네트워크 프로세서에 적용 가능한 클래스 별 레이트 제한 기법

노 진 택[†] · 이 진 선[††] · 최 경 희[†††] · 정 기 현[††††] · 임 강 빈[†††††]

요 약

본 논문은 기존의 범용 시스템 또는 전용 하드웨어 기반의 네트워크 시스템에서 사용하던 레이트 제한(rate limiting) 기법과 클래스 별 대역폭 관리 기법을 기가 비트 트래픽 처리를 위한 네트워크 프로세서에서 구현하기 위한 방안을 제시하고 이를 구현하여 실험하였다. 구현과 실험은 인텔사의 IXP1200 네트워크 프로세서에서 이루어졌으며 그 결과로서 의도한 대역폭으로 제한된 트래픽 레이트의 정확도와 변화하는 입력 레이트에 대한 대역폭 제한 알고리즘의 안정화 시간을 보여 주고 있다. 이를 통하여, 네트워크 프로세서에 적합하도록 구현된 클래스 별 레이트 제한 기능이 일반 시스템에서의 토큰버킷 알고리즘의 오차범위 10%에 근접한 성능으로 잘 동작하는 것을 확인하였다.

키워드 : 네트워크 프로세서, 대역폭, 레이트 제한, 트래픽 클래스

A class-based rate limiting method applicable to the network processor

Jintaek Noh[†] · Jinsun Lee[††] · Kyunghee Choi[†††] · Gihyun Jung[††††] · Kangbin Yim[†††††]

ABSTRACT

This paper proposes an implementation methodology of the rate limiting method and the class-based bandwidth management for the gigabit-powered network processor, which are used on general purpose or ASIC systems in order to efficiently manage network bandwidth. Implementation and experiments are done on Intel's IXP1200 network processor. The result shows the accuracy of limited bandwidth and settling time of the estimator against the dynamic traffic rate. Through the results, this paper proves the proposed method and implementation properly work as expected.

Key Words : Network Processor, Bandwidth, Rate Limiting, Traffic Class

1. 서 론

인터넷이 널리 보급되면서 다양한 서비스들이 출현하였고 각종 서비스의 사용에 대한 비용은 선로의 대역폭에 따라 결정되고 있다. 따라서, 비용에 따른 합리적인 서비스를 제공하기 위해서는 선로의 대역폭을 정확히 지켜주는 것이 필수적이다. 비용에 대한 문제 이외에도 대역폭을 정확히 할당해 주는 능력은 서비스 제공자로 하여금 보다 효율적이고 융통성 있는 대역폭 할당 능력을 갖도록 한다[14].

서비스 비용에 따른 대역폭을 할당하는 시스템을 설계하기 위해서는, 먼저 정책에 의하여 트래픽을 구분해 낼 수 있는 능력을 가져야 하고, 다음으로는 정책에 따라 결정된 대역폭을 정확히 분배해 주는 능력을 가져야 한다. 즉, 트래픽의 특성에 따라 클래스를 구별하고, 각 클래스에 배분된 대역폭을 보장해 주는 레이트 제한의 개념이 필요하다.

그 중에, 레이트 제한 기법에 대해서는 그 동안 많은 연구가 진행되어 왔다[1, 2, 3, 4, 5]. 많은 알고리즘들이 지정된 대역폭을 지키기 위해 확률 연산 등을 통하여 네트워크의 상태를 예측하고, 예측된 결과를 지정된 대역폭 한계 내에 있도록 큐의 크기를 조절하여 트래픽의 양을 조절한다. 여기서 확률 연산을 통한 네트워크 상태의 예측 과정은 레이트 제한을 위한 핵심이다.

한편 네트워크 서비스의 지속적인 다양화와 복잡화는 네트워크 장비의 고성능화를 요구하였으며 네트워크 프로세서라는 새로운 구조를 요구하게 되었다[13]. 네트워크 프로세서는 범용프로세서의 융통성과 전용하드웨어 기반 시스템의 성능을 장점으로 가지는 시스템으로 네트워크의 트래픽 처리 기능 중 최하위 부분은 하드웨어적으로 구현하고 나머지 처리

※ 본 논문은 2004학년도 순천향대학교 산업기술연구소 학술연구조성비 일반연구과제 및 과학기술부 국가지정연구실사업의 지원으로 연구되었음
† 준 회 원 : 삼성전자 연구원
†† 준 회 원 : 아주대학교 정보통신전문대학원 석사과정
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
†††† 정 회 원 : 아주대학교 전자공학부 교수
††††† 정 회 원 : 순천향대학교 정보보호학과 교수
논문접수 : 2004년 12월 15일, 심사완료 : 2005년 8월 8일

는 패킷 처리를 위한 명령어만을 가지는 패킷 처리 프로세서를 프로그램하여 담당하도록 설계되어 있다.

네트워크 프로세서를 이용한 네트워크 응용에서 상기한 레이트 제한을 구현하기 위하여는 프로세서의 특성을 반드시 고려하여야 한다. 레이트 제한에서의 확률 연산은 네트워크의 상황을 예측하기 위한 필수적인 과정이므로 반드시 구현되어야 하나 그 계산 과정을 보면, 지수적 연산을 하는 것이 일반적이어서 고속의 트래픽 처리를 위하여 특화된 네트워크 프로세서를 위한 알고리즘으로는 적합하지 못하며 그대로 구현하려면 네트워크 프로세서로서의 장점을 잃게 된다.

