論文2002-39TC-10-1

잉여 대역폭 소비 큐를 이용한 잉여 대역폭 페어 큐잉

(Excess Bandwidth Fair Queueing Using Excess Bandwidth Consumer Queue)

秋 皓 喆 * , 金 永 翰 *

(Ho Cheol Choo and Young Han Kim)

요 약

인터넷에서 서비스 품질을 제공하기 위해 대역폭에 대한 스케쥴링 기술은 중요한 요소 중 하나로서 많은 알고리즘이 개발되었다. 그러나 기존의 스케쥴링 알고리즘은 잉여 대역폭 분배에 있어 융통성을 제공하고 있지 않다. 이를 보완하여 잉여 대역폭 분배에 융통성을 제공하기 위해 DGPS(decoupled generalized processor sharing)가 제안되었지만[1] 구현이 복잡하고 기존의 다양한 알고리즘에 쉽게 적용하기에는 어려움이 따랐다. 본 논문에서는 잉여 대역폭 분배의 융통성을 제공하고 동시에 DGPS의 문제점을 개선하여 기존의 기반 알고리즘에 자연스럽게 적용할 수 있는 스케쥴링 알고리즘을 제안하고 공평성을 분석하였다. 또한 시뮬레이션을 통해 성능을 검증하였다.

Abstract

Scheduling technology is one of the most important elements required to support the QoS(quality of service) in the Internet and a lot of scheduling algorithms have been developed. However, most of there algorithms are not flexible in the distribution of the excess bandwidth. In order to improve the weakness of existing algorithms, DGPS(decoupled generalized processor sharing) has suggested recently. But, the DGPS algorithm is complex to implement and difficult to apply to the existing algorithms. In this paper, we propose a scheduling algorithm for distribution of the excess bandwidth which improves the implementation complexity of the DGPS and easy to be applied to ordinary algorithms.

Keywords : 페어큐잉, 스케듈링, 인터넷 QoS, 잉여대역폭

Ⅰ. 서 론

스케쥴링 기술은 인터넷에서 서비스의 질을 보장하기 위해서 필요한 요소 기술 중에 하나로써 많은 알고리즘들이 제안되었다[2-12]. 이들 대부분의 알고리즘들은 최소 예약 대역폭을 보장하는데 중점을 두었고 잉여 대역폭의 효율적 분배에 대한 고려는 없었다. 즉 기존 방식에서는 전체 대역폭 중 최소 요구 대역폭을 할당하고 남은 잉여 대역폭의 분배는 각 세션에서 요청한 최소 대역폭에 비례적으로 다시 각 세션에 분배된다. 그러나 이러한 알고리즘은 다양한 요구와 특성을 갖는 트래픽의 QoS를 모두 만족시키기에는 한계가 있다. 예를 들어, 파일 전송등과 같은 응용의 트래픽 특성은 최대한 많은 대역폭을 차지하기를 원하고, 그럴수록 대역폭의 효율성

* 正會員, 崇實大學校 情報通信電子工學部
(School of Electronic Engineering, Soongsil University)

※ 본 논문은 한국과학재단 목적기초연구지원(98-0102-10-01-3)의 결과임.

接受日字:2002年3月13日, 수정완료일:2002年9月16日

이 좋아진다. 그러나 음성과 같은 트래픽은 일정 대역폭 이상도 이하도 필요로 하지 않는다. 이처럼 다양한 트래픽의 요구를 만족시키기 위해서는 기본적인 요구 대역폭을 보장하고, 이와 더불어 잉여 대역폭 발생 시 트래픽의 특성에 따라 효율적으로 분배함으로써, 해당 트래픽의 요구를 최대한 만족시켜줄 수 있는 방식이 요구되었고, 이러한 방식으로서 최근 DGPS가 제안되었다[1]. 그러나 fluid 모델 DGPS에서는 잉여 대역폭을 융통성 있게 분배하기 위해 현재 시스템에 backlog되어 있는 세션들의 추적이 필요하므로 상당한 복잡도를 갖는다. 이러한 복잡도를 줄이기 위한 DGPS에서는 근사화된 알고리즘을 사용하여 각 세션들을 추적해야 하는 복잡도를 없앴지만 새로운 latency의 문제가 생겨났다. 따라서 백본에서와 같이 많은 수의 패킷이 입출력되는 환경에서는 시스템의 성능이 저하될 수 있다는 단점이 있다.

본 논문에서는 DGPS의 문제점을 개선하고, 또한 기존의 스케줄링 알고리즘에 쉽게 적용할 수 있는 EBFQ (excess bandwidth fair queueing) 알고리즘을 제안한다.

서론에 이어 제Ⅱ장에서는 GPS(generalized processor sharing)[2]를 기반으로 한 기존의 스케줄링 알고리즘의 잉여 대역폭 분배 특성을 살펴보고, 이어서 제Ⅲ장에서는 EBFQ(excess bandwidth fair queueing)와 DGPS의 잉여 대역폭 분배 특성을 살펴본다. 그리고 제Ⅳ장에서는 DGPS에서 제안한 근사화 방법의 문제점을 살펴보고, 이를 개선한 EBFQ의 구조와 동작을 살펴본다. 제Ⅴ장에서는 지연과 공평성 측면에서 EBFQ의 성능을 분석하고 제Ⅵ장에서는 시뮬레이션을 통해 EBFQ의 최소 대역폭 및 잉여 대역폭 보장 성능을 검증한다. 마지막으로 제Ⅶ장에서 결론을 맺는다.

