論文2003-40TC-12-1

잉여 대역폭 소비 큐를 이용한 계층적 잉여 대역폭 페어 큐잉

(Excess Bandwidth Hierarchical Fair Queueing Using Excess Bandwidth Consumer Queue)

金 永 翰 *, 秋 皓 喆 *

(Younghan Kim and Hocheol Choo)

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 구현의 복잡도를 감소시킨 EBFQ (excess bandwidth fair queueing) 알고리즘을 제안하였다[1]. 본 논문에서는 이를 확장하여 계층적 페어 큐잉 시스템에 적용한 알고리즘을 제안하였다. 제안된 알고리즘은 기존의 임의의 계층적 기반 알고리즘에 자연스럽게 적용할 수 있으며 동일한 공평성 등의 특성을 갖는다. 이러한 특성을 분석 및 시뮬레이션을 통해 검증하였다.

Abstract

Scheduling technology is one of the most important elements required to support the Quality of service (QoS) in the Internet and a lot of scheduling algorithms have been developed. However, most of these algorithms are not flexible to distribute the excess bandwidth. In order to provide flexibility for distributing the excess bandwidth, we proposed excess bandwidth fair queueing (EBFQ) algorithm with relatively low complexity[1]. In this paper, we propose the new extension to this EBFQ algorithm for the hierarchical fair queueing system. This extension can be naturally applied to the existing hierarchical algorithm and simultaneously provide the same level of fairness. Through the simulation and analysis, we verify it.

Keywords : 계층적 페어큐잉, 스케쥴링, 인터넷 QoS, 잉여대역폭

I. 서 론

인터넷 종합 서비스 망은 실시간 서비스, 최선형 서비스, 그리고 적응형 서비스등과 같은 다양한 서비스 클래스들을 동시에 지원하는 것을 목표로 하고 있다[2-5].

이를 위해서는 각 클래스 별로 집합적으로 모인 그룹 단위의 링크 분배를 지원할 수 있어야 한다. 여기서 각 서비스 클래스들은 소속별, 프로토콜 별, 트래픽 종류 별, 또는 이외의 다른 형태 별로 묶여질 수 있다. 이러한 각 클래스들은 또 다시 새로운 클래스로 그룹화 하여 계층적으로 링크를 분배할 수 있다. <그림 1>은 각 클래스 단위의 계층적인 링크 분배의 예를 나타냈다.

<그림 1>에서는 출력 링크의 대역폭을 각 에이전트 별로 각각 50%, 40%, 10%씩 사용할 수 있도록 하였고, 하위 계층에서는 실시간(real-time), 텔넷(t elnet), 파일 전송 프로토콜(ftp) 플로우들을 클래스 별로 집합 화하

* 正會員, 崇實大學校 情報通信電子工學部

(School of Electronics Engineering, Soongsil University)

※ 본 논문은 숭실대 교내연구비 지원 결과임

接受日字:2003年11月8日, 수정완료일:2003年12月3日

여 각 클래스 별로 다시 상위 계층에 할당된 대역폭을 재분배하도록 구성하였다. 이와 같이 각 클래스들을 계층적으로 구성하여 링크 분배(link-sharing)를 통해 얻어지는 이점은 다양한 특성을 지닌 여러 클래스들에게 각각의 특성을 고려하여 적절하게 링크를 할당하여 각 클래스에 적합한 서비스를 제공할 수 있다는 것이다. 이와 같은 계층적인 링크 분배 방식으로 CBQ(Class Based Queueing) [8]와 H-PFQ(Hierarchical Packet Queueing)[6]등이 제안되었다.

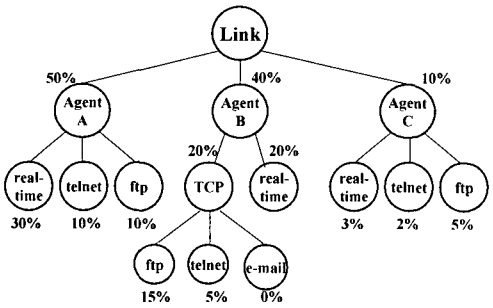


그림 1. 계층적인 링크 분배의 예
Fig. 1. Example of Hierachical Link Sharing.

