이중 큐 CHOKe 방식을 사용한 혼잡제어

준회원 최 기 현*, 준회원 신 호 진*, 정회원 신 동 렬*

Double Queue CHOKe Mechanism for Congestion Control

Kee-Hyun Choi*, Ho-Jin Shin*, Dong-Ryeol Shin*

요 약

기존의 단대단 혼잡제어는 단순히 중복된 ACK 신호를 이용하여 혼잡을 처리하며 일반적으로 혼잡을 제어하는데 시간이 많이 소요된다. 이러한 메커니즘은 혼잡발생시에 TCP의 혼잡 윈도의의 크기가 동기화 되는 현상을 피할 수 없으며, 더욱이 RTT가 증가될 경우 중복된 ACK 패킷을 받기 전에 혼잡이 사라지거나 받기전까지 전송되는 패킷으로 인해서 정확한 혼잡제어를 할 수 없다.

최근에 이러한 단대단 혼잡제어의 문제점을 개선하는 다양한 AQM(Active Queue Management)를 소개되고 있으며, 기본적으로 RED의 혼잡 제어 메커니즘을 사용한다. 본 논문에서는 우선 다양한 전송률과 혼잡에 반응하는 흐름(TCP)과 반응하지 않는 흐름(UDP)을 갖는 트래픽 상에서 RED, CHOKe, FRED 그리고 SRED등과 같은 AQM의 효과를 평가한다. 특히, CHOKe 메커니즘의 경우 혼잡에 반응하지 않는 흐름이 증가 할 경우 공평성이 떨어지는 단점을 갖는다. 따라서, 본 논문에서는 UDP 트래픽으로부터 TCP 트래픽을 보호하기위해 UPD 트래픽과 TCP 트래픽을 두개의 독립된 논리적인 큐에서 서로 다른 CHOKe 메커니즘으로 처리하는 이중 큐 CHOKe 메커니즘을 제안한다. 각 흐름의 정보를 유지하기 위해 LRURED에서 제안한 부분상태정보(partial state information)를 이용하여 트래픽 유형별로 구별하고 격리하여 보다 효율적인 혼잡제어를 제시한다.

Key Words : RED, Congestion Contorl, AQM, CHOKe, LRURED

ABSTRACT

Current end-to-end congestion control depends only on the information of end points (using three duplicate ACK packets) and generally responds slowly to the network congestion. This mechanism can't avoid TCP global synchronization in which TCP congestion window size is fluctuated during congestion period. Furthermore, if RTT(Round Trip Time) is increased, three duplicate ACK packets are not correct congestion signals because congestion might already disappear and the host may send more packets until it receives three duplicate ACK packets. Recently there are increasing interests in solving end-to-end congestion control using AQM(Active Queue Management) to improve the performance of TCP protocols. AQM is a variation of RED-based congestion control. In this paper, we first evaluate the effectiveness of the current AQM schemes such as RED, CHOKe, ARED, FRED and SRED, over traffic with different rates and over traffic with mixed responsive and non-responsive flows, respectively. In particular, CHOKe mechanism shows greater unfairness, especially when more unresponsive flows exist in a shared link. We then propose a new AQM scheme using CHOKe mechanism, called DQC(Double Queue CHOKe), which uses two FIFO queues before applying CHOKe mechanism to adaptive congestion control. Simulation shows that it works well in protecting congestion-sensitive flows from congestion-causing flows and exhibits better performances than other AQM schemes. Also we use partial state information, proposed in LRURED, to improve our mechanism.

* 성균관 대학교 네트워크 연구실([gyunee, hjshin, drshin]@ece.skku.ac.kr)
논문번호 : 030368-0820, 접수일자 : 2003년 8월 20일

Ⅰ. 서 론

인터넷에서 제공되는 서비스의 다양화와 사용자 수의 급증은 라우터에서 처리하는 트래픽의 증가를 가져왔다. 따라서 라우터에서 증가하는 트래픽에 따른 효율적인 큐 관리 방식은 단순히 혼잡제어뿐만 아니라 서비스의 질적 향상에도 영향을 미칠 것이다. 인터넷에는 혼잡이 발생하였을 경우 전송률을 줄이는 트래픽과 그렇지 않는 트래픽으로 구분할 수 있다. TCP를 사용하는 흐름은 혼잡이 발생할 경우 전송률을 조절하지만 음성과 비디오와 같은 UDP를 기본으로 한 멀티미디어 트래픽은 그 특성상 혼잡이 발생하여도 그 전송률을 유지한다. 기존의 end-to-end 피드백 혼잡제어는 오직 단말노드에서만 혼잡을 감지하고 제어할 수 있다. 이 방식은 혼잡이 발생하여 전송자가 혼잡을 감지할 때까지 걸리는 제어시간에 영향을 받기 때문에 혼잡 발생 시점에서 처리를 할 수 없다. 따라서 멀티미디어 트래픽의 증가에 따른 라우터 상에서의 새로운 메커니즘이 요구된다.

