DiffServ의 효율적인 자원활용을 위한 트래픽 컨테이너 설계

장 경 성† · 강 대 욱††

요 약

Diff-Serv는 개별적으로 구분된 다른 트레픽에 대해서 다른 종류의 네트워크 서비스를 제공하기위한 메카니즘이며 이용자들에게 양질의 서비스를 제공하기 위한 방법을 제시하고 있다. DS는 토큰 버켓 방법을 사용하여 고정된 호스트들에게 제공되기 때문에 이동성을 가지는 서비스나 개별적인 트레픽에 대한 차등적인 서비스를 제공할 방안이 존재하지 않는다. 본 논문에서는 트래픽컨테이너(TC)를 위해서 WFQ 방법과 변방 노드들과 경계 라우터들 내부에서 제어가 가능한 AggF(Aggregate Flow)를 모니터링하여 스케쥴링하는 방법을 제시한다. 이 방법은 트래픽 비율(traffic rate)을 유동적으로 제어가 가능하며 대역폭의 효율적인 이용성을 제공하고 있다.

The Design for Traffic Container to use resources efficiently in DiffServ

Kyung-Sung Jang† · Dae-Wook Kang††

ABSTRACT

Diff-Serv is a mechanism by which network service providers can offer differing levels of network service to different traffic, in so providing quality of service (QoS) to their customers. Because this mechanism has been deployed just for fixed hosts with the Token Bucket mechanism, DiffServ have been suggested can not satisfy the mobility service or the differential service for individual traffics. In this paper, we suggest WFQ mechanism for traffic conditioner and scheduling method for monitoring the AggF(Aggregate Flow) which will be controlled in edge nodes and border routers. So it will control traffic rate dynamically and suggest efficient usability of bandwidth.

키워드 : IP, 라우팅(Routing protocol), DiffServ, 네트워크 성능평가(Network QoS)

1. 서 론

최근 폭발적으로 사용자가 늘고 있는 인터넷은 멀티미디어 트래픽을 전송하는데 있어서 기존 IP 프로토콜의 best-effort형 서비스에 의존하고 있으나 이러한 서비스만으로 인터넷 사용자에게 음성 및 화상 회의와 같은 실시간 서비스를 제공하는 것은 사용자의 요구를 충족시키는데 많은 한계를 갖는다. DiffServ는 네트웍 서비스 제공자가 그들 고객에게 자신들의 네트워크 내에서 받아들여 질 수 있는 레벨의 서비스를 제공하는 메커니즘이며, 복잡한 자원 할당 절차가 불필요하며 우선 순위 지정 IP 헤드의 일부를 그대로 사용, 각 호스트 및 라우터의 트래픽 제어 모듈이 쉽게 구성 가능, PHB 사용으로 망 내부는 각 응용프로그램 또는 세션(session)당 플로우(flow)나 전송상태 등을 유지할 필요가 없다는 것 등이 있다.

본 논문에서는 차등서비스를 제공하기 위한 TC를 설계하는데 동일 서비스레벨을 제공받고자 하는 다수의 플로우들이 공정하게 서비스를 제공받을 수 있고, Dropper 컴포넌트에서의 지능적인 동작으로 잉여 자원을 효율적으로 사용할 수 있는 방안을 제시하고 있다. 이들 공정성과 잉여 자원의 효율성은 단지 시간적인 특성에 기반하여 TrafficStream을 측정하는 기존모델들과는 달리 새로이 제안하는 Monitor 컴포넌트를 추가하여 서비스를 제공할 수 있다. Monitor 컴포넌트에서는 사전에 Meter나 Shaper 컴포넌트에서 처리가능 하도록 모니터링을 통해 결과값을 생성한다. 다음단계에서 처리될 상황을 모니터링한 결과는 Meter나 Dropper 컴포넌트에서 유기적인 동작으로 자원을 공정하게 사용하도록 하며, 잉여자원을 효율적으로 사용할 수 있도록 한다. Meter 컴포넌트에서의 동작은 모니터 결과에 의해 공정한 자원사용의 범위를 초과하는 플로우를 시간적인 속성을 측정하기 전에 미리 Shaper 컴포넌트로 전송함으로서 공정성을 취하

† 정 회 원 : 초당대학교 정보통신공학과 교수
†† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
논문접수 : 2003년 1월 10일, 심사완료 : 2003년 12월 29일

는 방법을 사용하며, Shaper 컴포넌트에서는 재 측정을 위해 다시 Meter 컴포넌트로 보낼 플로우와 Dropper 컴포넌트로 바로 보낼 플로우를 식별하고, Dropper 컴포넌트에서는 잉여 자원이 존재할 경우 미사용 자원 양 만큼을 Marker 컴포넌트로 전송하는 방법을 사용하고 있다.

2. Differentiated Service

2.1 Differentiated Services 개요

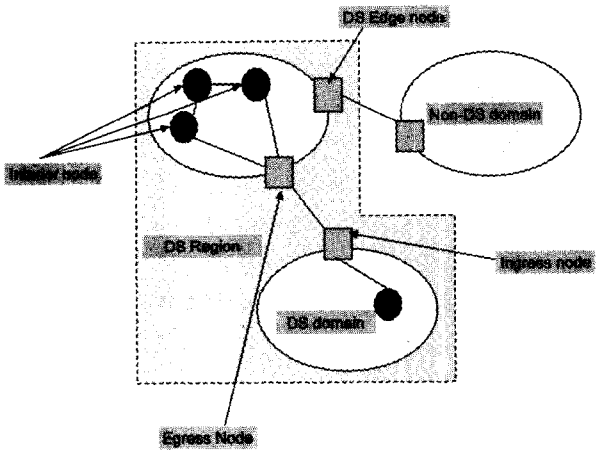
Differentiated Services 구조는 네트워크로 들어오는 트래픽을 등급화하고 네트워크의 경계에서 검사하여 다른 우선순위로 패킷을 처리하는 간단한 모델을 기반으로 한다. 네트워크의 중심(core)에서 패킷은 DSCP에 연관된 PHB에 따라서 포워딩된다. DS 영역내의 중요구성 요소들 다음과 같다.

