DiffServ 망에서 QoS를 보장하기 위한 새로운 동적 가중치 할당 알고리즘 개발

New Dynamic WRR Algorithm for QoS Guarantee in DiffServ Networks

정동수*, 김변곤*, 박광채**, 조해성***
국립군산대학교 전자정보공학부*, 조선대학교 전자공학과**, 건양대학교 전자정보공학과***

Dong-Su Chung(j0404@kunsan.ac.kr)*, Byun-Gon Kim(bgkim@kunsan.ac.kr)*
Kwang-Chae Park(kcpark@chosun.ac.kr)**, Hae-Seong Cho(hscho@konyang.ac.kr)***

요약

DiffServ망에서 많이 거론되고 있는 대표적인 스케줄러로 PQ(Priority Queue), WRR(Weighted Round Robin)등의 스케줄러가 연구되어져 있다. 그러나 이러한 스케줄링 방식들은 약간의 단점을 가지고 있다. 본 논문에서는 PQ와 WRR의 단점을 보완하여 WRR 스케줄러에 적용이 가능한 스케줄러 알고리즘을 제안한다. 본 논문에서 제안된 알고리즘은 DiffServ 망에서 각 클래스의 큐 상태를 체크하여 퍼지 이론을 적용한 제어 정책에 따라 WRR 스케줄러의 가중치를 동적으로 할당하였다. 제안된 알고리즘의 성능평가를 위하여 네트워크 시뮬레이터(NS-2)를 이용하여 컴퓨터 시뮬레이션을 수행하였다. 시뮬레이션 결과 제안된 알고리즘은 EF 클래스의 패킷 손실률에서 WRR 스케줄러 방식보다 평균 6.5% 향상되었으며, AF4 클래스에서는 PQ 방식보다 평균 45% 향상된 결과를 보였다.

■ 중심어 : | DiffServ | 스케줄링 | 우선순위 | WRR |

Abstract

There are two traditional scheduling methods known as PQ and WRR in the DiffServ network, however, these two scheduling methods have some drawbacks. In this paper, we propose an algorithm that can be adopted in WRR scheduler with making up for weak points of PQ and WRR. The proposed algorithm produces the control discipline by the fuzzy theory to dynamically assign the weight of WRR scheduler with checking the Queue status of each class. To evaluate the performance of the proposed algorithm, We accomplished a computer simulation using NS-2. From simulation results, the proposed algorithm improves the packet loss rate of the EF class traffic to 6.5% by comparison with WRR scheduling method and that of the AF4 class traffic to 45% by comparison with PQ scheduling method.

■ keyword : | DiffSserv | Scheduling | QoS | WRR |

* 본 논문은 2004년도 군산대학교 학술연구비의 지원에 의하여 연구되었습니다.

접수번호 : #060511-001
접수일자 : 2006년 05월 11일
심사완료일 : 2006년 07월 06일
교신저자 : 조해성, e-mail : hscho@konyang.ac.kr

I. 서 론

최근 인터넷에서 초고속 네트워크 실현으로 VoIP(Voice of IP), VPN(Virtual Private Network), VOD(Video On Demand), 화상회의 등과 같은 다양한 형태의 응용서비스를 등장시키고 있다. 이러한 서비스들은 다양한 수준의 서비스 품질을 요구하지만, 최선형 서비스(Best-Effort Service)를 근간으로 한 현제의 인터넷은 서비스 품질에 대한 고려가 없기 때문에 다양한 서비스 제공에 많은 문제점을 가지고 있다. 이러한 문제들을 해결하기 위해 새로운 IP QoS(Quality of Service)모델 등에 관한 연구가 진행되어 왔다[1].

QoS에 관련된 많은 연구 중에서 모든 패킷 흐름에 대해서 QoS를 제공해주기 위한 IntServ 모델은 현실적으로 구현이 복잡하고 오버헤드가 많기 때문에 확장성이 나쁜 단점이 있다[2]. 이러한 단점을 극복하기 위해 IETF(Internet Engineering Task Force)에서는 각기 다른 사용자 그룹에게 서로 다른 수준의 서비스를 제공할 수 있는 DiffServ 모델을 제안하게 되었다[3].

DiffServ는 복잡한 기능의 처리는 경계(edge) 라우터에서 수행하게 하고 코어(core) 라우터는 각 클래스의 PHPs(Per -Hop Behaviors)에 따라 패킷을 전달함으로써 확장성을 향상시켰다[4]. DiffServ에서 제공하는 서비스는 일반적으로 EF(Expedited Forwarding), AF (Assured Forwarding), DE(Default) 클래스로 나누어 진다. EF 클래스는 가장 우선순위가 높은 클래스로서 인터넷 전화, 화상회의 등에 알맞은 작은 패킷 지연시간과 지터, 작은 패킷 폐기율을 제공받는다. AF 클래스는 지연시간과 지터에 민감하지 않고 최선형 서비스보다는 좋은 서비스를 제공받으며, DE 클래스는 인터넷 최선형 서비스와 같은 수준의 서비스를 제공받을 수 있다. 이러한 차별화된 서비스를 받기 위해서 가입자는 미리 ISP(Internet Service provider)와 SLA(Service Level Agreement)를 맺어야 한다. SLA는 가입자와 ISP, 또는 ISP간의 계약으로 제공되는 서비스클래스와 각 클래스별로 허용된 트래픽 양을 트래픽 프로필에 명시하고 있다. 하지만, IETF에서는 라우터 내부의 구현 메커니즘에 대해서는 언급하고 있지 않으며 단지 외부에 드러난 패킷의 입출력 동작만을 정의하고 있다. 따라서 수많은 트래픽이 DiffServ를 통과할 때 각 클래스의 SLA에 따른 QoS를 제공하기위한 효율적인 스케줄링 기법의 연구가 필요하다. 현재 DiffServ에서 많이 거론되고 있는 대표적인 스케줄링 기법으로는 PQ(Prio-rity Queue), WRR(Weighted Round Robin)와 WFQ (Weighted Fair Queueing)등의 스케줄링 기법이 연구되어져 있다. 그러나 이러한 기법들은 장점을 가지고 있는 동시에 단점을 가지고 있다.