최근 혼잡에 반응하지 않는 트래픽에 대해서 효과적으로 제어하는 여러 기법들이 소개되었으며, 이러한 기법들은 혼잡이 발생했을 경우 UDP 트래픽으로부터 TCP트래픽을 보호하는데 주 목적이 있다. FIFO 큐(queue)는 큐가 다 채워지면 그 이후의 패킷을 누락시키는 큐 관리 기법으로 혼잡이 가중될 경우 여러 흐름(flow)들이 동시에 전송률을 줄이고 늘리는 동기화 현상(Global synchronization)이 발생하게 된다. FIFO 큐의 단점은 공평성이나 흐름-격리(flow-isolation; 임의의 flow를 다른 flow로부터 고립 하여 보호하는 것)를 얻지 못한다. RED[11]는 큐가 다 채워지기 전에 패킷을 확률적으로 누락시키는 기법으로 시스템 내에 흐름별 상태정보를 이용하여 확장될 수 있다. AQM[3]은 RED를 기본으로 한 메커니즘으로서 시스템 내에 상태정보를 이용하거나 라우터간의 통신으로 공평성이나 흐름-격리를 보장해주고 더 나아가 효율적인 큐 관리로 혼잡이 발생할 때 신속하게 또는 미리 처리 할 수 있는 방법을 제공하여 준다. 즉, 혼잡이 발생할 경우 전송자로 하여금 자신의 전송 속도를 조절하거나 라우터에서 상태정보를 이용하여 전송속도를 조절하도록 한다. 이러한 방법에는 SRED[4], FRED[9], ARED[6], CHOKe[2] 등이 있으며 각 알고리즘은 기본적으로 RED 메커니즘을 따른다. 다중 문턱 값(Multi threshold)이나 혼잡 처리에 있어 제어 패킷을 사용하고 있으며 전반적인 구조는 거의 유사하다. SRED는 라우터에서 좀비 리스트(zombie list)를 두어 각 패킷의 누락 확률을 결정한다. FRED는 모든 흐름의 정보를 라우터가 유지하며 각 흐름이 할당된 자원보다 많이 사용할 경우 해당 패킷을 누락시킨다. CHOKe는 도착한 패킷과 큐 내의 패킷을 임의로 선택하고, 두 패킷이 같은 흐름일 경우 두 패킷 모두 누락 시킨다. ARED는 현재 트래픽 양에 따라서 문턱 값을 동적으로 조절하여 큐를 관리한다. CHOKe는 라우터에서의 상태정보가 필요 없고 라우터에서의 계산이 간단하다는 장점이 있지만 UDP 트래픽으로부터 TCP 트래픽을 완전히 보호하지 못한다. 부분상태정보를 이용하여 트래픽을 보호하는 LRURED[1]는 각 트래픽을 캐시내의 정보를 이용하여 각 흐름별로 격리하여 큐를 관리한다.

본 논문에서는 TCP 트래픽을 UDP 트래픽으로부터 보호하기 위해 LRURED에서 제안한 부분상태정보를 이용하여 트래픽을 클래스별로 구분하고 각 클래스는 독립된 가상 큐에서 서로 다른 CHOKe메커니즘으로 처리하여 혼잡을 처리하는 이중 큐 CHOKe 기법을 제시한다. 논문의 구성은 다음과 같다. 2장에서는 CHOKe 메커니즘을, 3장에서는 이중 큐 CHOKe구조와 누락메커니즘을 4장에서는 모의실험 결과를 보이겠다. 5장은 본 논문의 결론이다.

Ⅱ. CHOKe 알고리즘

CHOKe 알고리즘은 기존의 RED 방식 위에 새로운 패킷 누락 메커니즘을 추가 시킨 것이다. 패킷이 큐에 도착하면 큐에서 임의의 패킷을 선택해서 도착한 패킷과 선택된 패킷이 같은 흐름에 속했을 경우 두개의 패킷을 모두 누락 시킨다. 같지 않을 경우는 기존의 RED 방식으로 패킷을 처리한다. CHOKe는 트래픽 종류에 관계없이 같은 누락 메커니즘을 사용하기 때문에 UDP 트래픽(Unresponsive Traffic)의 양이 증가 할 경우 상대적으로 TCP 트래픽은 큐에 들어가 확률이 적어진다. 따라서 UDP 트래픽의 증가에 따른 TCP 트래픽이 감소가 발생한다. UDP 트래픽의 전송률이 다른 경우 그에 따른 처리가 요구 된다.

"CHOKe with N"알고리즘은 UDP 트래픽의 양이 많을 때 사용할 수 있다. N 값은 문턱 값(Threshold) 값에 따라 정해진다. CHOKe방식은 RED의 알고리즘의 전처리 단계이기 때문에 RED에

서 사용하는 문턱 값을 그대로 사용한다. N 값은 min_th 와 max_th 사이를 K개의 영역으로 나눈 후 각 영역에 따라서 N값은 2*(Ki+1)을 갖는다. N 값에 따라 큐에서 선택할 패킷의 수가 정해지기 때문에 전송속도가 상대적으로 높은 흐름의 경우 선택된 N개의 패킷 중에 포함될 확률이 높다. 따라서 전송률에 따른 패킷 누락이 가능해진다.

