가상 버퍼를 이용한 공평성을 지원하는 확장된 FRED 기법

우 희 경[†] · 김 종 덕[††]

요 약

본 논문은 최선형 트래픽을 전송하는 미래의 프로토콜 설계에 종단점간 과밀 제어를 포함하기 위한 라우터의 큐 관리 알고리즘인 확장된 FRED(Ex-FRED)를 제안하였다. 본 논문에서는 TCP로 제어되는 트래픽을 robust와 fragile 트래픽으로 나누고 다양한 응용에 따라 발생하는 robust 트래픽과 fragile 트래픽의 불공평성 문제를 다루었다. 예를 들어, bursty 응용에서 발생하는 fragile 트래픽은 대응 속도가 느리므로 자원을 공평하게 사용할 수 없는 경우가 있다. 제안된 Ex-FRED는 현재 버퍼가 일부의 모습만을 반영하므로 잘못된 정보를 줄 수 있는 문제점을 갖고 있는 FRED를 개선하였다. Ex-FRED는 유실을 결정하는 점유율을 계산할 때 실제 버퍼보다 큰 가상 버퍼를 사용하였다. 성능 평가 결과에 따르면 Ex-FRED 방식이 공평하게 자원을 사용하였으며 우수한 성능을 보였다.

Extended FRED(Fair Random Early Detection) Method with Virtual Buffer

Hee-Kyoung Woo[†] · Jong-Deok Kim[††]

To promote the inclusion of end-to-end congestion control in the design of future protocols using best-effort traffic, we propose a router mechanism, Extended FRED(Ex-FRED). In this paper, we categorize the TCP-controlled traffics into robust and fragile traffic and discuss several unfairness conditions between them caused by the diverse applications. For example, fragile traffic from bursty application cannot use its fair share due to their slow adaptation. Ex-FRED modifies the FRED(Fair Random Early Drop), which can show wrong information due to the narrow view of actual buffer. Therefore, Ex-FRED uses per-flow accounting in larger virtual buffer to impose an each flow a loss rate that depends on the virtual buffer use of a flow. The simulation results show that Ex-FRED uses fair share and has good throughput.

1. 서 론

인터넷은 비연결형 종단점 간 패킷 서비스에 기반하고 있어서 유연하고 견고하나 많은 부하가 네트워크에 걸렸을 때 동적인 패킷 전달을 제어할 수 없음으로 인하여 성능이 급격히 떨어지는 현상이 발생한다. 이러한 문제점을 해결하기 위하여 과밀 현상이 발생했을 때 패킷을 유실시키고 TCP 호스트에게 이를 알려 윈도우 크기를 줄이도록 하는 반응성(responsiveness)을 부여하였다[1-3]. 그러나 인터넷의 확장에 따라, 종단점에서만 수행하는 TCP 과밀 회피 방식만으로는 충분치 않으므로 라우터와 같은 네트워크의 중간 노드에서의 제어의 필요성이 대두되었다. 이를 위하여 라우터에서 큐를 감시하여 이를 관리하는 방식들인 DropTail, RED, FRED 등이 제안되었다[4-7].

† 정 회 원 : 순천향대학교 컴퓨터학부 교수
†† 준 회 원 : 서울대학교 대학원 전산과학과
논문접수 : 1999년 9월 20일, 심사완료 : 1999년 11월 6일

TCP로 제어되는 트래픽은 robust 트래픽과 fragile 트래픽으로 분류할 수 있다[8, 9]. robust 트래픽은 항상 전송할 데이터가 있는 트래픽을 의미하며 fragile 트래픽은 그렇지 않은 트래픽을 의미한다. robust 트래픽은 TCP 과밀 회피 방식에 따른 동적 자원 재분배 상황에서 빠른 속도로 대응하여 공평하게 자신의 자원을 확보하며 fragile 트래픽은 상대적으로 느리게 대응하여 자신의 자원을 확보하지 못한다. 대응의 속도 차가 나게 되는 원인으로 TCP 연결의 속성 차이를 들 수 있는데 RTT 길이의 차이가 대표적이다. 그런데 TCP 연결의 속성이 동일하더라도 상위 응용 계층의 트래픽 발생 형태의 차이에 따라 대응 속도에 차이가 발생할 수 있다. 예를 들어 여분 대역폭이 생겼을 때 대용량의 파일을 전송하는 FTP 응용은 트래픽을 지속적으로 발생시켜 빠르게 자신의 윈도우 사이즈를 늘이지만 TCP 응용 중 대부분을 차지하는[11] Web, Telnet과 같은 상호 동작적 응용들의 경우 전송할 데이터가 없는 경우 자신의 윈도우 사이즈를 늘이지 못할 수 있다. 이는 TCP의 과밀 제어 방안이 Closed Loop, Implicit Congestion Detection 방법을 사용하기 때문이다.

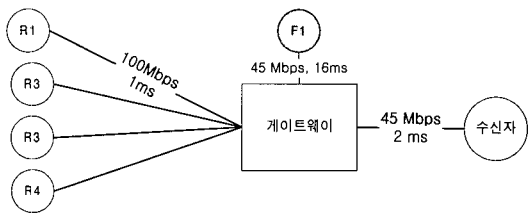
Fragile 트래픽과 robust 트래픽에 대역폭을 공평하게 사용하기 위한 큐 관리 방법으로 FRED(Fair Random Early Detection)[1]가 제안되었다[8]. FRED는 현재 데이터를 전송하는 흐름 각각에 대하여 각 흐름이 버퍼를 얼마나 차지하는지에 따라 유실율을 다르게 하였다. 그러나 FRED 방식은 RTT 길이의 차이에 따른 robust 트래픽과 fragile 트래픽 간의 공평성 문제만을 다루었으며 상위 계층의 트래픽 발생 형태에 따른 공평성 문제를 다루지 않았다. 상위 계층의 트래픽 발생 형태가 다른 경우 현재 큐 상태가 트래픽의 특성을 제대로 반영하지 못할 수 있으므로 문제가 생길 수 있다.

