层的套接字 SOCK_PACKET。Libpcap 通过该套接字应用程序接口从用户态进入到系统内核态，绕过内核协议栈中的 TCP 层和 IP 层处理过程直接从数据链路层捕获原始网络数据帧。为了深入剖析 Libpcap 的捕包机制，首先在 Linux-2.4.10 操作系统的基础上，研究了传统 TCP/IP 内核协议栈的报文传输机制和性能瓶颈。

2.1　TCP/IP 协议栈报文传输模型

Linux 操作系统的内核协议栈基本可以分为数据链路层、IP 层、TCP/UDP 层、INET Socket 层、BSD Socket 层和应用层等几个部分[2]。以传输层协议为 TCP 协议为例，在图 1 中给出了典型的 TCP/IP 协议栈层次和模块结构。

内核协议栈包括一组函数和若干个关键数据结构。各个协议层的函数调用关系如图 1 所示。用户层的 Socket 套接字库通过内核的数据结构 Socket{}和 Sock{}来维护。数据在发送/接收过程中通过两个数据结构 Msghdr{}和 SK_buff{}来管理数据缓冲区。其中 Msghdr{}由 BSD Socket 和 INET Socket 层来维护，Msghdr {} 用于存放应用层数据缓冲区的地址和大小。在 TCP/UDP 以下各层使用 SK_buff {} 作为数据缓冲区。内核在 tcp_sendmsg（）/tcp_recvmsg（）函数中通过内存拷贝操作实现这两种数据结构的转换。内核数据缓冲区 SK_buff{}在通过 TCP/UDP 以下各层时，不进行数据拷贝操作，仅仅将数据指针在不同报文协议头之间移动。这样就可以避免不必要的系统开销。

基金项目：国家 863 高技术研究发展计划项目资助（编号：2002AA142020）
作者简介：杨武（1974-），博士研究生，主要研究方向网络安全、高性能计算。方滨兴（1960-），教授，博导，主要研究方向为网络信息安全，并行处理。
?1994-2018 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

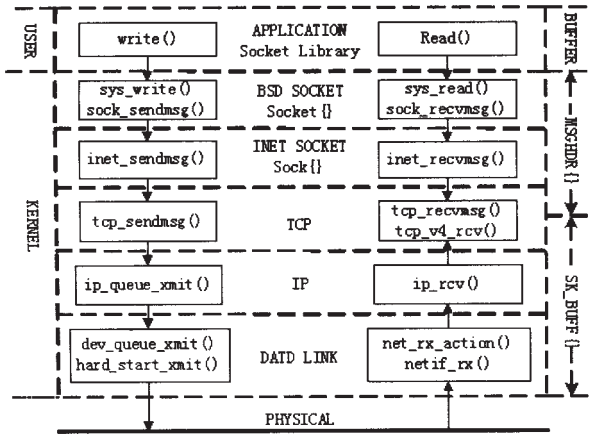

图 1　Linux-2.4.10 中的内核 TCP/IP 协议栈框架

2.2　TCP/IP 协议栈报文接收过程性能分析

以 Intel 100M 网卡 eepro100 作为物理层设备，对照图 1 中的内核协议栈框架，详细分析一下 Linux 系统内核协议栈的报文接收过程：

Linux 内核协议栈的报文接收分为两个过程：自上而下的过程和自下而上的过程。其中自上而下的过程是一个被动的过程，系统调用 read（或 recv（/recvfrom（通过上层协议栈传递‘需要数据’请求，通过 tcp_recvmsg（函数进入 TCP 协议层。函数 tcp_recvmsg（会到接收队列 sk{}->receive_queue 中读取数据，若接收队列中没有所需要的数据时，则函数 tcp_recvmsg（所在的当前进程就会休眠等待在 sk {}->receive_queue 接收队列上。当系统重新唤醒被挂起进程时，tcp_recvmsg（函数会读取所需的数据并调用 tcp_v4_do_rcv（函数将 sk{}->backlog 队列中的数据填充到 sk{}->receive_queue 队列中同时唤醒等待在该队列上的进程。tcp_recvmsg（函数在读取数据包的过程中，将数据包从内核缓冲区拷贝到应用程序缓冲区。自下而上的过程如下所示：

