Journal of Korea Multimedia Society Vol. 10, No. 1, January 2007(pp. 83-93)

최적 레이턴시 기반 공정 큐잉 방식의 개선에 관한 연구

김 태 준†

요 약

널리 이용되고 있는 공정 큐잉 방식인 WFQ(Weighted Fair Queuing)는 특히 인터넷 전화와 같이 저속이지만 엄격한 지연한계를 요구하는 서비스에 대해 대역폭 이용도가 저하되는 문제점을 갖고 있다. 이러한 WFQ의 문제점은 흐름의 레이턴시가 최적화 되지 못하기 때문에 발생하는 것으로 최근에 밝혀졌고, 이어서 최적 레이턴시 기반 공정 큐잉 방식인 LOFQ(Latency-Optimized Fair Queuing)가 도입되었다. 본 연구에서는 LOFQ에 점유자원 최적화 기능을 추가하여 성능특성을 더욱 개선하고, 반복적으로 수행되던 LOFQ의 자원변환 알고리즘을 개선하여 수행 복잡성을 줄인다. 아울러 WFQ에 비해 대역폭 이용도가 우수함을 증명한다. 시뮬레이션을 통해 성능개선 효과를 확인한 결과 20~30%의 개선이 있었다.

A study on Improving Latency-Optimized Fair Queuing Algorithm

Tae Joon Kim†

ABSTRACT

WFQ (Weighted Fair Queuing) is the most popular fair queuing algorithm, but it had the inherent drawback of a poor bandwidth utilization, particularly under the traffic requiring a low rate but tight delay bound such as internet phone. It was recently identified that the poor utilization is mainly due to the non-optimized latency of a flow and then LOFQ(Latency-Optimized Fair Queuing) to overcome the drawback was introduced. In this paper, we improve the performance of LOFQ by introducing an occupied resource optimization function and reduce the implementation complexity of recursive resource transformation by revising the transformation scheme. We also prove the superiority of LOFQ over WFQ in terms of utilization. The simulation result shows that the improved LOFQ provides 20 ~ 30 % higher utilization than that in the legacy LOFQ.

Key words: WFQ(WFQ), Fair Queuing(공정 큐잉), Delay Resource(지연 자원), Optimal Latency(최적 레이턴시), LOFQ(LOFQ)

1. 서 론

인터넷 전화와 같이 서비스 품질(QoS: Quality of Service)의 보장을 요구하는 실시간 멀티미디어 통신 서비스를 수용하기 위해 IETF(Internet Engineer Task Force)에서 자원예약 기반의 종합서비스(IntServ) 모델을 제시하였다[1]. IntServ 모델에서 품질 보장형 서비스를 지원하기 위해 자원예약 기반의 RSVP(Reservation protocol)-라우터를 도입하였는데, 트래픽 흐름의 요구 속도를 보장하고 허용되는 레이턴시(latency), 즉 지연(delay) 규격을 위반하지 않기 위해 RSVP-라우터는 공평 패킷 스케줄러를 탑재한다.

유체모델 기반의 이상적 패킷 스케줄링 방식인 GPS(General Processor Sharing)[2]가 연구되었고, GPS는 WFQ(Weighted Fair Queuing)[3]에 의해 구

※ 교신저자(Corresponding Author) : 김태준, 주소 : 충남 천안시 부대동 275번지(330-717), 전화 : 041)550-0209, FAX : 041)556-6447, E-mail : tjkim@kongju.ac.kr

접수일 : 2006년 8월 9일, 완료일 : 2006년 11월 20일

† 정회원, 공주대학교 정보통신공학부

현되었다. WFQ는 흐름간의 상호 간섭을 차단하여 각 흐름의 품질 특성이 다른 흐름으로부터 영향을 받지 않도록 하는 흐름 분리와 차별화된 품질 보장을 지원하는 등 공정 큐잉의 요구사항을 충실히 따르기 때문에 IETF IntServ 모델의 RSVP-라우터에 적용되었다[4]. 그러나 흐름의 스케줄링 속도를 높일 경우 스케줄링 속도와 요구속도와의 차이 만큼 속도(대역폭)자원이 손실되는데, 특히 인터넷 전화와 같이 트래픽 흐름의 속도는 낮지만 엄격한 종단간 지연 특성을 요구하는 트래픽 흐름의 경우 대역폭 이용도의 저하가 심각해짐이 밝혀졌다[5].