Ⅱ. 기존 스케줄링 알고리즘의 잉여 대역폭 분배 특성

본 장에서는 많은 스케줄링 알고리즘의 기반인 GPS의 잉여 대역폭 분배 특성을 살펴본다. 이에 앞서 분배 특성을 살펴보는데 필요한 몇 가지 수식 및 기호를 정의한다. GPS 시스템의 출력 링크의 용량을 C라고 할 때, 각 세션은 최소 요구 대역폭, r_k 를 가지고 자신의 최소 서비스 대역폭을 예약하고, 임의 순간에 세션 k가 서비스 받는 대역폭을 f_k 라 한다. 이때 r_k , f_k 는 각각 다음을 만족해야 한다.

$$r_k \le \sum_{j=1}^{N} r_j = C, \quad f_k \le \sum_{j=1}^{N} f_j = C \tag{1}$$

또한 $A_k(t)$는 시간 (0, t) 동안 세션 k에 도착한 트래픽의 총량이고, $W_k(t)$는 시간 (0, t) 동안 서버에 의해 서비스된 세션 k의 트래픽 총량으로 정의한다. 시간 t에 세션 k의 backlog되어 있는 트래픽 양을 $Q_k(t)$, 그리고 normalized 서비스 양을 $w_k(t)$라 할 때 각각을 다음과 같이 정의한다.

$$Q_k(t) \equiv A_k(t) - W_k(t) \tag{2}$$

$$w_k(t) \equiv \frac{1}{r_k} W_k(t) \tag{3}$$

$$w_k(t_1, t_2) \equiv w_k(t_2) - w_k(t_1) \tag{4}$$

여기서 normalized 서비스는 각 세션이 서비스 받은 양을 일반화한 것이다. 따라서 각 세션의 normalized 서비스를 비교하면 각 세션이 받고 있는 서비스의 정도를 알 수 있다. 이상적인 GPS 시스템에서 임의의 두 세션의 normalized 서비스 양을 비교하면 다음과 같다.

$$w_k(t_1, t_2) = w_j(t_1, t_2), \quad k, j \in B(t_1, t_2) \tag{5}$$

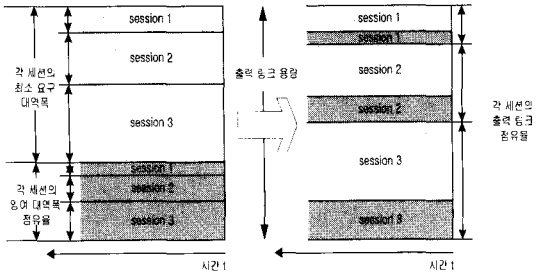
여기서 $B(t_1, t_2)$는 시간 (t_1, t_2) 동안 계속해서 backlog되어 있는 세션들의 집합을 나타낸다. 위의 식(5)가 idle 세션들에 의해 생긴 잉여 대역폭이 있는 상황하에서도 성립하기 위해서는 그 잉여 대역폭이 각 세션의 최소 요구 대역폭에 비례적으로 분배되어야한다. 그러므로 GPS에서 임의 순간 τ 에 각 세션에 제공되는 서비스 대역폭 f_k 는 다음과 같다.

$$f_k(\tau) = r_k + \frac{r_k}{\sum_{j \in B(\tau)} r_j} \left(C - \sum_{j \in B(\tau)} r_j \right) \tag{6}$$

식(6)은 GPS의 잉여 대역폭 분배 특성을 잘 나타내고 있다. 그림 1은 GPS의 잉여 대역폭 분배 특성을 나타낸 그림이다.

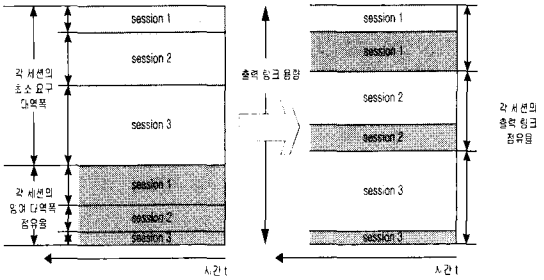
식(6)과 그림 1에서 나타난 바와 같이 GPS의 잉여 대역폭 분배 특성은 현재 backlog되어있는 각 세션들의 최소 요구 대역폭의 비율로 분배되는 것을 알 수 있다.

GPS를 기반으로 한 기존의 스케줄링 알고리즘은 GPS를 근사화한 알고리즘들이므로 잉여 대역폭 분배 특성에 있어서도 역시 GPS의 특성과 마찬가지로 현재



● 출력 링크 용량 : 1
● 최소 요구 대역폭 : 0.1(session 1), 0.2(session 2), 0.3(session 3)

그림 1. GPS의 잉여 대역폭 분배 특성
Fig. 1. Bandwidth distribution in GPS system.



● 출력 링크 용량 : 1
● 최소 요구 대역폭 : 0.1(session 1), 0.2(session 2), 0.3(session 3)
● 목표 대역폭 : 0.3(session 1), 0.2(session 2), 0.1(session 3)

그림 2. DGPS의 잉여 대역폭 분배 특성
Fig. 2. Bandwidth distribution in DGPS system.

backlog되어있는 세션들의 최소 요구 대역폭의 비율로 분배되는 특성을 갖는다.