본 논문에서는 현재 인터넷에 널리 사용되는 웹 서비스의 특성인 버스트한 특성을 갖는 연결과 그렇지 않은 경우의 연결에 있어서 자원을 공평하게 공유할 수 있는 큐 제어 방법인 확장된 FRED(extended FRED) 방식을 제안하고 그 성능을 살펴보았다. 확장된 FRED 기법은 현재의 큐 상태만으로 각 흐름의 점유율을 계산하는 FRED 방식과는 달리 실제 버퍼의 크기보다 큰 가상 버퍼를 이용하여 가상 버퍼에서의 각 흐름의 점유율을 이용하여 자원을 공평하게 이용하게 한다.

1) 저자의 또 다른 논문에서는 FRED를 Flow Random Early Drop의 줄임말이라고 밝히고 있다.

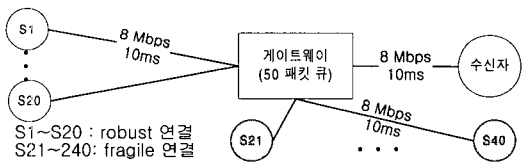
본 논문은 2장에서는 TCP로 제어되는 경우의 공평성 문제를 살펴보았으며 3장에서는 제안하는 알고리즘인 확장된 FRED 알고리즘을 기술하였다. 4장에서는 확장된 FRED 알고리즘의 성능을 시뮬레이션을 이용하여 살펴보았다. 5장에서는 결론을 기술하였다.

2. 공평성 문제



(그림 1) FRED 방법의 시스템 구성도

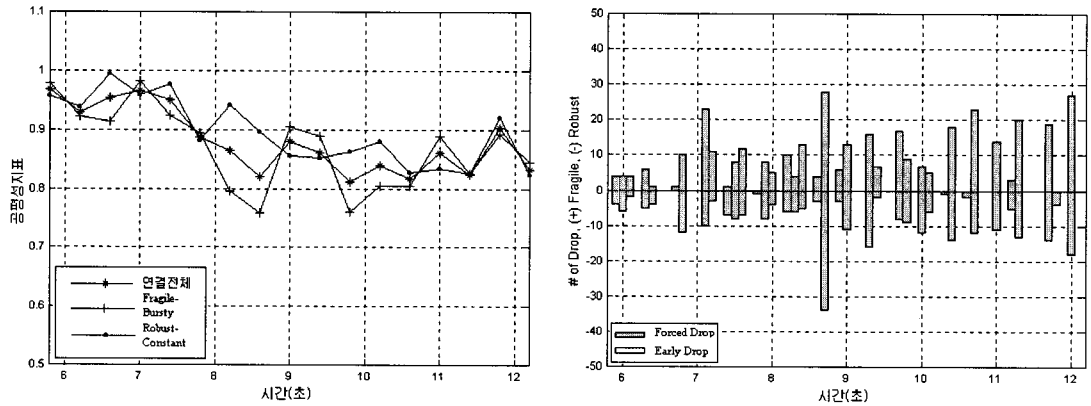
Lin과 Morris는 기존의 RED 방법을 사용하는 경우, fragile 연결이 robust 연결에 비하여 공평하게 자원을 사용하지 못 함을 보이기 위하여 (그림 1)과 같은 구성을 가정하였다[8]. 이 구성에서는 하나의 fragile 연결(F1)과 4개의 robust 연결(R1~R4)이 같은 게이트웨이에 연결되어 있다고 가정하였다. robust 연결과 fragile 연결을 시뮬레이션하기 위하여 연결의 종류에 따라 상이한 RTT를 가정하였다. 그러나, 이들이 가정한 (그림 1)의 시스템 구성은 응용에서 실제 전송할 데이터가 bursty하게 존재하는 경우를 직접 시뮬레이션하지 못하고 지연 시간의 차이에 따른 응답성의 차이에 따른 결과만을 시뮬레이션하였다.



