TCP와 UDP 플로우 간의 공정성 개선을 위한 새로운 큐 관리 알고리즘

채 현 석[†] · 최 명 렬[††]

요 약

인터넷의 혼잡상황을 해결하기 위하여 제안된 RED(Random Early Detection)와 같은 능동적 큐 관리(Active Queue Management) 알고리즘들은 TCP 데이터에 대하여 우수한 혼잡제어 효과를 나타낸다. 그러나 TCP와 UDP가 병목 링크를 공유하는 경우 불공정성 문제와 큐에서의 지연시간이 길어지는 문제점을 가지고 있다. 본 논문에서는 공정성을 개선함과 동시에 큐 지연시간을 감소할 수 있는 새로운 큐 관리 알고리즘인 PSRED(Protocol Sensitive RED) 알고리즘을 제안하였다. PSRED 알고리즘은 트래픽의 프로토콜 필드를 이용하여 플로우의 종류를 구분하고 각기 다른 패킷폐기함수를 적용함으로써 공정성을 개선하고 평균 큐 길이를 줄일 수 있다.

A New Queue Management Algorithm for Improving Fairness between TCP and UDP Flows

Hyun-Seok Chae[†] · Myung-Ryul Choi[††]

ABSTRACT

AQM (Active Queue Management) techniques such as RED (Random Early Detection) which be proposed to solve the congestion of internet perform congestion control effectively for TCP data. However, in the situation where TCP and UDP share the bottleneck link, they can not solve the problems of the unfairness and long queueing delay. In this paper, we proposed an simple queue management algorithm, called PSRED (Protocol Sensitive RED), that improves fairness and decreases queueing delay. PSRED algorithm improves fairness and decreases average queue length by distinguishes each type of flow in using protocol field of packets and applies different drop functions to them respectively.

키워드 : 능동적 큐 관리(Active Queue Management), RED, 혼잡제어(Congestion Control), 공정성(Fairness)

1. 서 론

Drop-tail 방식의 FIFO(First In First Out) 큐는 구현이 쉬우며 순간적으로 폭주하는 인터넷 트래픽을 처리할 수 있어 현재 인터넷을 구성하고 있는 대부분의 라우터에서 사용되고 있다. 그러나 높은 대역폭을 사용하는 응용프로그램의 경우에는 라우터에 혼잡상황(congestion collapse)을 야기하게 되며, 링크의 대역폭을 효율적으로 사용할 수 없게 된다. 또한 TCP와 같은 반응 플로우(responsive flow)와 UDP와 같은 비반응 플로우(unresponsive flow)가 동시에 하나의 큐, 혹은 링크에서 경쟁할 경우에는 심각한 불공정성 문제가 발생된다.

현재 이러한 문제를 해결하기 위한 연구는 수송계층에서 수행하는 혼잡제어 알고리즘을 개선하는 방법과 라우터의 큐에서 네트워크로 유입되는 트래픽을 제어하는 큐 관리 알고리즘을 개선하는 방법의 두 가지 방향으로 진행되고 있다. IETF(Internet Engineering Task Force)에서는 TCP의 혼잡제어 알고리즘 개선하여 TCP Tahoe, TCP Reno, TCP New-Reno 등의 TCP 버전을 발전시켜왔으며, 가장 최근에는 TCP Vegas가 제안되었다. 실제로 TCP Vegas는 기존에 사용하던 TCP에 비하여 인터넷에서의 성능이 40% 까지 향상된다[1]. 그러나 기존의 TCP와 하나의 링크를 동시에 사용할 경우에는 공정성에 심각한 문제가 발생하는 단점이 있다[2].

RED(Random Early Detection)로 대표되는 큐 관리 알고리즘은 라우터의 큐에 입력되는 패킷을 확률적으로 폐기함으로써 큐가 넘치는 것을 방지하고 TCP 윈도우 크기가 너무 커지는 것을 방지하여 네트워크로 유입되는 트래픽의 양을 효과적으로 제어할 수 있다[3]. RED는 인터넷의 중간 노드에서 동작하며, TCP와 같이 패킷의 폐기에 반응하여 데이터의 전송률을 제어하는 반응 플로우에 대하여 효과적

† 정 회 원 : 동원대학 인터넷정보과 교수
†† 정 회 원 : 한양대학교 전자컴퓨터공학부 교수
논문접수 : 2003년 9월 8일, 심사완료 : 2003년 11월 27일

으로 혼잡제어를 수행한다. 그러나 UDP와 같이 패킷의 폐기에 따른 혼잡제어를 수행하지 않는 비반응 플로우의 트래픽이 증가하면 효과적인 혼잡제어를 수행하지 못하여 비반응 플로우에 의한 혼잡상황이 더욱 가중되고, 반응 플로우와 비반응 플로우간의 성능에 불공정성 문제가 발생하는 단점이 있다[4].

인터넷방송이나 화상회의, 인터넷 폰과 같이 멀티미디어 형태의 데이터를 다루는 응용 프로그램은 데이터의 신뢰성보다는 낮은 전송지연과 균일한 데이터 전송률을 요구하고 있어 대부분의 경우 UDP와 같은 비반응 플로우를 사용하고 있다. 따라서 TCP 플로우와의 공정성을 해결하기 위한 연구는 필수적이며, 이를 해결하기 위한 가장 확실한 방법으로는 모든 트래픽을 공정하게 처리할 수 있는 FQ(Fair Queueing)가 있다. 그러나 알고리즘이 복잡하여 많은 수의 플로우들을 처리하여야 하는 실제 인터넷 환경에서는 적용하기 힘들며, 이의 대안으로 RED-PD, FRED, ARED 등의 RED를 변형한 여러 가지 알고리즘이 제안되었다. 그러나 이들 알고리즘들은 큐에서의 지연은 고려하지 않고 반응 플로우와 비반응 플로우간의 공정성을 개선하기 위한 방법만을 제시하고 있다.

본 논문에서는 큐에서의 비반응 플로우의 지연을 줄이면서 반응 플로우와의 불공정성을 개선할 수 있는 새로운 큐 관리 알고리즘인 PSRED(Protocol Sensitive RED) 알고리즘을 제안하였다. 제안한 PSRED 알고리즘은 RED 알고리즘의 기반에서 동작하며 반응 플로우와 비반응 플로우의 종류를 분리하여 처리함으로써 큐에서의 지연을 줄이고 불공정성을 해결할 수 있다. 또한 개별 플로우를 구분하지 않고 동작하므로 적은 오버헤드를 가지고 있다. 본 논문의 2장에서는 관련 연구에 대하여 기술하였고, 3장에서는 제안한 PSRED 알고리즘에 대하여 기술하였다. 4장에서는 모의실험을 통하여 제안 알고리즘의 성능을 검증하였으며 5장에서 논문의 결론과 향후 연구에 대하여 언급하였다.

2. 관련 연구

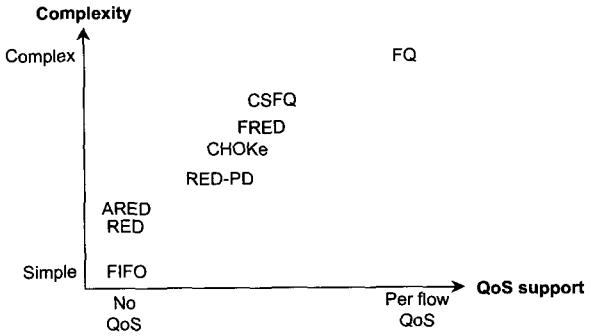
2.1 혼잡제어 및 공정성 관련 연구

인터넷의 혼잡제어와 관련된 연구는 수송계층 레벨에서의 혼잡제어 방법과 네트워크 레벨에서의 혼잡상황을 제어하는 두 가지 방향으로 진행되고 있다. 인터넷의 수송계층으로는 TCP와 UDP가 널리 사용되고 있다. 종단 시스템간의 흐름제어를 수행하는 TCP는 네트워크에서의 혼잡상황을 감지하여 데이터의 전송속도를 조절하며, 이 때 사용하는 혼잡제어 알고리즘에 따라 여러 가지 버전의 TCP가 제안되었다. 반면 UDP는 흐름제어를 수행하지 않고 정해진 속도로 데이터를 전송하며, 네트워크상의 혼잡상황을 고려한 혼잡제어를 수행하지 않는다. 최근에는 UDP와 같은 비반응 플로우에서도 네트워크의 혼잡상황을 적용하기 위한 TFRC(TCP Friendly Rate Control)[5] 와 같은 프로토콜이 제안되었으나, TCP 플로우와의 불공정성 문제 및 성능의 균일성 등의 문제가 해결과제로 남아있다[6].

