한국시뮬레이션학회 논문지
Vol. 18, No. 3, pp. 91-102 (2009. 9)

DiffServ 네트워크에서 공평성 향상을 위한 TRC Dropper의 성능 분석

김훈기[1] · 홍성화[1†]

The Performance Analysis of TRC Dropper to improve fairness on DiffServ Networks

Hoon-Ki Kim · Sung-Hwa Hong

ABSTRACT

The average window size is most closely related to average throughput. In order to improve fairness, the proposed dropper tries to control the window size of each flow to equal level by intentional packet drop. Intentional packet drop is performed only to the flows that have been occupied bandwidth in a large amount. Because of intentional packet drop, this flow cut down its transmission rate to a half. Accordingly, somewhat capacity of core link comes into existence. And other flow can use this new capacity of this link. Hence other flows have more throughput than before. In this paper, we propose the TRC (Transmission Rate Control) Dropper improving the fairness between individual flows of aggregated sources on DiffServ network. It has the fairness improvement mechanism mentioned above paragraph.

Key words : QoS, Diffserv, Fairness

요 약

평균 윈도우 크기는 평균 처리율과 많은 관계를 가지고 있다. 제안된 dropper는 fairness를 향상시키기 위해 윈도우 크기를 각 플로우의 레벨에 맞게 제어하도록 한다. 경쟁 패킷 드랍은 플로우가 많은 양의 대역폭을 차지하기 위해 동작한다. 경쟁 패킷 드랍 때문에 플로우는 전송 속도를 반으로 줄이게 되며, 이에 따라 코어 링크의 여유 용량이 존재하게 된다. 다른 플로우는 이 새로운 용량을 사용하게 되고, 따라서 다른 플로우는 좋은 처리율을 가지게 된다. 이런 방식으로 TCP의 전송속도를 제어하는 edge router의 새로운 기능 block인 TRC Dropper(Transmission Rate Control Dropper)를 제안한다. 제안한 TRC Dropper를 통하여 플로우들 간에 공평한 대역폭 분배를 실현하여 상대적으로 차별화된 서비스를 제공할 수 있다.

주요어 : QoS, Diffserv, Fairness

1. 서 론

네트워크에 있어 기술적인 발전이 계속 되어 오고 있지만, 최근 인터넷 서비스가 급속히 발전하면서 더욱더 빠르고 보다 안정적인 서비스 요구가 증가하고 있다.

현재 대부분의 네트워크는 파일이나 전자우편, 웹 브라우징 등 지연에 크게 민감하지 않고 평균적인 전송 속도의 제공에 중점을 두고 있었다. 하지만 점차 음성이나 동영상의 실시간 전송과 같이 지연이나 지터 등에 민감한 멀티미디어 서비스가 등장하면서 각각의 서비스에 대한 품질 보장 문제가 등장하고 있다. 기본적으로 인터넷은 네트워크 트래픽을 하나의 클래스(Class)로 취급해 동일한 정책을 적용하는 Best-effort(BE) 방법을 채택하고 있다.

이 방식은 인터넷 사용자가 적었을 때는 링크 이용률을 극대화 할 수 있는 효과적인 방법이었지만 근래에 들어 폭발적으로 인터넷 사용자의 수가 증가함에 따라 차츰 그 문제점과 한계점이 드러나고 있다. 이를 보완하기 위한 방법으로 다양한 QoS 프로토콜이나 표준이 만들어졌는데, 대표적인 QoS 관련 프로토콜과 표준에는 IntServ

2009년 6월 29일 접수, 2009년 9월 17일 채택
[1] 동양공업전문대학 전산정보학부
주 저 자 : 김훈기
교신저자 : 홍성화
E-mail; shhong@dongyang.ac.kr

(Integrated services), DiffServ(Differentiated services), RSVP(Resource reservation protocol), MPLS(Multi protocol label switching) 등이 있다.

기존의 IntServ는 트래픽 흐름(flow)에 대해 미리 필요한 시간에 필요한 만큼의 대역폭을 할당받는 방식으로, 그 흐름들에 대해 100% 보장된(guaranteed) 서비스를 제공받았다. 그러나 IntServ는 코어 라우터에서 per-flow management를 하기 때문에 확장성(Scalability)에 심각한 문제를 안고 있어서 현재는 DiffServ쪽으로 연구 관심이 집중되어 있다.

DiffServ는 패킷들을 비슷한 성질을 가진 클래스로 구분해서 중요한 클래스에 대해서는 가중치를 둬 서비스를 제공하는 방법이다. DiffServ는 100% 보장된 서비스를 제공하지는 못하지만, 구현이 쉽고 확장성이 뛰어나기 때문에 현재 인터넷에서 가장 많이 채택하는 방식이다.

DiffServ에서는 크게 서비스 종류에 따라서 보장의 불공평성이 나타나게 되는데, 이는 서비스 클래스 간에 상대적인 QoS인 CoS(Class of Service)를 제공하는데 그 취지가 있다. 즉, 높은 수준의 계약을 하면 낮은 수준의 계약을 한 사용자보다는 상대적으로 더 나은 서비스를 제공 받을 수 있어야 한다. 하지만 확률적이고 상대적인 보장만을 해 주는 assured class는 플로우 간에 처한 상황의 차이에 따라 평균 수율에 있어 불공평성이 자주 나타나게 된다.

이런 문제가 발생하는 이유는 TCP 알고리즘이 정해진 일정한 전송속도로 패킷을 전송하지 않고 sliding window mechanism을 이용함에 따라 순간 전송 속도가 수시로 변하기 때문이다. TCP는 패킷 드랍이 없으면 항상 윈도우 크기를 증가시키고, 패킷 드랍이 있으면 윈도우 크기를 감소시킨다. 즉, TCP는 이미 전송한 패킷의 드랍 여부에 따라 전송속도를 조절한다.

제안하는 dropper는 플로우의 전송 속도를 조절하기 위해 에지 라우터에서 인위적인 패킷 드랍을 수행한다. 패킷이 드랍되는 정도는 해당 플로우의 네트워크 자원 점유율에 따르게 된다. 네트워크 자원의 점유율을 나타내는 값으로는 in profile 패킷과 out-of profile 패킷의 전송속도의 비율인 profile index 값을 사용한다. 이 값을 기준으로 대역폭 점유율이 높다고 판단되는 플로우에게 인위적인 패킷 드랍을 실시한다. 그러면 이 플로우는 전송속도가 줄어들게 될 것이고 그만큼 네트워크 내부에는 여유 공간이 발생하게 된다. 이를 대역폭 점유율이 낮은 플로우들이 자연스럽게 차지함으로써 이들의 평균 수율이 커지게 되고 결과적으로 비슷한 수준의 평균 수율을 가지도록 해준다. 이런 방식으로 TCP의 전송속도를 제어하는 에지 라우터의 새로운 기능 block으로서 TRC Dropper (Transmission Rate Control Dropper)를 제안한다.

본 논문에서는 제안한 TRC Dropper를 통하여 플로우들 간에 대역폭의 공평한 분배를 하도록 네트워크 하에서의 여러 상황에 대한 경우의 실험을 통해 상대적으로 차별화된 서비스 품질을 제공할 수 있는지 실험한다.

