혼잡제어의 안정된 동작을 위한 새로운 큐 관리 알고리즘

구자헌[U], 정광수, 오승준
광운대학교 전자공학부 컴퓨터통신연구실
jhkoo@adams.gwu.ac.kr, kchung@daisy.gwu.ac.kr, sjoh@daisy.gwu.ac.kr

A New Queue Management Algorithm for Stabilized Operation of Congestion Control

Jahon Koo[U], Kwangsue Chung, Seoungjun Oh
School of Electronics Engineering, Kwangwoon Univ.

요 약

현재의 인터넷 라우터는 Drop tail 방식으로 큐 안의 패킷을 관리한다. 따라서 네트워크 트래픽의 지수적인 증가로 인해 발생하는 혼잡 상황을 명시적으로 해결 할 수 없다. 이 문제를 해결하기 위해 IETF (Internet Engineering Task Force)에서는 RED(Random Early Detection) 알고리즘과 같은 능동적인 큐 관리 알고리즘 (AQM: Active Queue Algorithm)을 제시하였다. 하지만 RED 알고리즘은 네트워크 환경에 따른 매개 변수의 설정의 어려움을 가지고 있어 잘 못 된 매개변수 설정으로 인하여 네트워크 성능을 저하시키는 문제를 발생시키며 전체 망에 불안정한 혼잡제어를 야기 시킨다. 본 논문에서는 기존의 AQM를 개선한 SOQuM (Stabilized Operation of Queue Management) 알고리즘을 제안하였다. 제안한 알고리즘의 성능을 검증하기 위해 기존의 방법과 시뮬레이션을 이용하여 비교하였다.

1. 서론

Drop tail 방식을 이용하는 라우터로 구성되어져 있는 현재 인터넷은 IP 프로토콜을 기반으로 하는 최선형 서비스를 하고 있다. 이러한 구조는 지수적으로 증가하는 인터넷 트래픽으로 인하여 발생하는 혼잡상황을 효과적으로 제어 할 수 없다. 이런 문제를 해결하기 위해서는 서비스를 이용하는 양 종단 및 네트워크의 트래픽이 집중되는 라우터나 게이트웨이에서 혼잡상황을 제어하는 혼잡제어 알고리즘이 필요하다. IETF(Internet Engineering Task Force)에서는 효과적인 혼잡제어를 위해 대표적인 능동적인 큐 관리 알고리즘(Active Queue Management)인 RED (Random Early Detection) 알고리즘을 권고하고 있다[1,2,3]. RED 알고리즘은 기존의 다른 큐 관리 알고리즘이 가지고 있던 여러 문제를 완화시키며 전역동기화 및 패킷 손실 문제를 해결하였다[4].

그러나 RED 알고리즘의 경우 다양한 매개변수로 구성되어져 있으며 이로 인한 설정의 어려움이 있다. 또한 네트워크 부하(load)에 맞추어 매개 변수(parameter)를 정확히 설정하지 않을 경우 전체 혼잡제어 구조를 불안정하게 만들어 기존의 Drop tail 방식 보다 효율이 떨어지는 결과를 초래하기도 한다.

본 논문에서는 현재 혼잡제어 알고리즘의 문제점을 개선하여 네트워크 특성 변화에 대한 매개변수 민감도를 줄여 안정된 혼잡제어 구조를 만드는 새로운 큐 관리 알고리즘인 SOQuM(Stabilized Operation of Queue Management) 알고리즘을 제안하였다. 제안한 알고리즘은 기존의 큐 관리 알고리즘 보다 혼잡제어 시 안정된 동작 및 좋은 성능을 보여주었다.

2장에서는 관련 연구로 혼잡제어와 능동적인 큐 관리 알고리즘의 동작에 대해 기술하였고 3장에서는 현재 인터넷의 혼잡제어 구조 및 문제점에 대하여 기술하였다. 4장에서는 새로 제안한 SOQuM 알고리즘에 대하여 기술하였다. 5장에서는 시뮬레이터를 이용하여 제안한 알고리즘을 검증하였다. 마지막으로 6장에는 결론을 맺었다.