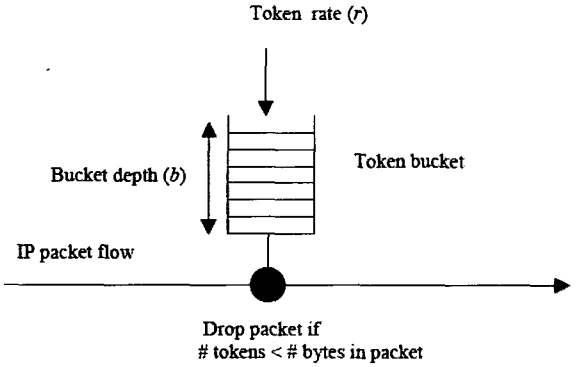
본 논문에서는 기가비트급 트래픽 처리를 위한 네트워크 응용에 특화된 네트워크 프로세서 시스템 상에서 보다 간단한 연산을 통하여 패킷 처리 성능의 저하를 최소화하면서 레이트 제한 기법을 구현함으로써 효율적인 대역폭을 관리하기 위한 방안을 제안하고 이를 실제의 시스템에서 구현한다. 논문에서 제안하는 방안을 구현하고 성능을 테스트하기 위하여 네트워크 프로세서의 일반적 특성을 고루 갖추고 저가 시스템 구현이 가능한 인텔의 IXP1200 프로세서를 사용하였다.

본 논문의 구성은 다음과 같다. 우선, 제2장에서 레이트 제한 방안에 대하여 설명하고, 제3장에서는 네트워크 프로세서에서 레이트 제한 기법의 구현 방안에 대하여 서술하며, 제4장에서 제안한 방안에 대한 실험과 그 결과에 대하여 분석하고, 마지막으로 제5장에서 제안 방안에 의하여 구현된 시스템의 장단점 등을 논함으로써 결론을 내린다.

2. 레이트 제한 방안

2.1 토큰 버킷 알고리즘

토큰 버킷 알고리즘은 패킷 전송시 일정한 전송율을 유지하기 위한 알고리즘으로, 일정한 시간마다 '버킷'에 쌓이는 '토큰'을 제어함으로써 전송율을 제어한다. (그림 1)은 토큰 버킷 알고리즘을 보여준다. 그림에서 token rate 'r' (byte/s)과 bucket depth 'b' (byte)은 각각 토큰의 최대 전송률과 버킷에 쌓일 수 있는 최대 크기를 의미한다. 일정한 전송률 r을 가지는 트래픽이 존재할 때, 전송률에 따라 토큰이 버킷에 쌓이게 되며 이 버킷에 일정한 b만큼의 토큰이 쌓이게 되면 토큰은 더 이상 쌓이지 않게 된다. 이 때, 패킷이 도착하고 패킷의 양만큼의 토큰이 남아있을 경우 이 패킷은 포워딩 된다. 반대로, 패킷의 양만큼의 토큰이 남아 있지 않을 경우 이 패킷은 폐기된다. 이에 따라, 버킷에 쌓이는 전송률 이상으로 패킷이 들어올 경우에는 패킷이 폐기되어 정해진 전송률을 지키게 되는 것이다.

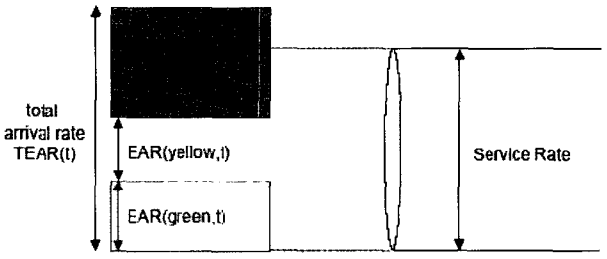


(그림 1) 토큰 버킷 알고리즘

2.2 Rate-Based n-RED

일반적으로 RED(Random Early Detection)는 혼잡 상황을 피하기 위해 큐의 점유율에 따라 확률 계산을 통해 패킷을 폐기함으로써, 전체적인 네트워크의 성능을 향상시키는 알고리즘이다. 이 알고리즘의 핵심 원리는 네트워크의 각 노드가 패킷의 손실에 능동적으로 대처한다는 것이다. 즉, 패킷 손실이 발생하면 해당 노드는 혼잡 상황으로 인식하고, 혼잡을 피하기 위한 동작을 하게 되는데, 이 동작이 자주 나타나게 되면 트래픽 레이트의 변화도 심해지고, 네트워크의 상태도 좋아지지 못한다. 따라서, 혼잡 상황을 제어함으로써, 혼잡 상황을 가능한 회피할 수 있고, 혼잡 상황이 발생하더라도 트래픽 레이트의 불안정한 변화를 줄일 수 있게된다.

n-RED는 RED의 변형으로써, 한 개 이상의 트래픽 클래스에 대해서 RED를 적용시키는 것이다. Rate-Based n-RED의 핵심 아이디어는 각 트래픽 흐름에 대한 장기적인 도착 레이트 EAR(t)를 추정하는 것이다. 이때 추정된 총 도착 레이트(TEAR(t) : Total Estimated Arrival Rate at time t)가 큐가 서비스할 수 있는 서비스 레이트 R보다 크면 패킷은 폐기되는데 그 결정은 다음의 식에 의하여 이루어진다.