이들 방식 중 CBQ는 각 클래스에 할당된 대역폭을 엄격하게 보장하지 못한다는 점이 있고, H-PFQ는 각 클래스에 할당된 대역폭을 엄격하게 지키는 반면, 트래픽을 발생시키지 않는 클래스들에 의해 생긴 잉여 대역폭(excess bandwidth)은 해당 클래스와 같은 수준의 클래스, 즉 형제 클래스(예를 들면, <그림 1>의 에이전트 A, B, C는 서로 형제 클래스이며, 또 다른 예로 에이전트 A의 직계 자식 노드들인 실시간(real-time), 텔넷(telnet), ftp 클래스는 서로 형제 클래스임)에 할당된 링크 분배율에 비례적으로 재분배된다. 예를 들면 <그림 1>의 에이전트 C 하위에 있는 실시간(real-time), 텔넷(telnet), ftp 클래스가 휴지(idle) 상태라고 가정하면 에이전트 C에 할당된 10%의 대역폭은 잉여 대역폭으로 간주되고 H-PFQ의 특성에 따라 에이전트 A와 B에 할당된 링크 분배율에 비례적으로 분배된다. 즉 A와B에 추가적으로 각각 5.6%, 4.4% 씩 더 분배된다. 이러한 특성의 가장 주된 요인은 각 스케쥴링 노드에 구현된 스케쥴링 알고리즘 자체의 잉여 대역폭 분배 특성에 의존하기 때문이다. 이점을 개선하여 최소요구대역폭의 분배와 잉여대역폭의 분배를 서로 다른 기준에 의해 할 수 있는 EBFQ(Excess Bandwidth Fair Queueing) 알고리즘이 제안되었다[1]. 그러나 EBFQ에서는 계층적 분배없이 모든 세션을 동일하게 취급한 것으로서 본 논문에서는 EBFQ 알고리즘을 확장하여 계층적 분배알고리듬에 적용할 수 있도록 하였다.

다음 각 절에서는 각 스케쥴링 노드에 잉여 대역폭을 원하는 별도의 비율로 배분할 수 있는 EBFQ를 적용하여 최하위의 클래스에서부터 최 상위 클래스들까지 잉여 대역폭을 효과적으로 분배할 수 있는 기법인 H-EBFQ(Hierarchical EBFQ) 알고리즘을 설명하고, 또한 H-EBFQ의 지연 한계 및 공평성 특성에 대해 분석한다. EBFQ와 본 H-EBFQ는 단순히 각 세션에게 균등한 비율로 공평하게 잉여대역폭을 분배하는 대신 사용자의 요구에 따라 별도의 잉여대역폭 분배 기준에 따라 분배하는 방식으로서 공평성의 판단 기준이 적용되며[1] 이러한 기준에 따라 공평성 및 지연특성 등을 분석하였다..

Ⅱ. H-EBFQ(Hierarchical-EBFQ)

<그림 2>는 H-EBFQ의 구조 예를 나타낸다.

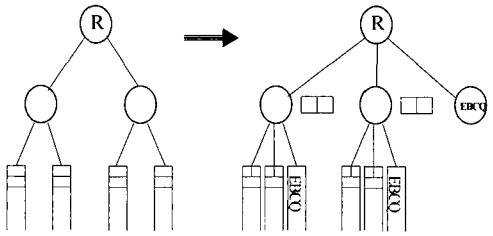


그림 2. H-EBFQ 구조 예
Fig. 2. Example of H-EBFQ Structure.

표 1. H-EBFQ 알고리즘에 사용된 표기들
Table 1. Symbols used for H-EBFQ Algorithm.

$VM_k(t)$	최소 서비스를 위한 노드 k의 가상 시간 함수
$VT_k(t)$	목표 서비스를 위한 노드 k의 가상 시간 함수
Q_k	노드 k의 큐
$Q_k(t)$	시간 t에 노드 k의 큐의 HOL에 있는 패킷
$F_k(t)$	노드 k의 최소 서비스 태그 값
$T_k(t)$	노드 k의 목표 서비스 태그 값
$Busy_k(t)$	시간 t에 노드 k가 backlog되어 있다면 참
$p(k)$	노드 k의 부모 노드

그림의 왼쪽은 일반적인 계층적 구조이고 그림의 우

측은 H-EBFQ를 적용할 경우의 계층적 구조를 개념적으로 나타낸 그림이다. EBCQ는 각 스케줄링 노드에 하나씩 존재하여 각 계층별로 잉여 대역폭을 효율적으로 분배하기 위한 도구로 사용된다.

