論文99-36S-1 1

LCFQ(Linear Clock Fair Queueing) 알고리즘의 설계와 성능 분석

(Design and Performance of Linear Clock Fair Queueing Algorithm)

金 永 翰 *, 李 在 容 **

(Younghan Kim and Jaeyong Lee)

요 약

종합 서비스망에서, 트래픽을 발생시키는 각각의 플로우에 대하여 QoS(Quality of Service)를 적절하게 제공하기위해서는 호스트와 라우터에 자원 예약뿐만 아니라 효율적인 트래픽 스케쥴링이 채택되어야 한다. 본 논문에서는 가상시간이 선형적으로 증가하는 새로운 페어 큐잉 알고리즘을 제안한다. 본 논문에서 제안된 알고리즘은 각 플로우에 대하여 기존의 SCFQ (Self-Clocked Fair Queueing) 알고리즘과 유사한 구현 복잡성을 갖으면서 더 감소된 최대 지연와 평균 지연시간을 제공하고 공평성 측면에서도 개선됨을 보여준다. 또한 자신에게 할당된 대역폭보다 더 많은 트래픽을 발생시키는 플로우에 의해 다른 플로우는 영향을 받지 않도록하는 독립성 특성 또한 SCFQ보다 더 좋은 성능을 보인다. 본 논문에서는 제안된 알고리즘에 대한 공평성을 증명했고, 최대지연과 평균지연시간에 대한 시뮬레이션 결과를 나타냈다.

Abstract

In order to provide appropriate Quality of Service(QoS) guarantee to each traffic flow in intergrated service networks, an efficient traffic scheduling algorithm as well as resource reservation must be adopted in the host and transit routers. In this paper, a new efficient fair queueing algorithm which adopts a linearly increasing virtual time is presented. The proposed algorithm is fair and the maximum and mean delay guaranteed for each flow are less than those of the SCFQ(self clocked fair queueing) algorithm which is one of the most promising traffic scheduling algorithm, while providing low implementation complexity as the SCFQ scheme. And, it has the better isolation property than SCFQ, which means that each flow is much less influenced by the violating traffic flows provided its allocated bandwidth gurantee. The fairness of the proposed algorithm is proved and simulation results of maximum and mean delay are presented.

I. 서 론

종합 서비스망은 여러 종류의 트래픽을 모두 수용할 수 있어야 하고 그러한 각각의 트래픽에 서비스 품질을 보장해 줄 수 있어야 한다. 즉 음성과 화상과 같은 실시간 플로우에 대해서는 최대지연시간을 보장해 주어야 한다[1]. 또한 각 플로우에 대해 적절하게 성능을 보장해 주기 위해서 망 대역폭은 각 플로우에 적

* 正會員, 崇實大學校 情報通信電子工學部
(School of Electronic Engineering, Soongsil Univ.)
** 正會員, 忠南大學校 情報通信工學科
(Dept. of Information communication Eng., Chungnam National University)
※ 본 논문은 정통부 대학기초 연구지원에 의해 이루어졌음.(96084-CT-I1)
接受日字:1998年6月3日, 수정완료일:1998年11月23日

절히 할당되어야 한다. 지금까지 이와 관련된 여러 가지의 트래픽 스케줄링 알고리즘들이 제안되어져 왔다[1-14]. 이중 PGPS(packet-by-packet generalized processor sharing)라 불리는 WFQ(weighted fair queueing)는 LB(leaky-bucket) 알고리즘과 조합해서 사용할 때 공정한 대역폭 분배를 제공하고 worst-case 패킷 지연시간을 보장해 주는 것으로 알려져 있다[2]. 그러나 WFQ 알고리즘 구현에는 스케줄링 알고리즘 구현에 필요한 가상시간을 계산하는데 있어 상당한 복잡도가 따른다. 즉 WFQ는 fluid GPS 알고리즘을 시뮬레이션 해야 하고 패킷의 도착과 출발 시 마다 backlog된 플로우들의 수를 추적해야 하므로 고속 응용에는 적용하기 어렵다.

이를 개선하여 Golestani에 의해 제안된 SCFQ(self-clocked fair queueing)은 현재 서비스 받고 있는 패킷의 가상 시간을 현재 시스템의 가상시간으로 사용하여 가상시간이 쉽게 계산되어질 수 있기 때문에 ATM과 같은 고속망에 사용될 수 있다[3]. 그러나 SCFQ알고리즘이 LB 알고리즘과 함께 사용하여 각 플로우에 최대지연시간을 보장하지만 보장된 최대지연시간이 WFQ 알고리즘의 최대지연시간보다 더 크고 트래픽 독립화(isolation) 특성에 있어서도 저하된 성능을 보인다.

