Architecture for Internal Communication in Multi-gigabit IP Routers

Young-Cheol Bang¹, W.B. Lee², Hyunseung Choo³, and N.S.V. Rao⁴

¹ Department of Computer Engineering, Korea Polytechnic University
Kyunggi-Do, Korea
ybang@kpu.ac.kr

² Network Technology Laboratory, Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350 Korea
leewb@etri.re.kr

³ School of Information and Communication Engineering
Sungkyunkwan University, Suwon 440-746, Korea
choo@ece.skku.ac.kr

⁴ Computer Science and Mathematics Division, Oak Ridge National Laboratory
Oak Ridge, TN 37831 USA
raons@ornl.gov

Abstract. The IP packets from a source are suitably forwarded by the routers along the path(s) to the destination. The packets destined to the routers themselves are called non-forwarding packets, and their processing is crucial to the overall speed of IP routers. An architecture is proposed here for efficiently handling the non-forwarding packets for high-speed routers. This architecture includes the Inter-Processor Communication Message Protocol for internal communication needed within the router for distributed processing of the non-forwarded packets. Our implementation results show that this architecture improves the processing speed by 10% as compared to the existing mechanism based on UDP/IP.

1 Introduction

High-speed routers for fast packet forwarding are crucial to meeting the growing bandwidth demands of the Internet users and applications. High-speed routers are being designed with decentralized multiprocessing architecture for IP routing functions. Such a router consists of a number of host processors, line interfaces and switch fabrics [2,3]. Routing lookup algorithms, optimal buffer management algorithms and silicon techniques have been studied in order to increase performance of high-speed routers [4–6]. But the software architecture techniques still leave room for improvement. In particular for speeding up the high-speed routers, we need to efficiently implement various software modules such as Forwarding Engine (FE) in line interface, switch fabric module, FIB manipulation module and non-forwarding packet processing module.

A packet whose destination address is the router itself is termed *non-forwarded*, such as a routing protocol packet or an application packet. The processing time of the non-forwarding packets is a crucial factor in the overall performance of high-speed routers. In this paper, we present a software architecture for high-speed routers to efficiently process the non-forwarding packets. For communication between the processors within the router, we propose Inter-Processor Communication Message Protocol (IPCMP), which is a mechanism for Inter-Processor Communication (IPC). Our implementation results show that this architecture improves the processing speed by 10% as compared to the existing mechanism based on UDP/IP.

2 Architecture for High-Speed IP Router

The first generation routers are PC-based with a single shared bus. In second generation routers, the multiple bus architecture is used to support distributed processing. The third generation routers are based on multiple processor-space switching architecture, which employs switch fabric for fast forwarding [3][13]. This architecture contains line interfaces, a switch fabric, and a routing processor as shown in Fig. 1. Each line interface has its own processor and typically supports 4, 8, or 16 ports.

Fig. 1 High-speed IP router architecture

An IP (Internet Protocol) router examines the header of a packet and typically removes the link layer header (such as Ethernet header), modifies IP header, and replaces the link layer header for retransmission. Typically, packets are received at an inbound network interface and stored in the buffer. Then they are forwarded through the switch fabric to the outbound interface that transmits them on the next hop. Forwarding Engine provides the intelligence and processing power to analyze packet headers, lookup the routing table, classify the packets based on their destination and source addresses and other control information and rules, and also to provide queuing and policing of the packets. The switch fabric provides high-speed interconnection of the line interfaces. The basic functions of a switch fabric are the spatial transfer of

packets from their incoming ports to destination ports and buffering of packets in case of contention.

The host processor chooses the next-hop for a packet based on the information in its routing database. It supports an interior gateway (IGP) to carry out distributed routing and an exterior gateway protocol (EGP) to exchange topology information with other autonomous systems. Generally, it OSPF (Open Shortest Packet First) and IS-IS (Intermediate System to Intermediate System) for IGP, and BGP (Border Gateway Protocol) as an EGP. The Operation and Maintenance (OAM) module provides network management and system support facilities, including logging, debugging, auditing, status reporting, exception reporting and control. It also provides means for diagnosing faults and measuring traffic.

3 Proposed Mechanism for Processing of Non-forwarded Packets

When a packet to be forwarded arrives at a line interface, the forwarding engine reads the header to determine how to forward the packet and then updates the header and sends the packet to the outbound line interface through the switch fabric for transmission. Processing speed of a forwarded packet is affected by forwarding engine's capacity. In Fig. 2, A and B represent the processing paths of the forwarded packets. The packet whose IP destination address is the router itself needs to be processed on the host processor. These packets include routing protocol packets, Simple Network Management Protocol (SNMP), application, and error reporting packets. In Fig 2, the C and D represent the processing paths of the non-forwarded packets.