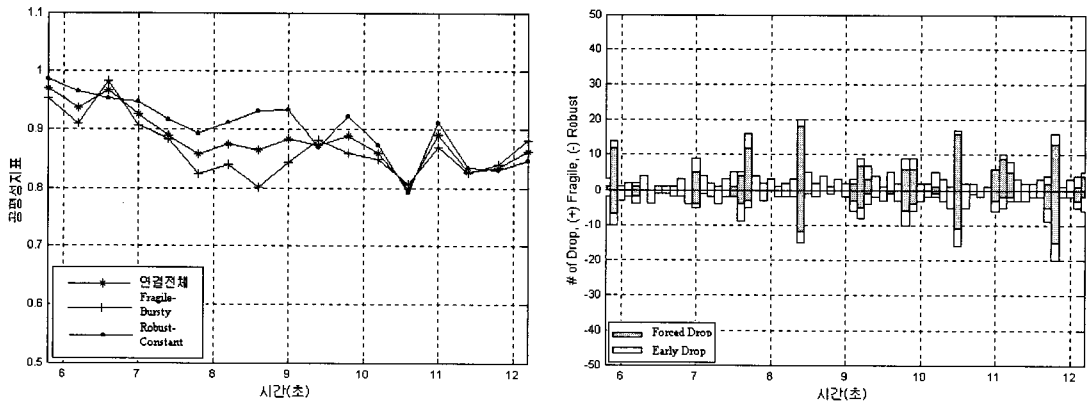
(그림 2) 제안 시스템 구성

본 논문에서는 실제 전송할 데이터가 bursty하게 있는 경우와 그렇지 않는 경우의 공평성 문제를 보이기 위하여 (그림 2)와 같은 시스템을 가정하였다. 비교의 대상이 되는 연결은 (그림 1)의 경우와 달리 네트워크의 연결 특성은 동일하나 상위의 응용이 서로 다른 특성을 가지는 트래픽을 생성한다. 시뮬레이션에서 다양

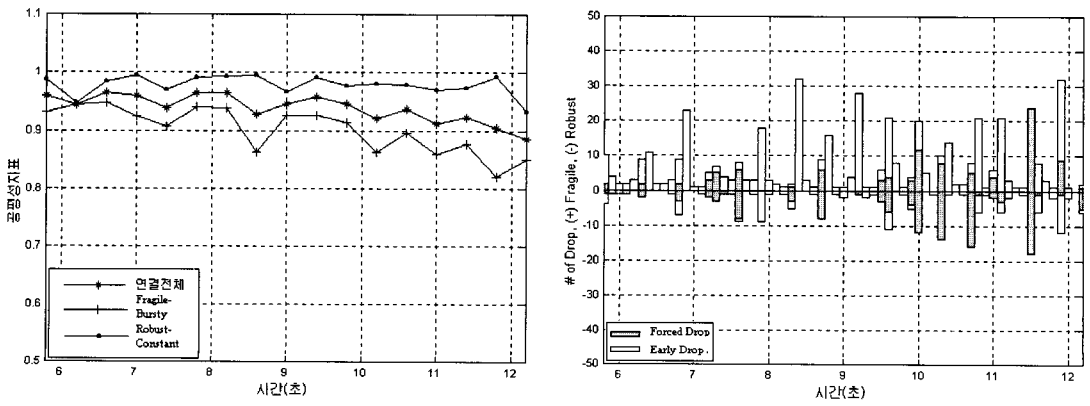
한 결과 분석을 위해 종류별 연결의 수를 충분히 많이 하였다. Robust 연결은 송신자가 일정한 전송율로 데이터를 전송하는 것을 가정하고 fragile 연결은 송신자가 특정 시간(busy period) 동안은 일정 전송율로 데이터를 전송하고 일정 시간(idle period) 동안은 아무 것도 전송하지 않는다고 가정하였다. 본 논문에서는 이



(a) DropTail 기법



(b) RED 기법



(c) FRED 기법

(그림 3) 게이트웨이의 큐 관리 방법에 따른 공평성 지표

경우의 robust 연결을 robust-constant 연결로 기술하며 fragile 연결을 fragile-bursty 연결로 기술한다.

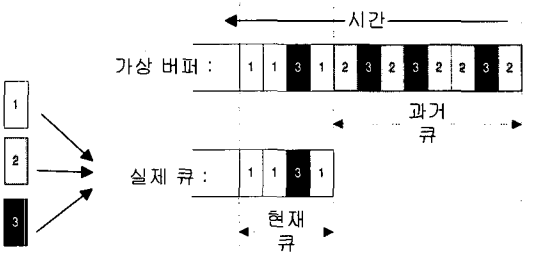
여러 가지 큐 관리 방법과 연결의 특성에 따른 공평성 변화를 비교하기 위하여 (그림 2) 시스템을 가정하여 시뮬레이션하였다[2](그림 3) 참조). 각 연결이 어느 정도 공평하게 대역폭을 점유하고 있는 지를 보이기 위하여 Jain의 공평성 지표인($(\sum_{i=1}^{n} b_i)^2 / n(\sum_{i=1}^{n} b_i^2)$, b_i: i 번째 연결의 대역폭 크기)을 사용하였다[10]. 이 때, 각 응용의 평균 트래픽 발생율은 200Kbps로 하였으며 fragile-bursty 연결의 경우 busy 구간 동안의 발생율은 2 Mbps로 하였다. fragile 연결의 대부분을 차지하는 web 트래픽은 busy 구간과 idle 구간이 주기적으로 반복되는 self-similar 특성을 가지며[13] request의 도착 시간 간격은 100ms~1000 ms을 가진다고 알려져 있으므로[13] busy 구간과 idle 구간의 길이의 비를 1 : 9로 하였다. 각 연결들 간의 동기화 현상을 막기 위하여 각 연결의 전송 시작 시간에 차이를 두었으며 처음에 TCP 윈도우 크기가 증가할 때까지의 각 연결의 차이를 나타내지 않기 위하여 6초부터의 공평성 지표를 비교하였다. (그림 3)의 왼쪽 그래프는 시간에 따른 공평성 지표의 변화를 보이며 오른쪽 그래프는 시간에 따른 유실 패킷의 특성을 막대 그래프로 표현하였다. 막대 그래프에서 x축을 기준으로 양의 방향은 fragile 연결, 음의 방향은 robust 연결을 표시하였다.