2.1.1 Differentiated Services 도메인

Differentiated Service가 제공되는 영역으로 보통 같은 관리를 받는 하나 이상의 네트워크로 구성된다. 이 영역은 DS 경계(boundary) 노드들(nodes)과 내부(interior) 노드들로 구성 될 수 있으며 경계노드들은 자신의 DS 영역과 다른 DS 또는 DS를 제공하지 않는 영역과 연결될 수 있다. 반면에 DS 내부노드들은 단지 다른 DS 내부나 자신의 경계노드들과 연결된다. 내부노드들은 DS codepoint 재표시(re-marking)과 같은 제한된 트래픽 조종(conditioning)를 수행하며 보다 복잡한 등급기(classificaiton)와 트래픽 조정을 구현한 것은 DS 경계노드들과 유사하다.

2.1.2 Differentiated Services 영역

DS 영역은 하나 이상의 DS 도메인으로 구성되고 그 영역내의 도메인들에 연결된 경로들을 따라서 DS를 지원할 수 있다. 다음 (그림 1)은 DS 영역과 DS 도메인, 내부노드(interior node), egress node를 나타내고 있다.



(그림 1) DS 도메인 & 영역

2.1.3 트래픽 등급화와 조종

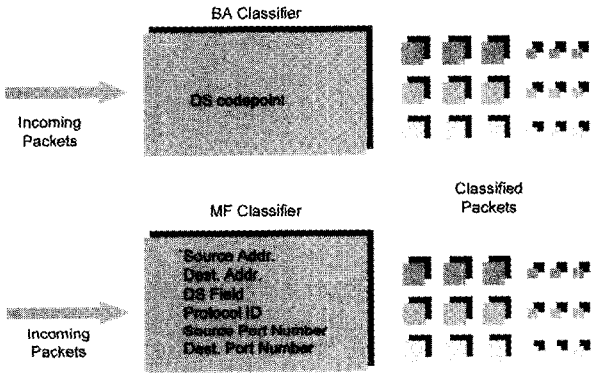
DS는 upstream 네트워크와 downstream DS 영역 사이에 SLA(Service Level Agreement)를 설정하여 확장될 수 있다. SLA는 패킷 등급화과 재표식 법칙과 트래픽 프로파일등을 규정한다. 또한 TCA(Traffic Conditioning Agreement)는 SLA로부터 구성되어진다. 트래픽 조종은 TCA에 규정된 협의내용에 일치하여 metering, shaping, policing과(또는) 재표시를 수행하게 된다.

(1) Classifiers

패킷 classifiers는 패킷헤더의 내용에 기초하여 트래픽 스트림에서 패킷을 선택하는 기능을 담당한다. Classifier는 다음과 같이 크게 두 가지의 형태로 정의 된다.

① BA(Behavior Aggreate) Classifier : DS codepoint만으로 분류.

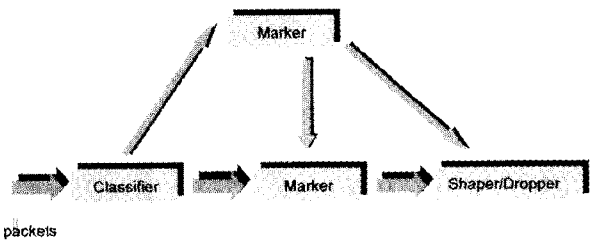
② MF(Multi-Field) Classifier : 발생지주소, 목적지주소, DS 필드, Protocol ID, 근원지 포트번호r, 목적지 포트번호r 등의 필드로 분류



(그림 2) Classifier의 종류

(2) Traffic Conditioners

Traffic conditioner은 meter, marker, shaper, dropper으로 구성되며, meter는 트래픽 프로파일에 대한 트래픽 스트림을 측정하는데 사용된다. 패킷에 대한 meter의 상태는 maring, dropping 또는 shaping등이 다음 동작을 취하게 한다.



(그림 3) Traffic Conditioner

트래픽 meters는 TCA에 규정된 트래픽 프로파일을 기초

로 하여 선택된 패킷스트림의 시간적인 특성을 측정하게된다. meter는 in 또는 out-of-profile인 패킷에 대해서 다른 조정 장치들이 특정한 행동을 시작할 수 있도록 상태 정보를 다른 장치들에게 전달하게 된다. Marker는 meter의 상태에 따라서, 모든 패킷을 단일 CP로 표식하거나 PHB 그룹의 PHB를 선택하는데 사용되는 CP로 패킷을 표식하는 작업을 담당한다. Marker가 패킷의 CP를 변경했을때, 재표식이라고 말한다. Shapers는 스트림이 트래픽 profile에 따르도록 트래픽 스트림내의 모든 (또는 일부분)패킷을 지연시킨다. shaper는 보통은 일정크기의 버퍼를 가지는데 지연된 패킷을 저장할 버퍼가 충분하지 않으면 패킷들은 폐기된다. Droppers는 스트림이 트래픽 프로파일의 규정을 따르도록 트래픽 스트림내의 모든 (또는 일부분)패킷을 폐기하는 기능을 담당한다. droppper는 shaper의 버퍼 크기를 0이나 작은 패킷으로하여 구현될 수 있다.

