2003년도 한국정보통신설비학회 하계학술대회 논문집 (2003.8.29-30)

네트워크 프로세서를 사용한 고성능 IP-DiffServ/IP-MPLS 시스템 구조

박준석, 이광용
LG전자 정보통신사업본부 기간망연구소

High Performance System Architecture for IP-DiffServ/IP-MPLS Using Network Processor

Joon-Seok Park, Gwang-Yong Yi
Infra Network Research Lab., Telecommunication Equipment & Handset Company, LG Electronics

Abstract - 본 논문에서는 Agere Systems사의 네트워크 프로세서를 사용한 고성능 IP-DiffServ/IP-MPLS 시스템의 구조를 제안하고 성능을 분석한다. 제안한 시스템은 기가비트 이더넷 뿐만 아니라 ATM과 POS 등 다양한 인터페이스를 제공하며 코어 및 에지 라우터로서 MPLS LER 또는 LSR로의 역할을 수행한다. 성능분석은 OPNET을 사용하여 시스템을 모델링한 후 입력 트래픽에 대해서 DiffServ 클래스별 지연시간과 지연의 주된 원인을 분석한다. 그리고 이를 바탕으로 시스템의 성능을 극대화할수 있는 네트워크 프로세서의 최적 파라메터를 도출한다. 성능분석 결과, 시스템이 각 서비스 클래스에 대해서 원활한 서비스를 제공하기 위해서는 프리미엄 서비스에는 최고의 우선순위를 부여하여 큐에 데이터 블럭들이 찰 때 마다 즉시 서비스해 주어야 한다는 것을 알 수 있었다. 그리고 트래픽이 특정 출구 라인카드로 몰리는 핫스팟이 발생할 경우 트래픽의 지연이 증가하게되는데 이 지연의 주요 원인은 출구 라인카드에서의 큐잉에 의한 것임을 알 수 있었다

1. 서 론

1990년대 후반까지 대부분의 라우터들은 컴퓨터 처럼 소프트웨어 기반으로 동작하는 구조를 가지고 있었다. 이러한 라우터들은 손쉽게 다양한 구성이 가능하고 서로 다른 버전의 운영체제에서도 동작할 수 있어 짧은 기간내에 라우터를 개발할 수 있다는 장점이 있다. 그러나 소프트웨어 기반 라우터는 고 대역폭에서 라인속도로 데이터를 처리하기가 힘들다.

IT기술의 수요가 세분화해 나감에 따라 특정한 응용 어플리케이션에서 더욱 나은 성능을 얻기 위해 기존의 범용 마이크로프로세서가 특화된 구조로 변형되는 추세이다. 네트워크 망이 점점 높은 대역폭을 갖게 되고 운용에 있어 응용력의 요구가 늘어남에 따라 이를 처리해 주기 위한 전용의 네트워크 프로세서가 새로이 출현하게 되었다. 즉, 네트워크 프로세서란 네트워크에서 패킷 처리를 위해 최적화된 마이크로프로세서를 지칭한다고 할 수 있다. 네트워크 프로세서가 수행하는 주요 기능으로는 헤더 파싱, 패턴 매칭, 비트 단위 조작, 패킷 순서 관리, 데이터 이동, 테이블 룩업, 패킷 수정 등 다양하다.

이에 본 논문에서는 Agere사의 네트워크 프로세서[1]를 사용한 대용량 고속의 IP-DiffServ/IP-MPLS[2-4] 시스템을 제안하고 성능분석을 통하여 시스템의 성능을 극대화할 수 있는 네트워크 프로세서의 최적 파라메터를 도출한다.

