MPLS LER을 위한 패킷 프로세서 기반의 포워딩 엔진

(A Forwarding Engine based on the Packet Processor for MPLS LER)

박 재 형 † 김 미 희 †† 정 민 영 ††† 이 유 경 ††
(Jaehyung Park) (Mi-Hee Kim) (Min Young Chung) (Yookyung Lee)

요 약 MPLS 망의 경계에 위치한 레이블 에지 라우터는 다른 망과의 연동을 위해서 여러 가지 형태의 프레임을 처리할 수 있어야 한다. 라우터에서 프레임 처리 및 전달을 담당하는 포워딩 엔진은 라우터의 성능에 큰 영향을 미치는 요소이다. 본 논문에서는 여러 형태의 망과 연동 가능한 MPLS LER을 실현하기 위해서, 프로그램 가능한 이더넷 패킷 프로세서를 이용하여 포워딩 엔진을 구현하였다. 포워딩 엔진의 기반이 되는 프로그램 가능한 이더넷 패킷 프로세서에서 ATM 인터페이스를 통해서 들어오는 프레임을 처리하여 그 프레임의 목적지로 향하는 ATM 인터페이스로 보내기 위해서 이더넷 패킷 프로세서의 되돌림 기능을 사용하였다. 본 논문에서 구현된 포워딩 엔진의 성능을 프레임 되돌림 기능의 영향과 프레임을 처리하기 위해서 수행되는 명령어의 수 측면에서 실험을 통하여 분석하였다.

키워드 : ATM 기반 MPLS, 레이블 경계 라우터, 패킷 전달 엔진, 패킷 프로세서, MAC 프레임 되돌림

Abstract *The forwarding engine, which handles the incoming frames and forwards them to the appropriate outgoing interface, is the crucial factor of the router's performance. As the MPLS label edge router provides the facility that it is capable of interworking with various kinds of networks, the forwarding engine should have the flexibility processing the corresponding types of frames from such network interfaces. In order to support the flexibility, we implement the forwarding engine for the MPLS LER with ATM interfaces based on the programmable Ethernet packet processor. By exploiting instinct loop-back functionality of Ethernet packet processor, our forwarding engine handles and forwards the frames from/to ATM interfaces. The performance of our forwarding engine is evaluated by experiments on the effect of looping frames back and the number of Ethernet packet processor's instructions.*

Key words : ATM-based MPLS, Label Edge Router, Forwarding Engine, Packet Processor, MAC Frame Loopback

1. 서 론

인터넷 사용자의 수가 지속적으로 증가하고, Web의 이용이 늘어나고, 음성 및 화상을 기반으로 한 새로운 응용 프로그램이 보급됨에 따라서, 인터넷 트래픽의 양이 급격하게 증가하고 있다[1,2]. 한편, 통신 링크의 기술의 발달로 많은 양의 데이타를 고속으로 전송할 수 있는 고속의 물리적 링크를 망에 적용할 수 있게 되었다. 그러므로, 인터넷의 성능에 중요한 부분은 고속의 전송 링크라기보다는 라우터의 성능이다[3].

멀티미디어 데이타를 기반으로 하는 새로운 인터넷 서비스를 효율적으로 제공하기 위해서 IETF는 MPLS (Multi-Protocol Label Switching) 프로토콜을 제안하여 표준화를 추진하였다[4,5]. MPLS는 계층 3의 라우팅과 계층 2의 레이블 스위칭을 통합한 동등 모델의 하나로 MPLS 네트워크 내에서 모든 패킷을 레이블 스위칭

· 본 연구는 첨단정보기술연구센터과제 "고성능 보안 미디어 네트워크"를 통하여 과학재단의 지원을 받았음.
† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
hyeoung@chonnam.ac.kr
†† 비 회 원 : 한국전자통신연구원 네트워크연구소 연구원
kimmh@etri.re.kr
leeyk@etri.re.kr
††† 비 회 원 : 성균관대학교 정보통신공학부 교수
mychung@ece.skku.ac.kr
논문접수 : 2001년 4월 12일
심사완료 : 2003년 3월 31일

을 수행하여 전달하는 방식이다[1]. MPLS 네트워크 내에는 크게 두 개의 라우터가 있는데, 하나는 네트워크 경계에 위치하여 일반 패킷을 레이블을 붙여 전달하거나 레이블된 패킷을 일반 패킷으로 전달하는 레이블 에지 라우터(Label Edge Router, LER)와 네트워크 내부에 존재하여 레이블된 패킷을 레이블 스위칭에 의해서 전달하는 레이블 스위칭 라우터(Label Switching Router, LSR)가 있다.

MPLS 네트워크를 구성하는 라우터에서, 포워딩 엔진은 들어오는 프레임의 유효함을 확인하고 패킷의 목적지에 따라서 나가는 인터페이스를 결정하여 전달하는 기능을 수행한다[6,7]. MPLS 네트워크의 LER에만 포워딩 엔진을 장착하는 경우가 있으므로, LER에서 포워딩 엔진은 인터넷 서비스를 지원하는 MPLS 망의 성능에 중요한 요소이다. 한편, LER은 MPLS 네트워크를 포함한 기존의 IP 네트워크와 연동을 지원하여야 하므로 포워딩 엔진은 여러 가지 형태의 프레임을 처리할 수 있는 유연성을 제공하여야 한다.

