폭주회피를 위한 큐 관리 기반의 패킷 탈락 알고리즘

A Packet Dropping Algorithm based on Queue Management for Congestion Avoidance

이 팔 진* 양 진 영**
Pal-Jin Lee Jin-Young Yang

요 약

본 논문은 능동적인 큐 관리를 이용한 새로운 패킷 탈락 알고리즘에 대해 연구이다. 능동적인 큐 관리 기법은 기존의 Drop Tail과 다른데, Drop Tail은 버퍼 오버플로우가 발생하면 패킷 탈락되는 반면, 능동적인 큐 관리 기법인 RED는 폭주가 발생하기 전에 패킷이 탈락된다는 것이다. 그러나 능동적인 큐 관리 기법은 버퍼 크기가 충분히 크지 않을 때 높은 패킷 손실률을 초래한다. 폭주를 탐지하고 무작위로 선택된 연결에 이를 통보함에 의한 글로벌 동기화와 공정성 문제를 야기하며, 최적의 평균 큐 길이를 찾기 위해서는 네트워크 트래픽 특성이 미리 알려져야 한다는 커다란 문제가 있다.

본 논문에서는 폭주제어를 위한 새로운 큐 관리 기법 기반의 효율적인 패킷 탈락 기법을 제안한다. 제안된 기법은 플로우별 도착률과 추정된 공정한 대역을 사용한다. 이를 이용하여 플로우 도착률과 링크 대역을 계산하기 위한 추정 알고리즘을 제안한다. 제안된 기법은 패킷 손실을 가져오는 패킷에 의한 큐 길이의 급속한 진동을 초래하지 않기 때문에 네트워크 성능을 향상시킬 수 있음을 보인다.

Abstract

In this paper, we study the new packet dropping scheme using an active queue management algorithm. Active queue management mechanisms differ from the traditional drop tail mechanism in that in a drop tail queue packets are dropped wher the buffer overflows, while in active queue management mechanisms, packets may be dropped early before congestion occurs. However, it still incurs high packet loss ratio when the buffer size is not large enough. By detecting congestion and notifying only a randomly selected fraction of connection, RED causes to the global synchronization and fairness problem. And also, it is the biggest problem that the network traffic characteristics need to be known in order to find the optimum average queue length.

We propose a new efficient packet dropping method based on the active queue management for congestion control. The proposed scheme uses the per-flow rate and fair share rate estimates. To this end, we present the estimation algorithm to compute the flow arrival rate and the link fair rate. We shows the proposed method improves the network performance because the traffic generated can not cause rapid fluctuations in queue lengths which result in packet loss

☞ Keyword : network queue management, network performance

1. 서 론

인터넷은 프로토콜을 이용한 비연결성 종단간 패킷 서비스에 기반하고 있으며, 대부분이 TCP 트래픽으로 구성되어 있을 뿐만 아니라 최선형 서비스로 이루어져 있다. 이들 서비스가 동일한 병목 링크로 다중화 되어 전송될 때 이들 사이의 공정성에 의한 서비스 품질을 유지하기 위해 종단에서 TCP 폭주 제어를 이용하고 있다. 그러나 멀티미디어 스트리밍 응용과 같은 새로운 서비스의 등장에 따라 TCP는 트래픽 전송 제어에 많은 문제점을 가지고 있다. 특히 과부하 상태 하에서 사용자에게 좋은 서비스를 제공하기 위해서는 폭주 제어 기법의 주의 깊은 설계가 요구되며, 그렇지 못할 경우 심각한 서비스 저하 또는 인터넷 붕괴를 초래한다[1,2,3].

* 정 회 원 : 초당대학교 컴퓨터과학과 부교수
pjlee@chodang.ac.kr(제1저자)
** 정 회 원 : 초당대학교 컴퓨터과학과 조교수
jyyang@chodang.ac.kr(공동저자)

폭주 제어를 위한 Random Drop, Drop Tail, Source Quench, Congestion Indication, Selective Feedback Congestion Indication 등과 같은 많은 라우터 알고리즘들이 연구되어 왔다[3,5,7,9,11,15].

