논문 00-25-6A-17 한국통신학회논문지 '00-6 Vol.25 No.6A

# 고속 라우터를 위한 Drop-tail방식의 공정한 대역할당 알고리즘

정회원 이 원 일*, 윤 종 호**

# A Fair Drop-tail Bandwidth Allocation Algorithm for High-speed Routers

Won-il Lee*, Chong-Ho Yoon** *Regular Members*

요 약

기존의 인터넷 라우터용 트래픽관리 알고리듬인 Random Early Detection(RED)은 기존의 최선형 서비스 특성을 가지는 호름들간에서는 올바르게 동작하지만, 다양한 특성을 가지는 호름들에 대하여 링크를 공정하게 할당하기에는 많은 문제점을 가지고 있어 새로운 per-flow방식의 알고리듬들이 제안되고 있다. 본 논문에서는 라우터의 호름간 공정한 대역할당을 위하여, per-flow방식을 사용하는 새로운 Fair Droptail알고리듬을 제안하고 이것이 기존 RED의 문제점을 해결할 수 있음을 보였다. 제안된 Fair Droptail은 RED와는 달리 동적으로 호름이 생성, 소멸될 때 호름들 마다 동일한 크기의 버퍼를 할당함 으로서, 호름들의 특성과 관계없이 호름들이 출력링크의 대역을 공정히 할당 받을 수 있도록 한다. 그리고, 이 Fair Droptail알고리듬은 기존의 복잡한 per-flow방식과는 달리 활성화된 상태의 호름에 대한 정보만 유지하는 알고리듬으로 구현되므로, 구현이 간단한 장점이 있다.

ABSTRACT

Because the random early detection(RED) algorithm deals all flows with the same best-effort traffic characteristic, it can not correctly control the output link bandwidth for the flows with different traffic characteristics. To remedy this problem, several per-flow algorithms have been proposed. In this paper, we propose a new per-flow type Fair Droptail algorithm which can fairly allocate bandwidth among flows over a shared output link. By evenly allocating buffers per flow, the Fair Droptail can restrict a flow not to use more bandwidth than others. In addition, it can be simply implemented even if it employs the per-flow state mechanism, because the Fair Droptail only keeps each information of flow in active state.

## I. 서 론

인터넷의 트래픽은 일정시간동안 같은 source/destination을 가지는 연속된 패킷들을 일컫는 다양한 특성을 가지는 호름들로 이루어 진다. 이러한 호름들의 특성은 일반적으로 unresponsive, bulk-responsive, responsive의 세 가지로 분류된다. 먼저, unresponsive 호름은 라우터가 혼잡을 감지하여 유입되는 호름의 일부 패킷을 버려 묵시적으로 혼잡이 발생하였음을 알림에도 불구하고 송신측이 전송률을 감소하지 않는 것으로서, UDP를 사용하는 화상회의응용이 그 예이다. 이 UDP는 TCP와는 달리 혼잡제어기능이 없기 때문에 혼잡발생시에도 자신의 전송율을 그대로 유지하므로 다른 호름들과 경쟁시 항상 우위를 차지하여 각 호름들이 공유하고 있는

* KDC정보통신(주) 연구소(wilee@kdcre.co.kr),
** 한국항공대학교 항공통신정보공학과(yoonch@mail.hangkong.ac.kr)
논문번호 : 00096-0317, 접수일자 : 2000년 3월 17일

출력링크를 공정히 사용을 할 수 없게 만드는 주 원인이 된다. 두번째인 bulk-responsive 흐름은 손실 발생시 혼잡제어알고리듬에 의해서 송신측의 전송량을 줄일 수 있으며, 이후 혼잡이 완화되었을 때 본래의 전송율로의 회복이 빠르며 항상 자신이 보낼 데이터를 가지고 있어서 최소한 자신에게 할당될 수 있는 대역 이상을 보내는 것으로서, 예로서 FTP 서비스 흐름이 있다. 마지막으로 responsive 흐름은 혼잡 제어를 따르며 bulk-responsive흐름들 보다 전송율이 낮으며 손실에 민감하고 전송량의 회복이 느린 흐름이다. 일반적으로 RTT가 크고 송신 윈도우크기가 작은 Telnet이 예가 된다. Responsive 흐름과 bulk-responsive흐름은 둘 다 혼잡제어 알고리듬을 가지고 있으므로 adaptive흐름라고도 한다[3]. 이렇게, 다양한 종류의 흐름의 특성을 고려하지 않은 기존 라우터용 흐름제어 알고리듬인 Random Early Detection(RED)은 라우터의 혼잡발생시 단말간의 TCP 와 같이 혼잡제어알고리듬을 따르는 adaptive흐름일 때에만 올바르게 동작한다[1]. 또한 같은 adaptive 흐름이라도 전송지연 시간 등 다른 환경변수를 가질 때 흐름간 공정한 대역할당을 할 수 없다. 이러한 문제점을 해결하기 위하여 CBQ, FRED, CBT, DECbit등의 여러 알고리듬이 제안되었다[3]. 그러나 이러한 알고리듬들은 좋은 성능에도 불구하고 알고리듬이 복잡하고 상태정보관리에 많은 처리 비용이 들기 때문에 라우터에 탑재하기에는 많은 어려움이 따른다[5].