Ⅲ. DGPS의 잉여 대역폭 분배 특성

앞장에서는 기존 스케쥴링 알고리즘의 잉여 대역폭 분배 특성을 살펴보았다. 본장에서는 잉여 대역폭 분배에 융통성을 제공하기 위해 제안된 DGPS의 잉여 대역폭 분배 특성을 살펴본다. DGPS는 최소 요구 대역폭 r_k와 별도로 잉여대역폭 분배를 위한 목표 대역폭 t_k를 사용하여 잉여 대역폭 분배에 융통성을 제공한다. DGPS에서는 idle 세션에 의해 생긴 잉여 대역폭을 최소 요구 대역폭의 비율로 분배하는 기존의 GPS 기반의 스케쥴링 알고리즘과는 달리 목표 대역폭의 비율에 따라 분배한다. 따라서 목표 대역폭의 적절한 설정으로 다양한 서비스의 품질을 요구하는 여러 트래픽에 맞추어 융통성있게 잉여 대역폭을 분배할 수 있다. 다음의 식은 DGPS의 임의 시간 τ에 각 세션에 제공되는 서비스 대역폭을 나타낸다.

$$f_k(\tau) = r_k + \frac{t_k}{\sum_{j \in B(\tau)} t_j}\left(C - \sum_{j \in B(\tau)} r_j\right) \qquad (7)$$

<그림 2>는 DGPS의 잉여 대역폭 분배 특성을 나타낸 그림이다.

식(7)과 <그림 2>에 나타낸 바와 같이 DGPS의 잉여 대역폭 분배 특성은 현재 backlog되어 있는 각 세션들의 목표 대역폭의 비율로 분배되는 것을 알 수 있다. 그러나 DGPS는 임의 순간의 서비스 대역폭을 계산하기 위해 현재 backlog되어있는 세션들의 목표 대역폭의 합과 최소 요구 대역폭의 합을 알아야 하고, 그러기 위해서는 매 계산 시마다 backlog되어 있는 세션들을 추적해야 한다. 이로 인해 DGPS에서 임의 순간의 서비스 대역폭을 계산하는 데는 O(N)의 복잡도가 따른다. 이 정도의 복잡도는 세션의 개수가 많은 고속 통신망 환경에 적용하기에는 어려움이 따른다. 제Ⅳ장에서는 DGPS에서 복잡도를 줄이기 위해 제안된 근사화 방법과 이의 문제점을 고찰하고, 이를 개선하여 제안하는 EBFQ 알고리즘의 구조와 동작을 살펴본다.

Ⅳ. EBFQ(excess bandwidth fair queueing)

1. DGPS에서 제안된 근사화 방법

앞에서 살펴본 DGPS의 문제점은 각 세션의 서비스 대역폭을 계산하는데 드는 복잡도가 O(N)로 크다는 것이다. 이러한 복잡도를 줄이기 위해서는 잉여 대역폭 측정을 위해 필요한 최소 요구 대역폭의 총합과 목표 대역폭의 총합을 구하는데 드는 복잡도를 줄여야 한다. DGPS에서는 목표 대역폭의 총합인 $T(\tau)$와 잉여 대역폭을 알기 위한 최소 요구 대역폭의 총합인 $R(\tau)$을 패킷이 HOL(head-of-line)에 들어오고 나가고 할 때마다 갱신하는 방법을 사용하여 임의 시간 τ 에서 각 세션의 서비스 대역폭을 계산하는데 드는 복잡도를 O(1)로 줄였다. 다음은 $T(\tau)$와 $R(\tau)$을 갱신하는 방법을 나타냈다.

경우 1) HOL에 패킷이 도착 시

$R(\tau)$ <-- $R(\tau)$ + r_k

$T(\tau)$ <-- $T(\tau)$ + t_k

경우 2) HOL의 패킷이 출력 링크로 출발 시

$$R(\tau) \leftarrow\!- R(\tau) - r_k$$

$$T(\tau) \leftarrow\!- T(\tau) - t_k$$

위의 방법을 사용하면 DGPS에서 서비스 대역폭을 계산하는 복잡도가 줄어들게 된다. 그러나 이 방법에서는 $T(\tau)$와 $R(\tau)$를 모든 세션들이 동시에 엑세스할 수 없으므로 동시에 여러 세션에 패킷이 도착했을 경우 특정 세션은 자신을 제외한 모든 세션들이 $T(\tau)$와 $R(\tau)$ 값을 갱신하고 난 후에 갱신을 할 수 있게 되므로 이에 따른 latency가 생기게 된다. 또한 임의의 시간 간격 동안 $T(\tau)$와 $R(\tau)$을 갱신하는데 드는 복잡도가 출력된 패킷 수에 비례하게 된다. 이러한 복잡도 역시 고속의 통신망에서는 오버헤드로 작용하므로 시스템의 성능을 저하시킬 수 있다. 다음절에서는 이러한 문제점을 개선한 EBFQ의 구조와 동작을 살펴본다.

2. EBFQ의 구조와 동작

다음 <그림 3>은 EBFQ의 구조를 나타낸다. EBFQ에서 각 세션은 최소 서비스 대역폭을 보장받기 위해 최소 요구 대역폭으로 계산된 서비스 태그 값을 사용하여 서비스 경쟁을 하고, 목표 대역폭의 비율로 잉여 대역폭의 분배를 보장받기 위해 목표 대역폭으로 계산된 서비스 태그 값을 사용하여 잉여 대역폭에 대한 서비스 경쟁을 한다. 또한 잉여 대역폭의 분배 시점을 결정하기 위해 가상 큐인 EBCQ(excess bandwidth consumer queue)를 둔다.

이 가상 큐는 실제 패킷이 큐잉되는 곳이 아니라, 현재 시스템에 남아있는 잉여 대역폭을 소비하는 가상의 큐이다. EBCQ는 현재 시스템에 남아 있는 잉여 대역폭이 자신에게 할당된 최소 요구 대역폭이라 가정하고, 그 대역폭으로 계산된 서비스 태그 값을 가지고 다른 큐와

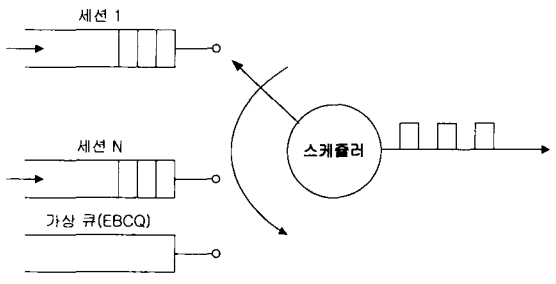


그림 3. EBFQ의 구조
Fig. 3. The structure of EBFQ.