본 논문에서는 포워딩 엔진이 프레임 처리의 유연성을 제공할 수 있도록, 프로그램 가능한 이더넷 패킷 프로세서에 기반한 MPLS LER을 위한 포워딩 엔진을 구현한다. 여러 가지 형태의 프레임을 처리할 수 있는 유연성은 이더넷 패킷 프로세서가 수행할 프로그램으로 제어된다. 한편, MPLS LER과 연결된 모든 ATM 인터페이스와 프레임을 송수신하기 위해서 이더넷 패킷 프로세서의 MAC(Medium Access Controller)를 통한 되돌림 기능이 사용되었다. 구현된 포워딩 엔진은 ATM 인터페이스로부터 들어오는 여러 가지 형태의 프레임을 처리하고 적절히 변경한 후, 이더넷 패킷 프로세서의 MAC의 되돌림을 통하여 프레임의 목적지로 가기 위한 ATM 인터페이스로 전달된다. 본 논문에서 구현된 포워딩 엔진의 성능을 프레임 되돌림 기능의 영향과 프레임을 처리하기 위해서 수행되는 명령어의 수 측면에서 실험을 통하여 분석하였다.

본 논문의 구성은 다음과 같다. 2장에서는 포워딩 엔진을 포함하는 일반적인 라우터와 본 논문의 기반이 되는 프로그램 가능한 이더넷 패킷 프로세서에 대해서 기술한다. 3장에서는 MPLS LER에서 처리하여야 할 여러 가지 형태의 프레임에 대해서 살펴보고, 프로그램 가능한 이더넷 패킷 프로세서에 기반한 MPLS LER을 위해 구현한 포워딩 엔진에 대해서 설명한다. 4장에서는 프레임 되돌림 효과와 수행되는 명령어 수 측면에서 구현된 포워딩 엔진의 성능을 실험적인 방법으로 측정하였고 마지막으로 5장에서 결론을 맺는다.

1) MPLS는 계층 3과 계층 2의 프로토콜에 무관하지만, 본 논문에서는 계층 3과 계층 2가 각각 IP와 ATM인 경우만을 고려한다.

2. 라우터의 구조와 패킷 프로세서

본 장에서는 포워딩 엔진을 포함하는 일반적인 라우터의 구조와 프로그램 가능한 이더넷 패킷 프로세서에 대해서 기술한다.

2.1 라우터 구조

패킷 포워딩을 담당하는 라우터는 그림 1에서와 같이 라우팅 제어 프로세서, 스위치 패브릭, 네크워크 링크 인터페이스, 포워딩 엔진으로 이루어진다[1,3,6]. 라우팅 제어 프로세서는 RIP, OSPF, BGP와 같은 라우팅 프로토콜 및 MPLS를 지원하기 위해서 LDP(Label Distribution Protocol)을 수행하고, 그러한 프로토콜에 의한 라우팅 정보를 이용하여 라우팅 테이블 및 레이블 테이블을 유지한다. 또한, 라우팅 제어 프로세서는 스위칭 패브릭의 연결과 큐를 관리한다. 스위칭 패브릭은 그림 1에서 보여진 라우터를 구성하는 네가지 구성 요소들을 연결한다. 그리고, 네트워크 링크 인터페이스를 통해서 외부의 타 라우터와 연결된다.

라우터에서 포워딩 엔진은 들어오는 프레임을 처리하여 적절한 네트워크 인터페이스를 통해서 원하는 목적지로 전달하는 역할을 수행한다. 포워딩 엔진은 하드웨어로 구현될 수도 있고, 소프트웨어로 구현될 수도 있다. 하드웨어에 의한 구현 방식은 빠른 패킷 처리 능력을 지원하지만, 비용이 높다는 단점이 있다. 소프트웨어에 의한 구현 방식은 기반이 되는 패킷 처리 프로세서의 성능과 패킷 포워딩 프로그램에 영향을 받는다. 그러나, 소프트웨어 방식은 다양한 형태의 패킷을 처리하는 유연성을 지원해 주는 장점이 있다.

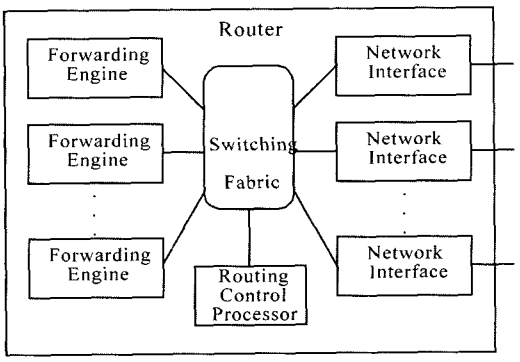


그림 1 일반적인 라우터의 구조

2.2 프로그램 가능한 이더넷 패킷 프로세서

프로그램 가능한 패킷 프로세서로는 MMC Network 사의 EPIF[8]와 C-Port 사의 C-5[9]와 IBM 사의 NP4GS3[10] 등이 있다. 이와 같은 패킷 프로세서는 포워딩 엔진이 프레임 처리의 유연성을 제공해 준다는 특징을 가지고 있다. C-5와 NP4GS3 패킷 프로세서는 높은 처리율을 지원하지만, MMC Network 사의 EPIF 프로세서에 비해서 비용이 많이 소요된다. 본 논문에서는 포워딩 엔진에 패킷 처리 유연성을 전달하기 위해서 EPIF를 기반으로 설계 및 구현하였다.