(그림 3)에서 보듯이 FRED인 경우, 전체 공평성 지표가 높으나 fragile-bursty 연결의 공평성 지표가 다른 방법에 비하여 크게 향상되었다고는 볼 수 없다. 이는 각 연결의 현재 큐 점유율에 근거하여 유실시킬 패킷을 결정하므로 fragile-bursty 연결은 busy 구간 동안에는 많은 패킷을 전송하여 큐를 많이 점유하고 있으므로 패킷이 유실될 확률이 높다. 유실 패킷의 시간에 따른 변화를 살펴보면 다른 기법과는 달리 FRED의 경우 fragile-bursty 연결의 early drop이 robust에 비하여 월등히 높다. 이는 공평성 유지를 위한 fair drop이 fragile-bursty 연결에 대하여 많이 적용됨을 의미한다. 이것은 FRED가 현재 버퍼의 상태만을 바탕으로 연결의 공평성 지수를 계산하여 fragile 연결의 bursty 특성을 제대로 반영하지 못 했기 때문이라고 여겨진다.

2) 시뮬레이션은 버클리 LBNL에서 구현한 ns 시뮬레이터를 사용하였다[14]. ns는 Tahoe, Reno, New-Reno. Vegas, SACK 등 여러 TCP 구현 및 RED, CBQ, DropTail 등의 Queue 모듈을 포함하고 있다.

3. 확장된 FRED 알고리즘

상위 응용에 따른 트래픽의 특성과 상관없이 모든 연결을 공평하게 서비스할 수 있는 FRED 방식은 다음과 같은 문제점을 가질 수 있다. 각 연결이 점유하는 자원을 계산하기 위하여 사용하는 척도인 현재 큐에 존재하는 연결 각각에 대한 점유율은 현재 큐 상태가 그 트래픽의 특성을 제대로 반영하지 못 하는 경우에 문제가 생길 수 있다. 즉, 연결의 특성이 매우 bursty한 경우에는 현재 큐가 burst 길이와 idle 길이에 따라 일부의 모습만을 반영하게 되므로 이를 이용하여 각 흐름 당 점유 자원을 계산하게 된다.



(그림 4) 가상 버퍼

이와 같은 문제점을 해결하기 위하여 본 논문에서는 가상 버퍼를 이용한 확장된 FRED(Ex-FRED) 기법을 제안한다. FRED에서는 평균 큐 길이를 계산할 때 시간에 따른 가중 평균을 사용하나 각 흐름 당 큐 길이를 계산할 때는 현재 버퍼에 포함된 흐름에 대해서만 계산하여 현재의 큐 상태만을 나타내나 Ex-FRED는 실제 큐보다 큰 가상 버퍼를 사용한다. (그림 4)에서 보듯이 실제 큐에는 연결 1에 속한 패킷이 큐의 대부분을 차지하고 있으나 이는 연결 1의 특성이 bursty하므로 나타나는 현상이다. 과거의 상황까지 살펴본다면 연결 1이 다른 연결에 비하여 대역폭을 많이 점유한다고 할 수 없으므로 패킷 유실 시 이를 고려해야 한다. Ex-FRED 의 상세한 알고리즘은 (그림 5)와 같다.

Ex-FRED 알고리즘에서 패킷을 유실할 때 기존의 FRED와 달리 가상 버퍼에 있는 흐름 당 패킷 수를 사용한다. 패킷을 유실하는 경우는 공평하게 자원을 사용하지 않고 더 많은 양을 사용하는 흐름의 패킷을 유실하는 경우(fair drop)와 평균 큐 크기에 따라 미리 유실하는 경우(early drop)로 나눌 수 있다. Fair drop

인 경우에 FRED에서는 평균 버퍼 크기가 최대 임계치 이상인 경우에만 흐름 당 큐 길이를 판단하여 패킷을 유실시켰으나 Ex-FRED에서는 평균 버퍼 크기가 최소 임계치 이상인 경우에도 흐름 당 큐 길이를 판단하여 패킷을 유실시켜 현재 버퍼의 상태가 꽉 차서 유실되는 현상을 방지하였다. 또한 평균 큐 길이가 최대 임계치 이상인 경우 FRED에서는 각 흐름 당 큐 길이의 2배 이상일 때에 패킷을 유실시켰으나 Ex-FRED는 β 값을 1.4로 하여 실제 큐 길이보다 큰 가상 버퍼에서는 실제 상황보다 트래픽 상태를 평활화(smoothing)함으로써 ex_qlen_i 값이 실제 큐의 상태보다 낮게 평가될 수 있음을 고려하였다.

Early drop에서는 평균 큐 길이가 최소 임계치와 최대 임계치 사이에 있는 경우, FRED에서는 흐름 당 평균 패킷 수가 최소 임계치를 넘고 흐름 당 패킷 수가 흐름 당 평균 패킷 수를 넘을 때 패킷을 유실 시켰으나, Ex-FRED에서는 가상 버퍼의 흐름 당 패킷 수가 가상 버퍼의 흐름 당 평균 패킷 수의 γ 배를 넘으면 유실율에 따라 패킷을 유실시킨다. γ 의 값을 1보다 작은 값(0.5)으로 하여 실제 큐보다 큰 가상 버퍼에서 평활화 현상으로 작아질 수 있는 흐름 당 패킷 수를 고려하고 (그림 3(c))에서와 같은 early drop에 의한 fragile-bursty 패킷의 유실을 방지하였다.

