# IXP1200 네트워크 프로세서를 이용한 Diffserv 설계

김경혜[0] 정영환 배국동 박우진 안순신
고려대학교 전자공학과
{rabear, youngh, gooibil, progress, sunshin}@dsys.korea.ac.kr

# The design of Diffserv in IXP1200 Network Processor

Kyunghye Kim[0], Younghwan Jung, Kukdong Bae, Woojin Park, Sun-Shin An
Computer Network Lab. Dept. of Electronics Eng., Korea University

**요 약**

현재의 인터넷은 트래픽의 양적 증가뿐 아니라 다양한 응용 수용이라는 질적인 측면의 문제에 봉착했다. 이를 수용하기 위해서 Diffserv 개념이 도입되었으며, 본 논문에서는 프로그램 가능한 네트워크 프로세서 IXP1200를 이용하여 간단한 Diffserv Interior-node PHB의 설계를 통하여 CoS(Class of Service)를 지원하는 Diffserv 를 설계하고 구현하였다.

## 1. 서 론

인터넷은 과거 수년 동안 비약적인 발전을 거듭했으며, 양적인 측면뿐만 아니라 질적인 측면에서 다양한 서비스들이 소개 되었다. 기존의 Mail, FTP, WWW 같은 응용은 Best-effort 서비스 형태에 적합하나, VoIP, Streaming Video, Audio 또는 Video Conferencing 등의 실시간 음성·영상의 전송 서비스를 지원하기 위해서는 기존의 인터넷 망은 부적합하였다. 그래서, IETF는 새로운 기술과 표준을 제안하고 개발하였는데 QoS를 보장하는 전송 형태의 Intserv와 Diffserv가 다양한 형태의 응용과 서비스들을 지원하도록 고안되었다. 그러나, 흐름별로 자원을 예약하는 Intserv는 좁은 서비스 영역에서는 잘 작동하나 응용마다 네트워크 자원을 할당 해야 하므로 처리량이 많아져 라우터에 부담을 주고, 넓은 영역에 서비스를 제공하기에는 부적절한 단점을 가지게 되었다. 그러므로, 클래스에 따라 트래픽을 처리하는 Diffserv가 제안되게 되었다. ISP 사이에서 서비스 도메인이 다를지라도 SLA에 따라 가입자와 서비스 제공자가 계약한 Profile을 검사하여 다른 도메인 상일지라도 같은 레벨의 차별화 된 네트워크 서비스 기능을 제공할 수 있게 되었다. 본 논문에서는 개발 기간을 단축시키고 다양한 응용이 적용 가능하게 하기 위해서 ASIC의 단점을 보완한 프로그램 가능한 인텔사의 IXP1200 네트워크 프로세서를 사용하여 Diffserv 기능을 구현하였다.

## 2. 관련 연구

### 2.1 IXP1200 네트워크 프로세서의 개요

인텔사에 의해 개발된 IXP1200 네트워크 프로세서는 프로그래밍이 가능한 6개의 32비트 RISC 마이크로 엔진과 Strong ARM 프로세서 코어를 탑재하고 있다. 6개의 마이크로 엔진은 패킷 포워딩을 담당하며 3계층에서 64 바이트 패킷을 초당 3백만개까지 포워딩 할 수 있다(약 1.5Gbps). 하나의 마이크로 엔진은 IXP 1200 Core Frequency (200MHz)에서 동작하고 모든 명령어는 한 사이클에 수행된다. 하나의 마이크로 엔진은 4개의 Context를 지원하고 전체 24개의 Context로 이루어져 있다.

Strong ARM 프로세서는 포워딩 테이블의 유지 및 보수, 망관리, 라우팅 프로토콜의 처리 등의 기능을 담당한다 [1][2].

### 2.2 IPv4 포워딩 엔진의 전체구성

포워딩 엔진 설계를 위한 코드의 전체구성은 Receive Scheduler, Receive Thread, Transmit Scheduler, Transmit Thread 모듈로 구성된다. 수신 부분은 입력포트에 수신된 64바이트 MAC 패킷을 RFIFO로 임시저장하고 헤더의 수정을 거친 후 완전한 패킷 출력을 위해 SDRAM에 저장하는 역할을 담당하며, 송신 부분은 SDRAM에 저장된 패킷을 TFIFO에 임시저장하고 출력포트로 패킷을 전송하는 역할을 담당한다. 그림 1은 포워딩 엔진의 전체구성을 보여 준다. Transmit Fill Thread를 부가하여 클래스 별로 구분된 큐에 저장된 패킷을 Transmit Algorithm을 사용하여 적절한 출력 port로 전송하는 기능을 담당하게 한다.