PQ는 클래스의 우선순위에 따라서 서비스해주는 방식이므로 EF 클래스에 높은 수준의 QoS를 제공할 수 있으나 EF 클래스의 트래픽이 증가하면 우선순위가 낮은 클래스들의 QoS를 보장해 주지 못한다는 단점이 있다. 또한, WRR은 각 클래스의 가중치에 따른 서비스를 제공하기 때문에 각 클래스가 공평하게 서비스를 받을 수 있는 장점이 있지만, EF 클래스의 트래픽이 집중되는 경우에 EF 클래스의 QoS를 보장해 줄 수 없는 단점을 가지고 있다.

따라서, 본 논문에서는 PQ와 WRR의 단점을 보완할 수 있는 스케줄링 기법을 제안한다. 본 논문에서 제안된 알고리즘은 유동적으로 변화하는 각 클래스의 큐 상태를 체크하여 큐의 상태에 따라 각 클래스 큐의 가중치를 동적으로 할당함으로써 보다 효율적인 서비스가 가능하다. 유동적으로 각 클래스의 큐 상태를 체크하기위하여 퍼지이론을 적용하였으며, 퍼지이론을 통하여 퍼지 제어 규칙을 생성하여 클래스가 가지고 있는 큐의 가중치를 효율적으로 할당하도록 하였다.

본 논문의 구성은 제 2장에서 관련 연구에 대하여 살펴보고 문제점을 분석하여, 제 3장에서는 제안된 스케줄링 알고리즘을 설명한다. 제 4장에서는 ns-2 시뮬레이터를 이용하여 기존 연구 및 제안된 알고리즘의 성능을 비교 분석한다. 마지막으로 제 5장에서 결론을 맺는다.

II. 관련 연구

1. DiffServ 개요

DiffServ는 IntServ와 함께 IETF에서 제시가 되어진 QoS를 제공하기 위한 대표적인 방법이다. DiffServ는 패킷을 몇 개의 클래스로 나누어서 각각의 클래스 마다 정의된 QoS를 제공해 주는 방법으로써 IntServ에 비하여 확장성이 좋은 장점을 가지고 있다[3][5].

DiffServ를 기반으로 하는 망에서는 망 내로 유입되는 트래픽이 망의 경계에서 분류되고 조절되며, 개별적인 BAs (behavior aggregates)가 할당되는 간단한 구조를 기반으로 하고 있다. 여기서 BA란 같은 DSCP를 갖는 사용자 흐름들의 군집을 의미한다. 한편, 망 내에서는 패킷들이 DSCP와 연관된 PHB (Per-Hop Behavior)에 따라 전달된다. [그림 2.1]에서는 DiffServ 모델의 망 구성 요소들을 보여주고 있다[5].

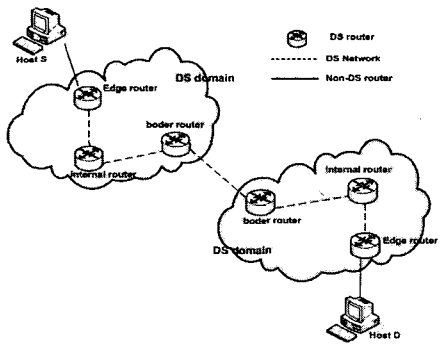


그림 2.1 DiffServ 네트워크 구성

DiffServ는 다양한 어플리케이션들의 서비스를 분류하는 간단한 방법이다. 다른 것들이 가능하다 하더라도 두 개의 서비스 수준(traffic class)을 효율적으로 표현하는 두 개의 표준 PHB와 QoS를 제공하지 않는 PHB도 정의가 되어있다[5][6].

i. Expedited Forwarding (EF) : EF 클래스는 지연(delay)과 지터(jitter)를 최소화시키며 가장 높은 수준의 aggregate 서비스 품질을 제공하고 규정된 트래픽 프로파일(Traffic Profile)을 초과하는 어떤 트래픽도 무시된다.

ii. Assured Forwarding (AF) : AF는 4개의 클래스(class)를 가지며, 각 클래스에는 3개의 폐기 우선순위가 있다. AF 는 일정 전송률을 보장해주며 초과 부분은 손실될 가능성이 높은 클래스이다.

iii. Default (DE) : DiffServ를 지원하지 않는 사용자를 허용하기 위한 형태로서 IP/4 에서의 Best Effort 서비스와 같은 방식으로 서비스되며 가장 낮은 우선순위를 갖는다.

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 QoS를 제공받는다.

