IXP1200 네트워크 프로세서를 이용한 DiffServ Edge Router의 설계

배국동° 박우진 정영환 김경혜 안순신
고려대학교 전자컴퓨터공학과
{gooibil°, progress, youngh, rabear, sunshin}@dsys.korea.ac.kr

The design of the DiffServ Edge Router on IXP1200 Network Processor

Kukdong Bae° Woojin Park Younghwan Jung Kyunghye Kim Sunshin Ahn
Computer Network Lab. Dept. of Electronics Eng., Korea University

요 약

급변하는 네트워크 서비스에 대한 요구에 신속히 대응하고 새로운 특징에 대한 시스템의 수정과 보완이 용이하도록 고안된 것이 네트워크 프로세서이다. 본 논문은 네트워크 관련 응용에 특화된 인텔의 IXP1200 네트워크 프로세서를 이용하여 Differentiated Service를 위한 간단한 DiffServ Ingress Boundary Node로서의 Edge Router를 설계한다.

1. 서 론

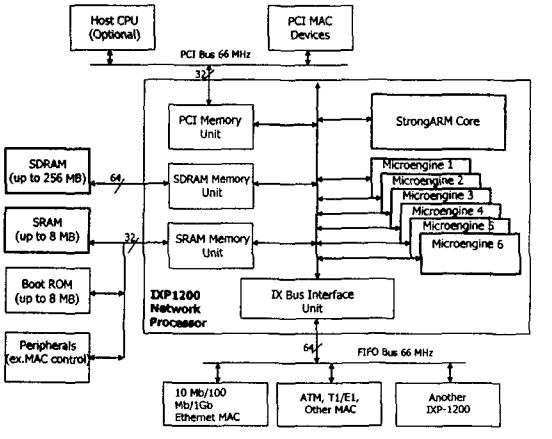
최근 수년간 네트워크 서비스들은 점점 복잡해지고 있으며, 이러한 서비스를 지원하기 위한 기존의 ASIC 기반의 방식은 여러 단점들을 지니고 있다. ASIC에 의한 하드웨어적인 방법은 고성능인데 반해, 장 시간의 제품 개발 기간과 일단 설계가 되면 수정이 불가능해서 새로운 특징들을 시스템에 추가하기 위해서는 많은 어려움이 뒤따른다. 이러한 단점들을 극복하기 위해 탄생한 것이 네트워크 프로세서이며, 이것은 프로그래밍이 가능한 소프트웨어적인 방법을 사용하므로 시장의 요구에 대해 빠르게 대응할 수 있으며, 새로운 특징에 대한 시스템의 수정과 보안이 용이하다. 또한, 집적회로 기술의 발전과 함께 성능이 계속 향상되고 있어 이러한 네트워크 프로세서를 이용한 네트워크 및 통신 장비의 보급은 날로 확대될 전망이다. 이러한 본 논문에서는 이러한 네트워크에 특화된 인텔사의 IXP1200 네트워크 프로세서를 이용하여 DiffServ Edge Router를 설계한다.

2. IXP1200 네트워크 프로세서의 개요

그림 1은 인텔사에 의해 개발된 IXP1200 네트워크 프로세서[1]의 하드웨어 구조를 보여주고 있다. 프로그래밍이 가능한 6개의 32비트 마이크로 엔진은 패킷 포워딩을 담당하며, 3계층에서 64바이트 패킷을 초당 3백만 개까지 포워딩 할 수 있다 (약, 1.5Gbps). 하나의 마이크로 엔진은 4개의 하드웨어 context (즉, thread)를 지원하며, IXP1200 코어 주파수인 200MHz에서 동작하고 모든 instruction은 한 사이클에 수행된다. 각각의 마이크로 엔진은 Programmable 1K Instruction Control Store (Program Memory)를 지니고 있으며, 이곳으로 Microprogram이 로딩된다. 각각의 마이크로 엔진은 128개의 32비트 General Purpose Register와 128개의 32비트 Transfer Register를 포함한다. 그리고 한 사이클에 ALU와 Shift연산을 수행할 수 있는 ALU와 Shifter를 포함한다

StrongARM Core 프로세서는 전체 시스템의 초기화를 관장하며 부분적인 패킷 처리 및 라우팅 테이블의 유지 및 보수, 라우팅 프로토콜의 처리, 망관리 등의 기능을 담당한다.

2Mbyte SRAM은 라우팅 테이블과 패킷 포워딩을 위한 큐에 대한 포인터를 저장하고 있으며, 24Mbyte SDRAM은 네트워크 프로세서가 처리해야 할 패킷들을 저장하는 버퍼 역할을 한다.



