인터넷 차별화 서비스를 위한 라우터의 공평성 향상 알고리즘

(Router Algorithms for Improving Fairness in Differentiated Services)

남 동 호 † 최 영 수 †† 김 병 철 ††† 조 유 제 ††††

(Dong-Ho Nam) (Young-Soo Choi) (Byung-Chul Kim) (You-Ze Cho)

요 약 본 논문에서는 차별화 서비스의 AS(Assured Service)의 문제점으로 지적되는 플로우 사이의 공평성 향상을 위한 개선된 버퍼 관리 방식과 미터를 제안하였다. 코어 라우터 (core router)를 위해 제안된 버퍼 관리 방식은 카운터를 이용하여 패킷 폐기 사이의 거리를 일정하게 유지시켜 TCP 플로우의 급격한 성능 저하와 global synchronization 문제를 해결하였다. 그리고, 경계 라우터 (edge router)를 위해 제안된 미터는 기본적으로 TSW(Time Sliding Window) 알고리즘을 고려하여, 공평성 향상을 위해 비례적 마킹과 가변적 윈도우의 개념을 도입하였다. 제안된 방식들은 기존 방식들에 비해 복잡한 연산이 필요하지 않으며, TCP 프로토콜의 변경 없이 공평성을 향상시킬 수 있는 방식이다. 그리고, 시뮬레이션을 통하여 기존의 RIO(RED with IN and OUT), TSW 방식과 제안한 방식 간의 성능 분석을 수행하였다. 시뮬레이션 결과 제안된 버퍼 관리 방식과 미터는 송신측의 목표 전송률과 근접한 수율을 제공하며, 플로우들 사이에 공평한 대역 사용을 제공하여 기존 방식보다 우수한 성능을 제공함을 보였다.

키워드 : 차별화 서비스, 서비스 품질, 공평성, 버퍼 관리, 트래픽 조절자

Abstract The IETF Differentiated Services (DiffServ) WG focused on providing service differentiation on the Internet. One problem of the DiffServ Assured Services (AS) architecture is that it cannot guarantee fairness and throughput assurance. In this paper, we propose two schemes for guaranteeing fairness among the various target rates in the AS architecture. One is a variant of RED with IN and OUT (RIO), called the improved RIO (IRIO). The other is a variant of Time Sliding Window (TSW), called the improved TSW (ITSW). To validate the proposed schemes, their behaviors are then examined under various simulation environments. The simulation results showed that IRIO and ITSW improved fairness and the throughput assurance in the AS architecture.

Key words : Differentiated Services, Quality of Service, fairness, buffer management, traffic conditioner

1. 서 론

최근 인터넷의 급속한 성장과 더불어 이를 사용하는 사람들의 요구도 확대되고 있다. 인터넷 이용자들은 현재까지의 단순한 데이타 전송뿐만 아니라 화상 회의, 오디오, 비디오 등과 같은 다양한 실시간 멀티미디어 응용들이 인터넷에서 지원되기를 요구하고 있다. 위와 같은 다양한 서비스 품질 요구는 인터넷이 현재 제공하는 최선형 서비스(best-effort service) 환경을 변화시키는 기술을 필요로 한다.

IETF에서는 현재 인터넷에서 서비스 품질을 보장하기 위한 방안으로 인터넷 통합 서비스(IntServ : Integrated Services)와 인터넷 차별화 서비스(DiffServ : Differentiated Services)에 대한 표준화 작업을 진행하고 있다[1,2]. 통합 서비스는 자원 예약 신호 프로토콜인 RSVP(Resource reSerVation Protocol)를 처음 제안할 당시에는 QoS(Quality of Service)를 훌륭히 보장할 수 있다는 점에서 많은 주목을 받았다. 하지만 통

· 본 연구는 ITRC 연구 지원으로 수행되었음.

† 비 회 원 : 삼성전자 연구원
dhnam74@orgio.net
†† 비 회 원 : 경북대학교 전자공학과
yschoi@m80.knu.ac.kr
††† 비 회 원 : NIST 객원 연구원
bckim@palgong.knu.ac.kr
†††† 종신회원 : 경북대학교 전자전기컴퓨터학부 교수
yzcho@ee.knu.ac.kr

논문접수 : 2001년 3월 7일
심사완료 : 2002년 5월 9일

합 서비스는 자원 예약 시 필요한 정보 및 상태를 모든 플로우 별로 유지해야 하는 확장성의 문제와 모든 응용 프로그램에서 RSVP를 지원해야 하기 때문에 기존의 응용 프로그램들과 호환성의 문제가 존재한다. 이에 따라서 IETF의 DiffServ WG(Working Group)에서는 통합 서비스와 같이 확실한 서비스 품질을 보장할 수는 없으나 최선형 서비스와는 구별된 더 나은 서비스 품질을 제공하는 차별화 서비스에 대한 연구가 진행 중이다. 차별화 서비스는 흐름을 각각의 플로우별로 관리하지 않고, 플로우들을 클래스별로 분류해서 라우터에서 처리함으로써 확장성의 문제를 해결하였다. 즉, 차별화 서비스는 다양한 서비스 품질의 요구를 차별화된 전달 방식으로 정의되는 PHB(Per Hop Behavior)에 따라 집합화(aggregation)하여 통합 서비스 모델보다 단순하고 실현이 용이하게 QoS를 제공하는 기술이다[1,3].

차별화 서비스는 확장성의 문제를 해결하기 위해 패킷을 클래스 별로 구분하고 클래스에 따른 PHB를 이용하여 사용자마다 다른 클래스의 QoS를 보장한다[3,4]. 차별화 서비스는 트래픽을 몇 개의 클래스로 나누어 동일한 클래스는 동일한 처리를 받도록 하며, 이를 PHB라고 정의한다. PHB는 다른 PHB들보다 상대적으로 버퍼나 대역폭 같은 자원에 우선 순위를 갖는 것으로 정의될 수 있고, 지연이나 손실 등의 트래픽 특성으로 정의 될 수도 있다. 경계 라우터는 외부에서 입력된 패킷들을 분류(classifying), 미터링(metering), 쉐이핑(shaping)과 폐기(dropping) 등을 수행하고, 코어 라우터는 차별화 서비스 망의 입구로부터 받아들인 패킷에 대해 각 PHB에 해당하는 클래스로 분류를 한 후, 출력 버퍼에서 간단한 버퍼 관리 방식인 RIO를 이용한 패킷 폐기를 수행하여 QoS 보장을 지원한다.