2. 스케줄링 기법

스케줄러는 자원 할당이 필요한 곳에서 널리 사용되고 있다. DiffServ에서는 각 클래스(EF, AFs, DE) 간에 링크 자원을 할당하기 위해서 스케줄러가 사용된다. [그림 2.2]는 스케줄링의 기본적인 문제를 표현하고 있다. 스케줄러는 각 클래스의 흐름을 분리하여 네트워크의 협약사항을 준수하지 않는 연결은 다른 연결들의 서비스를 가로채거나 방해하지 말아야 한다. 그리고 링크자원을 공평하게 분배하고 자원의 이용 효율성을 높일 수 있어야 한다.

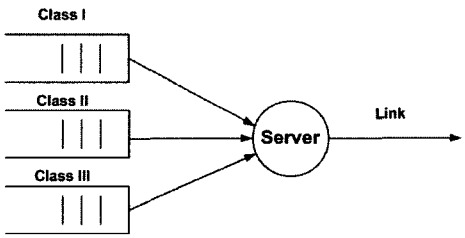


그림 2.2 패킷 스케줄링의 기본적인 문제

여기서는 DiffServ에서 고려되고 있는 PQ(Priority Queue)와 WRR(Weighted Round Robin) 스케줄러에 대해서 알아보고, 각각의 문제점에 대해서 살펴보고자 한다.

PQ 방식은 우선순위가 높은 클래스 큐가 항상 먼저

서비스 받는 방식이다[7]. 따라서 이 방식은 우선순위가 높은 트래픽에게는 짧은 지연과 작은 지터를 보장해 주고, 링크 용량도 많이 할당 받을 수 있도록 해준다. DiffServ에서의 우선순위는 EF, AFs, DE 순 이므로, PQ는 EF PHP를 구현하기에 매우 좋은 스케줄링 기법이다. 그러나 [그림 2.3]에서와 같이 EF 트래픽의 폭주가 발생할 경우에 상대적으로 우선순위가 낮은 AF4, DE 클래스 트래픽이 서비스 받지 못 할 수 있다. 특히, AF4 클래스는 최소 대역을 보장해 주어야 하는 클래스이므로 DiffServ 네트워크 전체에 나쁜 영향을 줄 수 도 있다.

WRR에서는 각각 클래스 큐에는 가중치(weight)가 주어지고 round robin 방식으로 서비스를 받게 된다. 즉, 링크 용량을 가중치와 평균 패킷 크기에 따라 분배해서 사용하는 것이다. 따라서, WRR은 어느 특정 클래스가 링크 용량을 독점하는 것을 방지한다. 따라서 우선순위가 높은 클래스의 양에 상관없이 각각의 클래스는 자신의 링크 용량을 사용할 수 있어서 각 클래스별로 QoS를 보장해 줄 수 있다[8]. 하지만 WRR은 그림 2.3에서와 같이 EF 트래픽의 폭주가 발생할 경우에 EF의 QoS를 보장해줄 수 없다는 단점이 있다.

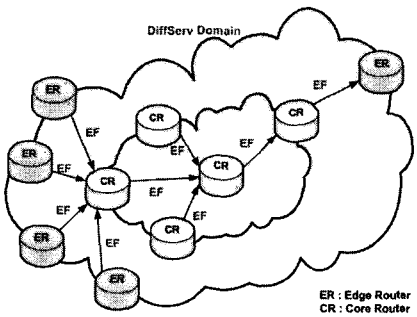


그림 2.3 PQ와 WRR의 문제점

이러한 문제를 해결하기 위한 관련 연구는 EF 트래픽의 폭주상황이 발생하면 PQ 방식을 사용하고 그렇지 않을 경우에는 WRR 방식을 사용하는 방법[9], 트래픽 상태와 정책에 따라 PQ와 WFQ 방식을 혼합하여 사용 방법[10], 각 클래스 트래픽 입력 상황에 따라 가중치를 동적으로 변화시키는 방법[11] 등이 제안되었다.

III. 제안된 알고리즘

DiffServ 구조에서 트래픽 흐름 제어를 위하여 본 논문에서는 스케줄러 알고리즘에 퍼지이론을 적용시킨 알고리즘을 제안한다.

2장에서 설명 했던 바와 같이 PQ는 EF 클래스의 서비스를 중점으로 하기 때문에 우선순위가 낮은 나머지 클래스는 서비스를 제대로 받지 못하는 단점을 가지고 있고, WRR은 특정 클래스가 링크 용량을 독점하는 문제는 없지만 DiffServ인 경우에는 망 안에서 EF 트래픽이 집중되는 경우에 EF 클래스의 QoS를 보장해 주지 못하는 단점을 보이고 있다.

이러한 문제점들을 보완하여 보다 효율적으로 각 클래스별로 QoS를 보장해주고자 퍼지 이론을 이용하여 각 클래스 큐의 상태정보를 구한다. EF 클래스에 폭주상태가 되어 자원이 부족하게 되는 경우에는 AFs 클래스나 DE 클래스의 가중치를 임시로 빌려와서 EF 클래스의 폭주 상태를 완화 시키고, EF 클래스의 폭주 상태가 완화되면 임시로 빌려온 가중치를 다시 되돌려 주는 스케줄러 알고리즘을 제안함으로써 기존의 방법보다 향상된 성능을 보여주고자 한다.

