인터넷에서 혼잡제어를 위한 개선된 RED 알고리즘

정규정°, 이동호
광운대학교 컴퓨터 과학과
{kjjung, dhlee}@cs.gwu.ac.kr

An Effective RED Algorithm for Congestion Control in Internet

Kyu-Jung Jung°, Dong-Ho Lee
Dept. of Computer Science, Kwangwoon University

요 약

기존의 네트워크에서는 혼잡상황이 감지된 이후에 네트워크 성능이 급격하게 저하된다. 이러한 문제를 해결하고자 RED(Random Early Detection)기법이 소개되어 게이트웨이에서 혼잡상황에 대하여 능동적으로 대처할 수 있는 알고리즘이 제시되었다. 하지만, RED는 매개변수 설정이라는 문제가 남아있다. 그리하여, 잘못된 변수값 설정으로 인한 네트워크 성능 저하가 현저하게 발생한다. 본 논문에서는 기존의 RED를 개선한 Effective RED를 제안한다. Effective RED는 RED 알고리즘의 문제점을 개선하여 네트워크의 상황에 맞추어 동적으로 매개 변수 값을 조정하는 알고리즘이다. 그리고, ns를 이용하여 Effective RED의 성능을 검증하였다.

1. 서 론

현재 인터넷은 Best-Effort 방식의 패킷 스위칭 네트워크에서는 가능한 한 패킷을 전송하려고 하기 때문에 갑자기 폭주하는 트래픽은 일정한 전송품질을 보장받지 못하게 되고 혼잡문제가 발생한다. 이러한 혼잡을 제어하기 위해서는 네트워크의 혼잡상태를 송신측에 알려주는 피드백이 필요하다. 송신측은 이러한 혼잡 신호를 받고 네트워크로 내보내는 패킷을 줄여야 하는데 이러한 전송율 조정이 혼잡제어의 기본개념이다.[1] 인터넷에서의 혼잡제어 분야에서 많은 연구가 이루어졌다. 네트워크의 규모가 점점 커지고 다양한 시스템이 연결됨에 따라 현재 인터넷은 끊임없는 변화에 직면하고 있으며 혼잡제어가 없는 UDP를 사용하면서 멀티캐스트 방식으로 실시간으로 전송되는 멀티미디어 트래픽이 날로 증가하고 있다. 현재 멀티캐스트 트래픽은 대부분 실시간 비디오/오디오 스트리밍 서비스와 같이 높은 대역폭을 필요로 하며 장시간 지속되며 흐름제어가 없는 특징 때문에 멀티캐스트 트래픽의 증가는 심각한 혼잡문제를 야기하게 된다. 현재 인터넷에서 TCP는 Slow Start, Congestion Avoidance, Fast Retransmit, Fast Recovery 방식의 혼잡제어 기법을 가지고 있다. [2,3] 이러한 방법은 현재 네트워크 상의 상태를 파악하여 최대의 처리율, 적은 지연과 높은 속도를 지원하고자 고안된 것이다.

이러한 방법 또한 기존의 방법을 개선시켰지만 혼잡상황이 감지된 이후에는 전역 동기화 및 패킷 손실로 인하여 네트워크 성능이 급격히 떨어지게 된다. 이런 문제점을 해결하고자 큐 관리 알고리즘으로 IETF에서는 RED (Random Early Detection) 알고리즘을 권고하고 있다.[4,5,6] RED 알고리즘은 혼잡 상황으로 발생되었던 문제점인 전역 동기화 및 패킷 손실율을 낮추는데 성공적이었다.[7]
그러나, RED 알고리즘의 경우 네트워크 상황에 따라 적절한 매개 변수 값을 설정하지 않는다면 Drop Tail 방법보다 효율이 저하되고 또한 매개 변수 값을 설정하는 데 있어서도 큰 어려움이 있다.

본 논문에서는 RED 알고리즘의 문제점으로 제기되고 있는 매개 변수 값을 현재 네트워크 상황에 맞추어 동적으로 변경하는 부분을 추가한 Effective RED 알고리즘을 제안하였다. 제안한 알고리즘은 기존의 RED 알고리즘보다 보다 효율적으로 큐를 관리하여 보다 좋은 성능을 제공한다.

2장에서는 현재 RED 알고리즘의 동작과 각 매개 변수 값에 대하여 설명하고, 문제점에 대한 설명을 하였다. 3장에서는 새롭게 제안한 Effective RED 알고리즘에 대하여 기술하였다. 4장에서는 시뮬레이터를 이용하여 제안한 알고리즘에 대한 검증을 하였다. 마지막으로 5장에서는 결론 및 향후과제를 제시하였다.