본 논문에서는 SCFQ 알고리즘을 개선하여 self-clocked 가상 시간이 선형적으로 증가하고 감소된 최대지연시간과 평균지연시간을 제공하며 또한 개선된 트래픽 독립화(isolation) 특성을 갖는 LCFQ (Linear Clocked Fair Queueing) 알고리즘을 제안하였다. LCFQ는 SCFQ와 같이 self-clocked 방식의 가상시간을 사용하기 때문에 구현이 용이하다.

서론에 이어 2장에서는 LCFQ 알고리즘의 동작원리를 기술하고 3장에서는 LCFQ의 공평성을 증명한다. 4장에서는 시뮬레이션을 통해 LCFQ 알고리즘을 SCFQ와 비교하여 지연 시간과 독립성 특성을 비교하고 5장에서 결론을 맺는다.

Ⅱ. LCFQ(Linear Clock Fair Queueing) 알고리즘의 동작

페어 큐잉 시스템은 각 트래픽 플로우에 할당된 대역폭을 보장해 주는 트래픽 스케줄링 시스템이다. 그림1에 N개의 트래픽 플로우들이 페어 큐잉 스케줄러에 의해 제어되는 시스템 모델을 나타내었다. 여기서 $\phi_k (k = 1, \Lambda, N)$는 플로우 k에 할당된 대역폭 비율을 나타낸다. 이 모델하에 $A_k(t)$와 $W_k(t)$를 각각 시간 $(0,t)$동안 플로우 k에 도착한 패킷의 총량과 서비스 받은 패킷의 총량이라 정의하고 플로우 k의 미처리된 트래픽의 총량을 $Q_k(t)$라 하면 $Q_k(t)$는 $A_k(t) - W_k(t)$가 된다. 또한 $w_k(t) = W_k(t)/\phi_k$라 하면, $w_k(t)$는 플로우 k에 제공된 normalized 서비스가 된다. 이와 비슷하게 $w_k(t_1,t_2) \equiv w_k(t_2) - w_k(t_1)$라고 놓으면 $w_k(t_1,t_2)$는 시간(t_1,t_2)동안 플로우 k가 제공받은 normalized 서비스이다. 트래픽 스케줄링 알고리즘의 공평성은 시간(t_1,t_2)사이에 연속적으로 미처리된 임의의 두 플로우에 대하여 동일한 normalized 서비스를 제공하는 것을 의미한다. 즉, 아래의 식을 만족하는 트래픽 스케줄링 알고리즘은 완벽한 공평성을 제공한다고 말할 수 있다.

$$w_k(t_1,t_2) = w_l(t_1,t_2), \quad k,l \in B(t_1,t_2). \qquad (1)$$

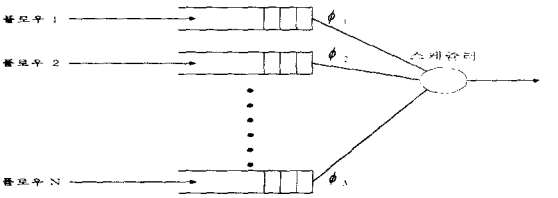


그림 1. 페어 큐잉 시스템 모델
Fig. 1. Model of a fair queueing system.

여기서 $B(t_1,t_2)$는 시간 (t_1,t_2)동안 연속적으로 미처리된 모든 플로우의 집합을 나타낸다.

페어 큐잉 알고리즘은 normalized 서비스를 참조하기 위하여 가상시간이라 불리우는 함수 $v(t)$를 정의한다. SCFQ 알고리즘에서 가상시간 $v(t)$는 그때의 시간 t에 서비스를 받고 있는 패킷의 서비스 태그와 같은 값으로 정의된다. P_k^i를 k번째 플로우의 i번째 패킷으로 나타내고 L_k^i는 그 패킷의 길이라고 정의하면 SCFQ에서 도착하는 각 패킷 P_k^i는 타임 스탬프(time-stamp) F_k^i값을 달고 큐에 입력된다. 각 큐의 패킷들의 서비스 순서는 이들 타임 스탬프 값이 작은

순서로 선택된다. 공평성을 유지하기 위한 각 패킷에 달게되는 타임 스탬프 값은 다음과 같이 계산된다.

$$F_k^i = \frac{L_k^i}{\phi_k} + \max\left\{F_k^{i-1}, v(a_k^i)\right\}, \qquad i = 1, 2\Lambda \ . \tag{2}$$

$$v(t) = F_l^j \ \ if \ \ j-th \ packet \ of \ flow \ l \ is \ in \ service.$$

만약 플로우 1의 j번째 패킷이 서비스 받고 있다면, $v(t) = F_1^j$ 이고, 여기서 a_k^i는 패킷 P_k^i의 도착시간을 나타낸다.