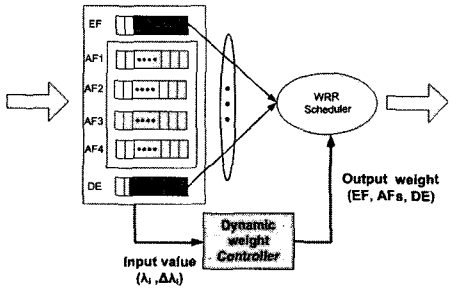


그림 3.1 제안된 스케줄러 구조

[그림 3.1]은 제안된 스케줄러의 구조를 나타내고 있다. 동적 가중치 제어기는 각 클래스 큐의 셀 수(λ)와 변화량($\Delta\lambda$)이 입력되면 퍼지 이론을 이용하여 큐의 상태정보 값을 계산하고 클래스의 우선순위와 상태정

보에 따라서 각 클래스 큐의 가중치를 제어한다.

1. 퍼지 제어 규칙

각 클래스 큐의 가중치를 동적으로 할당하기 위하여 각 클래스 큐의 상태에 따라 두 개의 입력과 한 개의 출력을 갖는 퍼지 제어 규칙을 생성한다. 퍼지 제어 규칙 생성을 위한 두 개의 입력은 각 클래스 큐의 현재 셀 수(qLen)와 셀 수의 변화량(dq)이다. 출력 값은 각 클래스 큐의 상태에 따라 -1.0~1.0 사이의 큐 상태(QSt) 값으로써, 이러한 큐 상태 값을 임계치(threshold)와 비교하여 각 클래스 큐의 상태에 따른 제어를 수행하게 된다.

제안된 퍼지 제어 규칙을 생성하기 위하여 [그림 3.2]와 같은 사다리꼴 맴버쉽 함수를 수식 1과 같이 정의한다.

$$f(x;a,b,c,d) = \begin{cases} 0 & \text{for } x \le a, \\ \dfrac{x-a}{b-a} & \text{for } a \le x \le b, \\ 1 & \text{for } b \le x \le c, \\ \dfrac{d-x}{d-c} & \text{for } c \le x \le d, \\ 0 & \text{for } x \ge d. \end{cases} \quad (1)$$

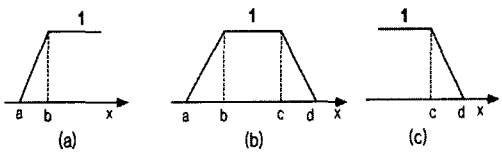


그림 3.2 함수 f 정의 (a) $f(x;a,b,+\infty,+\infty)$, (b) $f(x;a,b,c,d)$, (c) $f(x;-\infty,-\infty,c,d)$

퍼지 입력 변수로서 각 클래스 큐의 현재 셀 수와 변화량을 가지고 $|T(\text{qLen})| \times |T(\text{dq})|$와 같은 2차원의 배열이 형성된다. $|T(X)|$는 $T(X)$의 언어변수 항들의 수이다. 이와 같이 두 가지 변화량(qLen, dq)을 가지고 퍼지 입력 용어 집합을 다음과 같이 정의 하고,

$$T(\text{qLen}) = \{\text{GL}, \text{L}, \text{MD}, \text{H}, \text{GH}\}$$

$$T(\text{dq}) = \{\text{Decrease}, \text{Maintain}, \text{Increase}\}$$

퍼지 출력 용어 집합으로 큐 상태(QSt)값의 용어 집합을 다음과 같이 정의할 수 있다.

$$T(\text{QSt}) = \{\text{NB}, \text{NM}, \text{ZO}, \text{PM}, \text{PB}\}$$

퍼지 입출력 맴버쉽 함수를 다음과 같이 정의한다.

$$\mu_{\text{GL}}(\text{qLen}) = f(\text{qLen}; 0, 0, 0, 0.2\text{B})$$
$$\mu_{\text{L}}(\text{qLen}) = f(\text{qLen}; 0, 0.2\text{B}, 0.2\text{B}, 0.4\text{B})$$
$$\mu_{\text{MD}}(\text{qLen}) = f(\text{qLen}; 0.2\text{B}, 0.4\text{B}, 0.4\text{B}, 0.6\text{B})$$
$$\mu_{\text{H}}(\text{qLen}) = f(\text{qLen}; 0.4\text{B}, 0.6\text{B}, 0.6\text{B}, 0.8\text{B})$$
$$\mu_{\text{GH}}(\text{qLen}) = f(\text{qLen}; 0.6\text{B}, 0.8\text{B}, 1\text{B}, 1\text{B})$$
$$\mu_{\text{D}}(\text{dq}) = f(\text{dq}; -\infty, -\infty, -5, 0)$$
$$\mu_{\text{M}}(\text{dq}) = f(\text{dq}; -5, 0, 0, 5)$$
$$\mu_{\text{I}}(\text{dq}) = f(\text{dq}; 0, 5, +\infty, +\infty)$$
$$\mu_{\text{NB}}(\text{QSt}) = f(\text{QSt}; -1, -1, -1, -0.5)$$
$$\mu_{\text{NM}}(\text{QSt}) = f(\text{QSt}; -1, -0.5, -0.5, 0)$$
$$\mu_{\text{ZO}}(\text{QSt}) = f(\text{QSt}; -0.5, 0, 0, 0.5)$$
$$\mu_{\text{PM}}(\text{QSt}) = f(\text{QSt}; 0, 0.5, 0.5, 1.0)$$
$$\mu_{\text{PB}}(\text{QSt}) = f(\text{QSt}; 0.5, 1.0, 1.0, 1.0)$$

위에서 규정한 용어 집합을 기반으로 퍼지 제어규칙을 만들 수 있는데 총 15가지의 결과를 만들 수 있는데 [표 3.1]에서 보여주는 바와 같다.