Random Drop은 통계적으로 패킷을 탈락시킴으로써 폭주를 제공한 트래픽 사용자에게 피드백을 전달하는 기법이다. 폭주 기간동안에 폭주를 더 많이 초래한 연결은 더 높은 탈락 확률을 갖는다. 이 기법은 통계적 방법을 이용하여 패킷을 탈락시키는데, 도착되는 모든 연결로부터 패킷이 임의로 선택되어 탈락되며, 사용자의 평균 전송률에 비례하여 이루어지기 때문에 선택된 패킷은 모든 연결들 사이에 균일한 분포를 갖는다. 다만 이 기법은 패킷 도착이 항상 연결들 사이에 균일하게 분포되어 있다는 것을 전제로 한다. 그러나 실제 네트워크의 트래픽은 일반적으로 TCP처럼 상위 계층에서 제어되기 때문에 이와 같은 전제는 사실로 볼 수는 없다. 특히 이 기법은 폭주 회복을 위해 사용된다면 대단히 비효율적이다 라고 지적하고 있다[17].

Source Quench는 RFC-792 ICMP(Internet Control Message Protocol)의 메시지로서 패킷이 버퍼 오버플로우에 의해 탈락될 때마다 게이트웨이는 손실이 발생한 소스에 이 메시지를 전달한다. 이와 같은 동작은 패킷이 탈락될 때마다 취해지기 때문에 폭주 회복 기법으로 생각할 수 있다. 그러나 이 기법은 메시지를 보내기 위해서는 자원이 필요하기 때문에 매 탈락된 패킷마다 Source Quench 메시지를 보낸다는 것은 대단히 비효율적이며, 또한 이미 폭주가 발생한 상황에서는 더 많은 폭주를 발생시키기 때문에 상황을 더욱 더 악화시킨다[3].

Drop Tail은 모든 기법 중에서 가장 간단한 알고리즘으로서, 패킷을 선택적으로 탈락시키는 것이 아니라 이용 가능한 버퍼의 공간이 없을 때 곧 바로 탈락시킨다. Drop Tail과 Random Drop 게이트웨이의 단점은 글로벌 동기화를 들 수 있는데, 많은 연결로부터 패킷이 탈락됨에 따라 윈도우 크기가 줄어들며, 이는 처리율 감소로 이어진다[4].

본 논문에서는 네트워크 성능을 극대화시키기 위해 병목이 있는 라우터에서 각 연결들에게 자원을 효율적으로 할당하며, 기존의 버퍼 관리 기법들이 가지고 있는 문제점들을 해결하기 위해 큐 관리를 기반으로 한 새로운 패킷 탈락 기법을 제안한다. 제안된 기법은 기존의 능동적인 큐 관리 기법에서 목표로 설정하고 있는 큐잉 지연을 만족시키기 위해 평균 큐 크기를 가능한 적게 유지시킨다. 이는 큐에서의 패킷 손실을 줄일 뿐만 아니라 각 플로우에 대한 공정성을 확보할 수 있기 때문에 네트워크의 처리율을 향상시킬 수 있다.

2. 능동적인 큐 관리 알고리즘

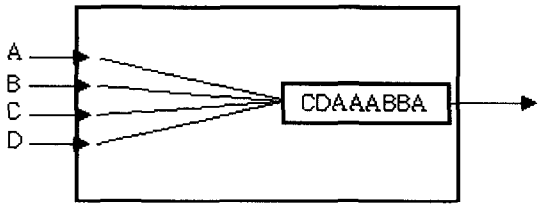
버퍼링은 패킷이 스위치를 빠져나가는 것보다 도착하는 속도가 더 클 때 발생하는 패킷 손실을 줄이기 위해 필요한 기능이다. 버퍼 관리는 버퍼를 초과하여 발생하는 폭주를 제어하며, 패킷의 손실을 줄이는 것으로 공유 버퍼 풀과 플로우별 할당의 두 가지 기법이 있다.

공유 버퍼 풀은 그림 1과 같이 모든 플로우가 하나의 버퍼를 공유하는 것이며, FIFO(First-In First-Out)로 큐잉되어 처리된다.

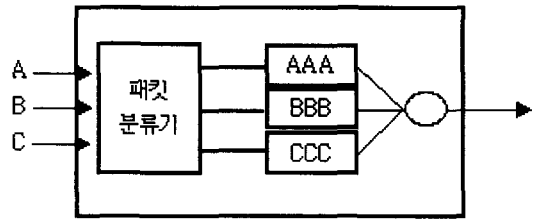
공유 버퍼 풀은 구현이 간단하기 때문에 인터넷에서 많이 사용하고 있지만 플로우가 다른 플로우로부터 보호받지 못하는 공정성의 문제가 있다[3]. 즉, 하나의 플로우가 모든 버퍼를 차지할 수 있기 때문에 다른 플로우는 서비스를 받을 수 없다는 것이다.