최근 이러한 차별화 서비스 요소 기술들의 조합으로 AS(Assured Service)에 관한 연구가 활발히 진행되고 있다[5-8]. AS는 전송 계층으로 TCP를 사용하는 플로우를 대상으로 하는 서비스로, 각 플로우는 사전에 설정된 목표 전송율을 보장 받으며 여분의 대역은 플로우간에 공평하게 사용한다. 하지만, 현재 차별화 서비스 구조에서 AS는 각 플로우간의 공평성이 저하되는 문제가 존재한다. 이에 대해, [5]에서는 현재 차별화 서비스 구조로는 각각의 플로우에 대해 공평성을 제공할 수 없다는 문제점을 제기하였고 [6]에서는 AS가 전송률과 공평성을 보장할 수 없는 원인을 분석하고, 공평성 향상을 위한 3가지 알고리즘을 제시하였다. 또한, [7]에서는 OUT 패킷을 제한하는 방법과 라우터의 패킷 폐기 방식을 수정하고 two-window TCP 등을 이용하여 공평성을 보장하는 기법을 제안하였다. 그리고 [8]에서는 토큰 버킷 미터의 문제점을 지적하고, TCP에서 군집적 패킷 폐기를 방지하는 알고리즘을 제안하였다. 차별화 서비스에서 공평성의 문제를 해결하기 위하여 제안된 위와 같은 방식들은 복잡한 연산이 요구되거나[6] 기존의 TCP 프로토콜을 수정해야 하는 문제점을 가지고 있다는 점이다[7,9].

본 논문에서는 차별화 서비스의 AS를 위한 코어 라우터의 버퍼 관리 방식인 RIO(RED with In and Out)와 경계 라우터의 미터링 방안의 일종인 TSW(Time Sliding Window)를 개선한 방식을 제안한다. 제안된 RIO는 군집적 패킷 폐기(burst packet loss)를 방지하여 TCP 플로우의 급격한 성능 저하와 플로우 간의 global synchronization을 방지한다. 그리고 제안된 TSW는 목표 전송률을 고려하는 비례적 마킹과 윈도우의 길이를 목표 전송률에 따라 가변 시키는 가변적 윈도우를 통하여 공평성을 향상시킨다.

본 논문은 다음과 같이 구성되어 있다. 제2장에서는 기존 RIO의 문제점을 분석하고, 제안된 RIO 버퍼 관리 방식을 설명한다. 제3장에서는 기존의 TSW 기법과 공평성의 향상시키기 위해 제안된 TSW의 구현 방식을 살펴 본다. 제4장에서는 시뮬레이션을 통하여 제안된 RIO와 TSW의 성능을 분석한다. 마지막으로 제5장에서는 결론을 맺는다.

2. 제안된 버퍼 관리 방식

차별화 서비스 모델의 코어 라우터에서 사용되는 대표적인 버퍼 관리 방식으로는 RIO가 있다. RIO는 RED를 기반으로 하여 코어 라우터로 입력된 IN 패킷과 OUT 패킷에 대한 차별화된 패킷 폐기 확률을 가지는 RED의 확장된 형태의 버퍼 관리 기법이다. 본 장에서는 RIO의 군집적 패킷 폐기 특성에 대해 살펴보고, 군집적 패킷 폐기가 플로우들에게 미치는 영향과 원인을 살펴본다[10,11]. 그리고, 이러한 문제점을 해결하기 위하여 개선된 RIO를 제안한다. 본 논문에서는 편의상 제안된 RIO를 IRIO(Improved RIO)라 한다[12].

2.1 기존 버퍼 관리 방식의 문제점

라우터에서 버퍼 오버플로우가 발생하는 경우 입력되는 모든 패킷은 패킷 폐기가 일어나면서, TCP 플로우 간의 global synchronization이 발생한다. RED는 라우터에서 버퍼 오버플로우가 일어나기 전에 패킷을 폐기함으로써 플로우 간의 global synchronization을 방지하기 위해서 제안되었다[10,13]. RED는 버퍼에서 버퍼의 평균 길이가 특정 임계치를 넘게 되면 버퍼의 평균길이

에 의해 결정되어지는 패킷 폐기 확률에 따라 확률적으로 패킷을 폐기한다.

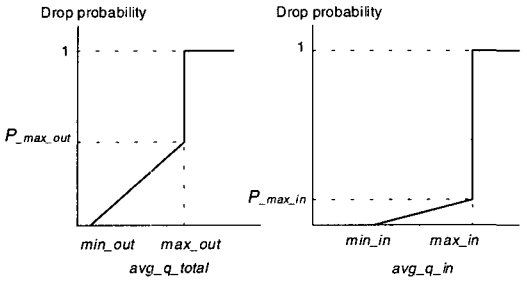


그림 1 RIO의 동작

RIO는 RED를 확장한 방식으로 코어 라우터에서 IN 패킷과 OUT 패킷에 대해 차별화된 RED 알고리즘을 적용한다. 그림 1은 IN 패킷과 OUT 패킷에 대한 차별화 된 동작을 보여준다. 그림과 같이 OUT 패킷은 IN 패킷 보다 빨리 폐기가 일어나며, OUT 패킷에 대한 패킷 폐기 확률은 IN 패킷에 대한 패킷 폐기 확률보다 높다. OUT 패킷은 트래픽 프로파일의 계약을 위반하는 패킷이므로 코어 라우터에서는 IN 패킷보다 빨리 패킷 폐기가 발생하며, 훨씬 높은 패킷 폐기 확률을 가진다. 코어 라우터에서는 패킷이 도착하여 출력 링크의 버퍼에 저장되는 순간 패킷이 IN 패킷인지 OUT 패킷인지를 판별한다. 패킷이 IN 패킷이라면 IN 패킷에 대한 평균 큐 길이(*avg_q_in*)를 이용하여 IN 패킷 폐기 확률을 계산하고, OUT 패킷이 도착하면 IN 패킷과 OUT 패킷의 큐 길이를 합한 전체 길이(*avg_q_total*)를 이용하여 OUT 패킷의 패킷 폐기 확률을 계산한다. 일반적으로 RIO에서 *P_max_in*의 값은 0.002를, *P_max_out*의 값은 0.2 정도의 값을 가지며, OUT으로 마킹된 패킷은 군집적으로 폐기 될 수 있다. 이러한 군집적인 패킷 폐기는 global synchronization 문제를 유발시킬 수 있으며, 또한 특정 플로우는 연속적인 패킷 폐기를 겪을 수 있으며, 급격한 수율 저하를 가질 수 있는 문제점을 가지고 있다.