본 논문에서는 RED의 문제점을 도출한 다음, 라우터의 출력링크를 각 흐름들이 자신의 흐름 특성에 관계없이 공정한 대역할당을 보장할 수 있으며, 구현성이 높은 새로운 Fair Droptail알고리듬을 제안하고, 라우터에서 올바르게 동작할 수 있음을 보인다.

본 서론에 이어, 2장에서는 기존 RED의 동작과 문제점을 기술하고, 3장에서는 이러한 문제점을 해결한 Fair Droptail 방식을 제안한다. 4장에서는 시뮬레이션을 통하여 제안된 Fair Droptail 알고리듬이 RED에 비하여 출력링크 대역을 보다 공정하게 할당하는 것을 보이고, 마지막으로 5장에서 결론을 맺는다.

## Ⅱ. RED(Random Early Detection)의 문제점

라우터는 유입되는 패킷들을 라우팅하기 위하여 먼저 패킷을 큐에 일시 저장한다. 이때 가장 간단한 저장방법은 FIFO 큐를 사용하는 것이다. 만약 패킷의 입력률이 출력링크의 처리율을 넘는 경우에는 유입되는 패킷들이 버려지게 된다. 이러한 기본적인 FIFO처리 방식을 Droptail이라한다. 이 방법은 가장 간단하여 구현하기는 쉽지만 흐름간의 불공정한 출력링크대역 할당, global synchronization에 따른 평균지연시간의 증가 등의 문제점을 가지고 있어, 널리 사용되지는 못했다. 이를 해결하기 위하여, 큐의 현재길이를 반영하여 패킷들을 관리하는 active queue 관리방법들이 제안되었는데 그 중에서 대표적인 것이 Random Early Detection(RED) 이다[1]. 이 방법은 라우터의 버퍼길이가 정해진 수준을 넘어서면 혼잡이 발생할 가능성이 있다고 판단하고 미리 임의의 확률에 의해서 유입되는 패킷을 선택하여 버림으로써 큐의 넘침을 미리 방지할 뿐만 아니라, 임의의 확률로 큐 내부의 패킷들을 선택적으로 버림으로써 TCP연결들의 global synchronization 문제를 예방한다. 또한 큐의 평균 길이를 설정된 수준이하로 유지함으로써 버스트성 데이터를 수용할 수도 있도록 하였다. RED에 따르면, 입력률이 높은 흐름일수록 확률적으로 많이 버려지게 되므로 각 흐름간의 공정성을 어느 정도 실현하고 있다고 할 수 있다.

그러나 이러한 패킷의 폐기시 참조하는 값은 전체 패킷길이와 현재큐길이 뿐으로서 각 흐름의 출력링크 사용비율을 고려하지 않기 때문에 각 흐름은 동일한 패킷폐기확률을 적용받게 된다. 이런 RED의 특성 때문에 다음과 같은 문제가 발생한다. 첫째, RED가 Droptail같은 기존 방식보다는 출력링크의 공정한 대역할당 성능이 좋지만 적은량의 패킷을 전송하는responsive 흐름들에게도 같은 확률을 적용시킨다. 즉 전송 패킷이 많은 unresponsive나 bulk-responsive흐름도 역시 많은 량이 폐기될 것이지만 전송량 관점에서는 여전히 많은 량을 전송할 수 있다. 그에 반해 responsive흐름은 적은 량을 보내지만 똑같은 확률로 폐기됨으로서 상대적으로 더 적은 량을 보내게 된다.

둘째, 폐기하지 않고 패킷을 받아들이게 되면 다음에 들어올 패킷들은 더 높은 폐기 확률로 적용되게 된다. 높은 입력률을 가지는 흐름의 패킷이 받아들여지면 큐의 길이가 증가 되어 낮은 입력률을 가지는 흐름의 패킷이 도착할 때 더 높은 확률이 적용되어 폐기될 수 있어서 이러한 문제점 때문에 공평한 링크 사용을 할 수 없게 된다.