## 3. 설계 및 구현

### 3.1 Diffserv를 위한 Interior-node PHB의 전체 설계

본 논문에서 그림 2는 Diffserv 기능을 지원하는 Interior-node PHB 의 전체적인 구조를 보여준다. 총 6개의 큐를 두어 EF (Expedited Forwarding) 트래픽을 위해서 1개의 큐를 할당하고, AF(Assured Forwarding) 트래픽을 위해서는 4개의 큐를

BE(Best-effort Forwarding) 트래픽을 위해서는 1개의 큐를 할당한다. BA classifier(Behavior Aggregate Classifier)가 수신된 IP 패킷의 TOS에 마킹된 DSCP 필드를 보고 각 서비스에 해당되는 큐 위치에 각 패킷을 enqueue시키게 된다. WRR(Weighted Round Robin) 스케줄러는 총 6개의 큐에 enqueue된 패킷을 큐의 Weight 크기에 비례하여 송신하게 된다.

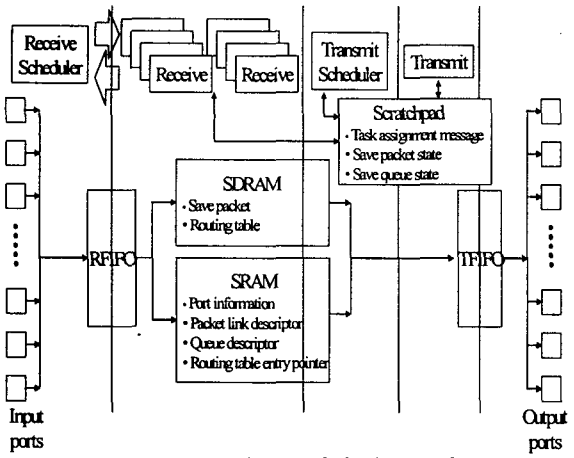


그림 1. 포워딩 엔진의 전체 구성

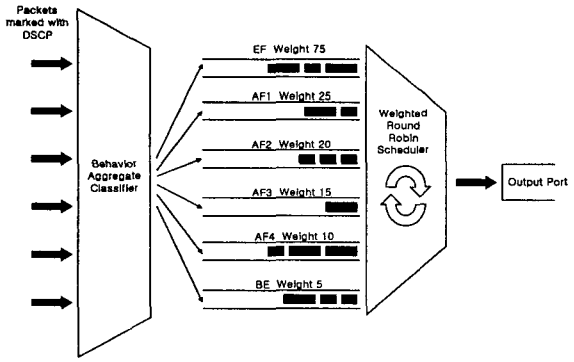


그림 2. Diffserv 구성의 개략도

### 3.2 IPv4 포워딩 기능의 개요

그림 2에서 설계된 Diffserv 기능을 지원하기 위해서 구현에 관련된 상세 설명을 하고자 한다. 본 논문에서는 Intel developer workbench 1.3을 사용하여 마이크로 코드 어셈블리 프로그래밍을 통해 L3 IPv4 포워딩 기능을 수정하였다.

표 1은 쓰레드 할당(Assignment)를 보여준다. ME는 마이크로 엔진을 나타낸다

Receive Thread-각 포트에 하나씩 할당되고 스케줄링 매커니즘이 불필요하다. 수신 모듈은 다음의 단계를 거치게 된다. 패킷의 수신 단계, 수신된 패킷의 검증 단계, 경로 설정 정보 참조 단계, 패킷 헤더의 수정 단계, 출력 포트 결정 단계, 클래스별 패킷의 분류 단계, 패킷의 enqueue 단계를 포함한다. Diffserv 기능을 지원하기 위해 BA classifier 기능을 첨부하고 트래픽 클래스 (예를들면, EF, AF, BE)에 해당하는 큐들을 마련하여 패킷이 클래스에 따라 enqueue 되도록 수정하였다.