플로우별 기반으로 버퍼를 할당하는 기법은 그림 2와 같은 구조를 갖는다. 이 기법은 각 플로우별 버퍼 이용률을 높일 수 있을 뿐만 아니라 개별 플로우별 할당된 버퍼의 점유 수준을 이용하여 패킷을 탈락시키며, 또한 버퍼의 사용량에 의하여 non-TCP 응용에 대한 플로우로부터 TCP 응용에 대한 플로우를 보호할 수 있다[12].

본 절에서는 이와 같은 버퍼 관리 기법 중 공유



(그림 1) FIFO 큐잉



(그림 2) 공정한 큐잉

버퍼 풀을 중심으로 현재 인터넷에서 가장 많이 사용되고 있는 PQM(Passive Queue Management)과 AQM(Active Queue Management)을 살펴본다.

2.1 PQM 알고리즘

PQM으로서의 라우터 버퍼 관리 기법은 1장에서 언급한 바와 같이 채워진 버퍼에 도착되는 패킷을 탈락시키는 Drop Tail 알고리즘이다[1,3]. 큐 길이를 관리하기 위한 기법은 임계치로서 큐의 최대 길이를 설정하여 패킷을 수신한다. 그리고 도착된 패킷이 임계치를 벗어나면 큐가 감소할 때까지 이후에 들어오는 모든 패킷을 탈락시킨다. 이 기법은 또한 Tail Drop이라 하는데, 큐가 채워질 때 가장 최근에 도착한 패킷은 탈락되기 때문이다.

2.2 AQM 알고리즘

Floyd는 AQM인 RED 라우터 알고리즘이라는 새로운 방법을 제안하였다[5,6,9]. RED는 Drop Tail 기법의 단점을 해결하기 위해 설계되었으며, 또한 폭주 회피 기법을 제공한다. 이 알고리즘의 운영은 FIFO 라우터의 기본 성질을 유지하며, TCP 트래픽의 폭주를 제어하기 위해 네트워크 부하의 지시자로 식 1과 같은 평균 큐 길이를 사용한다. 이 때 폭주를 제어하기 위해 개별 플로우에 대한 어떠한 상태정보도 이용하지 않기 때문에 구현이 간단하며, 또한 이 기법은 탈락에 의한 묵시적인 피드백을 제공한다.

$$Q_{avg} = (1 - W_q)Q_{avg} + W_q q, 0 \leq W_{q \leq 1} \qquad \text{(식 1)}$$

여기서

Q_{avg} : Average queue size
W_q : Queue weight
q : Queue size

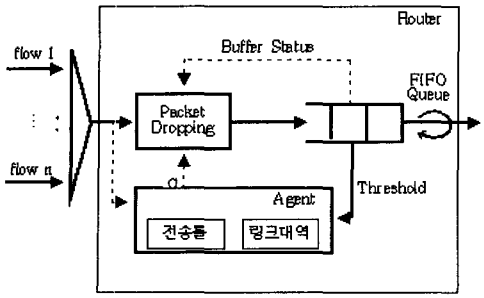
RED는 인터넷 성능을 제공하는 라우터에 대한 능동적인 큐 관리 알고리즘으로 버퍼가 채워질 때만 패킷을 탈락시키는 Drop Tail과 같은 PQM과는 달리 도착하는 패킷을 통계적으로 탈락시킨다.

3. 효율적인 패킷 탈락 알고리즘

본 장에서는 능동적인 큐 관리 기법이 가지고 있는 다양한 트래픽에 대한 글로벌 동기화나 공정성과 같은 문제점을 해결하기 위해 공정한 큐잉 기반의 폭주 제어를 위한 새로운 패킷 탈락 알고리즘을 제안한다. 기존의 기법이 버퍼의 상태만을 이용하여 폭주를 제어하는 방식과는 달리 이 기법은 각 플로우의 추정된 전송률과 대역을 이용하여 트래픽을 제어하기 때문에 폭주 제어의 목적인 지연을 줄이고 처리율을 향상시킬 수 있다.

3.1 시스템 구조

그림 3은 도착되는 패킷을 선별적으로 처리하여 기존의 능동적인 큐 관리 기법이 가지고 있는 록아웃, 풀 큐, 글로벌 동기화, 공정성과 같은 여러 가지 문제점을 해결하기 위해 제안된 큐 관리 기반의 새로운 패킷 탈락 시스템의 구조를 나타낸다. 이 구조는 패킷 드롭핑과 FIFO 큐, 그리고 추정