2. 관련 연구

2.1 혼잡 제어

현재 인터넷 구조에서 대표적인 혼잡제어 방법을 가지고 있는 알고리즘으로는 종단간의 흐름을 제어하는 전송프로토콜이 있다. 대표적인 전송 프로토콜인 TCP (Transmission Control Protocol) 프로토콜은 송신단에 일정 시간동안 Ack (acknowledgment) 패킷이 도착하지 않아 타임아웃이 발생하거나, 데이터 패킷의 손실을 의미하는 중복 Ack (Duplicate -Ack)을 받으면, 네트워크에 혼잡이 일어났다고 가정한다. 그리고 혼잡 윈도우(congestion window)의 크기를 줄이고 전송 속도를 천천히 증가시키는 Slow start와 Congestion Avoidance를 실행하여 네트워크 혼잡을 완화시켜 네트워크에서의 패킷 손실을 줄인다. 즉, 전송 중에 발생하는 패킷 손실의 원인을 네트워크 혼잡에 두고 이를 줄이기 위한 방향으로 개선되어 온 TCP 프로토콜은 통신의 양 종단간 신뢰성 있는 데이터 전송을 보장하며 인터넷에서 가장 보편적으로 쓰이고 있는 전송 프로토콜이다[2].

2.2 능동적인 큐 관리

현재 인터넷의 라우터나 게이트웨이에서 발생하는 혼잡상황을 해결하기 위해서 여러 가지 혼잡제어 방법이 연구되어지고 있다. 최근 그 중에서도 트래픽이 집중되어 혼잡상황의 문제를 야기 시키는 라우터나 게이트웨이의 큐에서 명시적으로 혼잡상황을 제어하려는 방법이 활발히 진행중에 있다. 이 방법은 혼잡상황이 발생한 큐에 향상된 큐 관리 알고리즘을 지원하여 혼잡 상황을 해결하는 방법이다[1].

IETF에서 권고하고 있는 라우터나 게이트웨이의 큐를 관리하는 혼잡제어 알고리즘은 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 처리

하기 위해서 복잡한 알고리즘을 필요로 하기 때문에 고속의 라우터 처리 능력을 요구한다. 또한, 많은 flow를 갖는 네트워크 환경에서 널리 사용하기에는 많은 비용이 필요로 하는 어려움이 있다.

두 번째로 처음부터 간단한 구조로 설계된 능동적 큐 관리 알고리즘이 있다. 이 방법은 간단하면서도 어느 정도의 공정성 및 혼잡상황을 제어하는 기능을 제공한다. 대표적인 능동적 큐 관리 알고리즘으로는 RED가 있다. 이 방법은 모든 flow가 하나의 큐를 통하여 처리가 되며 네트워크 혼잡정도에 따라 관리된다. 따라서 스케줄링 알고리즘보다 적은 비용으로도 네트워크에 발생하는 혼잡 상황 문제를 해결할 수 있다.

3. 현재 인터넷의 혼잡제어 구조 및 문제점

3.1 전송제어 시스템과 귀환(feed-back) 제어모듈

현재 인터넷에서 사용하는 혼잡제어 구조에서는 혼잡상황을 해결하기 위해서 종단 간 흐름 제어 (End-to-End Flow Control)방법을 이용하고 있다. 이 방법은 전송속도를 제어하는 전송제어 시스템과 네트워크 상황을 능동적으로 제어하는 귀환제어모듈로 구성되어 있다. 전송제어 시스템은 TCP와 같은 전송 프로토콜을 사용하여 Ack 패킷을 통해 전달 받은 네트워크 정보를 이용하여 TCP 송신단 (TCP Sender)에서 전송 속도를 윈도우 기반 (window-based)으로 제어한다. 귀환 제어모듈은 AQM과 같은 라우터 알고리즘을 사용하며 네트워크 트래픽의 부하에 따라 혼잡상황을 방지하기 위해 패킷 폐기율 및 흐름을 제어한다[2,5].