2. 본 론

2.1 Agere 네트워크 프로세서

Agere Systems사의Payload Plus 네트워크 프로세서는 크게 세 부분의 프로세서 구성요소로 이루어져 있다. 데이터 프로세싱을 수행하기 위한 네트워크 프로세서 구성요소로는 FPP(Fast Pattern Processor)와 RSP(Routing Switch Processor)가 있으며, 각 네트워크 프로세서 구성요소들의 인터페이스와 통계 정보 수집을 위하여 ASI(Agere System Interface)를 사용하고 있다. 각 네트워크 프로세서 구성요소들은 특화된 기능을 수행하는데 FPP는 룩업 관련 기능을, RSP는 큐잉에 관련된 기능을, ASI는 인터페이싱, 폴리싱(policing) 및 통계 정보 수집을 담당하고 있다

2.1.1 FPP

FPP는 PDU의 헤더 처리와 관련된 프로세서로 주된 기능은 룩업 및 패킷 분류이며 이를 위해서 FPP는 고유의 패턴 매칭 알고리즘을 사용하고 있다. FPP의 구조적 특징은 PDU 처리를 위하여, 첫 번째 패스(first pass)와 두 번째 패스(second pass)로 나뉘어져 있으며, 각 패스는 컨텍스트(context)라는 쓰레드(thread)를 사용하여 파이프라인 다중 프로세싱을 수행한다는 점이다. FPP에는 총 64개의 컨텍스트가 있으며 첫 번째 패스는 16개에서 최대 63개까지의 컨텍스트를 사용하며 두 번째는 1개에서 최대 48개까지의 컨텍스트를 사용할 수 있다.

첫 번째 패스에서는 FPP로 입력되는 데이터 스트림을 받아서 64바이트의 블럭으로 쪼갠다. 그리고 쪼개어진 블럭들은 다시 데이터 버퍼 제어부에 의해 패킷 블럭별로 재조립된다.

두 번째 패스에서는 재조립이 끝난 패킷 블럭들에 대해 큐 엔진이 다음 번에 서비스할 패킷 블럭들을 선택한다. 선택함과 동시에 블럭들은 룩업이 수행되고 또한 동시에 RSP로 전송이 이루어진다. 이때 FPP는 ASI로 폴리싱 기능 요청을 요구할 수 있으며, 요청을 받은 ASI는 폴리싱을 수행하여 그 결과를 FPP의 두 번째 패스 컨텍스트로 전달해 준다. 룩업이 끝나면 두번 째 패스는 전송 커맨드(transmit command)이라는 결과 패킷을 만들어서 패킷 블럭들과 함께 RSP로 전송을 하여 FPP 의 데이터 처리과정을 끝마친다. FPP의 구조 및 동작 흐름을 도식화하면 (그림 1)과 같다.

2.1.1 RSP

RSP는 큐잉과 관련된 데이터 처리를 담당하는 프로세서이다. 기본적인 구조는 버퍼 관리와 스케쥴링을 담당하는 부분이 있으며 입력된 패킷 블럭들의 헤더 수정 기능을 담당하는 부분이 있으며, FPP에서 보내오는 블럭을 다시 재조립하고 또 재조립된 패킷을 다른 PDU로 전환을 할 수 있도록 되어 있다

2.1.1 ASI

ASI는 FPP, RSP 그리고 다른 프로세서 혹은 로직간 인터페이스를 담당하는 프로세서이다. 일반적인 기능은 초기에 메인 프로세서로부터 FPP와 RSP의 설정 명령을 받아 이를 전달하는 기능과, 제어 메시지의 전달, 관리 메시지를 전달하고, FPP에서 수행하지 못하는 기능들을 대신 처리해 준다