표 3.1 퍼지 제어 규칙

	셀 수의 변화량(dq)	현재 셀 수 (qLen)	요구 가중치 (QSt)
1	D	GL	NB
2	D	L	NM
3	D	MD	Z0
4	D	H	Z0
5	D	GH	PM
6	M	GL	NB
7	M	L	NM
8	M	MD	Z0
9	M	H	PM
10	M	GH	PB
11	I	GL	NM
12	I	L	Z0
13	I	MD	PM
14	I	H	PB
15	I	GH	PB

퍼지 집합으로 추론되어진 결과로부터 제어를 수행하기 위해서 비퍼지화 (defuzzification) 단계가 필요하다. 비퍼지화 값을 구하는 단계는 [그림 3.3]에서 보여주고 있다. 두개 이상의 제어 규칙에 의하여 추론이 될 수 있다. 두개의 입력값(λ, $\Delta\lambda$)이 입력되면 입력값에 의하여 퍼지 맴버함수 값이 결정된다. 이중에서 최소의 입력된 값으로 구한 입력 맴버 함수 값으로 출력 맴버함수 값을 자르게 되면 사각형의 모양 두개가 나오는데 이 두개의 면적 중심값을 구하게 되면 비퍼지화 되어진 값을 구할 수 있다. 이렇게 구해진 각 클래스 큐의 비퍼지 값을 상태정보라 하고, 각 클래스의 상태정보에 따라서 폭주가 발생하였는지 아닌지를 판단하여 가중치를 동적으로 컨트롤 할 수 있다.

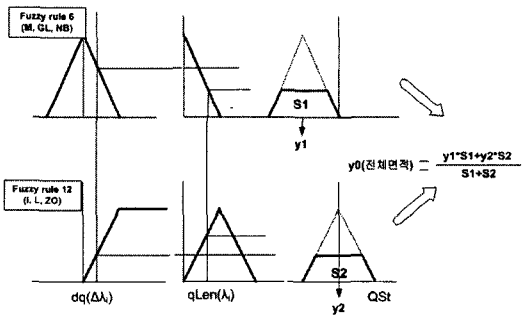


그림 3.3 비퍼지화 단계

2. 동적 가중치 할당 알고리즘

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

DiffServ 모델에서 사용자 패킷은 트래픽 특성에 따라 EF, AFs, DE 클래스로 집합화 되고 서비스 받는다. 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 상태 정보 값이 임계치(thHigh) 보다 크다면 폭주가 발생하여 패킷 손실이 발생할 수 있으므로 AF 클래스 내에서 남는 가중치가 있는지 찾는다. 만약에 AF 클래스 큐의 상태 정보 값이 임계치(thLow) 보다 작다면 AF 클래스의 가중치를 감소시키고, EF 클래스의 가중치를 증가 시켜준다. 만약에 AF 클래스에 남는 자원이 없다면 DE 클래스의 가중치를 감소시켜 EF 클래스의 가중치를 증가 시켜 준다. 이러한 방식으로 각 클래스 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 클래스에 반환하고 AF 클래스에서 빌려온 가중치가 없다면 DE 클래스에 반환하게 된다. [그림 3.4]는 제안된 알고리즘의 흐름도이다.

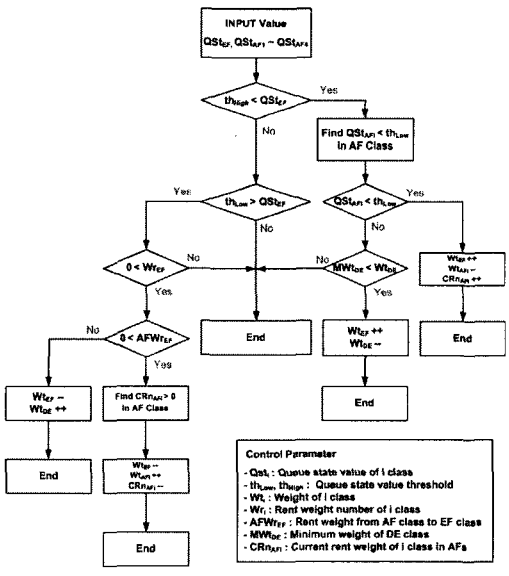


그림 3.4 제안된 알고리즘 흐름도

Ⅳ. 시뮬레이션 및 성능 평가

1. 시뮬레이션 환경

본 논문에서 제안된 알고리즘의 시뮬레이션은 NS-2의 DiffServ 노드를 기반으로 [그림 4.1]과 같은 망을 구성해서 시뮬레이션 하였다. [그림 4.1]에서 Source 1~Source 6은 트래픽을 발생시키는 노드로써 트래픽은 포아송 과정으로 발생되며, Dest 1~Dest 6은 발생된 패킷을 받는 노드이다. R1, R2, R3은 DiffServ 기능을 가지고 있는 라우터로써 R1, R3은 edge 라우터이고 R2는 core 라우터이다. 즉, R1, R3은 소스 노드로부터 패킷을 받으면 그 패킷을 분류, 측정, 표시, 쉐이핑/폐기를 하고 R2로 넘겨준다.

