공학석사 학위논문

Core-Stateless망에서 FNE 알고리즘을 이용한 공정한 대역폭 할당에 관한 연구

지도교수 박 승 섭

이 論文을 [illegible] 學位論文으로 提出함

2004년 [illegible]월

부경대학교 산업 대학원

전산정보학과

서 경 현

이 논문을 서경현의 공학석사
학위논문으로 인준함

2004년 6월 19일

주 심 이학박사 박 홍 복 

위 원 공학박사 정 연 호 

위 원 공학박사 박 승 섭 

<차 례>

<그림 차례>

<표 차례>

A Study on Fair Bandwidth Allocation using FNE algorithm in Core-Stateless Networks

Kyoung-Hyun Seo

Dept. of Computer and Information Graduate School of Industry, Pukyong National University

Abstract

Many per-flow scheduling algorithms have been proposed to provide fair rate and delay guarantees to data flows, and designed to achieve fair bandwidth allocation, such as fair queueing algorithm which has many desirable properties for congestion control in Internet. But the functions of such algorithms need to maintain rate state, manage buffer, and schedule packet on a per-flow basis; this complexity of functions may prevent them from being cost-effectively implemented.

Therefore, in this paper, to acquire cost-effectivelness for router's implementation, we propose a CS-FNE algorithm based on FNE, and evaluate CS-FNE scheme together with CSFQ, FRED, RED, and DRR in several different configurations and traffic source in core-stateless networks. Through simulation results, we showed that our proposed CS-FNE algorithm can allocate approximately fairer bandwidth than other algorithms, and CS-FNE is simpler than many per-flow basis queueing mechanisms and also can be easily implemented.

1. 서론

최근 인터넷 트래픽은 데이터 통신의 발달로 급속히 증가해 가고 있다. 단순한 비 실시간 데이터 전송 위주였던 인터넷 서비스에서 멀티미디어 음성 및 실시간 화상 데이터와 온라인 게임, P2P, 원격교육, 전자 상거래 등 다양한 서비스가 생겨남에 따라 네트웍상의 공정성, 확장성, 보안성 등 여러 방면에서 기존 인터넷 서비스보다 향상된 네트웍을 요구하게 된다.

인터넷망의 고속화와 실시간과 비실시간 서비스 품질을 위해 IETF(Internet Engineering Task Force)에서 MPLS(Multi-Protocol Label Switching) 기술을 정의하고 있다. 그러나 모든 망을 MPLS 라우터로 변환한다는 것은 많은 비용이 든다. 또한 네트웍 방식도 변모해야 한다. 그리고 네트웍 망에서 중계역할을 하는 노드가 같은 방식으로 대역폭 관리의 의미에서 TCP/IP 데이터를 처리를 하는 것도 많은 비용이 든다. 그래서 중심 노드 즉 Core Node와 경계 노드 즉 Edge Node의 역할을 따로 두어 처리하는 Core-Stateless 망에 대한 연구가 이루어지고 있다. 대표적으로 Core-Stateless 망에서의 대역폭 공정성 연구에 대한 CSFQ(Core-Stateless Fair Queueing) 알고리즘이 있다[1].

인터넷 트래픽은 음성, 비디오, 게임 데이터가 증가하고 있다. 그래서 네트웍은 자원을 효율적으로 제어할 필요가 있다. 이러한 인터넷 혼잡제어는 종단간 전송에 있어서 안전성과 효

율성을 요구한다. 그리고 종단간 혼잡제어 알고리즘은 대역폭 할당에 있어 공정하게 할당할 필요가 있다. 공정한 대역폭 할당을 위해 흐름마다(per flow) 큐잉 제어 메커니즘이 전형적으로 많이 연구되어 왔다. 대표적으로 FQ(Fair Queueing), WFQ(Weighted Fair Queueing), 흐름마다 폐기에 중점을 둔 FRED(Flow Random Early Drop) 등 있다[2][3][4]. 이러한 메커니즘은 기본적인 FIFO 큐잉 방식보다 구현상에 복잡하고 데이터 처리에 대한 속도도 늦을 뿐만 아니라, 모든 노드가 패킷의 분류와 흐름마다 혼잡제어를 하기 때문에 고속 망에서는 적합하지 않다.

최근 이러한 복잡성의 문제점을 해결하기 위해 여러 가지 프레임웍과 메커니즘의 제안되어 왔는데, 특히 Stoica는 Core Stateless 망에서 CSFQ를 제안하였으며, Cao는 RFQ(Rainbow Fair Queueing)를 제안하였다[1][5]. 두 방식의 차이점은 크게 CSFQ (with per flow)는 패킷 흐름마다 처리하고 흐름 번호에 따라 label i를 할당한다. RFQ (without per flow)는 패킷 흐름 당 두지 않고 제어하며 패킷의 평균율에 따라 color label i를 할당한다. 그러나 들어오는 패킷율을 지수분포로 계산하기 때문에 복잡성에 있어 단순화될 필요가 있다.

최근 제안된 Li의 FNE(Flow Number Estimation) 큐잉기법은 해쉬 함수를 이용해서 큐의 관리 기반 패킷률을 계산하는 방법을 단순하게 구현하는 방법을 제안하였다[6]. 본 연구는

이 알고리즘을 이용해서 Core-Stateless 망에서 큐잉의 복잡성을 줄이고 공정하게 대역폭을 할당하는 방식을 구현하기 위해, FNE 기법을 Core-Stateless 망에 적용 및 망에 맞게 구현하였다. 이를 본 연구는 CS-FNE(Core Stateless FNE) 명명하여 여러 알고리즘과 비교 분석한다.

본 논문의 구성은 서론에 이어, 2장 관련연구에서는 Core-Stateless 망과 공정 대역 할당의 대표적인 알고리즘들을 설명하고, 3장 CS-FNE에서는 FNE 기법을 Core-Stateless 망에 적용하는 방법을 설명하고, 4장 시뮬레이션 환경에서는 시뮬레이션 모델과 파라미터에 대하여 설명하고, 5장에서는 모의실험 결과 분석을 한다. 마지막 6장은 본 논문의 결론이다.