WFQ의 이용도 저하 문제를 해결하기 위해 다양한 연구가 이루어졌다. 음성 트래픽의 지원을 위해 감시기를 갖는 WFQ가 제안되었는데[6], 이는 음성 흐름의 스케줄링 속도를 다소 과다하게 할당하는 대신 필요시 감시기가 패킷을 드롭(drop)하는 방식을 도입하였다. 한편 가변 스케줄링 속도를 사용하는 최적 네트워크 서비스 곡선 방식[7]과 크레디트(Credit) 기반 프로세서 공유방식[8]이 제안되었는데, 이는 일시적으로 예약된 속도보다 높은 스케줄링 속도를 요구하는 흐름들에 대해서 대역폭에 여유가 있는 다른 흐름의 대역폭을 빌려주는 방식이다. 참고로 본 논문에서 속도(rate)와 대역폭(bandwidth)는 같은 의미로 사용된다. 그러나 이러한 방식은 빌려줄 만큼 대역폭의 여유를 가진 흐름이 줄어들 경우 그 효과가 반감되는 문제가 있다. 최근에 WFQ의 고질적인 문제의 근원은 레이턴시가 최적화 되지 않아서 발생하는 지연자원의 낭비 때문으로 밝혀졌고, 그러한 낭비를 방지하기 위해 레이턴시가 최적화되는 LOFQ(Latency-Optimized Fair Queuing) 방식이 연구되었다[9].

본 연구는 기 제안된 LOFQ 방식을 개선하기 위한 것이다. 먼저 점유된 자원의 최적화 기능을 새로 추가하여 더 많은 흐름을 수용하고, 그리고 지연자원과 대역폭 자원 사이의 자원 변환시 변환될 자원 량을 알 수 없어 미소량 단위로 반복적으로 수행되는 LOFQ의 자원변환 알고리즘을 개선하여 수행 복잡성을 줄인다. 아울러 WFQ에 비해 대역폭 이용도 특성이 우수함을 해석적으로 증명한다. 본 논문의 구성은 다음과 같다: 2장에서 관련 연구를 소개하고, 3장에서 개선된 방식을 기술하며, 4장에서 개선된 방식의 이론적인 근거를 제시하고 WFQ와의 대역폭 이용도를 해석적으로 비교한다. 그리고 5장에서 성능을 평가하고 6장에서 결론을 맺는다.

2. 관련 연구

먼저 스케줄러에 도착하는 흐름의 속성을 나타내는 흐름 규격 F(r,b,M)를 정의하는데, r은 흐름의 요구속도, b는 스케줄러에 할당되는 흐름의 지연예산(delay budget), M은 흐름의 최대 패킷크기이다. 흐름의 지연예산이란 종단간 지연한계가 각 노드에 분배된 것으로 본 연구에서는 간단히 지연규격이라 칭한다. 참고로 공정 큐잉 알고리즘은 흐름의 요구속도 r과 지연규격 b를 만족시키기 위한 것이다. WFQ에서 흐름 규격 F(r,b,M)을 갖는 임의 흐름의 고유 레이턴시 q^r은 다음과 같이 주어진다[2].

$$q^r = \frac{M}{r} + \frac{M^{max}}{C^B}, \qquad (2.1)$$

여기서 C^B는 출력링크의 용량으로 스케줄러의 총 대역폭을 의미하고, M^{max}는 모든 흐름의 최대 패킷크기이다. WFQ는 요구속도와 지연규격을 동시에 만족시켜야 하므로 임의 흐름의 스케줄링 속도 s는 (2.1)로부터 다음과 같이 계산된다.

$$s = \max(r, r^C), \text{ 여기서 } r^C = \frac{MC^B}{bC^B - M^{max}}. \qquad (2.2)$$

(2.2)내 r^C를 임계속도라 한다. 그러면 WFQ에서 임의 흐름의 실제 레이턴시 q는 다음과 같이 계산된다.

$$q = \frac{M}{s} + \frac{M^{max}}{C^B}. \qquad (2.3)$$

참고로 WFQ의 경우 (2.2)와 (2.3)으로부터 $q \le b$가 된다. (2.3)에서 $M^{max}/C^B << M/s$ 이므로 WFQ에서 각 흐름의 레이턴시와 스케줄링 속도는 서로 반비례하는 결합관계를 갖는다. 이러한 결합관계가 바로 WFQ의 성능저하의 주된 요인으로 이해되어 왔으나[5-8], 최근에 다음과 같이 지연자원의 낭비에 기인하는 것으로 밝혀졌다[9]. (2.1)로부터 고유 레이턴시 q^r와 지연규격 b는 $q^r=b$, $q^r>b$ 또는 $q^r<b$의 관계를 갖는다. $q^r=b$의 경우 자원의 손실 없이 흐름을 수용할 수 있다. $q^r>b$의 경우 여유 대역폭 자원이 있다면 스케줄링 속도를 높여서 지연규격을 충족, 즉 $q=b$가 되게 해야 한다. 이때 스케줄링 속도와 요구속도의 차이분 만큼 대역폭 자원이 손실되지만 지연 품질

을 만족시키기 위해 피할 수 없다. $q^r<b$의 경우 q=b가 되게 하려면 스케줄링 속도를 요구속도 보다 낮게 해야 한다. 하지만 흐름의 요구속도를 보장해야 하기 때문에 스케줄링 속도를 낮출 수 없어 지연규격 보다 더 엄격한 레이턴시를 갖게 된다. 이 결과 (b−q) 만큼의 지연 자원이 손실되어 스케줄러 자원 이용 효율이 저하된다는 것이다. 그리고 이러한 지연자원의 낭비를 근본적으로 차단하는 LOFQ가 연구되었다[9].