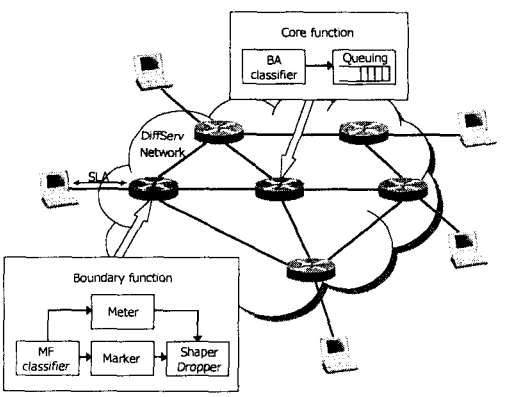
<그림 1> IXP1200 네트워크 프로세서의 구조

IXP1200 네트워크 프로세서의 패킷 포워딩 절차는 다음과 같다. 초기 부팅시 StrongARM은 Boot ROM 또는 serial/Ethernet 연결로부터 부트 이미지를 로딩하고 라우팅 테이블을 SRAM으로, 마이크로코드를 마이크로 엔진으로 각각 로딩하는 등 하드웨어 컴포넌트들을 초기화하고 나면 라우터는 패킷을 받을 준비가 된다. Ready Bus Sequencer가 MAC에 패킷이 들어왔음을 감지하면

해당 receive thread에게 알려주어 각각의 IP 패킷을 64byte의 MAC packet (MPKT) 단위로 끊어서 RFIFO에 순서대로 저장한다. 라우팅 테이블 참조 후 패킷의 IP 헤더를 수정 (즉, 헤더 검증, TTL 감소, 체크섬 재계산, 목적지 MAC 주소 변경)하고 포워딩을 위해 SDRAM의 큐에 저장하고 나면, 또다른 마이크로 엔진의 transmit thread는 TFIFO를 통해 다른 MAC으로 패킷을 포워딩하게 된다.

3. Differentiated Service의 구조

Differentiated Services[3]는 다양한 종류의 서비스를 제공함에 있어 공급자로부터 사용자에게 차별화된 레벨의 서비스를 통해 QoS를 제공하기 위한 방식이다. DS domain내로 들어오는 트래픽은 upstream 네트워크와 downstream DS domain 사이에 Service Level Agreement (SLA)라는 사전에 협의된 규약에 의해 차별화된 서비스를 받게 되는데, SLA는 packet classification, re-marking rule 및 특정 트래픽에 대한 profile을 규정하고, in- 또는 out-of-profile에 대한 처리 방식 등을 명시한다. 그림 2는 DiffServ node의 구조를 보여주고 있다.



<그림 2> DiffServ Boundary/Core Node의 구조

3.1 Classifiers

DS domain의 Ingress boundary node로 들어온 패킷은 Multi-field (MF) classifier를 통해 IP 헤더의 여러 필드 중 하나 이상의 조합 (예를 들어, source address, destination address, DS field, protocol ID, source port, destination port, incoming interface)을 기준으로 classification될 수 있다[3]. 반면, core node에서는 Behavior Aggregate (BA) classifier를 통해 오직 DiffServ codepoint (DSCP)만을 참조하여 classification한다.

3.2 Traffic Conditioners

Traffic conditioner는 기본적으로 Meter, (re)marker, shaper 그리고 dropper 4가지 요소로 구성된다. Meter는 트래픽 profile에 근거하여 들어온 트래픽을 측정하며, 이를 토대로 각각의 트래픽은 적절하게 marking, dropping 또는 shaping 된다.

4. DiffServ Edge Router 설계

4.1 Multi-field (MF) Classifier

DS domain의 ingress boundary node에서 MF classifier를 이용할 때, classifier는 IP 헤더의 source address, destination address, DS field, protocol ID, source port, destination port 그리고 선택적으로 incoming interface의 조합을 classification의 기준으로 사용한다.

이러한 패킷을 classification하는 기준으로서 SLA lookup table[5]을 생성하여, 이를 토대로 서비스 공급자와 사용자 간에 사전 약속된 profile을 매핑시키도록 한다. 그림 3은 SLA lookup table의 내용으로서 RFC 2475에서 언급한 6가지 필드 중 전부 또는 선택적으로 사용하여 간단한 classification 기준으로 사용할 수 있으며, 계약된 SLA를 충족시키기 위해 테이블 내의 항목들은 추가 또는 삭제 할 수 있다.