R2~R3 사이의 링크에 bottleneck를 두고, R2의 큐의 크기를 50packets로 해서 패킷 폐기와 지연이 발생하게 하였으며, 나머지 링크는 용량을 크게 하여 시뮬레이션에 영향을 주지 않도록 하였다.

R1 라우터에 6개의 클래스가 접속되어 있다. 클래스의 구성은 가장 우선순위가 높은 EF 클래스와 AF1~AF4까지 4개의 클래스로 구성되어 있는 AF 클래스와 우선순위가 가장 낮으면서 최저 서비스만을 기다리는 DE 클래스로 구성 되어있다. 모든 라우터에서 큐를 관리하기 위한 방법으로는 EF 클래스에서는 Drop Tail, AF 클래스에서는 RED, DE 클래스에서는 Drop Tail 방법을 사용하였으며 시뮬레이션은 100초 동안 수행하였다.

[그림 4.2]는 입력되는 트래픽의 양과 라우터 내부의 큐의 상태를 보여주고 있다. EF 클래스의 입력 트래픽은 EF 클래스의 링크 용량 6.7Mbps를 기준으로 입력부하가 0.6에서 1.1까지 시뮬레이션 하였으며, 특히 AF4 클래스의 입력율은 1.5Mbps인데, 이는 AF4 클래스의 남는 영역이 발생하게 된다. 이와 같이 어떤 클래스에 남는 영역이 있을 경우에는 남는 영역을 어떻게 하면 효율적으로 사용할지가 중요한 관점이다. 일반적으로 WRR 방식은 남는 영역이 발생할 경우에 가중치에 비례하는 양 만큼 분배하여 사용 되는데, 제안된 알고리즘에서는 우선순위가 가장 높은 EF 클래스에 할당하여 WRR 방식의 단점을 보완하였다.

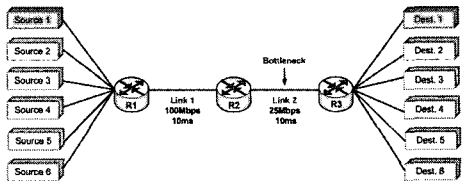


그림 4.1 NS-2로 구성한 시뮬레이션 네트워크 모델

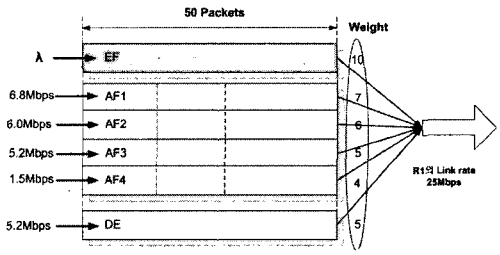


그림 4.2 라우터 내부 큐 상태

2. 시뮬레이션 결과 및 분석

시뮬레이션 결과는 DiffServ에서 많이 사용되어지고 있는 방식인 PQ방식과 WRR 방식, 그리고 제안된 방식을 비교분석하여 나타낸다. 제안된 방식은 [표 4.1]과 같은 퍼지 제어 파라미터에 따라 4가지 방식을 사용하였다. [표 4.1]에서 제어 주기는 라우터에서 각 클래스 큐의 가중치를 동적으로 할당하기 위한 시간이며, Min Wt DE와 Min Wt AF는 DE 클래스와 AF 클래스에 할당된 최소 가중치이다. 따라서, EF 클래스에 폭주가 발생하여 AF 클래스나 DE 클래스에서 가중치를 할당 받을 수 있으려면 AF 클래스나 DE 클래스에서 최소 가중치 이상을 가지고 있어야 한다. 이는 우선순위가 낮은 AF 클래스와 DE 클래스의 최소 QoS를 보장하기 위한 파리미터이다.

표 4.1 퍼지 제어 파라메터

Parameter	PRO	PRO_1	PRO_2	PRO_3
제어 주기	50ms	50ms	50ms	50ms
Min Wt DE	3	4	4	5
Min Wt AF	3	2	1	0

[그림 4.3]은 EF 클래스의 평균 패킷 폐기율을 보여

주고 있다. PQ의 경우에는 링크의 가중치를 우선순위가 가장 높은 EF 클래스에서 우선적으로 사용할 수 있기 때문에 PQ에서의 성능은 현저하게 좋으나 WRR의 경우 입력이 많아질수록 EF 클래스의 폐기율이 증가하는 것을 볼 수 있다.

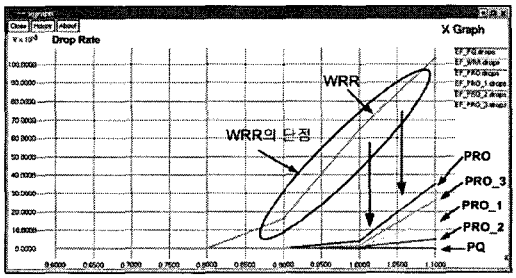


그림 4.3 EF 클래스의 패킷 폐기 확률

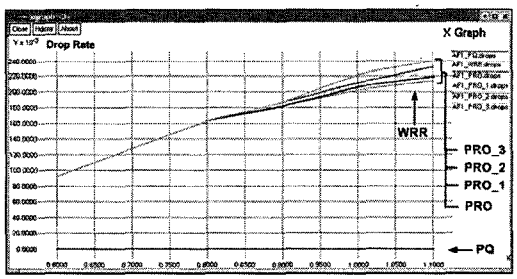


그림 4.4 AF1 클래스의 패킷 폐기 확률

제안된 알고리즘은 EF 클래스의 트래픽이 폭주하여 손실이 발생할 경우 가중치를 동적으로 할당함으로써 EF 클래스의 패킷 손실을 줄일 수 있어 WRR 방식의 단점을 보안할 수 있다. [그림 4.4][그림 4.5][그림 4.6]은 AF1, AF2, AF3 클래스의 패킷 손실율을 보여주고 있다. 이들은 유사한 결과를 보여주고 있으며 제안된 알고리즘의 손실율이 WRR 보다 약간 높게 나타나고 있는데 이는 fairness 때문에 생기는 현상이다.

