인터넷 QoS 제공을 위한 CBQ 기반의 스케쥴링 기법

문 준 현[†]·김 광 현[††]

요 약

본 논문에서는 Differentiated Service의 개념과 구조를 살펴보고 DiffServ상에서 효율적인 패킷 전송을 위해 Class Base Queuing(CBQ)을 사용한 패킷 전송 모델을 제안한다. 제안된 모델에서는 실험을 통해 얻어진 RED 파라미터를 이용하여 RED min_{th}와 max_{th}를 설정함으로써 각각의 서비스 큐를 적절하게 제어하고 WRR(Weighted Round Robin)과 DRR(Deficit Round Robin) 같은 패킷 스케쥴링 기법을 통해서 모든 서비스 클래스들이 기아현상을 경험하지 않고 공평한 스케쥴링이 이루어지도록 하였다.

Scheduling Mechanism Based on CBQ for Providing Internet QoS

Jun-Hyun Moon[†]·Gwang-Hyun Kim[††]

ABSTRACT

This thesis includes the detailed explanation of the concept and structure of the Differentiated Service and suggests a new model for the packet transmission scheme using the Class Base Queuing (CBQ) method. The model has controlled each service queue moderately by setting the optimal RED mix_{th} and max_{th} gathered through a series of experiments. It also suggests a balanced scheduling method without the starvation effect of the service classes through the packet scheduling methods such as WRR and DRR.

키워드 : 차별화 서비스(Differentiated Service), 서비스 품질(Quality of Service), CBQ(Class Based Queuing), PHB(Per Hop Behavior)

1. 서 론

최근 인터넷 사용자의 급속한 증가와 함께 인터넷 기반의 전자상거래 시스템, 화상회의 시스템, VOD 서비스, 인터렉티브 게임 등과 같은 다양한 인터넷 응용 서비스들이 생겨나고 있다. 이러한 인터넷 응용 소프트웨어가 증가함에 따라 인터넷에서 다루고 있는 데이터의 종류도 기존의 단순한 텍스트 위주의 데이터에서 오디오, 비디오 데이터를 비롯해서 실시간성이 요구되는 스트리밍 데이터 등 다양한 멀티미디어 트래픽으로 변모하게 되었다. 이러한 다양한 멀티미디어 트래픽의 증가와 함께 인터넷상에서 보다 안정적이고 빠른 서비스를 받기 위한 QoS(Quality of Service) 요구가 생겨나게 되었다.

QoS 보장을 위한 가장 손쉬운 방법은 폭주가 일어나 지 않도록 백본(backbone) 네트워크의 대역폭을 충분히 크게 확보하는 방법이다. 그러나 단순히 대역폭의 크기만을 증가시키고 현재의 최선형 서비스(Best-effort) 구조를 그대로 가지고 있을 경우 다양한 응용 서비스들이 요구하는 QoS를 보장할 수 없다. 이에 따라 최근 몇 년 전부터 인터넷에서 QoS에 대한 요구가 심화되었으며, 이를 해결하기 위한 다양한 기술들을 연구하게 되었다. 이러한 노력의 일환으로, 기존의 최선형 서비스만을 제공하는 현재의 인터넷의 서비스를 개선하기 위해 IETF(Internet Engineering Task Force)에서는 다양한 서비스의 품질을 제공하는 여러 서비스 모델을 제시하고 있다. 이렇게 해서 제시된 서비스 중 가장 먼저 제안되었던 서비스가 인터넷상에서 서비스의 차별화를 위해 패킷의 흐름(flow)을 단위로 하여 QoS를 보장하는 Integrated Services(IntServ)로써 RSVP(Resource Reservation Protocol)라는 신호 프로토콜(signaling protocol)에 기반을 두고있다. 이 방식의 경우 확실하게 차별화 된 서비스를 제공할 수 있으나, 인터넷 기반 구조의 전반적인 수정을 요구하는 단점이 있으며, 확장성(scalability)에도 많은 문제점을 가지고 있다[1, 2].

이러한 IntServ의 단점 때문에, 흐름단위에서 벗어나 흐름들의 집합(aggregate) 단위로 서비스를 차별화하는 Differentiated Service(DiffServ)가 주목을 받게 되었다. 패킷 마킹 기술을 기반으로 하는 DiffServ는 흐름의 집합별로 차별화 된 서비스를 제공함으로써 IntServ에 비해 훨씬 간단하고, 확장성도 우수하고 간단히 소프트웨어만을 업그레이

† 정 회 원 : 광주대학교 인터넷방송국 연구원
†† 정 회 원 : 광주대학교 컴퓨터전자통신공학부 교수
논문접수 : 2001년 3월 24일, 심사완료 : 2001년 7월 16일

드함으로써 현재의 인터넷의 문제점인 QoS 문제를 어느 정도 개선할 수 있는 방식이다[6].

현재 인터넷 서비스를 위한 패킷 스케쥴링 기법 FIFO(First In First Out) 방식과 버퍼관리 기법인 나머지 버림(Tail dropping) 방식을 이용하여 모든 패킷에 동일한 QoS를 제공한다. 하지만 이러한 방식은 차별화 된 서비스를 제공하고자하는 DiffServ 구조에는 적합하지 않으므로 기존의 방식과 다른 패킷 스케쥴링과 버퍼 관리 방식이 필요하다. 따라서 본 논문에서는 DiffServ의 개념과 구조를 살펴보고 Diff-Serv에서 효율적인 패킷 전송을 위해 CBQ(Class Base Queuing)를 사용한 패킷 전송 모델을 제안한다. 제안된 모델은 기존의 토큰버킷 방법과 달리 트래픽 클래스 별로 서비스 우선순위가 다른 여러 개의 큐를 사용하여 트래픽을 차별적으로 서비스하는 방법이다. 이러한 CBQ를 다양한 형태의 데이터 타입과 프로토콜 유형 및 네트워크 상태에 적용하고 최선의 스케쥴링 기법에 대해 시뮬레이션을 통해 분석하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 먼저 IETF에서 제안하고 있는 DiffServ의 개념과 구조를 살펴보고 3장에서는 인터넷에서 차별화 된 서비스 제공을 위한 CBQ 기반의 패킷 스케쥴링 기법에 대해서 설명하며 4장에서는 시뮬레이션을 통해 제안된 기법의 성능을 평가 및 분석하고, 마지막으로 결론을 맺도록 한다.

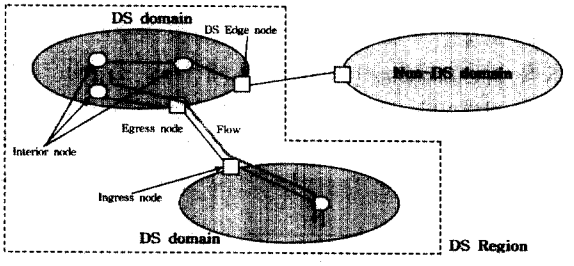
2. 인터넷 QoS 지원 서비스 모델

IP 기반의 QoS 제공 기술의 목적은 인터넷 상에서 어떤 사용자든지 시간과 장소에 관계없이 요구하는 조건에 합당한 서비스를 제공받도록 하는 것이다. 최근의 QoS 요구사항을 살펴보면 고속화/대용량의 처리가 가능하고 운영이 간편하면서도 신뢰성과 확장성이 있는 서비스를 요구하고 있다. IP 기반 QoS 기술의 대표적인 기술요소를 살펴보면 시그널링과 수락제어, 쉐이핑, 폴리싱, 라우팅, 스케쥴링, 버퍼관리, 트래픽 모니터링 등이 있다. 이 장에서는 위에서 나열한 여러 가지 기술요소를 포함하는 IP QoS 지원 서비스 중에서 DiffServ의 구조와 개념에 대해서 설명한다.