Fig. 2. Processing paths for forwarded and non-forwarded packets

Fig. 3. Software architecture in line interface

The software architecture of the line interface can be divided into two parts as shown in Fig 3. The first part is the NP (Network Processor) that processes the incoming packets. It validates TOS (Type of Service), TTL (Time To Live), and checksum fields. The NP reads the header to determine how to forward the packet, updates the header and sends the entire packet to the outbound interface through the switch fabric for transmission. The second part is the CP (Control Processor) which processes the non-forwarded packets which include packets with: 1) router itself as the IP destination address, 2) broadcast MAC address such as ARP request packet 3) multi-cast destination address such as OSPF packet, RIP packet. 4) unknown IP destination address invoking ICMP error-report message. 5) invalid header invoking ICMP error-report message.

Fig. 4. Software architecture in host processor

LDPPM (Line-interface Distributed Packet Processing Module) incorporates the decision algorithm to decide whether to process a non-forwarded packet in line interface DPPM. The ARP request or response packet has to manipulate in host-processor

DPPM for table consistency while the ICMP error-report packet is handled by LDPPM. OAM module handles the control messages from host interface such as interface configuration, FIB update information and access list and so on. IPCMP gives the method of data passing between processors, host interface and line interface.

16- bit Source Port Number	16- bit Destination Port Number
16- bit IPCMP length	Undefined
data	

Fig. 5. IPCMP header

The software architecture of the host processor is shown in Fig 4. HDPPM (Host-interface Distributed Packet Processing Module) receives the non-forwarded packets from the device driver and performs packet assembly and IP option processing. The remaining packet processing is performed at the higher layer such as TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). And ICMP messages (Echo reply message, Redirection message, Source Quench message, Time Stamp Reply message) are also managed by HDPPM

Since the most of IP packets generated by the host processor carry the IP address of the next hop, the host processor needs to access its individual routing table, which should be the master table of the entire router. The forwarding information base (FIB) in each line interface should match the master routing table of the host processor. The routing table in the host processor is updated either by the various routing protocol entities or by the network management action, which are assumed to happen infrequently. The FIB updates in the line interface can be done in two steps; first, any update information originated in the routing protocol has to be transferred to the line interface in the IPCMP format, and as a second step, pushed into the search database in the NP through API. Managing the routing table and FIB is up to the routing manager in OAM, but is not shown in Fig. 4.

This router architecture needs an IPC network to communicate between the routing processor and line interfaces, such as Cell Bus network, HDLC network, VME bus network, ATM Asynchronous Transfer Mode) switch bus network, and Ethernet switch bus network. Cell bus, VME and HDLC use a shared bus while ATM and Ethernet use point-to-point method. Throughput of HDLC and VME bus is low as compared to the point-to-point method [14,15]. In an Ethernet switch, we can implement IPC using TCP/IP or UDP/IP [16] using point-to-point or point-to-multipoint methods. The IPCMP in Figs. 3 and 4 is the substitute for the UDP/IP method, since it provides a direct Ethernet encapsulation to the application through the socket interface. In Fig. 5, Source Port Number is the port number of source application and Destination Port Number is the port number of destination application. IPCMP length field specifies the message size. Fig 6 shows the example code using IPCMP socket

API. IPCMP provides common sockets API which is also used for TCP/IDP sockets. We introduce a socket type called AF_IPCMP, which is similar to the AF_INET type used for TCP/IDP sockets. When a user calls “*socket (AF_IPCMP, SOCK_DGRAM, 0);*” to create a IPCMP type socket, we return the number as a new socket descriptor.

```

...
if ((sockfd = socket(AF_IPCMP, SOCK_DGRAM, 0)) < 0) {
    printf("Can't open IPCMP socket\n");
    exit(1);
}
...
while (tmp_count < max_packet) {
    slen = sendto(sockfd, (char *)&s_buf_person,
        sizeof(person), 0, (struct sockaddr *)&dst_addr,
        da_len);
    if (slen < 0) {
        printf("SENDTO ERROR\n");
        close(sockfd);
        exit(1);
    }
    rlen = recvfrom(sockfd, r_buf, 1500, 0,
        (struct sockaddr *)&src_addr, &sa_len);
    if (rlen < 0) {
        printf("RCVFROM ERROR\n");
        close(sockfd);
        exit(1);
    }
}
...
}
...

```

Fig. 6. Example code using IPCMP socket API

4 Experimental Results

The hardware platform used for performance evaluation consists of two Linux systems with customized boards running Linux kernel 2.2.14. The host processor consists of a PowerPC MPC750 (300MHz) microprocessor with 512 Mbytes of main memory and the other system, the line interface, consists of a PowerPC 405GP (266MHz) microprocessor with 512 Mbytes of main memory [17,18].