그러나 이와같은 SCFQ 알고리즘에 의한 성능은 각 플로우의 최대 지연 시간이 이상적인 WFQ에 비해 상대적으로 크고 트래픽간의 독립성 특성에 대해서도 좋지 않은 결과를 보인다. SCFQ 알고리즘에서 가상시간은 임의의 패킷이 서비스 받는 동안은 상수로써 유지된다. 반면 본 논문에서 제안하는 LCFQ 알고리즘에서는 현재 서비스 받고 있는 패킷의 가상시간이 이전에 서비스 받는 패킷의 가상시간보다 큰 경우, 가상시간을 선형적으로 증가시켰다. 다음에 LCFQ 알고리즘을 설명하였다. 우선 F_{priv}를 바로 이전에 서비스 받은 패킷의 가상종료시간, D_{priv}를 바로 이전에 서비스 받은 패킷의 실제 종료시간으로 정의한다. 또한 F_{now}를 현재 서비스 받는 패킷의 가상 종료시간, D_{now}를 현재 서비스 받는 패킷의 실제 종료할 시간으로 정의한다. 이에 따라 LCFQ 알고리즘의 가상시간함수 $v(t)$는 busy 기간 동안 감소하지 않는 값으로서 다음과 같이 정의된다.

$$v(t) = 0$$

$$v(t) = \begin{cases} F_{priv} & , \quad if \ F_{priv} \geq F_{now} \\ F_{priv} + \dfrac{t - D_{priv}}{D_{now} - D_{priv}}(F_{now} - F_{priv}) & , \quad if \ F_{priv} \leq F_{now} \ for \ D_{now} \leq t \leq D_{priv} \end{cases}. \tag{3}$$

위의 가상시간함수를 사용하여, 플로우 k의 i번째 패킷의 가상종료시간은 다음의 식에 따라 결정되며 이 값이 작은 순서로 패킷이 선택되어 전송된다.

$$F_k^i = \frac{L_k^i}{\phi_k} + \max\left\{F_k^{i-1}, v(a_k^i)\right\}, \ i = 1, 2, \Lambda \ . \tag{4}$$

그림 2는 $\phi_1 = \phi_2 = 0.5$ 인 두 플로우에 대한 SCFQ와 LCFQ 알고리즘의 동작예를 보인다. 여기서 P_2^1의 길이는 P_1^2 길이보다 약간 길다고 가정한다. SCFQ 알고리즘에서 $v(t)$는 $D_{priv} < t \leq D_{now}$ 동안에는 같은 값을 가지므로 그 사이에 도착한 패킷들의 가상도착시간은 같게 된다. 즉 P_2^1와 P_1^2는 같은 가상도착시간을 가진다. 이때 P_2^1는 P_1^2보다 크기 때문에 P_2^1이 P_1^2보다 먼저 도착했다고 할지라도 P_2^1보다 P_1^2가 먼저 전송된다. LCFQ 알고리즘에서는 가상시간이 선형적으로 증가하므로, P_2^1의 가상도착시간은 P_1^2의 가상도착시간보다 작게 되고 그 결과 P_1^2보다 P_2^1이 먼저 전송된다. 이러한 LCFQ의 가상시간의 선형적 특성에 의해 P_2^1의 지연시간은 SCFQ에서 보다 감소하게 된다.

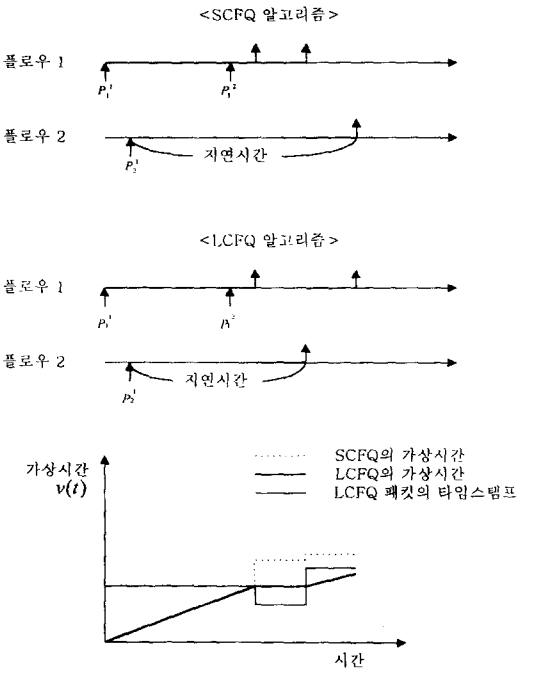


그림 2. SCFQ와 LCFQ 알고리즘의 동작예
Fig. 2. Example operations of the SCFQ and the LCFQ algorithms.