예를 들면 클래스 ABC의 가중치가 각각 4,2,4 라면 AAAABBCCCC순서로 서비스 하게 된다. 그런데 A 클래스 큐의 서비스슬롯에 한 개의 패킷 밖에 없다면 ABBCCCC 순서로 서비스 한다. 위와 같은 상황이 70개의 슬롯에 계속된다면, 클래스 ABC는 각각 10, 20, 40개의 패킷을 서비스 받게 된다. 그러나, 정상적인 경우 클래스 ABC는 각각 28, 14, 28개의 패킷을 서비스 받게 된다. 위의 서비스 패킷 숫자를 비교해보면 클래스 A가 사용하지 않는 18개의 서비스 슬롯을 클래스 B와 C가 가중치의 비에 따라 각각 6개와 12개의 패킷을 서비스 받고 있다는 것을 알 수 있다.

즉, WRR의 경우 AF4클래스의 남는 영역을 EF, AF1, AF2, AF3, DE 클래스에 나뉘어져 사용되지만 제안된 알고리즘의 경우에는 AF4 클래스에 여분의 대역폭이 존재하고, EF 클래스 트래픽이 폭주한다면, AF 클래스가 사용하지 않는 가중치를 줄이고 EF 클래스에 가중치에 더해줌으로써 우선순위가 높은 EF 클래스의 패킷 손실률을 줄일 수 있다.

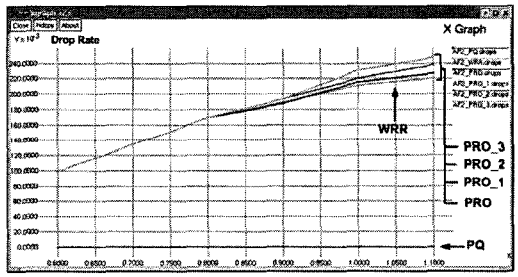


그림 4.5 AF2 클래스의 패킷 폐기 확률

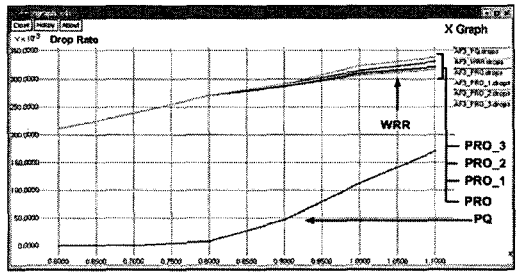


그림 4.6 AF3 클래스의 패킷 폐기 확률

[그림 4.7][그림 4.8]은 AF4 클래스와 DE 클래스의 손실율을 보여주고 있는데, PQ 방식은 우선순위가 낮은 클래스의 손실을 피할 수 없는 단점이 있으나, 제안된 알고리즘은 PQ의 이러한 단점을 보완하여 패킷 손실율과 평균 지연시간이 현저하게 떨어진 것을 알 수 있다. 특히 AF4 클래스의 경우에는 입력률이 낮음에도 불구하고 PQ 방식에서는 패킷 손실율이 매우 높게 나타나는 단점이 있다. 이는 최소 대역을 보장해 주어

야 하는 AF 클래스의 요구조건을 만족시킬 수 없는 문제가 발생한다. 제안된 알고리즘에서는 AF4 클래스의 남는 대역을 EF 클래스에 효율적으로 할당하여 AF4 클래스의 손실을 발생시키지 않고 EF 클래스의 패킷 손실을 줄일 수 있다.

제안된 알고리즘의 파라미터 특성을 보면, 최소 가중치 값이 작을 수 록 EF 클래스의 패킷 손실률은 줄고, DE 클래스의 손실은 증가하는 특성을 보이고 있다. 이는 최소 가중치 값이 작을 수 록 EF 클래스에 폭주가 발생할 경우 더 많은 가중치를 EF 클래스에 할당할 수 있기 때문이다.

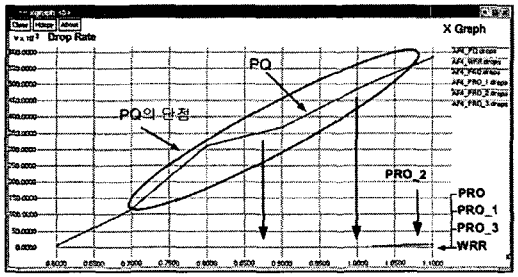


그림 4.7 AF4 클래스의 패킷 폐기 확률

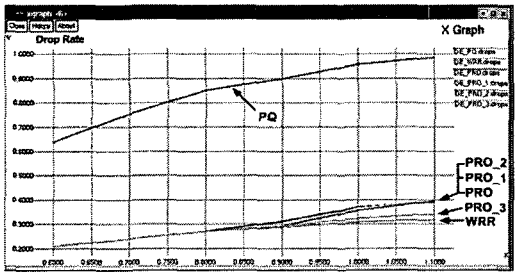


그림 4.8 DE 클래스의 패킷 폐기 확률

표 4.2 패킷 폐기율 비교

Class / Scheduler	EF	AF1	AF2	AF3	AF4	DE
PQ	0%	0%	0%	12%	48%	92%
WRR	6.7%	21%	26%	32%	0%	31%
PROposed	0.3%	21.5 %	26.5 %	32%	0%	35%

[표 4.2]는 각 클래스별 패킷 폐기율을 나타내고 있다.