논문은 다음과 같이 이루어져 있다. 2장은 DiffServ의 문제점을 알아본다. 3장에서는 제안하는 TRC Dropper를 설명하고, 4장에서는 TRC Dropper의 성능 비교를 수행한다. 5장에서는 이 논문의 결론을 맺는다.

2. 관련 연구

ItswTCM(Improved time sliding window Three Color Marker) 방식[2]은 tswTCM[16]을 개량한 marking 방식이다. 기본적으로 token bucket 기반의 marking을 하지 않고 평균 전송 속도 기반의 확률적인 marking을 수행하는 marker이다. 이름에서도 알 수 있듯이 green, yellow, red와 같은 3가지의 색깔을 가지는 marking을 수행한다.

현재까지의 연구에 따르면 3 color marking 방식은 플로우간의 공평성(fairness) 향상에 도움이 되는 것으로 알려져 있다. ItswTCM 방식 역시 플로우간의 공평성 향상에 주목적을 두고 있으며 특히 CIR의 개념을 포함한 알고리즘을 제안하여서 aggregate 간의 공평성에 역점을 둔 marking방식이다.

SCALE-WFS(Scalable Core with Aggregation Level labEling – Weighted Fair bandwidth Sharing) 방식은 코어 라우터에서 실질적인 제어를 한다는 것 자체가 DiffServ의 출현 배경과는 배치되는 생각일 수 있지만 확장성을 상실하지 않는다면 큰 문제가 되지 않을 수도 있다. 이 방식은 확장성을 잃지는 않았지만 전체적인 알고리즘이 복잡하고 코어에서 무리한 계산을 수행함으로써 실질적으로는 엄청나게 계산량이 증가한다는 문제점을 안고 있다.

알고리즘을 간단히 설명하면 다음과 같다. 에지 라우터로 들어온 어떤 플로우 i가 현재 전송하고 있는 패킷의 양이 어느 정도인지를 나타내는 index로 플로우's fairness index($f_i = a_i / r_i$)를 정의하고 계산하여 이 값을 패킷에 marking해 준다.

TCP는 전송한 패킷에 대해 성공적인 전송을 의미하는 ACK를 항상 받게 된다. ACK를 받게 되면 TCP는 한 번에 전송 가능한 패킷의 양을 2배로 증가시키게 된다. 이는 윈도우 크기를 2배로 증가시킴으로써 가능해진다. 패

킷이 네트워크를 통과하는 중에 병목을 겪게 되거나 전송 에러로 인하여 패킷이 손실 되었을 경우에는 ACK를 받지 못하게 된다. 이 때 TCP는 네트워크 내부에 병목이 발생하였다고 판단하고 이에 대한 반응으로 윈도우 크기를 1/2로 감소시킴으로써 순간 전송 속도를 반으로 줄이게 된다. 이때부터 네트워크에는 병목이 발생한 상황이기 때문에 도착하는 ACK 패킷 하나당 윈도우 크기를 2배로 증가 시키지 않는다. 패킷의 1 RTT에 해당하는 시간마다 1씩 윈도우 크기를 증가시키게 된다. 또다시 패킷 드랍이 발생하면 윈도우 크기를 1/2로 줄이고 위의 과정을 반복하게 된다. 이를 TCP 혼잡 제어 알고리즘의 하나인 slow-start with congestion avoidance라고 한다.

이 방식은 BE 서비스와 잘 어울리는 방식이다. 계약과 보장의 개념이 없는 경우에는 어떤 패킷이 드랍되면 병목이 발생한 경우이므로 호스트의 전송 rate를 줄여서 병목을 해결하게 된다. 하지만 DiffServ의 경우에는 사용자와 네트워크가 계약한 전송 품질이 있다. 그럼에도 불구하고 계약된 수준에 미치지 못하고 있는 경우에도 병목에 의한 패킷 드랍이 발생한다면 전송 rate를 감소시킬 수밖에 없게 된다. 이런 상황이 오래 지속된다면 계약한 수준을 네트워크에서 보장해 주는 것은 불가능하게 된다. 이런 부분에서 DiffServ와 TCP 혼잡 알고리즘은 부조화를 일으키게 된다.

3. 제안한 알고리즘

3.1 Profile Index

플로우간에 대역폭을 공평하게 나누어 쓰도록 하기 위해서는 특정 플로우가 점유하고 있는 대역폭이 어느 정도인지를 나타낼 수 있는 수치상의 지표가 필요하게 된다. 기존의 논문들[2,5,6]에서 가장 빈번하게 사용하고 있는 방식은 target rate(r_i)와 average arrival rate(a_i)를 조합하여 사용하는 것이었다.

여기서 시간에 따른 대역폭 점유를 바로 인지하기 위해서 분자(a_i)와 분모(r_i)가 모두 동시에 비례하게 변한다면 이런 문제가 발생하지 않을 것이다. 두 값이 모두 천천히 변할 경우에도 결국에는 두 값의 상대적인 비가 중요하기 때문에 시간에 따른 적절한 대역폭 점유 문제점이 없어지게 될 것이다. 따라서 본 논문에서는 이런 문제점을 갖지 않은 새로운 index값으로 식 (1)의 profile index를 사용한다.

$$Profile\ Index = \frac{the\ number\ of\ OUT\ of\ profile\ packets}{the\ number\ of\ IN\ profile\ packets} = \frac{a_{in}}{a_{out}} \quad (1)$$

a_{in} : arrival rate of in profile packets

a_{out} : arrival rate of out – of profile packets

3.2 TRC Dropper 동작 알고리즘

3.2.1 TRC Dropper의 공평성 향상 메커니즘

TRC Dropper는 플로우들이 공평하게 대역폭을 나누어 가지게 하기 위해서 에지 라우터에서의 인위적인 패킷 드랍을 통하여 평균 윈도우 크기의 크기를 조절한다. 먼저 profile index를 계산해서 네트워크 내부 대역의 점유율을 감시한다. 만일 허용된 대역폭 이상을 차지하고 있다면 점유한 정도에 따라 계산된 드랍 확률에 따라 그 플로우의 패킷들을 버리게 된다. 점유 정도가 높을수록 인위적인 패킷 드랍을 당할 확률은 높아지게 된다. 인위적인 패킷 드랍을 당한 플로우는 해당 패킷에 대한 ACK를 받지 못하게 된다. 해당 플로우의 TCP는 네트워크 내부에 병목이 발생하였다고 생각하여 윈도우 크기를 반으로 줄이며 전송 속도를 감소시키게 된다.

이런 과정에 따라 대역폭 점유율이 높은 플로우는 전송 속도가 감소될 것이다. 그러면 전송 속도가 감소된 만큼 네트워크 내부의 대역폭은 여유가 생기게 된다. 이 때 대역폭 점유율이 낮았던 플로우가 네트워크 내부에서 좀 더 적은 경쟁을 겪게 되고 자연히 여유 대역폭을 더 많이 차지할 수 있게 된다. 그러면 평균 전송 속도가 상대적으로 낮은 플로우들이 평균 윈도우 크기의 크기를 크게 유지하게 되고 그 반대의 경우에는 평균 윈도우 크기가 이전에 비하여 작게 유지 되게 된다. 이런 과정이 얼마간 진행되면 각 플로우간의 전송속도가 비슷해지게 되고 플로우간의 형평성은 향상되게 된다.

