논문 2006-43TC-3-17

DiffServ 망에서 QoS를 보장하기 위한 개선된 퍼지 기반 WRR 알고리즘 개발

(Modiied Fuzzy-based WRR Algorithm for QoS Guarantee in DiffServ Networks)

정 경 택*, 박 준**, 김 변 곤*, 전 병 실**

(Kyung Taek Chung, Joon Park, Byun Gon Kim, and Byoung Sil Chon)

요 약

DiffServ망에서 많이 거론되고 있는 대표적인 스케줄러로 PQ(Priority Queue), WRR(Weighted Round Robin)등의 스케줄러가 연구되어져 있다. 그러나 이러한 스케줄러들은 장점을 가지고 있는 동시에 단점을 가지고 있다. 본 논문에서는 PQ와 WRR의 단점을 보완하면서 어떠한 스케줄링 방법에도 적용이 가능한 스케줄러 알고리즘을 제안한다. 본 논문에서 제안된 알고리즘은 DiffServ 망에서 퍼지 이론을 스케줄러에 적용하여 동적으로 가중치를 할당하는 알고리즘을 개발하였다. 따라서 퍼지이론을 통하여 퍼지 제어 규칙을 생성하여 유동적으로 각 클래스의 큐 상태를 체크하여 클래스가 가지고 있는 큐의 가중치를 효율적으로 할당하도록 하였다.

Abstract

PQ(Priority Queue) and WRR(Weighted Round Robin) are the most famous schedulers, however, these schedulers have both points of advantages and disadvantages. In this paper, we propose an algorithm that can be adopted in any kind of scheduling type with making up for weak points of PQ and WRR. The proposed algorithm includes a fuzzy theory in the scheduler to assign priorities dynamically in the DiffServ network. This algorithm produces the control discipline by the fuzzy theory to assign priorities of the Queue effectively with checking the Queue status of each class dynamically under the discipline.

Keywords : DiffServ, QoS, Scheduling, WRR

I. 서 론

최근 인터넷 이용의 증가는 전 세계에서 폭발적으로 증가하고 있다. 그로 인해 많은 사람들이 손쉽게 인터넷에 접속하고 있으며, 이러한 폭발적인 인터넷의 증가와 사용자의 증가는 VoIP(Voice of IP), VPN(Virtual Private Network), VOD(Video On Demand) 등과 같은 다양한 형태의 응용서비스를 등장시키고 있다. 이러한 응용 서비스를 제공해주기 위해서는 IP QoS(Quality of Service)가 제공되어야 한다[1].

이에 따라 IETF(Internet Engineering Task Force)에서는 늘어나는 QoS에 대한 요구를 수용하기 위해서 많은 연구를 진행하고 있다. 이중에서 하나가 모든 패킷 흐름(packet flow)에 대해서 QoS를 제공해주기 위해 제안된 IntServ이다. IntServ는 신호 프로토콜로 RSVP(Resource Reservation Protocol)를 사용하여 응용프로그램을 사용하기 전에 종단간에 미리 자원을 예약한다. 이러한 IntServ는 패킷 흐름 단위로 QoS를 보장하기 때문에 안정적인 QoS를 보장할 수 있다는 장점

* 정회원, 군산대학교 전자정보공학부
(Division of Electronic and Information Eng., Kunsan National University)
** 정회원, 전북대학교 전자정보공학부
(Division of Electronic and Information Eng., Kunsan National University)
※ 이 논문은 2004년도 군산대학교 학술연구비의 지원에 의하여 연구되었음.
접수일자: 2005년11월2일, 수정완료일: 2006년3월14일

이 있지만, 라우터가 모든 사용자의 응용프로그램에 대한 패킷 흐름 정보를 가지고 있어야 하기 때문에 확장성이 나쁜 단점이 있다[2].

그러나, DiffServ는 IntServ와는 다르게 모든 패킷에 대한 정보를 라우터가 가지고 있지 않다는 것이다. 대신에 몇 개의 클래스(Class)를 두어서 패킷을 클래스별로 분류해서 그 클래스의 PHBs(Per-Hop Behaviors)에 따라 각 클래스마다 차별화된 서비스를 제공한다. DiffServ는 CN(Core Network) 부분의 QoS 보장을 위한 좋은 방법 중의 하나이다. DiffServ는 복잡한 기능의 처리는 경계(edge) 라우터에서 수행하게 하고 코어(core) 라우터는 패킷을 전달만 함으로써 확장성을 향상시켰다[3].

현재 DiffServ에서 많이 거론되고 있는 대표적인 스케줄러로 PQ(Priority Queue), WRR(Weighted Round Robin)등의 스케줄러가 연구되어져 있다. 그러나 이러한 스케줄러들은 장점을 가지고 있는 동시에 단점을 가지고 있다.

PQ는 우선순위가 높은 클래스를 먼저 서비스 해주는 방식이므로 EF를 구현하기에는 좋으나 EF 클래스의 트래픽 입력률이 높아지면 우선순위가 낮은 클래스들은 서비스를 받을 수가 없기 때문에 QoS를 보장해 주지 못한다는 단점이 있다.

WRR은 각각의 클래스가 주어진 가중치로 링크 용량을 나누어서 사용하기 때문에 각 클래스가 공평하게 서비스를 받을 수 있다는 장점이 있다. 하지만, EF 클래스의 트래픽이 집중되는 경우에 EF의 QoS를 보장해 줄 수가 없다는 것이 단점이다.

따라서, 본 논문에서는 PQ와 WRR의 단점을 보완하면서 어떠한 스케줄링 방법에도 적용이 가능한 스케줄러 알고리즘을 제안한다. 본 논문에서 제안된 알고리즘은 DiffServ 망에서 퍼지 이론을 스케줄러에 적용하여 동적으로 가중치를 할당하는 알고리즘을 개발하였다. 따라서 퍼지이론을 통하여 퍼지 제어 규칙을 생성하여 유동적으로 각 클래스의 큐 상태를 체크하여 클래스가 가지고 있는 큐의 가중치를 효율적으로 할당하도록 하였다.