최근에 진행되고 있는 인터넷에서의 혼잡제어와 관련된 연구는 네트워크 레벨에서 능동적 큐 관리(AQM : Active Queue Management) 알고리즘을 이용하여 네트워크로 유입되는 데이터의 양을 조절함으로써 혼잡상황을 제어하는 방향으로 많은 연구가 진행되고 있다. 큐 관리 알고리즘을 개선하는 방법은 네트워크 레벨에서 이루어지므로 컴퓨터와 같은 종단 시스템의 수정 없이 중간 노드만 수정하여 실행되며, 인터넷 QoS를 제공하기 위한 연구와도 관련될 수 있어 그 효용성이 매우 높다.

2.2 혼잡제어를 위한 큐 관리 알고리즘

TCP와 같은 반응 플로우와 UDP와 같은 비반응 플로우가 네트워크에서 경쟁하게 되면 혼잡상황을 적용하지 않고 데이터를 전송하는 비반응 플로우의 특성으로 인하여 반응 플로우의 성능이 저하되는 불공정성이 발생하며 혼잡상황이 더욱 가중된다. 불공정성을 해결하기 위해서는 큐의 복잡도가 증가하게 된다. 또한 QoS를 제공하기 위한 알고리즘이 추가되면 큐의 구조는 더욱 복잡하게 되어 실용성이 떨어진다. (그림 1)은 현재 연구 진행 중인 AQM의 복잡도와 QoS의 제공도를 나타낸 것이다.



(그림 1) AQM의 복잡도와 QoS 제공도

FIFO 큐는 가장 단순한 구조의 큐로서 현재 인터넷에서 가장 널리 사용되고 있다. 그러나 일단 큐에 데이터가 모두 차게 되면(full queue) 각 플로우의 특성을 고려하지 않고 패킷을 폐기함으로써 혼잡상황이 발생하게 된다. 큐의 크기를 크게 하여 FIFO의 효율을 높일 수 있는 방법[7]도 연구되고 있으나, 큐에서의 지연이 길어져 통신의 품질은 떨어진다.

이와는 반대로 가장 완벽하게 공정성을 보장할 수 있는 기법으로 FQ(Fair Queueing) 알고리즘이 있다. FQ는 모든 플로우에 대하여 개별적인 큐를 사용하므로 각 플로우에 대하여 공정한 대역폭을 할당할 수 있다. 그러나 모든 플로우에 대한 정보들을 수집하고 관리하여야 하기 때문에 플로우의 수가 많은 대용량 네트워크에서는 상대적으로 오버헤드가 커져 실용성이 떨어진다.

RED는 큐가 다 차기 전에 패킷을 확률적으로 폐기하여 데이터의 유입량을 조절함으로써 혼잡상황이 발생하는 것을 방지할 수 있다. RED는 복잡도가 낮고 반응 플로우에 대한 혼잡제어의 효과가 매우 우수하다. 그러나 알고리즘의 동작에 필요한 여러 개의 파라미터를 적절히 사용하여야 하는 문제[8]가 있다. Adaptive RED(ARED)[9]는 네트워크로 유입되는 트래픽에 따라 RED의 파라미터를 능동적으로 조절함으로써 실제 인터넷 환경에서 적합하도록 제안된 알고리즘이다. 그러나 RED와 ARED 알고리즘은 모두 반응 플로우와 비반응 플로우간의 불공정성 문제가 존재하며, 큐에서의 지연시간은 전혀 고려하지 않는다.

RED를 사용하여 공정성을 해결하기 위하여 제안된 FRED (Flow Random Early Drop)[10]는 현재 큐에 누적되어 있는 패킷의 플로우를 구분하여 처리함으로써 공정성을 개선한 알고리즘이다. 그러나 큐에 누적된 모든 패킷의 플로우에 대한 정보를 수집하고 관리하여야 하는 오버헤드를 가지고 있어 실제 구현하고 사용하기에는 어려움이 있다.

RED-PD(RED with Preferential Dropping)[11]는 일정 시간간격 동안 높은 대역폭을 사용하는 몇 개의 플로우를 구분하고 해당 플로우의 패킷을 우선 폐기함으로써 큐의 복잡도를 낮추고 반응 플로우와 비반응 플로우간의 공정성을 개선한 알고리즘이다. 그러나 알고리즘의 동작에 필요한 파라미터의 설정이 쉽지 않고, FRED와 마찬가지로 큐에서의 지연 시간을 고려하지 않아 통신의 품질을 보장하기 어렵다.

CSFQ(Core Stateless Fair Queueing)[12] 알고리즘은 중앙(core)의 라우터와 경계(edge) 라우터를 분리하여 많은 플로우가 집중되는 중앙 라우터에서는 플로우를 관리하지 않도록 함으로써 복잡도를 줄이면서 공정성을 개선할 수 있다. 그러나 경계 라우터는 여전히 오버헤드가 크며, 큐에서의 지연을 고려하지 않는 단점이 있다.

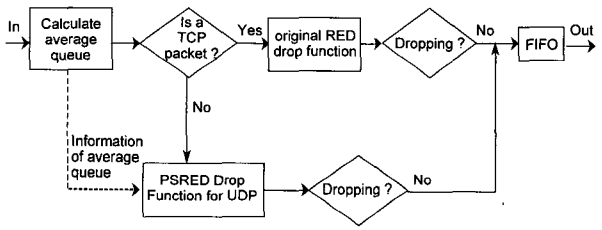
단일 큐를 사용하는 CHOKe(CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows)[13] 알고리즘은 높은 대역폭의 플로우는 큐에 많은 패킷이 누적되는 것에 착안하여 입력되는 패킷과 큐에 들어 있는 랜덤한 패킷을 비교하여 같은 플로우에 해당하면 폐기하고, 아니면 확률적으로 허락한다. FQ에 비하여 복잡도가 낮으면서도 높은 대역폭을 사용하는 패킷을 효과적으로 폐기하여 불공정성이 개선된다. 그러나 플로우의 수가 많아지면 높은 대역폭의 플로우도 적은 수의 패킷만이 큐에 누적되므로 그 효과가 제한적이다. 또한 큐에서의 지연 시간은 전혀 고려하지 않는다.

이밖에 인터넷에서 QoS를 제공하기 위한 접근 방법으로 DiffServ(Differentiated Service)[14]와 IntServ(Integrated Service)[15]가 가장 널리 사용되고 있다. AF(Assured Forwarding)나 EF(Expedited Forwarding)와 같은 DiffServ는 트래픽을 서비스별로 차별하여 처리함으로써 QoS를 제공한다. 그러나 서비스 레벨의 협상 과정이 복잡하고 트래픽 모니터기(traffic monitor), 서비스 분류기(classifier), shaper, 등의 DiffServ를 구성하기 위하여 필요한 요소들이 복잡하다.