LOFQ는 레이턴시 지수 β와 레이턴시 바이어스(bias) D를 도입한 후 (2.4)와 같이 각 흐름의 레이턴시 q를 β와 스케줄링 속도 h의 함수가 되게 하였다. 여기서 $0\le\beta$ 및 $0\le D\le D^B$이며, D^B는 흐름 중 가장 엄격한 지연규격 값이다. 그리고 β와 h를 조정하여 q를 지연규격 b와 동일하게 하는데, 대역폭 점유를 최소화 하기 위해 h를 우선적으로 줄인다. 참고로 h란 바로 그 흐름이 점유하는 대역폭이 된다.

$$q = D^B + \beta\frac{M}{h} = b. \tag{2.4}$$

레이턴시 바이어스 D는 흐름을 많이 수용할수록 증가하지만 그 값이 D^B를 초과할 수 없다. 따라서 대역폭(BW: BandWidth)과 마찬가지로 레이턴시 바이어스는 일종의 자원으로 취급될 수 있는데, 이를 지연자원(DW: Delay Width)이라고 한다. 그리고 D^B와 D를 각각 스케줄러의 DW 용량과 스케줄러의 점유 DW라 한다. 그리고 β의 레이턴시 지수를 갖는 흐름에 의해 증가되는 레이턴시 바이어스 값, 즉 그 흐름의 점유 DW가 되는 d는 다음과 같이 주어졌다.

$$d = \max(0,1-\beta)M/C^B. \tag{2.5}$$

그리고 스케줄러의 점유 DW, 즉 레이턴시 바이어스 D는 다음과 같이 주어졌다.

$$D = \max\left(\frac{M^{\max}}{C^B}, \frac{\sum_{B(t)}\max(0,1-\beta)M}{C^B}\right), \tag{2.6}$$

여기서 B(t)는 수용된 흐름 집합이다.

(2.4)와 (2.5)로부터 흐름의 점유 BW h와 점유 DW d는 서로 반 비례 하므로, 점유 BW를 늘리는 대신 점유 DW를 줄일 수 있고, 반대로 점유 DW를 늘리는 대신 점유 BW를 줄일 수 있는, 즉 DW와 BW가 상호 변환될 수 있음을 알 수 있다. 여기서 전자를 B2D 변환, 후자를 D2B 변환이라 한다. 그리고 늘어나는 BW 대비 줄어드는 DW의 비율을 B2D 변환 효율 E^{B2D}, 늘어나는 DW 대비 줄어드는 BW의 비율을 D2B 변환 효율 E^{D2B}라 한다. WFQ의 경우 스케줄러의 점유 BW가 C^B에 도달하면 더 이상 흐름을 수용할 수 없다. 하지만 LOFQ의 경우 다음과 같이 자원변환을 통해 대역폭 이용도를 높인다: 새로운 흐름 수용에 필요한 DW와 BW가 모두 부족하면 흐름 수락을 중단한다. 하지만 DW는 부족하나 BW는 충분할 경우 B2D 변환을 통해 부족한 DW를 확보하고, 반대로 BW는 부족하나 DW는 충분할 경우 D2B 변환을 통해 부족한 BW를 확보한 후 그 흐름을 수락한다.

3. LOFQ 개선

3.1 개선방향

[9]에서 연구된 LOFQ는 초기버전 (이하 LOFQ1)으로서 새로운 흐름의 수락 요청시 BW와 DW가 모두 충분하지 않을 경우 수락을 거절하는데, 본 연구에서는 스케줄러의 점유자원 최적화를 통해 더 많은 흐름을 수락하도록 한다. 그리고 LOFQ1은 D2B 변환과 B2D 변환시 변환될 자원 량을 알 수 없어서 미소량 단위로 반복적으로 변환동작을 수행하는데, 이로 인해 자원변환의 수행 복잡도가 너무 크다는 것이다. 그래서 본 연구에서는 변환될 수 있는 최대 자원 량을 산정하여 자원변환의 동작 횟수를 최소화 한다.

3.2 개선된 알고리즘

개선된 LOFQ (이하 LOFQ2)는 LOFQ1과 마찬가지로 각각 스케줄러의 점유 BW와 DW를 나타내는 두 레지스터 C와 D를 가지며 수락된 임의 흐름 i는 흐름 프로파일 $P_i(r,b,h,\beta,M)$를 할당 받는다. 참고로 P()내 변수의 아래첨자는 생략될 수 있다. 흐름 수락제어와 패킷 스케줄링 기능으로 구성되는 LOFQ1에 점유자원 최적화 기능이 추가되고, 흐름 수락제어 기능이 일부 보완된다.