2.1 Differentiated Service의 구조

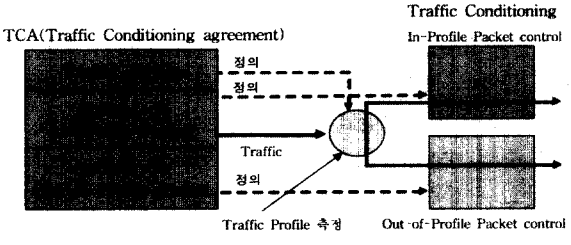
인터넷상에서 QoS를 보장하고자 하는 최초의 시도는 IETF에서 제안한 IntServ 모델이었다. IntServ는 RSVP라는 프로토콜을 사용하여 종단 호스트 어플리케이션 간에 QoS 요구신호를 주고 받는다. 이러한 신호는 종단간 패스에 존재하는 중간 라우터마다 종단간 호스트 어플리케이션 플로우의 QoS 요구사항에 대한 자원을 요청하게 된다. 그러면 중간 라우터들은 RSVP QoS 요청에 대해 할당 가능한 자원을 체크한 뒤 만약 자원의 할당이 가능하다면 트랜잭션이 지속되는 동안 자원을 예약하게 된다. 이러한 메커니즘을 통해서 IntServ는 종단간에 보장된 QoS 제공이 가능하다. 하지만 이렇게 보장된 QoS의 제공이 가능한 IntServ가 현재까지 글로벌 인터넷에서 사용되지 못하는 이유는 패킷의 전송이 끝날 때까지 종단간 패스에 존재하는 자원이나 대역폭을 계속 점유하게 됨으로써 발생할 수 있는 확장성 문제와 트랜잭션이 지속되는 동안 중간 라우터들과 예약한 QoS 요구사항을 지속하기 위해서 주고 받는 RSVP 메시지로 인해 심각한 자원의 낭비를 가져오게 되기 때문이다. 이러한 IntServ의 경험을 기초로 하여 IP QoS 제공을 위해 보다 확장가능한 서비스가 제안되었는데, 그것이 바로 패킷마킹 기술을 기반으로 하는 DiffServ이다. DiffServ는 패킷에 대한 복잡한 기능은 네트워크의 에지(edge) 라우터에서만 수행하고 네트워크의 코어에서는 패킷의 포워딩에만 중점을 두는 매우 단순한 기능만을 수행하게 한다는 것을 전제로 제안되었다. DiffServ 구조는 다음 두 가지 중요한 특징을 가지고 있다.

첫째, DiffServ는 IntServ처럼 하나의 IP 패킷 흐름, 즉 플로우 별로 서로 다른 QoS를 제공한다는 개념이 아니라, 플로우 집합을 단위로 각 집합별로 서로 다른 패킷 전달 품질을 제공한다. 둘째로 DiffServ는 종단간 신호를 사용하지 않고 PHB(Per-Hop-Behavior) 방식을 채택하였다. 따라서 DiffServ는 여러 ISP(Internet Services Provider) 망이 연결된 큰 규모의 인터넷 망에 적용할 수 있는 확장성을 갖는다. 차별화된 서비스를 제공할 수 있는 능력을 갖는 망은 여러 ISP 망으로 구성될 수 있다. ISP를 연결하는 링크상에 경계노드(Boundary node)가 존재하며, DiffServ망이 Non-DiffServ망과 연결되는 노드를 에지 노드라 부른다. (그림 1)은 DiffServ가 구현된 망의 구조를 나타낸다. DS (Differentiated Service) 도메인은 DS 노드들의 연속적인 집합으로 같은 관리를 받는 여러 네트워크로 구성되며 여러 개의 DS 도메인들이 서로 연결되어 DS region을 구성한다. (그림 1)과 같이 일반적인 DiffServ망은 DS 노드들의 집합으로 이루어진 여러 개의 DS 도메인으로 구성되며, DS 노드는 에지 노드와 내부 노드(interior node)로 구분된다. DS 에지노드는 다른 DS 도메인이나 Non-DS 도메인간의 연동을 수행할 수 있도록 되어 있다[13].



(그림 1) Differentiated Service 망 구조

DiffServ에서는 트래픽이 DS 도메인 내로 들어 올 때 이러한 트래픽이 이웃 도메인과의 계약을 준수하는지 확인하고 필요에 따라 표시 기능을 수행하며, 트래픽이 해당 DS 도메인을 벗어나는 경우 DS 도메인 내에서 정의하고 있는 PHB(Per Hop Behavior)에 따라서 패킷을 전달하도록 트래픽 제어를 수행한다. 그러므로 DS 도메인 내에서 QoS를 제어하기 위해서는 위에서 언급한 바와 같이 DS 노드의 동작을 나타내는 PHB, PHB에 해당하는 IPv4/IPv6의 각각의 코드 값, 그리고 PHB 구현을 위한 기본 메커니즘의 설정이 필요하다. 이러한 동작들이 적절히 수행되기 위해서는 DS 노드에 (그림 2)와 같은 기능들이 일반적으로 구현되어 있어야 한다. (그림 2)는 DS 도메인 내의 노드들이 QoS를 제어하기 위해서 필요한 일반적인 기능적 구조를 나타내고 있다[1, 2].

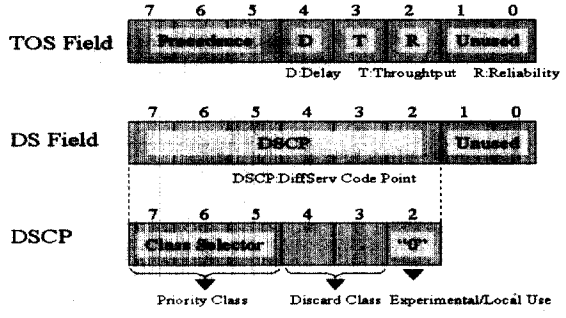


(그림2) Differentiated Service의 기능적 구조

DiffServ에서는 위의 (그림 2)에 나타나있는 것처럼 트래픽 프로파일에 따라 이를 위반한 패킷(out-of-profile)과 위반하지 않은 패킷(in-of-profile)으로 구분하며, 각각의 패킷들은 적절한 동작 모듈로 입력되어 처리 된다. DiffServ의 또 하나의 중요한 구성 요소인 폴리시 에이전트는 도메인 내의 정책을 적용하는 역할을 하는데 그러기 이전에 다른 도메인 내에 존재하는 폴리시 에이전트와 통신을 하여 상호간에 정책에 대한 협약인 SLA(Service Level Agreement)를 맺어야 한다. 또 서비스 접근 권한 확인 및 부여를 하며 트래픽 제어 기능의 구성을 결정하는 역할을 한다. SLA는 보통 서비스 가입자가 매달, 매분기, 매년 단위로 ISP와 체결하게 되며 SLA를 맺음으로써 ISP의 입장에서는 훨씬 과금을 용이하게 할 수 있는 장점이 있다. SLA 체결의 예를 들어보면 보통 DiffServ에서 가장 높은 우선순위를 가지는 EF(Expedite Forwarding) PHB의 경우는 IP 패킷 손실 0.01% 이하, 종단간 지연 4ms 이하이며, AF (Assured Forwarding) PHB는 IP 패킷 손실 0.1% 이하, 종단간 지연 40ms 이하 그리고 마지막으로 DE PHB의 경우는 일반적인 최선형 서비스이므로 IP 패킷 손실 1% 이하, 종단간 지연이 400ms 이하로 체결하게 될 것이다.

2.2 DiffServ에서 제안된 서비스

DiffServ는 사용자에게 높은 우선 순위를 갖고 ISP 사이에 패킷이 전송될 수 있도록 각각의 플로우의 흐름에 따라 패킷의 우선순위를 정의하고 있다. 이러한 서비스를 제공하기 위해서 IPv4에서는 IP 헤더의 DS 영역을 사용하여 패킷을 마킹하는 방법을 사용하고, IPv6의 경우는 헤더의 트래픽 클래스 필드를 사용한다. 현재 표준안으로 제안된 DS 바이트는 (그림 3)과 같이 6개의 비트가 패킷 전달 방식과 관련된 PHB를 결정하는 DSCP(Differentiated Service Code Point)로 할당되어 있다.