Ⅲ. LCFQ 알고리즘의 공평성

트래픽 스케쥴링 알고리즘의 공평성은 시간(t_1, t_2)사이에 연속적으로 미처리된 임의의 두 플로우에 대하여 동일한 normalized 서비스를 제공하는 것을 의미한다. 그러나 이러한 모델은 이상적인 유체흐름 모델에서만 가능하므로 실제 시스템에서는 공평성을 위해 임의의

시간(t_1,t_2)동안 연속적으로 미처리된 두 플로우간의 normalized 서비스가 다음과 같이 근사해지면 된다.

$$w_k(t_1,t_2) \approx w_l(t_1,t_2), \qquad k,l \in B(t_1,t_2).$$

또한 임의의 플로우는 새롭게 도착한 플로우에 의해 불이익을 당하지 않아야 한다. 이상의 공평성 증명을 위하여 $v_i(t)$를 시간 t까지 플로우 i가 유휴 기간 동안 제공받지 못한 서비스 $u_i(t)$와 대기하는 동안 플로우 i에 제공된 normalized 서비스 $w_i(t)$의 합으로 정의하고 플로우 i의 서비스 부족분은 다음과 같이 정의한다.

$$\delta_i(t) = v(t) - v_i(t)$$

위 정의에 따라 아래의 정리 1은 LCFQ 알고리즘에서 각 플로우의 서비스의 부족분이 제한된다는 것을 의미하며 결국 각 플로우의 normalized 서비스는 거의 같다는 것을 나타내어 LCFQ의 공평성을 증명하게 된다.

정리 1. LCFQ에서 각 플로우 k의 서비스 부족분은 다음과 같이 제한된다.

$$0 \le \delta_k(t) \le \max_{1\le i\le N}\left(\frac{L_i^{\max}}{\phi_i}\right) + \frac{L_k^{\max}}{\phi_k}, \qquad k = 1,\Lambda\ ,N. \tag{5}$$

여기서 $L_k^{\max}$는 플로우 k의 최대 패킷 크기이다.

위 정리는 다음 6개의 보조정리를 증명하므로써 증명한다.

보조정리 1. LCFQ의 가상시간 $v(t)$는 비감소형(nondecreasing) 함수이다.

증명) 식 (3)에서 $v(t)$의 정의는 $v(t)$가 비감소형(nondecreasing) 함수임을 나타낸다.

보조정리 2. 각 세션 k와 패킷 P_k^i에 대하여

$$u_k(d_k^{i-1},d_k^i) = \max\{0, v(d_k^{i-1},a_k^i)\} \quad \text{이다.} \tag{6}$$

여기서 a_k^i와 d_k^i는 각각 패킷 P_k^i의 도착과 출발시간이다.

증명) 플로우 k는 시간 (a_k^i, d_k^i)동안 미처리되어 있으므로 다음과 같이 나타낼 수 있다.

$$\begin{aligned} u_k(d_k^{i-1},d_k^i) &= u_k(d_k^{i-1},a_k^i) + u_k(a_k^i,d_k^i) \\ &= u_k(d_k^{i-1},a_k^i). \end{aligned}$$

만약 $a_k^i < d_k^{i-1}$이면, 플로우 k는 시간 (a_k^i, d_k^{i-1})동안 미처리되어있고, 그렇지 않으면 플로우 k는 (d_k^{i-1}, a_k^i)동안 비어있게 된다. 그러므로

$$\begin{aligned} u_k(d_k^{i-1},d_k^i) &= \left\{\begin{array}{ll} 0 & if\ a_k^i < d_k^{i-1} \\ v(d_k^{i-1},a_k^i) & otherwise \end{array}\right\} \\ &= \max\{0, v(d_k^{i-1},a_k^i)\} \end{aligned}$$

이라 할 수 있다.

보조정리 3. 패킷 P_k^i의 출발시간 d_k^i에서 서비스 부족분은 다음과 같이 제한된다.

$$0 \le \delta_k(d_k^i) \le \max_j\left(\frac{L_j^{\max}}{\phi_j}\right). \tag{7}$$

증명) d_k^i 시각에 $v_k(d_k^i) = F_k^i$이다.

i) 경우 1 : $F_{priv} \le F_k^i$

LCFQ의 가상시간함수의 정의로부터 $v(d_k^i) = F_k^i$이고, 그림 3(a)에 나타난 관계에 따라

$$\delta_k(d_k^i) = v(d_k^i) - v_k(d_k^i) = 0 \quad \text{이 된다.} \tag{8}$$

ii) 경우 2 : $F_{priv} \ge F_k^i$

S_{priv}와 D_{priv}는 P_k^i가 서비스 받기 바로전에 서비스 받은 패킷의 실제 서비스 시작시간과 서비스 종료 시간을 말한다. 만약 $v(S_{priv}) > F_k^i$라면 P_{priv}의 서비스가 시작되기 전에 P_k^i는 먼저 도착되어져야만 한다. 그러나 그 경우에 $F_{priv} \ge F_k^i$에 의해 P_{priv}는 P_k^i보다 먼저 전송되어지기 위해 선택되어져야 한다. 그러나 이것은 모순이 된다. 그러므로