마지막으로, 혼잡 제어를 따르지 않는 UDP와 같은 unresponsive흐름 때문에 혼잡이 발생할 경우에 RED는 모든 흐름들에게 동일한 높은 폐기확률을

적용시킨다. 이러한 문제는 소수의 unresponsive 흐름이 높은 입력율을 가질 때 다른 흐름들의 전송을 방해하게 된다.

## Ⅲ. Fair Droptail

제 2 장에서 살펴본 바와 같이 RED는 서로 다른 흐름특성을 가지는 흐름들이 라우터의 링크를 공유하여 사용할 때 공정한 대역 할당을 하지 못하는 문제점이 발생한다. 이러한 문제점을 보완하기 위해서 RED를 기반으로 한 FRED, CBT등 여러 알고리듬들이 제시 되었다[3]. 이러한 개선 방안들은 흐름의 특성을 구분해 내기 위해서 per-flow방식을 사용하고 있다. Per-flow방식은 흐름별로 정보를 유지하므로 흐름의 특성에 따라서 라우터에서 처리할 수 있으므로 공정한 대역할당을 할 수 있게 된다. 그러나 흐름들에 관한 정보를 유지하고 처리하기 위해서 복잡한 알고리듬을 필요로 하기 때문에 고속의 라우터 처리능력이 요구된다. 또한 일반적으로 라우터에는 수많은 흐름들이 통과 하기 때문에 라우터의 부하는 더욱 커지게 된다. 여기서는 소단원에 관한 내용을 간단히 살펴본다. 여기서는 소단원에 관한 내용을 간단히 살펴본다.

본 논문에서 제안하는 Fair Droptail알고리듬은 다른 개선된 RED방법과 마찬가지로 per-flow방식을 사용한다. 그러나 이 Fair Droptail알고리듬은 활성화된 흐름상태정보, 즉 현재 큐안에 버퍼링된 흐름들에만 한하여 흐름정보를 가지고 있으므로 라우터를 통과하는 많은 흐름이 존재하더라고 실제적으로 큐안에 버퍼링된 흐름만의 정보를 가지고 큐를 관리하게 되므로 라우터의 처리부하가 크지 않다. 또한, 사용되는 상태변수는 흐름이 사용하는 큐의 길이정보와 오버플로우 횟수뿐이므로 메모리 또한 많이 필요로 하지 않는다. 또한, 제안된 Fair Droptail알고리듬은 현재의 흐름 개수 만큼 라우터의 큐의 크기를 분할하여 사용하도록 제한한다. 큐의 분할크기의 조정은 가장높은 입력률을 가지는 흐름에 기준하여 동작하기 때문에 입력률이 낮은 흐름과 라우터에 공존할 때 생기는 under utilization을 최소화 하였다. 각 흐름들은 자신에게 할당된 크기의 큐를 넘어서는 경우 기존의 droptail방식 같이 무조건 폐기된다. 또한 자신에게 할당된 큐길이를 초과하여 흐름들에 대해서 버퍼링되는 횟수를 기록하여 그 횟수가 어느 수준이상이 되면 일정기간동안 그 흐름이 라우터의 큐에 버퍼링되지 않도록 제한한다. 이렇게 하는 이유는 unresponsive와 bulk-responsive흐름이 과도하게 링크를 사용하는 것을 방지하기 위해서이며, 이렇게 함으로서 적은 양의 데이터만을 전송하는 responsive흐름을 보호하는 장점이 있다. 결과적으로, 제안하는 Fair Droptail알고리듬은 active flow state방법과 Flow Droptail방법으로 구성되어 간단히 구현될 수 있으며 흐름들의 특성과 관계없이 흐름들 간에 공정한 대역할당을 할 수 있다. Fair Droptail의 구체적인 알고리듬의 동작은 <그림 1>과 같다.

```
For  arrival from Flow F
 If  Flow F has no state
          ActiveState(F,+)
 If(Qlen_F>AvgAllocQueLen)OR(FoulNumber_F>=K)
          Drop Packet
          FoulNumber_F++
 Else
          Queue Packet
          FoulNumber_F --
For  Departing  from Flow F
 If Qlen_F=0
          ActiveState(F,-)

ActiveState(F,Operator) :
 IF Operator is +
          N++
          Qlen_F = 0
          FoulNumber_F=0
          AvgAllocQueLen = Q_MAX/N
 Else if Operator is -
          N--
          Delete flow information
          AvgAllocQueLen = Q_MAX/N


Saved Variables:
 AvgAllocQueLen:Max Queue Length for One Flow
 N : Number of Flows
 Qlen_F : Queue Length Flow F
 FoulNumber_F:Number of Excess AvgAllocQueLen
Fixed parameters:
  K : allowed FoulNumber
  Q_MAX : Max Queue Length
```

그림 1. Fair Droptail 알고리듬.