(1)当网络数据包到达后，eepro100 网卡通过 DMA 方式将数据包传输到网卡驱动程序缓冲环中并在数据传输结束时产生硬件接收中断。系统根据中断类型调用相应的硬中断处理程序，由此内核协议栈完成从物理层到数据链路层的转换。

(2)在硬中断处理程序 speedo_rx（中，内核处理接收到的数据包并通过调用 netif_rx（函数将驱动程序缓冲环中的数据添加到系统接收队列中。操作系统在空闲时机调用上层软中断处理函数 net_rx_action（。在函数 net_rx_action（中，根据已经注册的数据包类型（如 ETH_P_IP）调用相应的处理函数 ip_rcv（。这样内核协议栈从数据链路层进入 IP 层。

(3)在 ip_rcv 函数中，根据路由结果判断数据应该发送到上层 TCP 协议处理时调用 tcp_v4_rcv（。该函数会调用 tcp_v4_do_rcv（，通过函数 tcp_v4_do_rcv（将 sk{}->backlog 队列中的数据填充到 sk{}->receive_queue 队列中。

为了测试内核协议栈在报文接收过程中各个部分的系统开销（消耗 CPU 的时间），作者采用了和文献[3]类似的方法，在 Linux 下修改了网卡的驱动程序并在系统内核中插入了一些代码来记录报文到达各个部分的时间戳，从而可以确定各部分的时间代价。测试结果如表 1 所示。

从表中可以看出，系统硬中断的处理时间（从 speedo_rx（到 net_rx_action（）约占 20%左右，系统软中断处理的系统开销（从 net_rx_action（到 tcp_recvmsg（）约占 15%左右，而从 tcp_recvmsg（到 inet_recvmsg（过程中由于执行数据内存拷贝操作，所以消耗了大约 55%左右的 CPU 时间。系统调用大约占用 5~6%的 CPU 时间。内存拷贝操作的代价是昂贵的，主要有以下几点原因：(1)内存总线带宽有限，每次内存拷贝都要占用带宽；(2)每次拷贝操作都会消耗大量的 CPU 周期。通常，CPU 都是逐字地将数据从源缓冲区移动到目的缓冲区。这意味着在拷贝操作过程中，CPU 是不可利用的；(3)数据拷贝操作影响了系统 Cache 性能。因为 CPU 通过 cache 访问主存，因此在拷贝操作之前 cache 中驻留的有用信息会被清空，然后被拷贝的数据所代替。

3　基于 Libpcap 的传统捕包机制分析

Libpcap 利用工作在数据链路层的套接字 SOCK_PACKET 来完成网络数据包的读取。同样以 Intel 100M 网卡 eepro100 为例，分析 Libpcap 的捕包流程如下所示：

(1)当网络数据包到达后，eepro100 网卡通过 DMA 方式将数据包传输到网卡驱动程序缓冲环中并在数据传输结束时产生硬件接收中断。系统根据中断类型调用相应的硬中断处理程序。

(2)在硬中断处理程序 speedo_rx（中，内核处理接收到的数据包并通过调用 netif_rx（函数将驱动程序缓冲环中的数据添加到系统接收队列中。操作系统在空闲时机调用中断处理下半部分即软中断处理函数 net_rx_action（。此函数的主要功能是，检查系统中已注册的数据包类型并调用相应的处理函数。对应 Libpcap 的注册包类型为 ETH_P_ALL，其处理函数为 packet_rcv（。packet_rcv（仍然工作在数据链路层。