3.2.2 TRC Dropper 내부 알고리즘

그림 1은 TRC Dropper 알고리즘의 이해를 돕기 위하여 논리적인 동작 구조를 나타낸 것이다. Rate Measure에서 profile index를 계산한 후, Index Calculation에서 이를 비교하기 쉬운 값인 share index로 변화시키고, 마지막 드랍 Decision에서 share index값을 토대로 대역폭 점유율이 높은 플로우에게 인위적인 패킷 드랍을 실시한다. 이와 같이 크게 3가지의 순차적인 기본 동작을 수행하며 중간에 발생하는 데이터는 LSDB(Leaf State DataBase)에 저장한다.

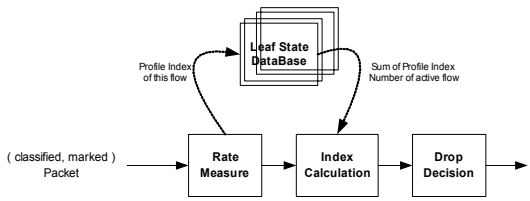

그림 1. TRC Dropper의 논리적인 동작 구조

본 논문에서 제안하는 TRC Dropper 내부 알고리즘의 pseudo code는 다음과 같다.

Assume that
The number of active flow are N.
This packet belongs to flow1(F(1)).

<Rate Measure>
if (Best Effort Service)
FG(1).InPacket is set to the minimum value among that of all AS flow
else if (in profile packet)
FG(1).InPacket ++ or Calculate ain
else
FG(1).OutPacket ++ or Calculate aout
Calculate profile index ($\frac{INPacket}{OutPacket}$ or $\frac{a_{IN}}{a_{out}}$)

<Index Calculation>
Sum of Profile Index = F(1).Profile Index + ….. + F(N).Profile Index;
F(1).ShareIndex = $\frac{FG(1).\mathrm{Pr}ofileIndex}{\sum of \mathrm{Pr}ofileIndex} * N$;

<Drop Decision>
if (In Packet) AND (Share Index ≤ 1.0)
Pdrop = 0.0
else
$$P_{drop} = \frac{1}{MaxSI-1} * (ShareIndex - 1);$$
With the probability of Pdrop, edge router performs a intentional drop.

<Rate Measure>는 Profile index를 계산하는 기능 블록이다. Profile index의 논리적인 의미는 in profile 패킷과 out-of profile 패킷의 비율이지만 실질적인 구현에 있어서는 in profile 패킷의 전송속도와 out-of profile 패킷의 전송속도의 비율을 사용한다.

BE의 경우에는 in profile 패킷이 전혀 발생하지 않기 때문에 다른 AS 플로우의 a_{in} 값 중 가장 작은 값을 이용한다. 이렇게 하면 가장 낮은 수준의 계약을 한 AS 플로우가 차지하게 되는 여유 대역폭 이상을 차지하지 못하게 되어 BE에 대해서 더욱 효과적인 제어가 가능해 진다.

<Index Calculation>는 대역폭의 index를 계산하는 기능 블록이다. Share index는 특정 edge router를 지나고 있는 특정 flow group이 상대적으로 차지하고 있는 대역폭의 비율을 비교가 쉬운 값으로 나타낸 것이다. Share index가 1인 경우는 해당 group에게 허용된 대역폭을 차지하고 있다는 것을 나타내며 1보다 큰 경우는 허용된 대역폭 보다 많이 차지하고 있음을 나타낸다. 반대로 1보다 작은 경우는 허용된 대역폭 보다 적게 차지하고 있음을 나타내게 된다. 예를 들어서 설명하면 하나의 edge router에 4개의 group이 있다고 하자. 각각의 을 나타낸다. 반대로 1는 0.2, 0.5, 0.7, 1.1이라고 하면 을 나타낸다. 반대로 1가 크면 상대적으로 더 많은 out-of profile packet을 발생시키고 있음을 의미하므로 1.1인 flow 가 가장 많은 대역폭을 차지하고 있으며 0.2인 flow가 가장 적은 대역폭을 차지하고 있음을 나타낸다. 하지만 이 값은 전체 DB에 저장된 값들을 일일이 비교해야 하기 때문에 비교가 용이하지 않다. 따라서 이 값을 share index로 변환하게 되며 이 경우에는 각각의 share index는 0.32, 0.8, 1.12, 1.76이 되며 이는 앞의 2개의 group은 허용된 양보다 적게 보내고 있으며 뒤의 2개의 group은 허용된 양보다 많이 보내고 있음을 뜻한다. 물론 적게 보내고 많이 보내는 정도에 있어서는 share index값이 작을수록 더 적은 대역폭을 차지하고 있는 것이며 클수록 더 많은 대역폭을 차지하고 있음을 의미한다.

<Drop Decision>는 패킷 드랍을 계산하는 기능 블록이다. In profile 패킷은 TRC Dropper에 영향을 받지 않는다. TRC dropper에서의 패킷 드랍은 네트워크에서 혼잡이 발생하여 패킷을 드랍시키는 것이 아니기 때문에 계약 rate에 따라 발생한 패킷은 드랍시키지 않는다. share index값이 1보다 크면 클수록 대역폭을 허용된 양보다 많이 차지하고 있다는 것을 의미한다. 따라서 share index값이 1보다 크면 클수록 더 높은 드랍 확률을 가지도록 해야 할 것이다. 물론 1보다 작을 경우에는 드랍 확률은 0이 되어야 할 것이다. 실험에서는 측정된 실험값인 Max_SI 값으로 55를 사용하여 실험을 수행하였다. 실험값이 55인 경우 평균이상의 결과값을 나타내었기 때문이다.

4. 실험에 의한 성능 평가와 분석

이 장에서는 본 논문에서 제안하고 있는 TRC Dropper의 성능을 ns2[22]를 사용하여 평가한다. 기존의 DiffServ 방식에서 발생하는 문제 현상을 실험으로써 밝히며 TRC Dropper를 적용할 경우 평균 수율 측면에서의 fairness이 크게 향상됨을 살펴본다.

기존에 제안된 marking방식인 ItswTCM과 성능을 비교한다. 병목 구간이 1개인 단순한 구조와 병목 구간이 3개인 복잡한 구조에서 모두 실험을 수행하여 그 결과를 제시한다.

4.1 성능 평가 기준

성능 평가를 위한 기준으로는 fairness index[2]값과, 네트워크 내부를 성공적으로 통과한 AS 플로우 트래픽 양을 비교한다. Fairness index는 다른 많은 공평성 보장 관련 논문에서 많이 사용되고 있는 값이다. 해당 관계식은 다음과 같다.

$$FI = \frac{(\sum_i x_i)^2}{N \times \sum_i (x_i)^2} \quad (2)$$

x_i : *the throughput of flow i*

이 값은 x_i값이 평균을 중심으로 흩어져 있는 정도를 나타내는 값으로 표준편차와 비슷한 의미를 가진다. FI의 값이 1인 경우가 각 x_i이 모두 동일한 경우이며, 1에서 작아질수록 xi값들의 산재도가 커짐을 나타낸다. 즉, x_i값이 커질수록 공평성은 감소하고 있음을 나타낸다.

하지만 (2)의 식은 계약이라는 개념이 추가된 네트워크에 바로 적용하기에는 무리가 따른다. 계약 수준이 여러 개인 경우에는 fairness index식은 아래의 식 (3)처럼 바뀌어야 한다.