본 논문의 구성은 제 Ⅱ장에서는 DiffServ 개요에 관해서 기술하고, 제 Ⅲ장에서는 스케줄러 개요 및 현재 사용되어지고 있는 패킷 스케줄러 알고리즘과 퍼지이론에 관하여 기술한다. 제 Ⅳ장에서는 퍼지 이론을 적용시켜 제안된 알고리즘을 제시하고 퍼지 이론 및 퍼지 규칙 등에 관하여 기술하고, 시뮬레이션 환경 및 결과에 관하여 결과 그래프로 현재 사용되어지고 있는 다른 스케줄러와 비교하여 기술하고, 마지막으로 제 Ⅴ장에서 결론과 향후 연구되어야 할 내용을 기술한다.

Ⅱ. DiffServ 개요

사용자 흐름 단위에서 벗어나 흐름들의 집합화(aggregates) 단위로 서비스를 차별화한 DiffServ 구조가 IETF DiffServ WG에서 1997년부터 논의되기 시작되어 빠른 속도로 관련 표준안이 개발되어 왔다. 이 DiffServ 모델에서는 사용자 흐름에 대한 제어가 망의 경계에서 이루어지게 하였으며, 사용자 패킷 흐름들이 망 내로 유입될 때는 소수의 트래픽 클래스로 집합화함으로써 QoS를 지원하기 위한 망 내에서의 복잡한 패킷 처리 과정을 단순화하였다. 사용자 흐름을 집합화함으로써 IntServ에서와는 달리 망 내의 코어 라우터들이 개별적인 사용자 흐름을 인식하면서 흐름의 상태 유지를 위한 시그널링(signaling)에 대한 필요성을 덜 수 있게 된 것이다. DiffServ의 또 다른 장점으로는 DiffServ를 지원하는 다수의 망이 연접되어 서비스가 이루어질 때에도 망간의 협상만을 통해서 종단간 서비스를 제공할 수 있기 때문에 대규모 망에도 적용이 가능하다는 것이다[3,4]. 즉, DiffServ는 다양한 어플리케이션들의 서비스를 분류하는 간단하고 조잡한(coarse) 방법이다. 다른 것들이 가능하다 하더라도 두 개의 서비스 수준(traffic class)을 효율적으로 표현하는 두 개의 표준 PHB와 QoS를 제공하지 않는 PHB도 정의가 되어있다.

i. Expedited Forwarding (EF) : 단일한 codepoint (DiffServ 값)를 가진다. EF는 지연(delay)과 변화(jitter)를 최소화시키며 가장 높은 수준의 aggregate 서비스 품질을 제공하고 규정된 트래픽 프로파일(Traffic Profile)을 초과하는 어떤 트래픽도 무시된다[5].

ii. Assured Forwarding (AF) : AF는 4개의 클래스(class)를 가지며, 각 클래스에는 3개의 폐기 우선순위가 있다. 따라서 전부 12개의 codepoiont가 있다. AF 트래픽을 초과하는 것은 전송 되지 않을 가능성이 높다. 이것은 등급이 떨어진다는 뜻이며 폐기되어질 필요는 없다[6].

iii. Default (DE) : 현재 라우터에서 널리 사용되고 있는 패킷 전달 방식인 BE 전달 방식을 정의하고 있다. 이 방식은 QoS를 위해서 특별한 일을 해주지 않는다. 단지, 패킷은 FIFO(First-In First-Out)에 따라 입력된

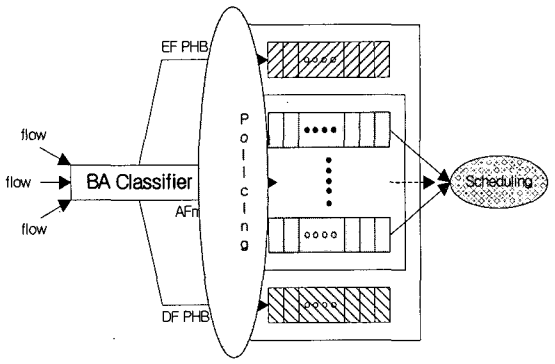

그림 2.1 PHB 동작
Fig. 2.1 PHB operation.

순서에 따라 출력이 된다. DiffServ를 지원하지 않는 사용자를 허용하기 위한 형태이다.

PHB들은 사전에 정의된 정책 결정요소에 따라 트래픽의 원활함을 위하여 네트워크 유입점(ingress point)에서 적용된다. 트래픽은 이 점에서 마크(Mark)되어지며 그 마킹(marking)에 따라 경로 설정이 된다[7]. 그런 다음 네트워크 인출점(egress)에서 언마크(unmark)되어진다. 호스트로부터 시작되어지는 것 또한 DiffServ 마킹이 적용되어 질 수 있으며, 그렇게 함으로써 많은 장점을 취할 수 있게 된다.

Ⅲ. 제안된 알고리즘

본 장에서는 DiffServ에서 많이 고려되고 있는 PQ(Priority Queue)와 WRR(Weighted Round Robin) 스케줄러의 장·단점에 대해서 살펴보고, 이들의 단점을 보완하기 위한 퍼지기반 동적 WRR 스케줄링 방식을 제안하고자 한다. 제안된 스케줄링 기법은 퍼지 이론을 이용하여 각 클래스(EF, AFs, DE) 큐의 상태정보를 구하고, 큐의 상태정보에 따라서 각 클래스 큐의 가중치(Weight)를 동적으로 할당해주고자 한다. 각 클래스의 가중치를 동적으로 할당함으로써 PQ의 단점인 우선순위가 낮은 클래스의 손실을 줄일 수 있고, WRR의 단점인 우선순위가 높은 클래스의 손실을 줄일 수 있으므로 보다 효율적인 서비스가 가능할 것이다.

1. Priority Queue

PQ방식은 우선순위가 높은 큐가 항상 먼저 서비스를 받게 되는 방식이다[5]. 따라서 이 방식은 우선순위가 높은 트래픽에게는 짧은 지연과 작은 지터를 보장해 주고, 링크 용량도 많이 할당을 받아 사용할 수 있도록 해

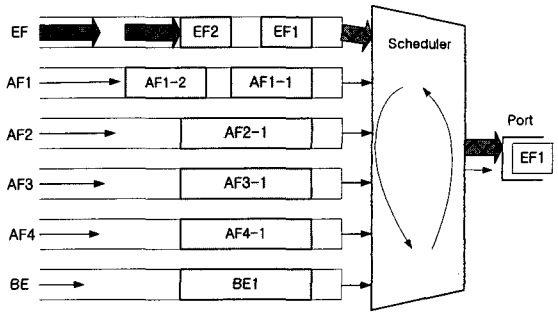

그림 3.1 PQ의 구조와 문제점
Fig. 3.1 Problem and structure of PQ.