(3)在 packet_rcv 函数中，直接调用 skb_queue_tail（将数据包存放在 sk{}->receive_queue 队列中。这样数据包在接收过程中就绕过了 TCP 层和 IP 层的处理。

(4)由休眠在队列 sk{}->receive_queue 上的函数 packet_recvmsg（接收链路层数据帧并将数据帧拷贝到应用层缓冲区中。

Libpcap 的捕包流程和内核 TCP/IP 协议栈的报文接收过程对比如图 2 所示。从图中可以看出相对于传统内核 TCP/IP 协议栈的收包过程而言，Libpcap 绕过了 TCP 层（UDP 层）和 IP 层的处理过程，直接将数据包从数据链路层拷贝到应用程序缓冲区中。这样能够节省数据包在接收过程中所消耗的 CPU 时间，从表 1 中可以看出节省了大约 10%的处理时间。但是在 Libpcap 捕包过程中，系统调用、数据拷贝和内核中断处理仍然是系统主要的性能瓶颈。

表 1　收包过程中 TCP/IP 协议栈各个部分的时间代价

	speedo_rx（→netif_rx（	netif_rx（→net_rx_action（	net_rx_action（→ip_rcv（	ip_rcv（→tcp_recvmsg（	tcp_recvmsg（→inet_recvmsg（	system call
Times (μs)	4.1	4.0	1.7	3.8	20.9	2.1
Percentage	11.2%	10.9%	4.6%	10.4%	57.1%	5.8%

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

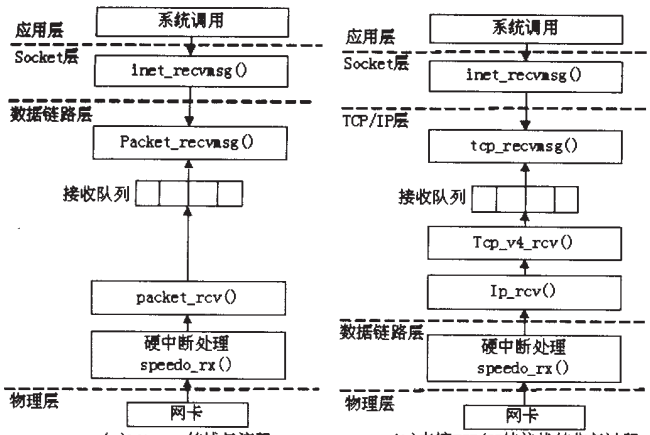

图 2　Libpcap 和内核 TCP/IP 协议栈的捕包流程对比

4　针对 Libpcap 的捕包优化措施

针对 Libpcap 捕包过程中的几个耗时环节，分别进行优化和改进以便减少系统资源消耗，同时提高 Libpcap 在大流量网络环境下的捕包性能。以下提出了几种主要的捕包优化措施并对其优缺点进行了分析。

(1)增加内核过滤机制

内核过滤就是在相当于软中断处理程序的部分(packet_rcv()函数)判断接收的数据包是否是应用程序感兴趣的报文。如果是才将其复制到应用层缓冲区内，否则就丢弃。这样可以大大降低系统实际处理的报文数量，从而提高了捕包效率。但这种方式仅对某些应用起作用，对某些应用如流量统计，常用协议分析等应用中就不会起很大作用，因为这些应用往往要处理网络上的绝大部分报文。

(2)将主要的应用处理移至内核模块处理程序中

由于系统调用涉及到 0x80 号中断现场保存和进程切换，在大流量网络环境下频繁的系统调用是很费时的工作。所以许多应用将主要的应用层处理部分作为 LKM 模块放在内核中处理，从而可以减少系统调用的次数。但是这样做需要将应用处理写在驱动模块或内核代码中，在内核中编写代码与应用层编写代码的方法以及使用的函数等有许多区别，而且内存分配、读写用户数据的方式也有所变化，内核编码的复杂性和难度都有所增加。另外内核代码对稳定性要求很高，一旦内核程序出现问题会造成整个系统的崩溃。所以，一般这种方法适用于一些处理效率要求高，处理过程相对简单的应用，比如防火墙等。