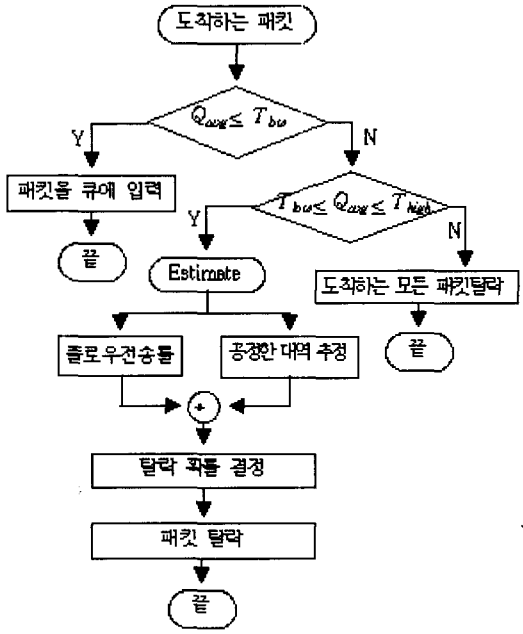


(그림 3) 제안된 큐 관리 시스템

에이전트의 3 가지로 구성된다. 버퍼의 임계치를 이용하여 수행될 패킷 드롭핑은 각 플로우로부터 도착되는 추정된 도착률과 FIFO 큐의 상태를 이용하여 탈락시킬 패킷에 대한 통계적 확률값을 계산하고 이를 이용하여 패킷을 탈락시킨다. FIFO 큐잉은 버퍼에 있는 패킷을 전송하기 위한 스케줄링 기법이며, 라우터에서의 복잡도를 피하기 위해 플로우별 큐잉이 아닌 모든 연결이 공유하는 큐잉 기법이다. 추정 에이전트는 RED 알고리즘에서 운영하고 있는 두 개의 임계치를 기반으로 적용되는 모듈이다. 버퍼에 도착하는 패킷이 최소임계치와 최대임계치 사이에 존재하면 네트워크로 하여금 추정 에이전트를 작동하게 한다. 추정 에이전트가 수행하는 두 가지의 역할은 플로우로부터 도착되는 패킷의 도착률과 그 플로우에 주어지는 공정한 추정 대역을 계산하는 것이다. 이를 이용하여 탈락시킬 패킷이 결정되면 패킷 드롭핑 모듈에 전달한다.

3.2 패킷 탈락 제어

패킷 탈락 제어는 그림 4와 같이 운영된다. 초기에 패킷이 도착하면 먼저 능동적인 큐 관리 기법인 RED 알고리즘에 따라 운영된다. 패킷이 큐에 도착하면 식 1을 이용하여 평균 큐 크기를 계산하고, 이를 최소임계치인 T_{low}와 비교한다. T_{low}보다 작으면 패킷은 큐에 입력되며, 최대임계치인 T_{high}을 초과하면 모든 패킷은 확률값 1로 탈락된다. 그리고 평균 큐 크기가 최소임계치인 T_{low}와



(그림 4) 제안된 패킷 탈락 알고리즘

최대임계치인 T_{high}사이에 존재하면 플로우별 패킷 전송률 $A_i(t)$과 그 플로우에 할당될 링크 대역 $\alpha(t)$을 계산한다. Min-max 대역 할당에 따라 플로우 i는 $\min(A_i(t), \alpha(t))$로 주어진 전송률을 수신한다. 만일 $A_i(t) \le \alpha(t)$라면 플로우 i는 공정한 전송률을 보장받을 수 있으며, 만일 $A_i(t) > \alpha(t)$라면 비트의 $\frac{A_{i(t)} - \alpha(t)}{A_{i(t)}}$ 만큼이 드롭될 것이기 때문에 정확히 $\alpha(t)$만큼의 출력전송률을 갖는다. 따라서 플로우 i로부터 도착되는 패킷은 식 2를 이용하여 탈락된다.

$$\max\left(0, 1 - \frac{\alpha(t)}{\gamma_i(t)}\right) \qquad \text{(식 2)}$$

(1) 플로우의 패킷 도착률

제안된 시스템의 구조에서 패킷의 도착률 $A_i(t)$는 라우터에서 추정되며, 이를 위해 지수 평균을 사용한다[10,13,14]. 여기서 t_i^k와 l_i^k를 플로우 i의 k번째 패킷에 대한 도착시간과 길이라 한다면 플로우 i에 대한 전송률 A_i는 식 3과 같이 나타낼 수 있

다. 여기서 A_i는 패킷이 도착할 때마다 갱신된다.

$$A_i^{new} = (1 - e^{T_i^k}/K)\frac{l_i^k}{T_i^k} + e^{-T_i^k}/ \cdot A_i^{old} \quad \text{(식 3)}$$

여기서,

$T_i^k = t_i^k - t_i^{k-1}$: Packet Arrival Time

K : Constant

(2) 공정한 대역 추정

공정한 대역 $\alpha(t)$의 추정은 다음과 같다. 라우터에서 알고리즘이 패킷을 수신하여 이를 전송하는 전송률은 공정한 링크 대역의 추정치에 대한 함수로 나타낼 수 있기 때문에 이를 $\alpha'(t)$로 표현할 수 있다. $F(\alpha'(t))$를 수신률이라 할 때 식 4와 같이 나타낼 수 있다.