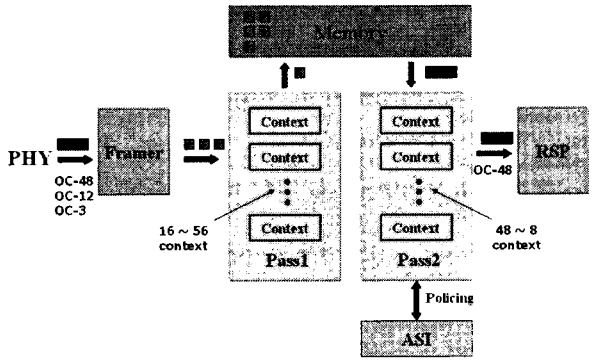

그림 1. 네트워크 프로세서(FPP) 구조 및 동작

2.2 IP-DiffServ/IP-MPLS 시스템

네트워크 프로세서를 사용한 IP-DiffServ/IP-MPLS 시스템의 전체적인 구성은 (그림 2)와 같이 네트워크 프로세서를 포함하는 진입 및 출구 라인카드와 스위치 패브릭으로 구성되어 있다. 진입 및 출구 라인카드에는 네트워크 프로세서를 포함하여 다른 많은 구성요소들이 있을 수 있으나 (그림 2)에서는 라인카드를 구성하는 일반적인 구성요소들은 생략하고 네트워크 프로세서의 구성요소만을 표기하였다. 본 논문에서의 스위치 패브릭은 진입 라인카드에서의 최대 트래픽을 패킷 유실이나 충돌 지연없이 출구 라인카드로 스위칭할 수 있는 충분한 대역폭을 가진 크로스바 패브릭으로 가정한다. 진입 및 출구 라인카드에서의 패킷 처리 과정에 대해서는 아래에 설명한다

2.2.1 진입 라인카드

진입 라인카드의 주요기능은 트래픽 분류(traffic classification)이다. 입력된 패킷의 헤더 필드를 보고 이 패킷이 속한 서비스 클래스를 분류해 낸다. 이를 위해 FPP의 두 번째 패스 컨텍스트는 다중필드 분류(multi-field classification)가 구현되어 있다. 일단 패킷 분류가 끝난 패킷에 대해서는 폴리싱을 수행하는데, 앞서 설명한 두 번째 패스 컨텍스트와 ASI사이의 폴리싱 알고리즘을 그대로 사용한다.

분류가 된 패킷들이 RSP에 입력되면, MPLS 프레임이 되기 위하여 레이블이 푸쉬(push)된다. 이를 위해 ROB라고 하는 큐 노드를 사용하였는데, ROB 노드에서는 입력된 패킷과 함께 FPP에서 전송된 전송 커맨드 패킷의 정보를 이용하여, 패킷에 푸쉬될 적절한 레이블을 찾아서 MPLS 프레임을 만들어 준다. 레이블 푸쉬가 끝난 패킷들은 다시 ROB에 의해서 각 출력 포트 큐로 보내진다.

Agere 네트워크 프로세서가 처리하는 기본 단위는 64바이트 혹은 48바이트의 블럭 단위이다. 따라서 블럭처리의 특성을 반영하기 위해 RSP에서는 MPLS 프레임을 64바이트 블럭으로 분할 하여 사용한다.

분류와 블럭 분할이 된 패킷 블럭들은 각 출력 포트 큐에 입력되어 차등 서비스를 받게 된다

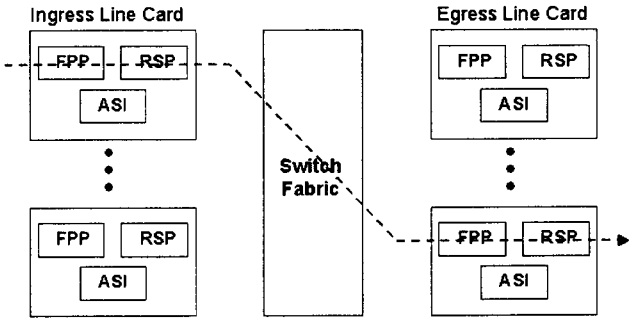

그림 2. 네트워크 프로세서를 사용한 IP-DiffServ/IP-MPLS 시스템 구조

2.2.2 출구 라인카드

출구 FPP에서는 스위치로부터 블럭들을 입력 받는데, 첫 번째 패스 컨텍스트에서는 이들을 다시 MPLS 프레임으로 재조합을 시킨다. 재조합을 위해 첫 번째 패스는 각 라인카드의 출력 포트에 해당하는 만큼의 입력 큐를 할당한다. 재조합이 끝난 MPLS 프레임에 대해서는 레이블 룩업을 실시하여, 이 MPLS 프레임이 속한 FEC(Forwarding Equivalence Classes) 찾는다.

출구 라인카드에서는 진입 라인카드와 달리 출력 큐에 MPLS 프레임 블럭들이 쌓이는 곳이므로 버퍼 관리를 수행한다. 이때 EF 서비스 클래스에 대해서는 무조건적인 서비스 보장을 위해 버퍼 관리를 수행하지 않고, BE와 AF 서비스 클래스에 대해서만 수행한다.

