인터넷 혼잡제어를 위한 안정적인 큐 관리 알고리즘

(A Stabilized Queue Management Algorithm for Internet Congestion Control)

구 자 헌 † 정 광 수 ††

(Jahon Koo) (Kwangsue Chung)

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(Active Queue Management)알고리즘을 제시하였다. 하지만 RED 알고리즘은 네트워크 환경에 따른 매개 변수의 설정의 어려움을 가지고 있어 잘못된 매개변수 설정으로 인하여 네트워크 성능을 저하시키는 문제를 발생시키며 전체 망에 불안정한 혼잡제어를 야기 시킬 수 있다.

본 논문에서는 기존의 큐 관리 알고리즘을 개선한 SQM(Stabilized Queue Management) 알고리즘을 제안하였다. SQM 알고리즘은 다양한 네트워크 조건에 대해 좋은 성능을 얻을 수 있도록 쉽게 매개 변수 설정이 가능하며 불필요한 패킷 폐기율을 줄여 효율적으로 혼잡상황을 제어한다. 제안한 알고리즘의 성능을 검증하기 위해 기존의 방법과 시뮬레이션을 이용하여 비교하였다.

키워드 : 혼잡제어, RED, 능동적 큐 관리, 스케줄링 알고리즘, 인터넷 QoS

Abstract In order to reduce the increasing packet loss rates caused by an exponential increase in network traffic, the IETF(Internet Engineering Task Force) is considering the deployment of active queue management techniques such as RED(Random Early Detection). But, RED configuration has been a problem since its first proposal. This problem is that proposed configuration is only good for the particular traffic conditions studied, but may have detrimental effects if used in other conditions. While active queue management in routers and gateways can potentially reduce packet loss rates in the Internet, this paper has demonstrated the inherent weakness of current techniques and shows that they are unstable for the various traffic conditions. The inherent problem with these queue management algorithms is that they all use static parameter setting.

In this paper, in order to solve this problem, a new active queue management algorithm called SQM(Stabilized Queue Management) is proposed. This paper shows that it is easy to parameterize SQM algorithm to perform well under different congestion scenarios. This algorithm can effectively reduce packet loss while maintaining high link utilizations and is good for the various traffic conditions.

Key words : Congestion Control, RED, Active queue management, Scheduling Algorithm, Internet QoS

1. 서 론

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 종단간의 혼잡제어 및 네트워크의 트래픽이 집중되는 라우터나 게이트웨이에서 혼잡상황을 제어하는 라우터 알고리즘이 필요하다. 그리고 최

· 본 연구는 한국과학재단 특정기초연구[R01-2002-000-00179-0(2002)]의 지원에 의해 수행되었음

† 비 회 원 : 광운대학교 전자통신공학과
jhkoo@innowireless.co.kr
†† 종신회원 : 광운대학교 전자공학부 교수
kchung@kw.ac.kr

논문접수 : 2002년 11월 7일
심사완료 : 2003년 8월 14일

근 인터넷 서비스의 빠른 확산과 이용자의 급증에 따른 다양한 사용자의 요구에 대한 처리와 사용자의 서비스 질(Quality of Service)의 개선에 대한 연구가 많이 진행 중에 있다. 특히, 최근 인터넷 서비스의 성능을 개선하기 위해서 라우터 알고리즘에 대한 연구가 활발하게 진행되고 있으며 그 중에서도 구현이 용이하며 적은 복잡성(complexity)을 가지는 큐 관리 알고리즘에 대한 연구가 활발히 진행되어 지고 있다.

IETF(Internet Engineering Task Force)에서는 효과적인 혼잡제어와 인터넷 서비스의 성능 개선을 위해 대표적인 능동적인 큐 관리 알고리즘(Active Queue Management)인 RED (Random Early Detection) 알고리즘을 권고하고 있다[1-3]. 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 SQM(Stabilized Queue Management) 알고리즘을 제안하였다. 제안한 알고리즘은 기존의 큐 관리 알고리즘 보다 혼잡제어 시 안정된 동작과 좋은 성능을 보여주었다.

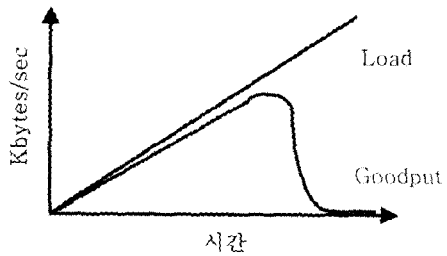
2장에서는 관련 연구로 혼잡제어와 능동적인 큐 관리 알고리즘의 동작에 대해 기술하였고 3장에서는 현재 인터넷의 혼잡제어 구조 및 문제점에 대하여 기술하였다. 4장에서는 새로 제안한 SQM 알고리즘에 대하여 기술하였다. 5장에서는 시뮬레이터를 이용하여 제안한 알고리즘을 검증하였다. 마지막으로 6장에는 결론을 맺었다.

2. 관련 연구

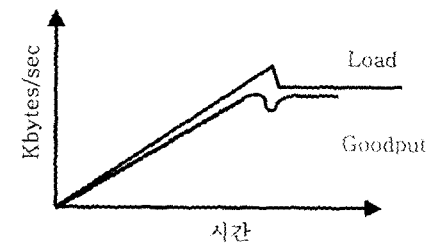
2.1 인터넷 혼잡 제어

현재 인터넷 구조에서 사용되어지고 있는 대표적인 혼잡제어 방법은 크게 두 가지로 분류 할 수 있다. 종단간(end-to-end) 흐름제어를 하는 전송 프로토콜과 혼잡상황이 발생하는 라우터에서 패킷을 직접 제어하는 라우터 알고리즘이 그것이다. 현재 인터넷에 많이 사용되고 있는 대표적인 전송 프로토콜인 TCP(Transmission Control Protocol) 프로토콜은 송신단(sender)에 일정 시간동안 Ack(acknowledgment) 패킷이 도착하지 않아 타임아웃이 발생하거나, 데이타 패킷의 손실을 의미하는 중복 Ack(duplicate-Ack)을 받으면, 네트워크에 혼잡이 일어났다고 가정한다. 즉, 전송 중에 발생하는 패킷 손실의 원인을 네트워크 혼잡에 두고 이를 줄이기 위한 방향으로 개선되어 온 TCP 프로토콜은 통신의 양 종단간 신뢰성 있는 데이타 전송을 보장하며 인터넷에서 가장 보편적으로 쓰이고 있는 전송 프로토콜이다[2].