1) 흐름 수락제어 기능

LOFQ2의 흐름 수락제어 기능에 대한 c언어 형태의 알고리즘이 그림 1에 정리되어 있다. $F(r_i,b_i,M_i)$를 갖는 임의 흐름 i에 대한 수락요청이 입력되면 수락제어 기능이 동작한다. 먼저 스케줄링 속도 h_i를 WFQ에서의 속도인 s_i로 설정하고 레이턴시 지수 β_i

```
Admit() { //*** 흐름 수락제어 기능 ***
h_i = s_i and β_i = (b_i − D^B)h_i/M_i                          //β 계산
if(β_i = 0) h_i = r_i                                          //h 조정 및 β갱신
if(β_i > 1) h_i = max(r_i, M_i/(b_i − D^B)) and β_i = (b_i − D^B)h_i/M_i
if(C + h_i > C^B and (D + d_i) > D^B) return fail              //모든 자원 부족이면 거절
else if((D + d_i) > D^B) if (B2DT(1, D + d_i − D^B,−1) return fail   //DW만 부족이면 B2D변환
else if(C + h_i > C^B) if(D2BT(−1, C + h_i - C^B) return fail   //BW만 부족이면 D2B변환
P_i(r, b, h, β, M) = P_t(r_i, b_i, h_i, β_i, M_i)              //수락되면 프로파일 할당
C+ = h_i and D+ = d_i                                          //점유 BW 및 DW 갱신
}

D2BT(j,ΔB) { // D2B 변환기능: DW 늘리는 대신 ΔB의 BW 줄임
While(ΔB>0) {
if(j≥0) j=−1 and j = {k | max_{k∈B(t), r_k<R_k and 0<β_k≤1} E_k^{D2B}}   //흐름 지정 안되면 효율 순서
if(j<0 or Δβ_j^B ≤ 0) return fail                              //실패시 에러 리턴
Δh_j = h_j Δβ_j^B/β_j and ΔB− = Δh_j                           //BW 감소분 계산, ΔB갱신
C− = Δh_j and D+ = M_j Δβ_j^B/C^B                              //점유 BW와 DW 갱신
P_j(r, b, h, β, M) = P_j(r, b, h − Δh_j, β − Δβ_j^B, M)        //흐름 j 프로파일 갱신
  }
}

B2D(j,ΔD,flag) { // B2D 변환기능: BW 늘리는 대신 ΔD의 DW 줄임, flag≥0이면 C>C^B허용
While(ΔD>0) {
if(j≥0) j=−1 and j = {k | max_{k∈B(t) and 0<β_k<1} E_k^{B2D}}   //흐름 지정 안되면 효율 순서
if(j<0 or Δh_j^B ≤ 0) return fail                              //실패시 에러 리턴
Δh_j^B = Δh_j^{B1} and if(flag >=0) Δh_j^B = Δh_j^{B2}
Δβ_j = β_j Δh_j^B/h_j and ΔD− = M_j Δβ_j/C^B                   //β 증가분 계산, ΔD갱신
C+ = Δh_j^B and D− = M_j Δβ_j/C^B                              //점유 BW와 DW 갱신
P_j(r, b, h, β, M)=P_j(r, b, h + Δh_j^B, β + Δβ_j, M)          //흐름 j 프로파일 갱신
  }
}
```

그림 1. 흐름 수락제어 알고리즘

를 계산한 후 $\beta_i \notin (0,1]$ 이면 h_i가 최소가 되도록 조정한 후 β_i를 갱신한다. 이는 앞에서 언급한대로 스케줄링 속도를 우선적으로 줄이기 때문이다. BW와 DW 모두 $P_i(r,b,h,\beta,M) = P_t(r_i,b_i,h_i,\beta_i,M_i)$ 부족하면 수락을 거절하고, 아니면 필요시 자원변환 후 수락한다. 그림 1에서 $P_t()$는 임시 프로파일이다. 참고로 LOFQ 스케줄러는 항상 새로운 흐름을 수락하기 전에 점유자원 최적화를 완료한다.

그림 1과 같이 수락제어 기능은 각각 B2D 변환과 D2B 변환을 위한 두 서브알고리즘 B2DT()와 D2BT()을 포함한다. 이들 알고리즘은 4장의 보조정리 1로부터 도출된 것으로 LOFQ1과 달리 각 흐름에 대해 한번의 변환 동작만 일어난다. 그리고 B2D 변환에서 투입되는 BW 량 Δh_j^B과 D2B 변환에서 감소되는 레이턴시 지수 $\Delta\beta_j^B$량은 각각 보조정리 3과 4에서 분석한다.

수락된 임의 흐름 i에 대한 해제 요청 시 다음과 같이 흐름의 점유 BW와 DW를 반환한다: $C = -h_i$과

$D- = d_i$. 그리고 흐름의 프로파일을 제거한다.