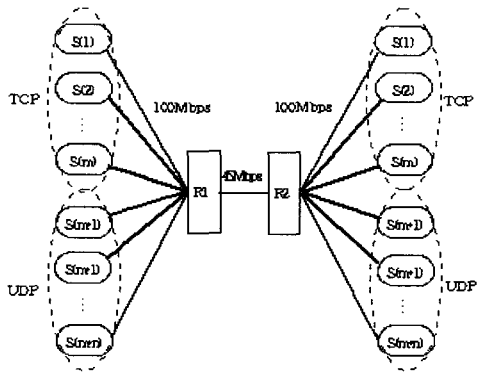
$$F(\alpha'(t)) = \sum_{i=0}^{n} \min(A_{i(t)}, \alpha'(t)) \quad \text{(식 4)}$$

링크에 폭주가 발생하면 $\alpha'(t)$을 유일한 해결책으로 선택한다. 링크에 폭주가 발생하지 않으면 링크를 경유하는 플로우 중에서 가장 커다란 전송률을 갖는 $\alpha'(t)$을 취하는데 식 5와 같이 표현할 수 있다.

$$\alpha'_{new}(t) = \max \le i \le n(A_{i(t)}) \quad \text{(식 5)}$$

식 5로부터 도착률 $A_i(t)$을 파악할 수 있다면 $\alpha'(t)$는 이로부터 직접 계산할 수 있다. α'에 대한 갱신은 링크의 상태가 폭주인지 아닌지에 따라 결정된다. 지수 평균에 의한 부정확한 추정을 피하기 위해 윈도우 크기 K_c 을 사용한다. 만일 길이 K_c 의 간격동안 항상 $A<C$ 라면 폭주가 발생한 것이 아니기 때문에 식 5를 이용하며, $A \ge C$ 이라면 링크에는 폭주가 발생한 것이기 때문에 식 6을 이용하여 공정한 링크 대역을 추정한다. 여기서 C 는 라우터에서의 병목링크의 용량을 나타내며, A 는 패킷의 총 도착률을 나타낸다. 또한 공정한 링크 대역을 추정하기 위해 식 3의 K 대신 K_α를 사용한다.

$$\alpha'_{new}(t) = \alpha_{old}\frac{C}{A^{new}} \quad \text{(식 6)}$$



(그림 5) 시뮬레이션 네트워크 구조

4. 시뮬레이션

4.1 네트워크 구조

제안된 기법의 시뮬레이션을 수행하기 위한 네트워크는 그림 5와 같이 표준 네트워크를 구성하였다[13]. 시뮬레이션 환경은 ULTRA2의 Solaris OS 환경에서 수행되었으며, 사용된 시뮬레이션 도구는 ns[19]와 Tcl/Tk[18]를 이용하였다.

시뮬레이션 네트워크에서 폭주가 발생한 병목 링크는 R1과 R2사이이다. 링크는 45Mbps의 용량을 가지며, mTCP와 nUDP 플로우에 의해 공유된다. 종단 호스트는 병목 링크 대역의 2.2배인 100Mbps의 링크를 사용하여 라우터에 연결된다. 모든 링크는 1㎳의 전송 지연을 가지며, 패킷이 겪는 지연은 전송 지연이라기보다는 주로 버퍼 지연으로 생긴 것으로 가정한다. TCP 플로우는 커다란 크기의 파일을 전송하는 FTP 세션으로 구성하였다. UDP 호스트는 r Kbps의 CBR로 패킷을 전송하며, 이때 r 은 가변이다. 모든 패킷 크기는 1K바이트로 설정한다.

라우터가 서로 다른 큐 관리 기법을 사용할 때

하나의 비적응형의 UDP 소스가 얼마나 많은 대역을 사용할 수 있는가를 알아보기 위해 그림 5와 같이 시뮬레이션 소스를 설정하였다. 네트워크에는 8개의 TCP 소스(플로우 1~플로우 8)와 8개의 UDP 소스(플로우 9~플로우 16)가 있다. UDP 소스는 CBR 크기를 2Mbps로 설정하여 전송하였다.