식 (3)은 계약 1M당 각 플로우가 받은 평균 수율의 공평성을 나타낸다. 즉 1M, 2M의 계약 수준이 있을 경우 후자가 전자의 2배의 평균 수율을 얻어야 fairness index 값은 1이 될 것이다.

$$FI = \frac{(\sum_i \frac{x_i}{r_i})^2}{N \times \sum_i \left(\frac{x_i}{r_i}\right)^2} \quad (3)$$

x_i : *the throughput of flow i*

r_i : *the contract rate of flow i*

두 번째 평가 기준으로는 네트워크 내부의 병목 구간을 성공적으로 통과한 AS 플로우의 트래픽 양을 사용한다. 왜냐하면 평균 수율이 하향 평준화된 경우에도 fairness index값은 1에 가까운 값이 될 수 있기 때문이다.

위의 두 개의 평가 기준 중 하나라도 심하게 성능 저하를 보일 경우에는 해당 알고리즘은 전체적인 성능의 향상을 가져오지 못한 것이 된다. 두 개 모두 우수하거나 최소한 한 개의 기준은 동일한 수준이고 다른 하나의 기준은 우수할 경우에만 성능 향상을 가져 왔다고 결론지을 수 있게 된다.

4.2 병목 구간이 한 개인 간단한 구조

4.2.1 일반적인 변수

그림 2와 같이 병목 구간이 하나인 단순한 모양에 대한 컴퓨터 모의실험 결과를 제시하고자 한다. 이런 네트워크 구조에서의 결과가 실제 네트워크에 적용되었을 때의 결과를 정확하게 나타내어 주지는 않지만 알고리즘의 성능 분석에 있어 비교가 쉽기 때문에 실험을 수행한다.

그림 2는 실험에서 사용한 네트워크 구조를 보이고 있다. 호스트와 에지 라우터간의 링크는 20Mbps를 전송가능하며 10ms의 지연을 가진다. Ingress 라우터와 코어 라우터를 연결하는 링크는 100Mbps를 전송가능하며 10ms의 지연을 가진다. 코어 라우터와 egress 라우터를 연결하는 링크는 그림 의의 구조에서 병목을 유발시키는 링크이다. 이곳을 통과하는 AS 플로우는 4개이다. 따라서 모두 1Mbps의 계약을 할 경우에 계약한 양의 트래픽이 모두 통과하기 위한 최소 대역폭은 4Mbps가 필요하게 되고, 1~4Mbps까지 계약 수준이 다른 경우에는 최소 10Mbps가 필요하게 된다. 여기서 최소 대역폭의 양에는 BE를 위한 대역폭은 배정하지 않았다.

병목 구간의 혼잡도를 변화시킬 수 있는 변수로 α를

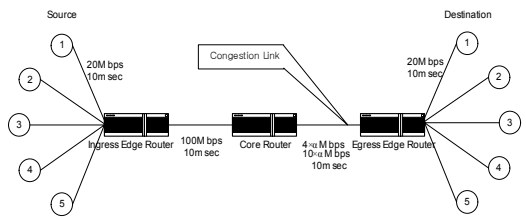

그림 2. 병목 구간이 1개인 간단한 실험 구조

정의하였다. α는 계약한 트래픽이 모두 통과할 수 있는 대역폭의 크기와 실제 병목 구간에 배정된 대역폭의 크기의 비율이다. 즉, α값이 1인 경우는 병목 구간에 최소 대역폭만큼 대역폭이 배정된 상황으로 여유 대역폭이 전혀 없는 경우를 나타낸다. α값이 1보다 큰 경우는 여유 대역폭이 존재하는 경우를 가리키며, 1보다 작은 경우는 최소 대역폭보다 적게 대역폭이 배정된 상황을 가리킨다. α값이 약 0.5~1.0인 경우는 under-provisioned 네트워크를 나타내고, 1.5~2.0인 경우는 well provisioned 네트워크를 나타내며, 이보다 큰 경우는 over-provisioned 네트워크를 나타낸다.

현재까지 뚜렷한 접속 제어 프로토콜이 정의 되지 않았고 네트워크는 시시각각 상황이 변하는 경우가 많기 때문에 여유 대역폭양에 영향을 받지 않고 항상 일정한 수준의 공평성을 제공할 수 있어야 좋은 알고리즘일 것이다.

따라서 모든 실험은 α값이 0.5, 1.0, 1.5, 2.0, 2.5, 3.0인 상황에 대해 수행하였으며 TRC Dropper 알고리즘의 경우에는 전체 범위에 걸쳐 어떤 성능을 보이는지 실험결과를 제시한다.

각 라우터의 버퍼 관리 방식으로는 RIO를 적용한다. 실험에 사용한 token bucket의 크기는 1M 계약한 플로우에 대해 80 패킷s을 사용한다. 2M를 계약한 경우에는 2배의 token bucket 크기인 160 패킷을 사용한다.

4.2.2 호스트별로 RTT가 다른 경우

그림 2의 구조에서 전송 호스트는 5개 이고 수신 호스트 또한 5개로 구성된다. 각각을 s1~s5, d1~d5라고 부르기로 한다. 호스트에 따라 다른 변수 값은 RTT뿐이다. s1과 d1간의 RTT는 80ms이고 s2와 d2간은 160ms, s3와 d3간은 240ms, s4와 d4간은 320ms, s5와 d5간은 80ms이다. s1~s4는 1M을 계약한 AS 플로우며 s5는 BE 플로우이다.

DiffServ의 문제점을 잘 나타나게 하기 위하여 s5에 가장 작은 RTT인 80ms를 할당한다. 또한 이는 TRC dropper가 RTT가 작은 BE에 대해 적절한 제어를 하고 있는지를 확인하기 위함이다.

4.2.2.1 여유 대역폭의 양에 따른 영향

현재까지의 연구 결과에는 DiffServ에 적용할 수 있는 대표적인 접속 제어 알고리즘이 존재 하지 않는다. 따라서 네트워크 내부의 여유 대역폭의 양과 상관없이 일관성 있는 안정된 성능을 보이는 것이 중요하다. 실제 잘 설계된 네트워크의 경우에는 보통 α가 1보다 큰 경우일 것이다. 따라서 이후의 결과들에 대해서도 이 경우에 대해 주목해서 관찰해야 할 것이다.