2. 관련연구

WFQ은 지연과 대역폭 공정한 할당을 고려하였다. 그러나 WFQ는 실질적으로 구현하는데 복잡하다. 정확성과 복잡성을 최소화하기 위해 변형된 WFQ가 제안되기도 하였다. 그럼에도 불구하고 WFQ와 관련된 방식들은 per-flow 상태 관리 유지와 classification을 요구하는 큐잉 기법이다. 그래서 많은 흐름을 가진 고속망에서는 복잡성의 단순화가 고려되어 진다.

RED는 단순 drop tail 방식을 향상하기 위해 제안되어 졌다 [8]. RED는 매 패킷이 도착할 때 Qavg를 구하고, 미리 정해놓은 파라미터 max_{th}와 min_{th} 값에 따라 처리한다. 그러나 파라미터 값이 큐잉 기법에 미치는 영향이 크고, 공정성을 보장할 수 없는 단점이 있다. RED를 개선시키기 위해 제안된 FRED는 흐름당 제어를 통해 공정성을 개선하였다. FRED는 공정성 개선은 되었으나, 평균 큐 길이가 max_{th} 보다 클 경우 RED 방식 보다 더 많은 패킷을 손실된다. 본 장은 Core-Stateless 망과 per-flow 상태 관리를 하는 대표적 알고리즘인 DRR과 per-flow 상태 관리를 하지 않는 대표적 알고리즘인 RED를 설명한다.

2.1 Core-Stateless Network

기존의 Network 구조는 (그림 1) (a)와 같이 나타낼 수 있으며, 모든 노드들이 FQ(Fair Queueing) 알고리즘으로 구성되어 있다.

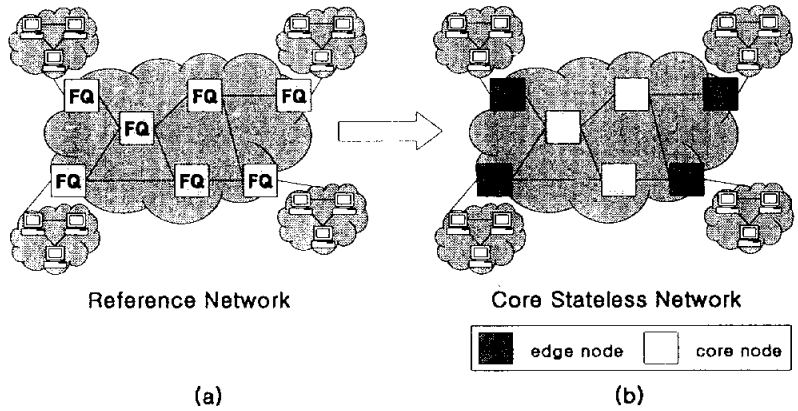
FQ 알고리즘들이 폭주 제어를 위해 많은 특성을 가지고 있다. 하지만 이러한 특성들로 인해 구현 복잡도가 야기되고, 이는 고속의 서비스를 제공하기 위해 높은 비용을 요구하게 된다.[1]

Core-Stateless 망 구조는 (그림 1) (b)와 같이 나타낼 수 있으며, 흐름 단위의 상태 관리를 수행하는 Edge node와 흐름단위 상태 관리를 수행하지 않는 Core node로 구성된다.

Edge node는 패킷을 흐름 단위로 분류하고, 흐름 단위로 속도를 예측하여 계산하며, 속도 또는 속도에 의해 계산된 흐름 정보를 패킷 헤더에 실어 Core node로 전달한다. Core node는 흐름 단위의 상태 관리 없이 이 레이블 정보를 사용한다. 또한 Core-Stateless 네트워크에서는 공평성 보장을 위해 필요한 흐름 단위의 버퍼링이나 스케줄링을 하지 않고 각 라우터는 확률적인 폐기 알고리즘을 사용하는 FIFO 큐잉을 사용한다. 패킷을 폐기하는 확률은 레이블에 실려온 정보와 라우터에서 통합 트래픽의 측정에 의한 값 즉 라우터의 상태에 의해 결정된다. 패킷 폐기 알고리즘도 각 공평 대역 할당 메커니즘의 레이블 정보와 통합 데이터에 따라 다양하게 나타난다.[14]

즉, Core-Stateless Network는 기존 Network FQ 통신망의

기능에 가까워지기 위한 Network 구성이며, 낮은 비용으로 고속의 서비스를 제공할 수 있게 제안되었다.



(그림 1) (a) Reference Network
(b) Core-Stateless Network

2.2 CSFQ(Core-Stateless Fair Queueing)

CSFQ은 Core-Stateless Network를 기반으로 대역 공정성을 지원하는 대표적인 메커니즘이다.

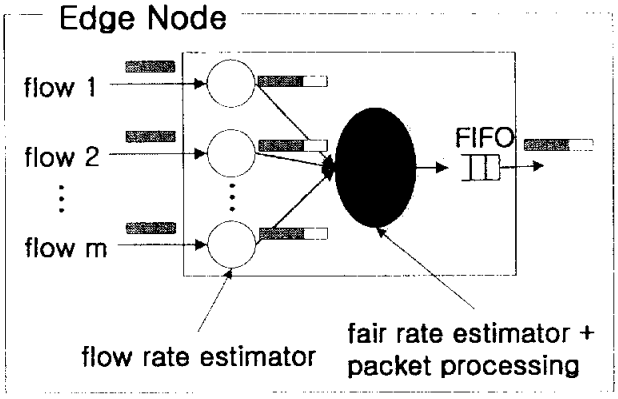
(그림 2)와 (그림 3)은 CSFQ 알고리즘에서 Edge node와 Core node의 동작 방식을 도식화 한 것으로, Edge node는 각 흐름의 유입율 r을 측정하고 패킷 헤더에 label을 붙여 Core node로 전송한다.