(그림 2) Rate Based n-RED의 기본 아이디어

$$P_{drop} = \max(0, \frac{TEAR(t) - R}{TEAR(t)}) \quad (1)$$

각 트래픽 흐름의 EAR(t)가 합리적으로 추정되어야만 가변적인 트래픽 레이트에 대해 적절히 대응하는 레이트 제한 기법의 성능을 보여줄 수 있는 것이다. 참고문헌 [1]에서는 다음과 같은 수식을 통하여 도착 레이트를 추정하였다.

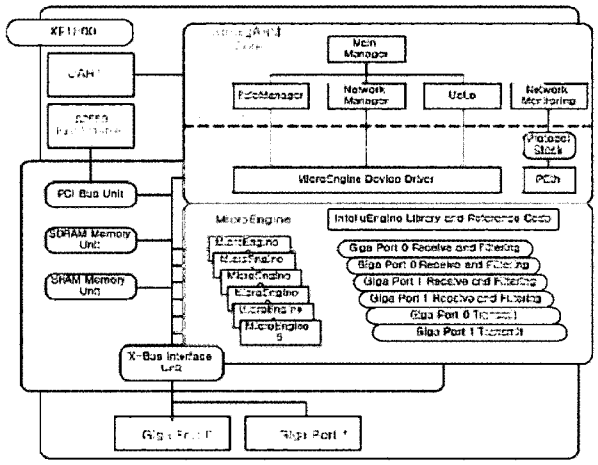
$$EAR(T) = (1 - e^{\frac{-T}{K}}) \times \frac{L}{T} + e^{\frac{-T}{K}} \times EAR(t_{prev}) \quad (2)$$

여기서 T는 $t-t_{prev}$, K는 0.1, 0.5, 1 등의 상수, L은 패킷

크기를 의미하며 추정된 값은 K가 작아질 수록 현재의 상태에 민감하게 반응할 것이다. 반면에 K가 커질수록 레이트의 급작스런 변화에도 천천히 따라가게 된다.

3. 네트워크 프로세서에서의 레이트 제한 방안

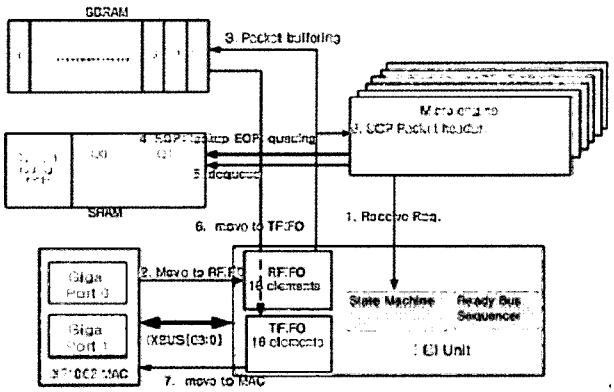
논문에서 사용한 네트워크 프로세서는 인텔사의 IXP1200로서 RISC 프로세서 기반의 다중 프로세서 구조를 가지고 있다. 이는 범용 StrongARM 프로세서를 가지며, 네트워크 데이터 처리를 위해 6개의 마이크로 엔진을 가진다. 6개의 각 마이크로 엔진은 독립적으로 각각의 프로그램을 수행할 수 있으며, 공유 메모리를 사용하는 효율적인 통신 방법을 제공하고 있다. 특히 메모리 접근에 대한 지연을 오버헤드가 없는 하드웨어 쓰레드들을 이용하여 보상할 수 있어 이 구조를 적극 활용하면 성능 개선에 도움이 될 수 있다. 또한, IXP1200은 최대 4.2 Gbps를 지원하는 별개의 네트워크 버스(IX bus)를 사용하므로 버스에 대한 병목현상이 없다. (그림 3)은 본 논문에 사용하는 시스템의 모델로서 32MB SDRAM과 8MB SSRAM을 이용하며, 2개의 기가비트 이더넷 포트를 지원한다.



(그림 3) 네트워크 프로세서 IXP1200의 구조

(그림 4)는 IXP1200에서 일반적인 패킷의 흐름을 순서대로 보여주고 있다. MAC에서 수신된 패킷은 수신을 담당하는 마이크로 엔진의 패킷 수신 요청에 따라 수신 FIFO로 이동하게 되고 다시 SDRAM에 버퍼링 된다. 송신을 담당하는 마이크로 엔진은 패킷을 SDRAM에서 송신 FIFO로 이동시키고, 마지막으로 MAC에 전달된다. 본 논문의 시스템 모델에서는 각 기가비트 이더넷 포트당 수신을 위한 2개의 마이크로 엔진과 송신을 위한 1개의 마이크로 엔진을 사용한다. 이때, 송신 마이크로 엔진 쓰레드들은 SRR(Static Round-Robin) 스케줄링 방식에 따라 패킷을 병렬처리 하게 된다. IXP1200은 하나의 패킷을 64바이트 단위(MPKT)로 나누어 처리하게 되는데, 이는 다시 SOP(Start Of Packet), MOP(Middle Of Packet), EOP(End Of Packet)로 분류된다.