2. 관련 연구

2.1 RED 알고리즘

RED 알고리즘은 버퍼에서의 평균 큐 크기(q_{ave})를 이용하여 네트워크의 혼잡상황을 미리 감지하고 제어하는 기능을 가지고 있다.[7] RED는 혼잡상황을 선송 단말들에게 알려주는 방법으로서, 패킷의 ECN(Explicit Congestion Notification) 비트를 마킹하거나 혹은 폐기시키는 방법을 사용하고 있다. TCP를 이용하는 종단 단말에서는 패킷의 마크 필드 혹은 폐기를 확인하고, 혼잡 윈도우 값을 줄인후 슬로우 스타트를 실행한다.[2,6]

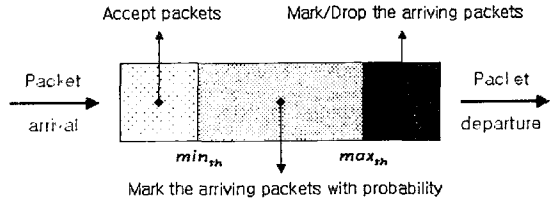

그림 1. RED 큐의 상태와 동작 알고리즘

일반적으로, RED 알고리즘은 혼잡상황을 미리 감지하고 대처하기 때문에 네트워크 자원의 낭비를 줄여서 Drop Tail 알고리즘보다 우수한 성능을 나타낸다.

그림 1과 같이 패킷이 도착하게 되면, 평균 큐 크기 (q_{ave})<min_{th} 이면 패킷을 큐에 넣고, 평균 큐 크기 (q_{ave})>max_{th} 이면 패킷을 모두 폐기하고, min_{th}<평균 큐 크기(q_{ave})<max_{th} 이면 랜덤하게 패킷을 마킹하거나 폐기하게 된다. 이 경우, 패킷이 폐기될 확률이 max_p 값에 따라 변하게 되므로, 이 매개 변수 값의 설정이 RED 알고리즘에 큰 영향을 주게 된다. max_p 값이 큰 경우에는 현재 유입된 패킷들이 마킹 혹은 폐기될 확률이 높아지게 되고, 반대로 max_p 값이 작은 경우에는 현재 유입된 패킷들이 마킹 혹은 폐기될 확률이 낮아지게 된다.

RED 알고리즘은 표 1과 같이 다양한 매개 변수에 의해 제어되며, 그 중에서도 가장 큰 영향을 주는 것은 max_p 이고, 이 값에 따라 전체적인 특성이 결정된다고 볼 수 있다. 그러나, 이렇게 다양한 매개 변수에 의해 동작되므로, 매개 변수 값이 잘못 설정된 경우에는 전체적인 성능이 Drop Tail 보다 떨어지게 된다.[8]

w_q 값은 평균 큐 크기(q_{ave})를 계산하는데 사용되는 가중치 값으로 이전의 평균 큐 크기가 현재의 평균 큐 크기에 어느 정도의 영향을 주는가를 결정하는 상수 값이다.

표 1. RED 매개 변수

$qlen$	최대로 수용 가능한 큐의 크기
min_{th}	폐기 여부를 결정하는 최소 임계값
max_{th}	폐기 여부를 결정하는 최대 임계값
w_q	q_{ave}(평균 큐 크기)를 계산하는 가중치 값
max_p	폐기 여부를 결정하는 최대 확률 값

$$q_{ave} = (1-w_q)q_{ave} + w_q q$$

위의 식과 같이, w_q 값에 따라 현재 큐 크기의 비중을 높일 것인가 혹은 과거 큐 크기의 비중을 높일 것인지가 결정된다. w_q 값을 결정하는 것도 중요한 것으로서, 이 값이 작은 경우에는 일시적인 큐 크기의 증가에 효율적으로 동작하지만, 게이트웨이에서 입력 패킷 모두를 폐기할 경우에는, 실제 큐 크기는 줄었으나 평균 큐 크기 (q_{ave})가 감소되는 비율이 너무 느려 계속적으로 모든 패킷을 폐기하는 경우가 발생한다. 반대로, w_q 값이 큰 경우에는 평균 큐 크기(q_{ave})도 빠르게 변화하지만, 일시적으로 폭주하는 패킷을 모두 수용하지 못하게 된다.

min_{th}, max_{th} 값의 선택은 네트워크 특성에 따라 결정된다. min_{th} 값이 너무 작게 되면, 랜덤하게 패킷을 마크 또는 폐기하는 구간이 크게 되어, 전체적인 네트워크 활용도가 떨어지게 된다. 반대로, min_{th} 값이 너무 크게 되면, 빈번하게 혼잡 상황이 발생하게 된다. max_{th} 값이 너무 작은 경우에도 큐의 사용가능한 크기가 작아져 네트워크 활용도가 떨어진다. 반대로, max_{th} 값이 너무 크면, Drop Tail 과 비슷한 동작을 하게 된다. 지금까지 서술한 바와 같이 RED 알고리즘에서는 매개 변수 값의 결정이 아주 중요하고, 그 값에 따라 RED 알고리즘의 성능이 크게 좌우되는 것을 알 수 있다. 표 2에서는 RED 매개 변수들의 값을 설정하는 데 있어 각 매개 변수 값의 권고치가 나타나있다.[9]

표 2. RED 매개 변수 값의 권고치

min_{th}	5 패킷
max_{th}	min_{th} 값의 3배
w_q	0.002
max_p	0.02 - 0.1

본 논문에서는 RED 알고리즘의 매개 변수 값중 가장 큰 역할을 하고 있는 max_p 값의 설정을 현 네트워크 상황에 맞추어 설정하는 Effective 알고리즘을 제안한다.