모든 Edge node와 Core node는 정기적으로 공정률 f를 측정하고, 각 패킷에 식 (1)에 의거한 확률 P를 계산하여 같이 전송하고,

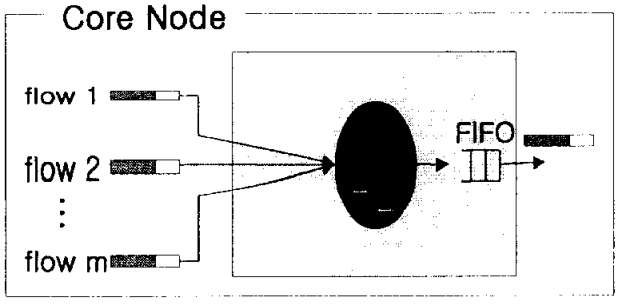
$$P = \min\left[1, \frac{f}{r}\right] \qquad (1)$$

패킷 label을 식 (2)에 의해 r'를 계산하여, label을 업데이트를 한다.

$$r' = r \times \min\left[1, \frac{f}{r}\right] = \min(f, r) \qquad (2)$$



(그림 2) Edge Node



(그림 3) Core Node

2.3 DRR(Deficit Round Robin)

DRR은 각 flow의 평균 큐 크기를 알지 않고도 가변길이의 패킷을 고려해 공정성을 제공할 수 있도록 고안된 방법이다. 각각의 흐름은 deficit이란 계수를 가지며 스케줄러는 각 흐름을 번갈아 가며 방문한다. 그리고 quantum 만큼의 양을 서비스한다. 각 흐름의 앞에 있는 패킷은 만약 패킷의 크기가 quantum의 크기 보다 크면 quantum은 각 흐름의 deficit 계수에 더해진다.[15]

DRR의 장점은 구현이 쉬우면서 WRR의 프레임 시간 내에서는 공정성이 약화된다는 단점을 극복한 것이다. 따라서 프레임 시간이 작거나 공정성 요구가 크지 않은 상황에서 잘 동작한다.[11]

2.4 RED(Random Early Drop)

RED는 큐의 길이가 임계치를 넘으면 평균 큐 길이에 기반해 계산된 확률로 패킷을 폐기하는 방식이다. RED의 목표는 평균 큐 크기를 제어하며 폭주를 회피하고 TCP의 전역 동기화를 해결하는 것이다.

RED알고리즘은 패킷이 도착할 때마다 평균 큐의 길이를 구해 정해진 최소 임계값과 최대 임계값을 비교하여 최소 임계값 이하이면 정상적인 상태로 인식, 두 임계값 사이면 랜덤하게 패킷을 드랍하고, 최대 임계값 이상이면 입력 패킷을 모두 드랍한다.[8]

RED가 특별한 타입의 flow들을 차별하지 않는 동안 이것이 모든 flow들에 대역폭의 공정한 배분을 보증하지 않는다. 사실상 RED가 저속의 TCP flow에 공정하지 않다. 이는 RED가

Minimum threshold를 지날 때 패킷을 random하게 drop시키기 때문이며 이것은 flow의 공정한 bandwidth 배분보다 현재 사용되는 것이 더 적은 flow에 패킷이 속하게 될 수도 있다.[15]

RED의 응용으로 FRED(Flow RED)는 트래픽 유형을 분리하여 서비스를 한다. 트래픽의 구분은 다음과 같다.

○ UDP 사용 트리픽(비디오나 오디오 트래픽) : TCP체증 제어 방식에 순응하지 않는 트래픽

○ TCP 체증 제어에 순응하지만 허락하는 범위에서 가용대역을 모두 차지하는 트래픽

○ Telnet 서비스 같이 가용대역을 사용하는데 민감하게 반응하지 않는 트래픽

라우터의 큐에서 각 TCP 연결이 차지하는 큐 길이를 추적함으로써 트래픽 유형에 상관없이 모두 공정하게 링크 사용하게 한다.[12]

3. CS-FNE(Core-Stateless FNE)

본 장에서는 FNE 기법을 Core-Stateless 망에 적용하는 방법을 설명한다.

3.1 FNE(Flow Number Estimation) 기법

FNE 기법은 문자 그대로 실질적으로 버퍼에서 차지하고 있는 흐름의 숫자를 파악하는 것을 말한다. 망은 지속적으로 데이터를 전송한다고 하더라도 on-off를 가진다. 그러므로 버퍼에 유입된 흐름의 개수를 정확히 산출해 내면, FIFO 큐에서 임계치를 알 수 있으며, 공정하게 대역폭을 할당 할 수도 있다. 기본적으로 이상적인 패킷 폐기를 위해 유입된 양은 링크 용량은 식(3)과 같이 C를 넘지 말아야한다.

$$\sum r_{i'<C} \qquad (3)$$

에서 min값은 min(arrival rate, threshold rate) 정의되어 질 수 있다. 만약 n 개의 흐름이 버퍼에 유입될 때 유입된 양이 전체 용량을 넘을 때, 패킷이 폐기된다. 따라서 전체 유입된 도착율은 $r_1 \leq r_2 \leq, \ldots, \leq r_n$으로 정렬 될 수 있다. 따라서 임계치 또는 CL(Cutting Line)은 n+1 중에 하나가 된다. 다시 말해서 $0 < CL \leq r_1$, $r_1 \leq CL \leq r_2$, ... , $r_{n-1} \leq CL \leq r_n$, or $r_n \leq CL < \infty$을 의미한다. 따라서 흐름 n+1에

서 CL값은 최대값을 식(4)와 같이 얻을 수 있다.

$$\sum_{i=1}^{n} \min(r_i, CL) = \frac{n \cdot CL + \sum_{i=1}^{n} r_i - \sum_{i=1}^{n} |r_i - CL|}{2} \qquad (4)$$

따라서 패킷이 CL을 넘을 시에는 폐기된다.

3.2 FLow Rate Measurement

유입되는 흐름율은 정확성과 trace 능력이 고려되어진다. 유입되는 흐름율을 계산하기 위해서 FNE 기법은 TSW(Time Sliding Window) 방식을 채택하였다[7]. TSW 알고리즘은 율 측정 기법 중 평균율을 기반으로, 경계라우터의 서비스 프로파일에 명시된 목표 전송률 초과 여부로 폐기 우선 순위를 결정하여 패킷을 IN이나 OUT으로 마킹하는 알고리즘이다. TSW는 매 패킷이 도착할 때마다 전송속도를 측정하게 되며, 윈도우의 크기에 따라 과거 전송률에 대한 정보가 현재의 전송률 계산에 반영되는 정도가 달라지게 된다. 따라서 전송률 측정부분에서 현재의 전송률을 과거의 전송률과 합한 평균으로 버스티한 데이터를 완만하게 만드는 작용을 한다.