4.2 시뮬레이션 파라메터

시뮬레이션을 위해 사용된 파라메터는 다음과 같다. 각 출력 링크는 45Mbps의 용량과 1㎳의 지연, 18 패킷으로 구성된 버퍼를 갖는다. 제안된 알고리즘에서 최소임계치인 T_{low}는 5 패킷이며, 최대임계치인 T_{high}는 15 패킷으로 설정하였다. 평균 상수인 플로우의 전송률을 추정하는데 사용하는 K, 공정한 링크 대역을 추정하는데 사용하는 K_α, 링크가 폭주인지 아닌지를 결정하기 위해 사용하는 K_c의 각각의 값은 모두 137ms로 설정하였다. 이들 상수는 일반적으로 최대 큐잉 지연(예를 들어, 18 packets/100Mbps=137㎳)으로 선택하였다. 이들 파라메터 값들을 요약하면 표 1과 같다. 그리고 패킷의 크기는 1000바이트로 하여 50초 동안 수행하였다.

4.3 결과 및 성능 분석

본 연구에서 제안된 기법의 타당성을 보이기 위해 벤치마크로서 기존의 능동적인 큐 관리 기법인 Drop Tail과 RED를 이용하였다. 또한 소스와 Sink 노드는 4.3-Tahoe BSD TCP와 동일한 폭주

(표 1) 시뮬레이션 파라메터 값

Parameter	Value	비고
T_{low}	5 패킷	
T_{high}	15 패킷	$3\times T_{min}$
p_{max}	1/50	
w_q	0.002	
K	137㎳	최대큐잉지연
K_α	137㎳	최대큐잉지연
K_c	137㎳	최대큐잉지연

(표 2) 각 트래픽별 패킷 수

버퍼크기 / 트래픽	8	10	12	14	16	18	20	22
TCP	4.5	5.2	6.9	8.5	9.0	11.5	12.5	12.5
non-TCP	6.0	6.8	8.5	10.0	10.5	13.0	13.0	13.0

(표 3) 각 트래픽별 링크 이용률

버퍼크기 / 트래픽	8	10	12	14	16	18	20	22
TCP	0.915	0.930	0.940	0.960	0.970	0.975	0.990	0.990
non-TCP	0.918	0.926	0.939	0.960	0.975	0.980	0.990	0.990

제어 알고리즘을 이용하였다. 종단간 폭주 제어 알고리즘으로는 4.3-Tahoe BSD Release를 이용하였다[8,16]. TCP의 윈도우 조정 알고리즘은 두 단계로 이루어진다. 초기임계치는 수신측이 광고한 윈도우 크기의 절반으로 설정된다. 연결은 Slow Start 단계에서 시작하며, 윈도우가 임계치에 도달할 때까지 현재의 윈도우는 매 RTT마다 2배로 증가한다. 다음에 Congestion Avoidance 단계에 들어서면 현재의 윈도우는 매 RTT마다 대략 1 패킷 단위로 증가하며, 윈도우는 결코 수신자가 광고한 윈도우 이상으로 증가할 수 없다.

4.3-Tahoe BSD TCP에서 패킷 손실은 폭주가 발생했다는 신호로 취급된다. 소스 노드는 패킷 손실을 탐지하기 위해 Fast Retransmission 알고리즘을 사용하며, 똑같은 패킷을 확인하는 3개의 패킷이 수신되면 패킷은 탈락된 것으로 판단한다. 이때 소스는 임계치를 현재의 윈도우의 절반으로 설정하여 패킷 손실에 반응하며, 현재의 윈도우를 1로 설정하여 Slow Start 단계로 진입한다. 소스 노드는 또한 잃어버린 패킷을 탐지하기 위해 재전송 타이머를 사용한다.

시뮬레이션은 TCP 트래픽과 non-TCP 트래픽으로 분류하여 수행하였다. 전체 32개의 노드 중 16개의 노드에 트래픽을 발생시켜 패킷을 전송하였고, 이 중 8개는 TCP 플로우를, 나머지 8개에는 non-TCP 플로우를 발생시켰다. 평가를 위해 사용된

각 알고리즘의 비교 인자로 평균 큐 크기, 평균 링크 이용률, 각 플로우에 할당된 대역을 사용하였다. 표 2와 표 3은 Drop Tail 알고리즘을 적용하여 버퍼 크기에 따른 각 트래픽별 패킷 수와 링크 이용률을 나타내고 있는데, 버퍼 크기가 20 이후부터는 평균 큐의 변화가 거의 없는 것으로 나타났다. 즉, 현재의 조건에서 만족할만한 버퍼 크기는 20임을 단적으로 확인할 수 있다. 버퍼 크기가 20 이후부터는 큐의 변화가 없는 것으로 볼 때 링크 이용률도 20 이후부터 변화가 없음을 알 수 있다.