EPIF 프로세서는 RISC 형태의 특수 목적용 마이크로 프로세서로서 그림 2와 같이 Search Machine, CHANNEL, MAC, slicer로 이루어졌다.

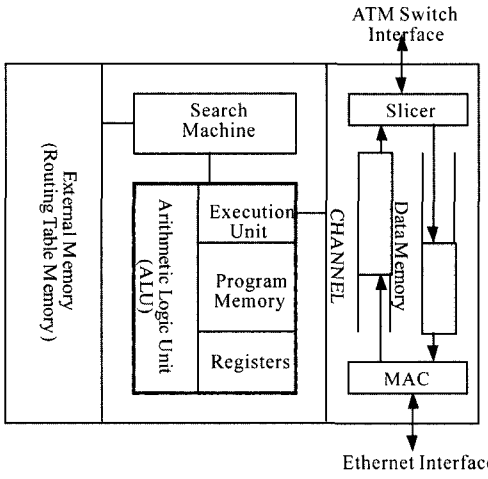


그림 2 EPIF의 구조

Search Machine은 패킷 포워딩 테이블을 유지하고 테이블에서 주소에 의한 탐색을 지원하기 위해서 Patricia Trie 구조를 기반으로 구현되었다. CHANNEL은 slicer와 MAC 사이에서 데이타 경로와 데이타 버퍼를 지원하는 패킷 메모리로 구현되었다. 이러한 CHANNEL은 마이크로 프로세서에 의한 패킷의 조작을 수행한다. MAC은 외부의 물리적인 디바이스인 10/100 이더넷과 연결할 수 있으며, slicer는 내부의 ATM 스위치 인터페이스와 연결을 지원한다. 특히, slicer에서는 외부 인터페이스의 프레임 형태와 내부 스위치의 프레임 형태를 지원하는데, AAL5 프로토콜에 의한 SAR(Segmentation And Reassembly) 기능을 수행한다.

마이크로 프로세서는 EPIF의 핵심으로 MAC을 통해서 들어오는 프레임과 slicer를 통해서 들어오는 프레임을 처리한다. 마이크로 프로세서는 패킷 처리를 위해서 RISC 형태의 명령어를 지원한다. 이러한 명령어는 다음과 같이 분류된다. ALU 관련 명령어, LOAD/STORE 명령어, 비교 명령어, 분기 명령어, Search Machine 명령어, CHANNEL 명령어로 분류된다. 이러한 명령어에서, Search Machine 명령어는 포워딩 테이블에서 탐색을 지원함은 물론 포워딩 테이블에 추가, 삭제, 변경을 수행한다. 그리고, CHANNEL 명령어는 처리하고 있는 프레임에 대한 임의의 바이트 추가, 삭제, 변경을 지원한다. 이러한 명령어로서 EPIF 마이크로 프로세서는 패킷을 처리하고 적절한 인터페이스로 전달한다.

3. MPLS LER을 위한 포워딩 엔진

본 장에서는 MPLS LER에서 처리하여야 할 프레임 형태에 대해서 기술하고, 그러한 프레임을 처리하기 위한 흐름도와 패킷 프로세서에서 수행되는 프로그램을 기술함으로써 프로그램 가능한 이더넷 패킷 처리 프로세서 기반의 포워딩 엔진을 구현한다.

3.1 프레임 형태

MPLS LER에서는 MPLS 네트워크와 송수신되는 MPLS 형태의 프레임과 ATM 인터페이스를 통한 기존의 IP 네트워크와 송수신되는 IP-over-ATM, PPP-over-ATM 프레임[11]을 처리하여야 한다. 표 1(a)는 32비트의 Shim 헤더를 포함하는 MPLS 프레임을 보여준다. Shim 헤더는 20비트의 레이블과 3비트의 실험용 비트, 레이블 스택킹을 지원하는 1비트의 스택비트와 IP 패킷의 TTL(Time to Live)와 동일한 8비트의 TTL을 포함한다.

표 1 프레임 형태

Label	Exp(S)	TTL
Labeled Frame 또는 IP Packet		
AAL5 Trailer		

(a) MPLS 프레임

LLC(0xAAAA03)	OUI
(0x000000)	PID
IP Packet, ARP Packet, 또는 Labeled Frame	
AAL5 Trailer	

(b) IP over ATM 프레임

LLC(0xFEFE03)	NLPID(0xCF)
PID	
NCP, LCP, 또는 IP Packet	
AAL5 Trailer	

(c) PPP over ATM 프레임

IP-over-ATM 프레임은 표 1(b)와 같이 LLC/SNAP 헤더를 갖으며, PID 값에 의해서 패킷의 형태가 결정된다. PID 값이 0x0800, 0x0806, 0x8847, 0x8848일 때, 각각 IP 패킷, ATM ARP 패킷, 레이블된 프레임, 레이블된 멀티캐스트 프레임을 의미한다. PPP-over-ATM은 표 1(c)와 같이 LLC/NLPID 헤더를 갖으며, PID에 의해서 패킷의 형태가 결정된다.