라우터에 패킷이 도착했을 때 그 패킷이 속하는 흐름이 새로운 흐름이면 흐름의 상태정보를 위한 테이블을 생성하고 현재 활성화된 흐름의 갯수 $N$을 증가 시킨다. $N$이 변화하면 각 흐름들이 최대로 버퍼링될 수 있는 큐의 길이인 *AllocQueLen*는 다시 재 조정된다. 패킷이 도착했을 때, 라우터는 그 패킷의 흐름이 이미 사용하고 있는 큐의 길이가 할당된 *AllocQueLen*을 넘어서는 경우 이 흐름을 일단 bulk-responsive흐름이라고 판단하고 그 패킷을 버린다. 또한, 이 흐름이 unresponsive흐름이거나 사용자가 임의로 조작한 TCP흐름일지도 모르므로, *AllocQueLen*을 넘어서는 횟수인 *FoulNumber*를 사용하여, *FoulNumber*가 K번 이상이 되면 라우터는 이 흐름을 unresponsive흐름으로 판단하여 버리도록 하였다. 일단 unresponsive흐름으로 판정되면 큐 안에 남아있는 unresponsive흐름의 패킷들이 다 서비스되어 그 흐름이 소멸될 때 까지 이 흐름에 속한 추가의 패킷들은 모두 버려지도록 하였다.그리고, 송신단말이 자신의 패킷손실 사실로 부터 망의 혼잡을 감지하여 전송율을 낮추는 경우, 그 흐름이 차지하는 큐의 길이인 $Qlen_F$는 *AllocQueLen* 보다 작게되고 *FoulNumber*는 다시 감소하므로 bulk-responsive 흐름을 responsive흐름으로 처리하게 된다. $K$값이 작은 경우 bulk-responsive흐름들이 쉽게 unresponsive흐름으로 판정되고, 큰 경우는 unresponsive 흐름들이 bulk-responsive 흐름으로 처리되는 경우가 많아지게 된다.흐름의 수가 많은 경우와 적은 경우에 따라서 $K$의 값은 조절되어 질 수 있으며 본 논문에서는 일반적인 경우 $K$가 2 일 때 가장 공정한 대역 할당이 이루어질 수 있음이 시뮬레이션을 통하여 확인되었다.

## Ⅳ. 성능비교

### 1. Unresponsive흐름과 Adaptive 흐름이 경쟁하는 경우

<그림 2>는 호스트 5번과 6번은 500바이트의 패킷 크기로 4Mbps의 constant bit rate(CBR)의 unresponsive UDP흐름을 발생시킨 예이다. 이 unresponsive 흐름과 경쟁하는 1번부터 4번까지의 TCP호스트들은 모두 1000바이트의 패킷길이로 패킷을 전송하는 Bulk-responsive 흐름이다. 입력과 출력링크의 속도는 각각 100Mbps와 10Mbps지만 패킷의 전송지연시간은 동일한 2ms를 가진다고 가정하였다.

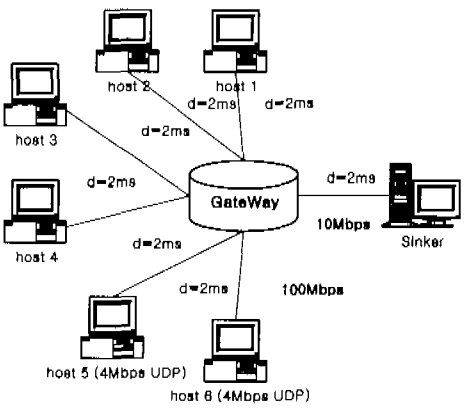
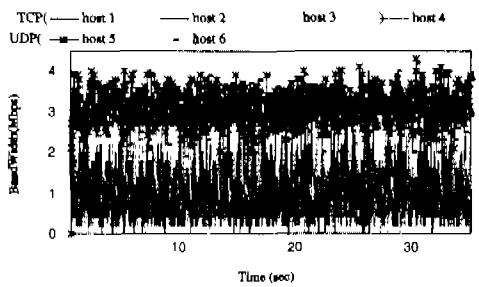
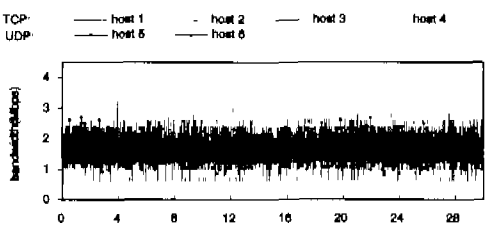



그림 2. Unresponsive흐름과 adaptive 흐름간에 경쟁하는 시험환경.