서비스 경쟁에 들어간다. 그리고 EBCQ는 자신의 HOL에 그 태그 값을 가진 패킷이 있다고 가정하고, 최소 요구 대역폭으로 계산된 태그 값을 가진 각 세션과 서비스 경쟁을 한다. 평상시 스케줄러는 최소 요구 대역폭으로 계산된 태그 값 중에서 가장 작은 값을 가진 세션을 선택하여 그 세션의 HOL에 있는 패킷을 서비스해 준다. 만약 서비스 경쟁에서 다음에 서비스 받을 세션으로 가상 큐가 선택되면 각 세션의 목표 대역폭으로 계산된 서비스 태그 값 중에서 가장 작은 값을 가진 세션을 다음에 서비스 할 세션으로 선택하고 그 세션 큐의 HOL에 있는 패킷을 서비스 해 준다. EBFQ에서 가상 큐의 서비스 태그 값을 계산하기 위해서는 서비스 태그 값을 계산할 당시의 시스템의 잉여 대역폭을 알아야 한다. 그 시점의 잉여 대역폭을 알기 위해서 EBFQ에서는 다음의 방법을 사용하여 잉여 대역폭, 즉 가상 큐의 서비스 대역폭인 r_{EBCQ}을 갱신한다.

경우 1) 세션이 새롭게 backlog될 때

$$r_{EBCQ}(\tau) \leftarrow\!- r_{EBCQ}(\tau) - r_k$$

경우 2) 세션이 idle하게 될 때

$$r_{EBCQ}(\tau) \leftarrow\!- r_{EBCQ}(\tau) + r_k$$

여기서 r_{EBCQ}의 초기 값은 출력 링크의 대역폭 C 이고, 임의의 시간 간격 동안 잉여 대역폭 계산에 드는 복잡도는 O(1)로 DGPS에서의 복잡도 보다 작다. 그리고 DGPS와는 달리 목표 대역폭의 총합은 계산하지 않아도 된다. 따라서 DGPS보다 계산에 필요한 복잡도를 훨씬 줄였다. 또한 EBFQ에서 가상 큐를 제외한 각 세션은 자신의 서비스 대역폭을 계산하기 위해 목표 대역폭의 총합인 $T(\tau)$ 및 최소 요구 대역폭의 총합인 $R(\tau)$을 엑세스할 필요가 없게되고, 세션이 새롭게 backlog될 당시에 단지 가상 큐인 EBCQ에 자신의 최소 요구 대역폭을 알려 주기만 하면 되므로 DGPS에서 제기되었던 latency는 발생하지 않게 된다. 최악의 경우를 고려하면, 시스템에서 수용 가능한 모든 세션이 backlog 된 상태인 경우, DGPS는 스케줄러에 의해 매 패킷이 서비스 받기 위해 선택될 때마다 최소 요구 대역폭의 총합 및 목표 대역폭의 총합을 갱신하여야 한다. 그러나 EBFQ의 경우 세션이 idle하게 되지 않는 한 최소 요구 대역폭 및 목표 대역폭의 총합을 갱신하지 않는다. 이 경우에서도 볼 수 있듯이 EBFQ는 DGPS보다 계산에 드는

복잡도가 작음을 알 수 있다. 또한 EBFQ는 기존의 스케줄링 알고리즘에 잉여 대역폭 분배를 위한 가상 큐를 추가하고 이 큐가 선택될 경우에만 잉여 대역폭 분배를 위한 서비스가 개시되고 이외의 경우는 기존의 스케줄링 알고리즘의 서비스 방식을 그대로 따르므로 기존의 스케줄링 알고리즘에 가상 큐의 추가만으로 자연스럽게 EBFQ를 적용할 수 있다. 또한 공평성을 보장하는 스케줄링 알고리즘을 기본으로 사용한 EBFQ는 기본 알고리즘의 공평성 특징을 그대로 따르며, 독립화 특성 및 최대 지연 시간 역시 기본 알고리즘의 특성을 그대로 따르게 된다. 따라서 공평성을 보장하는 스케줄링 알고리즘을 기본 알고리즘으로 사용한 EBFQ도 역시 공평성을 보장하는 알고리즘이다.

다음 V장에서는 EBFQ를 SCFQ(self-clocked fail queueing)[3]에 적용했을 경우, delay와 공평성 측면에서 EBFQ의 성능을 분석한다.

V. SCFQ 기반 EBFQ의 지연과 공평성 분석

1. SCFQ 기반 EBFQ의 지연 분석

GPS를 기본으로 하는 많은 스케줄링 알고리즘은 GPS 시스템을 근사화하기 위해서 가상 시간 함수를 사용하여 GPS 시스템을 흉내낸다. 가상 시간 함수는 스케줄러에 의해 각 세션에 제공된 normalized 서비스 양을 나타낸다. idle 세션이 임의의 시간에 backlog될 때, 현재 서비스 받고 있는 다른 세션들과 공정하게 서비스를 받게 하기 위하여 현재 새롭게 backlog된 세션은 가상 시간을 기준으로 자신의 가상 종료 시간을 계산하게 된다. 즉 가상 시간은 새롭게 backlog되는 세션들과 현재의 세션들이 공정하게 서비스 경쟁을 할 수 있도록 하는 기준을 제공해 주는 도구이기도 하다. 이와 같이 가상 시간 함수를 이용하여 GPS 시스템을 근사화한 많은 스케줄링 알고리즘들이 제안되었다. 그 중 SCFQ는 식(8)과 같이 가상 시간을 바로 이전에 서비스 받은 패킷의 가상 종료 시간 값을 사용하여 가상 시간을 갱신하는데 드는 복잡도를 O(1)으로 줄임으로써 scability 측면의 성능을 개선하였다[3].