(그림 3) IPv4/IPv6 헤더의 DS 바이트

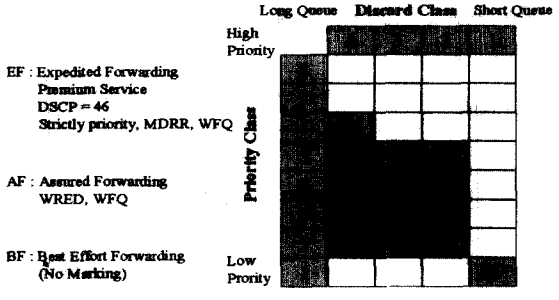
PHB는 32비트의 부호 없는 정수 값이 할당된 것으로서, DS 노드에서 특정 패킷 집합의 전송 속성에 대한 등급을 나타낸 것이다. 따라서 각 DS 노드에서는 PHB에 정의된 속성에 따라 패킷 전송 시 대역폭과 같은 자원을 다르게 할당하게 된다. 정의된 PHB들이 집합을 형성하여, 큐 관리 방식과 같은 공통적인 규약에 따라 서비스를 받을 수 있는데, 이러한 PHB들의 집합을 PHB 그룹이라고 한다. 현재 제안된 PHB 그룹으로는 EF PHB, AF PHB 그룹 등이 있다.

PHB 그룹들은 DS 도메인의 서비스 제공자에 따라 자유로이 선택되어 사용되며, 한 노드에서 하나 이상의 PHB 그룹이 사용될 수 있다. 또한 각 PHB와 DSCP간의 대응은 고정적으로 정의된 것이 아니라, 각 DS 도메인에 따라 자유롭게 정의 될 수 있다. 각 노드에서 정의된 PHB 그룹들은 노드에서 수행되는 패킷 스케줄링 방식과 버퍼 관리 방식에 따라서 구체적으로 실행된다.

DE 코드 값은 현재 인터넷에서 서비스 되고 있는 최선형 서비스를 DiffServ에서도 서비스 하기 위한 코드 값이다. EF 코드 값은 우선순위가 높은 전달방식으로서 확실한 대역폭 뿐만 아니라 낮은 지연과 손실률도 보장해야 하며, 다른 트래픽의 부하에 의해서 영향을 받지 않아야 한다. EF 트래픽에 대한 서비스는 두 부분으로 이루어져 있는데, 첫째로 각 노드의 EF PHB는 적절한 최소의 전송률을 설정해야 하며, 둘째로 DS 도메인의 경계 노드에서는 트래픽을 조절하여 내부 노드에서 패킷의 도착률이 전송률을 넘어서지 않도록 해야 한다. AF PHB 그룹은 4개의 클래스로 구성되어 있으며, 각 클래스는 다른 클래스와 독립적으로 대역폭이나 버퍼와 같은 자원이 할당된다. 또한 각 클래스별로 3등급의 패킷 폐

기 순서가 구분되어 총 12개의 PHB로 나누어져 있다[3-5].

(그림 4)는 현재 DiffServ WG에서 정의하고 있는 PHB 그룹들을 각각 큐의 길이와 큐의 우선순위에 따른 매트릭에 매핑해 놓은 그림이다. 매트릭의 우측으로 향할수록 큐의 길이는 점점 짧아지며 위쪽으로 향할수록 높은 우선순위를 가진 큐를 의미한다. 따라서 현재의 최선형 서비스를 위한 서비스 큐는 가장 우선순위가 낮으면서 큐의 길이가 짧고, EF의 경우는 가장 높은 순위를 갖으며 가장 길이가 긴 서비스 큐에 의해 서비스를 받게 된다.



(그림 4) DiffServ에서 제안된 PHB

2. 3 QoS 제공을 위한 트래픽 제어기 구조

DiffServ는 TCA(Traffic Conditioning Agreement)를 통해서 입력된 패킷들이 트래픽 프로파일을 준수하는지 여부를 체크하여 프로파일을 준수하는 패킷과 프로파일을 위반한 패킷으로 각각 분류한다. 트래픽 프로파일이란 시스템 운영자에 의해서 정의된 폴리시 정책기준으로서 DS 도메인으로 유입된 트래픽 스트림들을 분류하고 측정하는 규칙을 말한다. 유입된 패킷이 프로파일을 준수하는 패킷으로 분류된 경우에는 별다른 조작을 가하지 않거나, 또는 패킷의 DS 필드에 망 내에서 서비스 가능한 새로운 코드 값으로 마킹이 된다. 또 패킷이 프로파일을 위반한 패킷으로 분류된 경우에는 트래픽 쉐이핑, 패킷 교체 또는 드롭되거나 특별한 코드 값으로 마킹을 하게 된다. 가입자 도메인이 ISP와 SLA 협약을 거친 후에는, 가입자 도메인 내의 호스트들 간에 SLA에 명시된 서비스를 어떻게 공유할 것인가를 결정해야 하는데 이러한 과정을 서비스 할당 과정이라고 부른다[7, 8]. 서비스 할당과정은 기본적으로 각 호스트가 어떤 서비스를 사용할 것인가를 자신이 결정하는 방법과 BB(Bandwidth Broker)라는 자원제어기가 모든 호스트를 대신하여 결정하는 방법이 있다. BB는 일반 호스트가 될 수도 있으며 경계 라우터의 소프트웨어 프로세스 형태로 구성될 수도 있다. BB는 가입자 도메인의 정책에 따라서 구성이 되며 도메인 내의 모든 자원들을 관리하게 된다. DS 도메인의 에지 라우터에 새로운 패킷이 도착하게 되면 먼저 BB에 새로운 플로우에 대한 서비스 요청을 하게 되며 BB는 망 내의 정책정보와 경로정보, 각 노드의 QoS 상태 정보 등을 파악하여 QoS 요청에 대한 적절한 처리를 하게 된다. 처리된 결과에 따라 할당가능한 자원과 QoS 정책을 에지 라우터에 설정하게 되면 전송이 끝날 때까지 호스트는 계속해서 자신이 요구한 QoS 요구사항에 따른 서비스를 받게 된다. BB내에서 이러한 QoS 요청에 대한 처리는 각각의 서비스 모듈들이 담당하게 되는데 대표적인 모듈로서는 수락제어 모듈, QoS 라우팅 모듈, 정책모듈 등을 들 수 있다. 모든 호스트들은 SLA에 명시된 제한적인 자원을 공유해야 하므로 최근에는 자원할당을 위해서 BB를 사용하는 것이 거의 일반화되어 가고 있는 실정이다.

DiffServ 트래픽 제어기는 도메인 내의 폴리시 정책에 따라 정의된 TCA에서 명시한 룰에 의해 DS 도메인으로 유입되는 패킷을 분류, 측정, 리마킹 등을 수행하는 DS 구성요소이다. DiffServ의 트래픽 제어기는 크게 네 가지로 구성되어 있는데 먼저 Classifier는 망으로 유입된 플로우 집합의 패킷 헤더에서 일정 비율을 차지하는 패킷의 식별 정보를 읽어오는 역할을 한다. Classifier는 두 가지로 분류가 가능한데 BA(Behavior Aggregate) Classifier는 단지 패킷 헤더 내에서 패킷의 DSCP 코드 값을 식별하는 작업을 수행하고 MF(Multi-Field) Classifier는 각각의 패킷에 대한 전송 주소, 수신 주소, DS 필드, 프로토콜 번호, 송신측 포트 번호, 수신측 포트 번호 등의 정보를 식별하는 역할을 한다. Meter는 Classifier에 의해 선택 식별된 패킷들을 TCA의 프로파일을 기반으로 측정하는 기능을 하고, Marker는 패킷 헤더 내의 DSCP 필드 값을 특정 코드 값으로 설정하는 기능을 한다. 마지막으로 Shaper/Dropper는 트래픽 스트림을 트래픽 프로파일에 맞추기 위해 하나 또는 여러 개의 패킷들을 지연시키거나, 버퍼의 크기를 0 또는 작은 값으로 설정함으로써 패킷을 드롭시키는 기능을 각각 수행한다.