다음은 H-PFQ[6]에 EBFQ를 적용하기 위해 수정된 H-EBFQ 알고리즘을 나타내었다. H-EBFQ의 알고리즘은 크게 ARRIVE(), RESTART-NODE(), RESET-PATH(), SELECT-NEXT()의 네 부분으로 구성된다. 또한 사용되는 기반 알고리즘에 따라 VMk(t), VTk(t)가 정의된다. 본 논문에서 사용된 SCFQ를 기반으로 할 때는 VMk(t), VTk(t) 는 각각 Fk(t), Tk(t) 이 된다[7].

```
ARRIVE(k, Packet)
 ENQUEUE(Qk, Packet)
 if Qk(t) ≠ 0
 then return
 UPDATE ExcessBandwidth(k, minus)
 Qk(t) ← Packet
 Fk(t) ← max(Fn(t), VMp(n)(t)) + Lk(t)/rk
 Tk(t) ← max(Tn(t), VTp(n)(t)) + Lk(t)/tk
 if Busyp(k) = FALSE
 then RESTART NODE(p(k))
```

패킷이 노드에 도착하는 시점에 해당 큐에 서비스 받을 패킷이 없었다면, 즉 휴지(idle)상태에 있던 노드였다면 새롭게 활성화되는 노드이므로 부모 노드의 잉여 대역폭을 갱신한다. 이러한 동작을 UPDATE-ExcessBandwidth() 함수에서 수행한다. 또한 해당 노드의 최소 서비스 태그 값(Fk(t)) 및 목표 서비스 태그 값(Tk(t))을 계산한다. 그리고 부모 노드가 휴지(idle) 상태이면 RESTART-NODE()를 호출한다.

```
RESTART NODE(n)
 m ← SELECT NEXT(n)
 if m ≠ 0
 then
 ActiveChildn ← m
        Qn(t) ← Qm(t)
        if Busyn(t) ← TRUE
        then
 Fk(t) ← max(Fn(t), VMp(n)(t)) + Lk(t)/rk
        Tk(t) ← max(Tn(t), VTp(n)(t)) + Lk(t)/tk
        Busyn ← TRUE
 else
        ActiveChildn ← 0
        Busyn ← FAUSE
 if (n ≠ R) and (Qp(n)(t) = 0)
 then
 RESTART NODE(p(n))
 if (n = R) and (Qn(t) ≠ 0)
 then
 TRANSMIT PACKET TO LINK(Qn(t))
```

각 노드는 서비스를 위해 새로운 패킷을 선택하려고 할 때마다 RESTART-NODE()를 수행할 것이다. 임의의 노드 n은 SELECT-NEXT() 함수의 동작에 따라 합당한 노드의 패킷을 선택하게 되고, 그 패킷은 노드 n에 할당된 목표 대역폭 또는 최소 서비스 대역폭으로 계산된 태그 값을 가지고 노드 n의 형제 노드들과 서비스 경쟁을 한다. 이러한 동작은 idle하지 않은 부모 노드 또는 최상위 노드 R에 다다를 때까지 계속된다.

```
RESET PATH(n)
 Qn(t) ← 0
 if Leaf(n) = TRUE
 then
 DEQUEUE(Qn)
 if Qn(t) ≠ 0
 then if Selectedn = MIN
 then
 Fm(t) ← max(Fm(t), VMp(m)(t)) + Lm(t)/rm      else
        Tm(t) ← max(Tm(t), VTp(m)(t)) + Lm(t)/tm
        else
        Fn(t) ← 1
        Tn(t) ← 1
        UPDATE ExcessBandwidth(k, plus)
        RESTART NODE(p(n))
 else
 m ← ActiveChildn
        ActiveChildn ← 0
        RESET PATH(m)
```

최상위 노드에 도착하여 출력 링크로 패킷이 전송되면 해당 패킷이 속한 최하위 노드까지 이르는 각 노드에 설정된 플래그 또는 태그 값을 갱신 또는 리셋하여야 한다. RESET-PATH() 함수에 의해 최하위 노드에 도착하면 태그 값 등을 갱신하고 다시 RESTART-NODE() 함수를 호출하여 다음에 서비스할 패킷의 선택 및 전송 동작을 재개한다.