준다. PQ는 EF PHB를 구현하기에 상당히 좋은 방식의 스케줄러라고 할 수 있다. 하지만 PQ의 단점은 그림 3.1에서 보듯이 상위 우선순위를 가지고 있는 트래픽의 양이 많은 경우 우선순위가 낮은 트래픽은 QoS를 보장받을 수 없다는 것이다.

3.2 Weighted Round Robin 알고리즘

WRR 기법은 ATM(Asynchronous Transfer Mode)과 같은 고속 패킷 스위칭 네트워크에서 계산의 단순성과 저 비용 구현의 장점을 가지고 있어 셀 스케줄링 기법으로 널리 쓰인다. 이 기법은 특정 큐의 서버에 대한 접근 시간의 양을 제어함에 의해서 대역을 할당한다. 또한, WRR은 각 큐에 대역을 보장해 주기 때문에 스케줄링 기법으로 많이 사용되었다. WRR은 각 큐에 가중치(weight)를 할당하여 가중치에 따라 최소 대역을 보장한다. 그러나, WRR 기법은 버스티한 트래픽에 대해서 저 효율성의 문제점이 있다. 또한 WRR 기법은 입력 트래픽을 일정한 율로 스케줄링하기 때문에 ABR 서비스 클래스의 모든 공평성 기준을 만족시킬 수는 없다[8].

그림 3.2은 WRR 기법을 나타내는데, 각각의 큐 1, 2, 3에 할당된 가중치(weight) 값은 2, 1, 3이다. WRR은 어느 특정 클래스가 링크 용량을 독점하는 것을 방지한다. 따라서 우선순위가 높은 클래스의 트래픽 양에 상관없이 각각의 클래스는 자신의 링크 용량을 사용할 수

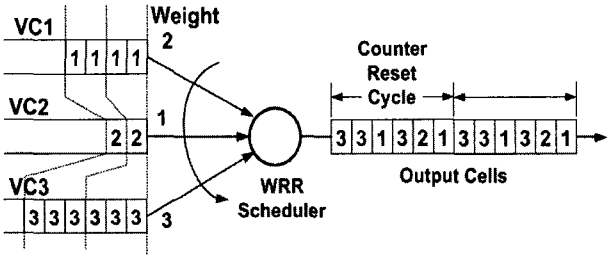

그림 3.2 WRR 기법의 구조
Fig. 3.2 Structure of WRR scheme.

있어서 각 클래스 별로 QoS를 보장해 줄 수 있다. 하지만, WRR의 단점은 DiffServ의 경우에 망 안에서 EF 트래픽이 한쪽으로만 집중되는 경우 EF의 QoS를 보장해줄 수 없다는 것이다.

3.3 퍼지 제어 규칙 생성

그림 3.3은 퍼지 제어 구조를 나타내고 있다.

퍼지 기반 가중치 제어기는 각 클래스 큐의 셀 수(λ)와 변화량($\Delta\lambda$)이 입력되면 퍼지 제어기에서 큐의 상태정보 값을 계산하고 클래스의 우선순위와 상태정보에 따라서 가중치의 증가와 감소를 WRR 스케줄러에서 처리하도록 제어한다.

퍼지 집합이론에 의하여, 퍼지 입력 변수는 그림 3.4와 같이 셀 수의 변화량과 그림 3.5와 같이 현재 셀 수를 가지고 $|T(\lambda)|\times|T(\Delta\lambda)|$과 같은 2차원의 배열이 형성된다. $|T(X)|$는 $T(X)$의 언어변수 항들의 수이다. 이와 같이 두 가지 변화량(λ, $\Delta\lambda$)을 가지고 퍼지 용어 집합을 만들 수 있는데 그 용어 집합은 다음과 같다.

$\Delta\lambda$(셀의 변화량) = {D, M, I}
λ(셀 수) = {Z0, NB, NM, PM, PB}
$\lambda+\Delta\lambda$(가중치 변화} = {NB, NM, Z0, PM, PB}

위에서 규정한 용어 집합을 기반으로 퍼지 제어규칙을 만들 수 있는데 총 15가지의 결과를 만들 수 있는데 표3.1에서 보여주는 바와 같다.

퍼지 집합으로 추론되어진 결과로부터 제어를 수행하기 위해서는 비퍼지값을 도입하지 않으면 안되기 때문에 비퍼지화 (defuzzification) 단계가 필요하다. 비퍼지화 값을 구하는 단계는 그림 3.6에서 보여주고 있다. 두개의 제어 규칙에 의하여 추론이 된다. 두개의 입력

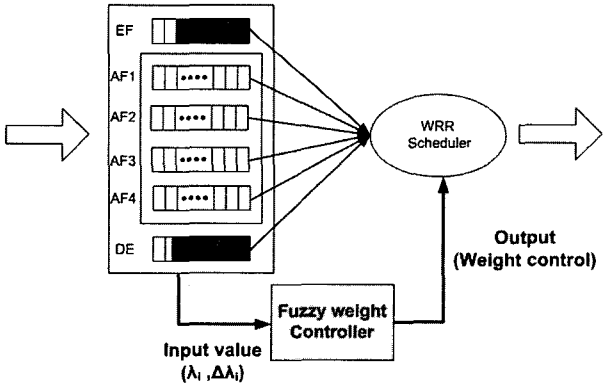

그림 3.3 퍼지 제어 구조
Fig. 3.3 Structure of fuzzy controller.

표 3.1 퍼지 제어 규칙
Table 3.1 Fuzzy control rule.