FRED 방법과 Ex-FRED 방법에서 유실없이 전송할 수 있는 최대 burst 길이를 각각 b_{FRED}, $b_{Ex-FRED}$라고 한다면 다음과 같은 식이 성립한다(<부록 1> 참조).

```
Constants :
        min_q(FRED) = 2;
        ex_size = Virtual Queue Size; (k * queue size, 4 < k < 8)
        ex_min_q = δ * min_q;                (ex_size:400인 경우 δ=2로 설정)
        α,β,γ : drop tuning parameters (1<β<2, ex_size:400인 경우 α=1.8,β=1.4,γ=0.5)
Global Variables :
        avg : average queue size
        avgcq : average per-flow queue size;
        NActive : Number of active connections(flow)
        ex_avgcq : average per-flow virtual queue size;
        ex_NActive : Number of Ex-Active Connections (flow)
Flow State :
        qlen_i : number of packets buffered in real queue;
        strike_i : number of over-runs
        ex_qlen_i : number of packets buffered in virtual queue;
Enque() : for each arriving packet P :
{       if flow i = conn(P) has no state table for its connection.
                create a new entry for flow i in state table.
        if queue is empty
                calculate average queue length(same as FRED)
// Idenitfy and manage non-adaptive flows
        if ( avg >= MINth && ex_qlen_i > α * ex_avgcq ||
             avg >= MAXth && ex_qlen_i > β * ex_avgcq )
                drop packet;                                    // Fair Drop
// Operate in random drop mode :
        if (MINth <= avg < MAXth) {                             // RED Alogorithm
                if ( ex_qlen_i >= ex_min_q && ex_qlen_i > γ * ex_avgcq)
                        drop packet with prob;                  // (FAIR) Early Drop
        } else if (avg < MINth) {
                no drop mode;
        } else { // avg >= MAXth
                drop packet;}                                   // Forced Drop
// Enque P
        if (New Flow) NActive++, ex_NActive++;
        Insert packet P to the (Virtual) Queue
        calculate average queue length
        accept packet P;}
Deque() : for each departing packet P :
{       calculate average queue length
        if (--qlen_i == 0) NActive--;
        if (--ex_qlen_i == 0) Delete Entry I from state Table, ex_NActive--;}
```

(그림 5) Ex-FRED 알고리즘

$$b_{FRED} \le x \cdot \frac{2}{N} \cdot T_B, b_{Ex-FRED} \le y \cdot \frac{\beta}{N} \cdot k \cdot T_B \quad \text{식 (1)}$$

(단, x, y > 1 인 상수)

Ex-FRED에서 $1<\beta<2$, $4<k<8$로 가정하였으므로 식 (1)에 의하면 Ex-FRED에서 유실없이 전송할 수 있는 burst 길이가 FRED 방식에 비하여 2~8배 정도 더 길다. 따라서 burst 길이가 긴 fragile 응용인 경우에도 Ex-FRED 방식이 유실없이 더욱 공평하게 서비스함을 알 수 있다.

4. 성능 분석

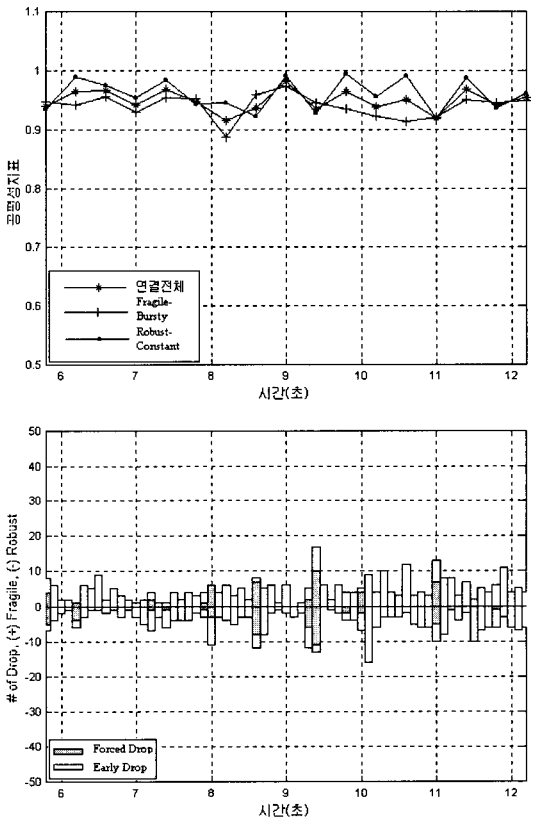
응용의 트래픽 특성에 따른 공평성 문제와 이를 해결하기 위한 Ex-FRED의 타당성 및 성능 검증을 위해 시뮬레이션을 수행하였다. 이를 위하여 FRED와 Ex-FRED 모듈을 직접 구현하여 ns에 추가시켜 실험을 수행하였다. FRED 구현의 타당성 검증을 위하여 [8]에서 수행한 여러 실험을 적용하여 그 결과를 비교하는 방법을 택했는데 대체로 일치하였다. FRED와의 호환성 검사를 위하여 [8]에서 수행한 여러 실험을 역시 적용하였으며 호환성 실험의 일부를 <표 1>에 표시하였다. 또한, 본 논문에서 살펴보고자 하는 fragile-bursty 연결과 robust-constant 연결의 공평성 비교를 위하여 (그림 2) 구성에서의 시뮬레이션을 수행하였다.

