High-speed Packet Capture Mechanism Based on Zero-copy in Linux

Xiaochen Xu
Department of Computer Science
Xiamen University
Xiamen, 361005, P.R China

Zhongwen Li*
Department of Computer Science
Chengdu University
Chengdu, 610081, P.R China

***Abstract*—The development of gigabit-network demand higher standard of the packet capture's efficiency. Traditional network packet capture mechanisms get poor performance when used in gigabit-network. So we proposed a new network packet capture mechanism using zero-copy and POLLING. Through the modification to the bottom-layer, we achieve the packet capture in user space, and bypass the intervention of operating system's kernel. The efficiency of data processing is improved .We achieve wire-speed capture rate, and the CPU usage rate is below 10%.**

Keywords-Linux; Zero-copy; Packet Capture

I. INTRODUCTION

Nowadays, all operating systems offer the mechanism of bottom-layer network packet capture in order to set up the network monitoring. Due to the popularization of the Internet and threaten to the network security, packet capture can also be used in high-performance communication, firewall, intrusion detection, and so on. After the coming of gigabit-network, improving the packet capture technology becomes an urgent affair.

In traditional packet capture mechanism, packets needed to be copied twice after being captured [1]: first from network interface to the kernel of operating system and then from kernel to the user space, which lead to the lost of packet and reduction of the rate of capture. And at the same time of data-copy, users should send system call to the kernel, which bring mass interrupt when the data transferred in high-speed, then affect CPU usage-rate badly.

To solve the above problem, zero-copy mechanism [2] is introduced. There are two types of zero-copy: one uses DMA [3] to transfer network packets to the address space preassigned by system kernel directly, in the mean time maps the memory storing data packet in kernel to the application space; the other one create a cache in user space, and then map it to the kernel space, similar to the kiobuf in Linux. We use the first one and perfect it to implement our packet capture machanism, improve the packet capture capacity.

II. RELATED WORK

In recent years, there are a growing number of studies in the area of high-performance communications, which was mainly divided into hardware and software realization.

The authors would like to acknowledge the support of Fujian natural science foundation (2008J0034), Program of 2007 New Century Excellent Talents in Fujian Province.

*Correspondence author

A. *Hardware*

There are two special Gigabit Ethernet NICs(Network Interface Card) for packet processing in the current market: DAG[12] of Endace and COMBO developed by Liberouter[13] and SCAMPI[12] project. These two NICs process network packets by built-in co-processor or built-in firmware, so all tasks are completed inside NICs instead of transferring packets to host though the PCI bus. When the size of packets is 1500 bytes and the packet rate is 916.8Mbps, COMBO-4MTX[14] has the capture rate of 72%. And when the size of packets is 64 bytes and the packet rate is 448.8Mbps, its capture rate is 72%. DAG4.3GE[15] with co-processor reaches the wire-speed and has the capture rate of 100%. Compare the special Gigabit Ethernet packet processing NICs to Linux using NAPI and address mapping, the special NICs have very obvious performance advantages in the packet capture rate.

But the special NICs are very expensive: DAG4.3GE is about €5000 to €7000, and COMBO-4MTX costs about € 7000 to €10000. Therefore, achieving Gigabit network packet capture by software becomes the first technical issues that need to be resolved if real-time Gigabit network packets processing systems such as intrusion detection systems and firework are supposed to be developed.

B. Software

Luca Deri did the most outstanding work in the area of packet capture by software. He proposed PF_RING[16], and then developed a new network packet capture mechanism known as nCap[17] which realized a packet transfer system that didn't interfere by kernel. NIC's register and memory are mapped to the user space where the applications located. And the network packet processing applications which run in the user space are interact with the NICs by controlling the device of Linux. Then Endace purchased nCap, and developed a new network accelerated software-enCap in 2006. However, the source codes of nCap and enCap are not made known by now, and the technical details are still confidential.