기반 스케줄링 알고리즘으로 SCFQ를 사용할 때의 EBFQ의 worst-case-delay bound와 공평성 분석을 용이하게 하기 위해 모든 패킷의 크기는 L로 같다고 가정한다. 다음은 분석에 사용될 몇 가지 기호들에 대한 정의이다.

$v(t)$: SCFQ의 가상 시간 함수
$v_k(t)$: 세션 k의 가상 시간 함수
p_k^i : 세션 k의 i번째 패킷
a_k^i : 세션 k의 i번째 패킷의 도착 시간
d_k^i : 세션 k의 i번째 패킷의 서비스 종료 시간
r_k : 세션 k의 최소 요구 대역폭
t_k : 세션 k의 목표 대역폭
$f_k(\tau)$: 세션 k의 임의 순간 τ에서의 서비스 대역폭

p_k^i가 τ 시간에 세션 k의 HOL에 도착했다고 할 때 p_k^i의 가상 종료 시간 값은 다음과 같다.

$$v(d_k^i) = \frac{L}{r_k} + \max(v(d_k^{i-1}), v(\tau)) \tag{8}$$

또한 여기서 $v(d_k^{i-1}) \le v(\tau)$이므로 $v(d_k^i)$은 식(9)가 된다.

$$v(d_k^i) = \frac{L}{r_k} + v(\tau) \tag{9}$$

시간 τ 시점에서 다른 세션들의 세션 가상 시간 값은 다음의 식을 만족한다.

$$v_j(\tau) \ge v(\tau) \tag{10}$$

여기서 세션 j가 시간 (τ, d_k^i) 동안 세션 k보다 먼저 서비스 받을 수 있는 양을 W_j 라고 한다면 W_j 는 다음과 같은 한계 값을 가진다.

$$v(d_k^i) - v_j(\tau) = \frac{W_j}{r_j} \tag{11}$$

식(10)을 식(9)에 적용하면

$$v(d_k^i) - v_j(\tau) \le \frac{L}{r_k} \tag{12}$$

식(12)가 되고, 식(11)을 식(12)에 적용하면 W_j 는 다음과 같은 조건을 만족하게 된다.

$$W_j \le \frac{L}{r_k} r_j,\ r_j \le C \tag{13}$$

여기서 최악의 상황을 고려하면 $r_j = C$인 경우이고 따라서 식(13)은 식(14)가 된다.

$$W_j \le \frac{L}{r_k} C \tag{14}$$

패킷 p_k^i의 실제 서비스 종료 시간은 각 세션들의 W_j를 모두 서비스하고 난 후 가 될 것이다. 따라서 p_k^i의 실제 서비스 종료 시간은 다음과 같은 한계값을 가진다.

$$d_k^i \le \tau + \frac{\sum_{j=1}^{N} W_j}{C} \tag{15}$$

식(15)에 식(14)를 적용하면 식(16)이 된다.

$$d_k^i \le \tau + \frac{\sum_{j=1}^{N} \frac{L}{r_k} C}{C} = \tau + N\frac{L}{r_k} \tag{16}$$

따라서 SCFQ 기반 EBFQ의 worst-case-delay bound는 다음과 같다.

$$d_k^i - \tau \le N\frac{L}{r_k} \tag{17}$$

식(17)은 순수한 SCFQ에서 패킷 크기가 L로 같다고 가정했을 때의 worst-case-delay bound와 같다. 즉 EBFQ는 기존의 알고리즘에 적용할 경우 기존 알고리즘의 특성을 변화시키지 않는다는 것을 알 수 있다.

2. SCFQ 기반 EBFQ의 공평성 분석

worst-case-delay bound 분석에서 사용한 가정을 SCFQ 기반 EBFQ를 사용할 경우의 공평성 분석에 그대로 사용한다.

먼저 임의의 시간에서 세션 k의 normailized 서비스 부족분, $\delta_k(t)$를 정의한다.

$$\delta_k(t) = v(t) - v_k(t) \tag{18}$$

여기서 $b_k^i = \max\{d_k^{k-1}, a_k^i\}$, $a_k^i \le b_k^i \le \tau < d_k^i$ 라 하면, $\delta_k(b_k^i)$는 항상 "0"이므로 식(18)은 식(19)가 된다.

$$\delta_k(\tau) = \delta_k(b_k^i) + \delta_k(b_k^i, \tau) = v(b_k^i, \tau) - v_k(b_k^i, \tau) \tag{19}$$

또한 시간 d_k^i에서 가상 시간 함수 값은 다음과 같다.

$$v(d_k^i) = \frac{L}{f_k} + v(b_k^i) \tag{20}$$

식(20)으로부터 식(21)이 아래와 같이 전개된다.

$$v(d_k^i, b_k^i) = v(d_k^i) - v(b_k^i) = \frac{L}{f_k} \tag{21}$$

그리고 $f_k = r_k + \alpha$이며, 또한 식(20), 식(21)에 의해 아래의 부등식이 유도된다.

$$0 \le v(b_k^i, \tau) \le v(b_k^i, d_k^i) \le \frac{L}{f_k} \le \frac{L}{r_k} \tag{22}$$

또한 세션의 가상 시간 함수에 대한 부등식은 다음과 같다.

$$0 \le v_k(b_k^i, \tau) \le v_k(b_k^i, d_k^i) = w_k(b_k^i, d_k^i) \le \frac{L}{r_k} \tag{23}$$