2.1.4 DS에서의 SLA, TCA

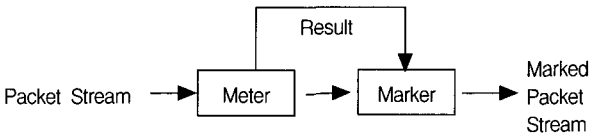
DiffServ는 aggregation(class)로 서비스를 제공하며, 사용자와 제공자 사이에 서비스에 대한 협정을 하여야 하는데, 일반적으로 사용자/제공자 경계에서 행해진다. 이러한 행위를 Service Level Agreement(SLA), Traffic Conditioner Agreement(TCA)라 칭한다. 각 DiffServ 사용자/제공자 경계에서 제공되는 서비스는 SLA 폼에 의해 정의된다. SLA의 중요한 부분집합으로서 TCA를 들 수 있는데, TCA는 각 서비스레벨에 대한 서비스 파라미터를 명세하는 것이다. 서비스 파라미터는 다음과 같은 것들을 포함한다.

- 예상되는 throughput, drop 가능성, 대기시간과 같은 상세화된 서비스 성능파라미터
- 토큰 버켓(token bucket) 파라미터와 같은 요청된 서비스를 제공하기 위해 첨부해야할 트래픽 profile
- 명세된 프로파일을 초과하지 못하게 하는 트래픽 설정
- Marking 서비스 제공
- Shaping 서비스 제공

2.2 Token Bucket을 사용하는 기존 모델

토큰버켓 Meter는 보다 더 정교한 방법으로 세 개의 파라메터 average rate, packet rate, burst size를 갖는다. 기본적으로 average rate와 패킷의 arrival rate를 비교하는 방법을 사용하며 Token Bucket profile의 일치여부로 측정하는 방법이다. IETF의 Integrated Services Internet Model에서는 토큰버켓 모델에 근거를 두고 있는데 이 모델에 근거를 이루고 있는 기존 제안된 방법들에는 다음과 같은 것들이 있다.

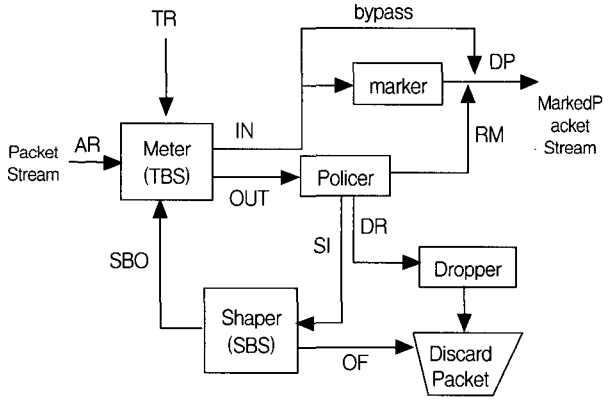
srTCM(Single Rate Three Color Marker)은 세 개의 트래픽 파라메터에 의해 트래픽 스트림을 측정하여 green, yellow, red로 패킷을 마크한다. 이 방법은 두 가지 모드(Color-Blind, Color-Aware)와 세 가지 트래픽 파라메터인 CIR(Committed Information Rate), CBS(Committed Burst Size), EBS(Excess Burst Size)로 Traffic rate를 측정한다. srTCM의 동작방식은 아래 그림과 같다.



(그림 4) srTCM의 블록도

trTCM(Two Rate Three Color Marker)은 srTCM과 비슷하게 동작하나 IP 패킷 스트림을 측정하는데 두 개의 rate(PIR, CIR)와 관련된 burst size을 기반으로 하여 green, yellow, red중 하나로 마크하는 방법이다. 이는 패킷이 PIR을 초과하면 red로 마크하고, 그렇지 않으면 CIR를 초과하는지 여부에 따라 yellow, green으로 마크된다. trTCM은 srTCM과 거의 동일한 방법으로 연산된다[6].

GTC(Generic Traffic Conditioner)는 아래 그림과 같이 다섯 개의 기능적인 구성요소로 구성된다.



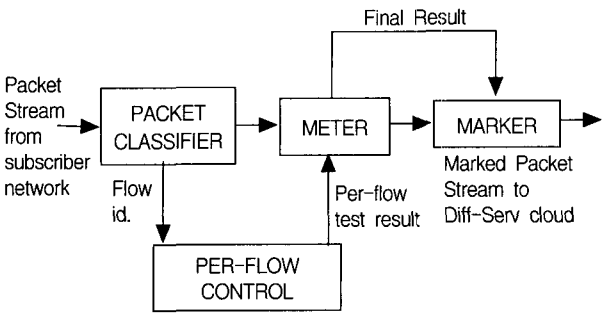
(그림 5) GTC의 블록도

〈표 1〉 GTC에 사용되는 파라메터

Token Bucket Occupancy(TBO)
Shaping Buffer Occupancy(SBO)
SI : 단위시간당 shaper에 도착되는 바이트의 수
IN : 단위시간당 in-profile 패킷의 바이트
OUT : 단위시간당 out-of-profile 패킷의 바이트
DR : 단위시간당 drop되는 out-of-profile 패킷의 바이트 수
RM : 단위시간당 re-mark되는 바이트의 수
OF : 단위시간당 shaping buffer 오버플로우 수

이 방법의 알고리즘은 크게 두 부분으로 구성되는데 Meter 부분과 Policer에 관한 부분이다[7].

FM(Fair Marker)은 동일 가입자망에서 유입되는 플로우들이 그들간의 공정성을 갖도록 하기 위해 토큰버킷으로부터 토큰 분배를 제어한다. FM의 블록도는 다음과 같다.