2.2 제안된 버퍼관리 방식

본 논문에서는 앞에서 제기한 문제점을 해결하기 위해 개선된 IRIO 알고리즘을 제안한다. IRIO에서 패킷의 폐기는 단순히 평균 큐 길이에 의존적인 *P_in* 혹은 *P_out* 뿐만 아니라 이전에 폐기된 패킷과의 간격을 고려한다. 그림 2는 제안된 IRIO의 의사 코드를 보여준다. IRIO는 일단 도착한 패킷이 IN 패킷인지 OUT 패킷인지를 판단하여 평균 큐 길이(*avg_q_in*과 *avg_q_total*)를 계산하고, 이에 따른 패킷 폐기 확률(*P_in* 혹은 *P_out*)을 계산한다. 만약, 카운터(*count_in*과 *count_out*)가 패킷 폐기 확률의 역수(1/*P_in* 혹은 1/*P_out*)를 넘는다면 입력된 패킷을 폐기한다. 여기서 카운터는 폐기된 패킷 사이의 간격을 나타내며, 패킷 폐기가 일어나면 0으로 초기화 된다. 의사 코드에서와 같이 IRIO에서 패킷 폐기 확률은 평균 큐 길이 뿐만 아니라 카운트 변수 값에 의존적이다. 평균 큐 길이가 일정하다고 가정하였을 때 RIO와 IRIO 방식은 장기적으로 거의 동일한 수의 패킷을 폐기하지만 IRIO는 군집적 폐기를 방지하는 차이점을 가지고 있다. RIO 알고리즘은 폐킷의 폐기 결정이 단순히 평균 큐 길이(*avg_q_in*과 *avg_q_total*)에 의해 이루어지나, IRIO 알고리즘은 군집적 패킷 폐기를 방지함으로써, 하나의 TCP 플로우에서 연속된 패킷 폐기로 인한 급격한 성능 저하와 TCP 플로우 간의 global synchronization의 문제를 해결할 수 있다. 4장에서는 IRIO와 RIO의 평균 큐 길이 및 군집적 패킷 폐기로 인한 TCP 플로우의 수율 저하 개선에 대한 성능 분석을 수행한다.

```
For each packet arrival
  if it is IN packet
     calculate the IN average queue size avg_q_in
  else calculate the total average queue size avg_q_total
If it is IN packet
  if ( max_in > avg_q_in ≥ min_in )
     increment count_in
     P_in← P_max_in × (avg_q_in - min_in)/(max_in - min_in)
     if ( count_in > 1/ P_in )
          drop packet
          count_in←0
  else if ( avg_q_in ≥ max_in )
     drop packet
     count_in←0
  else count_in←-1
If it is OUT packet
  if ( max_out > avg_q_total ≥ min_out )
     increment count_out
     P_out←P_max_out × (avg_q_total - min_out)/(max_out - min_out)
     if ( count_out > 1/ P_out )
          drop packet
          count_out←0
  else if ( avg_q_total ≥ max_out )
     drop packet
     count_out←0
  else count_out←-1
```

그림 2 IRIO 알고리즘

3. 공평성 향상을 위한 제안된 TSW

현재 차별화 서비스에서 경계 라우터에서는 미터 방식으로는 토큰 버킷을 이용하는 방식과 평균 전송률을 측정하여 마킹하는 TSW 방식이 있다. 토큰 버킷을 이용할 경우 버킷에 토큰이 없을 경우, 입력되는 패킷은 OUT으로 마킹 되는데, 토큰을 모두 사용하였을 경우 도착한 패킷들은 연속적으로 OUT으로 마킹되며, 이는

코어 라우터에서 군집적인 패킷 폐기를 유발하는 문제점을 가지고 있다. 따라서, 본 논문에서는 미터링 방안으로 TSW를 고려한다[8,14,15]. 본 장에서는 기존 TSW의 문제점을 살펴보고, 이 문제를 해결하기 위한 두 가지 개선된 TSW 알고리즘을 제안한다. 본 논문에서는 편의상 이들 두 가지 알고리즘을 결합한 방식을 ITSW(Improved TSW)라 한다.

3.1 기존 TSW의 문제점

TSW는 전송률 측정기와 마커로 나누어 질 수 있다. 전송률 측정기는 플로우의 평균 전송률을 측정하고, 마커는 패킷에 IN혹은 OUT을 마킹한다. 그림 3은 기존 TSW의 전송률 측정 알고리즘을 보여준다. 알고리즘은 새로운 패킷이 도착하면, 현재 패킷이 도착한 시간과 가장 최근에 도착한 패킷의 시간차와 도착한 패킷의 크기를 이용한 순간적인 전송률과 윈도우의 길이에 해당하는 이전 전송률을 moving average형식으로 평균 전송률을 계산한다. 이 알고리즘에서 *avg_rate*은 평균 전송률을 나타내고, *t_front*는 가장 최근에 도착한 패킷의 도착 시간을 나타내며, *now*는 현재 패킷의 도착 시간이고, *win_length*는 윈도우의 길이로 history의 크기를 나타낸다.

```
Initially;
      win_length = a constant;
      avg_rate   = connection's target rate (target_rate);
      t_front    = 0;
Upon each packet arrival, TSW updates its state variable
      bytes_in_TSW = avg_rate × win_length;
      new_bytes    = bytes_in_TSW + pkt_size;
      avg_rate     = new_bytes/(now - t_front + win_length);
      t_front      = now;
Whereas, now is the time of current packet arrival,
and pkt_size is the packet size of the arriving packet
```

그림 3 TSW의 전송률 측정 알고리즘

마킹 알고리즘은 일단 평균 전송률을 구한 다음, 평균 전송률이 목표 전송률보다 낮다면 입력되는 모든 패킷들에 대해 IN을 마킹하고, 평균 전송률이 목표 전송률을 넘으면 입력되는 패킷에 대해 아래식을 이용하여 확률적으로 OUT을 마킹한다.

$$P = \frac{avg\_rate - target\_rate}{avg\_rate} \quad (1)$$

차별화 서비스의 AS는 전송률에 기반을 둔 서비스를 지원하지만, AS는 TCP를 이용하여 전송 할 경우 TCP의 slow start에 따른 폭주 윈도우(cwnd)의 변동으로 인하여 목표 전송률에 맞게 패킷을 전송하기가 어렵다. 즉, 송신원이 TCP를 이용할 경우 목표 전송률이 높은 플로우는 자신의 목표 전송률만큼 전송하지 못하고, 목표 전송률이 낮은 플로우는 목표 전송률보다 높게 전송하는 문제가 발생하며, 이를 bandwidth skew problem이라 한다[5-7].

