인터넷 라우터의 혼잡제어를 위한 새로운 큐 관리 알고리즘

구자헌 °, 송병훈, 정광수
광운대학교 전자공학부 컴퓨터통신연구실
jhkoo, byungh@adams.kwangwoon.ac.kr, kchung@daisy.kwangwoon.ac.kr

A New Queue Management Algorithm for Congestion Control in Internet Routers

Jahon Koo °, Byunghun Song, Kwangsue Chung
School of Electronics Engineering, Kwangwoon Univ.

요 약

기존의 인터넷 라우터는 Drop tail 방식으로 패킷을 관리한다. 따라서 네트워크 트래픽의 지수적인 증가로 인한 혼잡 상황으로 발생하는 패킷 손실을 해결 할 수 없다. 이 문제를 해결하기 위해 IETF (Internet Engineering Task Force)에서는 RED (Random Early Detection)와 같은 능동적인 큐 관리 알고리즘을 제시하였다. 하지만 RED는 동적으로 변화하는 인터넷 트래픽에 대하여 단지 큐 크기의 변화 정보를 얻어 혼잡 상황을 제어하기 때문에 성능에 있어는 매우 비효율적이다. 본 논문에서는 기존의 RED를 개선한 MRED를 제안했다. MRED는 RED에 비하여 휴리스틱 한 방법을 이용하여 폐기 확률 값을 계산하고, 이를 실험을 통하여 MRED의 성능을 검증하였다.

1. 서론

현재 인터넷은 Drop tail방식의 네트워크를 구성하여 최선형 서비스를 하고 있다. 이러한 방법은 지수적으로 증가하는 트래픽으로 인한 패킷 손실률의 증가 문제를 명시적으로 해결 할 수 없다. 이런 문제를 해결하기 위해서는 트래픽이 집중되는 라우터나 게이트웨이가 향상된 큐 관리 알고리즘을 지원해야 한다.

이러한 혼잡 제어를 하기 위해서 IETF의 RFC 2309에서는 크게 "스케줄링 알고리즘(scheduling algorithms)"과 "큐 관리 알고리즘(queue management algorithms)"으로 라우터 알고리즘을 분류하고 있다. 일반적으로 라우터 알고리즘은 모든 flow에 대하여 공정성을 제공하여야 하며 모든 네트워크 환경에 대하여 구현에 있어서 용이하여야 한다. 하지만 두 조건을 동시에 만족하기 매우 어렵다. 왜냐하면 스케줄링 알고리즘의 경우 각 flow간의 공정성을 제공하는 이점이 있지만 각 flow 단위로 큐를 관리해야 하는 복잡성 및 구현에 있어 어려움이 있다. 그리고 큐 관리 알고리즘의 경우 flow간의 공정성을 완전히 제공하지는 못 하지만 큐 관리에 있어 하나의 FIFO 큐를 관리하며 구현에 있어 용이하다. 즉, 두 가지 타입의 알고리즘이 추구하는 목적이 다소 다르기 때문이다.

본 논문에서는 구현이 간단하면서도 공정성 및 성능을 개선할 수 있는 새로운 큐 관리 알고리즘을 제안하였다. 제안한 알고리즘은 기존의 큐 관리 알고리즘 보다 효율적으로 큐를 관리하며 좋은 성능을 제공한다.

2장에서는 혼잡상황 제어 알고리즘에 대한 개념과 RED에 대해 기술하였고 3장에서는 새로 제안한 알고리즘에 대하여 기술하였다. 4장에서는 시뮬레이터를 이용한 실험 결과에 대하여 기술 및 실제로 구현하여 간단히 성능을 분석하여 보았다. 5장에는 결론을 맺었다.

2. 관련 연구

2.1 혼잡 제어

현재 인터넷의 라우터나 게이트웨이에서 발생 할 수 있는 혼잡상황을 해결하기 위해서 여러 가지 혼잡제어 방법이 논의되고 있다. 그 중에서도 프로토콜 레벨의 혼잡 제어 방법으로 TCP와 같은 전송 프로토콜을 기반으로 한 혼잡 제어 방법이 많이 제시가 되었다. 또 다른 방법으로 네트워크 레벨의 혼잡 제어 방법으로 트래픽이 집중되는 라우터나 게이트웨이에서 향상된 큐 관리 알고리즘을 지원하여 혼잡 상황을 해결하는 방법이 제시되었다.