SELECT-NEXT() 함수는 EBFQ 동작에 의한 패킷 선택 과정을 보인다. 먼저 노드 n의 자식 노드들의 최소 서비스 태그 값들 중에서 가장 작은 값을 갖는 자식 노드를 선택한 다음 그 노드가 가상 노드(EBCQ)일 경우 목표 서비스 태그 값이 가장 작은 자식 노드를 서비스 해줄 노드로 선택하고, 그렇지 않은 경우 첫 번째 선택

```
SELECT NEXT(n)
m ← SELECT MIN_TAG(n)
if m ≠ EBCQ
then
Selected_m ← MIN
else
m ← SELECT TAR_TAG(n)
Selected_m ← TAR
UPDATE V(n)
return m
```

동작에 의해 선택된 노드를 서비스해줄 노드로 선택한다. 그리고 UPDATE-V(n)은 노드 n의 가상 시간 함수를 갱신하는 함수이고, 이 함수는 EBFQ의 기반 알고리즘에 의존적인 부분이다.

이상의 알고리즘은 EBFQ에서 분석하였듯이 기존의 알고리즘에 비해 융통성이 증가한 대신 구현의 복잡도가 약간 증가되나 DGPS알고리즘을 기반으로 하고 있는 계층적 잉여대역폭 분배 알고리즘에 비해서는 복잡도가 줄어든다[1].

Ⅲ. H-EBFQ의 지연 한계(delay bound)와 공평성 분석

1. H-EBFQ의 지연 한계(delay bound) 분석

본 절에서는 H-EBFQ의 스케쥴링 노드에 사용될 스케쥴링 알고리즘으로 SCFQ (Self-Clocked Fair Queueing)[7]를 기반으로 한 EBFQ를 사용할 때의 최악 상황의 지연 한계와 공평성을 분석한다. 분석을 용이하게 하기 위해 모든 패킷의 크기는 L로 같다고 가정한다. 임의의 최하위 노드를 k라고 할 때, k의 부모 노드는 p(k)이고, k의 최상위 노드를 H라고 하자. 따라서, h=1, … , H일 때, $p^h(k)$는 $p^{h-1}(k)$의 부모 노드로 나타낸다. 다음은 임의 노드에 패킷이 도착했을때 그 노드에서 지연 한계값에 대한 정의이다.

정리 1. 임의의 노드 k에 도착한 패킷이 서비스 받기 위해 부모 노드에 의해 선택되는데 까지 걸리는 시간을 $s_{p(k)}(p_i^k)$라고 정의하면, $s_{p(k)}(p_i^k)$는 다음과 같이 제한된다.

$$s_{p(k)}(p_i^k) \le \tau + N(k)\frac{L}{r_k}$$

증명) 노드 k의 i번째 패킷을 나타내는 p_k^i가 τ 시간에 노드 k에 도착했다고 할 때 p_k^i의 가상 종료 시간 값은 다음과 같다.

$$v_{p(k)}(d_k^i) = \frac{L}{r_k} + \max(v_{p(k)}(d_k^{i-1}), v_{p(k)}(\tau)) \quad (1)$$

여기서 $v_k(t)$는 노드 k의 가상 시간 함수를 나타내고, d_k^i는 p_k^i의 실제 서비스 종료 시간을 나타낸다. 또한 여기서 $v_{p(k)}(d_k^{i-1}) \le v_{p(k)}(\tau)$이므로 $v_{p(k)}(d_k^i)$ 은 식 (2)가 된다.

$$v_{p(k)}(d_k^i) = \frac{L}{r_k} + v_{p(k)}(\tau) \quad (2)$$

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

$$v_j(\tau) \le v_{p(k)}(\tau) \quad (3)$$

여기서 노드 j는 노드 i의 형제 노드이며, 노드 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.

$$v_{p(k)}(d_k^i) - v_j(\tau) = \frac{W_j}{r_j} \quad (4)$$

식 (3)을 식 (2)에 적용하면

$$v_{p(k)}(d_k^i) - v_j(\tau) \le \frac{L}{r_k} \quad (5)$$

식 (5)가 되고, 식 (4)를 식 (5)에 적용하면 W_j 는 다음과 같은 조건을 만족하게 된다. 여기서 C는 링크의 전체 대역폭이다.

$$W_j \le \frac{L}{r_k} r_j,\ r_j \le C \quad (6)$$

여기서 최악의 상황을 고려하면 $r_j = C$인 경우이고 따라서 식(6)은 식(7)이 된다.