그러나 네트워크의 혼잡상황 발생시 패킷 손실 정보를 혼잡상황으로 인지하여 동작하는 전송 프로토콜의 경우 의도적이거나 동적인 네트워크 혼잡상황을 근본적으로 해결하지는 못하는 한계가 있다. 이는 현재 인터넷 라우터에서 사용하고 있는 대표적인 라우터 알고리즘인 drop-tail 알고리즘의 구조상 발생되는 몇가지 문제로 인해 최악의 경우 혼잡붕괴(congestion collapse) 현상이 발생할 수 있다. 혼잡붕괴 현상은 그림 1의 (a)와 같이 네트워크의 트래픽의 량(load)은 증가하지만 실제 수신측(receiver)에서 유용하게 사용하는 유효 전송률(goodput)은 현저하게 떨어지는 현상이다. 결국 혼잡붕괴 시간동안 네트워크 자원의 낭비를 초래한다. 따라서 이를 해결하여 그림 1의 (b)와 같은 혼잡제어를 위해서는 근본적인 라우터 알고리즘의 문제를 해결하는 라우터 알고리즘이 필요하다.



(a) 혼잡붕괴



(b) 혼잡제어

그림 1 혼잡붕괴와 혼잡제어

2.2 혼잡제어를 위한 라우터 알고리즘

Per-hop 단위로 패킷 처리를 하는 IP 네트워크에서의 혼잡제어 방법은 혼잡성이 주로 발생되는 라우터나 게이트웨이의 출력(output interface)링크에서 링크 자원

인 큐를 효율적으로 관리하는 라우터 알고리즘을 이용하는 방법이다. 일반적으로 라우터는 유입되는 패킷을 라우팅하기 위하여 라우터로 들어오는 패킷들을 큐에 임시적으로 저장한다. 이때 가장 간단하게 저장하는 방법으로 FIFO(First-In First-Out) 큐가 사용된다. 만약 라우터로 들어오는 패킷의 입력 트래픽이 라우터를 통하여 나가는 출력링크로 처리할 수 있는 트래픽을 넘는 경우 임시로 큐에 저장을 하고 저장이 안 된 패킷들은 모두 버려지게 된다. 이러한 기본적인 FIFO 처리 방식을 drop-tail이라 한다. 이 방법은 큐 관리가 가장 간단하여 구현하기는 쉽지만 flow간의 불공정한 출력링크 할당, 전역 동기화(global synchronization)에 따른 평균 지연 시간의 증가 및 성능 감소, 그리고 혼잡붕괴의 가장 큰 원인이 되는 "풀 큐(full queue)"와 "락 아웃(lock out)"을 발생시키는 문제점을 가지고 있어 혼잡상황 제어에 적합하지 않다[5].

현재 인터넷의 라우터나 게이트웨이에서 발생하는 혼잡상황을 해결하기 위해서 여러 가지 혼잡제어 방법에 대한 연구가 활발히 진행되어지고 있다. IETF의 RFC 2309문서에서 권고하고 있는 라우터나 게이트웨이의 큐를 관리하는 혼잡제어 알고리즘은 크게 두 가지로 분류할 수 있다. 첫 번째는 스케줄링 알고리즘으로 대표적인 방법으로는 FQ(Fair Queueing) 알고리즘이 있다[6]. 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. 또한, 많은 flow를 갖는 네트워크 환경에서 널리 사용하기에는 많은 어려움과 오버헤드가 필요하다.

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 RED 알고리즘이 있다. RED 알고리즘은 모든 flow가 하나의 큐를 통하여 처리가 되며 네트워크 혼잡정도에 따라 큐가 관리된다. 따라서 스케줄링 알고리즘보다 적은 비용으로도 네트워크에 발생하는 혼잡 상황 문제를 해결할 수 있다[1].

3. 현재 인터넷의 혼잡제어 구조 및 문제점

3.1 RED 알고리즘

RED 알고리즘은 혼잡 상황이 발생하기 이전에 평균 큐 크기를 기반으로 그림 2와 같이 평균 큐 크기의 변화의 정도를 기반으로 혼잡 상황을 판별하며, 혼잡 상태에 대한 피드백 정보를 패킷의 폐기나 패킷 헤더에 혼잡 상황을 표시(marking)하는 ECN(Explicit Congestion Notification)을 이용하여 종단 호스트인 송신 측에게 혼잡정보를 알려 준다.

RED 알고리즘은 EWMA(Exponential Weighted Moving Average)를 이용하여 평균 큐 크기를 계산하며 혼잡상황이 발견되면 평균 큐 크기에 따라 비례적으로 적절한 패킷 폐기 확률 값을 계산한다. 예를 들어 평균 큐 크기가 최소 경계 값(minimum threshold: min_{th})을 초과하면 패킷들은 패킷 폐기 확률 값에 따라 랜덤(random)하게 폐기되거나 IP 패킷 헤더의 ECN bit를 이용하여 표시한다. RED 알고리즘은 네트워크 혼잡상황을 피하기 위해서 TCP와 같이 피드백 정보를 기반으로 혼잡제어를 하는 전송 프로토콜과 상호 동작한다. 그리고 평균 큐 크기가 최대 경계 값(maximum threshold: max_{th})을 초과할 경우 모든 패킷들은 폐기 또는 표시한다.

RED 알고리즘의 동작 방법은 이처럼 기존의 혼잡제어 방법과는 다르게 혼잡상황을 예상하여 라우터에 도착하는 임의의 플로우의 패킷을 폐기하기 때문에 drop-tail 방법이 가지고 있는 "락 아웃"과 "풀 큐" 문제를 해결 할 수 있다. 또한 미리 혼잡상황을 발견하여 TCP의 전송률을 줄여 혼잡제어를 행하기 때문에 불필요한 패킷 손실로 발생하는 네트워크 자원의 낭비를 drop-tail 방법 보다 줄여 좋은 성능을 나타낼 수 있다. 그리고 효과적인 큐 관리로 평균 큐 크기를 작게 유지

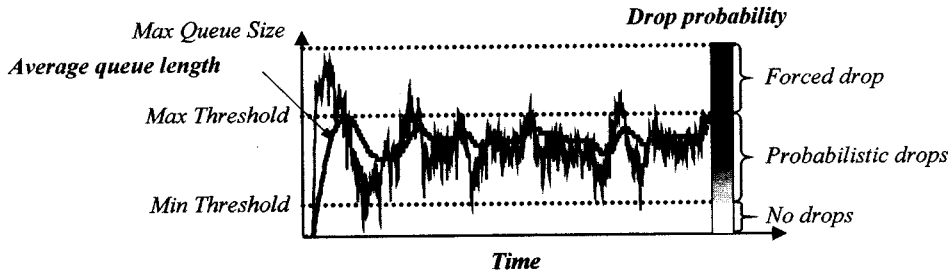


그림 2 RED 알고리즘의 동작

할 수 있어 큐잉(queueing)으로 생기는 전송지연 시간을 줄일 수 있다. 일반적으로 RED 알고리즘을 이용하는 방법이 drop-tail 방법을 사용하는 것보다는 좋은 성능을 제공한다[7].

drop-tail보다 좋은 성능을 제공하는 RED 알고리즘은 표 1과 같이 다양한 매개 변수에 의해서 제어되며 그 중에서도 P_{max} 값에 따라 전체적인 알고리즘의 특성이 결정된다[8].