Fig. 7. The brief internal architecture for performance evaluation

In Fig. 7, the client application creates and sends variable packets to the server application and then server application replies them, like a ping-pong test. We measured the RTT of different sized packets as shown in Table 1.

Table 1. The round trip time of packets

Type	UDP/IP on Ethernet								
Bytes	60			500			1000		
Number of Packet	1000	5000	10000	1000	5000	10000	1000	5000	10000
Avg. RTT(msec)	289.83	1428.56	2845.32	548.52	2727.79	5459.54	804.63	4083.43	8250.13
Type	IPICMP on Ethernet								
Bytes	60			500			1000		
Number of Packet	1000	5000	10000	1000	5000	10000	1000	5000	10000
Avg. RTT(msec)	246.63	1225.71	2462.95	504.16	2541.01	5057.31	764.13	3820.52	7653.78

Based on RTT results, IPICMP resulted is faster by an average of 9.3% compared to the UDP/IP (60-byte packets: Avg. 14.2%, 500-byte packets: Avg.7.4%, 1000-byte packets: Avg.6.2%). This result is due to the one-layer packet processing instead of two layers, UDP and IP. Fig 8 compares the packet processing rate of IPICMP with that of UDP/IP. IPICMP shows the processing rate of Avg. 1.95 Mbps for 60-byte packets while UDP/IP shows the processing rate of Avg. 1.67 Mbps. Overall, we can increase packet processing rate of IPICMP by Avg. 10.3% compared to the UDP/IP.

Fig. 8. Processing rate of packets

5 Conclusion

In this paper, we introduced a software architecture for efficient manipulations of high-speed non-forwarding packets such as routing protocol packets, application packets and so on. As shown in results, the non-forwarding packet processing is the crucial part in overall performance of the high-speed router for reliable routing. A proposed IPC mechanism called IPCMP outperforms the existing UDP/IP technique by 10% in terms of RTTs. Furthermore, we can reduce the complexity of software modules by removing TCP/UDP/IP stack in Line Interface.

Since non-forwarding packets affect in overall performances of high-capacity IP routers by drastic increment, the high-capacity IP router should be designed in taking into consideration non-forwarding packets.

Furthermore, we also plan to analyze non-forwarding traffics more in detail and design more efficient distribution method for the routing table information to forwarding table in line interfaces for high-speed IP router such as TeraBit-Router for the future study.

References

1. A. Bharagava and B. Bhargava, "Measurements and quality of service issues in electric commerce software," in Proc. Application-Specific System and Software and Technology, pp. 26–33, 1999.
2. S. Keshav, R. Sharma, "Issues and trends in router design," IEEE Communications Magazine, Vol. 36, pp. 144–151, May 1998.
3. V.P. Kumar, T.V. Lashman, D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet," IEEE Communications Magazine, Vol.36, pp. 152–164, May 1998.
4. N. McKeown, P. Gupta, and S. Lin, "Routing lookups in hardware at memory access speeds," INFOCOM '98. Proceedings. IEEE , Vol. 3, pp. 1240–1247, 1998.
5. S. Nilsson, G.Karlsson, "IP-address lookup using LC-tries," Selected Areas in Communications, IEEE Journal on, Vol. 17, pp. 1083–1092, June 1999.
6. C. Donpaul, et al, "Implementing scheduling algorithms in High-Speed Networks," Selected Areas in Communications, IEEE Journal on , Vol. 17, pp. 1145–1158, June 1999.
7. J. Moy, OSPF Version2, RFC2328, April. 1998.
8. Christian Huitema, "Routing in the Internet," 2nd Edition, Prentice Hall, 1999.
9. D. Estrin, et al, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," Internet Request For Comments RFC 2362, June 1998.
10. J. Moy, "Multicast Extensions to OSPF," Internet Request For Comments RFC 1584, March. 1994.
11. F. Baker, "Requirements for IP Version 4 Routers," RFC 1812, June 1995.
12. Wang-Bong Lee, et al, "An Architecture of Distributed Multi-Gigabit IP Router," AIC 24th Conference, Seoul, Nov. 2000.
13. A. Asthana, C. Delph, H. Jagdish, and P. Kryzhanowski, "Towards a gigabit IP router," Journal of High-Speed Networks, 1(4), 1992.
14. J. Furnuas, et al, "A prototype for interprocess communication support, in hardware," Ninth Euromicro Workshop on Real-Time Systems, pp. 18–24, 1997.
15. Bup Joong Kim, et al, "Designed and Implementation of IPC Network in ATM Switching system," ICATM 2001, pp. 148–152
16. W.R. Stevens, TCP/IP Illustrated Volume 1, Addison-Wesley, New York, 1996.
17. The Linux kernel Archives. <http://www.kernel.org>
18. LinuxPPC. <http://www.linuxppc.org>