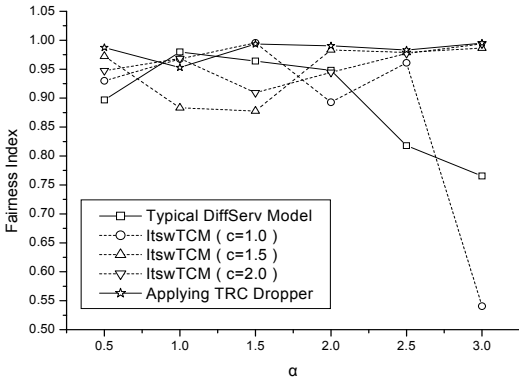

그림 3. Fairness Index

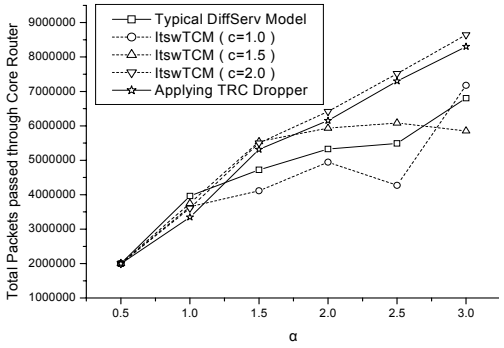

그림 4. 병목 구간의 AS 트래픽 통과량

그림 3은 α값을 변화 시켜가면서 fairness index 값을 관찰한 결과 이다. 기존의 DiffServ 방식의 경우에는 α값이 커질수록 공평성 성능이 급격히 떨어지고 있음을 확인할 수 있다. 이는 앞장에서 설명한 이유들 때문에 여유 대역폭을 공평하게 분배하여 사용하고 있지 못함을 나타내고 있다. ItswTCM 방식의 경우에는 tswTCM 방식과 달리 1이상의 값을 가지는 상수인 c값[2]에 따라서 성능을 조절하며, 이때 c값의 성능이 좋은 구간이 한정 되어 있다. c=1.0인 경우에는 값이 1.0~1.5인 범위에서 좋은 성능을 보이고 있고, c=1.5인 경우에는 α값이 2.0~3.0일 때, 그리고 c=2.0일 때는 α값이 2.5이상인 경우에 좋은 성능을 보이고 있다. 이런 결과로 볼 때 c값에 따라 공평성을 향상시킬 수 있는 범위가 한정되어 있다는 사실을 확인할 수 있다. 하지만 TRC Dropper의 경우에는 값에 상관없이 전 범위에 걸쳐서 좋은 공평성 성능을 보이고 있으며 특히 well-provisioned 네트워크에서는 매우 좋은 공평성 성능을 보이고 있다.

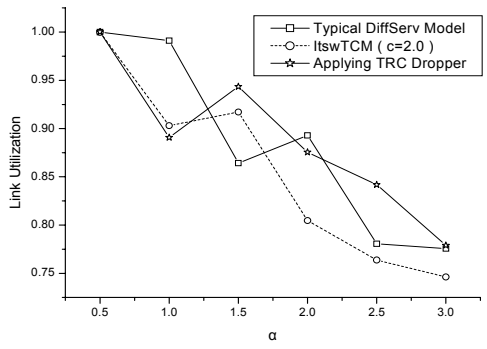

그림 5. 병목 구간의 링크 활용

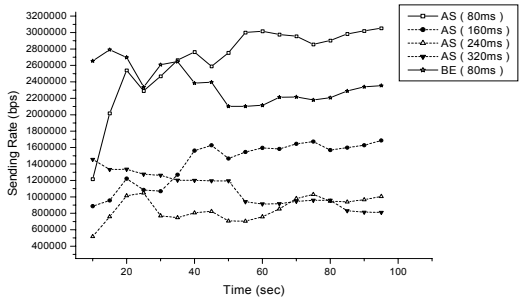

그림 6. Typical DiffServ 네트워크에서의 플로우별 전송 속도의 변화

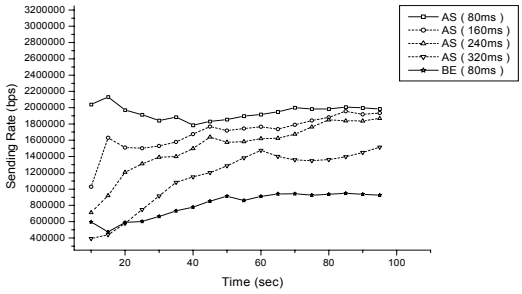

그림 7. TRC Dropper를 적용한 경우의 플로우별 전송 속도의 변화

그림 4는 병목 구간을 통과한 AS 트래픽의 양을 측정한 결과이다. 매우 근소한 차이이긴 하지만 c=2.0인 ItswTCM이 전체적으로 우수함을 알 수 있다. 하지만 이 결과에는 하나의 맹점이 있다. ItswTCM 방식의 경우에는 c값이 크면 클수록 yellow의 발생량이 높아지게 된다. 우수한 성능을 보인 c=2.0일 경우를 예를 들면 계약 속도의 2배 이하의 전송 속도를 유지할 경우에 해당 AS 플로우는 모든 패킷이 yellow로 marking될 것이다. 따라서 red color 패킷 밖에 발생이 안 되는 BE 트래픽의 경우에는 단 한 개의 패킷도 전송되 계약을 수 있게 된다. 이런 이유로 c = 2.0인 경우에는 BE의 트래픽이 네트워크를 거의 통과하지 못하고 드랍되기 때문에 당연히 AS 플로우를 위한 대역폭 공간이 넓어지게 되어 병목 구간을 통과할 AS 트래픽의 양이 조금 높게 나타난 것이다.

BE 트래픽은 계약과 보장이 없는 서비스지만 네트워크의 남은 대역폭을 사용하여 전체 네트워크 링크 활용(utilization)을 높이는 역할을 한다. 따라서 BE 트래픽이 어떠한 적절한 제어를 받아 AS 플로우를 방해하지 않으면서 링크 활용을 향상시킬 수 있어야 할 것이다. 하지만 ItswTCM의 경우는 AS 플로우의 네트워크 점유율이 어느 수준을 넘을 경우에만 BE 트래픽을 통과시키기 때문에 전체적인 링크 활용이 떨어지게 되는 것이다. 이는 그림 5의 결과에서 나타난다.

그림 5는 네트워크 내부 병목 구간의 링크 활용을 도시한 결과이다. Well-provisioned 네트워크에서 TRC Dropper가 5~10% 정도 우수한 성능을 보이고 있다. 이는 앞에서 설명한 이유에 따른 것으로 생각할 수 있다.

그림 6은 일반적인 DiffServ 네트워크에서의 전송 속도의 변화이며, 그림 7은 TRC Dropper를 적용한 경우의 전송 속도의 변화를 관찰한 결과이다. α=3.0인 경우에 실험한 결과이며 초기의 10초는 네트워크가 안정되기 이전의 시간으로 간주하고 10초 이후의 결과를 도시하였다.

일반적으로 TCP의 전송 속도는 송신 호스트의 윈도우 크기에 의해서 결정된다. 따라서 TCP의 전송 속도는 송신 호스트의 윈도우 크기와 거의 비슷한 값을 가지게 된다. 일반적인 DiffServ의 경우에는 시간이 지남에 따라 전송 속도가 수렴되는 모습을 전혀 찾아 볼 수 없다. 같은 계약 수준을 가지고 있는 AS 플로우 간의 전송속도의 폭이 상당히 큰 상태가 지속되고 있음을 확인할 수 있다. 게다가 RTT가 짧은 BE 플로우의 전송 속도가 RTT가 긴 AS 플로우의 전송속도보다 더 커지는 상황도 발생하고 있다. 이는 기존의 DiffServ는 플로우간의 상황에 따른 적절한 제어를 제공하지 못하기 때문이다.

TRC Dropper를 적용한 경우에는 전체적으로 각 AS 플로우의 전송속도의 차이가 크게 줄어있고, 시간의 흐름에 따라 차츰 수렴하고 있는 모습을 볼 수 있다. RTT가 320ms로 가장 큰 플로우의 경우에는 다른 플로우에 비해 다소 낮은 전송율을 보이고 있지만 이 또한 증가 추세에 있으며 전체적으로는 1.8Mbps 정도의 수준으로 차츰 접근하고 있다.