(3)一次复制多个报文到用户缓冲区

Libpcap 通过系统调用 revfrom()读取数据包，而且每调用一次该函数，它只向用户区传递一个数据包。因此可以在内核缓冲区中保存一定数量的报文，在报文达到一定数量时唤醒用户进程读取内核缓冲区中的所有数据包。这样就节省了用户进程切换的时间。在对大流量的数据报文进行处理时，是一个很有效的方法。不过这样会延迟报文到达用户进程的时间。当报文到达速率很大时，时间延迟会明显增加。

(4)旁路内核协议栈

从内核到应用层的数据内存拷贝操作是捕包系统的主要性能瓶颈。为此可以采用零拷贝技术，通过在内核添加处理模块来将用户缓冲区的虚拟地址转换为网卡可用的物理地址并锁定该地址。改进网卡驱动程序以获取用户缓冲区的物理地址并利用网卡异步 DMA 工作方式将数据包从网卡直接传送到用户空间，从而能够旁路操作系统内核协议栈，降低系统内核处理、数据拷贝和系统调用的开销。该方法能够大幅度的提高系统的捕包性能，甚至能够达到网卡的性能极限。但是由于需要修改网卡驱动程序，因而限制了这种方法的通用性。

(5)降低系统硬件中断频率

当网络报文到达系统速率过于频繁的时候，会出现 CPU 处理时间全部用于中断处理的情况。这时系统频繁运行硬中断处理程序，导致上层软中断处理得不到运行。硬中断处理程序将网络报文填充到系统缓冲区中，这样系统缓冲区很快就被添满，多余的数据包将被丢弃。同时其他进程也无法获得 CPU 的控制权。因此减少中断频率对于提高大流量网络捕包系统性能来说是必要的。在某些改进方法中，当发现中断频率过高时就禁止掉硬件中断[4,5]，这样虽然会丢失一些报文，但可使系统有机会响应其他的进程。一些系统使用了中断和轮询混合机制来降低系统中断频率，Macquelin 等人提出的轮询看门狗机制[6]就是这样一种方法。

5　结束语

报文捕获技术是网络性能监测、网络协议分析、网络入侵检测、防火墙等网络应用系统的基础。改进和提高报文捕获的效率是大流量网络环境下基于捕包技术的网络应用系统的重要研究内容。论文作为国家 863 计划资助项目‘基于异常检测与网络主动测量的大规模网络协作预警技术’的一部分，主要对现有报文捕获系统的捕包机制及其性能瓶颈进行了深入的分析并在此基础上提出了优化和改进的措施。这为进一步的工作指明了方向，可以针对具体的应用要求采取不同的优化策略，以便提高大流量网络环境下的系统捕包性能。

(收稿日期：2003 年 6 月)

参考文献

1.Libpcap.http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz

2.李善平，刘文峰等.Linux 内核 2.4 版源代码分析大全[M].机械工业出版社，2002

3.C Papadopoulos，G M Parulkar.Experimental evaluation of SUNOS IPC and TCP/IP protocol implementation，and internals[J].IEEE/ACM Transactions on Networking，1993，1(2)

4.Anja Feldmann.BLT：Bi-layer Tracing of HTTP and TCP/IP[J].Computer Networks，2000，33：321~335

5.J C Mogul，K K Ramakrishnan.Eliminating receivelivelock in an interrupt-driven kernel[M].USENIX Association，1996

6.Olivier Maquelin，Guang R Gao，Herbert H J Humt et al.Polling Watchdog：Combining and Interrupts for Efficient Message Handling[C].In：Proceedings of 23rd Annual International Symposium on Computer Architecture，ACM，Philadelphia，Pennsylvania，1996-05

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net