라우터나 게이트웨이에서 효과적으로 큐를 관리하는 혼잡제어 알고리즘으로 IETF에서 논의하고 있는 것은 크게 두 가지로 분류 할 수 있다. 첫 번째는 스케줄링 알고리즘으로 대표적인 방법으로는 FQ(Fair Queueing) 알고리즘이 있다. 이 알고리즘은 각 flow에 대하여 개별적인 큐를 분리하여 관리하는 per-flow방식을 사용한다. 그러나 흐름들에 관한 정보를 유지하고 처리하기 위해서 복잡한 알고리즘을 필요로 하기 때문에 고속의 라우터 처리 능력을 요구한다. 많은 flow를 갖는 네트워크 환경에서 널리 사용하기에는 많은 비용이 필요하다.

두 번째로 처음부터 간단히 설계된 큐 관리 알고리즘이 있다. 이 방법은 간단하면서도 어느 정도의 공정성을 유지하는 기능을 제공한다. 대표적인 알고리즘으로는 RED(Random Early Detection)가 있다. 이 방법은 모든 flow가 하나의 FIFO 큐를 통하여 처리가 되며 네트워크 상태에 따라 관리된다.

2.2 RED

기존의 네트워크에서 사용하는 Drop tail 방법에서의 혼잡상황을 해결하기 위해서 TCP와 같은 전송 프로토콜을 이용하여 혼잡제어를 한다. TCP의 혼잡제어 알고리즘은 큐의 오버플로우가 발생하여 패킷이 손실되면 일정 시간이 지난 후 패킷이 손실 된 것을 알고 소스의 전송률을 줄이므로 혼잡상황을 해결하는 방법이다. 하지만, TCP 혼잡제어 알고리즘이 동작하기 위해서 패킷 손실을 감지한 이후 전송률을 줄이는 동안 일정 시간이 필요하다. 결국 이 시간 동안 네트워크의 자원을 낭비하게된다. RED는 혼잡 상황이 발생하기 이전에 혼잡 상태를 종단 호스트에게 알려 줌으로 이러한 문제를 개선 할 수 있는 방법을 제공한다. 따라서 종단 호스트는 패킷이 폐기되기 전 소스의 전송률을 줄일 수 있다. 일반적으로 미리 혼잡 상황을 발견하여 혼잡제어를 수행하므로 네트워크 자원의 낭비가 적어 Drop tail 보다는 좋은 성능을 제공한다. 또한 RED는 항상 평균 큐 크기를 작게 유지하도록 큐를 관리하므로 큐의 대기 지연시간을 줄여 모든 flow의 전송 지연 시간을 줄여 준다.

그러나, 효율적으로 RED가 혼잡 상황을 미리 발견하여 혼잡을 제어하기 위해서는 링크용량을 수용 할 수 있는 충분한 크기의 큐가 필요하다. 만약 큐의 크기가 작아 혼잡상황을 알린 후 네트워크의 로드를 줄이기 이전에 패킷 폐기가 발생한다면 오히려 혼잡 제어로 인하여 기존의 방법보다 초기에 자원을 낭비하게 된다. 따라서 평균 큐 크기 정보를 이용 하여는 RED의 경우 효율적인 혼잡제어를 하기 위해서는 들어오는 패킷을 모두 수용 할 수 있는 충분한 크기의 큐 공간을 확보해야만 한다.

2.3 RED의 개선 방안

RED와 같이 패킷 폐기 방법으로 혼잡상황을 제어하는 방법에서 가장 중요한 요소는 어떤 정보에 기반으로 패킷 폐기 확률 계산하는 것이다. RED의 경우는 단순히 제한된 큐 공간을 통하여 계산된 평균 큐 크기를 이용하여 계산한다. 이렇게 계산된 패킷 폐기 확률 값은 평균 큐 크기에 선형적인 함수로 계산된다. 하지만 네트워크 트래픽이 동적으로 변화 하여 평균 큐 크기의 변화가 심하게 발생 할 경우 큐 크기 정보만으로 폐기 확률을 계산하기에는 문제가 있다.

본 논문에서는 이러한 RED의 단점을 보완 하여 작은 큐 크기에서도 패킷 손실 정보와 링크 이용 히스토리(link utilization history) 정보를 이용하여 효율적으로 큐를 관리하는 새로운 알고리즘을 제안 하였다. 제안하는 능동적인 큐 관리 알고리즘을 MRED(Modified Random Early Detection)라 부른다.