	셀 수의 변화량($\Delta\lambda$)	현재 셀 수(λ)	가중치(Weight)의 처리결과
1	D	NB	NB
2	D	NM	NM
3	D	Z0	Z0
4	D	PM	Z0
5	D	PB	PM
6	M	NB	NB
7	M	NM	NM
8	M	Z0	Z0
9	M	PM	PM
10	M	PB	PB
11	I	NB	NM
12	I	NM	Z0
13	I	Z0	PM
14	I	PM	PB
15	I	PB	PB

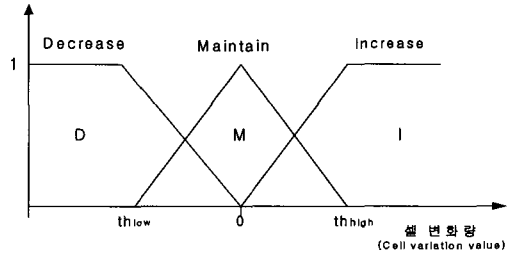

그림 3.4 셀 변화량($\Delta\lambda$)
Fig. 3.4 Differential rate of cell($\Delta\lambda$).

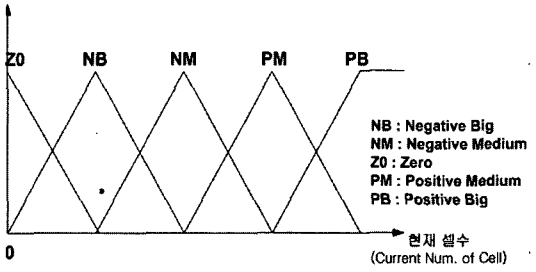

그림 3.5 현재 셀 수(λ)
Fig. 3.5 Current number of cell(λ).

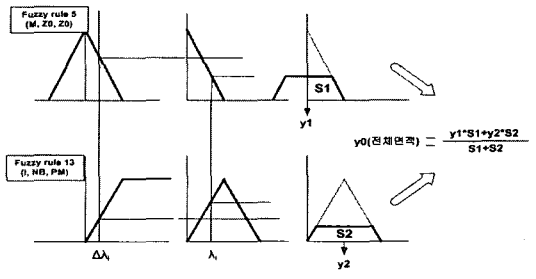

그림 3.6 비퍼지화 단계
Fig. 3.6 Step of defuzzification.

값이 입력되고 최소의 입력된 값으로 삼각형을 자르게 되면 사각형의 모양 두개가 나오는데 이 두개의 면적을 구하여 중심값을 구하게 되면 비퍼지화 되어진 값이 나오게 된다. 이렇게 구해진 각 클래스 큐의 비퍼지값을 상태정보라 하고, 각 클래스의 상태정보에 따라서 폭주가 발생하였는지 아닌지를 판단하여 가중치의 변화를 컨트롤 할 수 있다.

3.4 퍼지 기반 가중치 제어 알고리즘

본 논문에서 제안한 알고리즘은 PQ와 WRR의 단점을 보완하여 EF 클래스에만 서비스가 집중되는 것을 방지하고 나머지 우선순위가 낮은 클래스에도 서비스 지원을 향상시켰다. 우선 기본적인 알고리즘은 퍼지 이론을 이용하여 각 클래스 큐의 상태정보를 구하고, 상태정보에 따라 각 클래스 큐의 가중치를 동적으로 제어한다.

DiffServ 모델에서 사용자 패킷은 트래픽 특성에 따라 EF, AFs, DE 클래스로 집합화 된다. EF 클래스는 우선순위가 가장 높은 클래스로서 최소의 손실과 지연을 보장해야 한다. 다음으로 AFs 클래스는 3개의 폐기 우선순위에 따라서 서비스 되며, 약정 대역폭을 초과하는 트래픽은 전송되지 않을 가능성이 높다. 마지막으로 DE 클래스는 best effort 방식으로 전달되며 Diffserv를 지원하지 않는 사용자를 허용하기 위한 형태이다.

제안된 스케줄러 알고리즘은 우선순위가 가장 높은 EF 클래스를 기준으로 한다. EF 클래스의 큐 상태가 thHigh 보다 크다면 폭주가 발생하여 패킷 손실이 발생할 수 있으므로 AF 클래스 내에서 큐 상태를 체크하여 가장 작은 클래스를 찾는다. 가장 작은 클래스의 큐 상태가 thLow 보다 작다면 가장 작은 AF 클래스의 가중치를 감소시켜 EF 클래스의 가중치를 증가 시켜준다. 만약 그렇지 않다면 DE 클래스의 큐 상태를 체크하여 WM(Minimum Weight)보다 크다면 DE 클래스의 가중치를 감소시켜 EF 클래스의 가중치를 증가 시킨다. 이렇게 큐의 상태정보에 따라서 대역에 여유가 있는 AF 클래스나 우선순위가 낮은 DE 클래스의 가중치를 빌려옴으로써 EF 클래스의 폭주를 완화시킬 수 있고, 폭주가 완화되면 가중치를 반환하게 된다. EF 클래스 가중치의 반환과정은 AF 클래스에서 빌려온 가중치가 있다면 AF 클래스 중에서 가장 폭주가 심한 클래스에 반환하고 AF 클래스에서 빌려온 가중치가 없다면 DE 클래스에 반환하게 된다.

그림 3.7은 EF 클래스에서 AF 클래스와 DE 클래스에 연관된 제안된 알고리즘의 흐름도이다. 그림 3.8은 AF 클래스 내에서의 가중치 제어알고리즘을 보여주고 있다. AFi 클래스의 큐 상태를 체크하여 thHigh 보다 크다면 트래픽의 폭주가 발생 한 것으로 판단하여 AFn 클래스 중에서 큐 상태 값이 가장 작은 클래스를 찾는다. 가장 작은 클래스를 찾으면 클래스의 큐 상태를 체크하여 thLow 보다 작으면 가중치를 감소시켜 thHigh

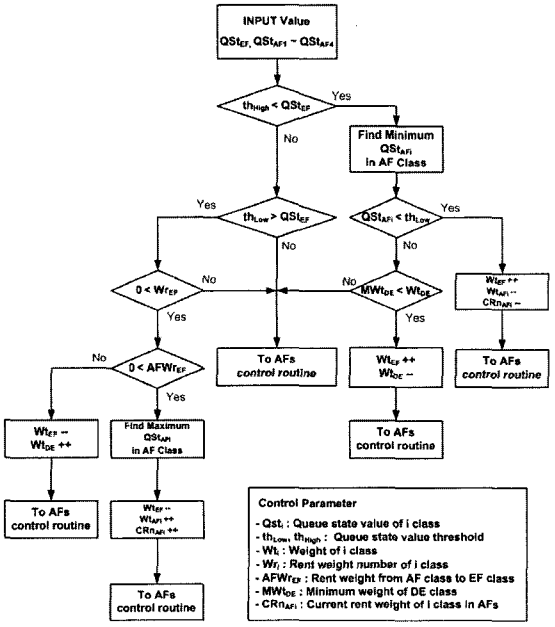

그림 3.7 제안된 알고리즘 EF 클래스의 흐름도
Fig. 3.7 Flowchart of proposed algorithm in EF class.