표 1 RED 알고리즘의 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
P_{max}	폐기 특성을 결정하는 최대 확률 값

RED 알고리즘은 위의 표 1과 같이 다양한 매개변수로 설정되어 동작하기 때문에 만약 네트워크 특성에 맞지 않는 잘못된 매개변수로 설정이 될 경우 혼잡상황에 대하여 적절한 혼잡제어를 하지 못한다. 최악의 경우 잘못된 매개변수 설정으로 인하여 전체적인 성능이 drop-tail 보다 떨어질 수도 있다[9].

3.2 전송제어 시스템과 귀환(feed-back) 제어모듈

현재 인터넷에서 사용하는 혼잡제어 구조에서는 혼잡상황을 해결하기 위해서 종단간 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 AQM(Active Queue Management)과 같은 라우터 알고리즘을 사용하며 네트워크 트래픽의 부하에 따라 혼잡상황을 방지하기 위해 패킷 폐기율 및 흐름을 제어한다[2,5].

일반적인 종단간 흐름제어구조는 그림 3과 같이 도시할 수 있다. TCP의 혼잡제어 알고리즘은 큐의 오버플로우(overflow)가 발생하거나 패킷이 폐기되어 패킷이 손실되면, Ack 정보를 통하여 혼잡상황을 인지하고 송신단의 전송 속도를 줄여 혼잡상황을 제어하려 한다.

현재 인터넷을 통하여 사용하고 있는 TCP는 기본적으로 혼잡상황을 제어하기 위한 동작 메커니즘으로 아래와 같은 혼잡 회피(congestion avoidance)방법을 가

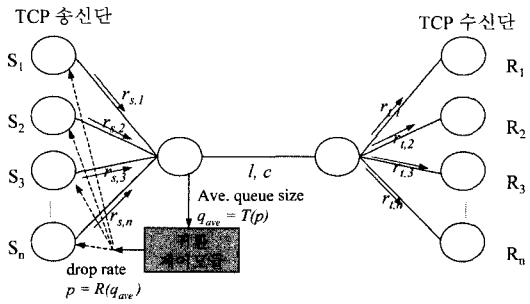


그림 3 n개의 플로우의 종단간 흐름제어 구조

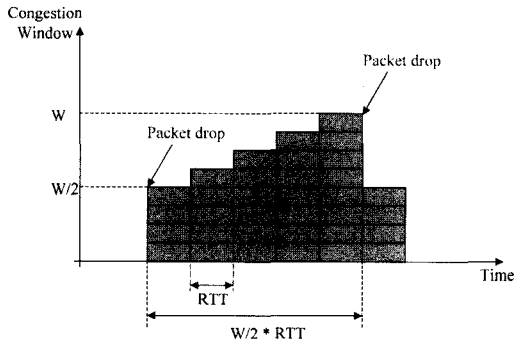


그림 4 TCP 혼잡 윈도우 크기의 변화

지고 있으며 동작은 다음과 같다. 예를 들어 TCP 플로우가 B 바이트의 패킷을 전송한다고 가정하면, 패킷이 망에서 폐기되거나 손실 될 때 TCP의 혼잡 윈도우는 W 개의 크기를 갖는다. TCP는 패킷 폐기나 손실에 의해 혼잡 윈도우의 크기를 반으로 줄이게 되고 다음 패킷 폐기 때까지 각 RTT(round trip time) 기간 동안 혼잡 윈도우의 크기를 하나씩 증가시킨다. 이를 도식하면 그림 4와 같다.

일반적으로 TCP 송신단에서 제어하는 전송 속도(T)는 식 (1)에서 보는 것과 같이 네트워크에서 발생하는 패킷 폐기율(drop rate : p)에 위해서 결정된다[3]. 여기서 R은 RTT, B는 패킷 크기를 말한다.

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

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.

이러한 동작의 상관관계는 *Victor Firoiu*의 논문[10]에 근거하여 TCP 플로우의 전송률을 나타내는 $f(p,R)$ 함수와 패킷 폐기 확률을 나타내는 p와 TCP 연결의 RTT인 R, TCP의 혼잡 윈도우 크기인 W, 그리고 평균 패킷 크기인 M으로 다음과 같은 수식을 유도할 수

있다[11].

$$W(p) = \frac{2}{3} + \sqrt{\frac{-8}{6p} + \frac{4}{9}} \quad (2)$$

$$Q(p, w) = \min(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w} \quad (3)$$

$$F(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \quad (4)$$

$$f(p, R) = \begin{cases} M \dfrac{\frac{1-p}{p} + \frac{W(p)}{2} + Q(p, W(p))}{R(W(p)+1) + \frac{Q(p, W(p))F(p)T_0}{1-p}} & \text{if } W(p) < W_{\max} \\ M \dfrac{\frac{1-p}{p} + \frac{W_{\max}}{2} + Q(p, W_{\max})}{R(\frac{1}{4}W_{\max} + \frac{1-p}{pW_{\max}} + 2) + \frac{Q(p, Wmax)F(p)T_0}{1-p}} & otherwise \end{cases} \quad (5)$$

TCP 각 플로우의 전송률로 인하여 발생되는 라우터 평균 큐 크기 $T(p)$는 식 (6)으로 표현할 수 있다.

$$T(p) = \begin{cases} \max(B, c(f^{-1}(p, c/n) - R_0)), & p \le p_0 \\ 0, & otherwise \end{cases} \quad (6)$$

귀환 제어모듈에서 결정하는 평균 큐 크기에 따른 패킷 폐기 함수는 RED 알고리즘을 가정할 경우 그림 5와 같은 패킷 폐기 함수를 가지며 식 (7)로 패킷 폐기 함수 $R(q_{ave})$을 유도할 수 있다. 여기서 B는 라우터의 최대 큐 크기이다.

$$R(q_{ave}) = \begin{cases} 0, & 0 \le q_{ave} < \min_{th} \\ \dfrac{q_{ave} - \min_{th}}{\max_{th} - \min_{th}} P_{\max}, & \min_{th} \le q_{ave} < \max_{th} \\ 1, & \max_{th} \le q_{ave} \le B \end{cases} \quad (7)$$

앞에서 유도한 $T(p)$와 $R(q_{ave})$을 이용하여 종단간 흐름제어 동작의 상관관계는 그림 6과 같은 관계를 도시할 수 있다.

그림 6과 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 평형 상태로 수렴을 하지 못할 경우 높은 패킷 폐기율을 가지게 되며 큐의 크기가 심하게 변동하는 불안정한 혼잡제어 동작을 한다[5].