3.2 프레임 처리 흐름도

MPLS LER이 단지 ATM 인터페이스를 통해서 외부와 연결을 하기 위해서는 포워딩 엔진의 기반이 되는 이더넷 패킷 프로세서의 되돌림 기능을 이용하여야 한다. 그림 3은 MAC의 되돌림 기능을 이용한 프로그램 가능한 이더넷 패킷 처리 프로세서의 프레임의 흐름도이다. 외부와 연결된 ATM 링크 인터페이스를 통해서 해당 포워딩 엔진으로 들어온 프레임은 이더넷 MAC으로 나가는 방향에서 프레임에 대한 처리를 수행하고, MAC에서 되돌림 된 후, 스위치로 들어오는 방향에서 프레임에 대한 남은 처리를 수행한다. 스위치로 들어오는 방향에 대해서 처리를 할 때, 해당 프레임이 나가야 할 적절한 ATM 인터페이스에 대한 정보를 이용하여 프레임을 스위치를 통해서 외부 ATM 인터페이스로 전달한다.

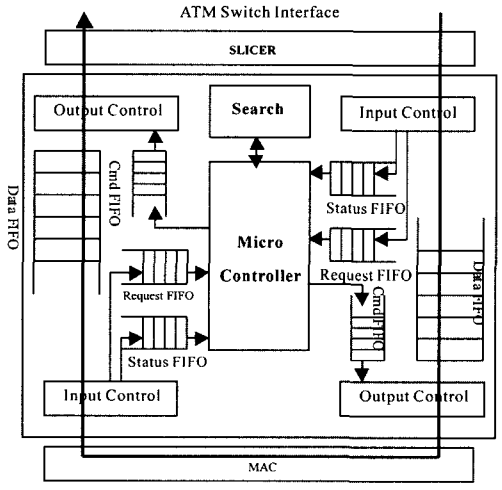


그림 3 프레임 흐름도

ATM 인터페이스를 통해서 FE으로 들어오는 IP 패킷을 처리하여 전달하는 마이크로 프로그램은 표 2와 같이 개략적으로 기술되었다.

표 2에서 기술된 프레임을 처리하기 위한 프로그램은

표 2 프레임 처리 프로그램

A	move imp, reqf, #WWW

B	load #IM_DATA_REL, r0_0, I_INCR, F_NODELAY
	move r4_0, r0_0, #DDD
	andi r4_3, r4_3, #0F0h, #BBB
	
	move scmde, cr1, #DDD
	smwr r4_0, #SM_SEARCH(SM_KEY48, ,)
	move r3, scmd, #LLL
	
	or tr0, r3_0, r3_4, #DDD
	JE (drop_frame)

C	cpib r3_3, #0fh, #IPOA_TYPE JE (ipoa_data) cpib r3_3, #0fh, #PPP_TYPE JE (ppp_data) cpib r3_3, #0fh, #MPLS_TYPE JE (mpls_data) JMP (drop_frame)
D	cpiw r0_4, #0FEFEh, #WWW JNE (drop_frame) cpiw r0_6, #003CFh, #WWW JNE (drop_frame) load #IM_DATA_REL, r0_0, I_INCR, F_NODELAY cpiw r0_0, #IPinPPP, #WWW JNE (send_to_RCP) load #IM_DATA_REL, r0_0, I_INCR, F_NODELAY cpib r0_3, #0ffh, #ICMP/#IGMP JNE (send_to_RCP)
E	cpib r0_2, #0ffh; #1 JLE (drop_frame) sbi r0_2, r0_2, #1, #BBB ; dec TTL adi r0_4, r0_4, #1, #BBB ; inc CRC
F	load #IM_DATA_REL, r0_0, I_INCR, F_NODELAY ; check Source or Destination Address ; SA or DA is zero? ; DA is one? ; DA is my address? ; None of above, do l3_lookup

G	move scmde, cr2, #DDD
	smwr r0_2, #SM_SEARCH(SM_KEY32, ,)
	move r1, scmd, #LLL
	
	or tr0, r1_0, r1_4, #DDD
	JE (send_to_RCP)

H	cpib r3_3, #0fh, #PPP_TYPE JE (send_to_PPP) cpib r3_3, #0fh, #MPLS_TYPE JE (send_to_MPLS) cpib r3_3, #0fh, #IPOA_TYPE JE (send_to_IPOA) JMP (drop_frame)
I	cmd r1_0, #XFER_IMM(O_NOPN,C_NCL,5,0), , cmdi #XFER_FRM(L_END,4,O_OPN,C_CL, , ,4), ,
J	move r0_0, sttf, #WWW ; check if frame status is bad btj r0_0, #ingress_error, #07h, 1

각 블록 별로 다음과 같은 기능을 구현한 것이다.