각 흐름을 정보를 기록하기 위해서 m 슬롯을 가진 해쉬 테이블을 사용한다. 이 정보는 각 패킷의 헤더 라벨링을 하기 위해서도 필요하다. 해쉬 테이블에서 해쉬 함수를 통해 슬롯의 위치를 식(5)과 같이 계산하는데, 해쉬 키값은 Src-Des 주소

를 짝으로 한다. 해쉬키S_Di는 해싱 함수에 의해서m 슬롯과 매핑된다.

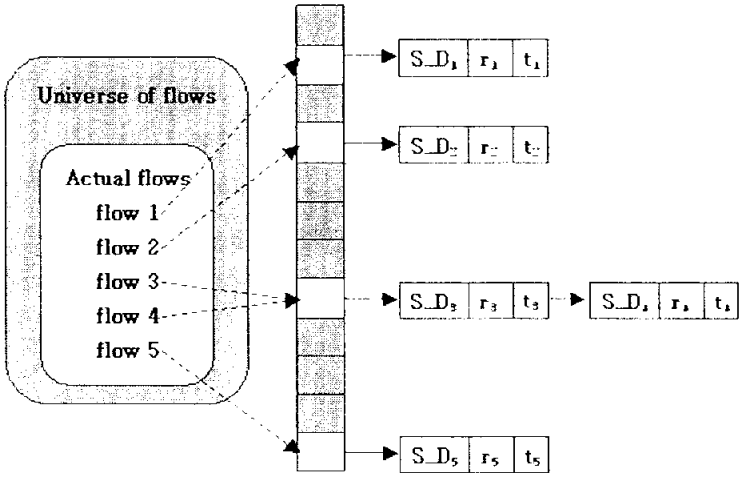
$$h(S\_D_i) = S\_D_i \bmod m. \qquad (5)$$

여기서 S_Di는 키값이고 m은 슬롯을 의미한다. 해쉬맵으로 흐름단위 정보(per-flow information)을 관리하는 것은 흐름단위 큐(per-flow queue)보다 메모리사용 효율성이 좋다. 그러나 같은 슬롯에서 per flow information 있어 문제가 발생 될 수 있다. 이러한 문제를 해결하는 방법으로 rehashing과 chaining 방법이 있다. FNE는 chaining 방법을 사용한다.식(6)는 n개의 흐름이 라우터를 통과할 때 해쉬 테이블이 m개의 슬롯을 가진다고 할 때, 같은 슬롯에 적어도 2개의 flow 맵이 있을 확률이다.

$$P_{collision} = 1 - \frac{m!/(m-n)!}{m^n} \qquad (6)$$

해쉬 테이블에서 충돌을 방지하기 위해서, (그림 1)에서 보는바와 같이 링크드 리스트를 이용해서 같은 슬롯에 있는 정보를 관리한다. 다시 말해서 per-flow information을 슬롯 당 포인터로 관리한다는 의미이다. (그림 4)에서 같이 flow 3과 flow 4는 링크드 리스트에 의해 해쉬되어 같은 슬롯에 위치하고 있다. 이와 같이 Src-Dst 해쉬 키 값과 측정한 흐름율, 시간과 같은 per-flow information만 해쉬의 링크드 리스트에 저

상된다.



(그림 4) chaining 방법에 의한 해쉬 테이블에서의 충돌 해결 방법

3.3 Active Flow 측정

실제적인 흐름(Active flows) N_{act}는 큐에서 랜덤하게 선택된 패킷과 유입된 패킷을 비교하여 측정한다. 그러나 패킷 기반 비교측정은 실질적인 네트워크 기반에 적합하지 않다, 왜냐하면 패킷의 길이는 고정적이지 않고 가변적이기 때문이다. 그래서 n 개의 흐름이 라우터를 통과한다고 가정할 때, r_i와 p_i는 흐름 i의 버퍼 용량과 TSW에 의해서 측정된다. 여기서 p_i는 큐에 있는 흐름 i에 속한 패킷의 비율이다. 그리고 N_{act}를 구하기 위해 r_{hit}와 r_{miss}의 값이 식(7)와 식(8)과 같이 산출된다.

r_{hit}은 큐에 랜덤하게 선택된 패킷중에 하나로 유입되는 패킷의 흐름율이다.

$$r_{hit} = \sum_{i=1}^{n} r_i \cdot P_i \qquad (7)$$

$$r_{miss} = \sum_{i=1}^{n} r_i \cdot (1 - P_i) \qquad (8)$$

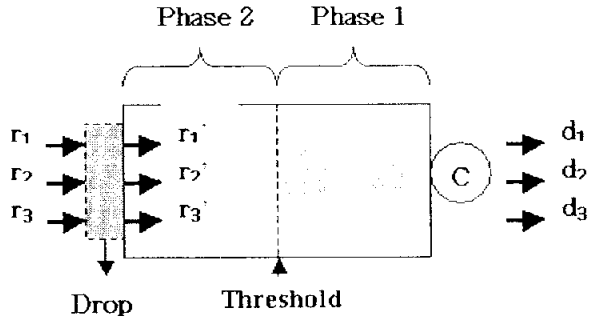
그래서 active flow는 식(9)과 같이 정의되어 진다.

$$N_{act} = \frac{r_{hit} + r_{miss}}{r_{hit}} \qquad (9)$$

따라서 FIFO 큐에서 p_i는 r_i에 비례되고, 식(10)과 같이 산출 될 수 있다.

$$N_{act} = \frac{n}{C_r^2 + 1} \qquad (10)$$

여기서 C_r^2는 $\delta_r / E[r]$의 도착률의 상관관계와 같다. 그러므로 *fair_rate* 지표는 $link\_capacity / N_{act}$ 이며, 다시 말해서 (그림 5)와 같이 버퍼 임계치가 된다. 이 값은 매 패킷이 들어올 때 마다 갱신된다.



(그림 5) FNE의 버퍼관리

간단히 예를 들어 링크 전체 용량은 10M이고, r_1= 2, r_2=6, r_3=8 *fare_rate*가 4라고 가정할 때, min(r_1, 4)=2, min(r_2, 4)=4, min(r_3, 4)=4로 결과 값을 나타낼 수 있다. 또한 폐기 확률은 식(11)와 같다.

$$p_1 = 1 - (4 / r_1) = 0$$

$$p_2 = 1 - (4 / r_2) = 0.33$$

$$p_3 = 1 - (4 / r_3) = 0.5 \qquad (11)$$

이러한 일련의 과정을 통해 Core-stateless 망에서 egress 노드와 ingress 노드를 관리하기 위해, Core-stateless 망에서의 FNE를 구현하여 모의실험을 하며, CS-FNE라 명명하였다.