<그림 3>은 <그림2>와 같은 망 환경에서 30초간 시뮬레이션을 실행했을 때 라우터의 출력 링크의 사용율을 도시한 것이다. 시뮬레이션은 객체지향 언어인 SIMULA를 사용하였다[8]. 이상적으로 공정하게 출력링크의 대역 할당이 각 단말에 대하여 이루어진다면 6개의 모든 호스트들은 10Mbps의 출력링크를 각각 1.66Mbps씩 나누어 사용해야 한다. 그러나, RED를 사용할 경우 <그림 3-(a)>에서 볼 수 있듯이 호스트 5와 6이 각각 자신의 전송율인 4Mbps에 가까운 대역을 모두 사용하게 되어, 출력링크 대역의 여유분 2Mbps를 4개의 TCP호스트들




(a)RED를 사용했을 경우




(b)Fair Droptail을 사용했을 경우

그림 3. Unresponsive와 adaptaive흐름 경쟁시 RED와 Fair Droptail 방식에 대한 라우터에서의 대역할당비교.

이 나누어 사용함을 확인할 수 있다. 이것은 TCP호스트들이 혼잡제어를 따르는 반면에 혼잡제어를 하지않는 UDP호스트들은 TCP호스트가 감소시켜 남은 대역까지 점유하여 사용하게 되기 때문이다.

표 1. RED와 Fair Droptail 방식에 대한 호스트별 대역할당비율과 손실률 비교

| | 대역할당비율(%) | | 손실율(%) | |
|---|---|---|---|---|
| | RED | Fair Drptail | RED | Fair Droptail |
| HOST1(TCP) | 10 | 16.89 | 22.04 | 7.7 |
| HOST2(TCP) | 10 | 17.04 | 22.9 | 7.3 |
| HOST3(TCP) | 10 | 16.91 | 21.7 | 7.7 |
| HOST4(TCP) | 10 | 17.04 | 22.6 | 7.29 |
| HOST5(UDP) | 30 | 16.46 | 22.3 | 58.9 |
| HOST6(UDP) | 30 | 15.65 | 25.3 | 60.9 |

반면에 <표 1>를 보면 Fair Droptail 알고리듬은 UDP호름인 5번,6번 UDP호스트들의 과도한 대역사용을 각 호스트의 공정한 할당 대역폭인 1.66Mbps로 제한 함으로써 다른 TCP호름의 대역사용을 가능하게 해주고 있음을 확인 할 수 있다. 또한, 시뮬레이션동안 전송된 호름별 전송 데이터량과 각 호름의 손실 확률값에서 알 수 있듯이, 제안된 Fair Droptail방식은 RED와 달리 과도한 패킷을 전송하는 unresponsive 호스트 5번과 6번의 패킷 손실율을 높임으로서 혼잡제어를 따르는 TCP트래픽의 대역을 보호하여 각 호름이 출력링크대역의 16~17%씩 공정하게 나누어 사용하도록 함을 확인할 수 있다.

## 2. Bulk-responsive호름과 Responsive 호름들이 경쟁하는 경우

인터넷에서의 전송 지연시간이 큰 호름과 전송지연이 작은 호름이 경쟁할 경우, 즉 호스트들의 Round Trip Time(RTT)에 차이가 있을 경우에 호름의 특성 또한 차이를 보이게 된다. TCP는 슬로우스타트(slow start)알고리듬을 사용하기 때문에 송신윈도우를 다음 단계로 증가 시키는데 한번의 RTT 시간 만큼 기다려야 한다[2][4][6]. 따라서 RTT가 작은 호름인 경우는 RTT가 큰 호름 보다 자신의 송신윈도우 크기를 더 빠른 속도로 증가시키게 되므로, 라우터 출력링크의 대역을 차지할 수 있게 된다. 따라서 올바른 제어를 위해서는 RTT가 작은 호스트들의 패킷폐기율을 크게하고 RTT가 큰 호스트들의 패킷 폐기율을 낮추어야 한다.