식(22)와 (23)을 통해 임의 시간에 세션 k의 normalized 서비스 부족분에 대한 부등식이 다음과 같이 구해진다.

$$0 \le \delta_k(\tau) \le \frac{L}{r_k} \tag{24}$$

따라서 임의의 시간 간격 $[t_1, t_2]$동안 세션 k의 normalized 서비스 부족분은 다음의 한계값을 가진다.

$$|\delta_k(t_1, t_2)| \le \frac{L}{r_k} \tag{25}$$

그리고 임의의 시간 간격 $[t_1, t_2]$동안 backlog되어있는 임의의 두 세션 간의 normalized 서비스 부족분의 차이는 아래와 같은 부등식을 만족한다.

$$|\delta_k(t_1, t_2) - \delta_j(t_1, t_2)| \le L(\frac{1}{r_k} + \frac{1}{r_j}) \tag{26}$$

두 세션간의 normalized 서비스 부족분을 나타낸 위의 부등식은 공평성 측면의 특성을 보여주는 식이다. SCFQ 기반 EBFQ의 공평성 특성을 보여주는 위의 식은 패킷의 크기가 모두 L로 같다고 가정하였을 경우의 SCFQ 공평성 특성과 같다.

VI. 시뮬레이션

본 장에서는 순수한 SCFQ 알고리즘과 SCFQ를 기반 알고리즘으로 사용한 EBFQ 알고리즘의 최소 대역폭 보장 및 잉여 대역폭 분배 성능을 비교 분석한다. 각 시뮬

레이션에서 사용된 플로우는 모두 CBR(constant bit rate)이고 각 플로우의 패킷 크기는 모두 53바이트이며 출력 링크의 용량은 1Mbps이다.

첫 번째 시뮬레이션에서는 <표 1>에 있는 두 플로우를 사용한다. <표 1>은 각 플로우의 최소 예약 대역폭 및 목표 대역폭, 그리고 각 플로우의 트래픽 발생 속도를 나타내었고, 예상되는 출력 링크의 점유 대역폭을 나타내었다. 다음은 식(6)과 식(7)을 이용하여 순수한 SCFQ와 EBFQ에 SCFQ를 적용한 경우에 대한 각 플로우의 출력 링크 점유 대역폭(f_k)을 계산한 식이다.

$$f_0 = r_0 + \frac{r_0}{\sum_{j\in B(\tau)} r_j}(C - \sum_{j\in B(\tau)} r_j)$$

$$= 0.1 + \frac{0.1}{0.1+0.4} \times 0.5 = 0.2Mbps \quad \text{(순수한 SCFQ)}$$

$$f_1 = r_1 + \frac{r_1}{\sum_{j\in B(\tau)} r_j}(C - \sum_{j\in B(\tau)} r_j)$$

$$= 0.4 + \frac{0.4}{0.1+0.4} \times 0.5 = 0.8Mbps \quad \text{(순수한 SCFQ)}$$

$$f_0 = r_0 + \frac{t_0}{\sum_{j\in B(\tau)} t_j}(C - \sum_{j\in B(\tau)} r_j)$$

$$= 0.1 + \frac{0.75}{0.75} \times 0.5 = 0.6Mbps \quad \text{(SCFQ 기반의 EBFQ)}$$

$$f_1 = r_1 + \frac{t_1}{\sum_{j\in B(\tau)} t_j}(C - \sum_{j\in B(\tau)} r_j)$$

$$= 0.4 + \frac{0.0}{0.75} \times 0.5 = 0.4Mbps \quad \text{(SCFQ 기반의 EBFQ)}$$

<그림 4>는 첫 번째 시뮬레이션에 대한 결과이다. 그림의 x축은 출력 링크의 점유 대역폭을 나타내며 y축은 시뮬레이션 시간을 나타낸다. 그림에서 실선은 각 플로우가 점유하는 대역폭의 총합을 나타내며, 짧은 점선은 플로우 0를, 긴 점선은 플로우 1을 나타낸다. 시뮬레이션 결과에 나타난 각 플로우의 점유 대역폭이 계산된 점유 대역폭 값과 같음을 볼 수 있다. 또한 EBFQ의 경우 플로우 1의 목표 대역폭을 설정하지 않았으므로 잉여 대역폭은 목표 대역폭이 설정된 플로우 0에만 할당되는 것을 볼 수 있다.

두 번째 시뮬레이션의 구성도는 첫 번째 시뮬레이션 구성도와 동일하고 <표 2>에 나타낸 플로우를 사용한다. 이전 시뮬레이션과의 차이는 플로우 1에 목표 대역폭을 0.25Mbps로 할당했다는 것이다. <그림 5>의 시뮬레이션 결과를 보면 표에서 계산되어진 예상 점유 대역폭과 같다는 것을 알 수 있다.

표 1. 플로우 별 예상 출력 링크 점유 대역폭 (시뮬레이션 1)

Table 1. Minimum reserved bandwidth, target rate, and arrival rate of each flow.