블록 A는 이더넷 패킷 프로세서의 Request FIFO에서 들어오는 패킷에 대한 정보를 얻는 것으로 패킷이 메모리에 저장되는 시작 주소를 IMP(Internal Memory Pointer) 레지스터에 저장한다. 이 때 저장되는 패킷의 정보는 내부 스위치의 VPI/VCI 값을 포함한다.

블록 B는 load 명령에 의해서 레지스터에 패킷의 8바이트를 읽어서 저장한다. VPI/VCI 값을 key로 하여 그 패킷에 대한 논리적인 인터페이스, 인터페이스에 대한 패킷 형태와 같은 정보를 얻기 위해서 레지스터 값을 Search Machine에 명령을 보내고 결과를 기다린다. 결과가 오면 Null인가 비교한다.

블록 C에서는 블록 B에서의 입력 패킷 형태에 따라 각각의 루틴으로 분기하는 명령을 수행한다.

블록 D에서는 LLC/NLPID의 헤더를 검사하고, IP 패킷만을 추출하기 위해서 다른 패킷을 RCP(Routing Control Processor)로 보낸다.

그리고, 블록 E에서 IP헤더의 TTL 값을 조정하기 위해서 1을 뺀 후 헤더의 CheckSum을 다시 생성한다.

블록 F는 IP의 목적지 주소를 검사하는 루틴으로 특정한 목적지를 갖는 패킷을 RCP로 보낸다.

블럭 G에서 IP목적지 주소를 key로 하여 패킷이 전달되어야 할 다음 홉에 대한 정보를 얻기 위해서 Search Machine에 명령을 보내고 결과를 기다린다.

블록 H는 나가게 될 패킷의 형태에 따라 각각의 루틴으로 분기하는 명령을 수행한다.

블록 I는 마이크로 컨트롤러에서 패킷을 전달하는 명령을 Command FIFO를 통해서 보내고, 블록 J는 전체 패킷에 대한 정보를 참조하여 패킷에 오류가 있는 경우에 오류처리 루틴을 수행한다.

이더넷 패킷 프로세서에서 수행되는 프로그램을 구현함에 있어서 프레임을 되돌리는 시점에 따라서 MAC으로 나가는 방향의 처리와 스위치로 들어오는 방향의 처리가 분리된다. 프레임 처리 흐름에 따라서 다섯 가지 방법이 있다. 그것은 A를 수행한 후, B를 수행한 후, D를 수행한 후, G를 수행한 후, I를 수행한 후가 있다. A를 수행한 후의 시점은 들어오는 패킷을 그대로 되돌리는 방법이고, B를 수행하는 후의 시점은 들어오는 인터페이스에 대한 정보를 구한 후에 구해진 정보를 포함하여 되돌리는 방법이고, D를 수행한 후의 시점은 프레임에 대한 헤더의 유효성을 검사한 후에 되돌리는 방법이고, G를 수행한 후의 시점은 프레임이 나가야할 출력 ATM 인터페이스에 대한 정보를 얻은 후에 그 정보를 포함하여 되돌리는 방법이고, I를 수행한 후의 시점은 모든 것을 처리한 후에 단지 출력 ATM 인터페이스에 대한 정보를 포함하여 되돌리는 방법이다.

3.3 프레임 처리의 예

본 절에서는 표 2에서 기술된 패킷 처리 프로그램의 예를 간단히 살펴본다. 이더넷 패킷 프로세서에 LLC/NLPID의 PPP 프레임이 ATM 인터페이스를 통해서 VPI/VCI값이 1/33을 갖고 들어왔다고 가정하자. 그림 4는 PPP로 캡슐화된 IP 프레임을 16진법의 표현으로 나타낸 것이다.

VPI VCI	LLC	NLPID	PID	IP 헤더의 일부분	TTL	P	Check Sum	SA	DA	IP 데이타
001 0011	FEFE03	CF	0021	8bytes	10	7	1111	4bytes	4bytes	

그림 4 PPP 형태로 캡슐화된 IP 패킷의 예

블록 A에서는 이 프레임의 시작 주소를 IMP(Internal Memory Pointer) 레지스터에 저장한다.

블록 B에서는 load 명령에 의해서 레지스터 r0에 패킷의 8바이트를 읽어서 VPI/VCI 값만을 추출한다. VPI/VCI 값(1/33)을 key로 하여 그 패킷에 대한 논리적인 인터페이스, 인터페이스에 대한 패킷 형태와 같은 정보를 얻기 위해서 레지스터 값을 Search Machine에 명령을 보내고 결과를 기다린다. 그 인터페이스가 LLC/NLPID 형태의 PPP-over-ATM이라는 정보를 얻는다.

블록 C에서는 입력 패킷 형태가 PPP-over-ATM이므로 PPP 형태를 처리하는 루틴으로 분기한다.