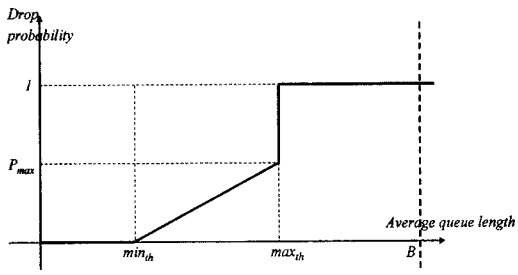


그림 5 RED 알고리즘의 패킷 폐기 함수

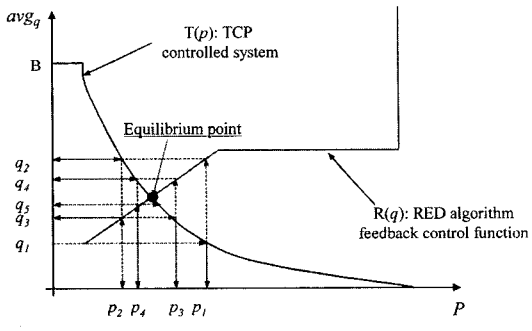


그림 6 종단간 흐름제어 모듈의 상관관계

3.3 RED 알고리즘의 문제점

네트워크 트래픽 특성에 대해 매개변수가 민감한 RED 알고리즘의 동작 특성을 자세히 알기 위해서 RED 알고리즘의 성능을 결정하는 주요 매개변수이며 RED 알고리즘의 대표적인 매개변수인 최대 확률 값 (P_{max} or max_p)에 따른 네트워크 트래픽의 특성과의 상관관계에 대하여 실험하여 보았다. 혼잡상황이 발생되는 중간 라우터에서 RED 알고리즘의 매개변수 P_{max}를 0부터 1까지의 다양한 값으로 조절할 때 라우터를 지나는 트래픽의 플로우수에 따른 패킷 폐기율을 알아보았다. 그림 7과 같이 P_{max} 값의 변화에 따라 패킷 폐기율은 P_{max}에 따라 변화함을 알 수 있다. 그림 7에서 RED 알고리즘은 W_q = *0.002*, q_{len} = *90KB*, min_{th} = *10KB*, max_{th} = *30KB*로 각 파라미터를 설정한 후 시험하였다.

실험 결과에서와 같이 플로우의 수에 따라 최소 패킷 폐기율을 결정하는 최적의 P_{max} 값이 서로 다른 것을 확인 할 수 있다. 즉 트래픽의 특성에 따라 RED 알고리즘의 매개변수가 설정되어야만 최소 패킷 폐기율 가지며 효과적으로 혼잡상황을 제어할 수 있음을 알 수 있다. 예를 들어 적절한 P_{max} 값이 설정되어 있지 않을 경우 앞장에서 설명한 종단간 흐름제어 구조의 두 모듈의 상관관계는 그림 7과 같은 분포를 가지게 되며 두 상관 함수 $q_{ave} = T(p)$, $p = R(q_{ave})$은 평형상태로 수렴

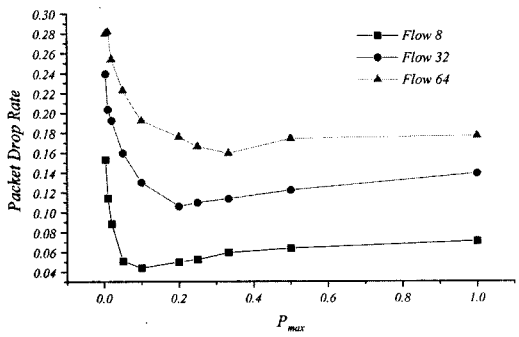
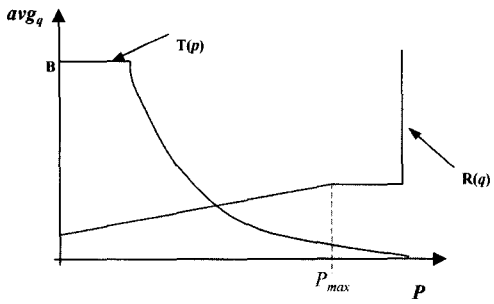
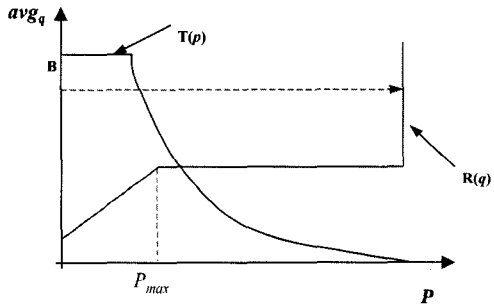


그림 7 패킷 폐기율에 대한 실험 결과



(a) 높은 Pmax 값



(b) 낮은 Pmax 값

그림 8 잘못 설정된 Pmax 값을 가지는 경우의 상관관계

하는 평형상태 점(p, q_{ave})을 가지지 못하게 된다.

종단간 흐름제어 구조의 상관관계가 그림 8의 (a)와 같이 적절한 P_{max} 값보다 너무 높은 P_{max} 값이 설정되어 있을 경우, 필요 이상의 패킷 폐기로 인하여 전체적인 큐 크기 및 평균 큐 크기가 작아져 좋은 전송 성능을 가지지 못한다. 그리고 적절한 P_{max} 값보다 너무 낮은 P_{max} 값을 가질 경우 그림 8의 (b)와 같은 종단간 흐름제어 구조의 상관관계를 가지며 큐 크기의 변화가 심해진다. 이로 인하여 불규칙적인 패킷 손실 및 재전송으로 인한 각 플로우의 지터(jitter)의 크기가 커지며, 라우터 버퍼의 overflow로 인하여 많은 패킷이 손실되어 전체적인 네트워크 성능이 감소한다. 앞 절에서 설명한 그림 6과 같이 네트워크 특성에 적합한 P_{max} 값이 선택되어지는 경우, 적당한 평균 큐 크기를 유지하며 큐 크기 변동이 적어져 안정적인 네트워크 서비스를 제공할 수 있다[12-14].

종단간 흐름제어 구조의 상관관계 그림에서 플로우의 수가 증가하거나 네트워크의 부하가 증가할 수 록 송신단의 전송률의 변화를 표현하는 전송 제어모듈 함수 $T(p)$는 오른쪽으로 점점 이동하는 특성을 가지고 있다. 예를 들어 RED 알고리즘의 매개변수들이 고정적일 경우, 트래픽의 양이 증가하게 되면 고정적으로 설정되어 있는 RED 알고리즘의 매개변수 P_{max} 값 밖으로 전송 제어모듈 함수 $T(p)$가 커져 평형 상태로 수렴을 하지 못하는 관계를 가질 수 있다. 또한 필요 이상의 큰 P_{max} 값이 설정되어질 경우 높은 패킷 폐기율로 인하여 낮은 평균 큐 크기를 가지게 된다. 이 경우 전체적인 전송률이 떨어지고 또 큐의 크기 변화가 심해져 네트워크 효율성이 나빠진다.