<그림 4>는 작은 전송지연 시간을 가지는 bulk-responsive 특성을 가지는 호스트들과 긴 전송지연시간을 responsive호스트5가 라우터의 링크를 공유하고 있는 네트워크 구성의 예이다.

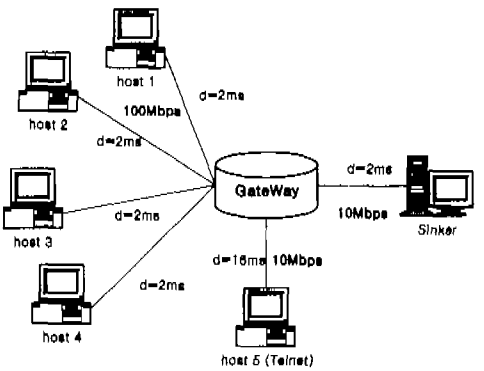



그림 4. Bulk-responsive와 Responsive호름간 경쟁하는 시험환경.

호스트 1번부터 4번은 모두 2ms 의 짧은 전송지연시간을 가지며100Mbps의 고속의 링크상에서 bulk-responsive 특성을 가지는 호름을 발생시킨다. 반면에, 호스트 5는 16ms의 비교적 큰 전송지연 시간을 가지며 10Mbps 의 링크속도를 가진다. 시뮬레이션에서 TCP 호스트 5번은 송신윈도우를 작게 제한하고 RTT의 크기를 크게 하여 전송 데이터량이 작고 혼잡제어 반응이 느린 responsive 호름 특성을 가지도록 했다.

RTT가 작고 송신윈도우 크기가 큰 bulk-responsive호름은 안정상태에서 송신윈도우의 크기는 bandwidth delay-product크기를 가지게 되며 라인상에 모든 패킷이 포화되어 흘러가는 경우가 된다[6]. 반면 RTT가 작고 송신 윈도우 크기가 작은 responsive호름은 혼잡제어상태가 아닌 안정상태에서 bandwidth delay-product보다 작은 크기의 데이터를 전송하게 된다. RED라우터에서는 혼잡발생시 패킷을 버리게 될 경우, bulk-responsive호름은 짧은 시간 안에 다시 많은 양의 패킷을 전송한다. 반면에 responsive호름은 혼잡제어에서 회복되는데 많은 시간이 걸릴뿐아니라 회복되더라도 bulk-responsvie호름이 이미 responsive호름이 사용하지 않는 대역까지 사용하고 있기 때문에 적은 수의 패킷을 전송하게 된다. 그 결과 원래 적은량의 패킷을 전송하는 responsive 호름은 더 적은 량의 패킷을 전송하게 된다.

<표 2>는 <그림4>의 망 환경에서 라우터에 RED와 Fair Droptail을 각각 적용하여 30초간 시뮬레이션을 행했을 때 각 호스트의 전송비율과 손실확률을 보이고 있다.

표 2. 호스트별 전송비율과 손실률 비교

| | 대역할당비율(%) | | 손실율(%) | |
|---|---|---|---|---|
| | RED | Fair Drptail | RED | Fair Droptail |
| HOST1(bulk) | 22.70 | 22.36 | 4.35 | 3.3 |
| HOST2(bulk) | 22.73 | 22.25 | 4.34 | 3.3 |
| HOST3(bulk) | 22.10 | 22.37 | 4.85 | 3.3 |
| HOST4(bulk) | 23.70 | 22.29 | 4.25 | 3.1 |
| HOST5 (responsive) | 7.77 | 10.73 | 3.05 | 0 |

Responsive흐름의 호스트 5는 최대로 자신의 패킷을 전송할지라도 자신의 공유 대역폭인 2Mbps를 다 사용하지 못한다. RED를 사용하였을 경우 호스트 5번의 전송 비율은 8%로 원래의 자신에게 할당될 수 있는 공유대역 비율인 20%에 크게 못 미치는 량을 전송함에도 불구하고 손실율은 다른 bulk-responsive흐름들과 그다지 차이가 없다. 반면에 Fair Droptail의 경우 responsive 흐름의 손실율은 0%로 유지시켜 responsive흐름을 보호하고 있음을 확인 할 수 있다.