3. CBQ 기반의 패킷 스케쥴링 기법

현재 DiffServ에서 특별히 표준안으로 제안된 스케쥴링 기법은 없으며 단지 구현자들의 몫으로 남기고 있다. 따라서 DiffServ에서는 요구되는 QoS를 보장할 수 있는 어떠한 패킷 스케쥴러도 사용이 가능하다. 따라서 본 논문에서는 버퍼 제어를 위한 RED(Random Early Detection) 모델에 대한 성능 평가와 함께 CBQ 기반의 패킷 스케쥴러를 이용한 DiffServ PHB 구현부분을 모델링하고자 한다. 사용되는 패킷 스케쥴러는 기본적인 라운드 로빈 스케쥴러의 변형인 PRR(Priority Round Robin)과 WRR(Weighted Round Robin)을 이용하여 각각의 PHB 그룹에 따른 서비스 성능 측정 실험을 수행하고 결과를 비교 분석하고자 한다.

3.1 Active Queue Management 기법

DiffServ는 각각의 서비스 큐 내에서 RED와 같은 적절한 큐 관리 기법이 필요하게 된다. RED는 버퍼의 평균 큐 길이가 일정 수준(minimum threshold)을 넘어서면 버퍼가

꽉 차지 않았더라도 패킷을 미리 랜덤하게 드롭시키는 기법으로써 특정 버퍼에서 오버 플로우가 발생됨으로써 발생할 수 있는 글로벌 동기화 현상을 막을 수 있다. 이 때 패킷을 드롭시키는 확률은 평균 큐 길이에 비례하게 되고, 만약 버퍼의 평균 큐 길이가 계속 증가하여 또 다른 수준(maximum threshold)을 넘어서게 되면 드롭 확률이 1이 되면서 모든 패킷이 드롭을 경험하게 된다. RED를 기존의 나머지 버림 방식의 큐와 비교했을 때 나은 점은 버퍼 오버 플로우 발생을 통해 동시에 여러 개의 패킷 손실을 경험함으로써 동기화 현상을 일으켜 네트워크 전송 성능이 떨어지는 현상을 막을 수 있으며, 평균 큐 길이와 랜덤 드롭을 사용함으로써 버스트한 플로우들이 불공평하게 더 높은 드롭 확률을 경험하는 현상을 줄일 수 있다는 점이다. 하지만 RED의 경우 네트워크 운영자로 하여금 RED 내에서 사용되는 파라미터 값을 적절히 결정할 책임이 있는데 대부분의 운영자들이 어떠한 값들이 적절한 파라미터 인지 알 수 없다는 문제점이 있다. 또 RED는 TCP(Transmission Control Protocol) 이외의 다른 특성을 가지는 트래픽 소스들이 부당하게 다른 TCP 플로우에게 피해를 줄 수 있는 문제점이 있다. 최근 이런 문제를 해결하기 위해서 마이크로-플로우 각각의 행동을 관찰하여 TCP보다 더 지나치게 공격적인 플로우들을 억제하고 방어할 수 있는 방법이 몇 가지 제안되었으나 각 플로우에 대한 오버헤드가 너무 커서 확장성이 좋지 않은 이유로 많은 호응을 얻지 못하고 있다 [9, 10].

(그림 5)는 RED상에서 min_{th}(minimum threshold)와 max_{th}(maximum threshold)값에 따라 적용 가능한 세가지 모델을 보여주고 있다. (그림 5) (a)의 경우는 DE와 AF의 min_{th}, max_{th} 값이 각각 똑같은 경우이다. 이 모델의 경우 DE와 AF가 거의 동시에 드롭이 발생하도록 설계되었다. (그림 5) (b)의 경우는 AF의 min_{th}, max_{th} 가 DE의 min_{th}, max_{th} 보다 좀 더 크게 겹쳐지도록 적용하는 방법이다. 이 모델의 경우는 점차로 트래픽이 몰려서 큐가 혼잡해 질 무렵 DE가 먼저 드롭을 시키게 되고 DE가 모든 패킷을 드롭시키는 상황에도 AF는 다소간의 여유가 있도록 설계되었다. 또한, (그림 5) (c)의 경우처럼 DE와 AF의 min_{th}, max_{th} 가 전혀 겹쳐지지 않게 설계할 수도 있다. 이 모델의 경우는 DE 트래픽이 100% 드롭되는 상황이 된 후에 AF 트래픽에 대한 드롭을 시작하는 모델이다. 따라서 DE 패킷이 모두 드롭되더라도 AF 패킷은 계속적인 서비스를 받을 수 있다. 이러한 세가지 모델에 대한 시뮬레이션 결과 및 분석내용은 4장에서 기술한다.

다음은 RED Gateway 알고리즘이다.

```
Initialization
avg ← 0
count ← 1
for each packet arrival
    calculate new avg. queue is avg :
    if the queue is nonempty
    avg ← (1-w_q) avg + W_q q
    else
    m ← f(time q_time)
avg ← (1-w_q)^m avg
  if min_th ≤ avg < max_th
      increment count
      calculate probability p_a
          p_b : max_p (avg ← (1-W_q)^m avg)
          p_a : p_b / (1 - count * p_b)
      with probability p_a :
          mark the arriving packet
          count ← 0
else if max_th ≤ avg
      mark the arriving packet
      count ← 0
else count ← 1
with queue becomes empty
q_time ← time
```

위 알고리즘에 사용되는 파라미터는 다음과 같다.

Saved Varibles :

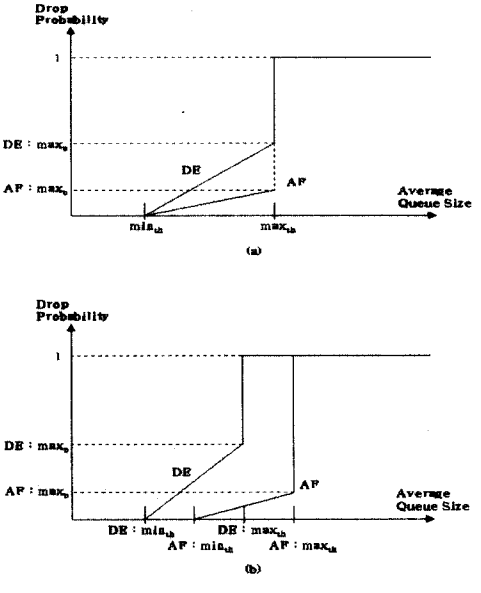
- avg : average queue size
- q_time : start of the queue idle time
- $count$: packets since last marked packet

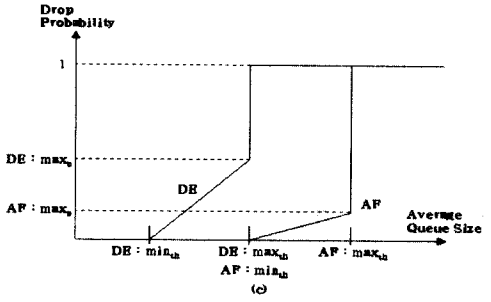
Fixed parameter :

- W_q : queue weight
- min_{th} : minimum threshold for queue
- max_{th} : maximum threshold for queue
- max_p : maximum value for p_b

Other parameter :

- Pa : current pkt-marking probability
- Q : current queue size
- Time : current time
- F (t) : a linear function of the time t