In order to solve the problem of packet capture, more schemes are introduced. [18] proposed a file transferring mechanism based on VIA(Virtual Interface Architecture) that data are transferred between kernel buffer and remote node instead of using user buffer or page flipping. Kang[19] used shared kernel buffer pages to replace memory mapping to solve the data transferring problem. [20] proposed and

978-1-4244-4134-1/09/$25.00 ©2009 IEEE

implement a scalable architecture for distributed packet capture(DiCAP). [21] and [22] put forward a concept of "semi-poll-driven", which was designed to improve the efficiency of network packets stream collection and processing. They proposed User-Level Message Passing Mechanism(UMPS) for packet capture, which requested a contiguous physical address space as buffer area and then copy the packets that captured by the kernel to this buffer. By address mapping, applications read the packets in the kernel space directly to avoid the copy operation from kernel to the user space. The experimental results show that UMPS has certain advantages to PF_RING for small packets (64bytes) , but this advantage depends on the differences performance of the test hardware; and PF_RING has 10% advantage in performance for large packets(512 bytes or 1500bytes). [6] implemented a High-Performance Packet Capture Platform(HPPCP), which improved UMPS in the same hardware platform and reduce one copy in kernel space compare to PF_RING.

We can see that the study in high-speed research is very hot. All research above modified the packet capture software based on Libpcap and improved the capture rate or reduced the CPU usage rate. However, they didn't really bypass the kernel of the operating system, so the performance cannot be really improved. Livelock still exist when receive small packets. And these mechanisms did not provide a unified interface, which lead to difficulties in transplanting the systems based on Libpcap.

Our system realized a high-speed packet capture platform which combines the concept of zero-copy and POLL, bypasses the kernel of operating system and resolves the problem of synchronization when receive high-speed network data. And it also provides an upper-level interface to the user which is compatible with Libpcap. Test results show that our platform improve the capability of packet capture.

III. IMPLEMENT DETAIL

Packet capture includes three major components: in the bottom there is packet capture mechanism for a typical operating system, in the top is the interface to user application, and the third is packet filter.

A. Bottom-layer Packet Capture

1) Memory Mapping: Memory mapping [5] means a mapping between physical address in physical device and virtual address in kernel space or user space. After mapping, the process in kernel or user space can access the physical register directly.

After getting the head address and address range from network adapter's physical address by /proc/iomem, we use ioremap() to map the physical address of I/O memory to the kernel virtual address space in the kernel space, and in user space we use mmap() to implement the mapping between physical address and user virtual address space. And then we can directly access the receiving buffer of network adapter in user space.

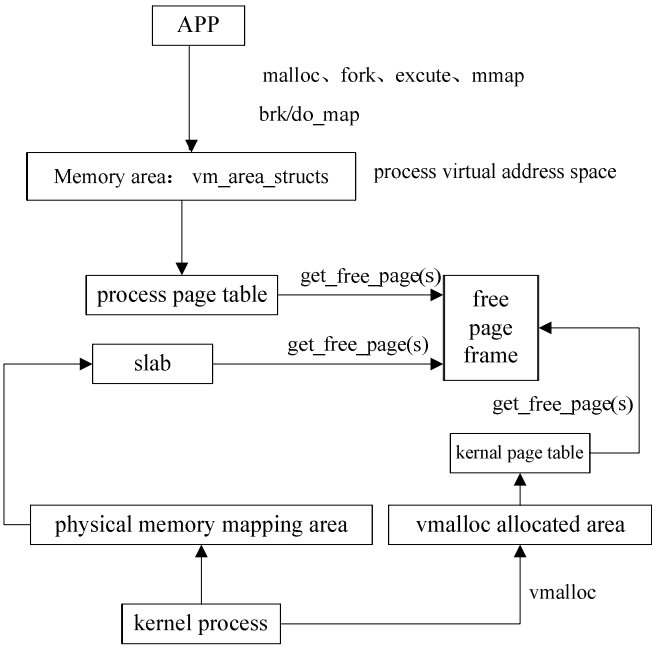

Figure 1. The mapping from kernel to the user space

2) Modification of Divece Driver: Modification of network adapter's driver contains three parts: modification of the memory buffer, control of circular buffer ring, and interrupt manipulation.

a) Modification of Memory Buffer[6]. In order to avoid the intervention of kernel and make user space application control the data transfer, we need to modify the network adapter's driver.