본 패킷 분류 알고리즘에서 중요시되는 데이터는 SOP에



(그림 4) 네트워크 프로세서 IXP1200에서의 패킷 흐름

담겨있는 패킷 헤더 정보이다. 처리할 데이터가 SOP인 경우 라우팅 테이블과 룰 검색(SRAM)을 위해 SOP 일부가 마이크로 엔진으로 이동하여 prefix를 추출하고 규칙 검색을 위해 메모리에 있는 테이블을 참조한다. 이러한 패킷 처리에 관한 모든 작업은 주로 수신 마이크로 엔진에서 발생하므로 시스템의 성능은 이 엔진의 작업 수행 방법에 의한 영향을 받게 된다.

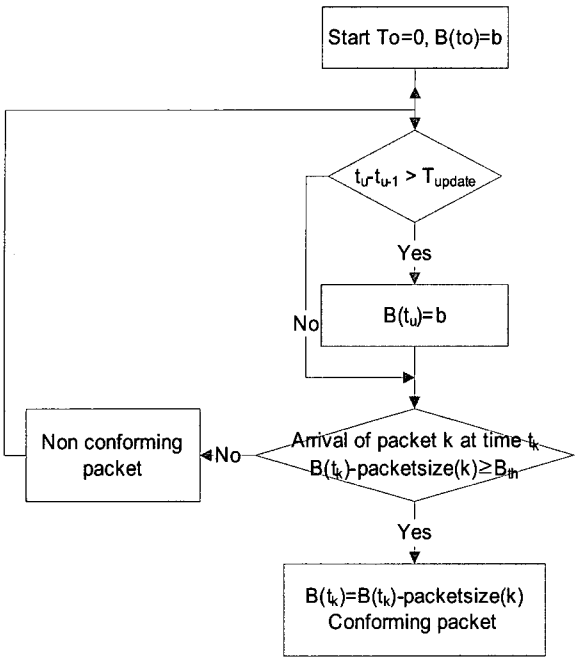
3.1 네트워크 프로세서에서의 구현 이슈

클래스 별 레이트 제한 시스템을 구현하려면, 첫째 트래픽을 정책에 따라 구별하는 능력을 가져야 하고, 둘째, 구별된 클래스 별로 레이트 제한을 가할 수 있어야 한다. 본 논문에서는 레이트 제한을 위해 토큰 버킷 알고리즘을, 클래스 별 레이트 분배를 위해 [1]의 Rate-based n-RED를 적용하고자 한다. 제안하는 시스템을 구현하기 위해서는 레이트 제한 기능 구현이 필수이고, 제한을 가하려는 한계값을 구하기 위해서는 트래픽 레이트를 추정 해야만 한다.

네트워크 프로세서인 IXP1200에서 패킷 처리하는 부분은 마이크로 엔진이다. 마이크로 엔진은 패킷의 내용을 변경하고 전송하는 기능을 위해 만들어진 구조로서, 메모리 참조명령, 시프트 명령, 분기 명령, 덧셈 명령만을 지원한다[10]. 식 (2)와 같은 연산을 위한 소수점 연산, 곱하기 연산 등 복잡한 수학 연산 기능을 가지고 있지 않으므로, 관련 연구에서 보여준 트래픽 레이트 추정 과정은 네트워크 프로세서에는 사용할 수 없다. 따라서, 클래스 별 레이트 제한을 하기 위해 Rate-based n-RED와 토큰 버킷 알고리즘의 특징을 추출하여 IXP1200에 적절하도록 변경된 알고리즘을 사용해야 한다.

3.2 레이트 제한

토큰 버킷 알고리즘은 패킷이 들어올 때마다 일정한 레이트 동안 쌓인 토큰을 버킷에 더하게 된다. 이에, 패킷이 들어올 때마다 전송율과 시간을 곱하여 토큰의 양을 구하는 방식이 아닌 해당 레이트에 맞게 버스트 크기를 정하여 일정 시간 내에 버스트 크기 이상의 패킷이 지나가지 않도록 하여 레이트를 제한하는 방식을 사용하기로 한다. 이를 그림으로 표현하면 다음과 같다.



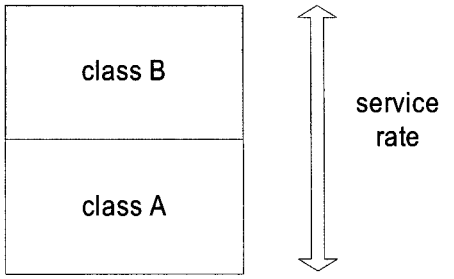
(그림 5) 레이트 제한 알고리즘

토큰은 일정한 시간(t)마다 버스트 크기(b)만큼 쌓이게 되며, 패킷이 전달될 때마다 패킷 양만큼 감소된다. 즉, 일정시간마다 쌓이게 되는 버킷의 양 b는 제한하고자 하는 패킷 양이 된다. 따라서, 전달되는 패킷 길이가 남은 토큰보다 클 경우에는 패킷이 전달되지 않고 폐기된다. 일정한 시간(t) 동안에 b만큼의 패킷만 보내주므로, b/t의 전송율을 유지하게 되는 것이다.