능동적인 큐 관리 기법의 목적은 평균 큐 크기를 낮게 유지하여 지연을 줄이고, 처리율을 높이는데 있다. 특히 Drop Tail은 버퍼 크기를 임계치로 사용하기 때문에 RED와 제안된 기법과는 비교를 할 수 없지만 RED와 제안된 기법 중에서 제안된 기법이 평균 패킷 수가 더 적게 나타남을 알 수 있다. RED 알고리즘은 IETF의 표준으로 지정되어 있어 본 논문에서 제안한 기법과 비교했을 때 좋은 척도라고 할 수 있다. 표 4, 표 5는 두 가지 유형의 트래픽에 대한 평균 큐 크기와 링크이용률의 비교 결과를 나타내고 있으며, 표 6, 표 7은 이들에 대한 평균치를 나타낸 결과이다.

그러나 Drop Tail 기법과 비교했을 때 RED는 상대적으로 평균 큐 크기가 크게 나타남을 볼 수 있는데, 이는 non-TCP 트래픽을 적용했던 결과로 해석된다. 또한 링크 이용률도 제안된 기법이 더 높게 나타나는데 다만 최소임계치가 14일 때 RED와 제안된 기법과는 커다란 차이를 발견할 수 없었다. 이와 같이 나타난 이유는 제안된 기법이 RED에서 사용된 두 임계치 사이에서 적용된 결과로 풀이된다. 각 플로우별 할당된 대역도 Drop Tail과 RED는 차이가 없지만 제안된 기법과는 커다란 차이를 발견할 수 있다.

(표 4) 각 트래픽별 평균 큐 크기

큐관리기법 \ 임계치 / 트래픽		4	6	8	10	12	14
RED	TCP	6.2	7.3	7.7	8.6	10.3	12.5
	non-TCP	6.5	7.5	8.2	10.2	10.5	13.0
제안기법	TCP	6.0	7.1	7.5	8.1	8.7	10.2
	non-TCP	6.3	7.2	7.7	8.2	8.9	10.5

(표 5) 각 트래픽별 링크 이용률

큐관리기법 \ 임계치 / 트래픽		4	6	8	10	12	14
RED	TCP	0.965	0.968	0.974	0.978	0.979	0.990
	non-TCP	0.967	0.972	0.978	0.982	0.985	0.990
제안기법	TCP	0.972	0.975	0.985	0.984	0.986	0.990
	non-TCP	0.973	0.975	0.983	0.985	0.987	0.990

(표 6) 2가지 알고리즘에 대한 평균 큐 크기

최소임계치 / 큐관리기법	4	6	8	10	12	14
RED	6.35	7.4.0	7.95	9.40	10.40	12.75
제안기법	6.15	7.15	7.60	8.12	8.80	10.35

(표 7) 2가지 알고리즘에 대한 평균 링크 이용률

최소임계치 / 큐관리기법	4	6	8	10	12	14
RED	0.966	0.970	0.976	0.980	0.982	0.990
제안기법	0.9725	0.975	0.984	0.985	0.987	0.990

5. 결 론

본 논문에서는 IP 네트워크에서의 폭주 제어를 효율적으로 수행하여 지연을 줄이고, 네트워크 성능을 높이기 위해 제안된 기법에 대한 연구를 수행하였다. 이는 기존의 버퍼 관리 기법인 PQM과 AQM이 가지고 있는 록-아웃, 풀 큐, 글로벌 동기화, 공정성과 같은 여러 가지 문제점을 해결하기 위한 것이다. 제안된 기법은 기존의 AQM인 RED가 평균 큐 크기를 낮게 유지함으로써 패킷 손실을 줄이는 것처럼 평균 큐 크기를 낮게 유지하여 패킷 손실을 줄이고, 동시에 각 플로우에 공정하게 대역을 할당하여 네트워크의 성능을 향상시킬 수 있었다.

인터넷은 시간이 지날수록 네트워크 성향이 강한 응용이 계속 증가하고 있다. 이에 대비하여 많은

문헌들에서는 인터넷 트래픽을 적응형 TCP 플로우, 그리고 집단 플로우로서 비적응형의 플로우, TCP와 호환되지 않는 전송 프로토콜로 구분하고 있다. TCP/IP 헤더도 이를 위해 8bit ToS(Type Of Service)를 설정하고 있다. 따라서 향후 본 연구는 이와 같은 다양한 트래픽을 고려한 시뮬레이션을 지속적으로 수행할 것이며, 이들 플로우에 적합한 버퍼 관리 기법의 탈락 수준 및 탈락 확률 분포 또한 보완할 예정이다.