$$W_j \le \frac{L}{r_k} C \quad (7)$$

패킷 p_k^i가 부모 노드에 의해 서비스를 받기 위해 선

택되어지는 시간은 각 노드 k의 형제노드들의 W_j 를 모두 서비스하고 난 후 가 될 것이다. 따라서 p_k^i가 부모 노드에 선택되어질 시간은 다음과 같은 한계 값을 가진다.

$$s_{p(k)}(p_k^i) \le \tau + \frac{\sum_{j=1}^{N(k)} W_j}{C} \tag{8}$$

여기서 N(k)는 k 노드의 형제 노드 수를 나타낸다. 그리고 식 (8)에 식 (7)을 적용하면 식 (9)가 된다.

$$s_{p(k)}(p_k^i) \le \tau + \frac{\sum_{j=1}^{N(k)} \frac{L}{r_k} C}{C} = \tau + N(k)\frac{L}{r_k} \tag{9}$$

따라서 임의의 노드에서 부모 노드에 의해 서비스 받기 위해 선택되는데 걸리는 지연은 다음과 같다.

$$s_{p(k)}(p_i^k) - \tau \le N(k)\frac{L}{r_k} \tag{10}$$

이로써 정리 1은 증명된다.

정리 1로부터 H-EBFQ의 최하위 노드에 도착하는 패킷이 최상위 노드에 의해 선택되어 서비스 받는데 까지 걸리는 최악의 지연 한계는 식 (10)을 최하위 노드의 경우에서부터 최상위 노드에까지 총합으로 구해질 수 있다. 따라서 임의의 최하위 노드에 도착한 패킷의 최악의 지연은 다음과 같은 한계값을 가진다.

$$d_k^i - \tau \le \sum_{h=0}^{H-1} N(p^h(k))\frac{L}{r_{p^h(k)}} \tag{11}$$

식 (10)에서 h=0일 경우, 즉 계층적인 구조를 이루지 않을 경우에 대해 계산하면 순수한 SCFQ에서 패킷 크기가 L로 같다고 가정했을 때의 지연한계와 같음을 알 수 있다.

2. H-EBFQ의 공평성 분석

H-EBFQ의 공평성 분석은 앞서 H-EBFQ의 지연 분석할 때 정의했던 계층적 구조에서 사용될 변수를 이용하고, 또한 공평성 분석 자체에서 사용되는 변수에 대해서도 계층적 구조를 감안하여 새롭게 정의되어야 한다. 먼저 임의의 시간에서 노드 k의 정규(normalized) 서비스 부족분, $\delta_k(t)$를 다음과 같이 새롭게 정의한다.

$$\delta_k(t) = v_{p(k)}(t) - v_k(t) \tag{12}$$

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 (12)은 식 (13)이 된다.

$$\delta_k(\tau) = \delta_k(b_k^i) + \delta_k(b_k^i, \tau) = v_{p(k)}(b_k^i, \tau) - v_k(b_k^i, \tau) \tag{13}$$

또한 시간 d_k^i에서 노드 k의 부모 노드 가상 시간 함수 값은 다음과 같다.

$$v_{p(k)}(d_k^i) = \frac{L}{f_k} + v_{p(k)}(b_k^i) \tag{14}$$

식 (14)로부터 식 (15)가 아래와 같이 전개된다.

$$v_{p(k)}(d_k^i, b_k^i) = v_{p(k)}(d_k^i) - v_{p(k)}(b_k^i) = \frac{L}{f_k} \tag{15}$$

그리고 식 (14), 식 (15)에 의해 아래의 부등식이 유도된다.

$$0 \le v_{p(k)}(b_k^i, \tau) \le v_{p(k)}(b_k^i, d_k^i) \le \frac{L}{f_k} \le \frac{L}{r_k} \tag{16}$$

또한 세션의 가상 시간 함수에 대한 부등식은 다음과 같다.

$$0 \le v_k(b_k^i, \tau) \le v_k(b_k^i, d_k^i) = w_k(b_k^i, d_k^i) \le \frac{L}{r_k} \tag{17}$$

부등식 (16)과 (17)을 통해 임의 시간에 세션 k의 정규(normalized) 서비스 부족분에 대한 부등식이 다음과 같이 구해진다.

$$0 \le \delta_k(\tau) \le \frac{L}{r_k} \tag{18}$$

따라서 임의의 시간 간격 [t_1, t_2]동안 세션 k의 정규(normalized) 서비스 부족분은 다음의 한계 값을 가진다.

$$|\delta_k(t_1, t_2)| \le \frac{L}{r_k} \tag{19}$$

식 (19)는 단일한 스케쥴링 노드에서 발생할 수 있는 정규(normalized) 서비스 부족분을 나타낸다. 따라서 H-EBFQ와 같이 계층적 구조에서 실제 서비스는 최상위 노드로부터 수신 받으므로, 계층적 구조에서의 정규(normalized) 서비스 부족분은 최하위 노드에서 최 상위 노드에 이르는 경로내의 모든 스케쥴링 노드에서 발생하는 정규(normalized) 서비스 부족분의 총합이 된다.