"CHOKe with N"알고리즘은 실제 각 흐름이 큐에서 점유하는 크기에 따라 N 값을 결정해야 하는 문제가 있다. 각 흐름이 점유하는 크기가 같은 전송률일 경우 다음과 같이 계산 할 수 있다.

Average Packet in Queue per Flow(APQF) = Queue Length / Flow Number (공식1)

각 흐름의 큐 내의 실제 점유량은 전송률에 따라서 APQF보다 크거나 작다. N 값의 결정에 있어 각 흐름의 점유량을 고려하지 않을 경우 많은 점유량을 갖는 흐름을 처리하는데 한계가 있다. 실제 점유량이 APQF보다 크고 N값이 그 값보다 작을 경우 패킷 누락 효과는 작기 때문이다. 따라서 N값의 선택에 있어 각 흐름의 실제 점유량과 APQF에 따른 처리가 필요하다. 본 논문에서는 각 흐름의 점유량을 저장하기 위해서 캐시를 이용한다.

Ⅲ. 이중 큐 CHOKe : DQC

본 장에서는 공격적인 UDP 흐름으로부터 TCP 흐름을 보호하기 위해 누락할 패킷의 선택과 누락 확률을 캐시내의 정보를 이용하여 처리 하는 "이중 큐 CHOKe(이하 DQC)"을 알아보겠다.

1 이중 큐 CHOKe 구조

"CHOKe with N"에서는 N을 임의로 선택하고 그 중에서 도착한 패킷과 동일한 흐름에 속한 패킷만 누락시키지만 DQC는 캐시내의 'cnt'를 이용하여 누락할 패킷을 선택한다. 'cnt'는 패킷이 큐에 도착했을 경우 큐에 같은 흐름이 있으면 그 값을 변경한다. 새로운 패킷이 도착할 경우 캐시내의 위치를 최상위로 변경하고 캐시 내에 저장 공간이 없을 경우 확률적으로 새로운 패킷을 받아 드린다. 캐시에 저장되는 정보는 흐름을 구별하기 위한 전송과 수신 노드의 주소 값이 저장 된다. 같은 흐름의 패킷이 큐 내에 도착하면 캐시 내의 'HIT CNT'는 증가하게 된다. 'HIT CNT'의 증가는 'Time TBL'의 값과 현재 패킷이 도착했을 때의 차이로 계산된다. 이

CHOKe Cache

ID	SRC ADDR	DST ADDR	HIT CNT	Time TBL
1	11	21	128	287
2	12	22	75	328
3	13	23	329	528
4	14	24	230	628
5	15	25	569	658
6	16	26	59	678

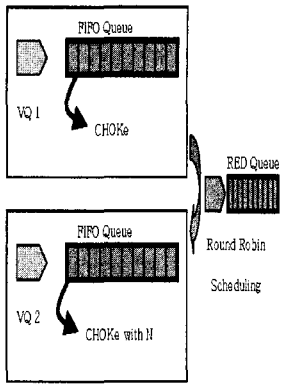


그림 1. DQC의 구조

와 같이 처리 하는 이유는 이전에 캐시에서 'HIT'된 시간과 다음 'HIT'된 시간과의 차이가 큰 경우 'HIT CNT'를 감소 시켜 전송 속도가 감소된 흐름을 보호하기 위해서다.

'HIT · CNT' = 'HIT CNT' + incr - (now-timestamp) / target_rate (공식 2)

(공식 1)에서 incr 값은 RED의 평균 큐 크기에 따라서 결정된다. 그림 2에서 min_th, mid_th, max_th를 이용하여 현재 평균 큐 길이가 min_th보다 크고 mit_th 보다 작을 경우 incr 값을 2를 사용한다. mid_th 보다 크고 max_th보다 작을 경우 3, max_th보다 클 경우는 4를 사용하여 HIT CNT를 갱신한다. tiemstamp 값은 캐시 내에 저장된다. target-rate은 라우터의 패킷당 처리 속도와 실제 큐에 존재하는 흐름으로 구할 수 있다.

target-rate = ((packetsize*8)/Bandwidth) / Active Flow number (공식 3)

(공식 3)은 packet 크기가 동일한 경우만 고려한 경우다. 크기가 다를 경우에는 흐름별로 가중치를 고려해야 할 것이다. 그림 1은 본 논문에서 제안한 이중 큐 CHOKe 메커니즘 구조다. 패킷이 도착하면 캐시내의 정보를 확인하고 'HIT CNT'값에 따라서 가상 큐에 저장되는 순서가 결정된다. 각각의 가상 큐는 RED 큐의 절반 크기로 설정된다.