플로우	트래픽 발생속도	최소 예약 대역폭	목표 대역폭		예상 출력 링크 점유 대역폭	
			SCFQ	EBFQ	SCFQ	EBFQ
0	0.85Mbps	0.1Mbps	-	0.75Mbps	0.2Mbps	0.6Mbps
1	0.85Mbps	0.4Mbps	-	0.00Mbps	0.8Mbps	0.4Mbps

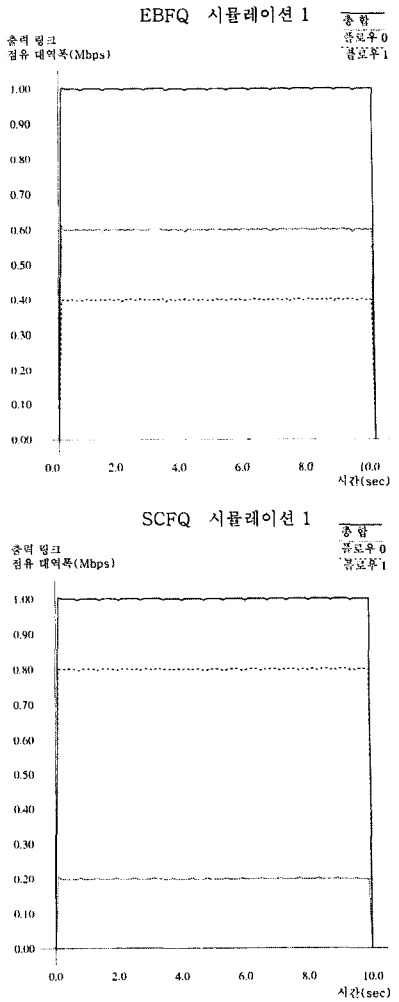


그림 4. EBFQ와 SCFQ의 시뮬레이션 1의 결과

Fig. 4. Result of first simulation.

세 번째 시뮬레이션에서는 <표 3>에 나타낸 플로우를 사용하였고, 각 플로우의 트래픽 발생 시작 시간을 0초, 2초, 4초로 각각 다르게 했다. <그림 6>은 세 번째 시뮬레

표 2. 플로우 별 예상 출력 링크 점유 대역폭 (시뮬레이션 2)

Table 2. Minimum reserved bandwidth, target rate, and arrival rate of each flow.

플로우	트래픽 발생속도	최소 예약 대역폭	목표 대역폭		예상 출력 링크 점유 대역폭	
			SCFQ	EBFQ	SCFQ	EBFQ
0	0.85Mbps	0.1Mbps	-	0.75Mbps	0.2Mbps	0.475Mbps
1	0.85Mbps	0.4Mbps	-	0.25Mbps	0.8Mbps	0.525Mbps

표 3. 플로우 별 예상 출력 링크 점유 대역폭 (시뮬레이션 3)

Table 3. Minimum reserved bandwidth, target rate, and arrival rate of each flow.

플로우	트래픽 발생속도	최소 예약 대역폭	목표 대역폭	
			SCFQ	EBFQ
0	0.85Mbps	0.4Mbps	-	0.0Mbps
1	0.53Mbps	0.1Mbps	-	0.1Mbps
2	0.53Mbps	0.1Mbps	-	0.2Mbps

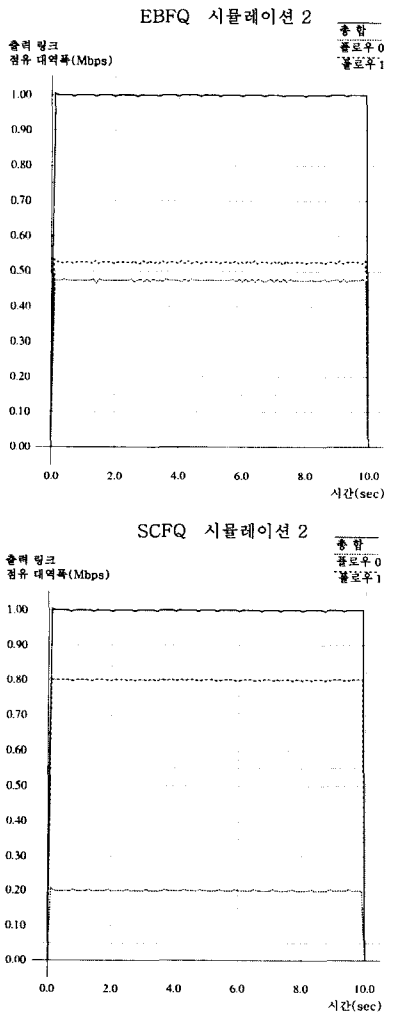


그림 5. EBFQ와 SCFQ의 시뮬레이션 2의 결과

Fig. 5. Result of second simulation.

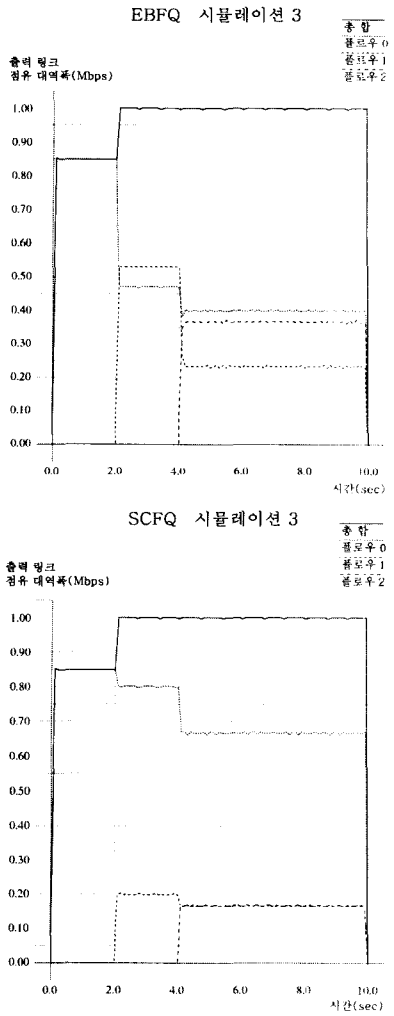


그림 6. EBFQ와 SCFQ의 시뮬레이션 3의 결과

Fig. 6. Simulation results in EBFQ and SCFQ.