참 고 문 헌

[1] Mark Gaynor, "Proactive Packet Dropping Methods for TCP Gateways," http://www.eecs.harvard.edu/~gaynor/final.ps.

[2] Darius Buntinas, "Congestion Control Schemes for TCP/IP Networks," http://www.cis.ohio-state.edu/~jain/cis788-95/tcpip_cong.

[3] A. Mankin, MITRE, K.Ramakrishnan, "Gateway Congestion Control Survey," Network Working Group, RFC:1254, http://www.uxsup.csx.cam.ac.uk/netdoc/rfc/rfc1254.txt.

[4] Raj Jain, "Congestion Control in Computer Networks : Issues and Trends," IEEE Network Magazine, pp.24~30, May 1990.

[5] S. Floyd, et al, "Recommen dations on Queue Management and Congestion Avoidance in the Internet," Network Working Group, RFC2309, April. 1998.

[6] Sally Floyd and Kevin Fall, "Router Mechanism to Support End-to-End Congestion Control," LBL Technical Report, 1997.

[7] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, Vol.1, No.4, pp.397~413, Aug. 1993.

[8] Wu-chang Feng, Dilip D.Kandlur, Debanjan Saha, Kang G. Shin, "A Self-Configuring RED Gateway," Proceedings of INFOCOM '99, March 1999.

[9] Dong Lin and Robert Morris, "Dynamics of Random Early Detection," ACM SIGCOMM, Vol.27, No.4, Oct. 1997.

[10] Wu-chang Feng, "Improving Internet Congestion Control and Queue Management Algorithms," Dept. of Computer Science and Engineering, 1999.

[11] Victor Firoiu and Marty Borden, "A Study of Active Management for Congestion Control," http://www-net.cs.umas.edu/~vfiroiu/papers/red-dynamics-conf.pdf.

[12] Bernhard Suter, T.V.Lakeshman, Dimitrios Stiliadis, Abhijit Choudhury, "Design Considerations for Supporting TCP with Per-flow Queuing," Proceedings of IEEE INFOCOM'98, April 1998.

[13] Ion Stoica, Scott Shenker, Hui Zhang, "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," CMU-CS-98-136, http://report.archive/adm.cs.cmu.edu/amon/1998/CMU-98-136.pdf.

[14] Ion Stoica, Scott Shenker, Hui Zhang, "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," Proceedings ACM Sigcomm '98, pp.118~130, Sep. 1998.

[15] A. Mankin, MITRE, K. Ramakrishnan, "Gateway Congestion Control Survey," Network Working Group RFC: 1254, Aug. 1991, http://www-uxsup.csx.cam.ac.uk/ netdoc/rfc/rfc1254.txt.

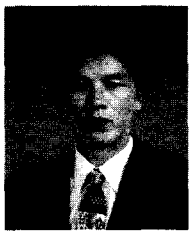
[16] Go Hasegawa and Masayuki Murata, "Survey on Fairness Issues in TCP Congestion Control Mechanisms," to appear in IEICE Transactions on Communication, 2001, http://www-ana.nal.ics.es.osaka-u.ac.jp/~hasegawa/english/papers.html.

[17] A. Mankin, "Random Drop Congestion Control," in Proceedings of ACM SIGCOMM'90, pp.1~7. Sep. 1990.

[18] Brent B. Welch, Practical Programming in Tcl

and Tk, Prentice Hall PTR, 1997.
[19] Kevin Fall, Kannan Varadhan, The VINT project : ns Notes and Documentation, Feb. 25, 2000.

◐ 저 자 소 개 ◑



이 팔 진

1986년 조선대학교 전산기공학과(학사)
1988년 중앙대학교 대학원 전자계산학과(이학석사)
1995년 전북대학교 대학원 전자계산기공학과(공학박사)
1995년~현재 : 초당대학교 컴퓨터과학과 부교수
관심분야 : 컴퓨터네트워크, 멀티미디어 등
E-mail : pjlee@chodang.ac.kr

양 진 영

1983년 조선대학교 경영학과(학사)
1988년 조선대학교 대학원 전자계산전공(공학석사)
2002년 목포대학교 대학원 컴퓨터공학과(공학박사)
1997년~현재 : 초당대학교 컴퓨터과학과 조교수
관심분야 : TCP/IP, HSI, 데이터베이스 등
E-mail : jyyang@chodang.ac.kr