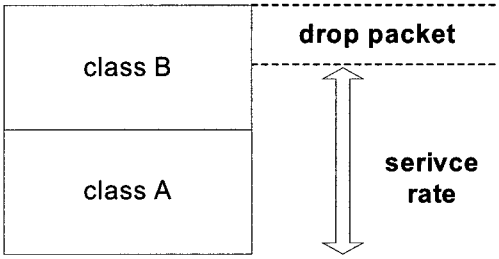
3.3 클래스 별 레이트 분배

Rate-based n-RED 알고리즘을 이용하여 클래스의 레이트를 분배한다. IXP1200은 RED 알고리즘을 그대로 구현하는 데에 한계가 있으므로, 본 시스템에서는 RED 대신에 tail drop 방식을 채택하여 레이트 제한을 구현한다. 또한, 클래스별 한계 대역폭을 정해 그 안에서 우선 순위에 따른 배분을 하도록 하여 클래스 별 대역폭 분배에 유동성이 있도록 한다. 예를 들어 2개의 클래스를 대상으로 클래스 기반의 레이트 분배를 수행 한다고 하면 Total Rate $T(t) = R(a,t) + R(b,t)$ 로 표현된다. 여기서 R(a, t), R(b, t)는 시각 t에서의 각 클래스의 트래픽 레이트이다. 이 때, 각 클래스의 트래픽 레이트에 따라 다음과 같은 시나리오로 클래스 기반의 레이트 분배가 동작하게 된다.

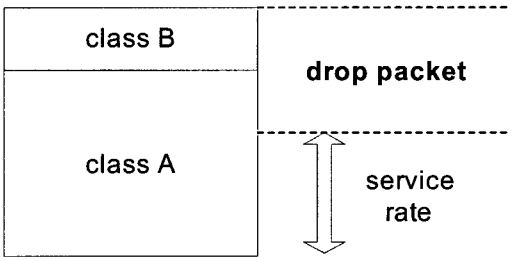
1. Total Rate <= Service Rate : (그림 6)과 같이, T(t)가 Service Rate보다 작으므로, 모든 클래스는 트래픽 레이트에 대해서 모두 서비스 받을 수 있다.
2. Total Rate > Service Rate, R(a) <= Service Rate : (그림 7)과 같이, T(t)가 Service Rate보다 크다. 따라서 이 때에는, Service Rate를 초과하는 트래픽 중, 우선순위가 낮은 클래스 B의 트래픽에 대해서 tail 드롭을 수행하게 된다.
3. R(a) > Service Rate : (그림 8)과 같이, Service Rate가 한 개의 클래스도 완전히 서비스 해줄 수 없는 상황이다. 따라서 우선순위가 가장 높은 클래스인 A 클래스를 서비스 해 줄 수 있도록 한다.



(그림 6) 모든 클래스의 수신



(그림 7) 클래스 B의 드롭 발생



(그림 8) 클래스 B의 서비스 불가, 클래스 A의 드롭

3.4 도착 레이트의 추정

Rate-based n-RED에서 사용한 트래픽 레이트 추정 방법은 IXP1200의 마이크로 엔진에서 연산할 수 없는 곱셈과 지수 연산으로 이루어져 있다. 식 (2)에서 확률과정을 나타내는 지수항을 제외하고는, 모두 정해져 있는 값이며, 네트워크 프로세서에서도 쉽게 구현할 수 있는 연산이다. 하지만 확률과정을 포함하는 지수항은 구현하는 것이 불가능하다. 따라서, 계산과정을 간단히 하면서 수식 (2)의 구조를 유지하기 위하여 지수항을 상수로 두었다. 결과적으로, 다음과 같은 수식을 통해 도착 레이트를 추정하였다.

$$R(t) = \alpha \times R(t-1) + (1-\alpha) \times (packet/time) \qquad (3)$$

위 식에서, 추정된 도착 레이트 R(t)은 α에 따라서 영향을 받게 된다. α가 큰 값을 가질수록 순간적인 레이트의 변화에 둔감하게 반응하고, α가 작은 값을 가질 수록 순간적인 레이

트의 변화에 영향을 많이 받는다.

4. 실험 및 평가

4.1 실험 환경

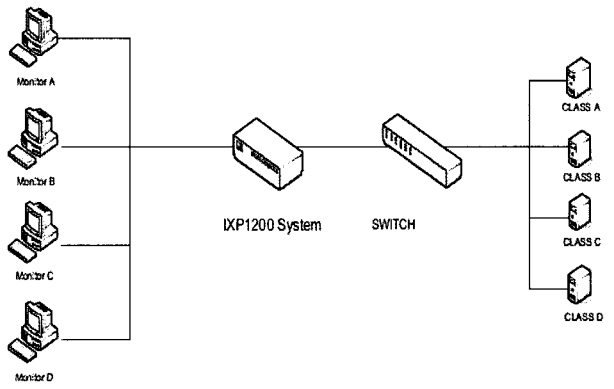
실험을 위하여 상기의 방안에 대한 프로그램을 작성하였다. 그 알고리즘을 보이면 (그림 9)와 같다. 알고리즘에서와 같이 테스트를 위하여 4개의 클래스를 적용하였다. <표 1>과 같이 각 클래스는 최대 레이트를 가지고 제한되며, 우선순위는 A > B > C >D의 순으로 하였다.