$$v(S_{priv}) \le F_k^i \quad \text{경우만 가능하다.} \tag{9}$$

그리고

$$S_{priv} \le a_k^i \le D_{priv} \quad \text{이다.} \tag{10}$$

그러므로 그림 3(b)에 A로 나타내어진 타임 스탬프 값의 차이$(F_{priv} - F_k^i)$는 normalized 서비스 L_{priv}/ϕ_{priv}보다 작아져야 한다. 즉,

$$\delta_k(d_k^i) = v(d_k^i) - v_k(d_k^i)$$
$$= F_{prev} - F_k^i \le \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\} \text{ 가 된다.} \quad (11)$$

경우 1과 경우 2로부터 보조정리 3은 증명된다.

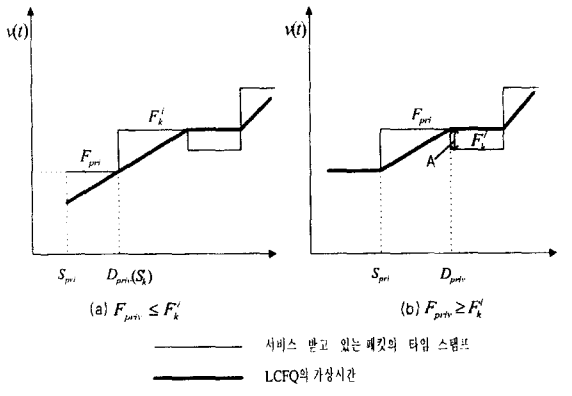


그림 3. LCFQ의 가상시간 함수와 타임 스탬프 값
Fig. 3. Sample path of the virtual time of LCFQ.

보조정리 4. 플로우 k가 비어있게 될 때 세션 k의 서비스 부족분은 다음 식에 나타낸 것 처럼 마지막 패킷인 P_k^i가 떠나는 시점에서의 서비스 부족분과 같게 된다.

$$\delta_k(t) = \delta_k(d_k^i) \left(\le \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\} \right), \quad for \quad d_k^i \le t \le a_k^{i+1}. \quad (12)$$

증명) 플로우 k가 비어있게 될 때, 그 세션의 가상시간 $v_k(t)$는 시스템 가상시간 $v(t)$의 증가량을 추적한다.

보조정리 5. LCFQ 알고리즘에서 각 순간에 임의의 플로우는 미처리된다. 그 결과 서비스의 부족분은 '영'이다.

$\delta_k(t) = 0$, 플로우 k는 시간 t에서 미처리된다.

증명) 임의의 플로우가 시간 t에서 미처리될 때, 플로우를 미처리되게 만든 패킷은 시스템 가상시간 $v(t)$를 사용하여 타임 스탬프된다. 그 결과 서비스 부족분은 '영'이 된다.

보조정리 6. 플로우 k가 시간 t에서 미처리될 때, 그 플로우의 서비스 부족분은 다음과 같이 제한된다.

$$0 \le \delta_k(t) \le \max_j \left(\frac{L_j^{\max}}{\phi_j} \right) + \frac{L_k^i}{\phi_k}. \quad (13)$$

여기서 P_k^i는 t 시간 이후에 플로우 k의 패킷중 가장 먼저 서비스가 끝날 패킷이다.

증명) $b_k^i = \max\{d_k^{i-1}, a_k^i\}$ 라 가정하자. 식을 전개하면 다음과 같다.

$$\delta_k(t) = \delta_k(b_k^i) + \delta_k(b_k^i, t)$$
$$= \delta_k(b_k^i) + v(b_k^i, t) - v_k(b_k^i, t). \quad (14)$$

i) $d_k^{i-1} > a_k^i$에 대하여, 즉 $b_k^i = d_k^{i-1}$이다.

$$\delta_k(t) = \delta_k(d_k^{i-1}) + v(d_k^{i-1}, t) - v_k(d_k^{i-1}, t$$
$$\le \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\} + \frac{L_k^i}{\phi_k} - v_k(d_k^{i-1}, t)$$
$$\le \max_j \left\{ \frac{L_j^{\max}}{\phi_k} \right\} + \frac{L_k^i}{\phi_k}. \quad \text{보조정리 4로부터} \quad (15)$$

ii) $d_k^{i-1} < a_k^i$에 대하여, 즉 $b_k^i = a_k^{i-1}$

보조정리 3으로부터

$$v(d_k^i) \le F_k^i + \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\}$$
$$= v(d_k^i) + \frac{L_j^{\max}}{\phi_j} + \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\}. \quad (16)$$