제안된 알고리즘은 WRR을 기반으로 가중치를 동적으로 제어하여 WRR의 단점인 EF 클래스의 손실율을 줄일 수 있고, PQ의 단점인 AF4 클래스의 손실율을 줄일 수 있었다. DE 클래스는 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 PQ 방식과 비교하면 많은 성능 향상을 보이고 있다.

따라서, 우선순위가 높은 클래스의 패킷 폐기율이 낮으면서 우선순위가 낮은 클래스에서도 성능이 향상되었음을 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 큐의 상태에 따라 각 클래스의 가중치를 공평하게 할당하는 방법을 제안하였다.

본 논문에서는 WRR기반의 스케줄러에 동적 가중치 할당 알고리즘을 적용시킴으로써 DiffServ 구조에서 가장 우선순위가 높은 EF 클래스의 QoS를 보장해 주기 위하여 AF 클래스 중에서 남는 영역의 가중치를 우선 할당하고 남는 영역이 없는 경우에 DE 클래스의 가중치를 할당하여 EF 클래스의 QoS를 제공하였다.

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 완화 시키면서 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 55% 정도의 성능 향상을 보였다. 그리고 기존 WRR 방식 보다는 EF 클래스에서 패킷 폐기율이 평균 6.5% 정도의 성능 향상을 보였다.

향후 연구과제는 DiffServ 구조에 첨부 될 수 있는 많은 기능들의 적용이 계속 이루어져야 할 것이다. 예를들면 bandwidth broker 또는 수락제어(admission control), 자원예약(resource reservation), 정책(policing) 등과 같은 기능을 적용시켜 스케줄러의 성능을 향상시키는 연구가 필요하다.

참 고 문 헌

[1] X. Xiao and L. M. Ni, "Internet Qos: A Big Picture," IEEE Network, Vol.13, No.2, pp.8-18, Mar., 1999.

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

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

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

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

[6] V. Jacobson, K. Nichols, C. Systems, K. Poduri, and B. Networks, "An Expedited Forwarding PHB," RFC 2598, June, 1999.

[7] D. Stiliadis and A. Verma, "Design and Analysis of Frame-based Fair Queuing: A New Traffic Scheduling Algorithm for Packet-Switched Networks," to appear in ACM Sigmetrics, May, 1996.

[8] C. S. Wu, "Link-sharing method for ABR/UBR services in ATM networks," Computer Communications21, pp.1131-1142, 1998.

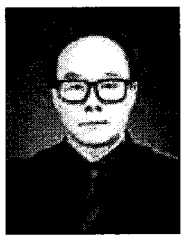
[9] J. Mao, W. M. Moh, and B. Wei, "PQWRR scheduling algorithm in supporting of diffserv", Communications, 2001. IEEE International Conference on, Vol.2, 2001.

[10] 남기호, *트래픽 모니터링과 정책에 기반한 QoS 보장 스케줄링 메카니즘*, 한국과학기술원 석사학위 논문, 2003.

[11] R. Kawahra and N. Komatsu, "A Scalable IP Traffic Control Method for Weighted Bandwidth Allocation per Flow," IEICE TRANS. COMMUN., Vol.E84-B, No.10 Oct., 2001.

저 자 소 개

정 동 수(Dong-Su Chung) 정회원



- 1975년 2월 : 한국항공대학교 항공통신공학과(공학사)
- 1982년 2월 : 전북대학교 전자공학과(공학석사)
- 1988년 2월 : 전북대학교 전자공학과(공학박사)
- 2006년 현재 : 국립군산대학교 전자정보공학부 정교수

<관심분야> : DiffServ 네트워크, BcN, Ad-Hoc

김 변 곤(Byun-Gon Kim) 종신회원

- 1990년 2월 : 한국항공대학교 항공전자공학과(공학사)
- 1997년 2월 : 전북대학교 전자공학과(공학석사)
- 2001년 8월 : 전북대학교 전자공학과(공학박사)
- 2005년 4월~현재 : 군산대 전자정보공학부 전임강사

<관심분야> : 초고속통신망, 광버스트, 트래픽제어

박 광 채(Kwang-Chae Park) 정회원

- 1975년 2월 : 조선대학교 전자공학과(공학사)
- 1980년 2월 : 조선대학교 전자공학과(공학석사)
- 1994년 8월 : 광운대학교 전자통신공학과(공학박사)
- 2006년 현재 : 조선대학교 전자공학과 정교수

<관심분야> : DiffServ, 디지털 교환기, OBS

조 해 성(Hae-Seong Cho) 종신회원

- 2001년 2월 : 전북대학교 전자공학과 대학원 졸업(공학박사)
- 2001년 9월~현재 : 건양대학교 전자정보공학과 교수

<관심분야> : 멀티미디어 통신, 유비쿼터스 네트워킹, 네트워크 QoS