Two circular buffer rings was produced when network adapter was initialized: one for sending, and another for receiving. In the originality driver, new memory buffer corresponding to the circular buffer ring is needed when a packet has been consumed and a new packet arrives. Modifying the device driver according to the memory mapping to avoid the kernel involving, the send ring and receive ring can corresponding to the fixed memory buffer during the process of sending and receiving. And the user space can read and write the data in the circular buffer ring directly.

b) Control of The Circular Buffer Ring: When the network adapter receives packets, it copies the packets to the memory buffer, and the head index of the receive ring moves forward. When the head index equals to the tail index, it means that the circular buffer ring is full and no more packets will be captured. To improve the speed and efficiency of data capture, we make the tail index move forward as soon as the head index moved so that the head index can never catch up with the tail index and the "wire-speed" can be reached. In the receive ring, the head index is controlled by the hardware, and that tail index is controlled by the software. Through memory mapping, tail index can be read and write, so the network adapter can be controlled to receive data.

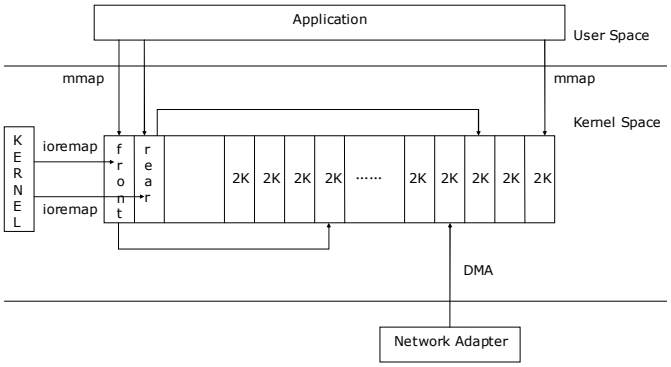

Figure 2. The logical structure of circular buffer ring

c) Interrupt Manipulation: When the packets were copied to the memory buffer, the network adapter produces a hardware signal to inform CPU to generate an interrupt. It will lead to livelock if the packets come at a high rate. We amend the NAPI [8]. It works as follow: first function e1000_probe() registers the polling method of E1000 NIC as function e1000_clean(), which adds the function of judging whether the circular buffer rings are full. Then e1000_intr() take control of the NIC's interrupt by disabling the interrupt and checking if the packets have been consumed by means of polling. Search the red-black tree's queue [9] of hrtimer, and wake the resting application of zero-copy up if found. The structure hrtimer_bases of function per_cpu() controls the root node of red-black tree. Red-black tree is a binary tree whose nodes are sorted in ascending order based on the sleeping time of the process, so we search the tree node's sleeping time to check for sleeping zero-copy application which is set a huge sleeping time (0x7fffffff). The flow of packets trigger the NIC's interrupt frequently, and the sleeping process is waked up soon to wake the zero-copy application by the sleeping value.

3) Solve The Synchronization [10] Problem: In the process of packet capture, the network device driver writes the network packet to the cache, and at the same time, user process analyzes the packet in cache directly. These two operations locate in different spaces, so complicated synchronization problem is needed to be solved. To solve this problem, we add a flag bit to the packet structure which identifies whether the packet can be read and write. When the network device driver fills the packet structure with data, the flag bit changes to readable. And after user process analyzes the packet, the flag bit changes to writable.

B. Upper-layer Interface to The User Application

Our system provides an interface similar to Libpcap which only needs to load a modified E1000 network adapter's driver module, without loading any other module. It have a core data structure pcap_t the same with Libpcap, and three core interface function: pcap_open_live() to open Zero-copy equipment, pcap_close() to close zero-copy equipment and release the resource, and pcap_next() to capture a packet from the circular buffer ring then return the user space virtual address of the packet.

C. Packet Filter Mechanism

In fact, the packet filter mechanism is operation functions to deal with the packet boolean value. If the function returns TRUE, it passes the filtration. And discard the packets if the function returns FALSE. Libpcap uses BPF (BSD Packet Filter) [12] as its main packet filter mechanism. When a packet arrives at the network interface, the driver at data-link layer will transfer it to the system's protocol stack. If BPF is listening in the interface, it filters the packet first and then stores the packet in buffer correlative to the filter. As our system is compatible with the Libpcap and the efficiency of BPF can fulfill our need, we only need to add the filter codes of BPF to the network device driver.