3.4 패킷 라벨링

패킷의 도착률이 측정되어진 후, 본 연구에서 패킷의 라벨은 해쉬 맵의 슬롯 번호에 따라 할당하였다. 패킷 라벨은 슬롯 m을 가질 때, 확률 1/m을 가지는 $0 \leq label \leq m$ 으로 할당하였다.[16]

4. 시뮬레이션 환경

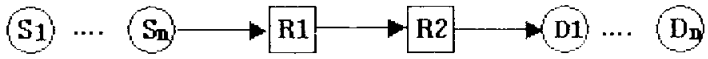
본 장에서는 시뮬레이션 모델과 파라미터에 대하여 설명한다. 시뮬레이션 툴로서는 버클리 대학에서 개발한 분산 객체 네트워크시뮬레터인 NS-2(Network Simulator)를 사용하였다.[13]

NS-2는 콜롬비아 대학에 의해 개발된 시뮬레이션 테스트베드인 NEST를 기반으로 UC Berkeley에서 1988년 REAL이라는 네트워크 시뮬레이션 도구를 개발하였다. 그리고 1989년에 NBNL(Lawrence Berkeley National Laboratory)의 네트워크 연구그룹은 이 REAL을 기반으로 연구를 시작하여 NS-1이라고 불리는 시뮬레이션 도구를 개발하였다. NBNL에 의해 개발된 NS-1은 확장된 tcl(Tool Command Language)을 시뮬레이션 기술 언어로 사용하며, 수행하고자 하는 시뮬레이션은 tcl 프로그램으로써 정의되게 된다. 이 tcl 대신에 MIT에 의해 개발된 otcl(Object tcl)을 사용하여 새로운 구조를 갖도록 NS-1을 개선한 것이 NS-2이다.

4.1 시뮬레이션 망 모델

모의실험 평가의 형평성을 가지기 위해, 기본 망 구성은 CSFQ에서 사용된 단일 혼잡망(Simple Congested Link) 모델과 다중 혼잡망(Multiple Congested Link) 모델을 사용하였다.

4.1.1 단일 혼잡망

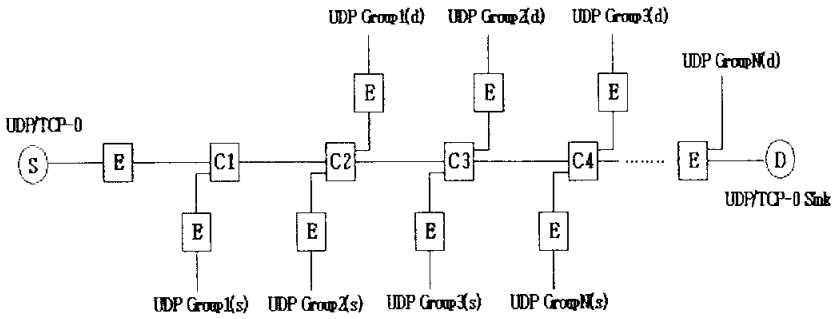


(그림 6) 링크용량 10Mbps, 지연 1ms 인 단일 혼잡망

(그림 6)은 모의 실험 망으로 단일 혼잡망에서 32개의 데이터 흐름을 가정한다. 이 망의 첫 번째 실험은 데이터 흐름 전부가 32개의 UDP는 데이터만을 가지고 실험하였다.

이 망의 두 번째 실험은 하나의 TCP 데이터를 가진 흐름과 31 개의 UDP 데이터 흐름을 가진다고 가정할 때의 모의 실험이다. UDP는 ack 메시지가 없다. 그러나 TCP는 반응적으로 목적지에 도착하면 ack 메시지가 송신지에 보내어 진다. 만약 혼잡이 발생되어 패킷이 손실되어 지면 slow-start, 빠른 복구 등 TCP의 혼잡 메커니즘이 동작한다. 그래서 병목 구간에서 버스티한 UDP 패킷들이 혼잡이 발생할 때, TCP도 영향을 받게 된다. 다시 말해서 TCP의 혼잡복구 동작이 UDP 데이터 흐름에도 영향을 미친다. 이러한 영향에 대한 실험이다.

4.1.2 다중 혼잡망



(그림 7) 링크용량 10Mbps, 각 노드간 지연 1ms 인 다중 혼잡망

(그림 7)은 GFC(Generic Fairness Configuration) 망으로 Stoica의 다중 혼합 망과 같이 구성하였다. 각 링크 10Mb/s 용량을 가지며, 지연 1ms로 혼잡 링크에 대한 실험이다. 여기서 S의 소스만 UDP 또는 TCP 데이터로 변경하며, UDP GroupN(s)는 그 다음 노드의 UDP GroupN(d)와 링크가 연결되어 실험하였다.

4.2 모의실험 파라미터

파라미터 값은 참고 문헌 [1]에서 사용되었던 값을 사용하였다.

CS-FNE와 비교되는 알고리즘으로는 DRR, CSFQ, RED, FRED가 있다. 여기서 WFQ의 변형된 형태인 DRR은 라운드 로빈 방식으로 흐름 단위 당 처리되는 큐잉 기법으로 공정한

대역 할당 표현에 있어, 아주 이상적인 공정성을 가지나 하나의 벤치마크로써 존재한다[10].