2 이중 큐 CHOKe 누락 메커니즘

그림 2는 DQC의 내부 큐 구조다. 각 가상 큐는

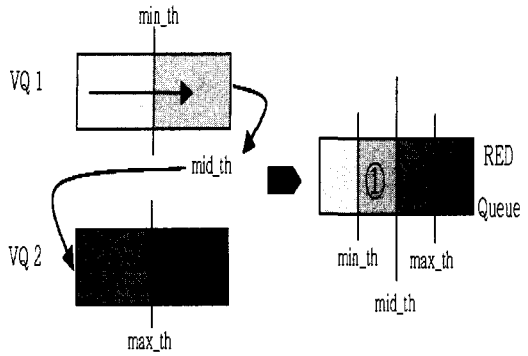


그림 2. DQC의 내부 큐 구조

RED 큐의 절반에 해당하는 역할을 하게 된다. RED 큐의 평균 큐 길이가 min_th와 max_th 사이에 있을 경우 DQC는 캐시의 정보를 이용하여 도착한 패킷과 큐상의 패킷을 비교한다. 평균 큐 길이가 min_th보다 크고 mid_th보다 작을 경우 VQ1에서만 비교할 대상 패킷을 찾는다. mid_th보다 크고 max_th보다 작을 경우에는 VQ1에서는 CHOKe 방식으로 처리하고 VQ2에서는 "CHOKe with N"을 사용한다. 평균 큐 길이가 max_th보다 클 경우 RED는 도착한 패킷만 누락 시키지만 DQC는 이 경우에도 VQ1과 VQ2에서 검색하여 "CHOKe with N" 방식으로 패킷을 처리한다.

VQ큐에 저장되는 순서는 캐시내의 'cnt'정보를 이용하여 결정한다. 만약 'cnt'값이 RED 큐의 'max_th'값보다 클 경우 VQ1에 우선 저장하고 VQ1에 공간이 없을 경우는 도착한 패킷은 누락 된다. 'cnt'값이 'max_th'값보다 작을 경우에는 VQ2에 우선 저장하고 VQ2의 공간이 없을 경우에는 VQ1에 저장한다. 이와 같이 전송률이 높은 트래픽은 큐의 한계 값이 실제 큐의 절반으로 조절되기 때문에 UDP 트래픽이 라우터의 모든 자원을 점유하는 가능성을 제거 할 수 있으며 TCP 트래픽과 전송률이 낮은 UDP 트래픽을 보호 할 수 있게 된다. 본 논문에서는 이와 같은 효과를 'self-blocking' 효과라 하겠다.

그림 3은 본 논문에서 제안한 이중 큐 CHOKe의 의사코드다. 패킷이 라우터에 도착할 경우 RED의 문턱 값을 이용하여 평균 큐 길이가 RED의 'min_th' 값을 넘게 되면 choke 함수를 이용하여 패킷을 관리한다. 이 함수에서 각 패킷은 캐시의 정보와 평균 큐 길이에 따라 처리된다. 그림 2의 구간

```
ving packet pkt
  : Virtual Queue for CBR Traffic
: : Virtual Queue for TCP Traffic
g : avgerage Pq's queue length
1 : RED queue limit
_th : Minimum threshold
x_th : Maximum threshold
l_th : Middle threshold
 : 'hit cnt' of cache

qavg >= th_min)
hoke(pkt)
nque(pkt)

que:
ue(pkt)

if(cnt < max_th)
{
  if(vq2->length()<qlim/2)
        vq2->enque(pkt)
  else  vq1->enque(pkt)
}else {
  if(vq1->length()<qlim/2)
        vq1->enque(pkt)
  else  drop(pkt)
}
```

```
  else {
      Select a packet (spkt1) from vq1
      Select a packet (spkt2) from vq21
      Set dropflag 0
      if(spkt1 == pkt)
      {
          drop spkt1
          set dropflag 1
      }
      if(spkt2 == pkt) drop spkt2
      {
         drop spkt2
         set dropflag 1
      }
      if(dropflag) drop pkt
      else enque(pkt)
    }
  }
Deque:
deque(pkt)
{
    if(deq_turn)
    {
        p = vq1->deque()
        if(p==0)
        {
            p = vq2->deque()
            deq_turn = 1
        } else deq_turn = 0
    } else {
        p = vq2->deque()
        if(p==0)
        {
            p = vq1->deque()
            deq_turn = 0
        } else deq_turn = 1
    }
}
```

그림 3. DQC의 Pseudo Code

①에 있을 경우 도착한 패킷과 비교할 패킷을 VQ1에서 선택하는 것을 제외하고 CHOKe와 동일하다. 구간 ②에 있을 경우는 VQ1과 VQ2에서 선택하여 VQ1에서 선택한 패킷부터 순서대로 비교한다. 구간 ③은 구간 ②와 동일하게 패킷을 선택하지만 선택된 패킷을 각각 비교하여 처리한다. enque 함수는 'self-blocking'효과를 얻기 위해 각 가상 큐의 길이와 캐시의 정보를 이용하여 패킷을 처리하게 된다. deque함수는 각 가상 큐에서 서비스되는 순서를 결정한다. 각 가상 큐에 가중치를 줄 경우 보다 효율적인 큐 관리가 이루어 질것이다. 본 논문에서는 라운드 로빈 방식을 이용하여 처리 하였다.