(그림 5) 세가지 다른 RED 모델

네트워크 운영자들은 (그림 5)에서 설명한 각각의 파라미터들을 적절하게 설정하여 네트워크의 성능 향상 및 체증상황을 예방할 수 있어야 한다. 하지만 대부분의 네트워크 운영자들은 어떠한 값이 적절한 파라미터인지를 알기가 매우 어렵다. 최근에 Adaptive RED라는 한가지 해결책이 제시되었는데, 이 기법은 평균 큐 길이가 min_{th} 이하나 max_{th} 이상으로 반복적으로 넘어갈 경우 최대 드롭 확률을 변화시켜 주어 평균 큐 길이를 min_{th}와 max_{th}사이에 항시 가두어 두는 기법이다. 그러나 아직도 임계값들을 정하는 데는 특별한 가이드 라인이 없는 실정이다. RED의 또 다른 단점은 TCP 이외의 특성을 가지는 트래픽 소스들이 부당하게 TCP 플로우에 피해줄 수 있으며 알고리즘 자체가 너무 복잡해서 분석이 어렵다는 점이다. 이러한 RED의 단점을 보완하기 위해 제안된 기법으로 RIO(RED with In and Out packets)기법을 들 수 있다. RIO는 각각의 드롭 우선순위에 대해 RED 알고리즘을 따로 수행하는 방법이다. RIO는 각 입력 패킷들을 IN과 OUT의 두 가지 레벨로 구별해서 표시를 해 두고 IN 패킷에 대해서는 OUT 패킷 보다 큰 임계값과 작은 최대 드롭 확률을 적용한다. 즉 폐기 비율을 다르게 하여 서비스를 차별화 시키는 방법이다. 그래서 체증이 일어났을 때 IN 패킷보다는 OUT 패킷을 먼저 그리고 좀 더 적극적으로 드롭하기 시작한다. 따라서 IN 패킷들의 드롭이 발생한 경우는 OUT 패킷들은 절대로 받아들여지지 않는다. RIO 서비스 차별화 성능역시 파라미터의 선택에 따라서 좌우된다. IN 패킷들은 IN 패킷들의 흐름만을 측정하는 avg_in으로 폐기율이 결정되지만 OUT 패킷들은 IN 패킷과 OUT 패킷들의 두 가지 흐름을 합한 흐름을 측정한 avg_total을 통해서 폐기율이 결정된다. 이렇게 차별화를 둠으로써 IN 패킷들이 상대적으로 높은 서비스 품질을 보장 받을 수 있다. 하지만 RIO 역시 이론적인 임계값의 가이드 라인은 없는 실정이다. 그래서 ISP의 재량에 따라 적절하게 설정을 하도록 하고 있다.

3.2 패킷 스케쥴링 기법

패킷 스케쥴링이란 패킷 교환기에 도착하여 전송을 기다리는 패킷들의 서비스 순서를 결정하는 방법이다. 가장 일반적으로 사용되는 방법으로는 FCFS(First Come First Service) 방법을 들 수 있는데 이 방식은 패킷이 도착한 순서대로 공통 버퍼에 저장하였다가 버퍼에 쌓인 순서대로 패킷을 아웃 포트로 전송하는 방법이다. 이 방법은 가장 단순한 방법으로써 현재 인터넷상에 존재하는 대부분의 라우터들이 FCFS 스케쥴링 방법을 사용하여 패킷들에 대한 서비스를 하고 있다. 그러나, FCFS 방법은 각 사용자 플로우의 독립성을 유지하지 못하고 공평성을 보장하지 못한다. 왜냐하면 FCFS 방법은 패킷을 무조건 많이 보내는 플로우에게 더 많은 망 자원을 할당하는 단점을 가지고 있기 때문이다. 따라서 특정한 플로우가 너무 많은 패킷들을 보낼 경우 결과적으로 다른 플로우들은 피해를 받게 된다. 따라서 DiffServ처럼 사용자의 플로우에 대해 차별적인 서비스를 가능하도록 하기 위해서는 기존의 FCFS 방식과 달리 최소한의 공평성과 성능 보장성 및 QoS를 지원해 줄 수 있는 패킷 스케쥴링 기법을 사용하여야 한다. 공평성은 각 플로우에게 자원을 공평하게 배분하는 것을 의미하고 성능 보장성이라 함은 성능보장을 위해서 각 플로우에 대해 최소한의 전송용량을 보장해 주는 것을 의미한다[12, 14].

본 논문에서는 DiffServ상에서 사용할 수 있는 패킷 스케쥴링 기법 중에서 공평성과 QoS를 지원할 수 있는 방법으로 기본적인 라운드로빈 기법과 라운드로빈 방식의 변형된 기법을 사용하고자 한다. 기본적인 라운드 로빈(Round Robin) 방식은 전송될 순서를 기다리는 패킷들이 들어있는 버퍼들을 차례대로 서비스 하는 일종의 Fair Queuing 방법이다. 만약 어떤 버퍼에 패킷이 없을 때는 다음 버퍼가 검사되고 패킷이 존재하는 경우에는 스케쥴링 라운드동안 스케쥴링 된다. 하지만 단순한 라운드 로빈 방식의 경우는 각 서비스 클래스마다 스케쥴링되는 동안 서비스되는 패킷의 크기가 모두 동일하므로 DiffServ의 경우처럼 플로우의 집합별로 서비스 차별화를 두어야 하는 서비스 구조에는 적합하지 못하다. 따라서 각 서비스 클래스마다 차별화를 주기 위해서는 기존의 일반적인 라운드 로빈 방식의 스케쥴링 기법이 아니라 PRR이나 WRR 같은 라운드 로빈 방식의 변형된 방식을 사용해야만 한다.

기본적인 라운드 로빈 방식의 변형 방식으로는 대표적으로 PRR 방식을 들 수 있다. PRR의 기본적인 메커니즘은 라운드 로빈과 같지만 각 서비스 클래스가 우선순위를 가지고 있어 우선순위에 의해 차별적인 서비스를 한다. PRR을 DiffServ에 적용한다면 가장 높은 우선순위 클래스인 EF를 서비스하고 다음 우선순위 클래스인 AF, DE 순으로 서비스를 하게 된다. 따라서 어느 정도 DiffServ에서 요구하는 서비스 차별화가 이루어지게 된다. 하지만 PRR의 경우는 우선순위가 높은 서비스 클래스에 너무 많은 트래픽들이 존재하는 경우 상대적으로 우선순위가 낮은 서비스 클래스의 트래픽들은 연속적인 드롭이 발행하게 됨으로써 계속해서 서비스를 받지 못하는 기아상태를 경험하게 된다는 단점이 있다. 따라서 PRR도 특정한 서비스 클래스가 기아 상태가 되지 않도록 요구하는 DiffServ의 구조에 역시

맞지 않게 된다. DiffServ는 이렇게 우선 순위가 높은 특정한 서비스 클래스가 너무 많은 대역폭을 독점하지 않도록 시스템관리자에 의해 결정된 SLA에 의해 관리된다.

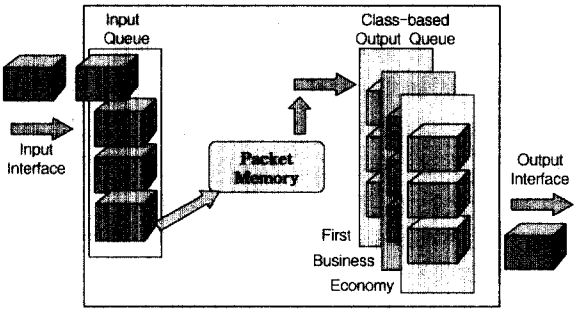
기본적인 라운드로빈 변형 방식 중에서 차별화된 서비스가 가능하면서도 기아현상이 발생되지 않는 기법으로는 WRR 방식이 있다. WRR 스케쥴링 알고리즘은 일정주기 동안 각 입력에 대하여 고정된 가중치를 두어 차별적으로 서비스를 제공하는 방식이다. WRR은 각 서비스 클래스의 가중치가 다를 경우에도 적용할 수 있도록 기본적인 라운드 로빈 방식을 개선한 방식으로서 현재 IntServ 등에서 서비스 클래스별로 차별화 된 서비스를 제공하기 위해 많이 이용되고 있다. 하지만 일초에도 수백만개의 패킷들이 지나가는 대형 네트워크의 백본 라우터에서 계속해서 입력되는 패킷들의 사이즈를 구해서 적절한 가중치를 결정한다는 것은 엄청난 오버헤드가 아닐 수 없다. 따라서 패킷의 길이를 모르는 상황에서도 적절한 서비스가 가능한 스케쥴링 기법을 이용해야 한다.