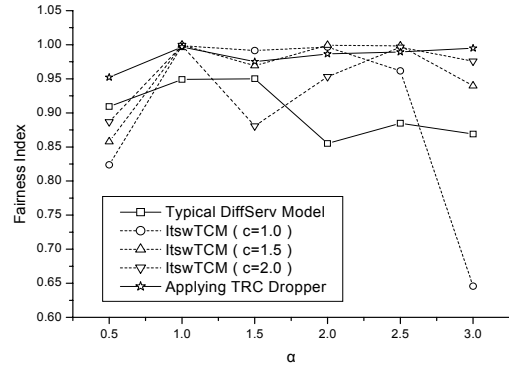

그림 8. Fairness Index

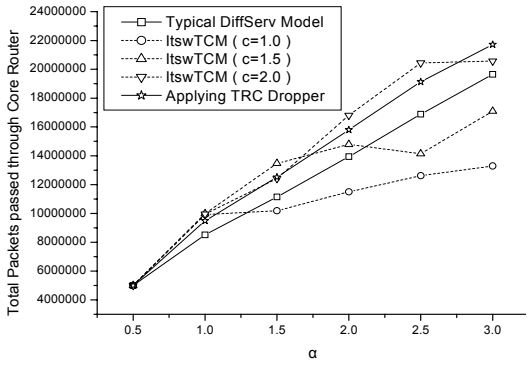

그림 9. 병목 구간의 AS 트래픽 통과량

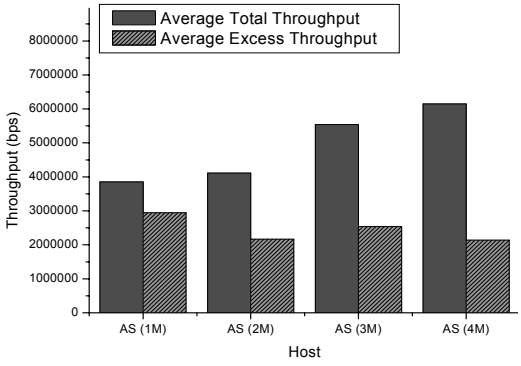

그림 10. Typical DiffServ 네트워크에서 평균 수율, 초과 수율

4.2.3 호스트별로 계약 수준이 다른 경우

4.2.3.1 여유 대역폭 양에 따른 영향

4.2.2절은 서로 다른 RTT값을 가지는 플로우간의 공평성이 개선되는지를 확인하기 위한 실험을 수행하였다. 이 절에서는 계약 수준의 차이에 따른 문제점이 해결됨을 보이려고 한다. 그림 2의 구조를 사용하였으며 s1~s4는 1M를 계약한 AS 플로우이며, s1의 계약 수준은 1M bps 이고 s2는 2M bps, s3는 3M bps, s4는 4M bps이다. s5의 경우는 BE 플로우로 계약이 존재하지 않는다.

그림 8은 네트워크 병목 구간의 대역폭을 변화에 따른 공평성 성능의 변화를 나타낸 것이다. α는 앞 절과 같은 의미를 가지고 있다. 4.2.2절의 결과와 마찬가지로 typical DiffServ 네트워크의 경우에는 α값이 큰 경우에는 공평성 성능이 떨어짐을 알 수 있다. ItswTCM의 경우에도 마찬가지로 c값에 따라 공평성 성능이 향상되는 구간이 한정되어 있음을 확인 할 수 있다. c값이 1.0일 경우에는 α값이 1.0~2.0일 때, c값이 1.5인 경우에는 α값이 2.0~2.5일 때, c값이 2.0인 경우에는 α값이 2.5보다 클 때 좋은 성능을 보이고 있다. 하나의 c값으로는 α의 범위 전체에서 좋은 성능을 보이지 못함을 알 수 있다. 이에 반해 TRC Dropper를 적용한 경우에는 α값과 거의 무관하게 안정된 공평성성능의 향상을 보이고 있다. 결과로 미루어 볼 때 TRC Dropper를 적용한 경우에는 α값이 1.0보다 큰 well-provisioned 네트워크에서는 이런 추이가 계속 유지될 것으로 추측된다.

그림 9의 병목 구간을 성공적으로 통과한 AS 트래픽의 양 역시 c값에 따라 일부 구간에서 ItswTCM이 더 우수한 성능을 보이지만 이 역시 4.2.2절에서 설명한 이유에 기인한 것이다. 따라서 전체적으로 TRC Dropper가 안정된 우수한 공평성 성능을 보인다고 할 수 있다.

계약이 다른 경우에 훌륭한 공평성 성능이 나타나는 이유는 4.2절에서 설명한 이유와 더불어 기존의 방식과 다르게 계약 수준에 따른 전송 가능한 out-of profile 패킷의 양이 다르기 때문이다.

대부분의 다른 방식에서는 out-of profile 패킷은 BE 패킷으로 취급한다. 따라서 남은 대역폭을 계약 수준과 상관없이 동등하게 나누어가지게 된다. 그림 10은 typical DiffServ 네트워크에서 α=3.0인 경우의 전체 평균 수율과 초과 수율을 도시한 결과이다. 계약 수준이 1M~4M에 이르지만 초과 수율의 양은 거의 비슷하다. 이럴 경우 그림을 보면 1M 계약을 한 플로우는 4M에 가까운 평균 수율을 얻고 있는 반면에 4배 더 높은 계약을 한 플로우의 경우에는 6M 정도의 수율만을 얻게 된다. 결과적으로 4배 더 높은 계약을 하였지만 실질적으로는 1.5배 높은 서비스를 제공 받은 셈이다.

그림 11은 TRC Dropper를 적용한 경우의 평균 수율과 초과 수율을 도시한 결과이다. 계약 수준이 높은 수록 이에 비례하게 초과 수율이 커지고 있음을 확인 할 수 있다. 이에 따라 전체 평균 수율에 있어서도 계약 수준에 비

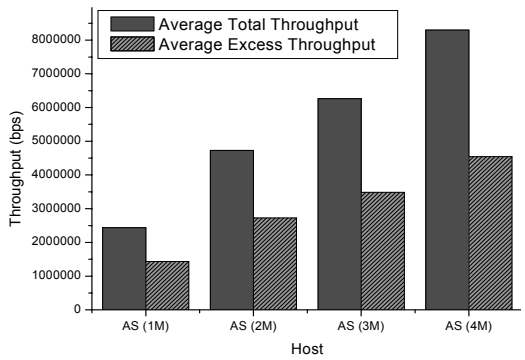

그림 11. TRC Dropper를 적용한 경우에 평균 수율, 초과 수율

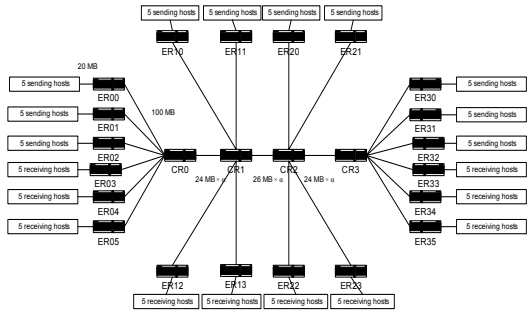

그림 12. 병목 구간이 3개인 복잡한 실험 구조

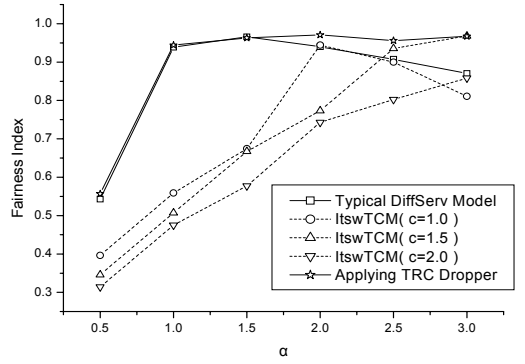

그림 13. Fairness Index

례하게 서비스를 받고 있음을 알 수 있다. 이는 profile index를 사용하였기 때문에 얻어진 결과이다. in profile 패킷에 비례하는 out-of profile 패킷을 발생시키도록 제어하기 때문에 이와 같은 결과를 얻을 수 있다. TRC Dropper를 적용할 경우에는 계약 수준이 다른 플로우간의 진정한 서비스 차등화를 이루어 내고 있다고 할 수 있다.