<표 1>에서 FRED*는 [8]의 실험 결과이며 FRED는 Lin과 Morris의 논문을 바탕으로 본 논문에서 구현한 FRED Queue의 결과이다[3]. Ex-FRED는 (그림 1)의 구성에서도 공평성을 잘 보장함을 알 수 있다.

〈표 1〉 (그림 1) 구성에서의 각 방법의 성능 비교
(버퍼 크기=32)

Queue 종류	DropTail	RED	FRED*	FRED	EX-FRED
F1의 BW/LinkRate	0.8%	6.1%	20%	22%	20%
F1의 Loss 비율	3.2%	0.135%	0.06%	0.08%	0.01%
R1의 Loss 비율	0.23%	0.16%	1.21%	0.90%	0.99%

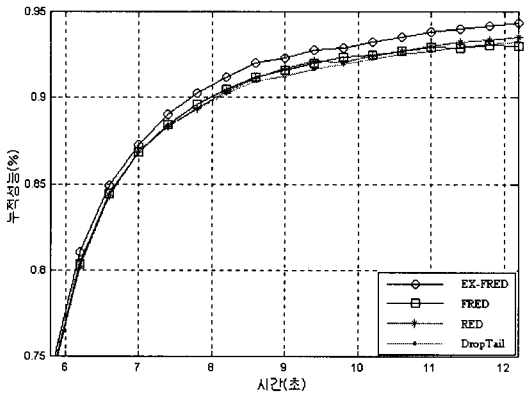
(그림 2)와 같은 구성에서 가상 버퍼의 크기를 400 패킷으로 하였을 때의 공평성 지수와 유실 패턴은 (그림 6)과 같다. (그림 3) 그래프와 (그림 6) 그래프를 비교하면 Ex-FRED 기법의 전체 공평성 지표가 더 높으며 robust-constant 연결과 fragile-bursty연결의 공평성 지표의 차이도 줄어들었으며 유실된 패킷의 분포도 fragile-bursty와 robust-constant 연결에 차이가 없음을 알 수 있다.



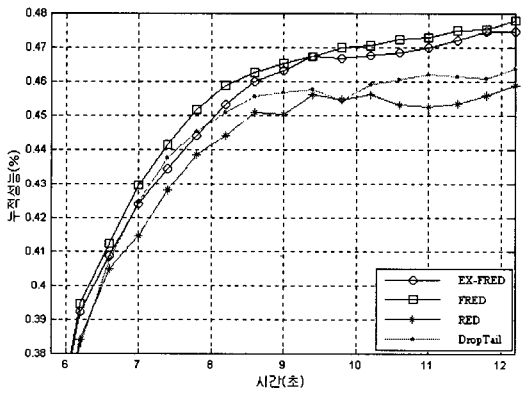
(그림 6) Ex-FRED의 공평성 지표과 유실 패턴

연결들 간의 성능을 비교한 그래프는 (그림 7)과 같다. 성능의 척도로는 시간 당 유실 없이 전송한 패킷을 나타내는 goodput을 사용하였다. (그림 7)의 y축은 각 방법의 측정한 goodput과 최대 처리량(8Mbps)의 비를 나타낸다. (그림 7)에서 보듯이 전체 성능의 차이는 미미하나 Ex-FRED가 가장 우수하며 연결의 종류와 관계없이 높은 처리율을 갖는다. robust-constant 연결의 성능인 경우에는 FRED 가장 우수하나 fragile-bursty 연결의 경우는 FRED가 가장 성능이 낮음을 알 수 있으며 가상 버퍼의 크기에 따른 성능의

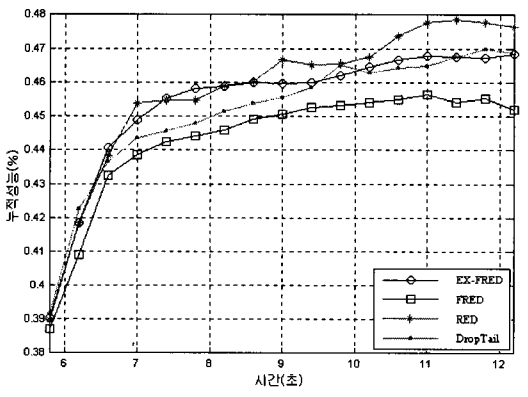
3) 두 결과가 다른 이유는 다음과 같다. 첫째, 상이한 TCP 구현을 사용하였다. [8]에서 TCP/New Reno를 수정한 구현을 사용하였으나 본 논문에서는 현재 가장 많이 사용되는 TCP 구현인 TCP Reno를 사용하였다. 둘째, 상이한 RED 인자를 사용하였다. [8]에서 적용한 RED 인자 중 Maxth 값이 다른 큐에 불리하게 동작하여 일반적으로 사용되는 인자인 3/4 * Queue Size로 조정하였다.



(a) 전체 연결의 성능



(b) robust-constant 연결의 성능



(c) fragile-bursty 연결의 성능

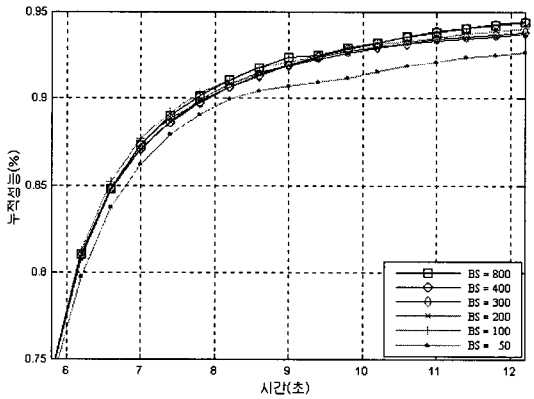
(그림 7) 누적 성능 비교

차이를 시뮬레이션한 결과는 (그림 8)과 같다. 가상 버퍼의 크기는 50, 100, 200, 300, 400, 800 패킷 크기로 변화함에 따라 확장 FRED의 성능과 공평성 지표를 비교하였으며 이 때 가상 버퍼 크기에 따른 인자들은 <표 2>와 같다. 각각의 인자는 가상 버퍼의 크기에 따른 트래픽의 평활화 요소를 고려하여 정한 것이다.