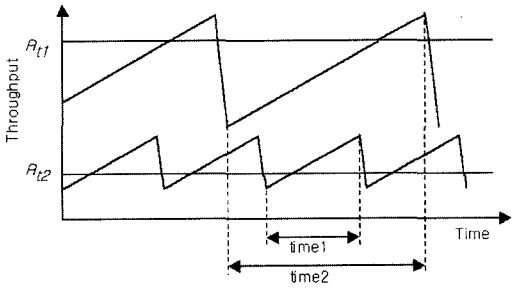


그림 4 Bandwidth skew problem의 원인

그림 4는 bandwidth skew problem이 발생하는 원인을 보여준다. 그림에서와 같이 bandwidth skew problem은 목표 전송률에 따른 TCP의 폭주 윈도우가 패킷 폐기가 일어난 후 다시 폭주 윈도우를 패킷 폐기가 일어난 크기까지 폭주 윈도우의 증가 시간이 서로 다르기 때문에 발생한다. 즉, 그림 4에 나타난 것처럼 목표 전송률 R_{t1}과 R_{t2}가 존재하고 $R_{t1} > R_{t2}$인 경우, 폭주 윈도우를 회복하는 시간은 time1 > time2가 된다. 만약 잉여 대역폭(excess bandwidth)이 존재한다면, 낮은 목표 전송률의 플로우가 높은 목표 전송률의 플로우보다 빨리 자신의 목표 전송률에 해당하는 폭주 윈도우의 크기보다 더 크게 폭주 윈도우를 증가시키며 이로 인해 낮은 목표 전송률의 플로우가 자신의 목표 전송률보다 높게 전송하는 문제가 발생하게 된다[5].

3.2 제안된 TSW

본 절에서는 제안하는 TSW는 앞에서 제기한 문제점을 해결하기 위해서 기존의 TSW 알고리즘을 개선한 ITSW를 제안한다. 비례적 마킹(proportional marking)은 마킹 알고리즘의 문제점을 수정한 방식이고, 가변적 윈도우(variable window)는 전송률 측정기의 윈도우의 길이를 수정한 방식으로, 제안된 두 가지 방식은 TSW의 서로 다른 부분을 개선한 방식으로 별개로 혹은 동시에 적용되어 질 수 있다.

3.2.1 비례적 마킹을 이용한 TSW

첫 번째로 제안하는 알고리즘은 평균 전송률과 목표 전송률의 차에 따른 비례적 마킹을 통하여 목표 전송률을 높이 초과하는 플로우에게는 평균 전송률이 목표 전

송률을 초과하는 크기에 비례하여 OUT을 마킹하는 방식이다. 제안된 방식은 평균 전송률이 목표 전송률을 넘어서면, 다음과 같은 식을 이용하여 입력되는 패킷에 OUT을 마킹한다.

$$P = \min\left\{\frac{avg\_rate - target\_rate}{target\_rate},\ 1\right\} \qquad (2)$$

그림 5는 제안한 비례적 마킹의 동작에 대한 플로우 차트를 보여준다. 이 그림에서 보듯이 패킷이 입력되면 일단 평균 전송률을 계산하고, 평균 전송률이 목표 전송률 보다 높다면 식 (2)를 이용하여 확률 P로 패킷에 OUT을 마킹한다.

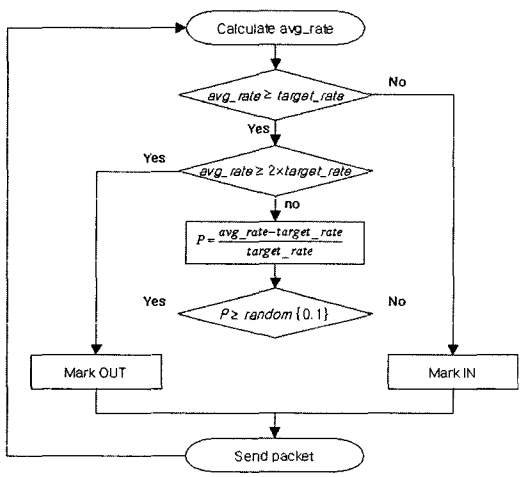


그림 5 비례적 마킹의 플로우 차트

그림 6은 기존 마킹 알고리즘의 문제점을 보여준다. 그림에 나타나듯이 기존 마킹 알고리즘은 평균 전송률이 목표 전송률을 얼마나 초과하는지 관계없이, 목표 전송률을 넘어서는 크기만큼 OUT을 마킹한다. 하지만, 목표 전송률이 낮은 플로우가 패킷 폐기로 인한 TCP의 폭주 윈도우가 감소하더라도, 폭주 윈도우가 목표 전송률만큼 증가하는 시간이 짧기 때문에 자신의 목표 전송률보다 많이 전송한다. 이러한 문제점을 해결하기 위하여, 제안된 비례적 마킹은 낮은 목표 전송률의 패킷 폐기 확률을 높이기 위해 제안되었다.

그림 7은 비례적 마킹의 예를 보여주고 있다. 그림과 같이 비례적 마킹은 평균 전송률과 목표 전송률의 비에 따라 OUT으로 마킹하여, 낮은 목표 전송률의 플로우에게 더욱 많은 OUT 마킹을 발생하도록 한다. 전체적인 OUT 패킷의 수는 많지만, 낮은 목표 전송률의 플로우가 기존 방식보다 많은 OUT 패킷으로 인해 패킷 폐기가 자주 발생하면서 목표 전송률에 근접하게 전송한다. 목표 전송률이 낮은 플로우는 기존 방식보다 수율이 낮아지고, 목표 전송률이 높은 플로우는 상대적으로 더 많은 트래픽을 전달하므로, 목표 전송률에 근접하게 전송할 수 있다.