RSVP(ReSource reSerVation Protocol)[16]로 대표되는 IntServ는 플로우별로 사용 가능한 대역폭을 보장하는 방법으로 가장 확실한 형태의 QoS를 제공하고 있다. 그러나 대역폭의 할당을 위한 호처리(signaling)가 복잡하고 모든 라우터가 각 플로우에 대한 정보를 관리하여야 하므로 라우터의 오버헤드가 높아진다.

3. PSRED 알고리즘

본 논문에서는 인터넷의 혼잡제어를 위한 새로운 큐 관리 알고리즘인 PSRED(Protocol Sensitive RED) 알고리즘을 제안한다. PSRED 알고리즘은 큐에 유입되는 패킷을 반응 플로우와 비반응 플로우로 구분하여 차별적으로 RED 알고리즘을 적용함으로써 비반응 플로우에 대한 큐에서의 지연시간을 줄임과 동시에 반응 플로우와 비반응 플로우간의 공정성을 개선한다. 반응 플로우와 비반응 플로우를 구분하기 위하여 IP 헤더의 프로토콜 필드가 TCP인 경우에는 반응 플로우로 판별하고, UDP인 경우에는 비반응 플로우로 판별한다. (그림 2)는 PSRED 알고리즘의 구조를 나타낸 것이다.



(그림 2) PSRED 알고리즘의 구조

3.1 PSRED의 패킷 폐기 정책

PSRED는 큐에 입력된 패킷의 종류에 따라 2 가지의 다른 패킷 폐기함수를 적용하여 혼잡제어를 수행한다. TCP 패킷의 경우에는 원래의 RED 알고리즘에 의하여 RED 패킷 폐기함수에 따라 확률적으로 패킷을 폐기한다. TCP 데이터가 입력된 시점의 현재 큐 길이를 q, 현재 평균 큐 크기를 q_{avg}라 하면 RED 알고리즘의 패킷 폐기함수 $d_{TCP}(q)$는 다음의 식 (1)과 같다.

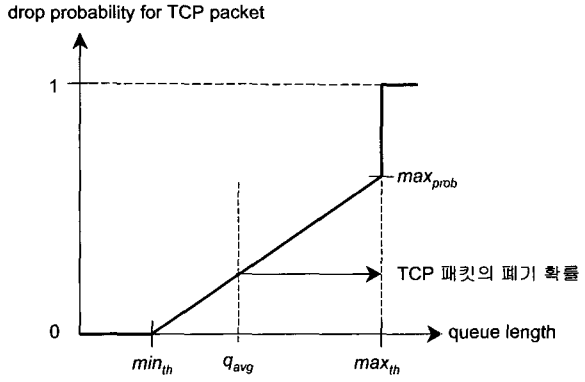
$$d_{TCP}(q) = \begin{cases} \dfrac{(q_{avg} - \min_{th})}{(\max_{th} - \min_{th})} \cdot \max_{prob} & \text{if } \min_{th} \le q_{avg} < \max_{th}, \\ 0 & \text{if } q_{avg} < \min_{th}, \\ 1 & \text{if } q_{avg} \ge \max_{th}. \end{cases} \quad (1)$$

식 (1)에서 $\min_{th}$와 $\max_{th}$는 RED 알고리즘의 패킷 폐기

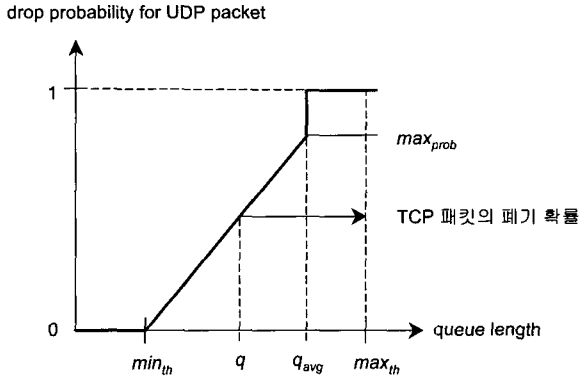
확률을 결정하는 최소, 최대 임계값이며, $\max_{prob}$는 폐기 특성을 결정하는 최대 확률 값이다. 큐에 입력된 데이터가 UDP 패킷인 경우에는 현재 평균 큐 크기인 q_{avg}와 현재 큐 길이인 q에 따라 동작한다. UDP 데이터에 적용되는 PSRED 알고리즘의 패킷 폐기 함수 $d_{UDP}(q)$는 다음의 식 (2)와 같다.

$$d_{UDP}(q) = \begin{cases} \dfrac{(q - \min_{th})}{(q_{avg} - \min_{th})} \cdot \max_{udp} & \text{if } \min_{th} \le q < q_{avg}, \\ 0 & \text{if } q < \min_{th}, \\ 1 & \text{if } q \ge q_{avg}. \end{cases} \quad (2)$$

식 (2)에서 $\min_{th}$와 $\max_{th}$는 RED 알고리즘과 동일한 값을 사용하며, $\max_{udp}$는 UDP 패킷의 폐기 특성을 결정하는 최대 확률 값이다. (그림 3)은 PSRED 큐에 입력되는 TCP 데이터와 UDP 데이터의 폐기 함수를 그래프로 표현한 것이다.



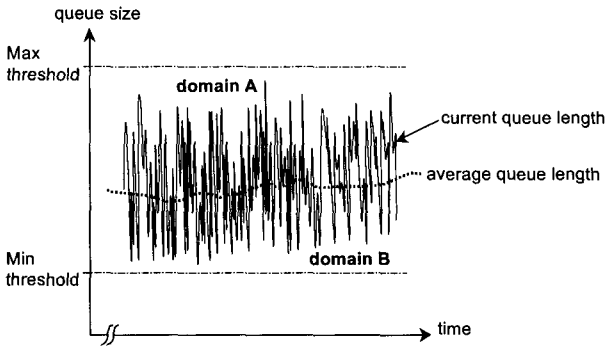
(a) TCP 패킷의 폐기 함수



(b) UDP 패킷의 폐기 함수

(그림 3) PSRED의 패킷 폐기 함수

이러한 패킷 폐기 함수를 적용하게 되면 현재의 평균 큐 크기와 큐 길이에 따라 TCP 데이터와 UDP 데이터가 폐기 함수를 통과하여 큐에 입력되는 영역이 서로 구분된다. (그림 4)는 PSRED 큐의 동작과 데이터 입력 영역을 나타낸 것이다.



(그림 4) PSRED의 동작

UDP 패킷은 큐에 입력되는 시점에서의 큐 길이가 평균 큐 크기보다 큰 경우에는 폐기확률 1이 적용되므로 A 영역에는 TCP 패킷만 입력되며 RED 알고리즘의 폐기함수가 적용된다. 큐에 데이터가 입력되는 시점의 큐 길이가 평균 큐 크기보다 작은 B 영역에서는 TCP 패킷과 UDP 패킷이 모두 입력될 수 있으며, 각각 다른 패킷 폐기함수가 적용된다.