(그림 6) FM 블록도

(그림 6)의 meter 컴포넌트는 두 개의 트래픽 파라메터값 할당에 의해 구성된다. PER-FLOW CONTROL 내의 fair 할당 알고리즘은 플로우가 다른 플로우들과 비교하여 부적절한 token 양을 사용하는지 시험한다[8]. 지금껏 나열한 Token Bucket Meter에 기반한 모델 외에 Average Rate Meter의 방법을 사용하는 TSWTCM(A Time Sliding Window Three Colour Marker)등이 있다[10].

하지만 이러한 모델들은 여러 하부네트워크에서 기인하는 협약된 플로우들을 측정하는 과정에서 토큰킷 Burst를 발생시키는 주체가 한 플로우에서 기인한 것인지 아니면 여러 플로우들에 의해 기인한 것인지를 전혀 고려하지 않고 있는 문제점을 지니고 있다. 즉, classifier를 거친 패킷들(aggregate flow)은 현 시점에서 자신과 같은 서비스를 받고자 하는 패킷들이 많아 Burst가 발생할 경우 예기치 않은 서비스 손실을 입을 수 있다는 것이다.

2.2.1 PHB 종류와 적용

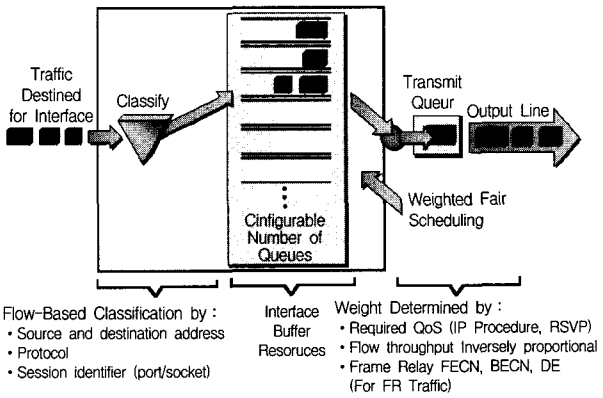
지금까지 제안된 PHB에는 다음과 같은 종류가 있다. IANA에서는 DS 영역중 6bit를 사용하여 64개의 CP를 사용할 것을 제안하고 있는데 크게 Default PHB(codepoint 000000), AF(Assured Forwarding) PHB, EF(Expedited Forwarding) PHB세 가지로 분류하고 있다[1].

또한 CBSD(Class Based Service Differentiation)에서는 과거 호환적인(Backward compatibility) 방법으로 상대적인 우선 순위에 기반을 두고 기존 IP precedence bit 값을 이용 가능하도록 제안하는 방법이다[13].

마지막으로 DSWG에서는 DSCP 값은 64개이지만 PHB의 수는 제한하지 않고 있다. 즉, network 도메인에서의 DSCP : PHB mapping은 지역적으로 정의할 수 있고, 표준화된 하나의 DSCP : PHB 값 mapping을 권장하지만 네트워크 관리자의 다른 방법이 선택 가능하도록 제약을 두지 않고 있다. 이런 이유로 IANA에서 16bit binary field을 사용하여 0~4095 범위의 PHB 사용을 제안하고 있다.

- Weighted Fair Queuing(WFQ)

WFQ(Weighted Fair Queuing)는 보장된 대역폭 서비스를 허락하는 패킷 스케쥴링 기술이다. WFQ의 목적은 동일 링크를 몇 개의 세션이 공유하게 하는 것이다. WFQ는 큐가 대역폭에 대해 기아상태에 빠지지 않고, 트래픽이 예측 가능 한 서비스를 얻을 수 있음을 보장한다. Low-volumn 트래픽 스트림이 우선적으로 서비스를 받고 High-volumn 트래픽 스트림이 그들 사이에 비례적으로 남아있는 용량을 공유하는 메커니즘을 이용한다[16].

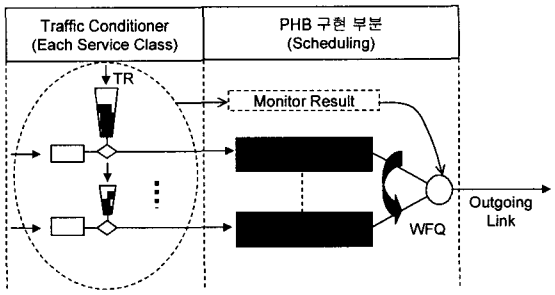


(그림 7) Weighted Fair Queuing

(그림 7)는 일반적인 WFQ의 수행과정을 보이고 있다. WFQ의 이점 중 가장 중요한 것은 QoS 시그널링과의 접목으로 더 향상된 기능을 수행할 수 있다는 것이다. 즉, IP precedence를 QoS 시그널링으로 사용할 경우 동적으로 자원 사용의 비율(link 이용비율)을 시그널링을 통해 변경할 수 있다. 또한 RSVP 시그널링을 사용해서 버퍼 공간 할당이나, 패킷 스케쥴링, 예약된 자원에 대해 보장한 대역폭을 제공하기 위해 WFQ를 사용할 수도 있다.

2.2.2 Traffic Conditioner와 PHB 구현의 관계

Traffic conditioner는 결과적으로 PHB 구현에 필요한 제어를 수행하는 곳이다. DS에서 제공되는 서비스는 PHB 구현을 어떻게 제공하느냐에 따라 서비스의 종류와 각 서비스가 제공하는 QoS가 결정이 되는데, Traffic Conditioner는 서비스를 제공할 플로우를 선별하는 기능을 제공하는 것이라 할 수 있다. 서비스를 제공할 플로우 선별을 DiffServ 모델에서는 일반적으로 토큰버켓 모델을 사용하는 meter를 통해 수행된다.