2) 점유자원 최적화 기능

새로 추가되는 기능으로서 먼저 점유자원 최적화에 대해 살펴본다. 점유자원 최적화란 스케줄러에 아무런 영향을 주지 않고 스케줄러의 점유 BW 또는 DW를 최소화 하는 것으로서 구체적인 이론적 근거는 보조정리 5와 6에 기술되어 있으므로 여기서는 예를 통해서 그 의미를 살펴본다. P_i(10Kbps,61ms, 10Kbps,0.6,1000bit)인 흐름 i와 P_j(10Kbps,21ms, 30Kbps,0.6, 1000bit)인 흐름 j를 생각해보자. 흐름 i의 경우 B2D 변환으로 β_i를 최대 0.4만큼 늘릴 수 있는데 이로 인해 줄어드는 흐름 i의 점유 DW는 $400/C^B$ 이고 늘어나는 점유 BW는 $6/(100C^B)*400/C^B$ =24Hz가 된다. $0.4M_i/C^B$의 DW를 흐름 j에 투입하여 D2B 변환을 하면 흐름 j의 줄어드는 BW는 $(400/C^B)*50C^B$=2000Hz가 된다. 이 결과 스케줄러의 점유 DW 변화 없이 점유 BW를 1976Hz 만큼 줄이게 된다.

최적화 기능에 대한 c언어 형태의 알고리즘이 그림 2에 정리되어 있다. 이 알고리즘은 보조정리 5에서 도출된 것으로 점유 BW를 최소화 하는 것이다. 이는 가장 효율이 좋은 B2D 변환가능 흐름 i에 대한 B2D 변환에서 Δd_i^B의 DW를 확보한 후 이를 가장 효율이 좋은 D2B 변환가능 흐름 j에 투입하여 $\Delta d_i^B E_j^{D2B}$의 BW를 확보하는 것으로 확보하려는 DW 량 Δd_i^B에 대한 분석은 보조정리 7에 기술된다. 참고로 $C=C^B$인 상태에서도 점유자원 최적화가 가능하도록 B2D 변환시 스케줄러의 점유 BW의 제약, 즉 $C \le C^B$의 조건을 무시하도록 한다.

3) 패킷 스케줄링

패킷이 도착할 때 동작하는 패킷 큐잉 기능과 흐름들의 큐에 대기중인 패킷을 출력링크를 통해 전송하는 패킷 큐 서비스 기능으로 구성되는 패킷 스케줄링 기능은 LOFQ 1과 동일하며 다만 임의 흐름 I의 k번째 패킷 P_i^k의 타임스탬프 T_i^k의 계산식을 다음과 같이 수정한다 (당초 β_i에서 $\max(0,1-\beta_i)$로 수정).

$$T_i^k = \max(F_i^{k-1}, v(t)) + \max(0,1-\beta_i) l_i^k / h_i, \qquad (3.1)$$

여기서 F_i^{k-1}은 (k−1)번째 패킷의 예상도 전송 종료 시간, l_i^k는 P_i^k의 크기, v(t)는 서버 가상시간이다.

```
While() {
  i = {k | max_{k∈E(t) and 0<β_k<1} E_k^{B2D}}
  j = {k | max_{k∈E(t), r_k<R_k and 0<β_k≤1} E_k^{D2B}}
  if(E_i^{D2B} *E_j^{B2D} ≤1) stop;
  B2DT(i, Δd_i^B,1)        //BW 제약 없이 B2D 변환
  D2BT(j, Δd_i^B E_j^{D2B})   //DW D2B 변환
}
```

그림 2. 점유자원 최적화 알고리즘

4. 성능특성 분석

4.1 기본 특성

흐름 i의 점유 DW 변화 량을 Δd_i, 점유 BW 변화 량을 Δh_i로 표기한다. 먼저 스케줄러에 수용된 각 흐름 내부에서 BW와 DW간의 상호변환, 즉 흐름 내(intra-flow) 자원변환과 그의 효율을 살펴보자.

보조정리 1: 임의 D2B 변환가능 흐름 i가 D2B 변환을 통해 자신의 점유 DW를 Δd_i 만큼 증가시키는 대신 자신의 점유 BW를 Δh_i만큼 줄일 수 있다. 여기서 $\Delta h_i = E_i^{D2B} \Delta d_i$ 및 $E_i^{D2B} = C^B/(b_i - D^B) = h_i C^B/(\beta_i M_i)$ 이다. 그리고 임의 B2D 변환가능 흐름 i도 B2D 변환을 통해서 자신의 점유 BW를 Δh_i 만큼 늘리는 대신 자신의 점유 DW를 Δd_i 만큼 줄일 수 있다. 여기서 $\Delta d_i = E_i^{B2D} \Delta h_i$ 및 $E_i^{B2D} = 1/E_i^{D2B}$ 이다.