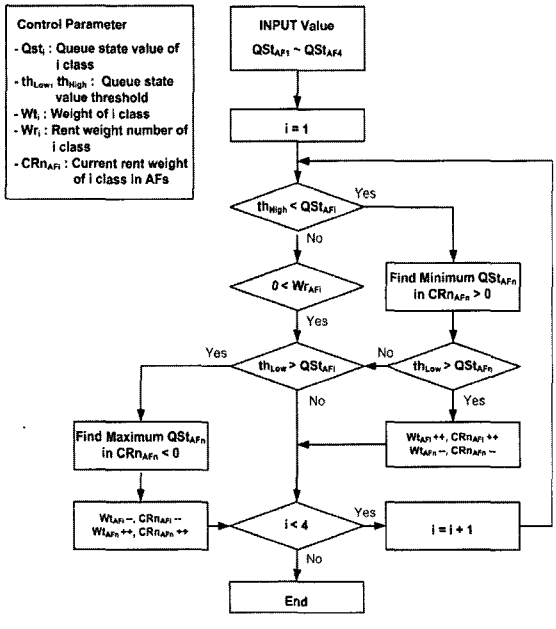

그림 3.8 제안된 알고리즘 AF 클래스의 흐름도
Fig. 3.8 Flowchart of proposed algorithm in AF class.

보다 큰 AFi 클래스의 가중치를 증가 시켜 폭주를 완화 시킨다. 만약 AFi 클래스의 큐 상태가 thLow 보다 작으면 AFn 클래스 중에서 가장 폭주가 심한 클래스를 체크하여 AFi 클래스의 가중치를 감소 시켜 AFn 클래스 중에서 가장 폭주가 발생한 클래스에 가중치를 증가 시켜 폭주를 완화 시킨다.

Ⅳ. 시뮬레이션 및 성능 평가

4.1 시뮬레이션 환경

본 논문에서 제안된 알고리즘의 시뮬레이션은 NS-2라는 Network Simulator Tool을 사용하여 DiffServ 노드를 기반으로 망을 구성하여 시뮬레이션 하였다.

제안된 알고리즘을 위한 네트워크 모델은 그림 4.1과 같다. 기본적인 네트워크 모델의 구성은 2개의 경계 라우터(edge router) R1, R3과 1개의 코어 라우터(core router) R2를 가지고 있다. 경계 라우터 에서는 소스 노드로부터 패킷을 받으면 패킷을 분류, 측정, 표시, 쉐이핑/폐기를 하는 역할을 한다. 이렇게 처리된 패킷들은 R2로 보내지게 된다. 그리고 각 경계 라우터에 6개의 클래스가 접속되어 있다. 이 클래스들은 DS 노드에 포함되어 있는 기본적인 PHB인 EF, AF, DE 클래스들로 구성되어 있다. 가장 우선순위가 높은 EF 클래스와 AF1~AF4까지 4개의 클래스로 구성되어 있는 AF 클래스와 마지막으로 우선순위가 가장 낮으면서 최저 서비스만을 기다리는 DE 클래스로 구성 되어있다.

R1과 R2 사이의 링크 1에는 25Mbps의 링크율(link rate)와 10msec의 링크 지연을 주어서 병목현상을 발생시켰고, R1의 큐의 크기는 50 패킷으로 하였으며, 병목현상으로 인한 성능을 분석하였다.

그림 4.2는 입력되는 트래픽의 양과 라우터 내부의 큐의 상태를 보여주고 있다. 시뮬레이션은 두 가지 모드로 테스트 하였는데 AF4 클래스의 입력율이 4.5Mbps인 경우와 2.0Mbps인 경우이다. 첫 번째는 EF 클래스를 제외한 나머지 클래스에 남는 가중치가 없는 경우와 두 번째로 EF 클래스를 제외한 나머지 클래스(AF4)에 남는 가중치가 있을 경우로 구분하여 시뮬레이션 하였다. 첫 번째는 폭주 상황이 발생하게 되었을 때를 의미하는데 폭주가 발생 하였을 때에는 효과적으로 자원을 분배하는지가 중요한 관점이고, 두 번째는 클래스에 남는 가중치가 있을 경우에는 남는 가중치를 어떻게 하면 공평하게 사용할지가 중요한 관점이다.
모든 라우터에서 큐를 관리하기 위한 방법으로는 EF 클래스에서는 Drop Tail 방법, AF 클래스에서는 RED 방법, DE 클래스에서는 Drop Tail 방법을 사용하였으며 시뮬레이션은 100초 동안 수행하였다.

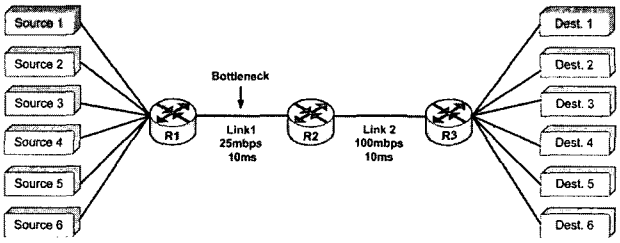

그림 4.1 NS-2로 구성한 시뮬레이션 네트워크 모델
Fig. 4.1 Network simulation model to using NS-2.

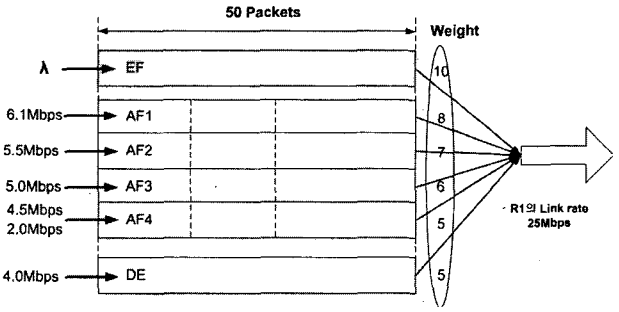

그림 4.2 라우터 내부 큐 상태
Fig. 4.2 Inner queue state of router.