(그림 8) DS Node에서의 TC와 PHB구현의 관계

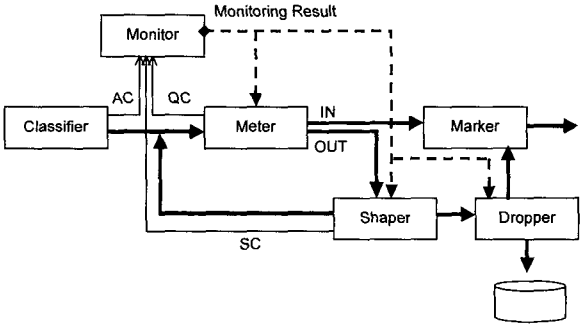
(그림 8)은 각 서비스 클래스에 대한 모니터링 결과값을 WFQ에서 링크 이용비를 산정하는 값으로 사용하는 것을 보인 것이다.

3. 제안 모델 연구

본 논문에서는 기존의 traffic conditioner을 구성하는 meter나 shaper, dropper가 traffic profile에 유입된 플로우들을 일치시키는 역할만을 수행하고 있는데, 이러한 기능에 자원을 공정하게 사용하고 잉여 자원을 효율적으로 사용할 수 있는 모델을 제시하고 있다. 기존 모델이 내포하고 있는 심각한 문제는 토큰 버킷에서 burst가 발생되는 시점에 도착하는 패킷은 항상 out-of-profile로 판명이 되어 예기치 않은 손실을 야기할 수 있다는 점이다. 자주 발생되지는 않지만 극단적인 경우(토큰 버킷에서 여유분의 토큰이 존재하지 않을 때, 토큰이 생성되는 시점에서 두 플로우가 경합될 때) 두 서브네트워크에서 진입하는 플로우들 또는 동일 서브네트워크로부터 진입하는 다수의 플로우들 간에 경합이 발생할 때 둘 중 시간상 늦게 도착하는 플로우가 항상 손실을 입을 수도 있다는 것이다.

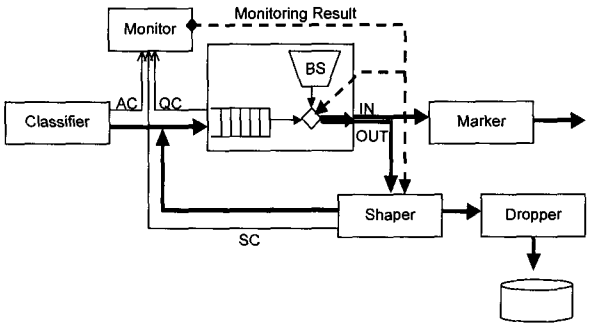
3.1 제안 모델 TC의 구성 및 기능

제안모델에서는 단순히 시간적인 특성에 의존하지 않고, classifier를 통해 각 서비스별로 분류된 협약된 플로우에 대해 Monitor라는 컴포넌트를 추가하여 버킷의 burst를 사전에 감지하여 burst를 일으키는 협약된 플로우중에서 평균 토큰 사용 비율을 넘는 망의 패킷을 식별하여 토큰 버킷에서의 측정 없이 바로 shaper로 넘겨지게 하는 방안을 제시한다. 또한 Dropper에서의 지능적인 동작으로 인해 자원 사용의 효율성을 제공할 수 있는 방안을 제시한다.



(그림 9) 제안하는 Traffic Conditioner 모델

(그림 9)은 TC 컴포넌트들 간의 유기적인 관계를 보이고 있다. 굵은 화살표는 패킷의 흐름을 표현한 것이고, 가는 화살표와 점선으로 된 화살표는 제어정보의 흐름을 의미한다. 제안하는 Traffic Conditioner는 6개의 컴포넌트로 구성된다.



(그림 10) TC 구성컴포넌트의 확장

<표 2>에서는 (그림 9)와 (그림 10)에서 사용되는 파라미터를 설명하고 있다.

〈표 2〉 제안모델에서 사용되는 파라미터 정의

NAME	설 명
AC	Classifier에 의해 분류된 협약된 플로우양
SC	remetering을 위해 되돌려질 플로우양
QC	TB의 처리를 대기하는 플로우양
IN	측정후 in-profile로 판명된 packet
OUT	측정후 out-of-profile로 판명된 packet
BS	버켓의 size

<표 4>는 제안하는 Traffic Conditioner 모델의 전체적인 알고리즘을 보이고 있다.

〈표 3〉 Monitor 컴포넌트 구성 파라미터

구성 파라미터		의 미
PHB(Traffic profile)		서비스 레벨
S_Bytes		토큰 버켓에서 초당 제공 가능한 바이트 수 (대역폭)
source address		플로우 식별자
F_Bytes		플로우 식별자에 대한 바이트 수
F_Num		동일 협약된 플로우를 구성하는 플로우 수
Byte_thresold		계산 시 사용될 수용가능 한 바이트 수
플로우 byte 수	AC	Classifier에 의해 분류된 협약된 플로우에 대한 바이트 수
	SC	재표식을 위해 되돌려질 플로우의 바이트 수
	QC	토큰 버켓의 처리를 대기하는 플로우의 바이트 수