UDP 패킷은 큐에 입력될 시점에서의 큐 길이가 평균 큐 크기보다 작은 경우에만 입력되므로 큐에서 대기하는 지연시간이 감소한다. 또한 TCP 플로우에 비하여 패킷이 폐기되는 확률을 높일 수 있어 공정성을 개선할 수 있다. 프로토콜에 따라 차별적으로 동작하는 PSRED의 패킷 폐기확률 알고리즘은 (그림 5)와 같다.

```
For Each Packet Arrival
    Calculate the Average Queue Size q_avg
    Identify the protocol filed P_proto
if ( p_proto ≡ TCP)
    Calculate the Packet Drop Probabiliry P_TCP(q)
else if ( p_proto ≡ UDP)
    Calculate the Packet Drop Probabiliry P_UDP(q)
```

(그림 5) PSRED 패킷 폐기 알고리즘

3.2 PSRED 알고리즘의 공정성 개선 효과 분석

식 (3)은 평균 손실률(average loss rate) l을 갖는 네트워크에서 TCP가 사용하는 평균 윈도우 크기(average window size : w)를 나타내는 근사식으로 Floyd나 Mathis 등의 연구에서 채택된 식이다[17, 18].

$$w = \frac{0.87}{\sqrt{l}} \quad (3)$$

TCP는 네트워크에서의 패킷 손실에 따라 데이터 전송 시에 사용하는 윈도우 크기를 조절한다. 위의 식은 네트워크에서의 평균 손실률을 이용하여 TCP가 사용하는 평균 윈도우 크기를 계산할 수 있음을 의미한다. 이를 식 (4)로 표현하면 TCP가 사용하는 평균 윈도우 크기를 이용하여 네트워크에서의 평균 손실률을 계산할 수 있다.

$$l = \frac{0.76}{w^2} \tag{4}$$

큐의 길이가 S 패킷인 병목링크를 N_{tcp}개의 TCP 플로우가 공유해서 사용하는 경우에는 각 TCP의 윈도우 크기를 합한 값은 S와 같다. 즉 $w = S/N_{tcp}$로 표현할 수 있다. 병목링크가 사용하는 큐가 RED인 경우, 평균 큐 크기가 q_{avg}로 안정된 상태에서는 $w = q_{avg}/N_{tcp}$가 된다. 이를 식 (4)에 대입하면 RED 라우터에서 TCP의 평균 윈도우 크기를 이용하여 네트워크의 평균 손실률을 계산할 수 있는 식을 구할 수 있다.

$$l = 0.76 \cdot \frac{N_{tcp}^2}{q_{avg}^2} \tag{5}$$

PSRED 라우터에 N_{tcp}개의 TCP 플로우와 N_{udp}개의 UDP 플로우의 데이터가 동시에 유입되고, 각 플로우들의 성능을 공정하게 처리하면서 평균 큐 크기가 q_{Afair}에서 안정되었다고 가정하면, PSRED 라우터의 평균 큐 크기 q_{Afair}를 각 플로우들이 균등하게 사용하여야 한다. 즉 각 TCP가 사용하는 윈도우 크기는 다음의 식으로 표현할 수 있다.

$$w = \frac{q_{Afair}}{N_{tcp} + S_r \cdot N_{udp}} \tag{6}$$

위 식에서 S_r은 UDP 와 TCP의 패킷 크기비율을 나타내는 값으로 TCP 패킷의 크기를 S_{tcp}, UDP 패킷의 크기를 S_{udp}라 하면 $S_r = S_{tcp}/S_{udp}$로 표시된다. 식 (6)을 식 (4)에 대입하면 N_{tcp}개의 TCP 플로우와 N_{udp}개의 UDP 플로우가 공정한 성능을 유지한다고 가정했을 때의 TCP 데이터의 평균 손실률이 계산된다.

$$l = 0.76 \cdot \frac{(N_{tcp} + S_r \cdot N_{udp})^2}{q_{Afair}^2} \tag{7}$$

(그림 4)의 A 영역, 즉 데이터가 평균 큐 크기 q_{Afair} 보다 큰 영역에서 큐에 유입되는 경우에는 TCP 데이터만이 큐이 들어올 수 있으며, 이 때에는 원래의 RED 알고리즘에 의하여 TCP 데이터가 처리된다. A 영역에서의 TCP 데이터의 손실률은 RED의 패킷 폐기 함수에 따르며, 이를 근사식으로 표현하면 다음의 식 (8)과 같다.

$$l_A = \frac{q_{Afair}}{max_{th}} \cdot max_{prob} \tag{8}$$

현재의 큐 길이가 q_{Afair} 보다 작은 B 영역에서는 TCP 데이터와 UDP 데이터가 모두 큐에 유입될 수 있다. 따라서 B 영역에서의 TCP 데이터의 손실률은 폐기함수를 통과하여 큐에 유입된 UDP 데이터에 의하여 증가하게 되며 다음의 식 (9)와 같이 표현할 수 있다.

$$l_B = \frac{q_{Afair}}{max_{th}} \cdot max_{prob} \cdot \gamma \tag{9}$$

위 식에서 γ는 UDP 데이터에 의한 TCP 데이터의 손실률의 증가비를 나타내는 것으로 UDP 플로우의 개수 N_{udp} 및 UDP 데이터의 통과율에 비례한다. UDP 데이터의 통과율은 $1 - d_{udp}(q)$로 표현할 수 있으므로 γ는 상수 σ를 이용하여 다음의 식 (10)과 같이 표현할 수 있다.

$$\gamma = N_{udp} \cdot \left(1 - \frac{q - min_{th}}{q_{Afair} - min_{th}} \cdot max_{udp}\right) \cdot \sigma \tag{10}$$

PSRED 라우터에서 TCP 데이터의 평균 손실률은 A 영역과 B 영역 손실률의 평균값이므로 $l = 1/2(l_A + l_B)$이 된다. B 영역에서의 큐 길이 q는 q_{Afair}와 min_{th} 사이에 존재하므로 평균적으로 $1/2 \cdot (q_{Afair} - min_{th})$의 값을 나타낸다고 가정하면 PSRED 라우터에서 TCP 데이터의 평균 손실률은 다음과 같은 근사식으로 표현할 수 있다.

$$l = \frac{1}{2} \cdot \frac{q_{Afair}}{max_{th}} \cdot max_{prob}\left(1 + N_{udp} \cdot \left(1 - \frac{max_{udp}}{2}\right) \cdot \sigma\right) \tag{11}$$

식 (11)과 식 (7)에서 손실률 l을 제거하면 PSRED 라우터에 N_{tcp}개의 TCP 플로우와 N_{udp}개의 UDP 플로우에 공정한 성능을 제공할 수 있는 q_{Afair}를 다음의 식 (12)와 같이 표현할 수 있다.

$$q_{Afair} = 1.15 \cdot \frac{(N_{tcp} + S_r \cdot N_{udp})^{2/3} \cdot max_{th}^{1/3}}{max_{prob}^{1/3} \cdot \left(1 + N_{udp} \cdot \left(1 - \frac{max_{udp}}{2}\right) \cdot \sigma\right)^{1/3}} \tag{12}$$

PSRED 라우터는 큐의 B 영역에서 UDP 패킷의 폐기 특성을 결정하는 최대 확률 값 max_{udp}을 조절함으로써 반응 플로우와 비반응 플로우의 성능을 제어할 수 있다. 이 때 사용하는 max_{udp}의 값에 따라 각 플로우간의 공정성을 개선할 수 있으며, max_{udp}의 값은 TCP와 UDP의 플로우 수에 의해 결정된다. 식 (12)는 N_{tcp}개의 TCP 플로우와 N_{udp}개의 UDP 플로우가 PSRED 라우터에 유입되는 경우, 각 플로우에 공정한 성능을 제공할 때의 max_{udp}값과 이 때의 평균 큐 크기 q_{Afair}와의 관계를 표시하고 있다. 위 식에서 UDP 패킷에 대한 최대 폐기확 max_{udp}값이 클수록 큐에 입력되는 TCP 데이터가 많아지며 이에 따라 평균 큐 크기가 커지는 것을 알 수 있다.

3.3 공정성 향상을 위한 비반응 플로우의 최대 폐기확률 : max_{udp} 조절 알고리즘

PSRED 알고리즘을 사용하여 반응 플로우와 비반응 플로우간의 공정한 성능을 제공하기 위해서는 UDP 패킷의 최대 폐기확률 max_{udp}의 값을 적절하게 설정하여야 한다. max_{udp}의 값은 최소 0에서 최대 1의 값을 갖으며, PSRED 큐에 유입되는 TCP와 UDP의 플로우 수 및 데이터의 양에 따라 적절한 값으로 설정되어야 한다. 또한 트래픽의 특성이 변경되면 이에 맞는 값으로 능동적으로 변경되어야 한다.