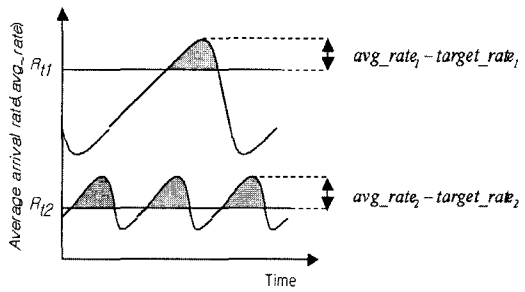


그림 6 기존 TSW의 OUT 마킹

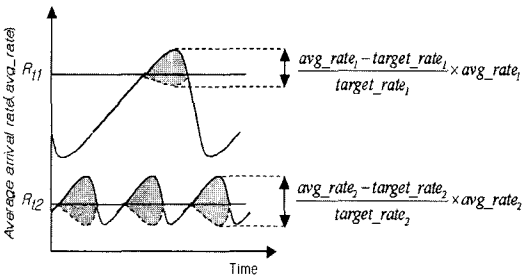


그림 7 비례적 마킹을 이용한 OUT 마킹

3.2.2 가변적인 윈도우를 이용한 TSW

두 번째로 제안하는 알고리즘은 목표 전송률에 따른 윈도우의 길이를 차별화 하여 목표 전송률에 맞게 *win_length*를 적용시키는 가변적 윈도우 방식이다. 전송률 측정기는 순간적인 전송률과 윈도우의 길이에 해당하는 이전 전송률을 moving average 형식으로 평균 전송률을 계산하고, 윈도우의 길이를 모든 플로우에 동일하게 적용 시켰다. TCP의 동작에서 패킷 폐기로 인한 수율의 감소와 다시 폭주 윈도우의 증가로 인한 수율이 증가하는 주기가 존재한다. 이러한 TCP의 주기와 변화의 크기는 목표 전송률에 따라서 달라진다. 목표 전송률이 낮은 플로우들은 주기가 짧으며, 목표 전송률이 높은 플로우들은 그 주기가 상대적으로 길다.

그림 8은 목표 전송률이 낮은 플로우들과 높은 플로우들에게 동일한 *win_length*가 적용될 경우의 문제점을 보여주고 있다. 이 그림에서 R_{t1}은 수율 변화의 주기에

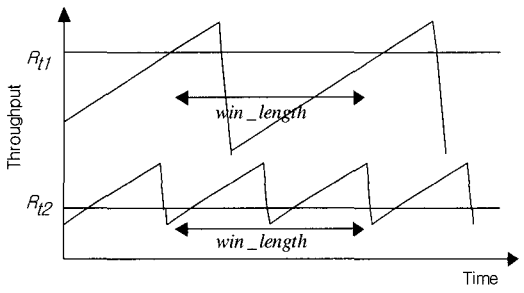


그림 8 기존 TSW의 *win_length*

비해서 상대적으로 짧은 *win_length*를 가지므로 수율의 변화에 따라 평균 전송률이 변화가 R_{t2}에 비해 상대적으로 심하게 된다. 그 결과, R_{t1}의 평균 전송률은 목표 전송률을 빨리 넘어가면서 OUT 패킷이 많이 발생된다. R_{t2}는 *win_length*가 상대적으로 길게 되므로 수율의 변화가 평균 전송률에 제대로 반영이 되지 않아서 OUT 패킷이 원래의 양보다 작게 된다.

이러한 문제점을 해결하기 위하여 제안된 가변적 윈도우 방식은 목표 전송률에 비례하여 윈도우의 길이를 가변 시킨다. 각각의 플로우에게 적당한 윈도우의 길이를 할당하기 위하여 아래식을 사용한다. 이 식에서는 모든 플로우의 목표 전송률에 대한 평균 목표 전송률을 구하고, 일정한 윈도우의 길이와 각각의 목표 전송률을 이용하여 플로우에게 적당한 윈도우의 길이를 할당한다.

$$win\_length_i = \frac{target\_rate_i}{\left(\sum_{i=1}^{n} target\_rate_i / n\right)} \times win\_length \quad (3)$$

그림 9는 가변적 윈도우 방식을 적용시킨 예를 보여준다. 이 그림에서는 가변적인 윈도우를 차별화 서비스의 미터에 적용시켜 목표 전송률에 따라 윈도우의 길이가 변하는 것을 알 수 있다.

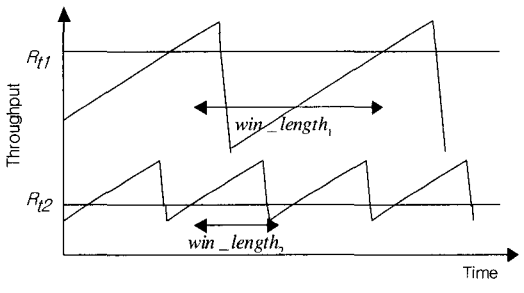


그림 9 가변적인 윈도우 방식의 *win_length*

4. 시뮬레이션을 통한 제안된 방식의 성능 분석

본 장에서는 차별화 서비스의 공평성 향상을 위해 제안된 버퍼 관리 방식과 미터에 대한 성능을 시뮬레이션을 통하여 비교 분석한다. 이를 위해 본 장에서는 먼저 시뮬레이션 모델과 성능 분석에 기준이 되는 평가 요소들을 기술하고, 다음으로 시뮬레이션 결과를 통해 성능을 비교 분석한다.

4.1 시뮬레이션 환경과 성능 평가 요소

제안된 방식들의 성능을 분석하기 위해 사용된 시뮬레이션 모델은 그림 10과 같이 10쌍의 송신원과 수신원을 가지는 one-hop 모델을 사용하며, 각 송신원은 TCP reno를 이용하여 패킷을 전송한다. 표 1은 목표 전송률이 일정하게 분포된 상태에서 플로우들 사이의 공평성을 측정하기 위해서 사용한다. 표 2는 폭주가 발생하는 R1과 R2사이 링크의 대역폭을 나타내며, 표 3은 R1과 R2에서 사용되는 RIO 파라미터를 나타낸다. 시뮬레이션은 송신원들의 목표 전송률은 고정시키고, 라우터와 송신원, 수신원 사이의 각 링크의 대역폭은 각각 10Mbps이며, RTT(Round Trip Time)는 100ms이고, R1과 R2의 버퍼크기는 200 패킷이다. 시뮬레이션 시간은 10분을 수행하였으며, 성능 측정 시간은 시뮬레이션 수행 5분 후부터 10분까지 5분 동안의 결과에 대한 성능을 측정하였다.

시뮬레이션에서는 AS의 서비스 정의에 따라 플로우들에 대한 목표 전송율 보장 및 공평한 잉여 대역 사용을 중심으로 성능 분석을 수행한다. 잉여 대역은 식 4, 5와 같이 각 플로우가 동일한 양의 대역을 사용하는 것을 공평하다고 가정한다.

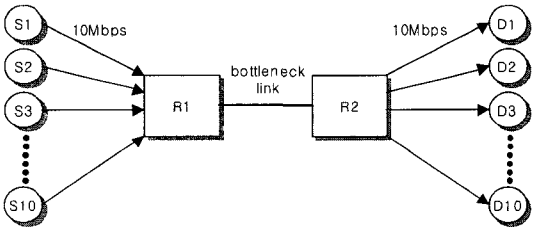


그림 10 시뮬레이션을 위한 네트워크 환경

$$e = C - \sum_{i=1}^{n} target\_rate_i \quad (4)$$

$$ideal\_rate_i = target\_rate_i + \frac{e}{n} \quad (5)$$

여기서,

- n : 송신원의 수
- $target\_rate_i$: i 번째 플로우의 목표 전송률
- C : R1과 R2사이 링크의 대역폭
- $ideal\_rate_i$: i번째 플로우가 서비스 받아야 하는 전송률을 나타낸다.