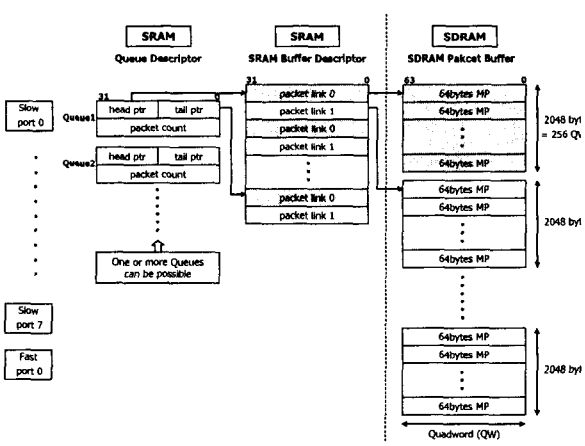
Src IP address	Dest IP address	DSCP	Src port	Dest port	Incoming interface
163.152.0.0	211.217.38.X	001011	4050	5004	-
		. . .			

<그림 3> MF Classifier의 SLA lookup table

예를 들어, source IP address가 163.152.0.0 이면서 오디오나 비디오와 같은 real-time 데이터 (UDP port number of RTP : 5004)를 전송하고자 하는 사용자에게 차별화된 서비스를 제공하고자 한다면, 그림3과 같이 사용자와 계약한 사항을 사전에 정의해 놓은 SLA lookup table을 탐색하여 관심있는 2가지 필드 (즉, Src IP address, Dest port)를 비교함으로써 ingress boundary node로 들어온 트래픽을 특정 클래스로 classification하고, 이에 해당되는 DSCP로 적절히 marking할 수 있다.

4.2 Queue Management & Algorithm

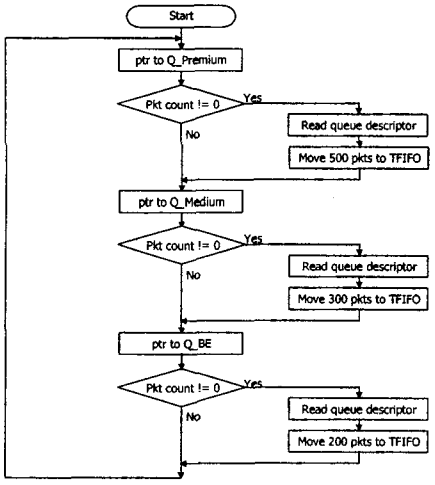
Classifier를 거친 패킷들은 해당 DSCP로 marking 또는 remarking되어 큐에 저장된다. 8개의 slow port와 1개의 fast port로 들어온 패킷은 SDRAM에 있는 큐에 저장되고, 패킷이 실제 저장된 SDRAM의 위치 정보는 SRAM의 Queue descriptor에 기록된다. Queue descriptor에는 받은 패킷의 시작과 끝을 알리는 포인터 정보와 MPKT의 개수에 대한 정보가 들어있으며, 각각의 패킷은 SRAM Buffer descriptor의 linked list의 구조로 유지된다. 그림 4는 하나의 포트당 하나의 큐의 버퍼(메모리) 할당 관계를 보여주고 있다.



<그림 4> 포트당 Transmit Queue의 구조

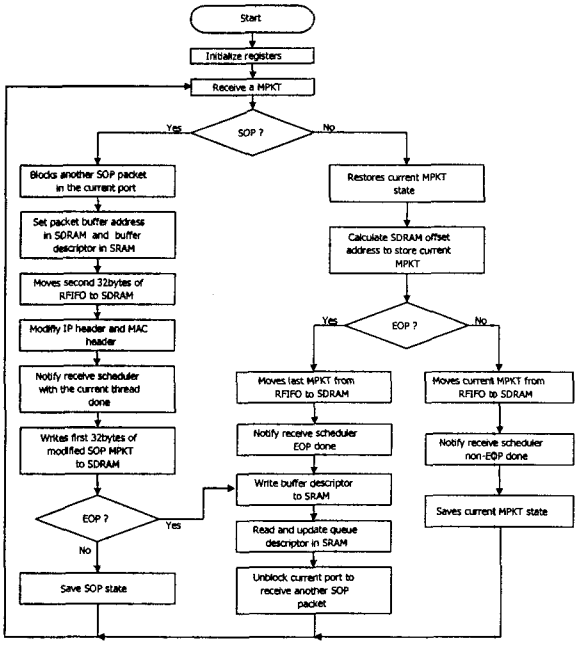
큐잉 알고리즘으로서는 Weighted Fair Queueing (WFQ)를 사용하였다. WFQ는 소량의 트래픽이 대량의 트래픽에 의해 잠식당하지 않도록 플로우 별로 큐를 두어 트래픽을 조절하는 공정성 측면과 더불어 특정 기준에 따라 가중치(weight)를 정하고 이에 따라 같은 양의 트래픽을 가진 플로우 간에도 차별을 두는 가중치 측면 또한 고려한 큐잉 방식이다. 큐간 기근현상(starvation problem)이 발생하지 않는다는 장점과 함께 상기 시스템에 구현하기 적합하다 판단되어 WFQ를 적용하였다.