공정성을 제공하기 위한 max_{udp}의 값을 결정하기 위하여 PSRED 큐는 max_{udp} 조절 알고리즘을 수행한다. max_{udp} 조절 알고리즘은 임의의 시간간격 ΔT 동안 큐의 A 영역에서 유입되는 TCP 패킷의 수와 폐기되는 패킷의 수를 측정한다. PSRED 큐의 A 영역은 데이터가 입력되는 시점의 큐 길이가 평균 큐 크기보다 큰 영역으로 TCP 패킷만이 입력될 수 있다. 이렇게 측정된 TCP 패킷의 수를 TCP_{inA}라 하고, 폐기된 TCP 패킷의 수를 TCP_{dropA}라 하면 A 영역에서의 TCP 데이터의 손실률 l_A은 다음의 식 (13)과 같이 계산할 수 있다.

$$l_A = \frac{TCP_{dropA}}{TCP_{inA}} \tag{13}$$

다음으로 현재의 평균 큐 크기 q_{avg}를 측정한다. 만일 현재 시스템에 설정되어 있는 max_{udp}가 모든 플로우들의 성능을 공정하게 처리할 수 있는 값을 갖는다고 가정하면 식 (8) 에 의하여 A 영역에서의 TCP 데이터의 손실률 l_{Aestm}을 계산할 수 있다.

$$l_{Aestm} = \frac{q_{avg}}{max_{th}} \cdot max_{prob} \tag{14}$$

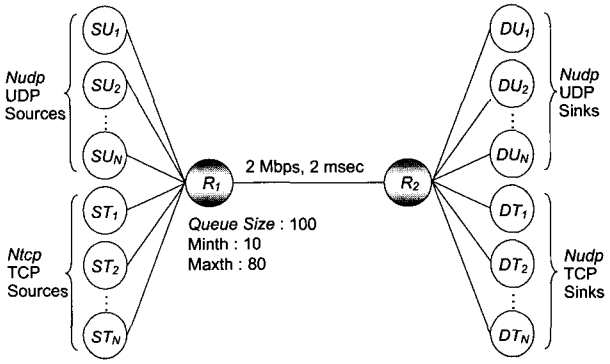
앞에서 측정한 A 영역의 TCP 데이터 손실률 l_A과 현재 큐 크기를 이용하여 계산한 l_{Aestm}의 값을 이용하면 현재 시스템에 설정된 max_{udp}의 값을 적절하게 조절할 수 있다. 현재 사용하고 있는 max_{udp}의 값이 커서 UDP 패킷에 대한 폐기 확률이 높다고 가정하면 상대적으로 TCP 데이터의 손실률 l_A은 낮아지게 된다. 또한 max_{udp}의 값이 크면 식 (12)에 의하여 평균 큐 크기가 커지게 되며 식 (14)에 의하여 공정성을 제공할 때의 A 영영의 TCP 데이터 손실률 l_{Aestm}의 값도 커진다. 따라서 $l_A < l_{Aestm}$인 경우에는 max $_{udp}$의 값을 감소시키고, 반대로 $l_A > l_{Aestm}$ 이면 max_{udp}의 값을 증가시켜야 공정한 성능을 제공할 수 있다. PSRED 큐에서 사용하는 max_{udp} 조절 알고리즘은 (그림 6)과 같다.

```
Every interval seconds : // 500 ms, by default
    Calculate l_A and l_Aestm
    If (l_A < l_Aestm)
        decrease max_udp :
            max_udp -= α // about 0.05, by default
    elseIf (l_A > l_Aestm)
        increase : max_udp
max_udp += β // about 0.05, by default
```

(그림 6) max_{udp} 조절 알고리즘

4. 모의실험 및 성능평가

본 장에서는 새로운 큐 관리 알고리즘인 PSRED 알고리즘의 성능을 평가하기 위하여 모의실험을 실시하였다. 모의실험은 ns-2(Network Simulator)[19]를 이용하였으며, 반응 플로우와 비반응 플로우간의 공정성의 개선 및 비반응 플로우의 큐 지연 시간의 감소 등을 평가하였다. 모의실험에서 사용한 네트워크 환경은 다음의 (그림 7)과 같다.



(그림 7) 네트워크 모의실험 환경

모의실험 환경은 N_{udp}개의 비반응 플로우와 N_{tcp}개의 반응 플로우가 라우터 *R1*과 *R2* 사이의 병목링크를 공유하여 사용하도록 구성하였다. 반응 플로우로는 TCP를 사용하였으며 트래픽은 인터넷에서 가장 많은 양의 데이터를 발생시키는 FTP를 사용하였다. 비반응 플로우로는 UDP를 사용하였으며 트래픽은 CBR(Constant Bit Rate) 타입을 사용하였다. CBR 트래픽의 패킷 크기는 500Byte, 전송률은 1024Kbps로 설정하였다.

4.1 반응 플로우와 비반응 플로우간의 공정성 실험

본 논문에서 제안한 PSRED 알고리즘의 반응 플로우와 비반응 플로우간의 공정성 개선 효과를 실험하기 위하여 (그림 7)의 라우터 *R1*에 DropTail과 RED, RED-PD, 그리고 PSRED 알고리즘을 차례대로 적용하여 실험하였다. 비반응 플로우의 수는 1, 2, 4, 그리고 8로 하고 반응 플로우의 수를 11에서 30까지 증가시키면서 각 플로우의 성능을

측정하였다. 또한 각 모의실험에서의 공정성을 측정하기 위하여 Jain이 제안한 공정지수(fairness index)[20]를 사용하였다. 공정지수 F는 다음의 식으로 계산된다.

$$F(Fairness\ Index) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N\sum_{i=1}^{N} x_i^2} \quad (15)$$