### 3. 복합적인 흐름들간의 경쟁

Fair Droptail알고리듬이 인터넷과 같이 다양한 흐름이 존재하는 복합적인 상황에서 동작하는지를 알아보기 위하여 <그림5>와 같이 unresponsive, bulk-responsive, responsive 흐름들이 각기 다른 전송지연시간을 가지며 공존할 때 RED와 Fair Droptail방식에 대하여 각각 어떻게 대역할당을 하는지 비교하였다. 각각의 호스트들은 서로 다른 전송지연 시간을 가지며 unresponsive, bulk-responsive, responsive세가지 종류의 흐름을 발생시키게 된다. 호스트 5번은 5Mbps의 CBR UDP흐름을 발생시키며, Telnet 특성을 가지는 호스트 3번의 송신 윈도우크기는 4000바이트로 제한하였고 호스트 4번은 responsive 특성에 최대 윈도우(64KB)를 설정하여 responsive이지만 송신윈도우 제한하지 않음으로써 자신의 최대 전송량인 bandwidth delay-product 크기를 보낼 수 있도록 설정하였다. 그리고 bulk-responsive 흐름 특성을 가지는 호스트 1번과 2번의 송신윈도우 크기는 Bandwidth delay-product크기인 최대값 10000바이트로 설정하였다[6].

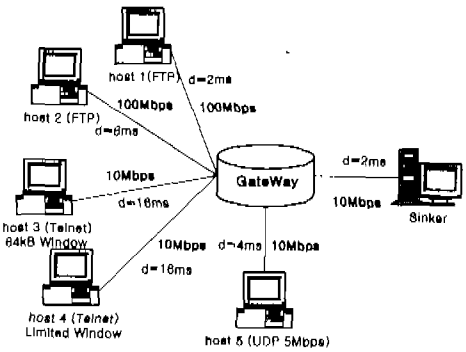
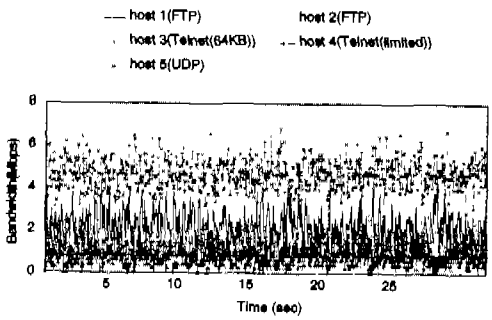
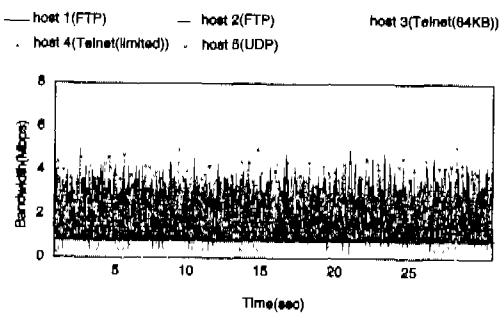



그림 5. 다양한 특성을 가지는 호름들이 있는 시험환경

<그림6>은 30초간의 RED와 Fair Droptail의 각각의 출력링크의 대역 사용량을 비교한 것이다. <그림 6-(a)>에서 확인할 수 있듯이 RED를 사용할 경우 CBR UDP 호스트 5번이 자신의 전송율인 5Mbps의 대역을 거의 사용함으로써 다른 호스트들의 대역 사용을 방해하고 있다. 또한 RTT가 가장 짧은 Bulk-responsive 호스트1번이 UDP 데이터를 발생하는 호스트 5번 다음으로 대역을 가장 많이 사용하고 있다. 이는 호스트 1번이 혼잡제어시에 빨리 반응하기 때문이다.




(a) RED를 사용한 경우




(b) Fair Droptail을 사용한 경우

그림 6. ORED와 Fair Droptail알고리듬의 대역할당능력 비교

반면 <그림 6-(b)>에서 보면 Fair Droptail의 경우 혼잡제어 알고리듬을 가지고 있는 bulk-responsive호스트인 1번과 2번은 unrepsonsive 호스트 5번에 영향을 받지 않고 공정하게 자신의 공유대역을 사용하고 있음을 확인할 수 있다. 또한 긴 전송시간을 가지고 있지만 최대윈도우(64KB) 크기의 송신윈도우를 가지고 있는 Telnet 호스트인 3번도 자신의 공유대역을 올바르게 사용하고 있다.Telnet호스트 4번의 사용 공유대역폭이 적은이유는 작은 송신 윈도우 크기 제한 때문이다. <표 3>에서 알 수있듯이 Fair Droptail을 사용했을 경우, 호스트4번은 자신의 사용 공유 대역 20%이하를 사용하므로 손실율은 0%자신의 모든 패킷을 손실없이 전송할 수 있다.