그러므로,

$$v(a_k^i) = \frac{L_k^i}{\phi_k} + \max_j \left\{ \frac{L_j^{\max}}{\phi_j} \right\}. \quad (17)$$

식 (15)를 이용하여,

$$\delta_k(t) = \delta_k(a_k^i) + v(a_k^i) + v(a_k^i, t) - v_k(a_k^i, t)$$
$$\le v(a_k^i, t) \ ; \text{(보조정리 5로부터, } \delta_k(a_k^i) = 0)$$
$$\le v(a_k^i, d_k^i)$$
$$\le \max_j \left(\frac{L_j^{\max}}{\phi_k} \right) + \frac{L_k^i}{\phi_k}. \quad (18)$$

i)과 ii)로부터 보조정리 6이 증명된다.

보조정리 3, 4, 5, 6에 의해 증명된 정리 1은 LCFQ 알고리즘에서의 가상시간과 임의의 미처리된 플로우의 normalized 서비스와의 차이가 제한된다는 것을 나타낸다. 또한 미처리된 임의의 두 플로우가 받은 normalized 서비스가 제한된다. 그러므로 LCFQ

는 페어 큐잉 알고리즘이다.

Ⅳ. 시뮬레이션 결과

LCFQ 알고리즘의 최대 지연시간의 이론적 해석은 가상시간의 선형증가 특성으로 용이하지가 않으므로 본 논문에서는 시뮬레이션을 통하여 SCFQ와 LCFQ 알고리즘에 대한 각각의 평균과 최대지연시간을 고찰하였다. 표 1과 그림 4는 시뮬레이션을 통한 이들값을 비교한 것이다. 사용한 시뮬레이션 모델은 그림 1과 동일하며 사용변수값은 다음과 같다. 패킷 크기는 53 바이트, 플로우는 8개의 CBR(constant bit rate) 플로우를 사용하였으며, 각 플로우의 트래픽 발생 시작 시간은 0.0초에서 0.7초 사이의 서로 다른 시점에서 시작하도록 하였다. 서버의 서비스 속도는 1Mbps로 하였고, 각 플로우들의 트래픽 발생률은 플로우 1을 제외한 나머지 7개의 플로우에 대해서 자신의 플로우에 할당된 대역폭내에서 발생시키도록 하였다. 그 결과 플로우들의 트래픽 발생 속도의 총합은 서버의 서비스 속도보다 약간 높은 1.025Mbps가 되게 하였다. 본 시뮬레이션에서 플로우 1은 자신에게 할당된 트래픽 발생률보다 더 많은 트래픽을 발생시키는 플로우로 작용하도록 했다. 이러한 플로우를 사용한 목적은 자신의 할당 속도를 넘지 않게 트래픽을 발생시키는 플로우들이 플로우 1에 의해 받게 되는 영향을 살펴보기 위한 것이다. 즉, 플로우 1을 사용하므로서 SCFQ와 LCFQ에서의 독립화 특성을 비교할 수가 있게 된다. 표 1에는 각 플로우에 대한 할당 대역폭과 도착 속도를 나타내었고, SCFQ와 LCFQ하에서 각 플로우들의 평균지연시간을 보인다. 표 1에 나타낸 평균지연시간에 대한 결과를 보면 플로우 1을 제외한 나머지 플로우의 평균지연시간은 SCFQ하에서 보다 LCFQ하에서 더 작다는 것을 볼 수가 있다. 이러한 결과는 SCFQ와 LCFQ에 대한 독립화 특성을 보여주는 것으로서 LCFQ 알고리즘의 독립화 특성이 SCFQ의 독립화 특성보다 좋다는 것을 입증시켜주는 결과이다. 또한 그림 4는 SCFQ와 LCFQ에서의 플로우 2와 6의 절대적인 지연시간에 대한 결과이다. 이 그림에서 살펴보면 SCFQ하에서 보다 LCFQ하에서의 플로우의 최대지연시간이 더 작음을 볼 수 있다. 또한 지연시간의 편차에 있어서도 역시 LCFQ가 더 작은 것을 알 수 있다. 시뮬레이션에서 플로우 1 이외의 플로우에서 loss는 발생되지 않도록 충분한 적정 크기의 버퍼를 설정하였다. 그러나 플로우 1은 할당 대역폭 이상으로 입력되는 트래픽이고 사용하는 버퍼가 유한 버퍼이므로 버퍼가 계속 쌓여나가다 이어서 많은 loss가 발생된다. 이것은 또 다른 policing 효과로서 loss 결과는 나타내지 않았다. 이에의해 다른 플로우들이 보호받게 된다.