(표 1) 모의실험 파라미터

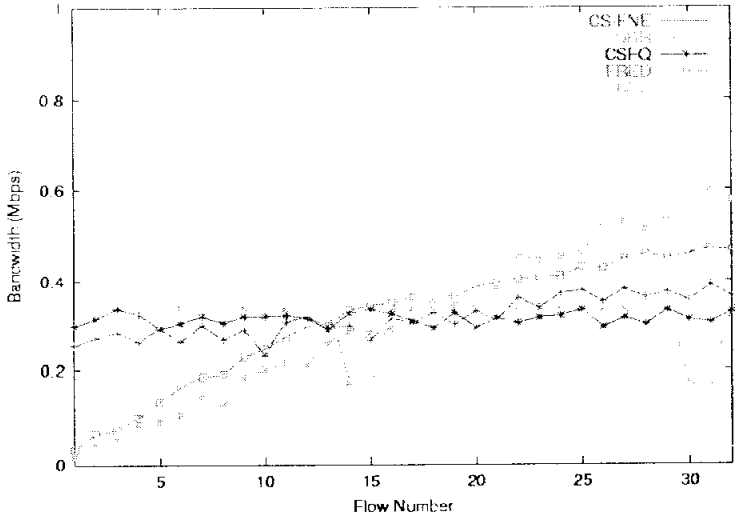
파라미터	값
버퍼크기	64000 bytes
패킷크기	1000 bytes
상수 K, K_a	각각 100 ms
RED, FRED 임계치	MIN : 16000 bytes MAX : 32000 bytes
지연시간	1 ms
링크용량	10 Mbps
TSW win_len	0.3 sec

5. 모의실험 결과분석

5.1 단일 혼잡망 결과분석

5.1.1 단일 혼잡망 첫 번째 모의실험 결과분석

(그림 8)에서 나타난 결과는 각 흐름 번호에 대한 평균 처리율이다. 여기서 DRR은 14, 15, 30, 31번 흐름에서 평균 공정 대역폭이 감소하는 현상을 보이기는 하나 이를 제외하고는 아주 이상적으로 나타났으며, 연구에서 제안한 CS-FNE는 흐름 번호가 증가함에 따라 평균 공정 대역폭도 증가하나 RED와 FRED 보다는 나은 결과를 나타내었으며, CSFQ보다는 평균 공정성에 차이가 있음을 보여 주고 있다.



(그림 8) 단일 혼잡망, 32개의 UDP 데이터흐름, packet size = 1000 bytes 일 때의 공정 대역폭

(표 2) 단일 혼잡망 첫 번째 모의실험 대역폭 비교

(소수점 네 번째 자리 반올림)

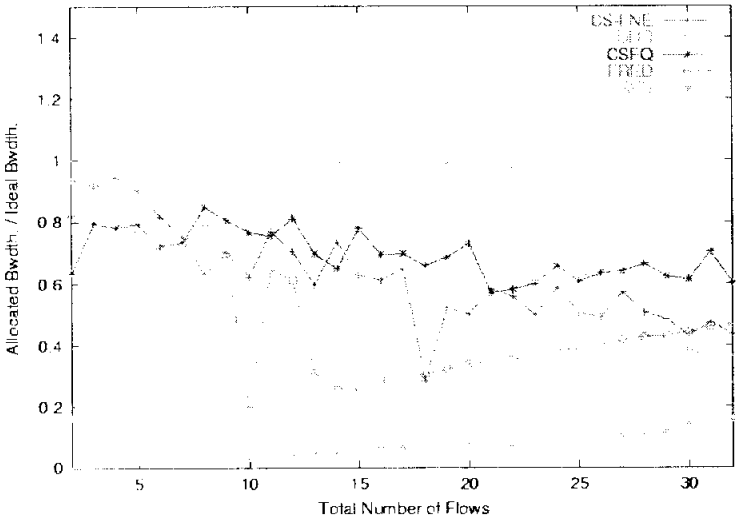
	CS-FNE	DRR	CSFQ	FRED	RED
평균 대역폭	0.312	0.312	0.310	0.312	0.312
최대값 -최소값	0.162	0.176	0.050	0.438	0.582
표준편차	0.046	0.056	0.015	0.132	0.176

(표2)에서 모든 알고리즘의 평균 공정성은 유사하나, CSFQ 알고리즘이 최대값, 최소값 간의 차이와 표준편차에서 가장 좋은 결과 값을 보인다. DRR은 14, 15, 30, 31번 흐름에서 평균 공정 대역폭이 감소하는 현상을 제외하고 계산하면 평균 대역폭은 0.333, 최대값, 최소값 간의 차이는 0.028, 표준편차는 0.005로 가장 이상적인 공정 대역폭을 갖는다. CS-FNE는 표준편차는 좋으나, 최대값, 최소값 간의 차이가 크다.

5.1.2 단일 혼잡망 두 번째 모의실험 결과분석

(그림 9)는 하나의 TCP 데이터를 가진 흐름과 31 개의 UDP 데이터 흐름을 가진다고 가정할 때의 모의 실험이다. 제시된 CS-FNE는 공정 대역폭에 대해서 CSFQ보다 진동이 폭이 많으나 흐름 번호가 커졌을 때 RED, FRED 보다 나은 결과를 나타내고 있다. DRR은 성능이 우수하게 나타났으나, 데이터 흐름 전부가 UDP 일 때의 경우와 다르게 흐름 번호 22

부터 성능이 저하되는데, 그 이유는 TCP와 버퍼를 공유하여 영향을 받았기 때문이다.



(그림 9) 단일 혼잡망, 하나의 TCP 데이터 흐름과 31개의 UDP 데이터흐름, packet size = 1000 bytes 일 때의 공정 대역폭

(표 3) 단일 혼잡망 두 번째 모의실험 대역폭 비교

(소수점 네 번째 자리 반올림)

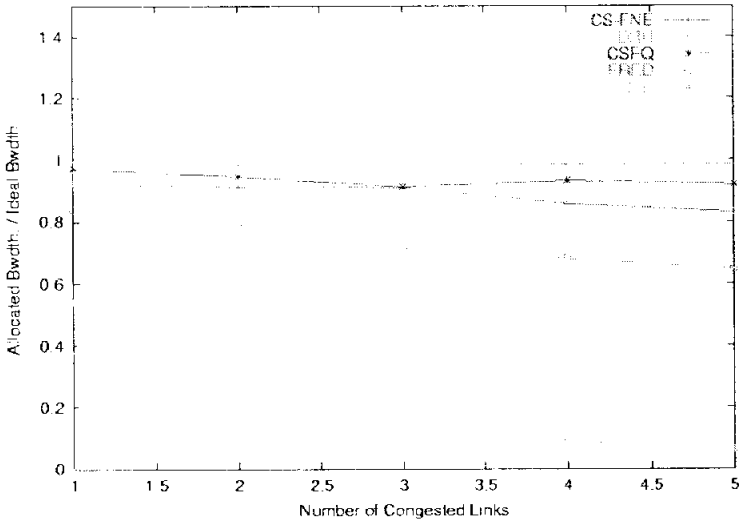
	CS-FNE	DRR	CSFQ	FRED	RED
평균 대역폭	0.634	0.850	0.703	0.339	0.101
최대값 -최소값	0.690	0.704	0.432	0.920	0.078
표준편차	0.171	0.248	0.093	0.331	0.042