Ⅳ. 모의 실험

본 논문에서 제안한 Double Queue CHOKe 방식의 성능을 비교하기 위해서 NS2[11] 사용하였다. NS2에서 제공하는 FIFO, RED와 LRURED, CHOKe와 본 논문에서 제안한 DQC를 NS2에 추가하여 실험하였다.

1 모의실험 환경

그림 4는 모의실험에 사용한 실험 환경이다. 각 실험은 시나리오에 따라 3개로 나뉘어 진다. 시나리오 1은 TCP 트래픽을 먼저 발생 후 특정 시간대에만 UDP 트래픽을 발생시킨다. 시나리오 2는 TCP와 UDP 트래픽을 동시에 발생시킨다. 시나리오 3

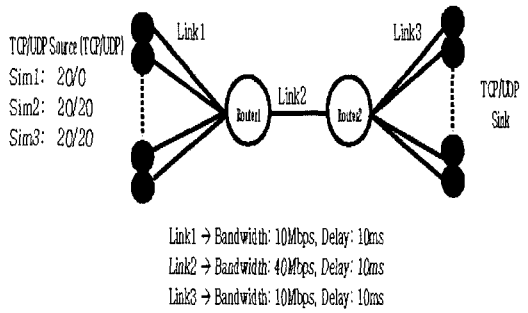


그림 4. 모의실험 환경

은 TCP와 UDP 트래픽을 모두 발생하지만 UDP 트래픽에 전송 속도가 상대적으로 큰 흐름을 추가로 발생시킨다.

각 실험에서 사용한 환경 값은 다음과 같다. Router1에서 Router2의 bandwidth는 10Mbps이고 이외의 링크는 40Mbps를 사용하였다. 따라서 Router1은 UDP 트래픽을 전송률에 따라서 혼잡이 발생하는 병목 지점이 되며 각 큐 알고리즘에 따라서 다른 큐를 사용한다.

2 시나리오 1 실험

시나리오 1에서는 TCP 트래픽이 UDP 트래픽이 특정 시간대에 발생 했을 경우 어느 정도 자원을 점유 당하는지 알아 보기위한 실험이다. 실험에서 TCP 트래픽은 일정한 시간간격으로 트래픽을 발생시킨다. UDP 트래픽은 실험 시작 후 15초에서 25초 구간에서 발생된다. 이 구간에서 TCP 트래픽의 감소량을 관찰 하였다.

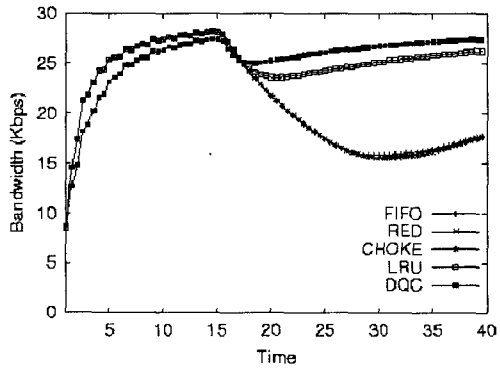


그림 5. 시나리오1: TCP 트래픽의 시간별 변화

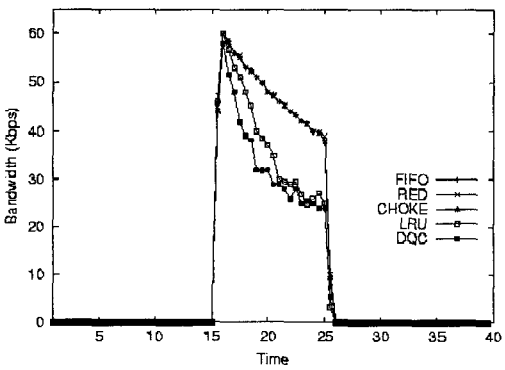


그림 6. 시나리오1: UDP 트래픽의 시간별 변화

그림 5에서 15초에서 25초 구간에서는 TCP 트래픽이 감소함을 알 수 있다. LRURED와 DQC의 경우 UDP 트래픽이 발생하는 시점에서 트래픽이 줄어들다 UDP 트래픽이 전송을 중지하기 전에 그 이전 전송 속도에 대등한 속도까지 상승하는 것을 볼 수 있다. 그 이유는 캐시에 UDP 트래픽의 정보가 저

장되어 RED 큐의 상태가 변하면서 UDP 트래픽이 라우터의 자원을 점유하지 못하도록 하기 때문이다. FIFO, RED, CHOKe의 경우 TCP 트래픽의 누락율이 높기 때문에 이전 속도까지 상승하는데 시간이 오래 걸리게 된다. 그림 5에서 UDP 트래픽이 중지하는 25초 이후에도 UDP 트래픽 발생으로 인한 다수의 TCP 패킷 누락 때문에 상승 속도가 낮음을 알 수 있다. LRURED와 DQC의 경우 DQC는 25초에 이르러서는 이전 속도와 거의 대등한 속도까지 올라감을 볼 수 있다. LRURED는 실험이 끝날 때까지 DQC가 25초 때에 가진 속도까지 상승 하지 못한다. UDP 트래픽은 CBR로 3.1Mbps의 같은 전송 속도로 전송하고 TCP는 FTP를 이용하여 전송한다. 그림 6은 UDP 트래픽의 시간대별 전송 속도를 관찬한 결과이다. 15초에서 발생해서 25초에 전송이 끝난다. FIFO, RED, CHOKe는 완만한 경사의 감소를 보이는 반면 LRURED와 DQC는 가파른 곡선을 보여준다. 캐시내의 'hit cnt'값이 시간에 따라 증가하기 때문에 그 감소 속도가 시간에 따라 빨라짐을 알 수 있다.