4.3 병목 구간의 수가 여러 개인 복잡한 구조

이번 절에서는 병목 구간이 3개이고 계약 수준도 3개인 복잡한 구조에 대한 실험을 수행한 결과를 제시한다. 4.2절의 실험은 다른 조건은 동일하게 하고 RTT와 계약 수준을 한가지씩만 바꾸어 가면서 실험한 결과이다. 이는 알려져 있는 문제점이 해결되는가를 알아보기 위한 실험이었지만, 그림 2와 같이 간단한 네트워크 구조는 실제 망에서는 찾아보기 힘들기 때문에 현실성이 떨어지는 결과들이다. 본 절에서는 RTT, 계약 수준, 병목 구간의 개수 모두가 동시에 다른 상황에서 성능이 어떤지를 확인한다. 실제 하나의 DiffServ 도메인과 유사한 구조에서 행해진 실험 이라고 할 수 있다. 앞 절에 비해 더욱 현실성 있는 결과들을 제시한다.

그림 12는 본 절에서 사용한 실험 구조이다. 총 100개의 호스트가 존재하며 이 중 50개는 송신 호스트이며 나머지 50개는 수신 호스트이다. 호스트와 에지 라우터를 연결하는 링크는 20M bps, 에지 라우터와 코어 라우터를 연결하는 링크는 100M bps의 전송 속도를 가진다. 코어 라우터를 연결하는 모든 링크는 병목 링크로 설정하였으며 각 링크는 동일한 병목 정도를 가지게 된다. 앞 절과 마찬가지로 각 링크를 통과하는 AS 트래픽들의 계약 수준의 합에 해당하는 최소 대역폭에 동일한 변수 α를 곱한 값을 사용하였다. 동일한 병목 정도를 가지지 않을 경우 그 정도가 가장 심한 병목 링크에 따라 성능이 결정되기 때문에 이런 방식을 사용하였다. 본 논문에서는 에지 라우터에서의 공평성만을 고려하였기 때문에, 플로우들이 네트워크에 들어가는 에지라우터는 동일하지만 서로 다른 경로를 거쳐서 목적지 혹은 출력 에지라우터에 전달되는 경우에 대해서는 설정하지 않았지만 향후 이를 고려하여 성능을 보완할 계획이다.

4.3.1 여유 대역폭 양에 따른 영향

네트워크 구조가 복잡해질수록 도메인을 통과하는 플로우의 수가 증가하고 이에 따라 네트워크 내부의 여유 대역폭 상황은 시시각각 다양하게 변화하게 된다. 따라서 여유 대역폭의 크기와 상관없는 안정된 성능이 4.2절의 경우보다 더욱 중요하게 된다.

그림 13은 복잡한 구조에서 50개의 송신 호스트들의 1M당 평균 수율에 대한 fairness index값을 나타낸 것이다. 그래프에서 쉽게 알 수 있듯이 모든 범위에 걸쳐 TRC Dropper를 적용한 경우에 최고의 공평성 성능을 보이고 있다. typical DiffServ 네트워크의 경우에는 낮은 provisioning rate를 가지는 네트워크에서 좋은 성능을 보이

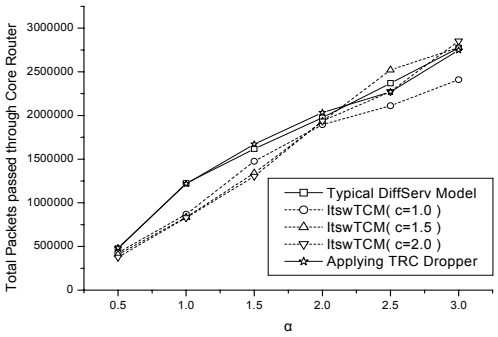

그림 14. 병목 구간의 AS 트래픽 통과량

지만, provisioning rate가 증가할수록 감소 추세를 보이고 있다. ItswTCM의 경우 c=1.0일 때는 α값이 2.0인 경우에, c=1.5인 경우에는 α값이 2.5인 경우에 좋은 성능을 보이고 있다. c=2.0인 경우에는 3.0이상의 매우 높은 provisioning rate를 가지는 네트워크에서 좋은 성능을 보일 것으로 추측된다. ItswTCM의 경우에는 5.2절의 결과와 마찬가지로 c값에 따라 특정한 구간에서만 좋은 공평성 성능을 보이고 있다. 이에 반해 TRC Dropper의 경우에는 α값이 1.0보다 큰 일반적인 경우에는 fairness index 값이 모두 0.95보다 더 큰 값을 가지고 있다. 즉, α값에 상관없이 매우 훌륭한 공평성 성능을 보이고 있다고 할 수 있다.

그림 14는 3개의 병목 구간을 성공적으로 통과한 AS 트래픽의 양을 도시한 결과이다. TRC Dropper를 적용한 경우가 전 범위에 걸쳐 매우 우수한 성능을 보이고 있다. α값이 2.5경우에 c=1.5인 ItswTCM 방식이 좀 더 우수한 성능을 보이지만 이 역시 4.2절에서 설명한 이유에 기인한다. 따라서 TRC Dropper는 AS 플로우 통과량 측면에 있어서도 매우 우수한 성능을 보이고 있음을 알 수 있다.

지금까지의 결과를 볼 때, TRC Dropper를 적용한 경우에 typical DiffServ 네트워크보다는 월등히 우수한 성능을 보임을 알 수 있었다. 그리고 ItswTCM는 c값에 따라 특정 구간에서만 좋은 성능을 보이는 반면 TRC Dropper를 적용한 경우에는 여유 대역폭의 양과 무관하게 일정하게 좋은 성능을 보이고 있음을 확인할 수 있었다. 4장에서 설명한 TRC Dropper 알고리즘은 공평성 성능을 향상시킴을 실험으로서 증명하였다.

5. 결 론

본 논문에서는 DiffServ 네트워크에서 평균 수율의 공평성 향상을 목적으로 하는 에지 라우터의 기능 블록인 TRC Dropper를 제안하였다.

가장 대표적인 transport layer protocol인 TCP는 패킷이 성공적으로 전송되면 전송 속도를 증가시키고, 패킷 드랍이 발생하면 전송속도를 감소시킨다. 이 점에 착안하여 대역폭 점유율이 높은 플로우에게 강제적인 패킷 드랍을 수행하여 그들의 전송속도를 떨어뜨린 후에 이에 따라 발생하는 네트워크 내부의 여유 대역폭을 다른 대역폭 점유율이 낮았던 플로우들이 차지할 수 있도록 함으로써 플로우간의 평균 수율에 있어서 공평성이 향상되도록 하였다.