IV. TEST AND ANALYSIS

A. System Environment

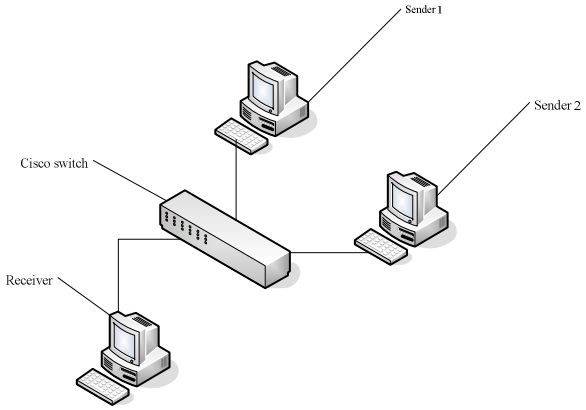

Figure 3. Topology of the test

- Hardware:

  Sender: P4 XEON 3.0G, 512M, Intel 82541GI gigabit network adapter

  Receiver: P4 2.4G, 512M, Intel 82541GI gigabit network adapter.

  Switch: Cisco WS-C3750-48TS-S gigabit switch.
- Software: RHEL4.0, kernel version is 2.6.6.19.

B. Result and Analysis

The packets sending program of sender is written by ourselves due to the high price of the special hardware and the poor performance of the existing packets sending programs. The packet rate of each senders can remains between 550Mbps to 700Mbps, so two senders are enough in our test.

The Rate of Capture is shown when the packets are all received, and the CPU Usage Rate is get by the "top" instruction of Linux.

1) Rate of Capture

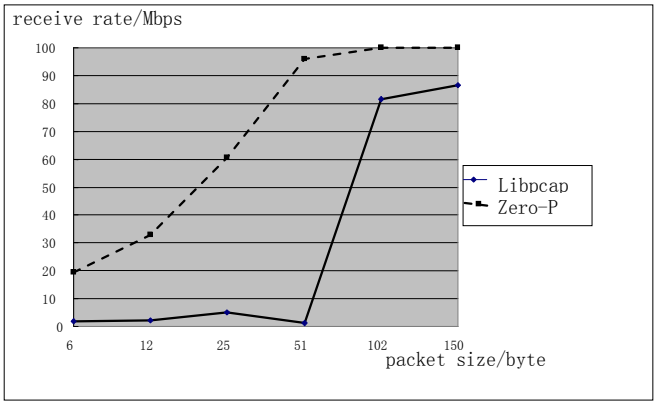

Figure 4. Rate of capture

2) CPU Usage

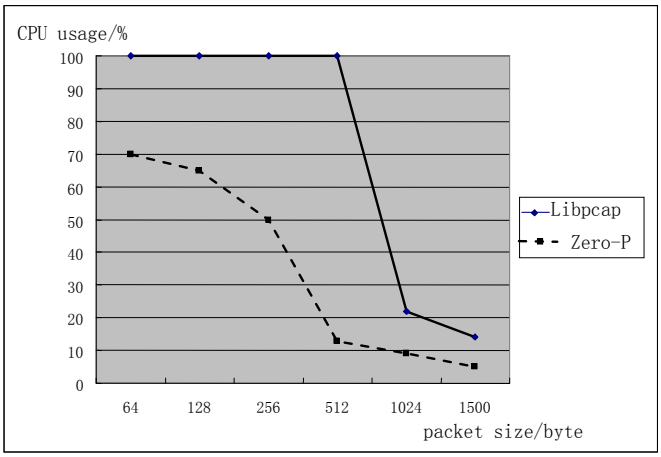

Figure 5. CPU usage

3) Analysis

As the figures show above, the mechanism we use improves the performance of capture compared to Libpcap both in capture rate and CPU usage. When the packets size is smaller than 512 bytes, Libpcap's CPU usage reach 100%, but most of the packets still can not be consumed and capture rate is far from the "wire-speed" of the network device due to its capture mechanism. Whereas our system increases the capture rate and decrease the CPU usage at the same time. Especially when the packets size is bigger than 512 bytes, the capture rate approach the wire-speed and the CPU usage goes down to 10%. Upon that, the system has enough time to deal with the packet like intrusion detection.

For packet filtering, we set a filter value such as the address of sender. Check the packet structure including address information and data structure that is printed out, and we can make conclusion that the captured data match the filter's value.

V. CONLUSION

We use the zero-copy to solve the problem of packet transmission in high-speed network, and improve the performance of system. To improve the system performance with small packets' will be our next step. Besides, gigabit-network bring the component of lager packet such as Jumbo Frame with MTU up to 9000 bytes, so how to deal with fragmentation and defragmentation will be another topic for us.