〈표 4〉 제안모델의 전체 알고리즘

```
AC = (Classifier를 거친 Aggregated flow양)
QC = AC                                  // 초기 대기 큐에 삽입
while(TRUE) {
    Token Bucket과 Shaper에서의 처리

    //== 다음단계 상황 Monitoring ==//
    AC = (Classifier를 거친 Aggregated flow양)
    TC = AC + SC + QC
                          // 다음단계에 사용될 총 플로우양 계산
```

```
if(TC > BS) {                    // token bucket burst 발생
    if(TC-SC > BS) {             // burst가 SC에서 기인한다면
        TC -= SC
        aggregated flow중 가장 많은 토큰을 사용하는
                패킷/양 식별(Shaper로 바로 전송할
                flow/flow 양)
        remetering을 위해 전송되는 플로우양을 0으로 지정
        SC flow를 dropper로 전달
    } else {
        TC = BS - (AC+QC)
        remetering을 위해 전송되는 플로우양을 TC로 지정
    }
} else {
    TC = BS - TC
    TC 값 만큼 Dropper에서 Marker로 전송
}
TC = 0
}
```

Meter와 Shaper 컴포넌트에서는 개별적으로 할당된 큐를 사용한다. Dropper 컴포넌트에서는 플로우 식별자를 관리하는 대신에 패킷이 도착된 시간 값을 사용한다. Meter, Shaper, Dropper 컴포넌트 모두 큐의 오버플로우를 처리하기 위해 플로우 크기의 총 합이 항상 정해진 큐 길이의 범위를 넘지 않도록 처리된다.

3.2 모델간의 성능평가

성능 평가는 ns에서 제공하고 있는 DiffServ 관련 파일들 중 일부를 수정하여 제안방법을 추가하였고 OTcl을 이용하여 시나리오를 작성하는 방법을 사용하였다. 또한 가시적인 결과를 보이기 위하여 OTcl 프로그램을 작성하여 최종적으로 Xgraph 유틸리티를 이용하여 결과를 보였다.

시뮬레이션을 위해서 노드번호 0과 1이 DS 서비스를 제공하는 노드이고, 양쪽 7개 쌍의 노드들이 DS 망을 이용하는 근원지와 목적지 노드로 구성되며, 7개의 근원지 노드는 두 개의 EF 서비스로 협약된 노드, 3개의 AF 서비스로 협약된 노드와 2개의 BE 서비스로 협약된 노드로 구성하였다.

〈표 5〉 시뮬레이션에서의 노드 특성

근원지 노 드	Agent	소 스	Agent	목적지 노 드
2,4	Agent/UDP EF	Application/Traffic /CBR	Agent/NULL BE	3,5
6,8,10	Agent/TCP AF	Application/FTP	Agent/TCPSink AF	7,9,11
12,14	Agent/TCP BE	Application/FTP	Agent/TCPSink BE	13,15

먼저 대역폭을 1 *Mbps*로 설정한 경우 각 노드들이 DS 망을 사용할 때 서비스를 제공받은 대역폭을 <표 6>에서 보이고 있다. 세 개의 AF11 서비스로 협약된 소스 2, 3, 4는 각각 114, 137, 148 *Kbps*의 서비스를 제공받았음을 의미하고 있다. 세 개의 동일한 서비스레벨에 속하는 플로우들 중 제공받은 서비스의 최고 차가 34 *Kbps*임을 보이고 있다.

〈표 6〉 1 *Mbps* 대역폭에서의 자원사용량

```
Goodput per source :
Goodput for source 0 (EF) : 99.372 kb/s
Goodput for source 1 (EF) : 0.63 kb/s
Goodput for source 2 (AF11) : 114.0 kb/s
Goodput for source 3 (AF11) : 127.90000000000001 kb/s
Goodput for source 4 (AF11) : 148.09999999999999 kb/s
Goodput for source 5 (BE) : 250.0 kb/s
Goodput for source 6 (BE) : 250.0 kb/s
```

<표 7>은 동일 환경에서 대역폭만을 5 *Mbps*로 변경하여 실행한 시뮬레이션 결과를 보이고 있다.

〈표 7〉 5 *Mbps* 대역폭에서의 자원사용량

```
Goodput per source :
Goodput for source 0 (EF) : 199.75200000000001 kb/s
Goodput for source 1 (EF) : 199.773 kb/s
Goodput for source 2 (AF11) : 638.5 kb/s
Goodput for source 3 (AF11) : 707.5 kb/s
Goodput for source 4 (AF11) : 691.0 kb/s
Goodput for source 5 (BE) : 1277.9000000000001 kb/s Kbps
Goodput for source 6 (BE) : 1278.0 kb/s
```

<표 7>의 결과에서도 <표 6>에서의 결과와 마찬가지로 AF 서비스로 협약된 플로우들이 자원을 공정하게 사용하고 있지 못함을 보이고 있다.

제안모델 적용전의 AF 서비스에 해당되는 플로우들의 자원사용의 최대 차가 34 *Kbps*의 결과였는데, Monitor 컴포넌트를 통한 사전의 모니터링 결과를 적용하였을 경우는 <표 8>와 같이 동일 서비스 레벨에 해당하는 플로우가 대략 5 *Kbps*의 차를 보임을 알 수 있다.

〈표 8〉 1 *Mbps* 대역폭에서의 실행 결과

```
Goodput per source :
Goodput for source 0 (EF) : 99.372 kb/s
Goodput for source 1 (EF) : 0.63 kb/s
Goodput for source 2 (AF11) : 136.0 kb/s
Goodput for source 3 (AF11) : 131.30000000000001 kb/s
Goodput for source 4 (AF11) : 131.19999999999999 kb/s
Goodput for source 5 (BE) : 250.0 kb/s
Goodput for source 6 (BE) : 250.0 kb/s
```

<표 9>에서는 링크 대역폭을 5 *Mbps*로 설정한 경우의 결과를 보인다. 이 결과에서도 동일 서비스레벨의 플로우 들간의 최대 차가 7 *Kbps*임을 확인할 수 있다.