MRED는 네트워크 혼잡 상황 시 RED가 제공하는 패킷 폐기 확률 값을 휴리스틱한 방법을 통하여 계산하는 알고리즘이다. 이 알고리즘을 이용하면 효율적인 폐기 확률 값을 계산하여 패킷 손실로 인해 발생할 수 있는 자원의 낭비를 막을 수 있다.

3. MRED 알고리즘

MRED는 평균 큐 크기를 이용하여 혼잡상황을 제어한다. 큐 관리 방법은 평균 큐 크기를 최소 경계 값 (min_{th})과 최대 경계 값(max_{th})과 비교하여 처리한다. 평균 큐 크기가 최소 경계 값보다 작다면 패킷 폐기 없이 처리가 된다. 그러나, 평균 큐 크기가 최대 경계 값 보다 크다면 도착하는 모든 패킷은 폐기된다. 만약, 평균 큐 크기가 최소 경계 값과 최대 경계 값 사이에 존재한다면 도착하는 패킷은 혼잡상황의 정도에 따라 계산된 P_a확률로 폐기된다. 여기서 확률 값 P_a는 이전 패킷 손실 정보와 링크 이용 히스토리에 위해 계산된다. 본 논문에서 제안한 MRED의 알고리즘은 그림 1과 같다.

```
if min_th ≤ avg ≤ max_th
    if avg increase
            if current time - last up time > up time
                P_a + d_1

    if avg decrease
            if current time - last up time > up time )
                P_a - d_2
```

그림 1. MRED 알고리즘

MRED와는 다르게 RED의 경우 평균 큐 크기가 최소 경계 값 아래에 있는 대기(idle) 시간 이후 혼잡상황이 다시 발생되면 단지 평균 큐 크기에 따라 다시 선형적인 확률 값을 계산한다. 반면 MRED는 이전 네트워크의 히스토리를 가지고 있는 P_a를 기초로 하여 폐기 확률 값을 계산하므로 폐기된 패킷에 대한 재 전송으로 발생 할 수 있는 혼잡 상황에 대하여 제어 할 수 있다. 따라서 MRED는 RED보다 휴리스틱한 방법을 통하여 정확히 혼잡 제어를 할 수 있다. 그림 2는 MRED의 폐기 확률 값의 변화를 도시한 그림이다.

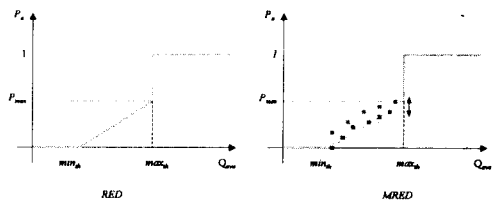

그림 2. 폐기 확률 값의 변화

4. 실험 및 성능 평가

4.1 공정성에 대한 실험

새로 제안한 MRED의 성능 평가를 위하여 본 논문에서는 LBNL(Lawrence Berkely National Laboratory)의 ns(Network Simulator) 네트워크 시뮬레이터를 이용하였다. MRED은 서로 다른 전송률의 flow들에게 어느 정도의 공정성을 제공하는지 알아보기 위해서 그림 3과 같은 네트워크을 구성하여 간단히 실험을 하였다.

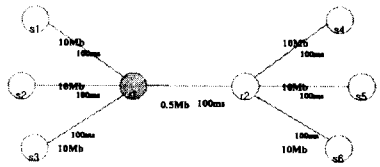

그림 3. 시뮬레이션 네트워크 구성

실험 방법은 왼쪽 호스트에서 오른쪽 호스트로 요구 전송률이 100Kbps, 200Kbps, 300Kbps인 exponential 트래픽을 같이 전송하였을 때 r1과 r2사이에 Drop tail, RED, MRED를 이용 시 요구 rate가 다른 3개의 flow를 모니터링 하였다. 그림 4의 MRED의 경우 모든 flow에 대하여 공정하게 서비스하므로 전체적으로 네트워크의 성능이 개선됨을 알 수 있다.