REFERENCES

[1] P. Balaji, P. Shivam, P. Wyckoff, and D. Panda, "High Performance User Level Sockets over Gigabit Ethernet", Proceeding of IEEE International Conference on Cluster Computing, Sept. 2002, pp.179-186.

[2] CHRISTIAN KURMANN, FELIX RAUCH ,THOMAS M. STRICKER. Speculative Defragmentation-Leading Gigabit Ethernet to True Zero-Copy Communication. Cluster Computing.vol 4.2001.p7-18.

[3] PCI/PCI-X Family of Gigabit Ethernet Controllers Software Developer's Manual (SDM) 82540EP/EM, 82541xx, 82544GC/EI, 82545GM/EM,82546GB/EB,and 82547xx, http://www.Intel.com/design/network/manuals/8254xGBe_SDM.htm.

[4] Hai Jin, Minghu Zhang, and Pengliu Tan ."Lightweight Messages: True Zero-Copy Communication for Commodity Gigabit Ethernet." X. Zhou et al. (Eds.): EUC Workshops 2006, LNCS 4097, pp. 153 ~162, 2006.

[5] Welsh M.,Basu A.,von Eicken T. "ATM and fast Ethernet network interfaces for user-level communication". Proceedings of the 3rd International Symposium on High-Performance Computer Architecture(HPCA), San Antonio, Texas, 1997, pp.332~342.

[6] Wang Bai-ling, Fang Bin-xing, Yun Xiao-chun, The Study and Implementation of Zero-Copy Packet Capture Flatform, 2005 Vol.28 No.1.

[7] Luca Deri, Improving Passive Packet Capture:Beyond Device Polling. 4th International System Administration and Network Engineering Conference - SANE 2004, october 2004.

[8] Xu Lin,Zhang Yunde. "Research on Technique of Data Packets Capture Based on NAPI". Computer Engineering and Applications.vol 40.pp138~159.2004.

[9] Luo Dongjun."A List Structure Capable of Fast Searching Based on Red-Black Tree" Journal of Yunnan University Nationalities(Natural Sciences Edition),vol 17,pp 250-254.2008.

[10] Wei Jianhua,She Kun. Complexity and suitability of data synchronization protocols for embedded devices. Journal of Chengdu University Of Information Technology.vol 20,pp 412~414,2005.

[11] Steven McGanne, Van Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture". Lawrence Berkeley Laboratory, Berkeley, CA, 1992.

[12] Wang Hanjiang, Zhu Niaoliang. "Research and Improvement of BSD Packet Filter". Computer Engineering and Applications. Vol 40, 2004, pp106~115.

[13] DAG cards. Endace corporation. http://www.endace.com.

[14] Liberouter project. http://www.liberouter.org.

[15] SCAMPI IST project. http://www.ist-scampi.org.

[16] Sven Ubik. Gigabit Ethernet Packet Capture Tools. CESNET Technical Report 6/2005. September 12, 2005.

[17] Luca Deri. nCap: Wire-speed Packet Capture and Transmission. Third IEEE International Workshop on End to End Monitoring - E2EMON. May 2005.

[18] Sejin Park , Sang-Hwa Chung .Design and implementation of an improved zero-copy file transfer mechanism. Lecture Notes in Computer Science. Parallel and Distributed Computing: Applications and Technologies.vol.3320.p446-450.2005.

[19] Dong-Jae Kang, Young-Ho Kim.Design and implementation of zero-copy data path for efficient file transmission. Lecture Notes in Computer Science. High Performance Computing and Communications.vol4208.p350-359,2006.

[20] Morariu, C; Stiller,B.DiCAP: Distributed Packet Capturing Architecture for High-Speed Network Links. 33rd Annual IEEE Conference on Local Computer Networks (LCN), Montreal, Quebec, Canada, 14 October 2008 - 17 October 2008, 168-175.

[21] Biswas,A.Sinha,P.On improving performance of Network Intrusion Detection Systems by efficient packet capturing. Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP.p1-4

[22] Tian Zhihong,Fang Binxing, Yun Xiaochun. User-Level Message Passing Mechanism Based on Semi-Polling Driven in RTLinux. Journal of Software, 2004 Vol.15 No.6.

[23] Tian Zhihong, Fang Binxing, Zhang Hongli. Design and implementation of network intrusion detection unit based on semi-polling driven . Journal on Communications, 2004 Vol.25 No.7.