다음은 H-EBFQ에서 최하위 노드의 정규(normalized) 서비스 부족분을 나타낸다.

$$\Delta_k(t_1, t_2) = \sum_{h=0}^{H-1} \delta_{p^h(k)}(t_1, t_2) \qquad (20)$$

따라서 H-EBFQ 시스템에서 [t_1, t_2] 동안 계속해서 서비스 받을 트래픽이 존재하는 임의의 두 최하위 노드 간의 정규(normalized) 서비스 부족분의 차이는 아래와 같은 부등식을 만족한다.

$$|\Delta_k(t_1, t_2) - \Delta_j(t_1, t_2)| \le L \sum_{h=0}^{H-1} \left(\frac{1}{r_{p^h(k)}} + \frac{1}{r_{p^h(j)}}\right) \qquad (21)$$

위의 식에도 역시 h=0을 대입하면 계층적인 구조를 이루지 않는 EBFQ에 SCFQ를 기반 알고리즘으로 사용할 경우, 패킷 크기가 L로 같다고 가정했을 때의 공평성 특성과 같음을 알 수 있다.

Ⅳ. 시뮬레이션

본 절에서는 계층적으로 링크를 분배하기 위해 제안된 알고리즘인 H-PFQ의 잉여 대역폭 분배 특성을 효율적으로 제공하기 위해 H-PFQ의 스케쥴링 노드에 EBFQ를 적용한 알고리즘인 H-EBFQ의 시뮬레이션을 수행한다. 본 시뮬레이션에서는 각 계층 그리고 각 최하위 노드에 할당된 최소 대역폭 보장 및 잉여 대역폭 분배 성능을 보인다. EBFQ 시뮬레이션에 사용되었던 것과 마찬가지로 기반 알고리즘은 SCFQ이고, 사용된 플로우는 모드 CBR, 그리고 각 플로우는 패킷 크기도 53 바이트로 동일하며, 출력 링크의 용량은 1Mbps이다. EBFQ와 SCFQ와의 분배특성의 시뮬레이션을 통한 검증은 이전논문[1]에서 수행하였으며 EBFQ의 분배특성을 검증하였다. 본 논문에서는 역시 같은 SCFQ를 사용한 일반방식에 비해 원하는 r기준에 따라 잉여대역폭이 계층적으로 분배됨을 검증하며 앞서의 공평성 분석결과를 재확인하게 된다. <표 2>는 H-EBFQ 시뮬레이션에 사용된 플로우의 도착 속도와 예약 대역폭, 그리고 목표 대역폭을 나타낸다.

<그림 3>은 H-EBFQ 시뮬레이션에 사용된 계층적 구조를 나타낸다.

그림의 괄호 바깥쪽의 숫자는 최소 예약 대역폭을 나타내고, 괄호 안의 숫자는 목표 대역폭을 나타내고, 이 두 숫자의 단위는 Mbps이다.

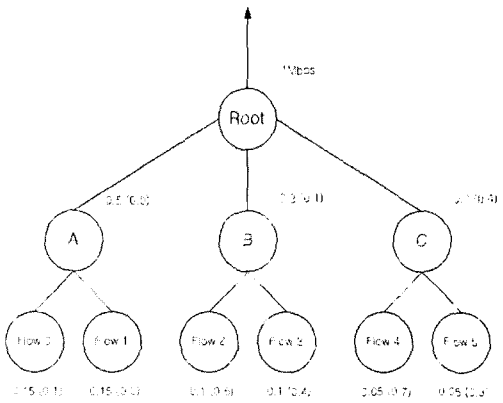


그림 3. H-EBFQ 시뮬레이션에 사용된 계층적 구조
Fig. 3. Structure of H-EBFQ used for Simulation.