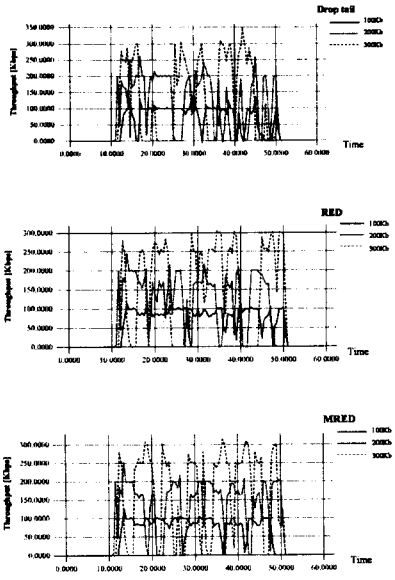

그림 4. Drop tail, RED, MRED 공정성 비교

4.2 전송 성능과 지연 시간 대한 실험

MRED가 큐 크기에 따라 어떠한 성능을 내는지 알아보기 위해서 다음과 같은 실험을 하였다. 실험을 위하여 그림 5와 같은 네트워크 망에서 먼저 s1과 s3사이에 TCP를 이용한 파일 전송을 하였고 임의의 시간 이후에 s2와 s4사이에 TCP를 이용한 파일 전송을 하였다. 큐의 크기를 변화시킬 때의 r1과 r2사이의 전송 성능을 모니터링 하였다.

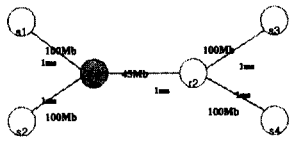

그림 5. 시뮬레이션 네트워크 구성

전송 성능은 그림 6에서 보는 것과 같이 버퍼의 크기가 증가함에 따라 전송성능이 증가함을 알 수 있고 MRED가 다른 라우터 알고리즘 보다 동일한 버퍼 크기에서 좋은 성능을 나타냄을 보여 주고 있다.

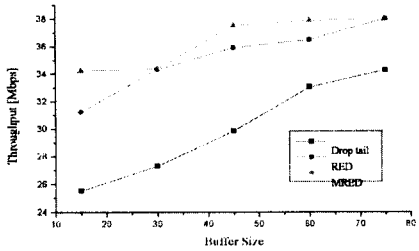

그림 6. 전송 성능 실험

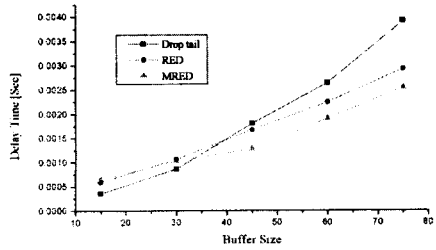

그림 7. 지연시간 실험

평균 전송 지연 시간에 대한 실험에서는 그림 7에서 보는 것과 같이 버퍼 크기가 증가함에 따라 평균 지연시간이 증가하고 있음을 보여준다. 일반적으로 전송지연 시간은 버퍼 크기에 비례하여 버퍼 크기가 커질수록 지연시간이 길어진다. MRED의 경우 RED보다 전반적으로 좋은 성능을 보여 주고 있다. 위의 실험 결과에서와 같이 MRED가 RED와 Drop tail 보다 개선된 성능을 나타냄을 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 동적으로 변화하는 네트워크 트래픽에 대하여 평균 큐 크기 정보로 혼잡 상황을 제어하는 RED 알고리즘을 개선한 새로운 MRED 알고리즘을 제안하였다. MRED는 휴리스틱한 방법을 통하여 폐기 확률 값을 계산한다. 제안한 알고리즘을 검증하기 위하여 실험을 통하여 공정성 및 성능, 지연 시간에 대하여 평가하였다

향후 연구 과제로는 비반응(unresponsive) flow에 대한 공정성을 개선하는 방법에 대한 연구가 수행되어야 하며, 실제 네트워크 환경에서의 적용에 대한 연구가 수행되어야 할 것이다.

참 고 문 헌

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

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

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

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

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

[6] Pan, R., Prabhakar, B., and Psounis, K., "CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation", Proceedings of INFOCOMM2000, February 2000.

[7] Feng, W., Kandlur, D., Saha, D., Shin, K., "Blue: A New Class of Active Queue Management Algorithms", Univ. of Michigan CSE-TR-387-99, April 1999.

[8] UCB/ LBNL/ VINT, "Network Simulator ns (Version 2)", http://www-mash.cs.berkeley.edu/ns/.