증명: (2.5)로부터 흐름 i의 점유 DW를 Δd_i만큼 증가시키기 위해 변화되어야 하는 레이턴시 지수 값 $\Delta\beta_i$ 는 다음과 같다.

$$\Delta\beta_i = -(C^B/M_i)d_i. \qquad (4.1)$$

q_i가 b_i와 동일한 값을 갖도록 해야 하므로 (2.4)로부터 Δh_i 는 다음과 같이 전개된다.

$$\Delta h_i = -\frac{M_i}{b_i - C^B}\Delta\beta_i = -\frac{C^B}{(b_i - C^B)}\Delta d_i = -\frac{h_i C^B}{\beta_i M_i}\Delta d_i \qquad (4.2)$$

(4.2)로부터 보조정리가 증명된다.

모든 흐름이 모두 D2B 변환과 B2D 변환의 대상이 되지 않는다. 임의 흐름이 이들 변환의 대상이 되는, 즉 D2B 변환가능과 B2D 변환가능이 되기 위한 조건을 살펴보자.

보조정리 2: $P_i(r,b,h,\beta,M)$를 갖는 임의 흐름 i에 대해 $\beta_i \in (0,1]$이고 $h_i > r_i$이면 D2B 변환가능이 되고, $\beta_i \in (0,1)$이면 B2D 변환가능이 된다.

증명: 흐름 i에 대해 (2.5)로부터 $\beta_i \in (0,1]$이면 β_i를 줄여서 d_i를 늘릴 수 있다. 하지만 $h_i=r_i$이면 요구속도 보장 때문에 h_i를 감소시킬 수 없다. 따라서 유효한 E_i^{D2B} 값을 가지려면 $\beta_i \in (0,1]$인 레이턴시 지수와 $h_i>r_i$인 스케줄링 속도를 가져야 하고, 그러면 보조정리 1로부터 D2B 변환의 대상이 될 수 있다. 한편 (2.4)로부터 $\beta_i>0$이면 β_i를 증가시켜 h_i를 늘릴 수 있고, (2.5)로부터 $\beta_i \in (0,1)$이면 β_i를 증가시켜 d_i를 줄일 수 있다. 따라서 $\beta_i \in (0,1)$이면 유효한 E_i^{B2D} 값을 갖게 되므로 보조정리 1로부터 B2D 변환의 대상이 될 수 있다.

3.2절에서 살펴본 바와 같이 D2BT() 알고리즘에서 요청 받은 BW 량인 ΔB를 확보하기 위해 임의 D2B 변환가능 흐름 i에 대한 D2B 변환을 수행할 때 그 흐름에 대해 한번의 D2B 변환만 수행한다. 이때 줄이는 레이턴시 지수의 값 $\Delta\beta_i^B$를 살펴보자.

보조정리 3: ΔB의 BW를 확보하기 위해 D2B 변환가능 흐름 i에 대한 D2B 변환을 수행할 때 줄일 수 있는 레이턴시 지수 값 $\Delta\beta_i$는 다음과 같이 주어진다.

$$\Delta\beta_i \le \min\left(\beta_i, \frac{h_i-r_i}{h_i}\beta_i, \frac{\Delta B}{E_i^{D2B}}\frac{C^B}{M_i}, \frac{(D^B-D)C^B}{M_i}\right) \equiv \Delta\beta_i^B. \quad (4.3)$$

증명: D2B 변환이 종료되기 전까지 흐름 i는 D2B 변환가능 해야 하므로 보조정리 2로부터 $\Delta\beta_i \le \beta_i$ 및 $\Delta h_i \le (h_i-r_i)$가 된다. (2.4)로부터 부등식 $\Delta h_i \le (h_i-r_i)$는 $\Delta\beta_i \le (h_i-r_i)\beta_i/h_i$로 전개된다. 그리고 획득하려는 BW의 량이 ΔB이므로 보조정리 1로부터 D2B 변환에 투입되는 DW량 Δd_i는 $\Delta B/E_i^{D2B}$보다 클 수 없다. 따라서 (2.5)로부터 $\Delta\beta_i \le (\Delta B/E_i^{D2B})C^B/M_i$가 도출된다. 또한 스케줄러의 점유 DW는 D^B를 초과할 수 없으므로 $\Delta\beta_i \le (D^B-D)C^B/M_i$가 된다. 상기 $\Delta\beta_i$ 관련 4개의 부등식으로부터 보조정리가 증명된다.

마찬가지로 3.2절의 B2DT() 알고리즘에서 요청 받은 DW 량인 ΔD을 확보하기 위해 임의 B2D 변환가능 흐름 i에 대한 B2D 변환을 수행할 때 그 흐름에 대해 한번의 B2D 변환만 수행한다. 이와 같은 B2D 변환에서 사용될 BW 증가 량 Δh_i^{BI}를 살펴보자.