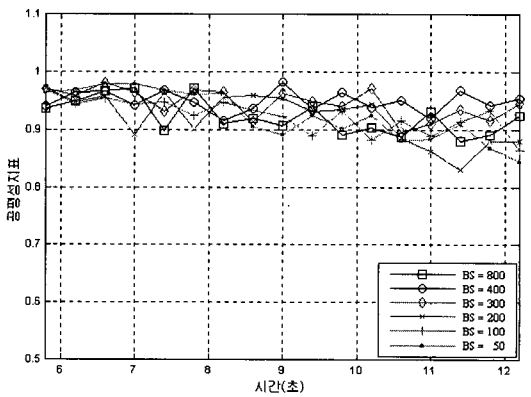
(그림 8)에서 보듯이 대부분 비슷한 성능을 보이나 가상 버퍼의 크기가 실제 버퍼 크기와 같은 50일 경우

〈표 2〉 가상 버퍼 크기와 인자

가상버퍼크기(패킷)	alpha	beta	gamma	delta
50	4.0	2.0	1.0	1.0
100	3.0	1.8	0.8	1.2
200	2.5	1.7	0.7	1.4
300	2.1	1.5	0.6	1.8
400	1.8	1.4	0.5	2.0
800	1.7	1.3	0.5	2.0



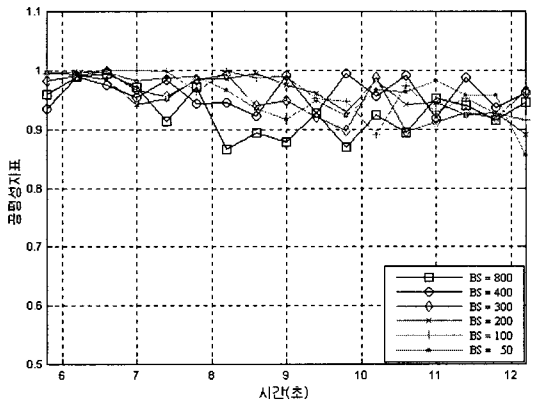
(a) 성능 비교



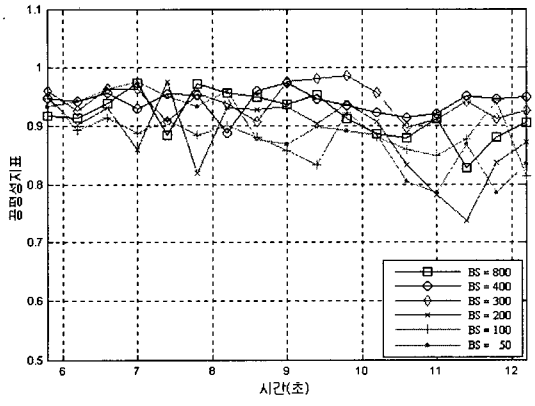
(b) 공평성 지표 비교

(그림 8) 가상 버퍼의 크기에 따른 성능 및 공평성지표

에는 성능이 낮았다. 공평성 지표는 가상 버퍼의 크기와 관계없이 대부분 우수하였다. 연결의 종류에 따른 공평성 지표를 비교해 보면 (그림 9)에서 보듯이 대체적으로 가상 버퍼의 크기가 크면 fragile-bursty 연결의 공평성을 잘 지원하며 작으면 robust-constant 연결의 공평성을 잘 지원한다.



(a) robust-constant 연결



(b) fragile-bursty 연결

(그림 9) 가상 버퍼 크기와 연결의 종류에 따른 공평성 지표 비교

5. 결 론

인터넷이 전달하는 트래픽은 과밀현상이 발생되었을 때 제어되는 방법에 따라 과밀 제어를 하지 않는 UDP 기반의 트래픽, 미리 트래픽 인자를 협상 제어하는 트래픽, 과밀 발생에 따라 동적으로 제어되는 TCP 기반의 트래픽으로 나눌 수 있다[9]. 이 중, 본 논문에서는 현재의 대부분의 트래픽이 해당되는 TCP로 제어되는 분류에 속하는 트래픽 중에 항상 전송할 데이터가 있는 응용에서 발생한 robust 트래픽과 상호 동작적인 응용에서 발생한 fragile 트래픽이 공평하게 자원을 사용할 수 있는 큐 관리 방법인 확장된 FRED(Ex-FRED)를 제안하였다.

같은 TCP 방식으로 과밀 제어를 수행하더라도 robust 연결과 fragile 연결이 공평하게 제어되기 힘들다. 이러한 불공평성의 원인으로 많은 논문들에서 RTT의 차이와 같은 TCP 연결 자체의 속성 차를 지적하였다. 그러나 본 논문에서는 연결 자체의 속성은 동일하더라도 응용 트래픽의 발생 형태에 따라 대응 속도에 차이가 발생할 수 있음을 보이고 이의 해결책을 제시하였다. 즉, 현재 게이트웨이의 버퍼의 상태에 따라 모든 연결에 공평하게 패킷을 유실시킨다는 것은 자신의 공평한 몫을 항상 일정하게 사용하지 못 하고 시간에 따라 과도하거나 사용하지 못 하는 경우가 발생하는 fragile 연결에 불리하게 작용할 수 있다.