1. 들어온 패킷 체크
2. 현재까지 들어온 패킷을 바탕으로 각 클래스별 평균 패킷양 계산: pA, pB, pC, pD
3. pA=min(maxA, pA), pB=min(maxB, pB), pC=min(maxC, pC), pD=min(maxD, pD)
4. pA=pA-minA, pB=pB-minB, pC=pC-minC, pD=pD-minD
5. tmp=maxT-minA-minB-minC-minD
6. pT=pA+pB+pC+pD
7. pT<=tmp : bX=pX(X=A,B,C,D)
8. pT>tmp, pA+pB+pC<=tmp : bX=pX(X=A,B,C), bD=tmp-pA-pB-pC
9. pA+pB+pC>tmp, pA+pB<=tmp : bX=pX(X=A,B), bC=tmp-pA-pB, bD=0
10. pA+pB>tmp, pA<=tmp : bA=pA, bB=tmp-pA, bC=bD=0
11. pA>tmp : bA=tmp, bB=bC=bD=0

(그림 9) 실험을 위한 클래스 별 레이트 제한 알고리즘

〈표 1〉 레이트 제한 환경설정

클래스	Max. Rate(Mbps)	Service Rate
A	50	100
B	40	
C	30	
D	20	

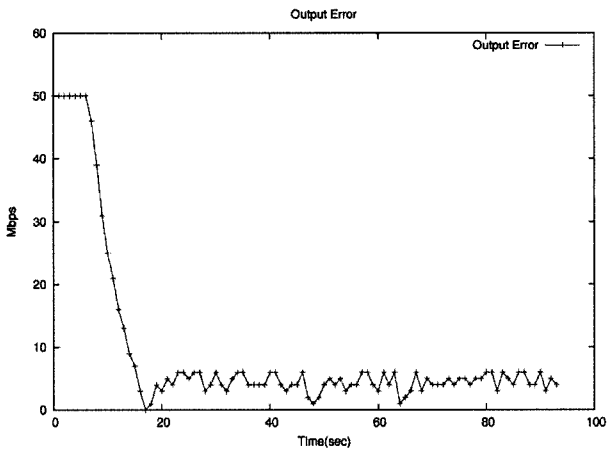


(그림 10) 테스트 환경

실험환경은 (그림 10)과 같이, 각 클래스를 담당하는 서버를 두어, 실험에 적합하도록 트래픽을 발생시켜 주도록 했다. 각 서버에서 만들어진 트래픽은 스위치에서 하나로 묶여 IXP1200 시스템으로 유입된다. 이 때, IXP1200은 <표 1>의 정책에 맞추어 레이트 제한을 수행한다. 처리된 결과를 쉽게 확인할수 있도록, 각 클래스를 서로 다른 100메가비트 이더넷 포트로 출력하도록 했다. 레이트 제한의 성능을 측정하기 위해 IXP1200의 100메가 출력포트에는 각 클래스 별 레이트를 측정하는 모니터 서버를 설치했다. 트래픽을 만들어 내는 소프트웨어는 MGEN을 사용했고, 모니터 서버에서는 Sniffer를 사용하여 트래픽 레이트를 측정하였다. Sniffer는 패킷 샘플링 타임이 1초이므로, 실험결과를 확인하기 위해 버킷 업데이트 시간을 1초로 하였다.

4.2 결과 분석 및 성능 평가

구현한 시스템의 성능 평가를 위한 기준으로는 정확성, 레이트 분배특성, 예측기 성능 등을 고려할 수 있다. 첫째로, 레이트 제한의 정확성은 구현된 알고리즘이 입력 트래픽을 얼마나 정확이 주어진 레이트로 제한하는가를 말한다. 구현된 시스템에서는 입력 트래픽이 시스템의 레이트 제한 알고리즘을 통과하게 되므로, 제한이 가해진 클래스의 출력 레이트는 구현된 알고리즘의 특성을 따르게 된다. (그림 11)은 출력을 50메가비트로 제한하고, 입력을 100메가 비트로 입력하여, 시스템이 입력 트래픽을 제한시키고 있는 실험에서 목표치와의 오차를 보여주는 그래프이다. 출력 레이트가 안정화 된 후의 오차는 12%이내에 머물면서 목표치를 따라가고 있다. 이는 [12]에서 보여주는 토큰 버킷 알고리즘의 성능인 10%와 근사한 성능을 보여주고 있다.



(그림 11) 출력 레이트 오차

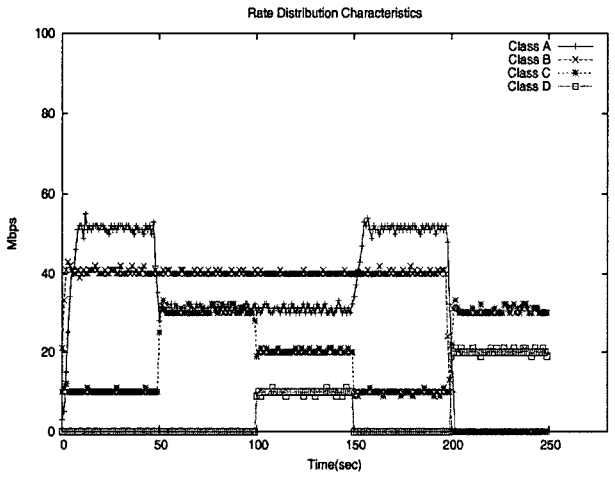
둘째로, 레이트 분배 특성은 정책에서 결정된 우선순위에 따라 각 클래스에 레이트를 분배하는가를 말한다. 본 논문에서 구현된 레이트 제한은 정해진 전체 서비스 레이트에서 각 클래스에 정해진 양의 레이트를 분배를 한다. 이 때, 클래스의 요청이 전체 서비스 레이트를 초과하면 클래스에 분배되는 레이트의 양은 우선순위에 따라서 결정된다. <표 2>는 레

이트 분배 특성을 보이기 위해 각 서버에서 만들어낸 트래픽의 양이다. 각 행의 트래픽은 약 50초 동안 유지가 된다. 각 클래스의 우선순위는 실험 설계시 정해졌던 순위대로 유지된다. <표 2>의 입력을 시스템에 유입 시켰을 때의 출력은 (그림 12)와 같이 나타난다.