Transmit Scheduler-Transmit Scheduler는 WRR 알고리즘에 따라 이때 PWP 벡터와 QWP 벡터의 정보를 읽어 들여 어떤 순서로 포트가 액세스 될 것인가를 결정한 후 그 포트의 어떤 큐에 있는 패킷이 송신될 것인가를 결정한다. Diffserv 기능을 지원하기 위해 WRR Scheduler를 부가하였으며, 포트마다 할당된 Transmit Thread에 큐를 할당하는 방법을 수정하였다.

Transmit Fill Thread-Transmit Scheduler로부터 전송된 작업 할당 메시지를 읽어 들여 할당된 포트로 패킷을 전송하는 역할을 한다. 각 포트에서 어떤 큐를 읽어 들여야 하는지에 관한 쓰래드 할당 read 동작을 수정하였다.

| ME | Thread | Usage |
|---|---|---|
| 0 | 0~3 | Receive Thread for slow port 0~3 |
| 1 | 4~7 | Receive Thread for slow port 4~7 |
| 2 | 8~11 | Receive Thread for fast port 8 |
| 3 | 12~15 | Receive Thread for fast port 8 |
| 4 | 16 | Transmit Scheduler for slow port 0~7 |
| | 17~19 | Transmit Fill Thread for slow port 0~7 |
| 5 | 20 | Transmit Fill Scheduler for fast port 8 |
| | 21~23 | Transmit Fill Thread for fast port 8 |

표 1. 쓰레드 할당

### 3.3 송신 큐의 구조

패킷이 수신되고, 송신되는 과정에서 송신 큐의 구조는 다음의 그림 3과 같다.

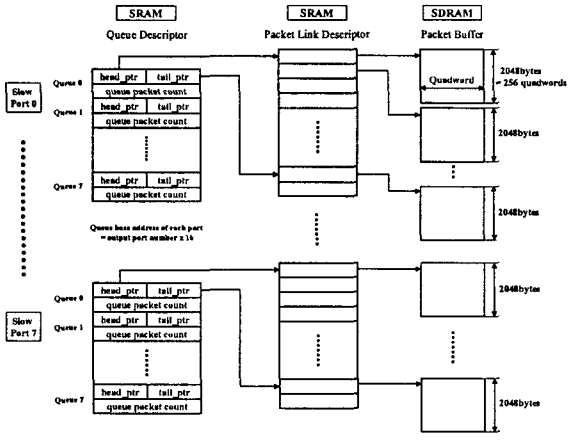


그림 3. 송신 큐의 구조

SDRAM에 저장되는 패킷의 최대 크기는 2K 바이트고 이 크기는 이더넷 프레임 최대크기(1528 바이트)를 넘지 않는다. 각 패킷은 마이크로 엔진에서 64바이트의 크기로 처리되고 저장된다. Packet Link Descriptor는 다른 Link Descriptor 및 Queue Descriptor와 함께 Linked List의 구조를 지니며 패킷의 정보를

포함하고 있다. 각각의 포트는 8개의 Queue Descriptor를 가지며 각각의 Queue Descriptor는 각각의 큐를 구성한다. 그러므로, 포트당 8개의 큐가 Diffserv의 클래스 레벨로서 사용될 수 있다.

### 3.4 수신 처리

Diffserv를 지원하기 위해서 수신된 IP 패킷의 TOS에 마킹된 DSCP 필드를 보고 해당 Q ID를 결정하고 CoS 레벨을 결정하는 BA classifier가 필요하다. 수신 처리에 관한 흐름도는 그림 4에 있다. 다음의 각 패킷의 수신 처리에 있어 필요한 단계에 대하여 이후에 설명한다.

패킷의 수신 단계 -2 계층 프로토콜을 처리
수신된 패킷의 검증 단계 -TTL과 checksum이 유효한지 체크(패킷 drop 여부 결정)
경로 설정 정보 참조 단계 -3 계층 프로토콜 처리로서 라우팅 테이블 참조하여 출력 포트를 결정(라우팅 결정)
패킷 헤더의 수정 단계-TTL 감소, 체섬 재계산, 목적지 MAC주소 업데이트
출력 포트 결정 단계
클래스 별 패킷의 분류 단계-수신된 패킷을 enqueue시키기 위해서 IP 패킷의 TOS에 마킹된 DSCP 필드를 보고 해당 Q ID를 결정
패킷의 Enqueue 단계
상위 단계에서 결정된 출력 포트와 Q ID에 해당하는 Queue Descriptor의 address를 계산한다.
Queue Descriptor와 Packet Link Descriptor를 업데이트 한다.
PWP와 QWP를 업데이트 한다.