DRR(Deficit Round Robin)은 패킷의 길이를 모르는 상황에서 사용할 수 있도록 제안된 스케쥴링 방식이다. WRR 방식의 경우는 매 라운드마다 플로우가 전송할 패킷을 보유하고 있는 경우, 각 플로우마다 최소 하나 이상의 패킷을 서비스한다. 그러나 DRR방식은 무조건 모든 플로우를 서비스하는 것이 아니라, 일정한 조건을 만족시키는 플로우만을 선택하여 서비스를 한다. DRR은 매 라운드마다 퀀텀(qunantum)이라는 서비스 단위만큼 각 플로우 적자누적기(deficit counter)를 증가시킨다. 퀀텀은 바이트(또는 비트) 단위이며 만약 어느 플로우의 적자누적기가 현재 대기하고 있는 패킷 길이보다 크다면 패킷을 서비스하고 서비스한 패킷 길이만큼 적자누적기를 감소시킨다. 물론 각 플로우마다 증가하는 퀀텀은 플로우의 서비스 가중치에 따라 표준화된다.

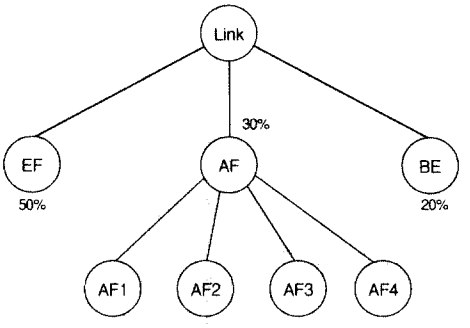
3.3 차별화 서비스를 위한 Class Base Queuing 모델

CBQ는 우선순위 큐잉의 변형으로 서비스 클래스에 따라 서비스 큐를 정의하고 있으며, 특정 서비스 클래스가 낮은 우선 순위로 인해 자원을 할당 받지 못하여 생기는 기아현상을 막기 위하여 고안되었다. 또한 관리자는 각 큐가 서비스되는 성향이나 큐잉되는 트래픽의 양 등을 결정할 수 있다. CBQ는 트래픽의 형태에 따라 우선 순위에 기초하여 큐잉 서비스를 수행하고 특정 서비스 클래스의 트래픽이 시스템 자원과 대역폭을 독점하는 것을 막음으로써 공평성을 제공하게 된다.

(그림 6)은 high, medium, low 3개의 우선 순위를 갖는 CBQ의 예이다. CBQ는 각 큐의 우선 순위에 따라 각기 다른 양의 서비스 한계값을 두고 이 값에 따라 스케쥴링 라운드마다 각 큐를 서비스함으로써 특정 서비스 클래스가 영구히 서비스 받지 못하는 기아현상을 막을 수 있으며, 각 패킷들은 상당히 적은 양의 지연값으로 서비스 되어 진다.



(그림 6) Class Base Queuing 모델

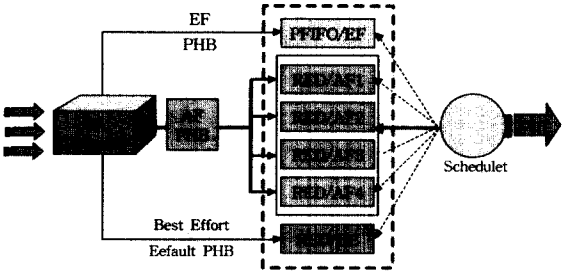


(그림 7) 서비스 큐의 우선순위에 따른 대역폭 할당 예

일반적으로, CBQ는 각 서비스 클래스에 일정한 양의 대역폭을 할당하는 방법으로 인식되어 지는데 CBQ가 높은 우선순위의 트래픽에 더 많은 양의 자원을 할당하고 낮은 우선순위의 트래픽에는 조금 적은 양의 자원을 할당함으로써, 우선순위 큐잉에서 높은 우선 순위를 갖는 트래픽을 무조건 먼저 서비스해 줌으로써 낮은 우선 순위를 갖는 트래픽이 자원을 할당 받지 못하여 발생하는 기아현상 문제를 해결할 수 있었다. 즉 우선순위 큐잉과 비교해 낮은 우선 순위를 갖는 트래픽에 대해 자원을 무조건 빼앗는 것이 아니라 약간의 일정한 자원을 할당하여 서비스를 수행하게 된다. 또한 CBQ는 트래픽을 여러 가지 서비스 클래스로 분류하는 기본적인 방법으로 간주되어지며 각 서비스 클래스에 대해 링크 공유를 제공하고 큐 자원을 관리하는 방법을 제공한다. CBQ에서 패킷 스케줄러에 의해 사용되어지는 스케쥴링 알고리즘은 링크-공유 스케줄러(link-sharing scheduler)와 GPS(generalized packet scheduler)이다. CBQ에서 사용되는 링크-공유 스케줄러는 탑-레벨 링크-공유 스케줄러로 망이 과부하 되었을 때 동작하여 우선 순위가 낮은 클래스의 링크 사용량을 한정된 값으로 제한하고 우선 순위가 높은 클래스가 할당된 대역폭보다 더 많이 사용할 수 있도록 대역폭의 사용량을 조절하는 기능을 수행한다. 대역폭의 사용량을 조절하는 방법은 오버플로우 된 큐의 데이터를 버리거나 데이터의 전송을 지연시킨다. GPS는 PRR 패킷 스케줄러와 WRR 패킷 스케줄러로 구현되어 우선 순위에 기초하여 스케쥴링을 수행한다.

(그림 8)은 본 논문에서 제안하고 있는 DiffServ상에서 CBQ를 이용해서 PHB 그룹들을 스케쥴링하고 포워딩하는

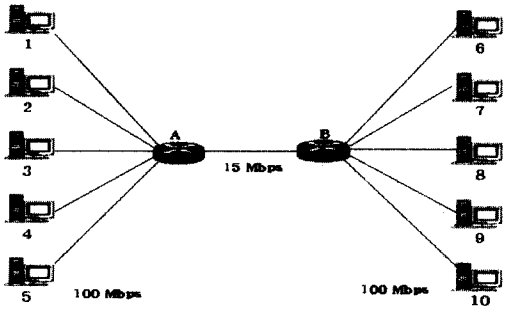
모델이다. 패킷이 수신되면 먼저 BA Classifier는 수신된 패킷의 헤더정보를 식별하고 패킷 스케줄러에 의해 이 패킷이 서비스 될 클래스가 결정된다. 이때 패킷 분류자는 큐 정보와 기타 클래스 관련 정보를 가지고 있는 클래스 구조체에 포인터를 넘기고, 패킷 스케줄러가 해당 큐가 오버 플로우 되었는지를 체크하여 넘치지 않았다면 각각의 PHB 그룹에 맞는 서비스 클래스 큐에 각각 입력될 것이다. 각각의 서비스 클래스 큐는 개별적인 버퍼 제어 기법을 이용 플로우들을 제어하게 된다. 출력 드라이버는 스케줄러와 비동기적으로 동작하므로 링크 레이어로 전송할 데이터가 있을 때 구동 되어 패킷 스케줄러를 호출하고 패킷 스케줄러는 각 큐의 사용량과 우선 순위에 따라 다음 패킷을 전송할 큐를 결정하여 출력 드라이버로 전송하게 된다.