(표3)에서 DRR은 TCP와의 버퍼 공유로 인해 22번 흐름부터 성능이 저하되어 최대값, 최소값의 차이가 크고, 표준 편차도 크지만 평균 공정 대역폭은 다른 알고리즘에 비해 좋은 결과가 나왔다. CSFQ는 표준편차에서 좋은 결과가 나왔다. 이는 CSFQ가 각 흐름에 대해 공정한 대역폭을 할당하고 있다는 것을 알 수 있다. RED는 최대값, 최소값 간의 차이와 표준편차에서 가장 좋은 값이 나왔으나, 다른 알고리즘에 비해 공정 대역폭 할당에 있어 효율적이지 않다는 것을 알 수 있다.

5.2 다중 혼잡망 결과분석

5.2.1 다중 혼잡망 첫 번째 모의실험 결과분석

(그림 10)은 모두 UDP 그룹일 때의 다중 혼잡망에 대한 모의 실험이다. 여기서 맨 좌측 경계 노드의 소스만 제외한 모든 ingress 경계 노드는 하나의 그룹에 10개 씩의 UDP소스가 있다고 가정한 결과이다. (그림 10)의 결과에서 DRR은 아주 이상적인 결과 값을 나타내고 있다. 그리고 CS-FNE는 CSFQ와 미세한 차이를 나타내고 있으며, FRED도 전체의 데이터가 UDP 일 경우에 단일 혼잡망에서의 성능보다 나은 결과를 볼 수 있다. RED는 링크 혼잡의 수가 증가할수록 성능이 저하를 볼 수 있다.



(그림 10) 다중 혼잡망, 한 그룹당 10씩의 개의 UDP 데이터흐름, S = UDP, packet size = 1000 bytes 일 때의 공정 대역폭

(표 4) 다중 혼잡망 첫 번째 모의실험 대역폭 비교

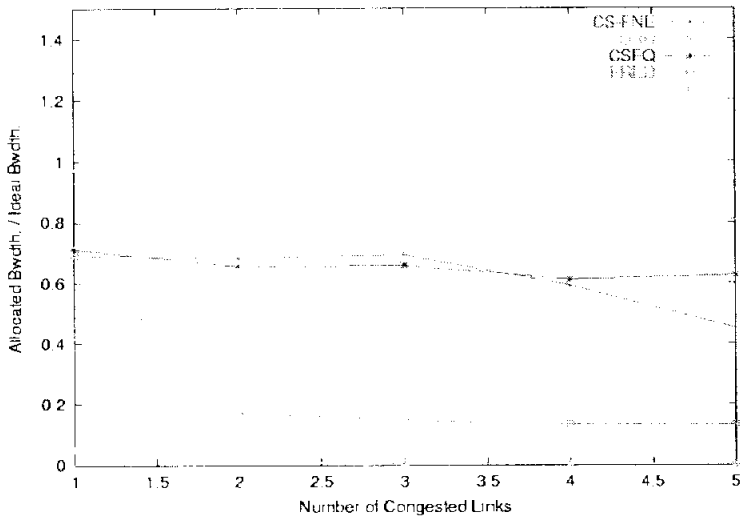
(소수점 네 번째 자리 반올림)

	CS-FNE	DRR	CSFQ	FRED	RED
평균 대역폭	0.889	0.986	0.936	0.732	0.233
최대값 -최소값	0.088	0.008	0.058	0.192	0.476
표준편차	0.038	0.003	0.023	0.078	0.195

(표4)에서 DRR은 모든 면에서 가장 좋은 결과 값을 보여주고 있다. RED를 제외한 알고리즘은 전체적으로 단일 혼잡망에서의 실험보다 좋은 결과 값을 보여주고 있다.

5.2.2 다중 혼잡망 두 번째 모의실험 결과분석

(그림 11)은 맨 좌측 경계 노드의 소스를 TCP로 설정하고, 10개 씩의 UDP소스를 하나의 그룹으로 하였다. (그림 11)의 결과에서 CS-FNE는 CSFQ보다 혼잡 링크 2와 3.5 사이의 공정 대역폭이 나은 결과를 보이고 있다. 그러나 링크 혼잡이 5일 경우 감소하는 것을 볼 수 있다. (그림 11)에서 FRED의 결과는 (그림 10)의 결과와 현격하게 차이를 나타나고 있다.



(그림 11) 다중 혼잡망, 한 그룹당 10씩의 개의 UDP 데이터흐름, S = TCP, packet size = 1000 bytes 일 때의 공정 대역폭

(표 5) 다중 혼잡망 두 번째 모의실험 대역폭 비교

(소수점 네 번째 자리 반올림)

	CS-FNE	DRR	CSFQ	FRED	RED
평균 대역폭	0.621	0.895	0.651	0.143	0.013
최대값 -최소값	0.241	0.165	0.099	0.689	0.048
표준편차	0.104	0.062	0.038	0.306	0.023

(표5)에서 RED는 평균 공정 대역폭에서 현격한 성능 저하를 보이고 있으며, FRED는 혼잡링크 2번부터 성능 저하의 발생으로 좋지 않은 결과를 보이고 있다. DRR, CS-FNE, CSFQ는 다소 성능의 저하가 있기는 하나 비교적 좋은 결과를 보이고 있다.

6. 결론

FQ, WFQ, FRED 등 많은 흐름 단위 스케줄링 알고리즘들이 전송률과 지연의 보장을 위해 제안되어 왔었으며, 혼잡제어를 위해 설계되었다. 그러나 흐름 단위 처리를 기본으로 하는 알고리즘들은 율 상태, 버퍼 관리, 패킷 스케줄링을 필요로 하기 때문에, 구현이 복잡하고 데이터 처리에 대한 속도도 늦을 뿐만 아니라, 모든 노드가 패킷의 분류와 흐름마다 혼잡제어를 하기 때문에 고속 망에서는 적합하지 않은 문제점이 있다.