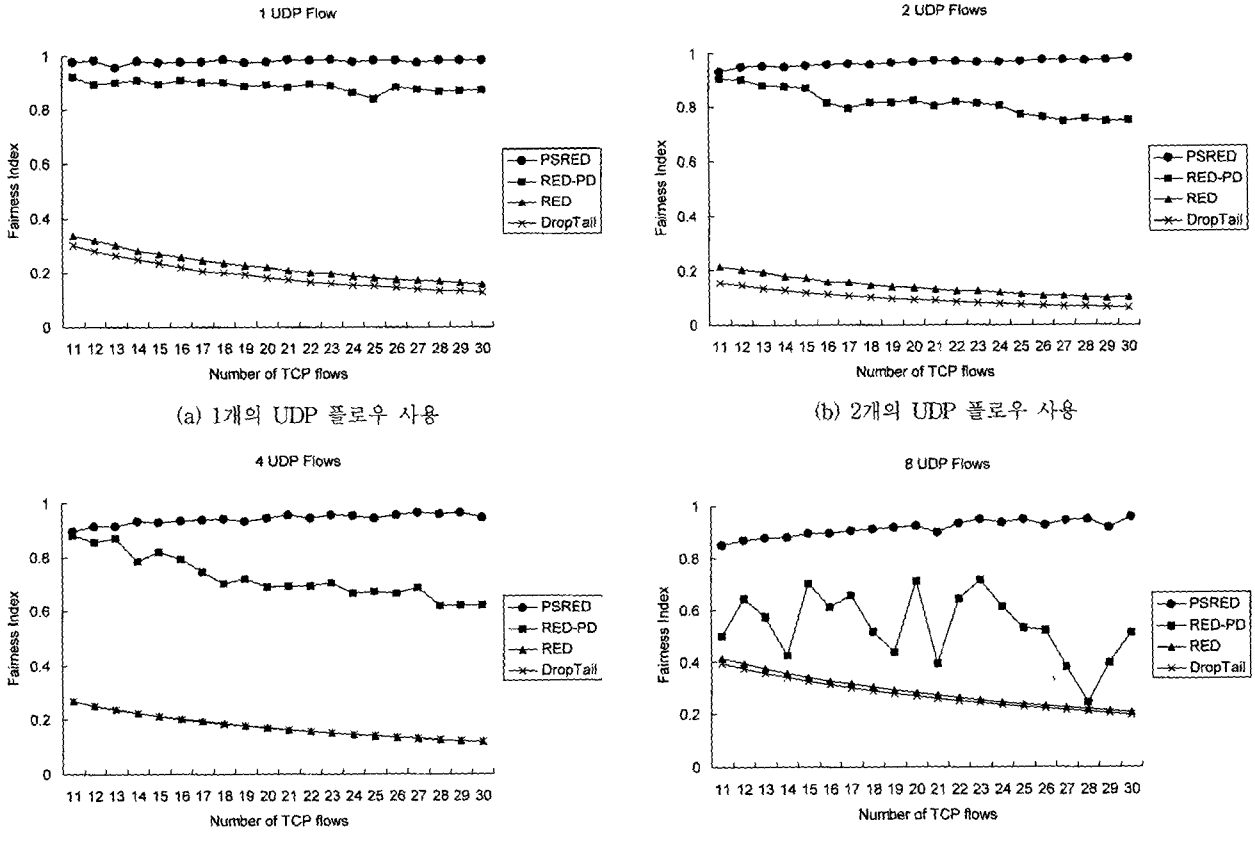
위의 식에서 N은 플로우의 수를 나타내며 x_i는 i번째 플로우의 성능을 표시한다. 공정지수의 값은 0에서 1 사이의 값을 가지며, 1에 가까울수록 각 플로우의 성능이 공정함을 나타낸다. (그림 8)은 라우터 $R1$에 DropTail과 RED, RED-PD, 그리고 PSRED 알고리즘을 적용하여 실험하였을 때의 공정지수를 표시한 것이다.

DropTail 큐를 사용한 경우에는 반응 플로우와 비반응 플로우가 병목링크를 공유해서 사용하게 되면 패킷의 종류에 관계없이 패킷이 폐기되므로 패킷의 폐기에 반응하여 전송률을 낮추는 TCP 플로우의 성능이 저하된다. 또한 TCP와 UDP 플로우의 수가 증가할수록 큐에서 폐기되는 TCP 패킷의 수가 증가하게 되고 TCP 플로우의 전송률은 더욱 낮아지게 되어 TCP 플로우와 UDP 플로우간의 공정지수가 저하된다.

RED 큐를 사용한 경우에는 RED 알고리즘의 혼잡제어로 인하여 DropTail 큐에 비하여 공정지수가 증가함을 알 수 있다. 그러나 RED를 사용한 경우에도 TCP 플로우가 증가하면 공정지수는 저하된다. 이는 RED 알고리즘의 혼잡제어 동작에도 불구하고 트래픽의 증가에 따라 평균 큐 크기가 증가하게 되고 TCP 패킷에 대한 폐기확률이 증가하기 때문이다.

DropTail과 RED 큐의 경우 UDP 플로우의 수가 1에서 2로 증가하면 공정지수가 감소한다. 이는 UDP 플로우에 의하여 TCP 플로우의 성능이 더욱 하락하기 때문이다. 그러나 UDP 플로우의 수가 4, 8로 증가하면 UDP 플로우 간의 공정한 성능으로 인하여 공정지수는 다시 증가한다. DropTail이나 RED 큐의 경우 UDP와 같은 비반응 플로우간에는 공정하게 높은 성능을 나타낸다. 따라서 UDP 플로우의 수가 증가하면 TCP 플로우의 성능은 감소하지만 UDP 플로우 간의 공정한 성능으로 인하여 공정지수 계산 값은 증가하게 된다.

RED-PD 알고리즘을 사용한 경우에는 일정 시간간격 동안 높은 대역폭을 사용하는 몇 개의 플로우를 구분하고 해당 플로우의 패킷을 우선 폐기함으로써 DropTail이나 RED 큐에 비하여 공정성이 개선됨을 알 수 있다. 그러나 플로우의 수가 증가할수록 일정 시간간격 동안 높은 대역폭을 사



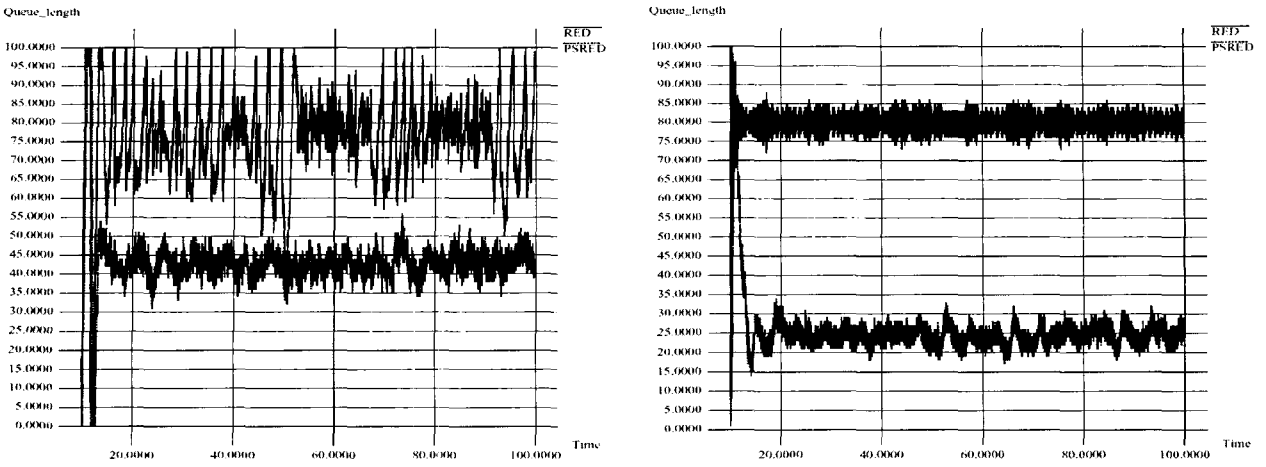
(a) 1개의 UDP 플로우 사용

(b) 2개의 UDP 플로우 사용

(c) 4개의 UDP 플로우 사용

(d) 8개의 UDP 플로우 사용

(그림 8) TCP와 UDP 플로우의 공정지수(Fairness Index) 실험



(a) 1 UDP와 20 TCP 플로우 (b) 8 UDP와 20 TCP 플로우

(그림 9) RED와 PSRED의 큐 길이 측정

용하는 플로우를 감지하는 확률이 감소하게 되어 공정지수는 감소한다.

PSRED 알고리즘의 경우 큐에 입력되는 트래픽의 특성에 따라 UDP 패킷에 대한 최대 폐기확률 max_{udp}의 값을 자동으로 조절함으로써 각 플로우간의 공정성이 다른 알고리즘에 비하여 크게 개선됨을 확인할 수 있다. 특히 반응 플로우의 수가 증가할수록 공정지수가 증가하는 결과를 보이고 있는데, 이는 반응 플로우의 수가 비반응 플로우의 수보다 큰 경우, 즉 $N_{udp} < N_{tcp}$이면 평균 큐 크기가 min_{th}와 max_{th}값 사이에서 유지됨으로써 A영역에서 반응 플로우의 폐기율에 따라 공정한 성능을 제공할 수 있는 max_{udp}의 값을 결정하기가 용이하기 때문이다.