따라서 네트워크 특성(플로우의 수)에 대해 매개변수가 민감한 RED알고리즘의 경우 실제 네트워크에서 사용의 어려움이 있다. 최근에는 능동적으로 동작하는 새로운 큐 관리 알고리즘에 대한 연구가 진행 중에 있다. 이 방법은 플로우 수에 따라 즉 네트워크의 변화에 따라 적절한 P_{max} 값을 능동적으로 조절하는 방법이다. 하지만 적절한 P_{max}을 계산하기 위해서는 현재 네트워크 정보를 보다 명시적으로 알아야 한다. 결국 명시적인 정보를 얻어야 하는 어려움이 있으며 이상적인 P_{max} 값을 위해서는 per-flow단위의 전체 트래픽 량을 계산해야하는 오버헤드가 필요하다[13].

본 논문에서는 동적으로 변화하는 네트워크 트래픽으로 인하여 RED 알고리즘의 중요 매개변수인 P_{max} 값이 잘못 설정되어 발생하는 높은 패킷 폐기율을 해결하기 위해서 매개변수 민감도가 낮으며 특별한 오버헤드 없이 동적인 네트워크 트래픽 환경에서 효과적으로 동작하는 SQM(Stabilized Queue Management) 알고리즘을 제안하였다.

4. SQM(Stabilized Queue Management) 알고리즘

본 논문에서는 기존의 RED 알고리즘이 가지는 다양한 네트워크 특성에 따른 매개 변수의 설정의 어려움과 잘못된 매개변수 설정으로 인한 네트워크 성능을 저하시키는 문제를 개선한 SQM 알고리즘을 제안 하였다. 제안한 알고리즘은 기존 알고리즘 혼잡제어 구조상 발생되었던 불안정한 혼잡제어 구조를 개선하기 위해 총 4개의 혼잡구간을 분리하여 동작한다.

SQM 알고리즘의 혼잡상황에 따른 동작은 그림 9와 같다. SQM 알고리즘은 EWMA(Exponentially Weighted Moving Average)값으로 평균 큐 크기를 계산하여 라우터에 유입된 트래픽의 정도를 판단한다, 계산 평균 큐 크기에 비례적으로 혼잡상황의 정도에 따라 혼잡제어를 수행하는데 총 4개의 구간으로 혼잡구간을 분리하여 혼잡상황 정도에 따라 패킷 폐기율을 조작한다. SQM 알고리즘의 혼잡구간은 네트워크 관리자가 네트워크 특성를 고려해서 설정하는 cq_{th}(Congestion Queue)값과 SQM 알고리즘의 혼잡제어 특성을 결정하는 $\alpha(0 \leq \alpha \leq 1)$을 기반으로 구분이 된다. 4개의 구간

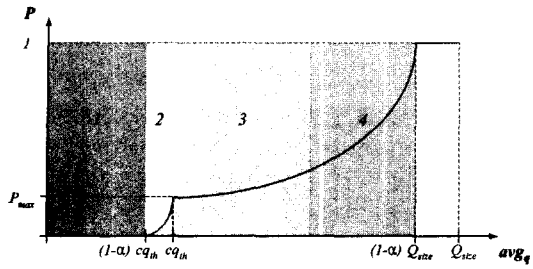


그림 9 SQM 알고리즘

- *if* $q_{avg} < (1-\alpha) \cdot cq_{th}$
  - No Packet Drop
- *if* $(1-\alpha) \cdot cq_{th} < q_{avg} < cq_{th}$
  - Random Packet Drop with RED or SQM_1

$$P_{cc} = P_{RED/SQM1} \times \frac{(q_{avg} - (1-\alpha) \times cq_{th})^2}{(\alpha \times cq_{th})^2}$$

- *if* $cq_{th} < q_{avg} < (1-\alpha) \cdot Q_{size}$
  - Random Packet Drop with SQM_2

$$P_{cc} = P_{RED/SQM1} + (1 - P_{RED/SQM1}) \times \frac{(q_{avg} - cq_{th})^2}{((1-\alpha) \times Q_{size} - cq_{th})^2}$$

* α 는 SQM 알고리즘의 혼잡제어 특성 결정

그림 10 SQM 알고리즘의 Pseudo code

중 첫 번째 구간은 0부터 $(1-\alpha) \cdot cq_{th}$까지며 이 구간에서는 패킷 폐기 없이 패킷을 처리한다. 두 번째 구간은 $(1-\alpha) \cdot cq_{th}$부터 cq_{th}까지며 이 구간에서는 초기 혼잡제어 구간으로 동작하여 평균 큐 크기가 cq_{th}에 근접할 경우 비선형 특성의 비례 함수를 이용하여 cq_{th} 정도의 혼잡상황임을 알리기 위해 낮은 확률 값으로 패킷을 폐기한다. 세 번째와 네 번째 구간은 cq_{th}부터 $(1-\alpha) \cdot Q_{size}$까지며 SQM 알고리즘의 비선형 패킷 폐기 함수를 이용한다. 초기 구간은 조심스런(Conservative) 모드로 동작하여 평균 큐 크기를 cqth 값 주변에 위치하도록 혼잡제어를 하며 혼잡상황의 정도가 심해 질 경우 과감한(Aggressive) 모드로 혼잡 상황을 제어하여 평균 큐 크기의 연속적으로 증가하는 현상을 줄이도록 동작한다. 지금까지 설명한 SQM 알고리즘의 Pseudo 코드는 그림 10과 같다.

총 4개의 구간으로 구성되어져 있는 SQM 알고리즘은 적은 매개변수를 이용하여 다양한 혼잡구간을 구분한다. 전체적인 안정된 혼잡제어를 위해 SQM 알고리즘의 각 혼잡구간에서는 각 혼잡상황에 적합한 패킷 폐기율을 결정하여 불필요한 패킷 폐기없이 혼잡제어를 수행한다. 본 논문에서 제안하는 SQM 알고리즘의 동작을 앞에서 설명한 종단간 흐름제어 구조의 상관관계로 도시하면 그림 11과 같다.

SQM 알고리즘은 동적으로 변화하는 네트워크 특성 대해서 항상 그림 11과 같은 구조로 안정적인 혼잡제어

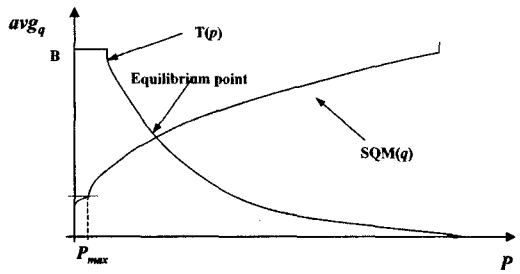


그림 11 SQM 알고리즘의 종단간 흐름제어 구조의 상관관계

를 수행한다. 이러한 SQM 알고리즘의 안정된 혼잡제어는 큐의 크기의 변화의 정도를 줄여 전체적인 패킷 폐기율(손실율)을 줄인다.