보조정리 4: ΔD의 DW를 확보하기 위해 D2B 변환가능 흐름 i에 대한 D2B 변환을 수행할 때 증가시킬 수 있는 BW 량 Δh_i 다음과 같이 주어진다.

$$\Delta h_i \le \min\left(\frac{h_i}{\beta_i}-h_i, \frac{\Delta D}{E_i^{B2D}}, C^B-C\right) \equiv \Delta h_i^{BI}, \quad (4.4)$$

여기서 C는 스케줄러의 점유 BW이다.

증명: B2D 변환이 종료되기 전까지 흐름 i는 B2D 변환가능 해야 하므로 보조정리 2로부터 $\Delta\beta_i \le (1-\beta_i)$가 된다. 그러면 (2.4)로부터 $\Delta h_i \le (h_i/\beta_i-r_i)$로 주어진다. 획득하려는 DW 량이 ΔD이므로 보조정리 1로부터 $\Delta h_i \le \Delta D/E_i^{B2D}$가 되고 스케줄러의 점유 BW는 C^B를 초과할 수 없으므로 $\Delta h_i \le (C^B-C)$가 된다. 상기 Δh_i 관련 3개의 부등식으로부터 보조정리가 증명된다.

보조정리 4에서 구한 Δh_i^{BI}는 $C=C^B$인 경우 항상 0의 값을 가져 B2D 변환이 되지 않는다. 따라서 점유 자원 최적화를 위한 B2D 변환시 $C=C^B$인 상태에서 B2D 변환이 가능하도록 하기 위해 $C \le C^B$의 조건을 무시하는데, 이 경우 B2D 변환시 증가시킬 수 있는 BW 량인 Δh_i^{B2}는 (4.4)의 Δh_i^{BI}에서 세 번째 항인 (C^B-C)부분이 제외된 것이 된다.

다음은 서로 다른 흐름 사이에 BW와 DW의 상호 변환, 즉 흐름 간 (intra-flow) 자원변환에 대해 살펴보자. 임의 두 흐름 i와 j를 생각하자. 그러면 다음과 같은 두 가지 유형의 연속(pipelined)변환이 가능하다: 하나는 흐름 i에 대한 B2D 변환을 통해 Δd_i의 DW를 먼저 확보한 다음 이를 흐름 j에 대한 D2B 변환에 투입하는 것이다. 그러면 흐름 i의 점유 BW가 늘어나는 반면 흐름 j의 점유 BW가 줄어든다. 물론 스케줄러의 점유 DW는 변화가 없다. 이러한 연속변환을 B2B 변환이라 하고, 그 효율을 E_{ij}^{B2B}로 표기한다. 그러면 $E_{ij}^{B2B} = E_i^{B2D}E_j^{D2B}$가 된다. 다른 하나는 반대로 흐름 i에 대한 D2B 변환을 통해 Δh_i의 BW를 먼저 확보한 다음 이를 흐름 j에 대한 B2D 변환에 투입하는 것이다. 그러면 흐름 i의 점유 DW가 늘어나는 반면 흐름 j의 점유 DW가 줄어든다. 물론 스케줄러의 점유 BW는 변화가 없다. 이러한 연속변환을 D2D 변환이라 하고, 그 효율을 E_{ij}^{D2D}로 표기한다. 그러면 $E_{ij}^{D2D} = E_i^{D2B}E_j^{B2D}$가 된다.

보조정리 5: $E_{ij}^{B2B}>1$인 임의 B2D 변환가능 흐름 i와 D2B 변환가능 흐름 j가 존재하는 한 LOFQ 스케줄러는 B2B 변환을 통해 자신의 점유 BW를 줄일 수 있다.

증명: 보조정리 1로부터 흐름 i에 대한 B2D 변환은 $\Delta d_i=-E_i^{B2D}\Delta h_i$로 계산되는 Δd_i의 DW를 산출한다. 이 DW를 흐름 j에 대한 D2B 변환에 투입하면 흐름 j의 점유 BW는 $\Delta h_j=-E_j^{D2B}(-\Delta d_i)=-E_{ij}^{B2B}\Delta h_i$

로 계산되는 Δh_i만큼 줄어든다. $E_{ij}^{B2B}>1$이면 이러한 B2B변환은 스케줄러의 점유 BW를 $(|\Delta h_j|-|\Delta h_i|)>0$ 만큼 줄어들게 하므로 보조정리가 증명된다.

보조정리 6: $E_{ji}^{D2D}>1$인 임의 D2B 변환가능 흐름 j와 B2D 변환가능 흐름 i가 존재하는 한 LOFQ 스케줄러는 자신의 점유 DW를 줄일 수 있다.