블록 D에서는 LLC/NLPID의 헤더에서 LLC가 FEFE03이고, NLPID가 CF인지 검사하고, load 명령에 의해서 레지스터 r0에 그 다음 8바이트를 읽어서 PID가 0021이므로 PPP 형태로 캡슐화된 IP 패킷임을 확인한다. 그 다음 8바이트를 읽어서 IP 데이타의 프로토콜(그림 4에서 P 필드)이 무엇인지 비교하여 ICMP나 IGMP가 아니므로 포워딩 엔진에서 처리한다.

블록 E에서는 IP헤더의 TTL 값을 조정하기 위해서 10에서 1을 뺀 후 헤더의 CheckSum을 다시 계산하여 1211로 변경한다.

블록 F에서는 다음 8바이트를 읽어서 IP의 목적지 주소를 검사하여 주소가 없거나 브로드캐스트거나 멀티캐스트인지 확인한다.

블럭 G에서는 현재 패킷의 IP목적지 주소가 블록 F에 해당되는 것이 없다고 가정하여 목적지 주소를 key로 하여 패킷이 전달되어야 할 다음 홉에 대한 정보를 얻기 위해서 Search Machine에 명령을 보내고 결과를 기다린다.

블록 H에서는 Search Machine의 결과에 출력되어야 될 패킷의 형태가 결정된다. 만약 출력 패킷의 형태가 IP-over-ATM 형태라면 그에 해당하는 루틴으로 분기하는 명령을 수행한다.

블록 I에서는 마이크로 컨트롤러에서 출력되어야 할 VPI/VCI 값을 cmd 명령어를 통해서 알려주고, 패킷의 LLC/SNAP에 대한 IP 형태를 AAAA03 0800을 전달한다. 또한 현재 프레임의 10번째 바이트부터 즉, IP 헤더의 처음부터 cmdi 명령어를 통해서 프레임을 전달한다.

그러면, ATM 인터페이스를 통해서 들어온 PPP-over-ATM 프레임 형태의 IP 패킷이 IP-over-ATM 형태의 패킷으로 변경되어 ATM 인터페이스를 통해서 출력된다.

3.4 포워딩 관련 테이블

이더넷 패킷 프로세서 기반한 MPLS LER의 포워딩 엔진을 구현하기 위해서 IP 패킷의 처리 및 전달할 때 Search Machine에는 두 개의 테이블이 유지 관리된다. 하나는 ATM에서 들어오는 패킷에 대해서 RCP로의 내부 스위치 VPI/VCI값, 인터페이스 등의 정보를 얻고자 할 때 사용되는 것으로 48비트 고정 길이 탐색(Fixed Length Search)으로 이루어진다. 28비트 VPI/VCI 값과 20비트의 무의미한 값으로 결합되어 key를 이룬다. 다른 하나는 32비트의 IP 주소를 통해서 다음 홉에 대한 정보를 얻는 것으로 가변 길이 탐색의 하나인 Longest Prefix Search이다.

4. 실험결과

본 장에서는 여러 가지 형태의 프레임을 생성하고 해석할 수 있는 시험장비(Smartbits, ADtech)를 통해서 프로그램 가능한 이더넷 프로세서로 구현된 포워딩 엔진의 성능을 측정하였다.

4.1 이더넷 패킷 프로세서의 최대 성능

우선, 포워딩 엔진 구현의 근간이 되는 이더넷 패킷 프로세서의 최대 성능을 조사하였다. 이더넷 패킷 프로세서에서는 프레임에 대해서 어떤 처리도 수행하지 않고 단지 패킷을 이미 정해진 인터페이스로 전달하는 기능만을 수행하게 하고, MAC으로 나가는 방향의 성능과 스위치로 들어오는 방향의 성능을 측정하였다.

그림 5에서 X 축은 생성되는 프레임의 크기를 바이트 단위로 나타낸 것이며, Y 축은 입력되는 프레임 개수에 대한 출력되는 프레임의 개수를 비율로 나타낸 것이다. 이 때, 트래픽의 생성은 이더넷의 최대 처리 값인 100Mbps를 유지하였다. 그림 5에서 보는 바와 같이, 스위치로 들어오는 방향의 처리는 최대의 성능을 유지하지만, MAC으로 나가는 방향의 처리는 그렇지 못함을 볼 수 있다. MAC으로 나가는 경우 slicer의 SAR 기능이 MAC의 처리 속도에 의해서 제한을 받기 때문이다. 특히, 성능의 차이가 큰 경우는 ATM 셀 payload의 정수 배인 48*n바이트보다 8바이트 적을 때이다. 즉, n이 2보다 클 때, 패킷 크기가 48*n - 8바이트인 경우이다.

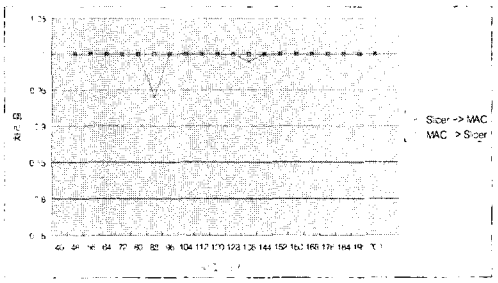


그림 5 이더넷 패킷 프로세서의 최대 성능

4.2 프레임 되돌림의 효과

이더넷 패킷 프로세서의 프레임 되돌림으로써 초래되는 부하에 포워딩 엔진의 성능에 미치는 영향을 측정하였다. 프레임은 임의의 시점에서 MAC을 통해서 되돌려질 수 있지만, 프레임 처리의 되돌림되기 전의 상황과 같은 상황에서 처리되기 위해서는 필요한 정보를 포함하여 MAC으로 되돌림되어야 한다. 표 3은 되돌림되기 전에 포함되어야 하는 정보량을 보여준다.