시뮬레이션에서 플로우 1을 제외한 모든 플로우는 "0" 초에 트래픽 발생을 시작하고, 플로우 1은 2초부터 발생시킨다. 그리고 에이전트(agent) A에 속한 플로우 0과 플로우 1은 모두 4초에 트래픽 발생을 중단하고 더 이상 트래픽 발생을 하지 않는다. <그림 4>는 각 플로우의 출력 링크 점유 대역폭을 보인다. 최초 0초에서 2초까지 출력 링크 대역폭 점유 양상을 살펴보면 플로우 0을 제외한 모든 플로우가 트래픽을 발생시킨다. 이는 모든 에이전트가 엑티브하다는 것을 말하므로 에이전트 수준에서는 잉여 대역폭이 생기지 않는다. 그러나 에이전트 A에 속한 플로우 0이 휴지(idle) 상태이므로 에이전트 A의 하위에서 생긴 잉여 대역폭은 에이전트 A에 속한 플로우들의 잉여 대역폭의 비율에 따라 분배된다. 이때 에이전트 A에 속한 플로우 중 플로우 1만이 트래픽을 발생시키므로 에이전트 A에서 생긴 잉여 대역폭은 모두 플로우 1에 분배된다. 이러한 양상은 플로우 0이 트래픽 발생을 시작하는 2초에서 4초까지 약간의 변화가 생긴다. 그러나 플로우 0의 최소 예약 대역폭은 0.15Mbps이고 목표 대역폭은 없으므로 플로우 0과 1은 각각 자신들이 요구하는 최소 예약 대역폭인 0.15Mbps를 할당받고, 에이전트 A에서 발생하는 잉여 대역폭 0.2Mbps는 모두 플로우 1에 분배된다. 에이전트 A와 마찬가지로 에이전트 B와 C도 에이전트 수준에서는 최소 예약 대역폭에 해당하는 서비스를 받고, 각 에이전트 하위에서 발생한 잉여 대역폭은 하위 노드들에 설정된 목표 대역폭의 비율로 각 하위 노드들에게 분배한다. 이러한 양상은 0에서 4초까지 지속된다. 그러나 4초에 에이전트 A의 모든 플로우가 트래픽 발생을 중단하므로 에이전트 A는 휴지(idle) 상태가 된다. 이로 인해 에이전

트 B와 C는 에이전트 A로 인해 생긴 잉여 대역폭을 자신들의 목표 대역폭의 비율로 분배받는다. 이렇게 분배된 잉여 대역폭은 하위 노드들의 목표 대역폭의 비율로

표 2. H-EBFQ 시뮬레이션에 사용된 플로우 특성
Table 2. Characteristics of Flows used in Simulation.

에이전트	플로우	도착 속도 (Mbps)	최소 예약 대역폭 (Mbps)	목표 대역폭 (Mbps)
A			0.50	0.0
	0	0.50	0.15	0.1
	1	0.15	0.15	0.0
B			0.30	0.1
	2	0.25	0.10	0.6
	3	0.20	0.10	0.4
C			0.20	0.4
	4	0.45	0.05	0.7
	5	0.25	0.05	0.3

플로우 2 = 0.1 + 0.1*0.6 + 0.1*0.6 = 0.22Mbps
플로우 3 = 0.1 + 0.1*0.4 + 0.1*0.4 = 0.18Mbps
플로우 4 = 0.05 + 0.1*0.7 + 0.4*0.7 = 0.4Mbps
플로우 5 = 0.05 + 0.1*0.3 + 0.4*0.3 = 0.2Mbps

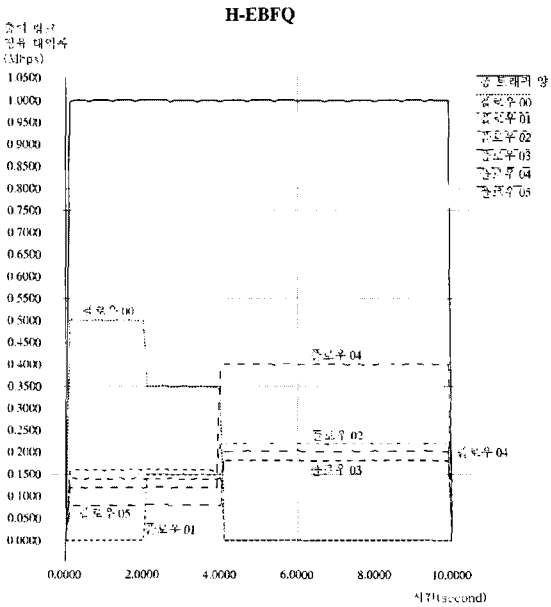


그림 4. H-EBFQ 시뮬레이션 결과
Fig. 4. Simulation Result of H-EBFQ.