제안한 Ex-FRED 방식에서는 이러한 문제점을 해결하기 위하여 각 흐름의 가상 버퍼 점유율에 기반하여 혼잡 시 패킷을 유실시키는 방법을 사용하였다. 가상 버퍼를 사용함으로써 burst 길이가 긴 보통의 상호 동작적 응용에서 과도하게 패킷이 유실되어 자원을 덜 사용하게 되는 경우를 방지하였다. 실험 결과에 의하면 Ex-FRED 방식이 다른 방식에 비하여 연결의 종류에 관계 없이 공평하게 자원을 사용하였으며 고른 성능을 보였다. 또한 가상 버퍼의 크기가 클수록 fragile 연결의 공평성을 잘 지원하였다.

물론 가상 버퍼를 사용함에 따라 라우터에서 유지하여야 할 연결별 공평성 정보의 양이 가상 버퍼의 크기에 비례하여 증가한다. 그러나 연결의 실제 데이터 정보가 아닌 공평성 제어 정보를 유지하므로 필요한 메모리의 양이 크지 않다. 그리고 알고리즘 처리와 관련한 부담은 기존 FRED 수준이라고 할 수 있다. 따라서 이 방법을 사용한다면 웹 트래픽과 같은 상호 동작적 응용의 불평등한 자원 사용을 피하여 사용자에게 효과적으로 좀 더 빠른 응답을 제공해 줄 수 있을 것이다.

부 록

FRED 알고리즘[8]에 의하면 유실없이 버퍼에 저장

될 수 있는 패킷의 수는 $qlen_i < \frac{avg}{N_{active}} \cdot 2$이고 $\frac{avg}{N_{active}} \leq \frac{B}{N_{active}} = \frac{B}{N} \cdot \frac{N}{N_{active}} = x \cdot \frac{B}{N}$(단, N : 전체 연결의 수, B : 버퍼 크기)이다. b_{FRED}는 $\frac{qlen_i}{C}$(단, C는 링크의 전송율)로 계산될 수 있으므로 $b_{FRED} \leq x \cdot \frac{2}{N} \cdot \frac{B}{C} = x \cdot \frac{2}{N} \cdot T_B (T_B = \frac{B}{C})$이다. 마찬가지 방법을 Ex-FRED 알고리즘에 적용하면 $b_{Ex-FRED} \leq y \cdot \frac{\beta}{N} \cdot \frac{V}{C} = y \cdot \frac{\beta}{N} \cdot k \cdot T_B$(단, V는 가상 버퍼 크기)이다.

참 고 문 헌

[1] V. Jacobson, "Congestion Avoidance and Control," ACM Computer Communication Review, Vol.18, No.4, pp.314-329, Aug. 1988.

[2] P. Ferguson and G. Huston, "Quality of Service in the Internet: Fact, Fiction, or Compromise?," INET98, 1998.

[3] M. Mathis, and et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," ACM Computer Communication Review, Vol.27, No.3, July 1997.

[4] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, Vol.1, No.4, August 1993.

[5] S. Floyd and V. Jacobson, "On Traffic Phase Effects in Packet-Switched Networks Part 1 : One-way traffic," ACM Computer Communication Review, April 1991.

[6] E. Hashem, "Analysis of Random Drop for Gateway Congestion Control," MIT-LCS-TR-465.

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

[8] D. Lin and R. Morris, "Dynamics of Random Early Detection," ACM SIGCOMM 97, pp.127-137, 1997.

[9] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, August 1999.

[10] R. Jain, "The Art of Computer Systems Performance Analysis : Techniques for Experimental Design, Measurement, Simulation and Modeling," John Wiley & Sons, New York, 1991.

[11] D. Lin, "Internet Congestion Control : Cooperative End-System and Gateway Algorithms," phD thesis, Harvard Univ., May 1998.

[12] G. Abdulla, "Analysis and Modeling of World Wide Web Traffic," phD thesis, Virginia Univ., May 1998.

[13] M. Nabe, et. al., "Analysis and modeling of World Wide Web traffic for capacity dimensioning of Internet access lines," Performance Evaluation, Vol.34, No.4, 1998.

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

우 희 경

e-mail : woohk@cse.sch.ac.kr

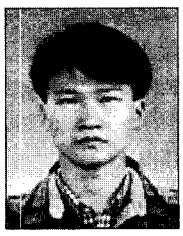
1992년 서울대학교 계산통계학과 졸업(학사)

1994년 서울대학교 대학원 전산과학과 졸업(석사)

1998년 서울대학교 대학원 전산과학과 졸업(박사)

1998년~현재 순천향대학교 컴퓨터학부 전임강사

관심분야 : 멀티캐스트, 과밀제어, 차세대인터넷, TCP, 멀티미디어 등



김 종 덕

e-mail : kimjd@brutus.snu.ac.kr

1994년 서울대학교 계산통계학과 졸업(학사)

1996년 서울대학교 전산과학과 졸업 (석사)

1996년~현재 서울대학교 전산과학과 박사 과정 재학 중

관심분야 : TCP, 차세대인터넷, 과금, Integrated service, diffserv 등