표 1 목표 전송률

Source	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Target rate [kbps]	0	200	300	400	500	600	700	800	900	1,000

표 2 R1-R2사이 링크의 대역폭

Source	*Over-subscribed* case	*Under-subscribed* case
Bandwidth [kbps]	5400	6400

표 3 RIO의 파라미터

	min_th [packets]	*max_th* [packets]	*P_max*
IN	60	100	0.02
OUT	40	80	0.2

4.2 제안된 버퍼 관리 방식에 대한 성능 분석

본 절에서는 기존 RIO가 가지고 있는 문제점을 시뮬레이션을 통해 살펴보고, IRIO와 기존 RIO의 알고리즘에 대한 성능을 시뮬레이션을 통하여 비교 분석한다. 그림 11은 RIO와 IRIO의 패킷 폐기 방식에 대한 패킷 폐기의 군집성을 보여준다. 시뮬레이션 환경은 평균 큐 길이가 일정하고, 1000개의 입력되는 패킷에 대해 0.02의 패킷 폐기 확률을 적용하였다. 그림과 같이 RIO와 IRIO는 비슷한 양의 패킷을 폐기하며 기존 RIO의 패킷 폐기는 군집적 폐기 구간이 존재하지만, IRIO에서는 평균

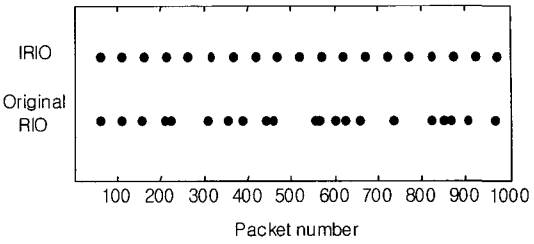
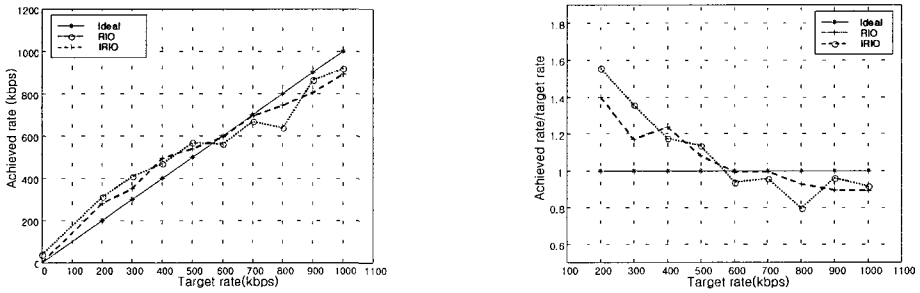


그림 11 기존 RIO와 IRIO에 대한 패킷 폐기 순서의 비교

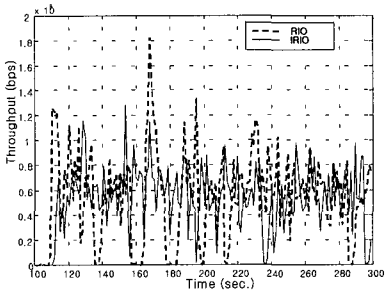
큐 길이가 일정하므로 패킷 폐기 간격이 일정하게 유지된다. RIO에서 OUT 패킷에게 사용하는 P_out은 일반적으로 IN 패킷에서 사용하는 확률보다 높은 값을 가지게 되므로, 그림 11에 나타난 것보다 심한 군집적 패킷 폐기가 발생할 수 있으며, IRIO에서는 패킷 폐기 확률이 일정할 경우 카운터에 의해 패킷 폐기의 간격을 일정하게 유지하므로, 군집적 패킷 폐기 구간이 존재하지 않는다.

군집적 패킷 폐기와 이에 따른 수율 및 공평성 저하 문제의 성능 분석을 위해 over-subscribed 환경에서 표 1과 같은 목표 전송률을 가질 때 각 TCP 플로우의 수율을 측정하였다. 그림 12에서는 각 플로우의 수율 및 공평성을 보여주며, 그림과 같이 RIO를 사용하였을 경우 S6와 S8 플로우는 목표 전송율에 미치지 못하는 낮은 수율을 보장 받는다. S6과 S8 플로우의 수율 저하 원인 분석을 위해 그림 13에서는 각 플로우의 시간에 따른 수율을 나타내었다. 그림과 같이 RIO를 사용하였을 경우 S6, S8 플로우는 수율이 급격하게 저하되는 구간이 존재하며, 이는 군집적인 패킷 폐기로 인해 발생한다[10]. IRIO는 군집적 패킷 폐기를 방지하여 TCP의 급격한 성능 저하 없이 패킷이 전달되며, 공평성이 RIO보다 향상됨을 알 수 있다. IRIO의 카운터 값은 단순히 패킷 폐기를 지연시키는 개념이 아니라 폐기되는 패킷

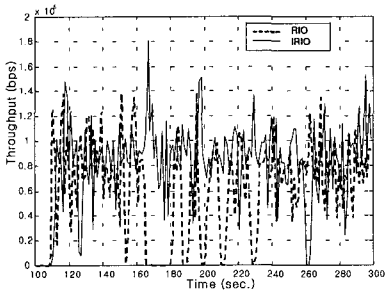


(a) 전송률 (b) 공평성

그림 12 RIO와 IRIO간의 목표 전송률에 대한 실제 전송률 및 공평성 비교



(a) S6 플로우



(b) S8 플로우

그림 13 RIO와 IRIO간의 수율 비교

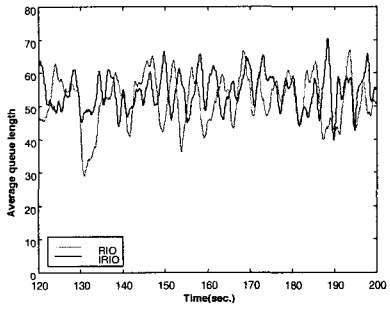


그림 14 RIO와 IRIO의 평균 큐 길이(avg_q_total) 비교

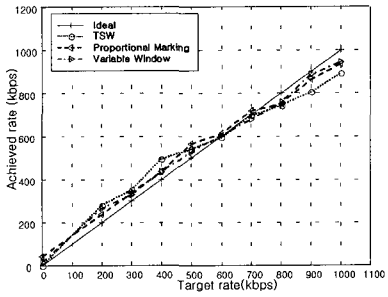
간의 간격을 보장하는 개념이다. 카운터 값에 따라 RIO에서는 폐기 될 패킷이 IRIO에서는 폐기되지 않을 수 있으나 이 경우 큐 길이 증가하며 이에 따라 패킷 폐기 확률이 증가한다. 이에 따라 장시간적으로 RIO 및 IRIO는 패킷 폐기의 수의 차이가 크지 않으며, 그림 14와 같이 유사한 큐 길이를 가진다.