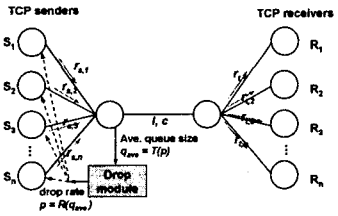


그림 1. n개 플로우의 종단간 흐름제어 구조

일반적인 종단간 흐름제어구조는 그림 1과 같이 도시 할 수 있다. TCP의 혼잡제어 알고리즘은 큐의 오버플로우(overflow)가 발생하거나 패킷이 폐기되어 패킷이 손실되면, Ack 정보를 통하여 혼잡상황을 인지하고 송신단의 전송 속도 줄여 혼잡상황을 제어하려 한다. 일반적으로 TCP 송신단에서 제어하는 전송 속도는 식(1)에서 보는 것과 같이 네트워크에서 발생하는 패킷 폐기율(drop rate)에 위해서 결정된다.

$$T \le \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}} \qquad (1)$$

이러한 전송속도의 조절은 네트워크의 평균 큐 크기나 트래픽의 부하를 변화 시킨다. 이러한 변화는 큐를 관리하는 귀환 제어모듈(drop module)에서 송신단에 전달되는 네트워크 정보인 패킷 폐기율를 제어한다. 종단간 흐름제어 동작의 상관관계는 그림 2와 같이 표현 할 수 있다.

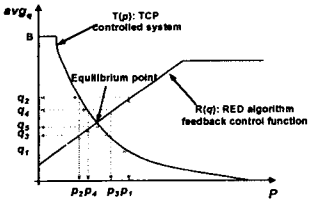


그림 2. RED를 이용하는 귀환 제어모듈의 평형 상태

그림 2와 같이 전송속도와 패킷 폐기율은 평균 큐 크기와 패킷 폐기 확률 값으로 표현이 되며 두 상관관계에 있는 함수($q_{ave} = T(p)$, $p = R(q_{ave})$)는 평형상태(Equilibrium)로 수렴하도록 동작한다. 일반적으로 안정적인 혼잡제어 동작을 할 경우 평형 상태로 수렴한다. 만약 평형상태로 수렴을 하지 못 할 경우 높은 패킷 폐기율을 가지게 되며 큐의 크기가 심하게 변동하는 불안정한 혼잡제어 동작을 한다[5].

3.2 RED 알고리즘의 문제점

RED 알고리즘은 혼잡 상황이 발생하기 이전에 평균 큐 크기를 기반으로 그림 3과 같이 혼잡 상태를 판별하여 혼잡 상태에 대한 정보를 패킷의 폐기나 ECN(Explicit Congestion Notification)을 이용한 표시 방법을 이용하여 종단 호스트에게 혼잡정보를 알려주는 방법이다.

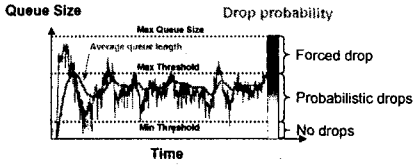


그림 3. RED 알고리즘의 동작

일반적으로 미리 혼잡 상황을 발견하여 혼잡제어를 수행하므로 네트워크 자원의 낭비가 적어 Drop tail 보다는 좋은 성능을 제공한다. 그러나 RED 알고리즘의 경우 다양한 매개변수로 설정되어 동작하기 때문에 잘못 된 매개변수를 설정 할 경우 전체적인 성능이 Drop tail 보다 떨어진다[6]. 표 1은 RED 알고리즘의 다양한 매개 변수를 보여 주고 있다.