4.2 시뮬레이션 결과 및 분석

시뮬레이션 결과는 앞서서 기술했던 두 가지 모드, EF 클래스를 제외한 클래스에 남는 가중치가 없을 때와 남는 가중치가 있을 때의 상황을 설정하여 시뮬레이션 결과들을 그림으로 나타낸다. 그리고 현재 사용되어지고 있는 방식인 PQ방식과 WRR 방식, 그리고 제안된 방식을 비교분석하여 나타낸다. WRR 방식은 EF 클래스의 손실을 줄이기 위해 EF 가중치를 미리 11, 12로 증가시켜서 WRR_11, WRR_12 방식을 병행하여 시뮬레이션을 수행하였으며, 제안된 방식에서 가중치 동적으로 제어한 방식을 PRO_D라 하였는데, 동적인 방식이란 제어주기를 고정시킨 방식이 아니라 가중치의 변화가 있을 경우 제어 주기를 짧게 조정하고 변화가 없을 경우 길게 조정하여 시뮬레이션을 수행한 것이다.

그림 4.3에서 4.8은 첫 번째 모드에서의 시뮬레이션 결과를 보여주고 있다.

그림 4.3은 EF 클래스의 평균 패킷 폐기율을 보여주고 있다. PQ의 경우에는 링크의 가중치를 우선순위가 가장 높은 EF 클래스에서 우선적으로 사용할 수 있기 때문에 PQ에서의 성능은 현저하게 좋으나 WRR의 단

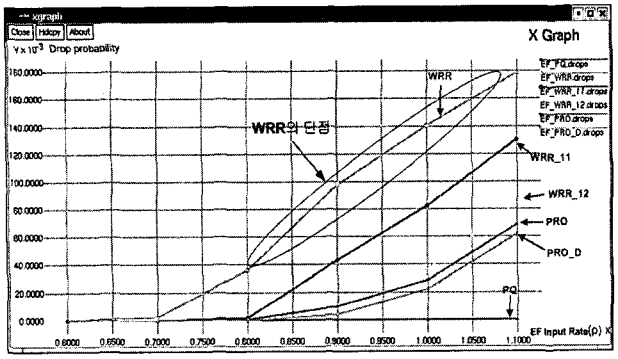

그림 4.3 EF 클래스의 패킷 폐기 확률 (CASE1)
Fig. 4.3 Packet drop rate of EF class (CASE1).

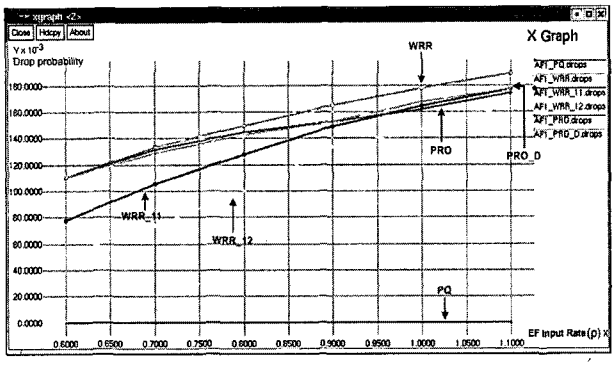

그림 4.4 AF1 클래스의 패킷 폐기 확률 (CASE1)
Fig. 4.4 Packet drop rate of AF1 class (CASE1).

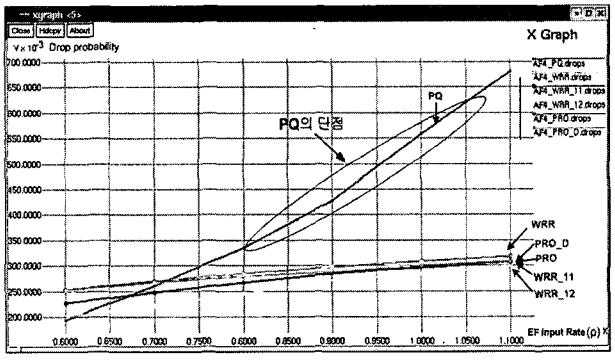

그림 4.5 AF4 클래스의 패킷 폐기 확률 (CASE1)
Fig. 4.5 Packet drop rate of AF4 class (CASE1).

점인 입력이 많아질수록 EF 클래스의 폐기율이 증가하는 것을 볼 수 있는 결과이다. 제안된 알고리즘은 EF 클래스의 트래픽이 폭주하여 손실이 발생할 경우 가중치를 동적으로 제어함으로써 EF 클래스의 손실을 줄일 수 있어 WRR 방식의 단점을 보안할 수 있다.

그림 4.5와 4.6은 AF4 클래스와 DE 클래스의 손실율을 보여주고 있는데, PQ 방식은 우선순위가 낮은 클래스를 손실을 피할 수 없는 단점이 있으나, 제안된 알고리즘은 PQ의 이러한 단점을 보완하여 패킷 손실율과 평균 지연시간이 현저하게 떨어진 것을 알 수 있다. 마지막으로 AF4 클래스를 제외한 클래스들은 EF 클래스

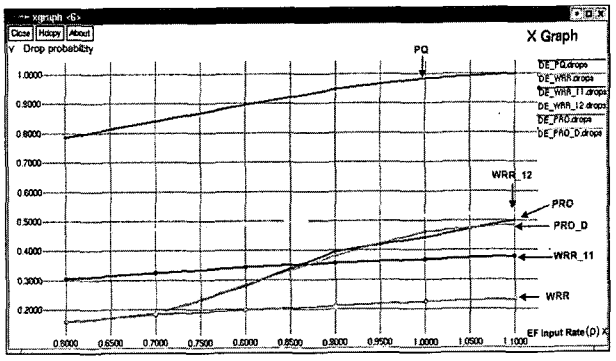

그림 4.6 DE 클래스의 패킷 폐기 확률 (CASE1)
Fig. 4.6 Packet drop rate of DE class (CASE1).