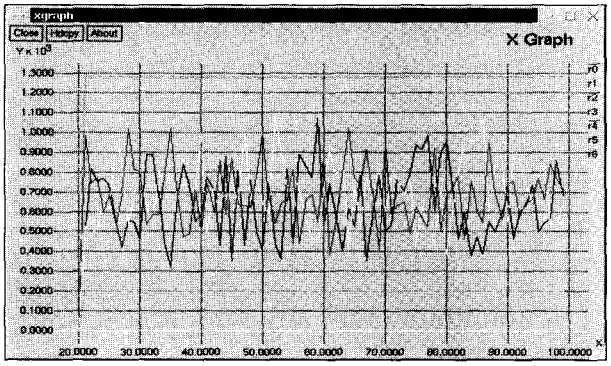
또한 <표 7>와 <표 8>를 비교해 보면 심한 폭주로 인해 자원사용이 비슷하지만 <표 6>과 <표 9>을 비교해보면 제안모델이 3 *Kbps*의 자원을 더 사용하고 있음을 알 수 있다. Dropper에 저장된 자원을 모니터링 경과 잉여 자원이 존재할 때 사용하여 자원사용의 효율성이 높아짐을 알 수

있다.

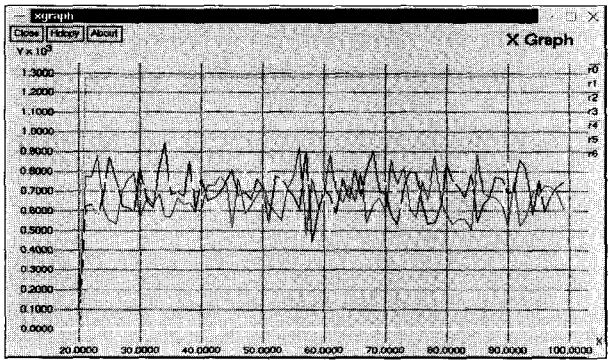
〈표 9〉 5 *Mbps* 대역폭에서의 실행결과

```
Goodput per source :
Goodput for source 0 (EF) : 199.75200000000001 kb/s
Goodput for source 1 (EF) : 199.75200000000001 kb/s
Goodput for source 2 (AF11) : 678.89999999999998 kb/s
Goodput for source 3 (AF11) : 684.29999999999995 kb/s
Goodput for source 4 (AF11) : 677.20000000000005 kb/s
Goodput for source 5 (BE) : 1277.8 kb/s
Goodput for source 6 (BE) : 1278.0 kb/s
```

대역폭을 5 *Mbps*로 설정한 경우의 결과를 (그림 11)에 나타내었다. r0에서 r6까지는 source 0에서 source 6까지의 플로우들을 의미하며, x 축은 시간축(sec 단위)을 의미하며, y 축은 단위의 전송률을 보이고 있다.

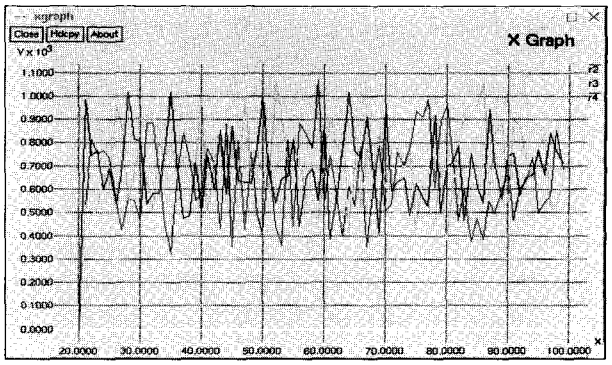


(그림 11) 기존 토큰 버켓 DiffServ 모델에서 각 플로우의 대역폭 사용

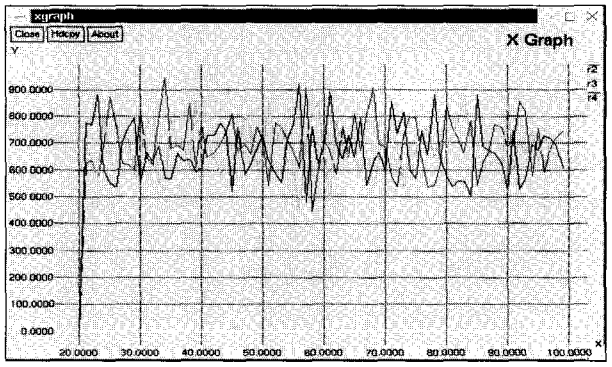


(그림 12) 제안모델 적용 후 대역폭 사용량

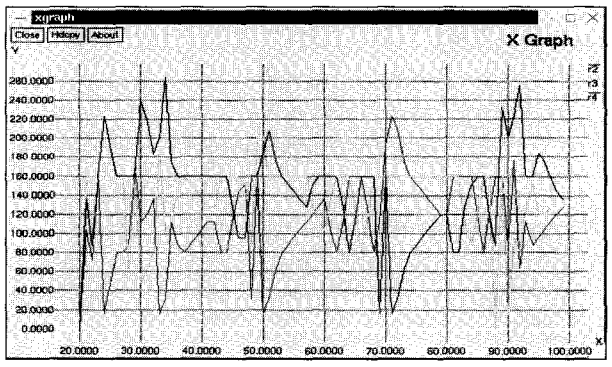
(그림 12)은 제안모델 적용전의 결과와 동일한 환경에서 제안모델 적용 후의 결과를 보인 것이다. 동일하게 7개의 플로우가 초당 할당받아 사용한 대역을 나타낸 결과인데, AF 플로우의 상한과 하한의 폭이 현저하게 좁아짐을 볼 수 있으며, 이는 AF 플로우들 간의 자원사용이 공정하게 분배됨을 보여주고 있다. (그림 13)에서 (그림 16)까지의 $r2$, $r3$, $r4$는 AF11 협약을 맺은 서비스에 대한 플로우들의 전송률을 보이고 있다.