(그림 8) CBQ를 이용한 패킷 전송

4. 시뮬레이션 및 성능 분석

본 논문에서 제안한 모델에 대해서 성능분석을 위해 사용한 시뮬레이터는 UCB/LBNL Network Simulator Version 2로서 Adel Linux 6.2 서버 상에 설치하여 실험하였다. 본 시뮬레이션에 사용한 실험 모델은 (그림 9)를 기본 실험 모델로 한다. (그림 9)에서 보는 것처럼 양쪽에 두 대의 라우터를 두고 각각의 라우터에는 5대의 호스트가 연결되어 있다. 라우터 A와 B는 15 Mbps 속도의 링크로 연결되어 있고 각각의 노드와 게이트웨이는 100 Mbps를 사용하는 것으로 가정하였다.



(그림 9) 성능분석을 위한 네트워크 모델

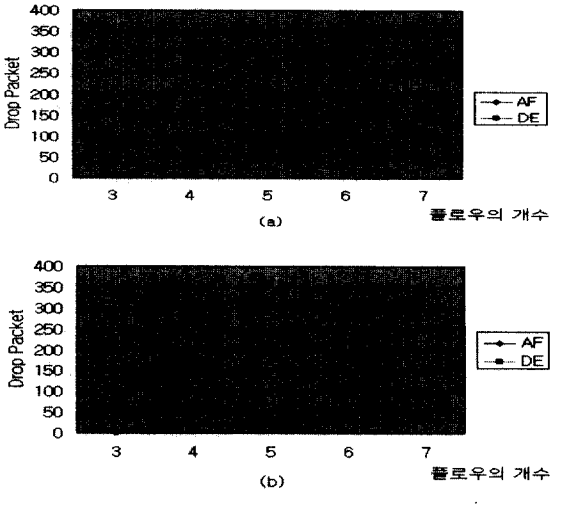
두 대의 라우터를 사이에 두고 좌측에 존재하는 호스트들은 단순한 송신만을 담당하는 송신 호스트이고 우측에 존재하는 5개의 호스트는 수신을 담당하는 수신 호스트의 역할을 수행하도록 하였다. 시뮬레이션 시나리오는 송신 호스트로부터 어떠한 패킷들이 발생이 되면 라우터 A에서 DSCP 코드 값이 마킹되고 CBQ내에서 각각의 서비스 클래스에 따라 서비스 큐에 담기게 되고, 스케쥴링 알고리즘에 의해 스케쥴링 된다. 패킷이 라우터 B에 도착하면 패킷 헤더의 DSCP 값을 참조하여 각 서비스 큐에 입력되고 여기서 목적지 호스트로 전송을 하게 된다.

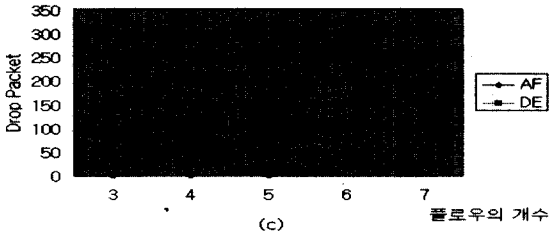
첫 번째로 진행한 실험은 일반적인 DiffServ 망에서 각각의 PHB 그룹에 대한 RED 파라미터 값의 조정만으로 어떠한 효과가 나타나는 가를 실험하기 위하여 (그림 5)에서설명한 세 가지의 다른 RED 모델에 대한 전송 실험을 하였다. 본 시뮬레이션에서는 RED 모델을 사용하는 AF와 DE PHB만을 마킹하여 각 PHB에 따른 드롭율을 비교하였다. 시뮬레이션에 사용한 RED 파라미터 값은 다음 <표 1>과 같다.

〈표 1〉 RED 모델에 따른 min_{th}, max_{th} 값 할당

RED Model	AF min_{th}	AF max_{th}	DE min_{th}	DE max_{th}
(a)	20	60	20	60
(b)	40	80	20	60
(c)	60	100	20	60

위와 같은 파라미터를 사용하여 시뮬레이션 한 결과는 (그림 10)와 같다. X축은 클래스 당 플로우의 개수이고 Y축은 드롭되는 패킷의 개수를 의미한다. 시뮬레이션의 결과를 보면 DE PHB의 경우는 실험 모델 (a)에 비해 실험 모델 (b)에서는 약 2/3정도, 그리고 모델 (c)에서는 약 1/2정도로 드롭 되는 패킷의 개수가 줄어드는 것을 볼 수 있었다. AF PHB의 경우는 모델 (a)의 경우 DE PHB의 거의 절반에 가까운 패킷 드롭율을 보였지만 모델 (c)의 경우는 거의 대부분의 패킷들이 드롭 되지 않고 전송되는 것을 볼 수 있다. 따라서 본 실험의 결과 RED 모델의 경우 DE와 AF PHB의 min_{th}와 max_{th}가 겹쳐지지 않은 형태로 파라미터를 설정하는 것이 좋은 성능을 얻을 수 있음을 알 수 있었다.



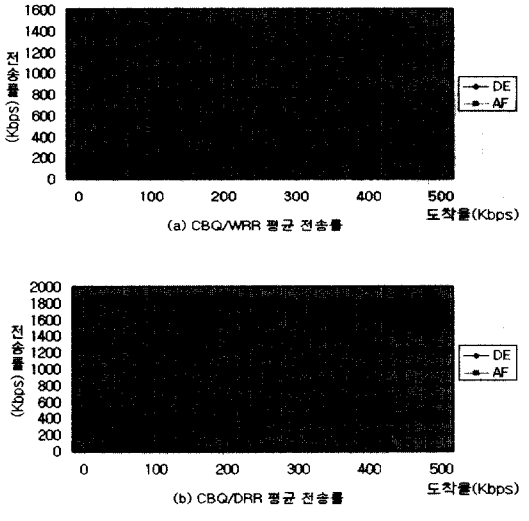


(그림 10) RED Model 시뮬레이션 결과

두 번째 실험은 본 논문에서 제안한 CBQ를 이용한 패킷 스케줄링 기법의 성능 분석을 위해 CBQ/PRR을 사용했을 때와 CBQ/WRR을 사용했을 때 어떠한 성능의 차이를 보이는지에 대해 시뮬레이션을 실시하였다. 그런데 실제로 측정해 본 결과 일부 실시간 데이터를 전송한 경우는 WRR 스케줄링 기법이 약간 우세한 결과가 나온반면 일반적인 FTP(File Transport Protocol) 데이터의 경우는 두 가지 패킷 스케줄링 기법이 거의 비슷한 결과를 나타내었다.

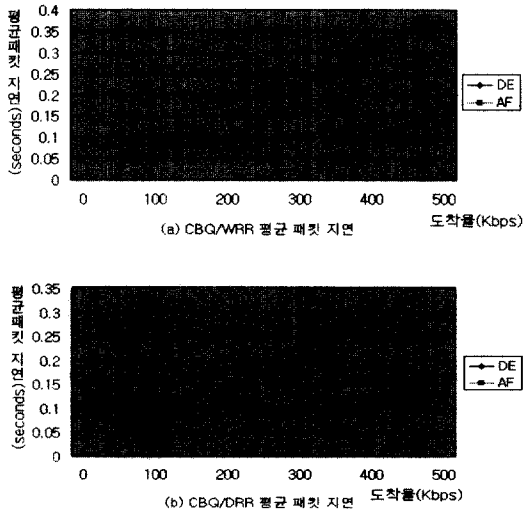
마지막으로 진행된 실험은 CBQ/WRR과 CBQ/DRR에 대한 성능 비교 실험을 하였다. 이 실험에서도 실시간 데이터 트래픽과 FTP 트래픽을 소스로 첨부해서 시뮬레이션 하였고 물론 RED 파라미터는 위 실험의 결과를 토대로 (그림 5) (c) 모델을 사용하였다. 이 실험에서도 역시 AF와 DE PHB 만을 고려하여 시뮬레이션을 하였다. 실험은 패킷 전송률, 평균 패킷 딜레이 및 평균 패킷 드롭율 이렇게 세 종류의 결과를 비교 분석 하였다.