3. Effective RED 알고리즘

기존의 RED 알고리즘의 문제점을 해결하고자 제안된 Effective RED 알고리즘은 유입되는 패킷의 양에 따라 변화하는 큐의 크기를 줄여 평형상태에 도달할 수 있도록 max_p 값을 조정하는 알고리즘이다. 제안한 알고리즘은 그림 2와 같다.

```
if ((현재 큐 크기 > 이전 큐 크기) &&
    (현재 큐 크기 > 평균 큐 크기))
   maxp 값 = maxp 값 + d1

if ((현재 큐 크기 > 이전 큐 크기) &&
    (현재 큐 크기 < 평균 큐 크기))
   maxp 값 = maxp 값 + d2

if ((현재 큐 크기 < 이전 큐 크기) &&
    (현재 큐 크기 > 평균 큐 크기))
   maxp 값 = maxp 값 - d3

if ((현재 큐 크기 < 이전 큐 크기) &&
    (현재 큐 크기 < 평균 큐 크기))
   maxp 값 = maxp 값 - d4
```

그림 2. Effective RED 알고리즘

본 논문에서 제안하는 Effective RED 알고리즘의 동작은 큐에 패킷이 유입하게 되면, 현재 큐 크기와 이전 큐 크기를 비교하여 큐 크기의 진동차를 줄여 평형상태에 가까워지도록 max_p 값을 조정하는 알고리즘이다. 큐 크기의 진동차가 크다는 것은 네트워크의 상황에 맞지 않는 max_p 값이 현재 설정되어 있다고 볼 수 있다. Effective RED 알고리즘은 이러한 큐의 진동을 줄여 평형상태가 되도록, 즉 패킷 폐기율을 줄여주게 된다.

4. 실험 및 성능 평가

제안한 알고리즘의 성능을 분석하기 위하여 큐의 변동치가 얼마나 개선되었는지 살펴보았다. 네트워크 환경은 총 16개의 송신측이 전송하는 혼잡링크에서 큐의 변화를 모니터링 하였다. 그림 3과 같이 기존의 RED 방법보다 큐 크기의 변동이 줄어들었다.

Effective RED 알고리즘은 현재 네트워크 상황에 맞추어 max_p 값을 동적으로 설정해주어 기존의 RED보다 우수한 성능을 나타내었다.

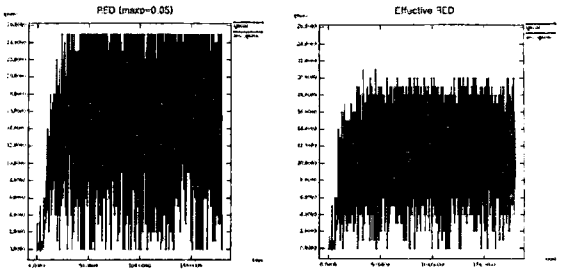

그림 3. RED와 Effective RED 큐 크기의 변화

5. 결론 및 향후 과제

본 논문에서는 기존의 RED 알고리즘에서는 현재 네트워크 상황이 고려되지 않은 고정된 매개 변수 값으로 동작하여 발생하는 문제점을 개선한 Effective RED 알고리즘을 제안하였다. Effective RED 알고리즘은 네트워크 상황에 맞추어 max_p 값을 동적으로 변화시켜 주어, 그 문제점을 해결하였고, 그 성능은 큐 크기의 변화에 대한 실험으로 검증되었다.

향후 과제로는 보다 평형상태에 가까운 형태가 되도록 개선하는 방법에 대한 연구가 수행되어야 하며, 그 연구를 토대로 실제 네트워크의 적용에 대한 연구가 수행되어야 할 것이다.

참고문헌

[1] C. Lefelhocz,, "Congestion Cortol for Best-Effort Service : Why We Need a New Paradigm", IEEE Network, January 1996.

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

[3] M. Allman,, V. Paxson,, W. Stevens,, "TCP Congestion Control', RFC 2581, April 1999.

[4] Braden, B., "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[5] Jacobson, V., "Congestion Avoidance and Control", Proceeding of SIGCOMM88, August 1988.

[6] Floyd, S., Fall, K., "Router Mechanisms to Support End-to-End Congestion Control", LBL Technical report, February 1997.

[7] Floyd, S., Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transaction on Networking, August 1993.

[8] Christiansen, M., Jeffay, K., Ott, D., and Smith F.D. "Tuning RED for Web Traffic", IEEE/ACM Transactions, June 2001.

[9] Floyd, S., "Ns Simulator Tests for Random Early Detection(RED) Gateways", Technical report, October 1996.