입력 인터페이스에 대한 정보에는 인터페이스의 번호와 패킷 형태 및 인터페이스의 서브넷 정보를 포함하고, 출력 인터페이스에 대한 정보에는 인터페이스의 번호와 패킷 형태를 포함한다. D나 G를 수행한 후에는 계층 2의 헤더가 필요가 없으므로 헤더의 크기만큼 제거하고 보내도 처리 가능함으로 S(L2Header)만큼의 바이트 수가 줄어든다. 본 논문의 성능 측정시에는 MPLS 프레임인 경우 그 크기가 최소인 4바이트이므로 (L2Header)의 값을 4로 정하였다.

그림 6은 되돌림에 의해서 추가되는 정보량이 적을수록 성능이 좋아짐을 보여준다. 되돌림에 의한 송신 및 수신에 드는 전송지연이 있기 때문이다. 그림 5에서 보

표 3 되돌림에 의해 추가되는 정보량

되돌림 시점	포함되는 정보	정보의 크기(바이트 수)
A 수행 후	입력 VPI/VCI 정보	4바이트
B 수행 후	입력 인터페이스 정보	16바이트
D 수행 후	없음	- (L2Header) 바이트
G 수행 후	출력 인터페이스 정보	8바이트 (L2Header)
I 수행 후	출력 VPI/VCI 정보	4바이트

여준 것과 같이 Slicer에서 MAC으로의 성능 한계에 근접한 패킷 크기인 경우(88바이트, 136바이트, 184바이트인 경우)에 되돌림에 의해서 추가되는 정보량의 크기가 성능에 큰 영향을 미친다. 그러므로, 본 논문의 구현에서는 MAC 되돌림을 위해서 추가되는 정보량이 가장 적은 경우인 블록 D의 수행 후에 패킷을 되돌리는 방법을 사용하였다.

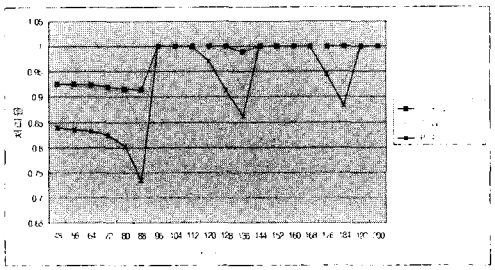


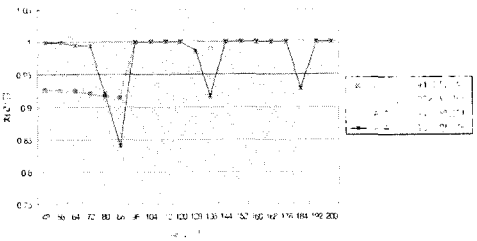
그림 6 되돌림에 의해 추가되는 정보량과 포워딩 엔진의 성능

4.3 수행되는 명령어의 개수에 의한 영향

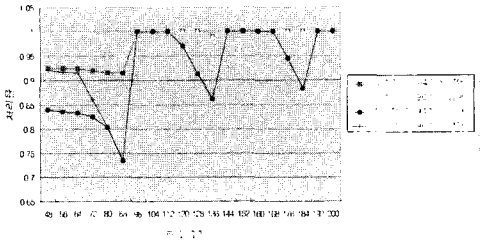
이더넷 패킷 프로세서에서 프레임 처리를 하기 위해서 수행되는 명령어의 수에 의한 포워딩 엔진의 성능에 미치는 영향에 대해서 측정하였다. MPLS 프레임, IP over ATM 프레임, PPP over ATM 프레임을 처리하기 위해서 이더넷 패킷 프로세서에서 수행되는 명령어의 수의 93~94개이다. 본 논문의 구현에서는 표 2에 기술된 패킷 처리 프로그램 중에서 블록 F부분의 목적지 주소가 NULL 또는 멀티캐스트 브로드캐스트를 검사하는 루틴을 제거하여 90이하의 명령어를 갖는 구조로 개발하였다.

그림 7에서 보는 것과 같이, 명령어의 수가 적을수록 성능이 향상된다. 그렇지만, 그림 6의 되돌림에 의한 정보량의 크기와 처리율과의 관계에서와 마찬가지로 Slicer에서 MAC으로의 성능 한계에 근접한 패킷 크기인 경우 88바이트, 136바이트, 184바이트일 경우 명령어 수가 적어지더라도 성능이 좋아지지는 않는다. 이유는 이더넷 패킷 프로세서 내부에 있는 SAR의 성능에 좌우되기 때문이다.