본 연구는 이러한 복잡성을 줄이기 위해, FNE의 큐잉 기법을 Core-Stateless에 적용하였다. 이를 CS-FNE라고 하였으며, CSFQ, FRED, RED, DRR과 같은 알고리즘을 함께 평가하였다. 하나의 혼잡 공유 망에서는 RED, FRED보다 나은 결과를 나타내었으며, 다중 혼잡 망에서도 RED, FRED 보다 나은 결과를 나타내었다. 그러나 이상적인 공정성에 대한 결과는 산출되지 않았으나, fare_rate를 지수 분포로 계산하지 않아 단순히 구현할 수 있었다. 그래서 고속망에서 계산의 복잡성을 줄이고, 하드웨어에 쉽게 구현이 될 수 있다는데 의의가 있다.

향후 과제로는 공정성에 대한 결과를 더욱 향상시킬 필요가 있으며, TCP 데이터 혼잡제어에 대해서도 적응적으로 처리할 수 있도록 개선할 필요가 있다.

참 고 문 헌

[1] Ion Stoica, Scott Shenker, and Hui Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair BAndwidth Allocation in High Speed Networks," in Proceeding of SIGCOMM'98, Oct, 1997

[2] A. Demers, S. Keshav, and S. Shenker ,"Analysis and simulation of a fair queueing algorithm," J. Internetw. Res. Experience, pp. 3-26, Oct 1990

[3] Parekh, A. A generalized processor sharing approach to flow control

[4] D. Lin andR. Morris, "Dynamics of random early detection," in Proc. ACM SIGCOMM, Cannes, France, Oct. 1997, pp. 1427-137

[5] Z. Cao, Z Wang, E. Zegura, "Rainbow fair queueing: fair bandwidth sharing without per-flow state," Proceedings INFOCOM. March 2000, pp. 922-931

[6] D .D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," IEEE Trans (1998), 362-373.

[7] L Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," in Proc . ACM, SIGCOMM 90, 1990, pp. 19-29.

[8] S. Floyd and V. Jacobson, "Random early detection for congestion avoidance," IEEE/ACM Trans. Networking, vol. 1, pp. 397-413, July 1993.

[9] NS simulator, available from http://www.isi.edu/nsnam/ns.

[10] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," IEEE/ACM Trans. Networking, pp. 375-385, June 1996.

[11] "QoS Features for Voice over IP," Cisco Quality of Service for Voice over IP Solution Guide Chapter2

[12] http://www.cisco.com/networkers/nw00/pres/

[13] The VINT Project, "ns Notes and Documentation," February 25, 2000.

[14] 김화숙, 김상하, 김영부, "대역 공평성 보장을 위한 Core-Stateless 기법 연구", 한국통신학회 논문지, Vol.28, No.4C, 2003. 04.

[15] 유인태, 홍인기, 서덕영, "실시간 인터넷 서비스를 위한 상태 의존 RED 및 동적 스케줄링 기법에 관한 연구", 한국통신학회 논문지, Vol.28, No.9B, 2003. 09.

[16] 서경현, 김문경, 육동철, 박승섭, "Core-Stateless망에서의 공정한 대역폭 할당에 관한 연구", 제11회 한국정보과학회 영남지부 학술발표대회, pp.53-59, 2003. 12.

감사의 글

길다면 길었던 2년 6개월이 어느덧 지나갔습니다. 그 동안 많은 지도와 조언을 아끼지 않으시고 부족한 면을 채워주셨던, 박승섭 교수님께 진심으로 감사를 드립니다. 또한 귀중한 시간을 할애하여 논문이 완성되기까지 도와주신 박홍복, 정연호 교수님께도 감사를 드리며, 지금은 고인이 되셨지만 많은 가르침을 주신 박지환 교수님께도 감사를 드리며, 여러 가지 면에서 지도해주신 성순호, 여정모, 윤성대, 이경현, 김창수, 박만곤, 김영봉, 신상욱 교수님들께도 감사드립니다.

불철주야 연구에 매진하고 동고동락을 같이하며 제게 항상 힘이 되어 주시고 많은 도움을 주셨던 육동철 선배님, 연구실 생활동안 많은 도움을 주셨던 전홍진, 고영웅, 김영하 선배님, 모든 일에 열심히인 김문정 후배님, 교직생활을 하시고 계시는 주영선배님과 영재형, 미국에서 일하고 계시는 동렬이 형께 감사드립니다. 입학동기이며 이번에 같이 졸업논문을 발표한 하미숙 선생님, 졸업논문을 준비중이신 김도광, 김정숙 선생님, 정동균 선배님, 이미 졸업하신 박희대 계장님, 오명석, 김명희 선배님, 이규남, 문병윤, 이행남, 공옥춘, 홍승희 선생님께 감사드립니다. 같은 연구실 산업대학원 후배님들인 김승수, 김현근, 노태열 선생님, 교육대학원 후배님들인 강경민, 노대종, 윤성희, 장호상 선생님, 열심히 노력하고 있는 상영이 형, 오래 같이 있지는 못했지만 따뜻이 대해준 도기, 정섭이에게 감사를 드립니다. 연구실 생활동안 잘 따라 준 후배들인 기석이, 윤석이, 승주, 현정이, 형기, 형두, 정원이, 소민이, 기현이, 영주에게 고마운 마음을 전합니다.

연구실은 달랐지만 언제나 저에게 힘이 되었던, 황순환, 박현호, 김대업, 이찬희, 박상일 선배님들과 종민이형, 진현씨, 진호씨에게 감사드립니다.

또, 힘이 들 때 항상 같이 해 주고 위로해 준 친구들인 강산이, 정훈이, 중석이, 부곤이, 윤권이, 수현이, 성철이, 승규, 현우, 채길이, 창수에게 감사드립니다.

항상 저를 믿고 지원해 주신 가족, 할아버님, 할머님, 아버님, 어머님, 큰 고모부님, 큰 고모님, 작은 고모부님, 작은 고모님, 숙부님, 숙모님께 감사드리며, 집의 맏이로써 항상 모범을 보여야 하는데 그렇지 못한 저를 믿고 따라준 동생들, 주현이, 상윤이, 선화, 정묘, 영조, 영철이에게 고마운 마음을 전합니다.

비록 다 나열할 수는 없었지만, 저를 아는 모든 분들께 감사드리며, 현실에 안주하지 않고 앞으로 더욱 더 노력하는 사람이 되겠습니다.