AF 서비스 클래스는 원칙적으로 3개의 드롭 우선순위(drop precedence)를 가지기 때문에 WRED알고리즘을 적용하여, 각 드롭 우선순위별로 서로 다른 손실률을 갖도록 하였고, 각각의 드롭 우선순위에 대해 최소 임계치(min threshold)를 서로 다르게 설정하는 방법을 사용하여 손실률을 조정하였다. 그리고 BE 서비스 클래스에 대해서는 일반적인 RED알고리즘을 사용하여 버퍼 관리를 구현하였다.

마지막으로 RSP의 각 출력 포트 큐에 대한 차등 지연 서비스는 진입 라인카드에서와 마찬가지로 두 단계 스케쥴링(two-level scheduling) 구조를 사용하여 서비스 하였다

2.3 성능분석

IP-DiffServ/IP-MPLS 시스템의 성능분석은 OPNET을 사용하여 수행하였으며 크게 FPP의 컨텍스트 배분에 따른 패킷의 처리시간 분석과 네트워크 프로세서로 구성된 전체 시스템의 트래픽 처리 능력을 분석한다

2.3.1 FPP의 컨텍스트 배분에 따른 첫 번째와 두 번째 패스의 처리시간

먼저 FPP의 총 64개의 컨텍스트 중 첫 번째 패스와 두 번째 패스에서 사용하는 컨텍스트의 수에 따른 처리 시간을 알아본다. 성능분석은 두 번째 패스의 컨텍스를 기준으로 8개에서 48개 까지 변화시키면서 수행하였다.

성능분석 결과, (표 1)에서와 같이 pass1과 pass2에서의 단위 블럭당 처리 시간은 각각에 할당된 컨텍스트 수에는 영향이 없음을 알 수 있다. 이는 입력되는 트래픽(블럭이나 패킷)을 실시간으로 처리할 수 있을 만큼의 컨텍스트 개수를 할당한다면, 즉 컨텍스트의 활용도를 90% 이하 정도로 유지한다면, 각 컨텍스트에서의 처리시간은 pass1 혹은 pass2에 몇 개의 컨텍스트가 할당되었는가에는 영향이 없음을 의미한다.

그리고 패킷의 길이가 길어질 수록 pass1의 처리시간은 줄어들고, pass2의 처리시간은 늘어남을 알 수 있다. pass1에서는 패킷이 블럭으로 쪼개어져서 처리되고, 한 패킷에서 첫 번째 블럭이 가장 긴 처리시간을 가지게 되므로 패킷 길이가 길어지면 pass1 컨텍스트에서 패킷의 첫 번째 블럭이 처리되는 빈도가 줄어들고 따라서 평균 처리시간이 줄어들게 된다. pass2에서는 쪼개어진 블럭들이 각 컨텍스트에서 패킷 단위로 처리되기 때문에 패킷 길이가 길이질 수록 한 패킷에 대한 평균처리시간이 늘어나게 된다

표 1. 컨텍스트 수 변화에 따른 지연시간

Packet Size : 200 bytes (4 blcoks)

pass 2 context	48	40	32	24	16	12	8	
pass1 util	34.175	22.783	17.088	13.670	11.392	10.515	9.764	%
pass2 util	7.725	9.270	11.588	15.450	23.175	30.900	46.350	
pass1 delay	0.797	0.797	0.797	0.797	0.797	0.797	0.797	usec/block
	0.992	0.992	0.992	0.992	0.992	0.992	0.992	
	0.895	0.895	0.895	0.895	0.895	0.895	0.895	
pass2 delay	2.376	2.376	2.376	2.376	2.376	2.376	2.376	usec/packet
	2.624	2.624	2.624	2.624	2.624	2.624	2.624	
	2.459	2.459	2.459	2.459	2.459	2.459	2.459	
FPP delay	3.669	3.669	3.669	3.669	3.669	3.669	3.669	usec/packet
	4.278	4.278	4.278	4.278	4.278	4.278	4.278	
	3.896	3.896	3.896	3.896	3.896	3.896	3.896	