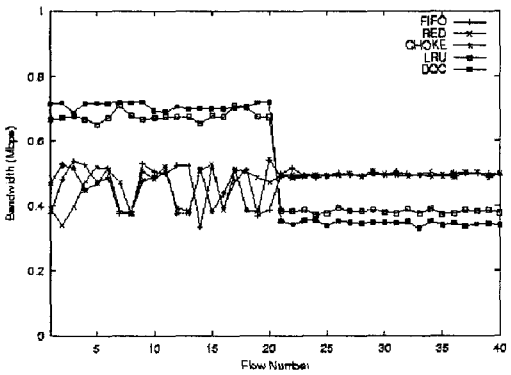


그림 7. 시나리오1: 흐름별 전송속도

그림 7은 TCP(1~20)와 UDP(21~40)의 흐름별 전송속도를 관찰한 결과이다. TCP 트래픽은 FIFO, REDO, CHOKe의 경우 각 흐름별로 전송속도를 조절 하지 못하기 때문에 각 흐름별로 전송 속도는 진동함을 알 수 있다. LRURED와 DQC의 경우 DQC는 UDP 트래픽을 RED 큐의 상태와 캐시의 'hit cnt' 값에 따라 두개의 큐 중에 하나에 저장하고 그 큐에 저장할 공간이 없을 경우 누락 시키게 된다. 따라서 LRURED보다 UDP 트래픽을 보다 더 격리 시킬 수 있음을 알 수 있다. TCP 트래픽은 이와 같은 이유로 LRURED보다 전송속도가 각 흐름별로 높게 나타난다.

3 시나리오 2 실험

시나리오 1에서는 UDP 트래픽을 특정 시간대에만 발생시켜 흐름별 전송속도와 시간대별 전송속도를 관찰하였다. 시나리오 2에서는 TCP와 UDP 트래픽을 동시에 발생시켜 전송속도를 관찰한다.

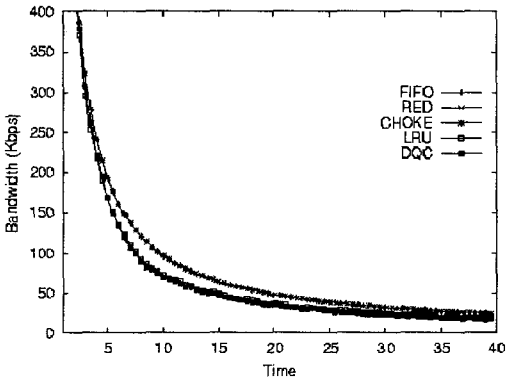


그림 8. 시나리오2: UDP 트래픽의 시간별 변화

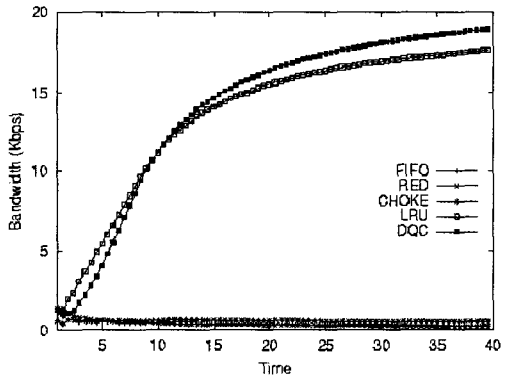


그림 9. 시나리오2: TCP 트래픽의 시간별 변화

그림 8은 TCP 트래픽의 시간대별 전송 속도를 나타낸다. FIFO, RED, CHOKe 는 UDP 트래픽으로 인해 라우터의 자원을 거의 사용하지 못하는 것을 알 수 있다. 그 이유는 UDP 트래픽의 전송 속도가 상대적으로 빨라 라우터의 큐에 저장될 확률이 작을 뿐만 아니라 다수의 패킷 누락으로 인해서 그 전송 속도가 상승 하지 못하기 때문이다. 그림 9에서 UDP 트래픽은 시간에 따라서 그 전송 속도가 감소한다. 그 이유는 라우터의 자원을 시간이 지남에 따라 각 흐름이 점유하기 때문이다. 각 UDP 트래픽과 TCP 트래픽은 일정시간 간격으로 발생 한

다. UDP 전송속도는 시나리오 1과 동일하게 실험 하였다.