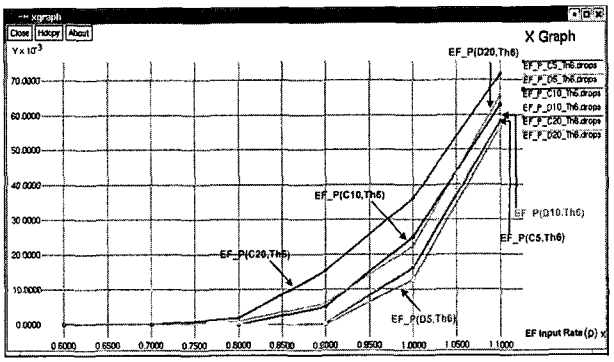

그림 4.7 제안된 파라미터에 따른 EF 클래스의 패킷 폐기 확률
Fig. 4.7 Packet drop rate of EF class on proposed parameter. [ThLow -0.8, ThHigh 0.6 : CT (15, 30, 60 msec)]

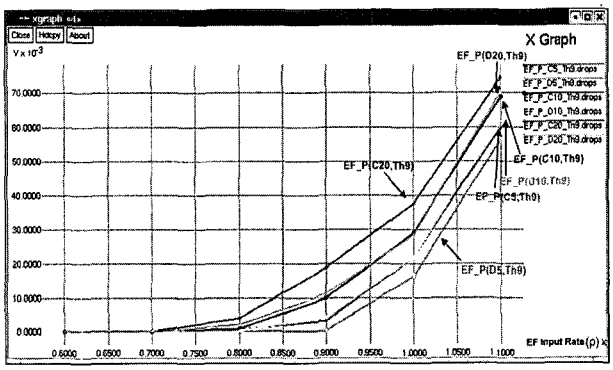

그림 4.8 제안된 파라미터에 따른 EF 클래스의 패킷 폐기 확률
Fig. 4.8 Packet drop rate of EF class on proposed parameter. [ThLow -0.8, ThHigh 0.9 : CT (15, 30, 60 msec)]

의 입력율이 작을 때는 WRR과 비슷한 성능을 보이고, 입력율이 증가하여 폭주가 발생할 경우에는 WRR_11, WRR_12와 비슷한 성능을 보이고 있다. 제안된 알고리즘은 WRR 기반으로 가중치를 동적으로 제어하여 트래픽 폭주가 발생할 경우 EF 클래스의 손실을 줄일 수 있고, AF 클래스의 QoS를 보장하해 주면서도 DE 클래

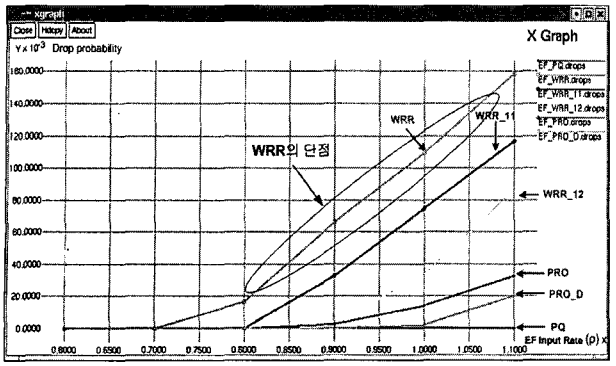

그림 4.9 EF 클래스의 패킷 폐기 확률 (CASE2)
Fig. 4.9 Packet drop rate of EF class (CASE2).

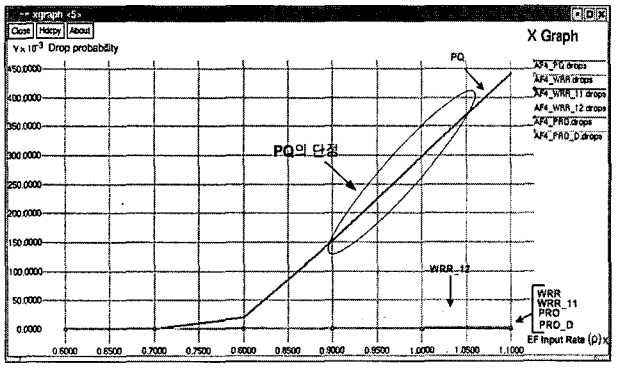

그림 4.10 AF4 클래스의 패킷 폐기 확률 (CASE2)
Fig. 4.10 Packet drop rate of AF4 class (CASE2).

스도 어느 정도 보장해주는 것을 볼 수 있다.

그림 4.7과 4.8은 퍼지 제어를 위한 CT(Control Time)과 EF 클래스의 퍼지 임계치의 변화에 따른 시뮬레이션 결과를 보여주고 있다. 이 결과들은 CT이 각각 15msec, 30msec, 60msec 일 때이다. 그리고 EF 클래스의 가중치를 반환하기 위한 최소 임계치(thLow)값은 -0.8이고 가중치를 증가시키기 위한 최고 임계치(thHigh)값을 0.6과 0.9의 경우에 시뮬레이션 결과 그래프이다. EF 클래스의 패킷 손실율은 임계치가 작고 제어 주기가 빠를 수 록 나은 결과를 보여주고 있다. 그러나 가중치의 변화가 자주 일어날 수 있으므로 시스템의 부하를 고려하여 CT 주기가 길면서 성능의 차이가 없는 것이 효과적이다. 따라서 CT 주기는 길게 하고, 성능저하를 막기 위해 가중치의 변화가 있을 경우 제어 주기를 짧게 조정하고 변화가 없을 경우 길게 조정하는 동적 제어 방식을 제안하였다.

그림 4.9에서 4.11은 두 번째 모드에서의 시뮬레이션 결과를 보여주고 있다. 두 번째 모드의 시뮬레이션 결과를 살펴보면 첫 번째 모드에서와 마찬가지로 WRR의 단점인 EF 클래스의 손실율을 줄일 수 있고, PQ의 단점인 AF4 클래스의 손실율을 줄일 수 있었다. DE 클

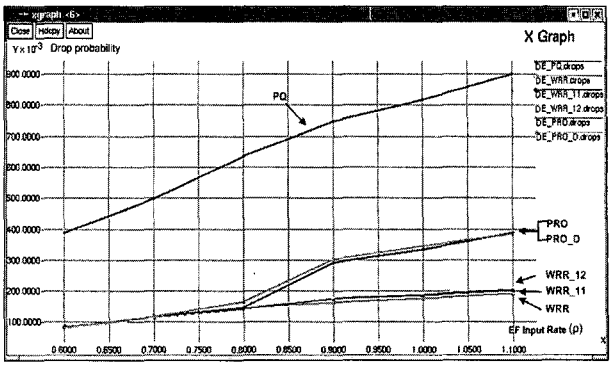

그림 4.11 DE 클래스의 패킷 폐기 확률 (CASE2)
Fig. 4.11 Packet drop rate of DE class (CASE2).