표 1. RED 제어 매개 변수

$qlen$	최대로 수용 할 수 있는 큐 공간 크기
min_{th}	폐기 확률을 결정하는 최소 임계 값
max_{th}	폐기 확률을 결정하는 최대 임계 값
w_q	Avg_q 변화의 특성을 결정하는 가중치 값
max_p	폐기 특성을 결정하는 최대 확률 값

즉 적절한 매개 변수 값이 설정되어 있지 않을 경우 그림 2의 평형상태로 수렴을 하지 못하여 그림 4와 같은 불안정한 혼잡제어 동작을 보인다. 그림 4에서는 큐의 크기가 심하게 변동(oscillation)하는 동작을 보이고 있다. 이러한 동작은 네트워크에 높은 패킷 폐기율을 보인다.

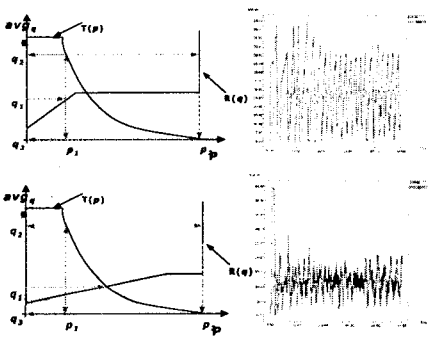


그림 4. 불안정한 혼잡제어 동작

본 논문에서는 현재 혼잡제어 알고리즘의 문제점을 개선하여 네트워크 특성 변화에 대한 매개변수 민감도를 줄여 안정된 혼잡제어 구조를 만드는 새로운 큐 관리 알고리즘인 SOQuM(Stabilized Operation of Queue Management) 알고리즘을 제안하였다.

4. SOQuM 알고리즘

현재 혼잡제어 알고리즘의 문제점을 개선하기위해서 제안한 SOQuM 알고리즘은 동적인 네트워크 변화에 대해 항상 안정적으로 혼잡제어를 하는 알고리즘이다. 또한 간단한 매개변수를 가지는 구조로 설계하여 손쉽게 실제 네트워크에 사용할 수 있는 알고리즘이다. 제안한 SOQuM 알고리즘은 그림 5와 같은 동작 특성을 가진다. SOQuM 알고리즘은 EWMA(Exponentially Weighted Moving Average)값으로 평균 큐 크기를 계산하여 혼잡상황의 정도에 따라 혼잡제어를 수행한다. 혼잡제어

방법은 혼잡제어 정도에 따라 패킷 폐기율을 조작한다. 혼잡상황 정도가 네트워크 관리자가 설정하는 기본 평균 큐 크기(네트워크의 평균 지연시간을 조절할 수 있다.)인 cq_{th} (Congestion Queue)값 보다 클 경우 혼잡제어 동작을 수행하며 초기 구간은 조심스럽게(Conservative) 모드로 동작하며 혼잡상황의 정도가 심해 질 경우 과감하게(Aggressive) 모드로 혼잡 상황을 제어한다. SOQuM 알고리즘은 그림 6과 같다.

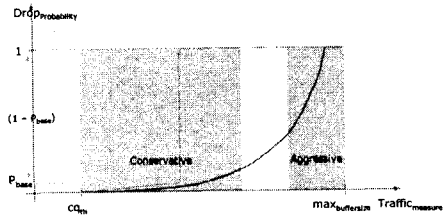


그림 5. SOQuM 알고리즘의 동작 특성

$$\text{if } (q_{ave} < cq_{th})$$
$$\quad No\ Packet\ Drop$$
$$\text{if } (cq_{th} < q_{ave})$$
$$Packet\ Drop: P_{cc} = P_{base} + (1 - P_{base})\frac{(q_{ave} - cq_{th})}{(max_{buffer} - cq_{th})}$$

그림 6. SOQuM 알고리즘

본 논문에서 제안하는 SOQuM 알고리즘은 동적으로 변화하는 네트워크 특성 대해서 항상 그림 7과 같은 구조로 안정적인 혼잡제어를 수행 한다. SOQuM 알고리즘의 안정된 혼잡제어는 큐의 크기의 변동의 정도를 줄여 전체적인 패킷 폐기(손실)율을 줄여준다.