〈표 2〉 레이트 분배실험의 입력 레이트

	A	B	C	D
1	50	40	30	20
2	30	40	30	20
3	30	40	20	20
4	80	40	20	20
5	0	0	60	30

1번 입력/출력에서, 우선순위에 따라 클래스 C는 일부가 폐기되고, 클래스 D는 차단된다. 2~4번 실험에서도 우선순위에 따라 레이트가 분배되고 있음을 볼 수 있다. 5번의 실험에서는 전체 서비스 레이트에 이르지 않더라도, 클래스에 분배된 레이트에 따라 제한되고 있음을 볼 수 있다.

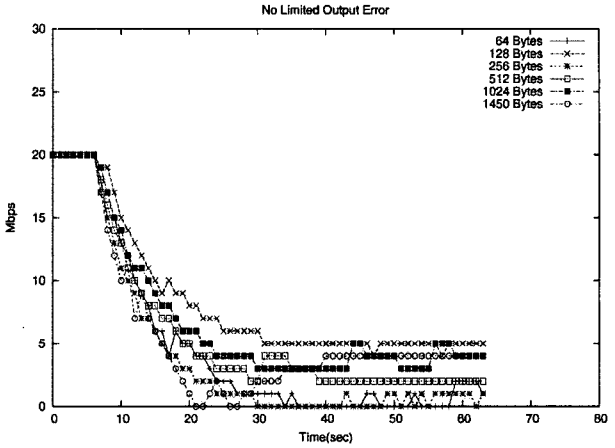


(그림 12) 레이트 분배실험의 출력 레이트

셋째로, 예측기의 성능은 시시각각 변화하는 트래픽 레이트를 얼마나 정확이 추정해 내는가를 보기 위함이다. 예측기의 성능은 주어진 입력 트래픽을 주어진 레이트로 얼마나 빨리 안정화 시키느냐로 나타낼 수 있으며 이 시간은 안정화 시간(settling time)으로 정의할 수 있다.

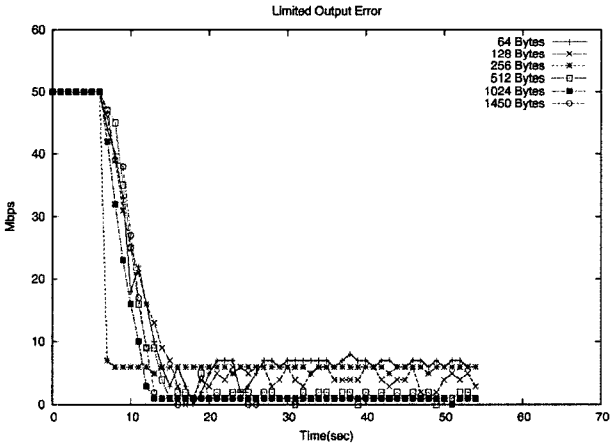
트래픽의 출력 양의 결정은 예측기를 통해서 결정된다. 따라서 예측기가 얼마나 자주 실행되느냐에 따라 출력량의 안정도에 영향을 미친다. 네트워크 프로세서에서는 SOP 및 EOP이 발생할 때에만 프로그램이 실행된다. 그러므로 도착하는 패킷의 크기에 따라서 예측기의 실행빈도가 결정되고, 그에 따라 출력 레이트의 안정화에도 영향을 미치게 된다.

(그림 13)은 트래픽에 제한을 가하지 않아도 되는 상황에서의 패킷 크기에 따른 출력 레이트의 오차를 보여준다. 결과에서 보듯이 오차율에서는 차이를 보이지만, 패킷 크기가 작은 트래픽의 안정화 시간은 짧고 패킷 크기가 큰 트래픽은 안정화 시간이 길다는 것을 볼 수 있다.



(그림 13) 패킷 크기별 비제한 출력 레이트의 오차 추이

(그림 14)는 트래픽에 제한을 가해야 하는 상황에서 패킷 크기에 따른 출력 레이트의 오차 추이를 보여준다. 전체적으로 제한을 가하지 않는 경우보다는 안정화 시간이 짧고, 오차율이 크지 않은 것을 확인할 수 있다. 또한 패킷 크기에 따른 안정화 시간의 차이가 두드러지게 나타나고 있지 않다. 이는 입력 트래픽의 양이 제한량을 넘었을 때, 입력 레이트의 변화를 고려하지 않고, 제한값으로 제한시키는 알고리즘의 영향인 것으로 볼 수 있다.