표 2는 burst한 입력 트래픽에 대한 시뮬레이션 결과로서 입력 트래픽은 on-off source모델을 이용하였다. 즉 on 기간 동안에는 일정 속도로 패킷이 발생되고, off 기간에는 발생되지 않도록 하였으며 각 on, off기간은 지수 분포를 따르게 했다. 그 결과 CBR에서와 유사한 결과를 얻을 수 있었다.

본 시뮬레이션 결과로 LCFQ 알고리즘은 SCFQ 알고리즘 보다 독립화 특성이 좋은 것을 알 수 있고, 지연시간에 있어서도 LCFQ 알고리즘이 더 감소된 최대지연시간과 평균지연시간을 제공하는 것을 알 수 있다.

표 1. SCFQ와 LCFQ에서 평균 지연 시간 (CBR 트래픽)

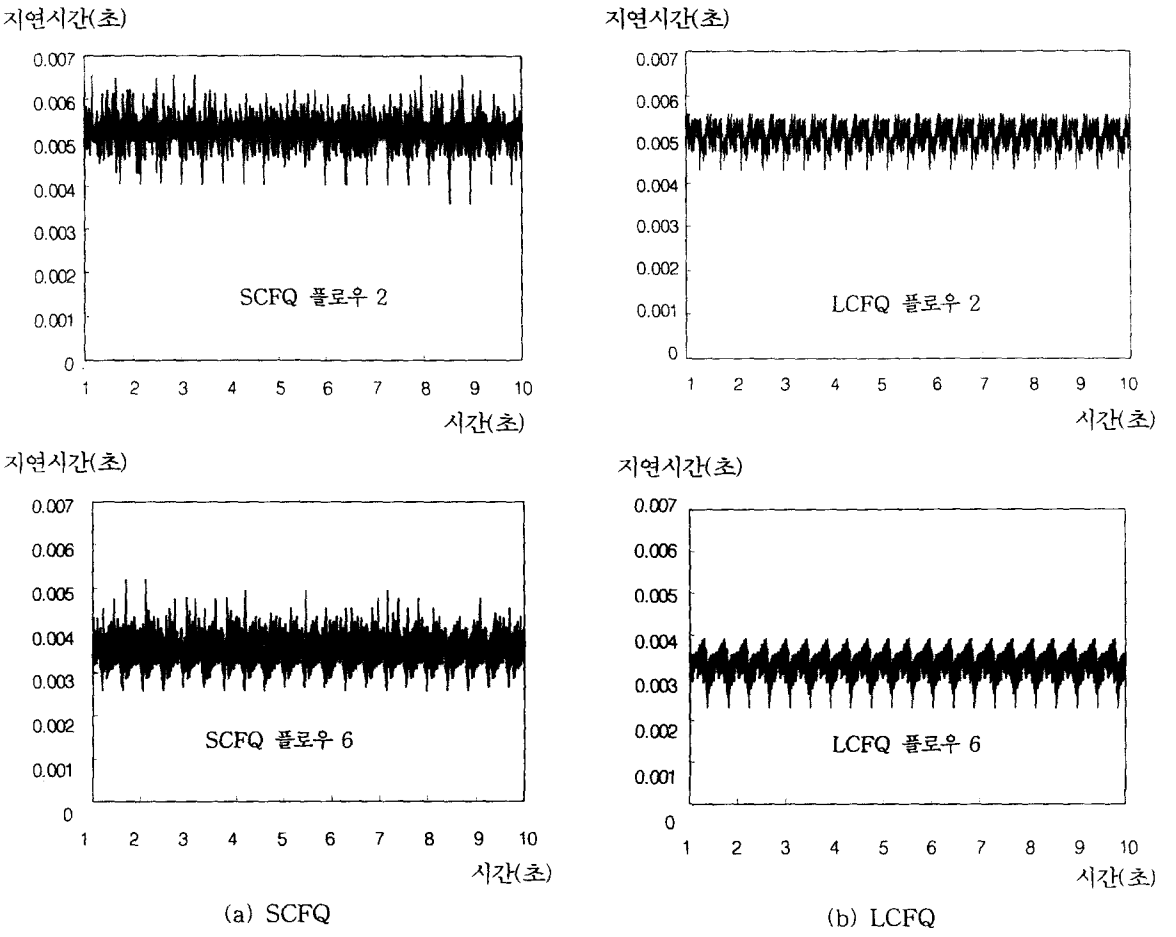
Table 1. Average delays of the SCFQ and the LCFQ algorithms(CBR traffic).

플로우	할당 대역폭 (Mbps)	패킷 도착 속도(Mbps)	평균 지연 시간 (초)	
			SCFQ	LCFQ
0	0.5	0.498	0.000968	0.000735
1	0.0625	0.100	1.153755	1.155240
2	0.0625	0.062	0.005249	0.005092
3	0.0625	0.061	0.005249	0.004992
4	0.078125	0.076	0.004234	0.004044
5	0.078125	0.076	0.003784	0.003618
6	0.078125	0.076	0.003531	0.003293
7	0.078125	0.076	0.003400	0.002929

표 2. SCFQ와 LCFQ에서 평균 지연 시간 (on-off 트래픽)

Table 2. Average delays of the SCFQ and the LCFQ algorithms(on-off traffic).