4.2 평균 큐 크기와 큐 지연 시간

UDP와 같은 비반응 플로우의 경우 인터넷 폰이나 인터넷방송과 같이 네트워크에서의 지연이 작아야 하는 멀티미디어 통신에서 주로 사용된다. 따라서 큐에서의 지연 시간이 통신의 품질을 결정하는 중요한 요소가 된다. 큐 지연시간은 데이터가 큐에 입력된 시간에서부터 큐에서 출력될 때까지 데이터가 큐에서 대기하는 데에 소요되는 시간이다. 데이터가 큐에 입력되는 시점에서의 큐 길이를 Q_l이라 하고 큐에 대기하고 있는 데이터의 길이를 S_r, 큐에 연결되어 있는 링크의 전송 대역폭을 BW라 하면 큐 지연시간 τ_q는 다음의 식으로 표현된다.

$$\tau_q = Q_l \cdot \frac{S_r}{BW} \tag{16}$$

위의 식에서 링크의 대역폭과 데이터의 길이가 정해진 환경에서의 큐 지연시간을 결정하는 가장 중요한 요인은 큐 길이임을 알 수 있다.

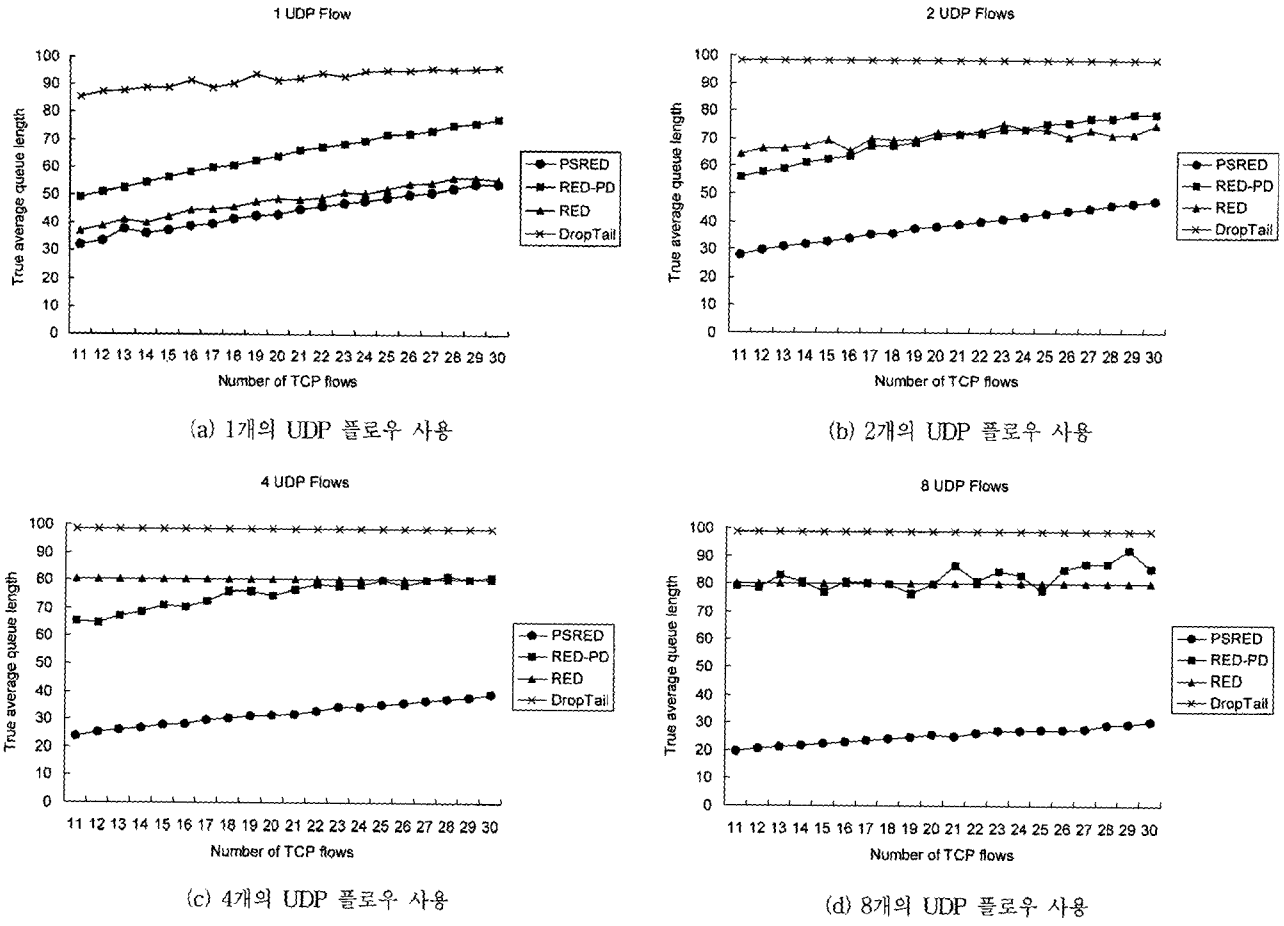
(그림 9)는 (그림 7)의 모의실험 환경에서 라우터 $R1$에 RED와 PSRED 알고리즘을 적용하고 20개의 TCP 플로우와 1 및 8개의 UDP 플로우를 부가한 경우에 각각의 큐 길이를 측정한 결과이다.

그림에서 RED 보다 PSRED 알고리즘을 사용한 경우에 큐 길이가 낮게 나타나는 것을 확인할 수 있다. 즉 PSRED 알고리즘을 사용하면 RED 알고리즘을 사용한 경우와 비교하여 큐 지연시간이 감소되는 것을 알 수 있다.

본 논문에서 제안한 PSRED 알고리즘의 큐 지연시간을 다양한 환경에서 평가하기 위하여 (그림 7)의 라우터 $R1$에 DropTail과 RED, RED-PD, 그리고 PSRED 알고리즘을 차례대로 적용하여 실험하였다. 비반응 플로우의 수는 1, 2, 4, 그리고 8로 하고 반응 플로우의 수를 11에서 30까지 증가시키면서 각 알고리즘의 실제 평균 큐 길이(true average queue length)를 측정하였다.(그림 10)은 라우터 $R1$에 Drop Tail과 RED, RED-PD, 그리고 PSRED 알고리즘을 적용하여 실험하였을 때의 실제 평균 큐 길이를 표시한 것이다.

병목링크가 연결되어 있는 Drop-tail 라우터에 UDP와 같은 비반응 플로우가 유입되면 큐에 데이터가 지속적으로 누적되어 평균 큐 길이가 커지며 이에 따라 큐 지연시간도 같이 늘어난다. 특히 병목링크의 대역폭보다 많은 양의 UDP 데이터가 유입되는 경우에는 평균 큐 길이가 최대로 증가하게 되어 큐에서의 지연시간이 최대로 늘어난다. 모의실험 환경의 경우 2개 이상의 UDP 플로우가 유입되면 UDP의 전송률이 링크의 대역폭을 넘어 평균 큐 길이가 최대 큐 크기인 100이 된다.

RED 라우터에 비반응 플로우가 유입되면 혼잡제어 동작에 영향을 받지 않고 동일한 성능으로 전송하는 UDP 데이터에 의하여 평균 큐 크기가 증가하며 이에 따라 큐 지연시간도 같이 길어진다. RED의 경우 병목링크의 대역폭보다 많은 양의 UDP 데이터가 유입되는 경우에는 평균 큐 길이가 $\max_{th}$와 비슷해진다. 위의 실험에서는 2개 이상의 UDP 플로우가 유입되면 UDP의 전송률이 링크의 대역폭을 넘어



(a) 1개의 UDP 플로우 사용

(b) 2개의 UDP 플로우 사용

(c) 4개의 UDP 플로우 사용

(d) 8개의 UDP 플로우 사용

(그림 10) 실제 평균 큐 길이(True average queue length) 비교

평균 큐 길이는 max_{th}인 80이 된다.