이 방식의 가장 큰 장점은 네트워크 내부의 여유 대역폭의 양과 무관하게 안정된 성능 향상을 보인다는 점이다. 현재 접속 제어에 대한 많은 연구가 이루어지지 않았기 때문에 네트워크 내부에 어느 정도의 여유 대역폭이 있는지를 예상할 수 없으므로 이는 매우 주목할 만한 부분이다. 또한 TCP와 네트워크 내부 라우터에 대한 수정이 전혀 필요치 않기 때문에 기존의 네트워크에 적용이 쉬우며 확장성에 대한 문제점도 없다.

현재 제안한 알고리즘은 한 도메인의 에지 라우터를 통과하는 플로우들은 전송 조건이 좋은 플로우와 나쁜 플로우가 골고루 섞여 있다는 가정에서 출발하였다. 따라서 같은 에지 라우터를 통과하고 있는 플로우들 간의 상황만을 고려하여 개별 플로우의 전송 속도를 제어하더라도 1.5배 이상의 성능 향상을 얻을 수 있었다. 하지만 하나의 에지 라우터에는 전송 조건이 좋은 플로우만 존재하고, 다른 에지 라우터에는 전송 조건이 나쁜 플로우만 존재할 경우에는 성능 저하가 예상된다. 그러나 실제 네트워크에서 이런 상황이 발생하는 경우는 거의 없을 것으로 예상된다.

추후의 연구에는 이 부분까지 보완을 위해 중앙 통제 개체인 BB를 도입한 방식을 고려하고 있다. 물론 확장성 문제를 일으키지 않는 방향의 접근이 필요할 것이다. TRC Dropper를 이용하여 하나의 에지 라우터 내의 플로우간의 밸런스를 유지하고 적절한 signaling을 이용하여 각 에지 라우터간에 밸런스를 유지 시킬 수 있다면 더욱 안정되고 훌륭한 공평성 성능을 보일 수 있을 것으로 예상된다.

참 고 문 헌

1. Yeom, I., Narasimha Reddy, A.L. “Realizing throughput guarantees in a differentiated services network,” Proceedings of IEEE International Conference on Multimedia Computing and Systems, Vol. 2, pp. 372-376, 1999.
2. Hongjun Su, Atiquzzaman, M. “ItswTCM: a new aggre-

gate marker to improve fairness in DiffServ," Proceedings of GLOBECOM '01, Vol. 3, pp. 1841-1846, 2001.
3. Wu-chang Feng, Kandlur, D.D., Saha, D., Shin, K.G., "Adaptive packet marking for providing differentiated services in the Internet," Proceedings of Sixth International Conference on Network Protocols, pp. 108-117, 1998.
4. Saad E.M., El-Ghandour O.M., Jehan M.K., "Evaluation of QoS in UMTS backbone network using differentiated services," Proceedings of NRSC 2008, Vol. 2, pp. 1-11, 2008.
5. Sang, A.; Zhu, H.; Li, S.-Q. "Weighted fairness guarantee for scalable DiffServ assured forwarding," Proceedings of ICC 2001, Vol. 8, pp. 2365-2369, 2001.
6. I. Stoica, S. Shenker, and H, Zhang, "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocation in High Speed Networks," Proceedings of ACM SigComm 98, Aug. 1998.
7. Seungchul Park, "DiffServ Quality of Service Support for Multimedia Applications in Broadband Access Networks," Proceedings of ICHIT'06 vol. 2, pp. 513-518, 2006.
8. Yeom, I., Reddy, A.L.N., "Modeling TCP behavior in a differentiated services network," IEEE/ACM Transactions on Networking, Vol. 9, No. 1, pp. 31-46, Feb. 2001.
9. Floyd, S., Jacobson, V. "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, Aug. 1993.
10. Clark, D.D., Wenjia Fang, "Explicit allocation of best-effort packet delivery service," IEEE/ACM Transactions on Networking, Vol. 6, No. 4, pp. 362-373, Aug. 1998.
11. Lopes, N.V., Nicolau, M.J., Santos, A., "Efficiency of PRI and WRR DiffServ Scheduling Mechanisms for Real-Time Services on UMTS Environment," Proceedings of NTMS '08, pp. 1-5, 2008.
12. D. Grossman, "New Terminology and Clarifications for Diffserv," RFC3260, April 2002.
13. B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis. "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC3246, March 2002.
14. Kyeong Hur, "Performance Analysis of Random Early Dropping Effect at an Edge Router for TCP Fairness of DiffServ Assured Service," 한국통신학회논문지 Vol. 31, No. 4B, pp. 255-269, 2006년 4월.
15. Adegboyega Abiola, Lambadaris Ioannis, "Fairness Guarantees and Achievable QoS in Differentiated Services," Proceedings of MILCOM2007, pp. 1-7, Oct. 2007.
16. W. Fang, N. Seddigh, B. Nandy, "A Time Sliding Window Three Color Marker (TSWTCM)," RFC2859, June 2000.
17. Hayder Natiq, Zuriati Ahmed, Mohamed Othman, Shamala Subramaniam. "The TCPBased New AIMD Congestion Control Algorithm", International Journal of Computer Science and Network Security, Vol. 8 No. 10, 2008, 331-338.
18. Hanal Abuzanat, Benoit Trouillet, Armand Toguyeni, "Routing Fairness Model for QoS Optimization in Wireless Network," Proceedings of SENSORCOMM 2008, pp. 776-781, 2008.
19. M. A. Elshaikh, M. Othman, S. Shamala and J. Desa, "A New Fair Weighted Fair Queuing Scheduling Algorithm in Differentiated Services Network," International Journal of Computer Science and Network Security, Vol. 6, No. 11, pp. 267-271, Nov. 2006.
20. W. Feng, D. Kandlur, D.Saha and K. shin, "Understanding and Improving TCP Performance over Networks with Minimum Rate Guarantees," IEEE/ACM Transactions on Networking, Vol. 7, No. 2, pp. 173-187, April 1999.
21. Hill, R.; Kung, H.T. , "A DiffServ enhanced admission control scheme", Proceedings of GLOBECOM '01, Vol. 4, pp. 2549-2555, 2001.
22. UCB, LBNL, VINT Network Simulator ns2, http://www.isi.edu/nsnam/ns/

김 훈 기 (kimhk@dongyang.ac.kr)

1988 한양대학교 전자공학과 공학사
1990 한양대학교 전자공학과 공학석사
2002 한양대학교 전자공학과 공학박사
1990～2001 (주)LG전자 정보통신연구소 선임연구원
2001～현재 동양공업전문대학 전산정보학부 부교수

관심분야 : 임베디드 시스템, 센서 네트워크, 로봇 S/W, 통신 시스템

홍 성 화 (shhong@dongyang.ac.kr)

1996 고려대학교 컴퓨터학과 이학사
2002 한국항공대학교 정보통신공학과 공학석사
2008 고려대학교 전자컴퓨터공학과 공학박사
1997～2000 (주)동원시스템즈 연구원
2009～현재 동양공업전문대학 전산정보학부 전임강사

관심분야 : 유비쿼터스 네트워크, 홈 네트워크, 무선네트워크, 임베디드 시스템