5. 시험 및 성능 평가

본 논문에서 제안한 SQM 알고리즘의 성능을 평가하기 위하여 세 가지 실험을 하였다. 첫 번째 실험은 RED 알고리즘과 SQM 알고리즘의 안정된 동작 여부를 비교하기 위해 혼잡상화 시 라우터 큐의 변화를 비교하였다. 두 번째 실험은 RED 알고리즘과 SQM 알고리즘의 각 플로우별 동작 특성을 알기 위해서 혼잡링크를 지난 각 플로우의 시간당 패킷도착시간 변화를 비교하였다. 세 번째 실험은 혼잡링크를 지나가는 플로우의 수를 변화 시켰을 때 전체 패킷 손실률을 RED 알고리즘과 SQM 알고리즘에 대하여 비교하여 보았다.

5.1 라우터 큐 변화

앞장에서 설명한 종단간 흐름제어 구조에서의 라우터의 큐 크기 변화가 발생하는 것은 매우 일반적인 현상이다. 이러한 현상은 종단간 흐름제어 구조에서 사용하는 TCP와 같은 전송 프로토콜의 혼잡제어 메커니즘 때문에 발생하며, 네트워크의 혼잡상황에 따라 매우 동적이고 가변적인 특성을 가진다. 특히, 잘못된 네트워크의 귀환 제어모듈 설정으로 인하여 혼잡상황을 제대로 제어하지 못할 경우, 라우터 큐 크기 변화의 정도가 큰 폭(width)으로 발생하며 전반적으로 네트워크의 성능을 저하시키는 결과를 가져온다. 예를 들어, 큐 크기 변화의 폭이 클 경우 큐에서는 오버플로우 현상이 발생하여 높은 패킷 손실률이 발생하며 재 전송으로 인한 네트워크 자원(resource) 낭비와 TCP 알고리즘의 불필요한 연속적인 혼잡제어와 타임아웃(time-out)으로 인한 네트워크 활용도(utilization)가 떨어지게 된다. 또한 큐 크기의 변화가 클 경우 큐잉 지연의 변화가 커져 네트워크 서비스 품질이 저하된다.

본 논문에서 제안한 SQM 알고리즘에 따라 변화하는

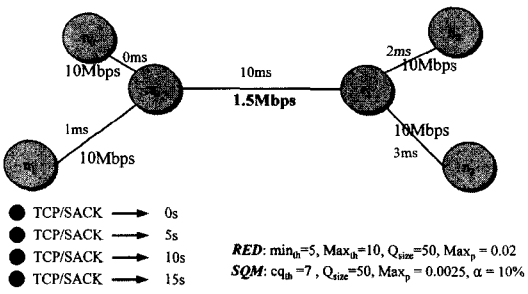


그림 12 시뮬레이션을 위한 네트워크 환경

큐 크기의 변화를 알아보기 위해서 ns2(network simulation 2)[15]을 이용하여 그림 12와 같은 네트워크 환경에서 실험하였다.

실험 네트워크에 총 4개의 플로우를 전송하여 혼잡링크인 라우터 r_0에서의 큐 크기의 변화를 모니터링 하였다. RED 알고리즘과 SQM 알고리즘 이용 시 시간당 혼잡상황이 발생되는 라우터의 큐 크기 및 평균 큐 크기를 도시하면 그림 13과 같다. 그림 13의 (a)와 같이 기존의 RED 알고리즘을 사용할 때, 현재 네트워크 트래픽의 특성에 RED 알고리즘의 매개변수가 적합하지 않을 경우 큐 크기가 심하게 변화됨을 확인할 수 있다. 본 논문에서 제안한 SQM 알고리즘을 적용하였을 경우 그림 13의 (b)와 같이 큐 크기 변화의 정도가 작으며 목표 값으로 설정한 cq_{th}=7에 근처에서 평균 큐 크기가 변화하고 있음을 확인할 수 있다. 라우터 큐에서 발생하는 불필요한 큐 크기의 변화는 응용프로그램에게 높은 지터를 가지게 하여 네트워크의 서비스를 나쁘게 할 수 있다.

5.2 플로우별 동작 특성

제안한 SQM 알고리즘의 성능을 자세히 알아보기 위해서 혼잡링크를 지난 각 플로우의 시간당 패킷 도착시간 변화를 비교 기존의 방법과 비교하여 얼마나 안정적으로 혼잡제어를 하는지 알아보았다. 그림 14와 같은 네트워크 환경에서 총 8개의 플로우가 전송되는 혼잡링크에 대하여 각 플로우의 시간당 패킷 도착시간 변화를 비교하였다.

그림 15의 (a)에서와 같이 RED 알고리즘의 경우 트래픽 특성에 맞지 않아 각 플로우의 패킷 도착 시간이 매우 불안정한 동작을 보이고 있다.

결국 RED 알고리즘의 불안정한 동작은 그림 15의 (a)와 같이 연속적인 패킷 폐기율을 보인다. 그림 15의 (b)의 SQM 알고리즘의 각 플로우의 패킷 도착 시간은

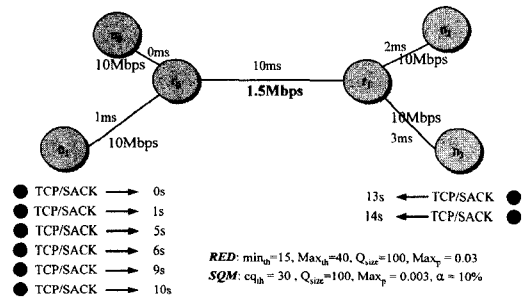
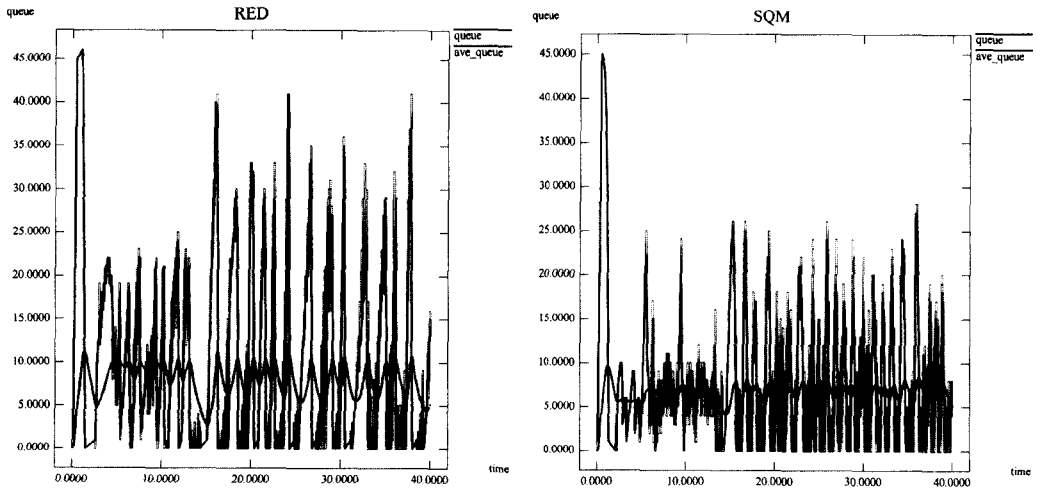
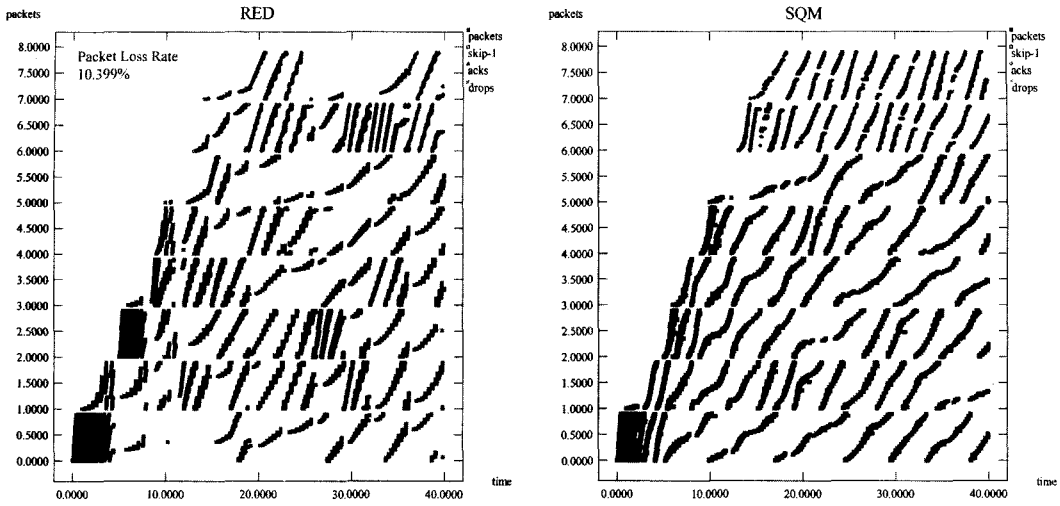