플로우	할당 대역폭 (Mbps)	패킷 도착 속도(Mbps)	평균 지연 시간 (초)	
			SCFQ	LCFQ
0	0.5	0.493748	0.123798	0.123539
1	0.0625	0.095062	0.729414	0.733936
2	0.0625	0.061911	0.078310	0.077068
3	0.0625	0.060230	0.080871	0.079346
4	0.078125	0.075918	0.088121	0.087915
5	0.078125	0.068061	0.043720	0.043211
6	0.078125	0.077101	0.076622	0.076006
7	0.078125	0.075870	0.119420	0.118924



(a) SCFQ (b) LCFQ

그림 4. SCFQ 와 LCFQ에서의 지연 시간
Fig. 4. The delay of SCFQ and LCFQ

V. 결 론

본 논문에서 우리는 선형적으로 증가하는 가상시간 함수를 가진 새로운 LCFQ 페어 큐잉 알고리즘을 제안하였다. 제안된 LCFQ 알고리즘에 대하여 공평성을 증명하였고, 시뮬레이션을 통하여 성능을 분석하였다. 분석 결과 LCFQ 알고리즘은 공평하고 또한 SCFQ 알고리즘보다 최대 및 평균지연시간이 개선됨을 보였다. 또한 예약된 트래픽의 양보다 더 많은 트래픽을 발생시키는 플로우에 영향을 받지않는 독립화 특성이 SCFQ에 비해 좋음을 알 수 있었다. LCFQ 알고리즘은 유체 흐름 시스템을 흉내내지 않고 가상시간을 계산하기위해 현재 서비스 받고 있는 패킷의 타임 스탬프 값을 바탕으로 가상시간을 계산하기 때문에 쉽게 구현될 수 있어 고속 종합 서비스망에 적합한 트래픽 스케쥴링 방식이라 할 수 있다.

참 고 문 헌

[1] Integrated Service : RFC 1633

[2] Parekh A. K. and Gallager R. G., "A generalized processor sharing approach to flow control in intergrated services networks : The single node case", *IEEE/ACM Trans. Networking*, June 1993, vol. 1, no. 3, pp. 344-357.

[3] Golestani S. J., "A self-clocked fair queueing scheme for broadband applications", *IEEE Proc. INFOCOM*, 1994,

pp. 636-646.

[4] Golestani S. J., "New delay analysis of a class of fair queueing algorithm", *IEEE J. Select. Areas Commun.*, Aug. 1995, vol. 13, no. 6, pp. 1057-1070.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm.", In *Proc. ACM SIGCOMM'89*, pp. 3-12.

[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*, pages 19-29, Philadelphia. PA. 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] J. C. R. Bennett and H. Zhang, "WF^2Q: Worst-case fair weighted fair queueing.", In *Proceedings of IEEE INFOCOM'96*, pp. 102-128. San Francisco, CA, March 1996.

[10] 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.

[11] J. C. Bennett H. Zhang, "Hierarchical Packet Fair Queueing Algorithms.", *Proc. ACM SIGCOMM'96*, Sept. 1996, pp. 143-156.

[12] S. S. Lam and G. G. Xie, "Group Priority Scheduling", *IEEE/ACM Transaction on Networking*. vol. 5, no. 2, April 1997.

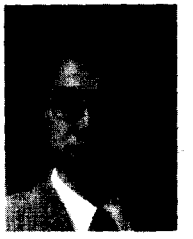
[13] A. Varma and D. Stiliadis, "Hardware Implementation of Fair Queueing Algorithm for ATM Network", *IEEE Communication Magazine*, December 1997.

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

저 자 소 개

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 조교수. 주관심분야는 컴퓨터네트워크, 인터넷 네트워킹, ATM, 이동 데이터 통신망 등임



李 在 容(正會員)

1965년 2월 10일생. 1983년 3월 ~ 1988년 2월 서울대학교 전자공학과 (학사). 1988년 3월 ~ 1990년 2월 한국과학기술원 전기 및 전자공학과 (석사). 1990년 3월 ~ 1995년 2월 한국과학기술원 전기 및 전자공학과 (박사). 1990년 1월 ~ 1995년 8월 디지콤정보통신연구소(선임연구원). 1995년 9월 ~ 현재 충남대학교 정보통신공학과(조교수). 주관심분야는 ATM과 인터넷 트래픽 제어, 인터넷프로토콜, 통신시스템 성능분석 등임