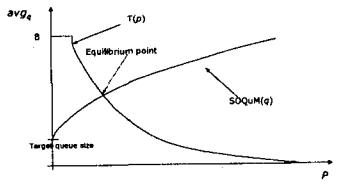


그림 7. SOQuM 알고리즘을 이용하는 귀환 제어모듈의 평형 상태

5. 실험 및 성능 평가

제안한 SOQuM 알고리즘의 성능을 알아보기 위해서 ns를 이용하였다. 기존의 방법과 비교하여 얼마나 안정적으로 혼잡제어를 하는지 큐의 크기의 변화 및 각 플로우 동작 특성을 알아보았다[7].

그림 8과 같은 네트워크 환경에서 총 8개의 플로우가 전송되는 혼잡링크에 대하여 큐의 변화를 라우터 r_0에서 모니터링 하였다.

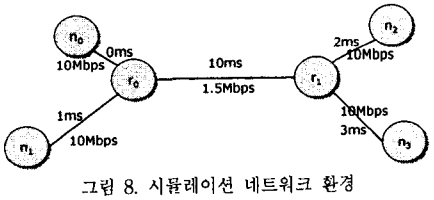
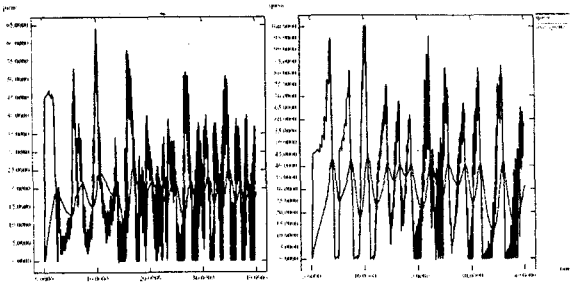


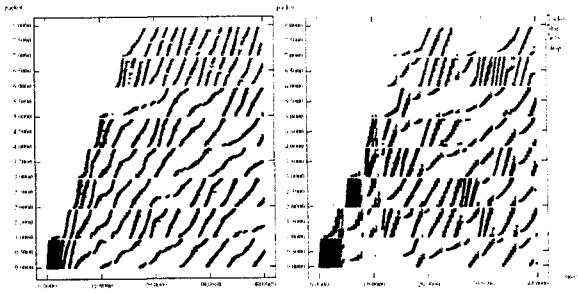
그림 8. 시뮬레이션 네트워크 환경

그림 9에서와 같이 RED 알고리즘의 경우 트래픽 특성에 맞지 않아 매우 불안정한 동작을 보이고 있다. SOQuM 알고리즘의 경우 RED 알고리즘 보다 큐 크기의 변동이 작으며 안정적인 동작을 보이고 있다. 이러한 RED 알고리즘의 불안정한 동작은 그림 10과 같이 연속적인 패킷 폐기율을 보인다. 이 실험에서 SOQuM 알고리즘의 경우 3.398%의 패킷 폐기율을 보였으며 RED 알고리즘의 경우 10.399%의 패킷 폐기율을 보였다. 실험 결과를 통해서 제안한 SOQuM 알고리즘은 기존의 RED 알고리즘 보다 안정적인 혼잡제어 동작을 보임을 확인 할 수 있었다.



(a) SOQuM (b) RED

그림 9. SOQuM 과 RED 알고리즘의 큐 변화



(a) SOQuM (b) RED

그림 10. 각 플로우의 동작 특성 변화

6. 결론 및 향후 과제

본 논문에서는 동적으로 변화하는 네트워크 트래픽으로 인하여 발생하는 혼잡상황을 기존의 방법 보다 안정적으로 제어하는 새로운 큐 관리 방법인 SOQuM 알고리즘을 제안하였다. 본 논문에서 제안한 알고리즘은 기존 방법보다 간단한 매개 변수 설정으로 기존의 RED 알고리즘의 문제점을 개선하였고 시뮬레이션을 통해 성능을 검증 및 확인하였다.

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] Firoiu, V., and Borden, M., "A Study of Active Queue Management for Congestion Control", Proceedings of INFOCOM 2000, 2000.

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

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