이션 결과를 나타낸다. 그림에서 0초에서 2초 사이에는 플로우 0만 트래픽을 발생시키므로 출력 링크가 허락하는 한도에서 플로우 0의 트래픽이 손실없이 모두 서비스되는 것을 볼 수 있다. 그러나 2초에 플로우 1의 트래픽 발생이 시작되므로 두 플로우는 자신의 최소 예약 대역폭을 할당받고, 잉여 대역폭은 목표 대역폭의 비율로 서비스 받는다. 여기서 플로우 0의 목표 대역폭은 설정되어있지 않으므로 잉여 대역폭은 모두 플로우 1에게

정되어있지 않으므로 잉여 대역폭은 모두 플로우 1에게 할당된다. 하지만 플로우 1의 트래픽을 모두 서비스하고도 잉여 대역폭이 남으므로 그 양은 플로우 0에게 할당된다. 4초에 플로우 2의 트래픽 발생이 시작되면 대역폭 점유 양상은 새롭게 바뀐다. 각 플로우는 최소 예약 대역폭을 할당받고 잉여 대역폭은 목표 대역폭의 비율로 서비스 받으므로 EBFQ를 사용할 때 4초에서 10초 동안 각 플로우의 예상 출력 링크 점유율은 다음과 같이 계산된다.

$$f_0 = 0.4 + \frac{0.0}{0.1+0.2} \times 0.4 = 0.4Mbps$$

$$f_1 = 0.1 + \frac{0.1}{0.1+0.2} \times 0.4 = 0.233Mbps$$

$$f_1 = 0.1 + \frac{0.2}{0.1+0.2} \times 0.4 = 0.367Mbps$$

그림에 나타난 4초에서 10초 사이의 시뮬레이션 결과는 계산된 예상 점유 대역폭과 거의 일치한다.

각 시뮬레이션 결과를 종합하면 순수한 SCFQ 알고리즘은 최소 예약 대역폭을 보장하고 남은 잉여 대역폭에 대해서 각 플로우의 최소 예약 대역폭에 비례적으로 각 플로우에 할당되는 것을 볼 수 있다. 이에 반해 SCFQ를 기반으로 한 EBFQ 알고리즘은 최소 예약 대역폭을 보장하고 남은 잉여 대역폭에 대해서는 목표 대역폭에 비례적으로 분배되는 특성을 볼 수 있었다. 이러한 시뮬레이션 결과로 EBFQ 알고리즘은 융통성 있는 잉여 대역폭 분배 알고리즘임을 알 수 있다.

Ⅶ. 결 론

본 논문에서는 잉여 대역폭 분배의 융통성을 제공하기 위한 알고리즘인 EBFQ를 제안하고 성능을 분석하였다. EBFQ는 DGPS의 문제점을 개선하여 복잡도 측면과 성능 면에서 DGPS보다 뛰어나며, 특히 기존의 알고리즘의 고유한 특성을 유지하면서 함께 적용할 수 있어 잉여 대역폭 분배의 융통성을 제공하는 알고리즘으로 다양한 영역에서 적용할 수 있을 것이다.

참 고 문 헌

[1] Francois Toutain, "Decoupled Generalized Processor Sharing : A Fair Queueing Principle for Adaptive Multimedia Applications," Proc. INFOCOM'98, pp. 291~298, March 1998.

[2] A .K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Service Networks: The Single-Node Case," IEEE/ACM Trans. Networking, pp. 344~357, June 1993.

[3] S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," Proc. INFOCOM'94, pp. 636~646, June 1994.

[4] J.C.R. Bennett and H. Zhang, "WF2Q : Worst-case fair weighted fair queueing," Proc. INFOCOM'96, pp. 120~128, March 1996.

[5] J.C.R. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms," Proc. SIGCOMM'96, pp. 143~156, Aug. 1996.

[6] G. Xie and S. Lam, "Delay guarantee of virtual clock server," IEEE/ACM Transactions on Networking, 3(4) : 683-689, December 1995.

[7] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," In Proceedings of ACM SIGCOMM'90, pp. 19~29, September 1990.

[8] P. Goyal, H. M. Vin, and H. Chen, "Start-time Fair Queueing : A scheduling algorithm for integrated service," In Proceedings of ACM-SIGCOMM'96, pp. 157~168, Palo Alto, CA, August 1996.

[9] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," IEEE Jounal on Selected Areas in Communications, 8(3):363-376, 1990.

[10] S. S. Lam and G. G. Xie, "Group Priority Scheduling," IEEE/ACM Transaction on Networking. Vol. 5, No. 2. April 1997.

[11] A. Varma and D. Stiliadis, "Hardware Implementation of Fair Queueing Algorithm for ATM Network," IEEE Communication Magazine, December 1997.

[12] D. Stiliadis and A. Varma, "Latency-Rate Servers : A General Model for Analysis of Traffic Scheduling Algorithms," Proc. IEEE INFOCOM'96, pp. 111~119, April 1996.

저 자 소 개

秋 皓 喆(正會員)

1998년 2월 : 숭실대학교 정보통신공학과(학사). 2000년 2월 : 숭실대학교 정보통신공학과(석사). 2000년 1월~현재 : LG전자㈜. <주관심분야 : 차세대 인터넷 QoS 기술, 무선 인터넷>

金 永 翰(正會員)

1984년 2월 : 서울대학교 전자공학과 졸업(공학사). 1986년 2월 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1990년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학박사). 1987년 1월~1994년 8월 : 디지콤정보통신연구소 데이터통신연구부장. 1994년 9월~현재 : 숭실대학교 정보통신전자공학부 부교수. 현재 : VoIP 포럼 차세대기술분과위원장, 통신학회 인터넷 연구회위원장. <주관심분야 : NGN, All-IP Network>