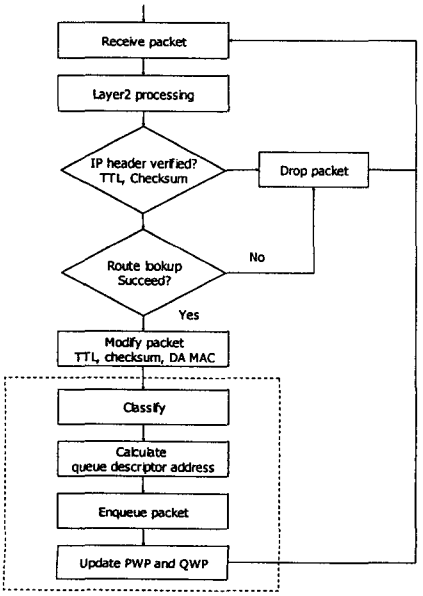


그림 4. 수신 처리 흐름도

### 3.5 송신 처리

송신 처리 과정을 살펴보면 Transmit Scheduler는 어떤 포트가 패킷을 가지는지 어떤 큐가 패킷을 가지는지 검사하기 위 SRAM에 있는 Queue Descriptor를 읽어 들인다. 그러나, 이것은 마이크로 엔진 사이클을 낭비하는 단점이 있다. 그러므로 메모리 참조 시간을 줄이고 성능을 향상하기 위해서 Scratchpad 메모리에 있는 PWP(Port With Packet)과 QWP(Queue With Packet) 벡터를 사용하였으며 Scratchpad 메모리 참조에 의해 소요되는 마이크로 엔진 사이클은 이상적으로 12 cycles으로 SRAM 참조에 의해 소요되는 17 cycles 보다 30% 적다. 모든 포트와 모든 큐에 패킷이 존재하는지 여부를 알려주는 PWP와 QWP는 Receive Thread가 패킷을 큐에 enqueue시키거나 Transmit Thread가 패킷을 송신할 때 업데이트 된다. PWP와 QWP의 구조는 그림 5 에서 보여준다.

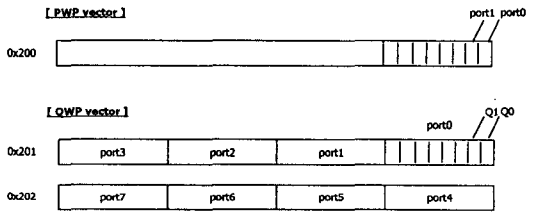


그림 5. PWP와 QWP의 구조

## 4. 결론 및 향후 연구

본 논문에서는 IXP 1200 네트워크 프로세서를 이용하여 단순한 형태의 Diffserv를 제공하는 Interior-node 기능을 설계하였다. BA classifier와 WRR 큐 스케줄링 알고리즘이 구현 방법으로 채택되었다. Diffserv 의 구현은 Receive Thread, Transmit Scheduler, Transmit Fill Thread 3부분으로 구성되었다.

향후 완전한 형태의 Diffserv를 지원하기 위해서 AF PHB의 drop 우선순위를 결정하는 매커니즘인 WRED(Weighted random early detection)와 같은 Drop Mechanism 를 첨가해야 할 것이며 더욱 최적화된 큐 스케줄링 알고리즘이 개발되어야 할 것이다.

## 5. 참고 문서

[1] Intel "IXP1200 Network Processor Family Microcode Programmer's Reference", June, 2001
[2] Intel "IXP1200 Network Processor Family Hardware Reference Manual", June, 2001
[3] Intel "IXP1200 Network Processor Family Microcode Software Reference Manual", June, 2001
[4] RFC 2475, "Architecture for Differentiated Services."
[5] RFC 2474, "Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 Headers."
[6] Manolis Katevenis, et al. "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip", Oct 1991