4.3 제안된 TSW에 대한 성능 분석

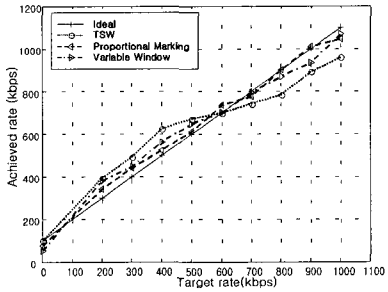
본 절에서는 제안된 비례적 마킹을 이용한 TSW와 가변적인 윈도우를 이용한 TSW 방식에 대한 성능을 시뮬레이션을 통하여 비교 분석한다.

4.3.1 제안된 방식들에 대한 성능 분석

제안된 방식들간의 성능 분석을 위해, 그림 10과 같은 망 환경에서 표 1과 같은 목표 전송율을 가지는 TCP 플로우 간의 수율 및 공평성을 측정하였다. 그리고, 망의 부하 상황에 따른 성능 분석을 위해 표 2와 같이 under, over subscribed 환경에 대해 성능을 분석하였다. Under-subscribed 환경은 1Mbps의 잉여 대역폭이 존재하며, 각 플로우는 이상적으로 100kbps의 추가 대역을 사용한다. R1과 R2의 버퍼 관리 방식으로는 군집적 패킷 폐기를 방지하기 위하여 IRIO를 이용하였다.



(a) Over-subscribed case



(b) Under-subscribed case

그림 15 망의 부하 상화에 따른 수율의 비교

그림 15에 나타나듯이 기존 TSW를 이용한 경우 bandwidth skew problem이 심하게 발생하는 것을 알 수 있다. Bandwidth skew problem의 원인은 목표 전송률이 낮은 플로우들은 패킷 폐기 발생 후 목표 전송률이 높은 플로우보다 빨리 폭주 윈도우를 회복하기 때문에 목표 전송률보다 높은 수율을 나타내는데, 이런 문제는 잉여 대역폭이 존재하는 경우 심하게 발생한다. 잉여 대역폭이 존재하는 경우 목표 전송률이 낮은 플로우가 TCP의 폭주 윈도우를 빨리 회복하면서 잉여 대역폭을 빨리 사용하게 되므로 잉여 대역폭이 발생하는 경우 bandwidth skew problem이 심하게 발생한다.

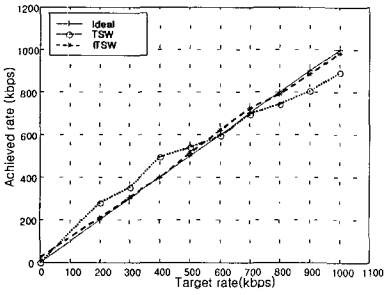
비례적 마킹 방식은 목표 전송률을 넘어서는 크기에

비례하여 플로우들의 패킷에 OUT 마킹을 수행하기 때문에, 기존 방식보다 목표 전송률이 낮은 플로우들에 대해서 훨씬 많은 패킷들에 대해서 OUT 마킹을 수행한다. 목표 전송률이 낮은 플로우는 기본 방식보다 많은 OUT 패킷을 발생시키므로 패킷 폐기가 많이 일어나면서 목표 전송률에 가깝게 전송되고, 상대적으로 목표 전송률이 높은 플로우는 기존 방식보다 트래픽을 많이 전달하므로 자신의 목표 전송률에 가깝게 전송하여 플로우들 사이의 공평성을 향상시킨다.

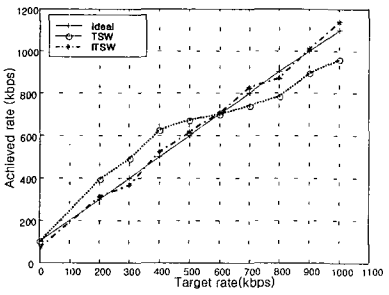
가변적인 윈도우는 목표 전송률의 크기에 따른 가변적인 윈도우 크기를 전송률 측정기에 적용한다. 목표 전송률이 낮은 플로우는 전송률 측정기에서 TCP의 변화율을 빨리 반영하여 OUT 패킷이 많이 발생하도록 하고, 목표 전송률이 높은 플로우들은 상대적으로 느리게 반영하여 OUT 패킷을 적게 발생시킨다. 가변적 윈도우는 윈도우의 길이를 변화시켜 기존의 문제점을 해결하여 각 플로우들 사이의 공평성을 향상시킨다.

4.3.2 제안된 방식들을 결합한 TSW에 대한 성능 분석

본 절에서는 시뮬레이션을 통하여 가변적인 윈도우를 이용하는 방식과 비례적 마킹을 결합한 ITSW에 대한 성능을 비교 분석한다. 시뮬레이션 환경은 앞 절과 동일하다.



(a) Over-subscribed case



(b) Under-subscribed case

그림 16 망의 부하 상화에 따른 수율의 비교

비례적 OUT 마킹은 플로우의 평균 전송률과 목표 전송률의 차에 해당하는 크기로 OUT 마킹을 비례적으로 수행하고, 가변적인 윈도우를 이용하여 목표 전송률에 따른 TCP의 변화율을 수용하도록 하였다. 그림 16은 제안한 미터 및 마커 알고리즘을 결합하여 사용함으로써, 제안한 방식이 bandwidth skew problem을 해결하고 있음을 보여준다.

각 방식에 대한 공평성의 정도를 알아보기 위해 아래 식들을 이용하여 공평성 지수(fairness index)를 계산하였다[16]. 공평성 지수는 1에 가까울수록 공평하게 전송한다는 것을 의미한다. 기존 방식과 제안된 방식에서 어느 방식이 가용 대역폭을 더 공평하게 사용하는 가를 알아 보기 위해 세 방식의 공평성 지수를 표 4에 나타내었다. 기존 TSW에서는 공평성을 제대로 보장하지 못하지만, 가변적 윈도우 방식보다는 비례적 마킹이 공평성에서는 향상되는 것을 알 수 있고, 두 가지를 결합한 방식의 공평성이 가장 나은 성능을 보여준다.