그림 5는 WFQ를 이용한 간단한 큐 스케줄링 알고리즘을 보여주고 있다. Classification된 패킷은 SLA lookup table 참조 후 3가지 레벨 (예를 들어, Premium, Medium, Best Effort)로 분류되고, 각각을 위한 3개의 FIFO 큐를 두어 이곳에 해당 패킷들이 저장된다. WFQ를 적용한 알고리즘의 개요는 다음과 같다. 높은 우선 순위를 갖는 큐인 Q_Premium부터 Queue descriptor를 체크하여 수신된 MPKT이 있는지를 확인하고, MPKT이 있으면 처리하고 없으면 다음 순위의 큐인 Q_Medium의 Queue descriptor를 체크하고 처리한다. 이런 식으로 3개의 큐를 한번씩 돌아가며 반복 체크하되, 처리해야 할 수신된 MPKT이 존재할 때는 각 큐별로 한번에 처리해야 할 양에 대해 차별(예를 들어, 5:3:2)을 두어 TFIFO로 포워딩한다.



<그림 5> WFQ를 이용한 Queue Scheduling Flow Chart

RFIFO로 수신된 MPKT을 Receive Thread가 처리하는 절차는 다음과 같다. 해당 포트로 Start of Packet (SOP)가 수신되면 그 포트로는 또 다른 패킷의 SOP가 수신되지 못하도록 막고 (세마포어 이용), End of Packet (EOP)를 수신하여 하나의 완전한 IP 패킷에 대한 처리가 끝난 후에야 새로운 SOP를 받아들 일 수 있게 함으로써 패킷의 순서를 유지한다. (참고로 하나의 IP 패킷이 64byte의 MPKTs으로 쪼개지면 SOP + Not SOP/EOP + ... + EOP로 구분된다). 그림 6은 Slow Port에서의 MPKT 처리 알고리즘을 보여주고 있다. Fast Port에서의 알고리즘은 Slow Port에서와 유사하나, 하나의 Receive Thread 대신 여러 Receive Threads에 의해 MPKT이 처리되므로 이를 관리하기 위해 마이크로 엔진들이 공용으로 사용하는 영역 (Scratchpad)에 상태 정보를 기록하여 마이크로 엔진들간 동기를 맞추는 루틴이 추가된다.



<그림 6> Slow port에서 MPKT 처리 알고리즘

5. 결론 및 향후 연구

본 논문에서는 IXP1200 네트워크 프로세서를 이용하여 간단한 DiffServ ingress boundary node를 설계하였다. 메모리 크기 (2Mbyte SRAM)의 제약을 고려한 설계로 인해 traffic conditioning 기능의 상세한 부분은 설계의 대상에서 제외되었으며, 시뮬레이션을 통한 시스템의 성능 평가나 기대 효과는 다음 논문에서 발표할 예정이다.

현재까지 추상화 한 설계를 토대로 실제적인 구현후 성능 측정 및 향상을 통해 최적의 시스템을 설계하고, 보다 많은 기능들을 수용할 수 있는 자원의 필요성을 인식하고 하드웨어 업그레이드를 통해 실질적인 QoS를 제공하도록 구현해야 하는 과제가 남아있다.

6. 참조 문헌

[1] IXP1200 Hardware Reference Manual, Intel, June 2000.
[2] IXP1200 Programmer's Reference Manual, Intel, May 2000.
[3] S.Blake, D.Black, M.Carlson, E.Davies, Z.Wang and W.Weiss, "An Architecture for Differentiated Services", RFC 2475, Dec 1998.
[4] Ying-Dar Lin, Yi-Neng Lin, Shun-Chin Yang and Yu-Sheng Lin,"DiffServ over Network Processors: Implementation and Evaluation", High Performance Interconnects, 2002. Proceedings. 10th Symposium on, pp. 121-126.
[5] Daeho Kim and Junkyun Choi "Multi-functional Boundary Node System Architecture in DiffServ Network for Voice over IP", ICACT '2002, Feb 2002
[6] Zheng Wang, "Internet QoS: Architectures and Mechanisms for Quality of Service", Morgan Kaufmann Publishers