RED-PD 라우터 역시 큐의 기본 동작은 RED 알고리즘을 사용하므로 많은 양의 UDP 데이터가 유입될수록 평균 큐 길이는 max_{th}에 가까워진다. 위의 실험 결과에서는 2개 이상의 UDP 플로우가 유입되면 UDP의 전송률이 링크의 대역폭을 넘어 평균 큐 길이는 max_{th}인 80과 비슷해진다.

본 논문에서 제안한 PSRED 알고리즘의 경우 각 플로우에 공정한 성능을 제공할 때의 평균 큐 크기는 식 (12)와 같이 표현되며, 모의실험 결과 Drop-tail이나 RED, RED-PD에 비하여 낮게 측정되었다. 이는 반응 플로우와 비반응 플로우를 구분하고 각 플로우의 수에 따라 max_{udp}값을 조절하여 공정성을 유지할 수 있도록 UDP 데이터의 유입을 억제함으로써 평균 큐 크기가 낮아지기 때문이다. 따라서 PSRED 알고리즘을 사용하면 큐에서의 지연시간을 줄일 수 있으며, 비반응 플로우를 사용하는 멀티미디어 통신의 품질을 높일 수 있음을 확인하였다.

5. 결 론

인터넷에서의 혼잡제어는 그 알고리즘에 따라 통신의 품질이나 성능에 많은 영향을 끼칠 수 있는 매우 중요한 기능이다. 최근에는 RED와 같이 라우터의 큐에서 실행되는 큐 관리 알고리즘을 통한 혼잡제어 방법에 대한 연구가 많이 진행되고 있다. 그러나 반응 플로우와 비반응 플로우간의 불공정성 문제로 인하여 혼잡상황이 악화되는 문제를 가지고 있다. 또한 이를 해결하기 위하여 제안된 다른 방법들도 불공정성을 근본적으로 해결하지 못하거나 라우터의 복잡도가 증가하는 문제를 갖고 있다. 특히 멀티미디어 통신에서 주로 사용하고 있는 비반응 플로우의 경우 전송 효율보다는 네트워크에서의 지연시간이 통신의 품질을 결정하는 중요한 요소가 되고 있으나 대부분의 큐 관리 알고리즘에서는 큐에서의 지연시간을 고려하고 있지 않고 있다.

본 논문에서는 반응 플로우와 비반응 플로우간의 공정성을 개선함과 동시에 비반응 플로우에 대한 큐에서의 지연시간을 줄일 수 있는 새로운 큐 관리 알고리즘인 PSRED 알고리즘을 제안하였다. PSRED 알고리즘은 데이터의 프로토콜 필드를 이용하여 반응 플로우와 비반응 플로우를 구분하고 각기 다른 패킷 폐기확률을 적용함으로써 공정성을 개선하고 평균 큐 길이를 줄일 수 있다. PSRED 알고리즘은 개별 플로우 정보를 수집하지 않아 적은 오버헤드를 가

지고 있으며 구현이 용이한 장점이 있다. 제안 알고리즘의 모의실험 결과 기존의 Drop-Tail이나 RED, 혹은 RED-PD 알고리즘에 비하여 공정성이 개선되고 큐 지연시간이 감소되는 것을 확인하였다.

향후 연구과제로는 다양한 트래픽 환경에서도 완벽한 동작이 되도록 알고리즘을 개선하는 연구가 필요하며, 인터넷의 서비스 질(QoS)을 제공하기 위한 연구에도 활용할 수 있도록 알고리즘의 기능을 추가하는 연구를 진행하여야 할 것이다.

참 고 문 헌

[1] Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas : End to End Congestion Avoidance on a Global Internet," *IEEE JSAC*, Vol.13, pp.1465-1480, 1995.

[2] J. Mo, R. J. La, V. Anantharam and J. Walrand, "Analysis and Comparison of TCP Reno and Vegas," *IEEE INFOCOM '99*, pp.1556-1563, 1999.

[3] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Vol.1, No.4, pp.397-413, August, 1993.

[4] Raghavendra, A. M. and Kinicki, R. E., "A Simulation Performance Study of TCP Vegas and Random Early Detection," *Decision and Control, 2000. Proceedings of the 39th IEEE Conference*, Vol.1, pp.61-66, Dec., 2000.

[5] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC) : Protocol Specification," *RFC 3448*, Jan., 2003.

[6] Jin Tian, Sheng Xiangzhi and Wu Wenjun, "The Effect on the Inter-fairness of TCP and TFRC by the Phase of TCP Traffics," *Computer Networks and Mobile Computing 2001 International Conference*, pp.131-136, Oct., 2001.

[7] G. Iannaccone, M. May and C. Diot, "Aggregate Traffic Performance with Active Queue Management and Drop from Tail," *ACM Computer Communication Review*, July, 2001.

[8] G. Hasegawa, K. Kurata and M. Murata, "Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet," *Network Protocols, 2000 International Conference*, pp.14-17, Nov., 2000.

[9] Sally Floyd, Ramakrishna Gummadi and Scott Shenker, "Adaptive RED : An Algorithm for Increasing the Robustness of RED's Active Queue Management," Under submission, http://www.icir.org/floyd/papers/adaptiveRed.pdf, 2001.

[10] D. Lin and R. Morris, "Dynamics of Random Early Detection," *ACM SIGCOMM* 97, pp.127-137, Oct., 1997.

[11] R. Mahajan and S. Floyd, "Controlling High-Bandwidth Flows at the Congested Router," *ACIRI*, Berkeley, California, Nov., 2000.

[12] I. Stoica, S. Shemker and H. Zhang, "Core-Stateless Fair Queueing : Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," *Proceedings of ACM SIGCOMM 98*, Aug., 1998.

[13] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation," *Proceedings of INFOCOM 2000*, Feb., 2000.

[14] Y. Bernet, J. Binder, S. Blake, M. Carson, et. Al., "A Framework for Differentiated Services," *Internet Draft*, October, 1998.

[15] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture : An Overview," *RFC1633*, June, 1994.

[16] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," *IETF RFC 2210*, Sep., 1997.

[17] R. Morris, "Scalable TCP Congestion Control," *IEEE INFOCOMM 2000*, 2000.

[18] S. Floyd, "Connections with Multiple Congested Gateways in Packet Switched Networks Part 1 : One-way Traffic," *Computer Communications Review*, Vol. 21, No. 5, Oct., 1991.

[19] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/, Nov., 2000.

[20] Raj Jain, "Throughput Fairness Index : An explanation," *ATM Forum Contribution 99-0045*, Feb., 1999.

채 현 석

e-mail : hschae@tongwon.ac.kr

1987년 한양대학교 공과대학 전자공학과 학사

1990년 한양대학교 대학원 전자공학과 석사

1990년~1999년 LG정보통신 중앙연구소 선임연구원

1998년~현재 한양대학교 전자컴퓨터공학부 박사과정

1999년~현재 동원대학 인터넷정보과 조교수

관심분야 : 큐 관리 알고리즘, TCP 성능제어, QoS 라우팅, 고성능 라우터

최 명 렬

e-mail : choimy@asic.hanyang.ac.kr

1983년 한양대학교 공과대학 전자공학과 학사

1985년 미시간 주립대학 전기공학과 석사

1991년 미시간 주립대학 전기공학과 박사

1991년~1992년 생산기술연구원산하 전자부품종합기술연구소 선임연구원

1992년~현재 한양대학교 전자컴퓨터공학부 교수

관심분야 : 스마트카드, DSP 응용, RF/IR 통신, ITS, LCD, ATM/internet