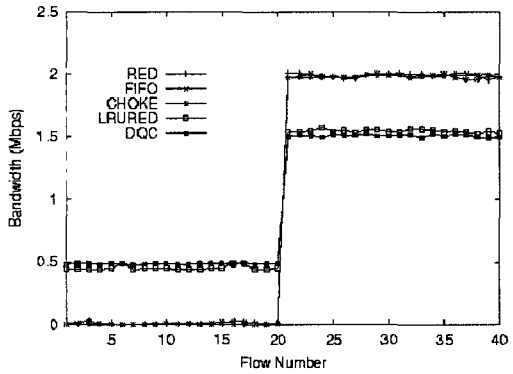


그림 10. 시나리오2: 흐름별 전송속도

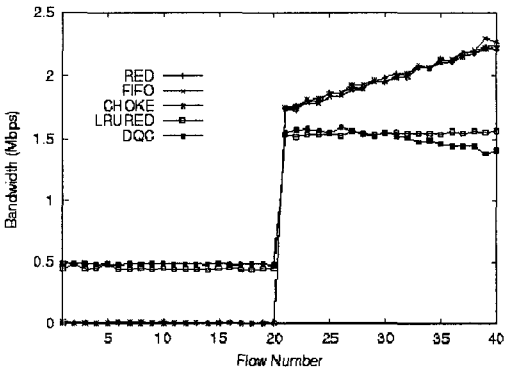


그림 11. 시나리오2: 전송률이 다른 UDP 트래픽 전송 속도

그림 10은 UDP 트래픽의 전송속도를 동일한 경우의 결과이고, 그림 11은 전송률을 일정한 간격으로 증가 시켰을 경우의 결과이다. 전송률은 3.0Mbps에서 4.0Mbps까지 2개의 흐름씩 0.1Mbps 단위로 증가하였다. 그림 10의 경우 LRURED나 DQC가 거의 같은 결과를 나타내지만 그림 11처럼 UDP 트래픽의 전송률이 각 흐름 별로 틀린 경우 DQC는 공격적인 흐름을 격리 시킬 수 있음을 보여준다. 전송률이 보다 큰 흐름의 대한 LRURED의 처리는 전송률이 낮은 흐름과 동일하게 처리 되는데 LRURED는 캐시에서 'hit cnt'의 값을 고정된 값을 이용하여 찾기 때문이다. DQC는 공격적인 흐름을 격리 하는데 보다 효과가 있음을 관찰 할 수 있다. 흐름 30번 이후의 UDP 트래픽은 그 전송률이 감소함을 알 수 있다. 본 논문에서는 공격적인 흐름의 격리를 보다 효과 적으로 하기 위해서 (공식 4)를 이용하여 가상 큐 선택에 사용하였다. 그림 3의 'enque'함수 또한 (공식 4)를 이용한 threshold 값으로 수정 할 수 있다.

threshold = qlim / 'flow number' + th_mid (공식4)

'active flow number' 는 캐시 내의 저장된 흐름의 수가 된다. qlim / 'flow number' 는 (공식 1)의 결과다. 각 흐름별로 큐에 들어 갈 수 있는 APQF와 th_mid 값을 이용하여 'hit cnt'가 이 값을 넘을 경우 VQ1에 저장하고 그렇지 않을 경우 VQ2에 저장 한다.

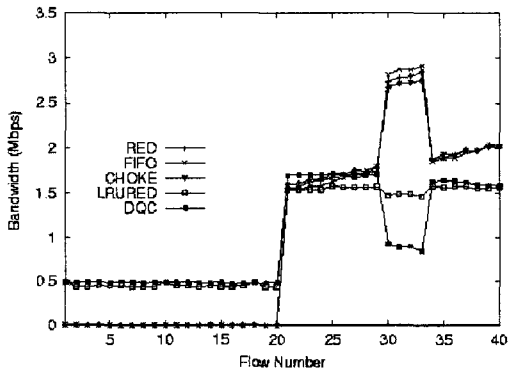


그림 12. 시나리오3: 전송률이 큰 UDP 트래픽의 격리

그림 12는 특정 UDP 트래픽의 전송 속도를 증가해서 실험한 결과이다. TCP 트래픽은 특정 UDP 트래픽의 증가에 영향을 받지 않는다. LRERED 또한 전송률이 높은 특정 흐름의 전송률이 감소되는 것을 알 수 있다. 하지만 하나의 큐에서 처리하기 때문에 DQC에서 보여주는 UDP 트래픽의 감소를 얻을 수 없다. DQC는 두개의 큐를 이용하여 캐시의 'hit cnt'의 값에 따라서 트래픽을 분류할 수 있기 때문에 TCP 트래픽은 UDP 트래픽에 영향을 받지 않을 뿐만 아니라 전송률이 낮은 UDP 트래픽을 보호 할 수 있다. 그림 12에서 흐름 번호 31에서 34번의 UDP 트래픽의 감소로 다른 UDP 트래픽의 전송률이 높아짐을 알 수 있다.