(그림 14) 패킷 크기별 제한 출력 레이트의 오차 추이

5. 결론 및 향후 과제

본 논문은 네트워크 프로세서에서 효율적으로 레이트 제한 알고리즘을 구현하기 위한 방안을 제안하고 이를 구현하여 실험함으로써 제안한 방안이 실제의 시스템에서 잘 동작함을 입증하였다. 정수 연산만이 가능한 네트워크 프로세서에서 프로세서의 실행시간을 낭비하지 않고 트래픽 레이트를 추정하

기 위하여 계산을 간단화 하였다. 그 결과, 실험결과에서 확인할 수 있듯이 레이트를 잘 지켜주고 있음을 볼 수 있고 예측기의 성능도 실제 트래픽 레이트를 잘 따라 가고 있음을 볼 수 있다.

현재 레이트 제한을 위한 네트워크 프로세서의 가장 큰 한계는 예측기 설계의 한계에 있으며 그 원인은 네트워크 프로세서의 연산 능력에 있다. 본 논문에서는 확률 연산 과정을 무시하고 상수를 적용함으로써 연산부하를 줄이고 예측기의 성능을 축소하는 방향으로 시스템이 구현되었다. 따라서, 향후에는 이를 보안하기 위하여 부가적으로 제공되는 하드웨어를 활용하는 등 예측기의 성능을 높이기 위한 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] Stefaan De Cnodder, Kenny Pauwels, Omar Elloumi : A Rate Based RED Mechanism, 2000.

[2] CISCO systems : Committed Access Rate white paper, 1999.

[3] P.F. Chimento : Standard Token Bucket Terminology, 2000.

[4] CISCO systems : quality of service solutions configuration guide - Policing and Shaping Overview.

[5] M.J.C. Büchli, D. De Vleeschauwer, J. Janssen, G.H. Petit : Policing Aggregates of Voice traffic with the Token Bucket Algorithm.

[6] Niraj Shah, Kurt Keutzer. "Netowkr Processors : Origin of Species, " Proceedings of ISCIS XVII, The Seven- teenth International Symposium on Computer and Information Sciences, October, 2002.

[7] P. Crowley, M. E. Fiuczynski, J. L. Baer, and B. N. Bershad. "Characterizing processor architectures for programmable network interface," In Proceeding of the International Conference on Supercomputing, 2000.

[8] Takeshi Miei. Mitsuru Maruyama, Tsuyushi Ogura, Naohisa Takahashi, "Parallelization of IP-Packet Filter Rules," IEEE, 1997.

[9] Nie, X., Gazsi, L., Engel F., Fettweis, G. "A New Network Processor Architecture for High-Speed Communications." In Proc. of IEEE Workshop on Signal Processing Systems, Taipei/Taiwan, Oct., 1999.

[10] Intel IXP1200 Network Processor Family, Hardware Reference Manual, Intel Corp. December, 2001.

[11] Intel IXP1200 Network Processor Family, Development Tools user's Guide, Intel, December, 2001.

[12] Eun-Chan Park, Chong-Ho Choi, "Adaptive token bucket algorithm for fair bandwidth allocation in DiffServ networks", Global Telecommunications Conference, Dec., 2003.

[13] 임강빈, 박준구, 정기현, 최경희, "네트워크 프로세서를 위한 다중 쓰레드 스케줄링", 정보처리학회논문지C, 제11-C권 3호, pp.337-344, 2004. 06.

[14] 임강빈, 이창희, 김종수, 최경희, 정기현, "클래스 기반의 대역 제한 기법을 통한 이메일 서버의 보호", 대한전자공학회 논문지 제41권 TC편 6호, pp.17-24, 2004. 06.

노 진 택

e-mail : jintech@korea.com
2003년 아주대학교 전자공학과(학사)
2005년 아주대학교 전자공학과(석사)
2005년~현재 삼성전자 연구원
관심분야 : 실시간 운영체제, 임베디드 시스템, 네트워크 시스템 등

이 진 선

e-mail : jstaira@ajou.ac.kr
2004년 아주대학교 전자공학과(학사)
2004년~현재 아주대학교 정보통신전문대학원 석사과정
관심분야 : 네트워크 시스템, 임베디드 시스템 등

최 경 희

e-mail : khchoi@ajou.ac.kr
1976년 서울대학교 사범대학 수학교육과(학사)
1979년 프랑스 그랑데꼴 ENSEIHT, 정보공학 및 응용수학(석사)
1982년 프랑스 Univ. of Paul Sabatier(박사)
1991년 프랑스 렌느 IRISA 연구소 교환 교수
1982년~현재 아주대학교 정보 및 컴퓨터 공학부 교수
관심분야 : 운영체제, 분산처리, 실시간 시스템 등

정 기 현

e-mail : khchung@ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 Univ. of Illinois, EECS(석사)
1990년 Univ. of Purdue, 전기전자공학부 (박사)
1991년~1992년 현대반도체 연구소
1993년~현재 아주대학교 전자공학부 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

임 강 빈

e-mail : yim@sch.ac.kr
1992년 아주대학교 전자공학과(학사)
1994년 아주대학교 전자공학과(석사)
2001년 아주대학교 전자공학과(박사)
2000년 (미)아리조나 주립대 객원연구원
2003년~현재 순천향대학교 정보보호학과 교수
관심분야 : 네트워크 및 시스템 보안, 실시간 운영체제, 임베디드 시스템, 멀티미디어 시스템 등