표 4.1 패킷 폐기율 비교(CASE2)
Table 4.1 Comparison of drop probabilities at EF input rate=1.1.

Class / Scheduler	EF	AF1	AF2	AF3	AF4	DE
PQ	0%	0%	0%	2%	45%	90%
WRR	16%	14%	14%	20%	0%	20%
WRR_11	12%	15%	15%	23%	0%	20%
WRR_12	9%	17%	17%	24%	5%	20%
PROposed	2%	14%	14%	20%	0%	38%

래스는 최소 보장 서비스만을 하고 있기 때문에 효과가 조금은 감소하였다. 그 이유는 EF 클래스와 AF 클래스의 효과가 증가한 만큼 조금의 감소를 보이게 된 것이다. 하지만 제안된 알고리즘은 EF와 AF 클래스의 QoS를 보장해주면서도 DE 클래스도 PQ와 비교하였을 때 어느 정도 보장해주는 것을 볼 수 있다.

표 4.1은 AF4 클래스에 남는 가중치가 발생하였을 때의 각 클래스별 패킷 폐기율을 나타내고 있다(EF input rate=1.1에서). 그리고 제안된 스케줄러의 패킷 폐기율을 보면 PQ방식에서 발생하는 AF4클래스의 손실을 방지하였으며, EF 클래스의 손실율을 WRR 방식에 비해 현저히 작게 나타나고 있다.

따라서, 우선순위가 높은 클래스의 패킷 폐기율이 낮으면서 우선순위가 낮은 클래스에서도 성능이 향상이 되었음을 알 수 있다. 따라서 남는 가중치를 효율적으로 분배하여 사용하고 있음을 알 수 있다.

V. 결 론

본 논문에서는 차별화된 서비스를 제공하기 위한 DiffServ의 스케줄러로서 다양한 스케줄러들이 있지만 그 중에서 PQ(Priority Queue)와 WRR(Weighted

Round Robin)이 사용되어질 때의 단점들을 보완하고자 각 클래스에 입력되는 큐를 처리하는 스케줄러에 퍼지 이론(fuzzy theory)을 적용시켜 각 클래스의 가중치를 공평하게 할당하는 방법을 제안하였다.

본 논문에서 제안한 퍼지 이론을 이용한 스케줄러에 적용시킴으로써 DiffServ 구조에서 가장 우선순위가 높은 EF 클래스의 QoS를 보장해 주기 위하여 우선적으로 가중치를 할당을 받아 입력되는 트래픽의 처리를 보장해주면서 AF 클래스들도 서로 남는 가중치가 있을 때에는 입력되는 트래픽이 폭주가 발생되어진 클래스에 가중치를 주고받게 함으로써 AF 클래스의 QoS와 효율성을 보장을 해주게 되었다.

DE 클래스는 PQ 스케줄러에서 보여진 입력되는 패킷들의 높은 폐기율과 지연의 단점을 보완하고자 최저 보장 서비스를 제공하고, EF와 AF 클래스에 가중치가 필요할 때에는 수시로 자신의 최저 할당된 가중치를 제외한 나머지 가중치를 할당 해줌으로써 EF와 AF 클래스에 폭주 상태가 발생 하였을 때에 폭주 상태를 완화시키면서 자신에게 입력되는 트래픽 또한 WRR 보다는 조금 떨어지지만 PQ 보다는 패킷 처리율이 향상되었다.

현재 사용이 되어지고 있는 스케줄러 방식인 PQ 보다는 DE 클래스에서 패킷 폐기율은 대략 50% 정도의 성능 향상을 보였다. 그리고 기존 WRR 방식 보다는 EF 클래스에서 패킷 폐기율이 평균 12% 정도의 성능 향상을 보였다.

또한 퍼지 제어기의 동작 주기에 있어서 항상 고정된 주기로 제어하여 스케줄러가 동작하도록 하는 방법을 사용하였을 때보다 유동적으로 트래픽의 양과 현재 클래스의 상태를 파악하여 퍼지 제어기를 동작하게 하여 스케줄러의 동작 주기를 유동적으로 동작하게 함으로써 단점을 효과적으로 보완하였다.

참 고 문 헌

[1] C. Kalmanek, H. Kanakia, S. Keshav, Rate Controlled Servers for Very High Speed Networks," in IEEE Global Telecommunications Conference, Dec. 1990.

[2] S. Shenker, C. Partridge, R. Guerin, "Specification if Guaranteed Quality of Service to Intergrated Services(IntServ)," RFC 2212, Sep. 1997.

[3] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field(DS Field) in the IPv4 and IPv6 Headers," RFC 2474, Dec. 1998.

[4] S. Blake, D. Black, M. Carlson, Wang and Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.

[5] V. Jacobson, K. Nichols, Cisco Systems, K. Poduri, Bay Networks, "An Expedited Forwarding PHB," RFC 2598, June 1999.

[6] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB," RFC 2597, June 1999.

[7] A. Parekh, R. Gallager, "A Generalized Processor Sharing Approach to Flow Control - the Single Node Case," in Proceedings of Infocom '92, vol. 2, pp. 915-924, May 1992.

[8] Chiung-Shien Wu, "Link-sharing method for ABR/UBR services in ATM networks," Computer Communications 21, pp. 1131-1142, 1998.

[9] 정경택, 박준성, 박현, 전병실, "ATM ABR의 공평성들을 위한 새로운 스케줄링 알고리즘" 대한전자공학회 논문지 제39권 TC편 제4호, Apr. 2002.

[10] S. Floyd, Ramakrishna Gummadi, and Scott Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," unpublished, Aug. 2001.

저 자 소 개

정 경 택(정회원)
제41권 TC편 제1호 참조

김 변 곤(정회원)
제39권 TC편 제12호 참조

전 병 실(정회원)
제41권 TC편 제1호 참조

박 준(정회원)
1998년 여수대학교 전자통신공학 과 공학사
2003년 전북대학교 전자공학과 공학석사
2006년 현재 전북대학교 전자공학과 박사과정
<주관심분야 : 통신/음성/오디오 신호처리, DiffServ, QoS>