표 3. 호스트별 전송비율과 손실률 비교

| | 대역할당비율(%) | | 손실율(%) | |
|---|---|---|---|---|
| | RED | Fair Drptail | RED | Fair Droptail |
| HOST1(FTP) | 21.48 | 25.32 | 6.0 | 3.1 |
| HOST2(FTP) | 12.66 | 21.84 | 7.2 | 2.6 |
| HOST3 (Telnet 64K) | 10.95 | 18.80 | 5.9 | 2.5 |
| HOST4(Telent) | 7.37 | 10.56 | 5.9 | 0 |
| HOST5(UDP) | 47.54 | 23.47 | 6.8 | 5.3 |

## V. 결 론

본 논문에서는 기존의 RED의 문제점을 도출한 다음, 흐름들의 특성에 관계없이 공정한 대역할당을 보장할 수 있는 Fair Droptail 알고리듬을 제안하였다. 기존의 라우터 출력링크를 관리하기 위해 사용되는 RED는 모든 최선형(best-effort)서비스를 사용한다는 가정하에서 고안된 알고리듬이기 때문에 흐름 특성이 다른 흐름들이 출력링크 대역을 경쟁할 때 그 대역을 올바르게 할당하지 못한다.

제안된 Fair Droptail 알고리듬은 버퍼링되어 있는 흐름별로 정보를 관리하는 acitve state 방식을 사용하고 흐름별로 사용할 수 있는 큐의 크기를 제한함으로써 라우터의 출력링크 대역을 올바르게 할당할 수 있는 특징이 있다.

시뮬레이션 분석결과 제안된 Fair Droptail 방식은 UDP와 같이 혼잡제어 기능이 없는 unresponsive흐름을 제한하여 다른 흐름들의 대역을 보호할 뿐만 아니라, 적은 양의 패킷을 전송하는 responsive 흐름을 다른 종류의 흐름으로 부터도 보호 할 수 있음을 확인하였다. 제안된 Fair Droptail 알고리듬은 기존의 최선형 서비스 기반의 망의 라우터에 RED를 대신하여 탑재될 경우 라우터의 출력링크 대역을 공정히 할당하지 못하는 등의 기존의 여러 문제점을 해결할 수 있음을 알 수 있다.

앞으로 많이 사용될 인터넷 폰 등의 실시간 전송용 UDP패킷들에 대하여, TCP흐름들과 더불어 상호간에 공정한 대역할당을 제공할 수 있는 이 알고리듬은 Internet2등 앞으로 전개될 새로운 인터넷 서비스에 활용될 수 있는 기술 중에 한가지가 될 수 있을 것이다.

## 참 고 문 헌


[1] Sally Floyd and Van Jacobson, Random Early Detection gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, 1(4), pp. 397-413, August 1993.

[2] Van Jacobson and Michael J.Karles, Congestion Avoidance and Control, *SIGCOMM Symposium on Communication Architectures and Protocols*, pp. 314-329, 1988.

[3] Dong Lin and Robert Morris, Dynamics of Random early Detection, http://www.eecs.harvard.edu/ networking/papers/fred-abstract.html.

[4] William Stallings, High-Speed Networks, Prentice Hall, 1998.

[5] Sally Floyd and Kevin Fall, Router Mechanisms to Support End-to-end Congestion Control, LBL lab., Berkely CA,February,1997.

[6] W. Stevens, TCP/IP Illustrated, Volume I, Addison-Wesley, 1994.

[7] M. Parris, K. Jeffay, F. Donelson Smith, Lightweight Active Router Queue Management for Multimedia Networking, *Multimedia Computing and Networking 1999,SPIE Proceedings Series*, Vol. 3654, San Jose, CA, pp.162-174, January 1999.

[8] G.M, Birtwistle, SIMULA Begin, Petrocelli/ Charter, 1975.

이 원 일(Won-il Lee) 정회원

1998년 2월 : 한국항공대학교 항공통신정보공학과 (공학사)
2000년 2월 : 한국항공대학교 항공통신정보공학과 (공학석사)
2000년 3월~현재 : KDC정보통신(주) 연구원
<주관심 분야> 인터넷응용, 성능분석, 광통신 공학

윤 종 호(Chong-Ho Yoon) 정회원

1984년 2월 : 한양대학교 전자공학과(공학사)
1986년 2월 : 한국과학기술원 전기 및 전자공학과 (공학석사)
1990년 2월 : 한국과학기술원 전기 및 전자공학과 (공학박사)
1995년~1996년 : U. of Arizona 방문교수
1996년~1998년 : 한국항공대학교 전산소장
1991년~현재 : 한국항공대학교 전자정보통신컴퓨터 공학부 부교수
<주관심 분야> 인터넷응용, 성능분석