본 논문에서는 프로그램 가능한 이더넷 패킷 프로세서에 기반한 포워딩 엔진의 성능을 향상시키기 위해서 되돌림에 의해 추가되는 정보량이 없고, 90개의 명령어로 패킷을 처리하여 적절한 출력 포트로 패킷을 보낼 수 있도록 하여 그림 5에서의 최대 성능을 유지할 수 있도록 구현하였다.



(a) 되돌림 시점 D와 A인 경우



(b) 되돌림 시점 D와 B인 경우

그림 7 패킷 처리하는 명령어 개수와 포워딩 엔진의 성능

5. 결 론

포워딩 엔진은 라우터에서 패킷을 처리하는데 병목요소이며, 특히, MPLS LER과 같이 여러 형태의 다른 네트워크와 연동되는 경우에 포워딩 엔진은 프레임 처리에 유연성을 제공하여야 한다. 본 논문에서는 MPLS LER을 위해 프레임 처리의 유연성을 지원하기 위해서 프로그램 가능한 이더넷 패킷 프로세서에 기반한 포워딩 엔진을 구현하였다. 프레임의 되돌림에 포함되는 정보량을 줄이고 수행되는 명령어의 수를 줄임으로써 성능을 향상시킬 수 있다. 실험 결과에 의하면, 인터넷 상의 대부분의 IP 패킷의 크기가 64, 512, 1024인 경우 좋은 성능을 나타낸다. 구현된 포워딩 엔진은 기존 IP 네트워크와 인터넷 서비스를 위한 MPLS 네트워크에 적용하기 적합하다.

참 고 문 헌

[1] S. Keshav and R. Rharma, "Issues and Trends in Router Design," IEEE Communications Magazine, Vol. 36, No. 5, pp.144-151, May 1998.

[2] V. Kumar, T. Lakshman, and D. Stiliadas, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet," IEEE Communications Magazine, Vol. 36, No. 5,

pp.152-164, May 1998.
[3] C. Partridge and et al., "A 50Gbps IP Router," IEEE/ACM Transactions on Networking Vol. 6, No. 3, pp.237-238, Jun. 1998.
[4] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF RFC3031, Jan. 2001.
[5] A. Viswanathan, N. Feldman, Z. Wang, and R. Callon, "Evolution of Multiprotocol Label Switching," IEEE Communication Magazine", Vol. 36, No. 5, pp.165-173, May 1998.
[6] J. Aweya, "On the Design of IP Routers Part 1: Router Architectures," Journal of Systems Architecture, Vol.46, No.6, pp.483-511, Apr. 2000.
[7] F. Baker, "Requirements for IP Version 4 Routers," IETF RFC 1812, Jun. 1995.
[8] MMC Networks Co., "EPIF4-L3 Reference Manual," Oct. 1998.
[9] C-Port, C-5 Digital Communications Processors, 1999.
[10] IBM Co., IBM Network Processor, 1999.
[11] J. Heinanen, "Multiprotocol Encapsulation over ATM Adaptation Layer 5," IETF RFC1483, Jun. 1993.

박 재 형
1987년 3월~1991년 2월 연세대학교 전산학과(학사). 1991년 3월~1993년 2월 KAIST 전산학과(석사). 1993년 3월~1997년 8월 KAIST 전산학과(박사) 1997년 9월~1998년 11월 KAIST 인공지능연구센터 Post-Doc 연수연구원 1998년 12월~2002년 2월 ETRI 네트워크연구소 선임연구원. 2002년 3월~현재 전남대학교 전자컴퓨터정보통신공학부 조교수. 관심분야는 MPLS, 인터넷 프로토콜, 인터넷 시스템, 라우터 구조, 이더넷, 네트워크 연결망, 병렬처리, 멀티캐스트

김 미 희
1993년 3월~1997년 2월 이화여자대학교 전자계산학과(학사). 1997년 3월~1999년 2월 이화여자대학교 컴퓨터학과(석사). 1999년~현재 ETRI 네트워크연구소 연구원. 관심분야는 차세대 인터넷, MPLS/GMPLS, 인터넷 시스템, 기가비트 이더넷, 메트로 이더넷

정 민 영
1986년 3월~1990년 2월 KAIST 전기및전공학과(학사). 1992년 3월~1994년 2월 KAIST 전기및전자공학과(석사) 1994년 3월~1999년 2월 KAIST 전기및전자공학과(박사). 1999년 1월~2002년 2월 ETRI 네트워크연구소 선임연구원. 2002년 3월~현재 성균관대학교 정보통신공학부 조교수. 관심분야는 차세대 인터넷, 광/홈/Ad hoc 네트워크, 이동통신망 성능 및 신뢰성 분석

이 유 경
1974년 3월~1978년 2월 한국항공대학교 전자공학과(학사). 1978년 3월~1980년 2월 연세대학교 전자공학과(석사) 1980년 8월~1984년 3월 공군2사관학교 교관. 1995년~1996년 일본 NTT R&D 방문연구원. 1984년 4월~현재 ETRI 네트워크연구소 책임연구원. 관심분야는 차세대 인터넷, 네트워크구조, MPLS/GMPLS, TCP/IP, 광통신, 인터넷 라우터, ATM 시스템