먼저 패킷의 평균 전송률의 경우 WRR 기법을 사용한 경우는 DE PHB가 증가함에 따라서 AF PHB의 전송률이 감소하는 경향이 훨씬 심하게 나타났다. 하지만 DRR의 경우는 패킷의 평균 도착률이 500K이상이 되더라도 거의 1700Kbps 정도의 안정된 전송률을 보이는 것을 볼 수 있었다.



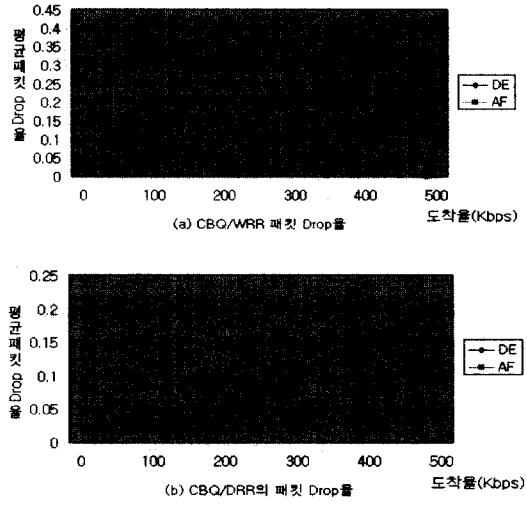
(그림 11) WRR과 DRR의 평균 전송률

두 번째로 평균 패킷 지연의 경우는 AF PHB는 두 가지 트래픽의 경우 모두 0.15초 미만의 지연을 보여서 안정적인 반면, DE PHB의 경우는 WRR을 사용한 경우에 패킷의 도착율이 증가함에 따라 DRR을 사용하는 경우보다 급격하게 지연이 증가되는 것을 볼 수 있었다.



(그림 12) WRR과 DRR의 평균 패킷 지연

마지막으로 패킷 드롭율의 경우는 WRR을 사용한 경우 DE PHB가 0.45, AF PHB에서 0.15 이하의 패킷 드롭율이 발생한 반면 DRR의 경우는 DE PHB가 0.2 이하가 드롭되고 AF PHB의 경우는 0.05 미만으로 거의 드롭이 발생되지 않고 안정적으로 패킷이 전송되는 것을 볼 수 있었다.



(그림 13) WRR과 DRR의 평균 드롭율

5 결 론

현재의 인터넷 환경은 기하 급수적으로 양적, 질적인 팽창을 거듭하고 있다. 따라서 일초에도 수백만 개의 패킷들을 처리해야 하는 백본 라우터의 부담은 갈수록 커져만 가

고 있다. 그래서 이러한 백본 라우터에서 수많은 패킷들을 아주 빠르게 처리하기 위해서는 매우 단순한 방법으로 스케쥴링을 해야 한다.

본 논문에서는 DiffServ상에서 효율적인 패킷 전송을 위해 CBQ를 사용한 패킷 전송 모델을 제안하였다. 제안된 모델은 기존의 토큰-버킷 방식과 달리 각각의 트래픽의 우선순위에 따라 각각의 서비스 우선순위가 다른 여러 개의 큐를 사용하여 각각의 트래픽을 차별적으로 서비스하는 방법이다. 본 논문에서 사용한 스케쥴러는 기본적인 라운드로빈을 변형한 WRR과 DRR을 이용하여 스케쥴링을 한다. 제안된 모델에 대해 시뮬레이션 한 결과RED 파라미터는 mix_{th}와 max_{th}가 겹쳐지지 않은 모델이 가장 좋은 성능을 나타냈고 패킷 스케쥴링 기법은 DRR을 사용한 경우에 가장 전송률이 좋고 지연이나 드롭이 적게 나타나는 결과가 나왔다.

앞으로의 연구는 실제로 DiffServ를 이용하여 DiffServ 네트워크를 구현했을 때 사용자들이 요구하는 수준의 QoS를 보장할 수 있을 만큼 단순하면서도 성능이 보장되는 버퍼 제어기법이나 패킷 스케쥴링 기법에 대한 계속적인 연구가 필요하다. 또한, 최근에 많은 주목을 받고 있는IntServ와 DiffServ의 연동망 즉 네트워크 자원의 예약이 가능한 망의 내부에서는 RSVP를 사용하고 백본망에서는 DiffServ를 사용하는 구조에서 무난히 사용할 수 있는 버퍼 제어기법과 패킷 스케쥴링 기법에 대한 연구도 필요하다.

참 고 문 헌

[1] K. Nichols, S. Blake, "Definition of the Differentiated Services Field (DS Byte) in the Ipv4 and Ipv6 Headers," RFC 2474, December, 1998.

[2] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December, 1998.

[3] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assure Forwarding PHB Group," RFC 2597, June, 1999.

[4] V. Jacoban, K. Nichols, K. Poduri, "An Expedited Forwarding PHB," RFC 2598, June, 1999.

[5] S. Brim, B. Carpenter, F. Le Faucheur, "Per Hop Behavior Identification Codes," RFC 2836, May, 2000.

[6] D. Black, EMC Corporation, "Differentiated Services and Tunnels," RFC 2983, October, 2000.

[7] A. Chapman. and Kung H. T, "Automatic Quality of Service in IP Network," Proceeding of the Canadian Conference on Broadband Research, Ottawa, pp.184-189, April, 1997.

[8] B. Braden, D. Clark, and S. Shenker, "A Connectionless Approach to Providing QoS in IP Networks," IFIP Conference on High Performance Networking (HPN 98), Vienna, September, 1998.

[9] D. Clark, and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," IEEE/ACM Transaction on Networking, August, 1998.

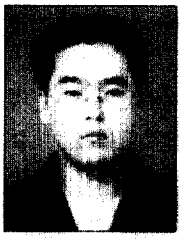
[10] G. Mamais, et. Al, "Efficient Buffer Management and Scheduling in a Combined IntServ and DiffServ Architecture : A Performance Study," ELISA, ACTS AC310, 1999.

[11] N. Seddigh, B. Nandy, P. Pieda, J. Hadi Salim, A. Chapman, "An Experimental Study of Assured Service in a Diffserv IP QoS Network," SPIE Proceedings Vol.3529, October, 1998.

[12] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transaction on Networking, Vol.1 August, 1993.

[13] S. Floyd, and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," IEEE/ACM Transaction on Networking, Vol.3, August, 1995.

[14] 김종권, "QoS 지원을 위한 패킷 스케쥴링 알고리즘", TELECOMMUNICATIONS REVIEW, 제10권 제3호, pp.5-6, 2000.



문 준 현

e-mail : impbravo.kwangju.ac.kr

1999년 2월 광주대학교 컴퓨터학과 졸업 (공학사)

2001년 2월 광주대학교 대학원 컴퓨터학과 졸업 (공학석사)

2001년 3월~현재 광주대학교 인터넷방송국 연구원

관심분야 : 인터넷 QoS, 실시간 프로토콜(RTP/RTCP), 차세대 인터넷(IPv6)

김 광 현

e-mail : ghkim@hosim.kwangju.ac.kr

1989년 2월 광운대학교 컴퓨터학과 졸업 (공학사)

1991년 2월 광운대학교 대학원 컴퓨터학과 졸업(공학석사)

1997년 2월 광운대학교 대학원 컴퓨터학과 졸업(공학박사)

1997년 3월~현재 광주대학교 컴퓨터전자통신공학부 조교수

2001년 8월~현재 Pennsylvania State University Post-doc

관심분야 : 인터넷 QoS, 차세대인터넷(IPv6), 멀티캐스트 프로토콜