시뮬레이션을 통하여 블럭당 소요되는 지연시간과 Pass1및 Pass2에 할당되는 컨텍스트의 수와는 무관함을 확인하였고 (그림 3)은 입력되는 패킷의 크기 변화에 따른 Pass1, Pass2 및 FPP의 평균 지연시간을 보인 것이다

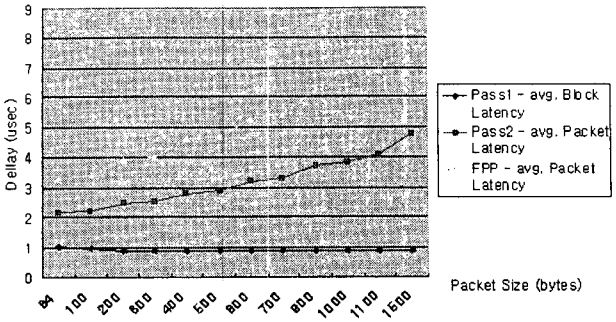

그림 3. 입력 패킷 크기 변화에 따른 FPP에서의 지연시간

2.3.2 서비스 클래스별 및 시스템 지연

(그림 4)와 (그림 5)는 각각 진입/출구라인카드에서의 각 서비스 클래스별 지연과 그리고 IP-DiffServ 시스템의 입력에서 출력까지의 전체 지연을 보여주고 있다. 결과 그림에서 가장 특이한 점은 바로 EF(Expedited Forwarding) 서비스의 지연이 오히려 AF(Assured Forwarding) 클래스의 지연보다도 크며, 심지어 BE(Best Effort) 서비스의 지연보다도 더 크게 나오는 것을 볼 수 있다. 그 이유는 EF 서비스 클래스와 AF 서비스 클래스를 서비스 해주기 위해 각각의 서비스 클래스에 할당한 서비스 가중치에 있다. 즉 EF 서비스 클래스는 그 특성상 인터넷 트래픽에서 차지하는 양이 적어 낮은 서비스 가중치가 할당되었기 때문에, EF 서비스 클래스 패킷 블럭들이 정확한 클럭에 맞추어서 입력되지 않으면 AF 클래스의 서비스 가중치에 밀려 계속 지연이 증가하게 되는 것이다 이를 해결하기 위해서는 EF 서비스 클래스에는 최고의 우선순위를 부여하여 큐에 블럭들이 도착하면 바로 서비스해 주도록 하여 EF 클래스의 지연을 최소화할 수 있다.

앞에서 문제가 되었던 EF 클래스의 지연을 개선하기 위하여 EF 서비스 클래스에 최고의 우선순위를 부여하여 EF 클래스 큐에 패킷 블럭들이 입력될 때마다 바로 서비스해 주도록 한다. 이렇게 함으로써 (그림 5)와 같이 EF 서비스 클래스의 지연이 개선되었음을 확인할 수 있다. 그리고 EF 서비스 클래스에 최고의 우선순위를 부여하여 서비스 해준다 하더라도 다른 AF 서비스 클래스에 전혀 문제가 발생하지 않음을 알 수 있다. 결과에서 보듯이 AF 서비스 클래스도 AF 각각의 서비스 클래스에 할당된 서비스 가중치에 맞도록 서비스 되고 있음을 확인할 수 있었다