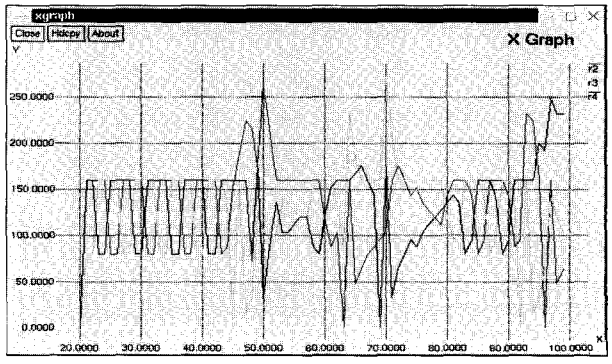
(그림 13) 기존 토큰 버켓 서비스에서 AF PHB 서비스의 자원 사용량



(그림 14) 제안모델 적용 후의 AF PHB의 자원 사용 량



(그림 15) 기존 토큰 버켓 모델의 1Mbps 대역에서의 자원사용량



(그림 16) 제안 모델 적용후 1Mbps에서의 자원 사용량

4. 결 론

현재 활발히 연구 중에 있는 DiffServ 아키텍처에서는 동일 서비스 레벨의 플로우들에 대해 자원공유의 불공성이라는 문제를 내포하고 있다. 이 문제는 동일한 서비스레벨로 협약된 사용자들 중 불특정 사용자에 대한 QoS 손실을 초래할 수 있으므로 요금 부과를 전제로 하는 DiffServ 아키텍처에서는 반드시 지켜져야 할 중요한 문제라 할 수 있다. 본 논문에서는 공정한 자원공유를 할 수 있는 Traffic Conditioner를 설계 구현하고 그에 대한 성능을 평가하였다. 또한 토큰버킷 방식의 단점을 보완할 수 있는 TC를 만들기 위하여 사전 모니터링을 통해 burst를 감지하여, 하나의 협약된 플로우들을 이루는 다수개의 플로우들 간에 공정한 자원 공유를 할 수 있는 방법과 잉여 자원을 효율적으로 사용할 수 있는 방안과 그에 대한 알고리즘을 제안하였다.

현재 논문에서는 트래픽 특성을 고려하지 않고 단일 트래픽이라 가정하였는데 향후 연구방향은 트래픽 특성을 고려하여 좀더 정교한 QoS를 만족시킬 수 있는 사항들을 연구하고자 한다.

참 고 문 헌

[1] D. Black, S. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," RFC 2475.

[2] Davies, E., Keshav, S., Verma, D., Carlson, M., Ohlman, B., Blake, S., Bernet, Y., Binder, J., Wang, Z., Weiss, W., "A Framework for Differentiated Services," Internet Draft, October, 1998.

[3] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC2474, December, 1998.

[4] Jacobson, V., Nichols, K., Poduri, K., "An Expedited Forwarding PHB," RFC 2598, August, 1998.

[5] J. Heinanen et al., "Assured Forwarding PHG Group," RFC 2587, February, 1999.

[6] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker," Internet Draft, March, 1999.

[7] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker," Internet Draft, March, 1999.

[8] Y. Bernet, A. Smith, S. Black, "A Conceptual Model for Diffserv Routers," Internet Draft, December, 1999.

[9] Fang-Ching Ou, "A Generic Traffic Conditioner," Internet Draft, October, 1999.

[10] Constantinos Dovrolis, "Class-Based Service Differentiation," Internet Drafe, December, 1998.

[11] Marty Borden, Christopher White, "Management of PHBs," Internet Draft, August 1998

[12] B. Braden et al., "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," Internet RFC 2205, September, 1997.

[13] Peter, F., Bernet, Y., "Integrated Services Over Differentiated Services," Internet Draft, March, 1998.

[14] Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Nichols, K., Speer, M., "A Framework for the Use of RSVP With Diff-serv Networks," Internet Draft, June, 1998.

[15] UCB, LBNL, VINT Network Simulator - ns, http://www-mash.cs.berkeley.edu/ns/ns.html.

[16] UC Berleley, LBL, USC/ISI and Xerox PARC, "Ns Notes and Documentation," http://www-mash.cs.berkeley.edu/ns.

장 경 성

e-mail : unixhunt@chodang.ac.kr

1986년 전남대학교 물리학과(학사)

1991년 호주 타스마니아주립대 응용컴퓨터공학과 Graduated Dip. 졸업

1996년 전남대학교 전산통계학과(이학석사)

2002년 전남대학교 전산통계학과(이학박사)

1997년~현재 초당대학교 정보통신공학과 조교수

관심분야 : Internet, Wireless & Mobile Computing, Security, Distributed System, Operating System

강 대 욱

e-mail : dwkang@chonnam.ac.kr

1982년 전남대학교 계산통계학과(이학사)

1985년 전남대학교 계산통계학과(이학석사)

2000년 영국 Newcastle University 박사과정 수료

1984년~현재 전남대학교 전자컴퓨터정보통신공학부 교수

관심분야 : Mobile Computing, distributed Computing, Network, Mobile Agent