그림 14 시뮬레이션을 위한 네트워크 환경



(a) RED 알고리즘의 큐 변화 (b) SQM 알고리즘의 큐 변화

그림 13 RED 알고리즘과 SQM 알고리즘의 큐 변화 비교



(a) RED 알고리즘의 각 플로우 특성 (b) SQM 알고리즘의 각 플로우 특성

그림 15 RED 알고리즘과 SQM 알고리즘의 각 플로우 특성 비교

전체적으로 안정적인 동작을 보이고 있다. 그림 15의 (a) 경우 각 플로우들의 전송 패턴을 살펴보면 불안정한 큐 변화로 인하여 트래픽의 연속적인 패킷 폐기가 발생하였음을 알 수 있다. 그 결과 SQM 알고리즘의 실험결과인 그림 15의 (b)보다 각 플로우의 전송 패턴이 중간 중간 단절된 전송 패턴을 보임을 확인할 수 있다. 이러한 변화는 한개의 플로우에서뿐만 아니라 전체적인 플로우에 걸쳐 발생됨을 알 수 있다. 이는 SQM 알고리즘의 경우 5.1절의 실험에서와 같이 RED 알고리즘 보다 큐 크기의 변동이 작으며 안정적인 동작을 하기 때문이다. 또한 SQM 알고리즘의 구조상 매개변수에 대한 민감도가 상대적으로 감소하여 불필요한 패킷 폐기 없이 혼잡제어를 위한 적절한 패킷 폐기를 통해 안정적인 동작을 보이는 것이다. 이 실험에서 SQM 알고리즘의 경우 3.398%의 패킷 폐기율을 보였으며 RED 알고리즘의 경우 10.399%의 패킷 폐기율을 보였다. 이 실험 결과를 통해서 제안한 SQM 알고리즘은 기존의 RED 알고리즘 보다 안정적인 혼잡제어 동작을 보임을 확인할 수 있었다.

5.3 패킷 폐기율

라우터에서의 패킷 폐기율의 결정은 네트워크의 효율성을 결정한다. 필요 이상의 패킷을 폐기하거나 적절한 패킷 폐기율을 가지지 못할 경우 네트워크의 효율성은 급격히 떨어진다. 예를 들어 패킷 폐기율을 너무 높을 경우 네트워크에서 처리할 수 있는 트래픽의 양을 감소시켜 네트워크의 자원을 완전히 활용하지 못하는 경우를 발생시킨다. 그리고 패킷 폐기율이 적정 폐기율보다 너무 낮을 경우 혼잡상황을 계속 야기시켜 네트워크의 성능을 급격히 저하시킨다. 따라서 라우터에서 네트워크 상황에 맞는 패킷 폐기율을 결정하는 것은 매우 중요하다.

본 절에서는 앞장에서 설명한 잘못된 매개변수의 설정으로 인하여 발생하는 불필요한 패킷 폐기율을 가지는 라우터 알고리즘을 개선한 SQM 알고리즘의 성능을 실험하였다. 다양한 매개변수로 인해 네트워크의 트래픽 특성을 결정하는 플로우의 수의 변화에 민감한 RED알고리즘을 개선한 SQM 알고리즘의 동작을 알아보기 위해서 라우터에서 발생하는 패킷 폐기율을 계산하였다. 그림 16과 같은 네트워크 환경해서 FTP 플로우의 수를 개에서 8개, 32개 그리고 64개로 변화시키면서 각 알고리즘의 매개변수 P_{max} 변화에 대한 전체 플로우에 대한 전체 패킷 폐기율을 알아보았다.

각 큐 관리 알고리즘에서 폐기하는 패킷의 비율을 계산하면 그림 17과 같다. 그림 17과 같이 두 알고리즘 모두 플로우 수가 증가함에 따라 전체 패킷 폐기율이 증가하는 것을 알 수 있다.

그림 17에서 보는 것과 같이 본 논문에서 제안한 SQM 알고리즘을 사용할 경우 기존의 RED 알고리즘 방법보다 전반적으로 낮은 패킷 폐기율을 나타냄을 알 수 있다. 실험 결과에서와 같이 RED 알고리즘의 경우 플로우의 수에 따라 최소 패킷 폐기율을 결정하는 P_{max} 값이 서로 다른 것을 확인 할 수 있다. 즉 트래픽의 특성에 따라 RED 알고리즘의 매개변수가 P_{max} 설정되어야만 최소 패킷 폐기율 가지며 효과적으로 혼잡상황을 제어할 수 있다. 예를 들어 적절한 P_{max} 값이 설정되어 있지