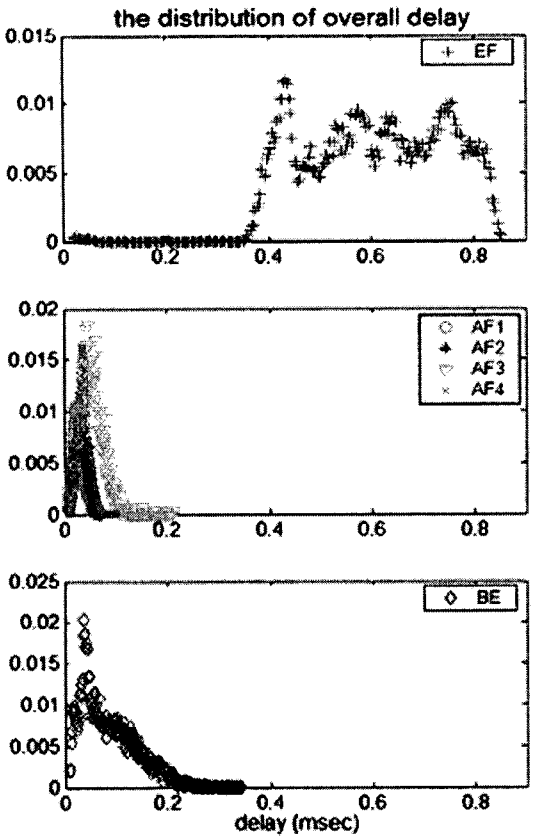

그림 4. 서비스 클래스별 지연

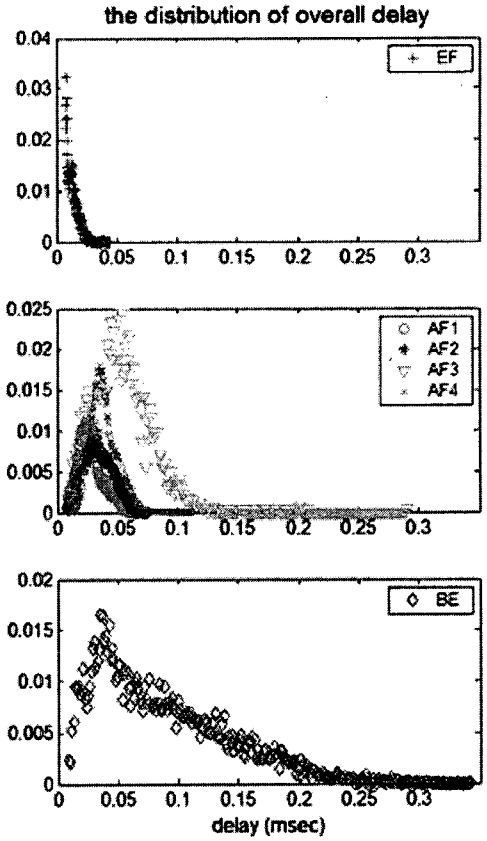

그림 5. 시스템의 전체 지연

3. 결 론

본 논문에서는 Agere 네트워크 프로세서로 구성되는 IP-DiffServ/IP-MPLS 라우터의 성능평가를 해보았다. 실험을 통해서 얻은 결과를 종합하면, IP-DiffServ/IP-MPLS 라우터가 각 서비스 클래스에 대해 제대로 된 서비스를 제공해 주기 위해서는, 일단 EF 서비스 클래스, 즉 프리미엄 서비스에는 최고의 우선순위를 부여하여 큐에 블럭들이 찰 때 마다 바로 서비스해 주어야 한다는 것을 알 수 있었다. 그리고 이렇게 했을 경우에도 다른 서비스 클래스에 영향을 미치지 않는다는 것을 확인할 수 있었다. 그리고 트래픽이 한쪽 입구 라인카드에 몰리는 핫스팟이 발생하는 경우, 트래픽의 지연이 증가하게 되는데 이때 지연의 주 원인은 출구 라인카드에서의 큐잉에 의한 것으로 밝혀졌다

[참 고 문 헌]

[1] Agere Systems, Payload Plus Data Book

[2] M. Yukscl, S. Kalyanaraman, "A strategy for implementing smart market pricing scheme on diff-serv," Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE , pp. 1430-1434, Vol. 2, Nov 17 - 21, 2002.

[3] Minghai Xu; Zhengkun Mi; Xiaofang Feng; Wenke Xie, "Implementation techniques of intserv/diffserv integrated network," Communication Technology Proceedings, 2003. ICCT 2003. Int'l Conference on , Vol. 1, pp. 231-234, April 9 - 11, 2003

[4] Chin-Ling Chen, "DiffServ operational model," Local Computer Networks, 2002. Proceedings. LCN 2002. 27th Annual IEEE Conference on, pp. 353-362, Nov. 6-8, 2002