V. 결 론

혼잡이 발생할 경우 UDP 트래픽으로 인해 TCP 트래픽이 잠식당하는 현상을 막기 위해 여러 메커

니즘이 소개되었다. CHOKe 메커니즘은 라우터에서 상태정보가 필요하지 않으며 복잡한 처리가 필요하지 않다. 그러나 CHOKe 메커니즘은 UDP 트래픽이 많고 전송률이 서로 다를 경우 공평성에 있어 그 한계를 보인다. 본 논문에서 제안한 "Double Queue CHOKe" 방식은 UDP 트래픽이 많고 그 전송률이 다를 때 TCP트래픽과 UDP 트래픽을 두개의 가상 큐와 부분적인 상태 정보를 이용하여 보호한다. 부분적인 상태정보를 이용하여 전송률에 따라서 큐를 관리하기 때문에 TCP 트래픽이 UDP 트래픽으로부터 잠식당하는 현상을 방지할 뿐만 아니라 전송률이 낮은 UDP 트래픽 또한 공격적인 UDP 트래픽으로부터 보호된다. 각 흐름은 전송속도에 따라서 큐에 저장될 수 있는 packet의 수가 제한되는 self-blocking 메커니즘을 사용했기 때문에 LRURED에서 보여준 흐름 격리를 보다 세분화 시켰다. 흐름 격리를 위해 사용한 LRURED는 라우터에 도착한 모든 packet에 O(1)의 패키 처리 비용을 가지며, cache에 저장되는 확률 값을 정할 경우 보다 효율적인 처리가 가능하다. 라우터에서 추가적으로 필요한 저장 공간은 LRU cache의 크기에 비례한다.[1] CHOKe에서는 추가적인 메모리는 필요치 않으며 LRU cache의 정보에 따라서 CHOKe 메커니즘이 동작하므로 LRURED의 패킷 처리 비용에 포함 된다고 볼 수 있다. 본 논문의 결과를 토대로 트래픽 성격에 따라서 class별로 처리 할 수 있는 흐름 격리 메커니즘의 연구가 필요하다.

참 고 문 헌

[1] Smitha, A. L. Narasimha Reddy, "LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers", *in Proc. of Globecomm,* Nov. 2001.

[2] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe- A stateless active queue management scheme for approximation fair bandwidth allocation", *IEEE Infocom '2000,* March 2000.

[3] M. Parris, K. Jeffay, F.D. Smith, "Responsive vs. Unresponsive Traffic: Active Queue Management for a Better-Than-Best-Effort Service", *Technical Repor*t, September 2000. *http://www.cs.unc.edu/~jeffay/papers/IEEE-Networks-01.pdf*

[4] TJ Ott, TV Lakshman, and LH Wong, "SRED: stabilized RED", *Proc. IEEE INFOCOM Conf.*, pp. 1346~1355, Mar. 1999. [5] W. Feng, D. Kandlur, D. Saha, K. Shin, "Blue: A New Class of Active Queue Management Algorithms", *U. Michigan CSE-TR-387-99*, April 1999

[6] W. C. Feng, D. Kandlur, D. Saha, and K. Shin., "A Self-configuring RED Gateway", *Infocom'99*, 1999

[7] Stoica, I., Shenker, S. and Zhang, H., "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High speed Networks", *Proceedings of ACM SIGCOMM* '98.

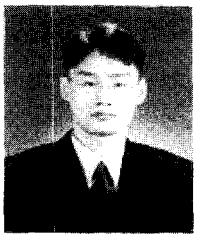
[8] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, January 1997.

[9] Lin, D. and Morris, R, "Dynamics of Random early Detection" *Proceedings of SIGCOMM* '97

[10] S. McCanne, S. Floyd, K. Fall, UCB/LBL Network Simulator NS, 1996.

[11] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", *IEEE/ACM Trans. and Networking*, vol.1, no. 4, Aug. 1993.

최 기 현(Kee-Hyun Choi) 준회원



2000년 2월 : 성균관대학교 전기 전자및 컴퓨터 공학부 졸업
2002년 2월 : 성균관대학교 전기전자 및 컴퓨터 공학과 석사
2002년~현재 성균관대학교 정보통신공학부 박사과정

〈주관심분야〉 Active Network, AQM, 혼잡제어

신 호 진(Ho-Jin Shin) 준회원

1994년 2월 :성균관대학교 전기공학과 졸업
1999년 2월 : 성균관대학교 대학원 석사
2001년 2월 : 성균관대학교 대학원 전기전자 및 컴퓨터 공학과 박사과정 수료
2001년~현재 성균관대학교 대학원 정보통신 공학과 박사 학위 과정
1994년~1995년 삼성중공업㈜

〈주관심분야〉 ATM, Industrial Networks, Discrete Event Systems 등

신 동 렬(Dong-Ryeol Shin) 정회원

1980년 성균관대학교 졸업
1982년 한국과학기술원 석사
1992년 Georgia Institute of Tech. (Ph.D)
1994년~현재 성균관대학교 정보통신공학부 교수
1982년~1986년 Assistant researcher, Daewoo Heavy Ind., Limited
1992년~1994년 Senior researcher, Samsung Data Systems

〈주관심분야 〉 Industrial Networks including MMS,ICCP, and Fields, High-Speed telecommunication networks with emphasis on ATM modeling and switching. Performance evaluation with markovian and queuing networks 등