표 4 공평성 지수의 비교

	기존 TSW	가변적 윈도우	비례적 마킹	ITSW (결합한 방식)
Over-subscribed case	0.9771	0.9885	0.9935	0.9995
Under-subscribed case	0.9775	0.9911	0.9971	0.9984

5. 결 론

본 논문에서는 인터넷 차별화 서비스로 제안되고 있는 AS를 공평하게 지원하기 위한 IRIO와 ITSW를 제안하고, 시뮬레이션을 통해 성능을 비교 분석하였다. IRIO 는 코어 라우터의 버퍼 관리 방식이며, 기존 RIO의 패킷 폐기 확률 뿐만 아니라 카운터를 이용하여 폐기되는 패킷간의 간격을 고려하여 군집적 패킷 폐기가 일어나는 것을 방지하였다. 제안된 IRIO는 TCP 플로우에서 다수의 패킷 폐기와 TCP 플로우간의 global synchronization을 방지하였다. 그리고, ITSW는 비례적 마킹과 가변적 윈도우 방식을 사용함으로써, 목표전송율을 고려하여 AS에서 공평성을 향상시킨다.

시뮬레이션을 통한 성능 분석 결과를 살펴 보면 다음과 같다. 우선, IRIO 알고리즘의 경우 기존의 RIO에서 발생하는 군집적 패킷 폐기를 방지하여, 하나의 TCP 플로우에서 급격한 성능 저하 문제가 해결됨을 알 수 있었다. 다음으로, 기존 TSW의 마킹 방식은 목표 전송률이 낮은 플로우가 목표 전송률보다 높게 전송하기 때문에 공

평성의 문제가 발생하였다. 비례적 마킹은 목표 전송률을 넘어 서는 크기만큼 비례적 OUT 마킹을 통하여 목표 전송률이 낮은 플로우에게 보다 많은 OUT 패킷을 유도하여 bandwidth skew problem을 해결한다. 마지막으로, 기존 TSW의 전송률 측정기에서는 플로우에 관계없는 동일한 윈도우를 적용시키는 문제를, 목표 전송률에 따른 윈도우의 길이를 가변 시켜 목표 전송률에 따른 TCP의 변화를 전송률 측정기에 반영시켰다. 이를 통하여 각 플로우들 사이에서의 공평성 향상을 살펴 볼 수 있었다.

앞으로 좀 더 다양한 환경에서의 시뮬레이션을 통한 제안된 라우터들의 성능 분석이 필요하며, 다양한 망 상황에서의 라우터 파라미터가 성능에 미치는 영향을 분석하여 파라미터의 최적 설정에 대한 연구가 요구된다.

참 고 문 헌

[1] K. Nichols et al., "Definition of the Differentiated Services Fields(DS Field) in the IPv4 and IPv6 Header," *RFC-2474*, Dec. 1998.

[2] R. Braden et al., "Integrated Services in the Internet Architecture: an Overview," *RFC-1633*, July 1994.

[3] S. Blake et al., "An Architecture for Differentiated Services," *RFC-2475*, Dec. 1999.

[4] S. Brim et al., "Per Hop Behavior Identification Codes," *RFC-2836*, May 2000.

[5] J. Ibanez et al., "Preliminary Simulation Evaluation of an Assured Service," *draft-ietf-diffserv-assured-eval-00.txt*, Aug. 1998.

[6] W. Lin et al., "How to Make Assured Service more Assured," In *proc. ICNP '99*, pp. 182-191, Oct. 1999.

[7] I. Yeom and N. Reddy, "Realizing Throughput Guarantees in a Differentiated Services Network," In *proc. of ICMCS*, pp. 372-376, June 1999.

[8] A. Feroz et al., "TCP-Friendly Traffic Conditioners for Differentiated Services," *draft-azeem-tcpfriendly-diffserv-00.txt*, Mar. 1999.

[9] W. Feng and D. Kandlur, "Adaptive Packet Marking for Providing Differentiated Services in the Internet," *Proc. ICNP '98*, pp. 108-117, Oct. 1998.

[10] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computer Communication Review*, vol. 26 no. 4, pp. 5-21, July 1996.

[11] S. Sahu and D. Towsley, "A Quantitative Study of Differentiated Services for the Internet," *Proc. of IEEE Global Internet*, pp.1808-1817, Dec. 1999.

[12] 남동호, 김병철, 조유제, "DiffServ에서 TCP의 성능 향상을 위한 버퍼 관리 방안", 한국통신학회'2000 하계 학술발표 논문집, pp.775-778, 2000년 7월.

[13] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Vol.1 No.4, pp. 397-413, Aug. 1993.

[14] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Services," *IEEE/ACM Transactions on Networking*, Vol 6, No 4, pp. 362-373, Aug. 1998.

[15] W. Feng et al., "A Time Sliding Window Three Colour Marker(TSWTCM)," *RFC-2859*, June 2000.

[16] R. Jain, *The Art of* Computer *Systems Performance Analysis*, John Wiley and Sons Inc, 1991.

남 동 호

1999년 2월 경북대학교 전자공학과 졸업(공학사). 2001년 2월 경북대학교 전자공학과 졸업(공학석사). 2001년 3월 ~ 현재 삼성전자 연구원. 관심분야는 차세대 인터넷 프로토콜, 차세대 보안

최 영 수

1998년 2월 경북대학교 전자공학과 졸업(공학사). 2000년 2월 경북대학교 전자공학과 졸업(공학석사). 2002년 3월 ~ 현재 경북대학교 전자공학과 박사과정 수료. 관심분야는 차세대 인터넷 프로토콜, 광 인터넷에서의 서비스 품질 보장 제어

김 병 철

1995년 2월 경북대학교 전자공학과 졸업(공학사). 1997년 2월 경북대학교 전자공학과 졸업(공학석사). 1997년 3월 ~ 현재 경북대학교 전자공학과 박사과정 수료. 2002년 3월 ~ 현재 NIST 객원 연구원. 관심분야는 트래픽 제어, 차세대 인터넷 프로토콜, 광 인터넷

조 유 제

1982년 2월서울대학교 전자공학과 졸업(공학사). 1983년 8월 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1988년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학박사). 1989년 3월 ~ 현재 경북대학교 공과대학 전자전기컴퓨터학부 교수. 1992년 8월 ~ 1994년 1월 Univ. of Toronto, 객원교수. 2002년 3월 ~ 현재 NIST 객원 연구원. 관심분야는 트래픽 제어, 차세대 이동통신망, 광 인터넷, 차세대 인터넷 프로토콜