재분배된다. 즉 에이전트 B와 C는 A에 의해 발생한 0.5Mbps의 잉여 대역폭을 자신의 목표 대역폭의 비율로 분배받는다. 따라서 에이전트 B는 0.2Mbps를 C는 0.3Mbps의 잉여 대역폭을 분배받는다. 또한 에이전트 B와 C의 하위 노드들은 자신이 속한 에이전트에 분배된 잉여 대역폭에 대해서도 역시 자신들의 목표 대역폭의 비율로 분배받는다. 따라서 플로우 2, 3, 4, 5는 다음과 같이 출력 링크 대역폭을 점유한다.

본 시뮬레이션 결과를 통해 EBFQ를 계층적 페어큐잉 알고리즘에 적용하여도 사용자가 정한 기준에 따라 잉여대역폭이 분배됨을 확인할 수 있다.

V. 결 론

본 논문에서는 최소요구 대역폭의 엄격한 보장은 물론 잉여 대역폭의 분배에 있어서도 융통성을 제공하기 위해 제안된 EBFQ 알고리즘을 계층적인 링크 분배 알고리즘에 적용하여 보다 더 다양하고 효율적인 성능을 제공하는 H-EBFQ를 제안하였다. 계층적인 잉여 대역폭 분배 메카니즘을 적용하면 보다 더 다양한 서비스를 제공할 수 있으며, 정책적으로 트래픽을 관리하고 제어하는데 많은 이점을 제공한다. 본 논문에서 제안된 H-EBFQ는 기존에 제안된 H-PFQ (Hierarchical Packet Fair Queueing)의 잉여 대역폭 분배의 효율성을 제공하기 위해 각 스케쥴링 노드에 EBFQ를 채택하였으며, EBFQ를 H-PFQ에 적용하기 위해 기존의 H-PFQ 알고리즘을 수정하였다. 그 결과 H-EBFQ는 기존의 H-PFQ의 장점인 엄격한 대역폭 보장은 물론 EBFQ의 장점인 잉여 대역폭의 효율적인 분배 특성을 지닌 계층적인 링크 분배 알고리즘이다. 따라서 H-EBFQ를 계층적 구조에 적용하면 여러 트래픽이 요구하는 다양한 서비스를 수용할 수 있을 뿐 아니라 클래스 또는 플로우의 집합적 단위로 뿐 아니라 최하위 계층의 개개 플로우에 대해서도 엄격하게 관리할 수 있으므로 트래픽에 대한 다양한 서비스 정책을 제공할 수 있다. 이와 같은 서비스 및 정책의 제공은 향후 다양한 서비스를 요구하는 멀티미디어 플로우 수용을 용이하게 할 것이다.

참 고 문 헌

[1] 추호철, 김영한, "잉여 대역폭 소비 큐를 이용한 잉여 대역폭 페어 큐잉," 전자공학회논문지, 제 39

권, TC편, 제 10호, pp. 421-430, 2002년 10월

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

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

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

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

[6] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," In Proc eedings of ACM SIGCOMM'96, pp. 143-156, Palo Alto, CA, August 1996.

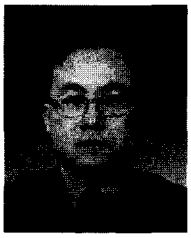
[7] S.Jamaloddin Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applicati ons," In Proceeding of IEEE INFOCOM'94, pp 636-646, Toronto, CA, June 1994

[8] S. Floyd and V. Jacobson, "Link sharing and resource management models for packet networks," IEEE/ACM Transactions on Net working, 3(4):365-386, August 1995.

저 자 소 개

秋 皓 喆(正會員)

1998년 2월 : 숭실대학교 정보통신공학과(학사). 2000년 2월 : 숭실대학교 정보통신공학과(석사). 2000년 1월~2002년 8월 : LG전자(주). 현재 : 팬택큐리텔(주). <주관심분야 : 차세대 인터넷, 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 인터넷연구회위원장. <주관심분야 : NGN, All-IP Network>