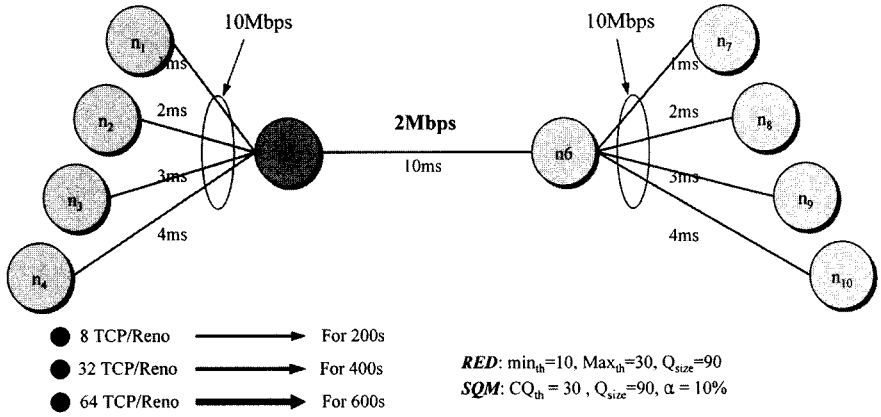
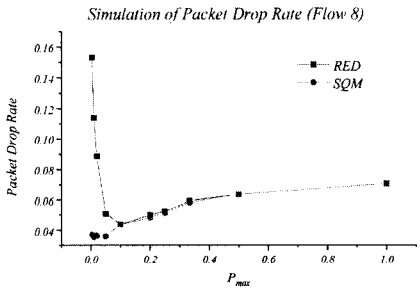
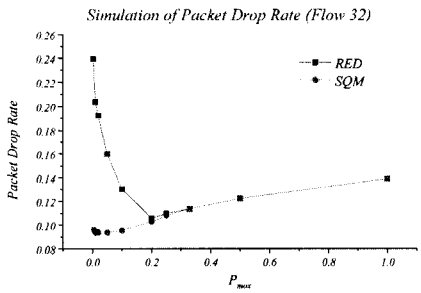


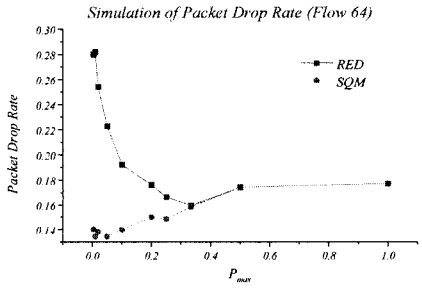
그림 16 시뮬레이션을 위한 네트워크 환경



(a) 8개 플로우에 대한 패킷 폐기율



(b) 32개 플로우에 대한 패킷 폐기율



(c) 64개 플로우에 대한 패킷 폐기율

그림 17 RED 알고리즘과 SQM 알고리즘의 패킷 폐기율

않을 경우 앞장에서 설명한 종단간 흐름제어 구조의 두 모듈의 상관 함수 $q_{ave} = T(p)$, $p = R(q_{ave})$은 평형 상태로 수렴하는 평형 상태 점$(p, qave)$을 가지지 못하게 된다.

SQM 알고리즘은 RED 알고리즘에 비하여 네트워크 상황에 대해 매개변수 민감도가 상대적으로 적으며 불필요한 패킷 폐기율이 감소됨을 확인할 수 있다. 이를 통하여 라우터에서의 오버플로우 및 불필요한 패킷 재전송을 줄여 혼잡상황에서도 안정적으로 동적인 네트워크 트래픽을 처리한다.

세 가지 실험 결과를 통하여 본 논문에서 제안한 SQM 알고리즘은 라우터 알고리즘의 특성을 결정하는 매개변수에 대한 민감도를 줄여 동적인 네트워크 특성에 대하여 안정적인 혼잡상황을 제어하는 것을 확인할 수 있다.

6. 결 론

IETF에서는 효율적인 혼잡제어를 위하여 라우터 알고리즘으로 RED 알고리즘과 같은 큐 관리 알고리즘을 권고하고 있다. 하지만 RED 알고리즘은 네트워크 환경에 따른 매개 변수의 설정의 어려움을 가지고 있어 잘못된 매개변수 설정으로 인하여 네트워크 성능을 저하시키는 문제를 발생시키며 또한, 이로 인하여 전체적으로 네트워크 상에 불안정한 혼잡제어 구조를 가지게 하는 문제가 있다.

이러한 문제를 해결하기 위해서 본 논문에서는 동적으로 변화하는 네트워크 트래픽으로 인하여 발생하는 혼잡상황을 기존의 방법 보다 안정적으로 제어하는 새로운 큐 관리 방법인 SQM 알고리즘을 제안하였다. 안정적으로 동작하는 SQM 알고리즘의 성능을 검증하기 위하여 평균 큐 크기의 변화, 각 플로우의 특성 변화,

전체 패킷 폐기률에 관한 시뮬레이션을 통해 SQM 알고리즘의 성능을 검증 및 확인하였다. 시뮬레이션 결과를 통하여 SQM 알고리즘은 기존 방법보다 간단한 매개 변수 설정으로 기존의 RED 알고리즘의 문제점을 개선하여 안정된 혼잡제어 동작을 수행하는 것을 확인하였다.

향후 연구 과제로는 비반응(unresponsive) 플로우에 대한 공정성을 개선하는 방법에 대한 연구가 수행되어야 하며, 실제 네트워크 환경에서 적용할 수 있도록 다양한 시험 및 검증에 대한 추가적인 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] W. Feng, D. Kandlur, D. Saha, and K. Shin., "Techniques for Eliminating Packet Loss in Congested TCP/IP Network," U. Michigan CSE-TR-349-97, November 1997.

[6] Demers, A., Keshav, S. and Shenker, S., "Analysis and simulation of a fair queueing algorithm," Journal of Internetworking Research and Experience, October 1990. Also in Proceedings of ACM SIGCOMM '89.

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

[8] S. Floyd, "RED: Discussions of Setting Parameters," November 1997. http://www.aciri.org/floyd/REDparameters.txt.

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

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

[11] Padhy, J., Firoiu, V., Towsley, D., and Kurose, J., "A Stchastic Model of TCP Reno Congestion Avoidance and Control," Technical Report CMPSCI TR 99-02, Univ. of Massachusetts, Amherst, 1999.

[12] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," INFOCOM 2001, 2001.

[13] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman., "An Optimization-oriented View of Random Early Detection," Computer Communications, 2001, to appear.

[14] Feng, W., et al "A self-configuring RED gateway," INFOCOM 99, April 1999.

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

구 자 헌

1999년 광운대학교 전자통신공학과 학사 2001년 광운대학교 전자통신공학과 석사 2001년~현재 광운대학교 전자통신공학과 박사 과정. 2003년~현재 InnoWireless 정보통신연구소 연구원. 관심분야는 컴퓨터 통신, 인터넷 QoS, 멀티미디어통신

정 광 수

1981년 한양대학교 전자공학과 학사 1983년 한국과학기술원 전기 및 전자공학과 석사. 1991년 미국 University of Florida 전기공학과 박사(컴퓨터공학전공). 1983~1993년 한국전자통신연구원 선임연구원 1991~1992년 한국과학기술원 대우 교수. 1993년~현재 광운대학교 전자공학부 교수 (정보통신 연구원)