증명: 보조정리 1로부터 흐름 j에 대한 D2B 변환은 $\Delta h_i=-E_i^{D2B}\Delta d_i$로 계산되는 Δh_i의 BW를 산출한다. 이 BW를 흐름 i에 대한 B2D 변환에 투입하면 흐름 i의 점유 DW는 $\Delta d_i=-E_i^{B2D}(-\Delta h_i)=-E_{ji}^{D2D}\Delta d_j$로 계산되는 Δd_i만큼 줄어든다. $E_{ji}^{D2D}>1$이면 D2D변환은 스케줄러의 점유 DW를 $(|\Delta d_i|-|\Delta d_j|)>0$ 만큼 줄어들게 하므로 보조정리가 증명된다.

그런데 $E_{ij}^{B2B}=E_i^{B2D}E_j^{D2B}=E_j^{D2B}E_i^{B2D}=E_{ji}^{D2D}$이다. 따라서 $E_{ij}^{B2B}>1$인 임의 B2D 변환가능 흐름 i와 D2B 변환가능 흐름 j가 없으면 $E_{ji}^{D2D}>1$인 임의 D2B 변환가능 흐름 j와 B2D 변환가능 흐름 i도 없고, 그 반대도 성립한다. 이로부터 다음의 추론이 도출된다.

추론 1: LOFQ 스케줄러는 B2B변환 또는 D2D 변환을 통해 자원 점유 량을 최소화, 즉 점유 자원 최적화를 달성할 수 있다.

참고로 2장에서 B2B 변환에 의해 자원 최적화 알고리즘을 구현한 것은 바로 추론 1에 근거를 두고 있다. 이제 이 알고리즘에서 사용되는 Δd_i^B를 구해보자.

보조정리 7: 임의 B2D 변환가능 흐름 i와 D2B 변환가능 흐름 j에 대한 B2B 변환에 있어서 흐름 i에 대한 B2D 변환을 통해 얻어질 수 있는 DW 값 Δd_i는 다음과 같다.

$$\Delta d_i \le \min\left(\frac{(1-\beta_i)M_i}{C^B}, \frac{\beta_j M_j}{C^B}, \frac{(h_j-r_j)\beta_j M_j}{h_j C^B}\right) \equiv \Delta d_i^B. \quad (4.5)$$

증명: B2B 변환이 종료되기 전까지 흐름 i는 B2D 변환가능 해야 하고 흐름 j는 B2D 변환가능 해야 한다. 따라서 보조정리 2에 의해 전자로부터 $\Delta\beta_i\le(1-\beta_i)$의 부등식, 후자로부터 $\Delta\beta_j\le\beta_j$ 및 $\Delta h_j\le(h_j-r_j)$의 부등식이 각각 도출된다. (2.4)에 의해 $\Delta h_j\le(h_j-r_j)$로부터 $\Delta\beta_j\le(h_j-r_j)\beta_j/h_j$의 부등식이 도출되므로 $\Delta\beta_j\le\min(\beta_j,(h_j-r_j)\beta_j/h_j)$가 된다. (2.5)로부터 $\Delta d_i=\Delta\beta_i M_i/C^B\le(1-\beta_i)M_i/C^B$ 및 $d_j=\Delta\beta_j M_j/C^B\le\min(\beta_j M_j/C^B,(h_j-r_j)\beta_j M_j/(h_j C^B))$가 도출된다. 그리고 스케줄러의 점유 DW에는 변화가 없어야 하므로 $\Delta d_i=\Delta d_j$가 된다. 따라서 보조정리가 증명된다.

한편 흐름은 두 가지 품질 변수인 요구속도 r과 지연규격 b의 상호관계 측면에서 세 가지 유형으로 나눌 수 있다. (2.2)에 의해 b는 r^C에 반비례하므로 r과 b 대신 r과 r^C의 관계로 표현하면, $r>r^C$, 즉 속도가 지연보다 더 엄격한 속도제약(RR: Rate-Restricted) 흐름, $r<r^C$, 즉 지연이 속도보다 더 엄격한 지연제약(DR: Delay-Restricted) 흐름, 그리고 $r=r^C$인 최적(OP: OPtimal) 흐름이 바로 그것들이다.

보조정리 8: 임의 DR 흐름은 D2B 변환가능이다.

증명: F(r,b,M)를 갖는 임의 DR 흐름을 생각하자. $s=r^C$이므로 (2.2)와 (2.3)으로부터 q=b. 그림 1의 수락제어 알고리즘의 수행 결과로 h=s가 되므로 (2.3)과 (2.4)로부터 $(M/s+M^{max}/C^B)=(D^B+\beta M/s)$가 되고, 이로부터 $\beta=(1-(D^B-M^{max}C^B)s/M$. $D^B>M^{max}/C^B$이므로 $\beta<1$이 되며, 따라서 보조정리 2로부터 DR 흐름은 D2B 변환가능이 된다.

4.2 이용도 비교

먼저 RR, DR 및 OP의 각 흐름 유형별로 두 방식의 대역폭 이용도를 비교 분석 해보자. WFQ와 LOFQ 스케줄러에서 최대 수용 흐름의 수를 각각 N^{WFQ}와 N^{LOFQ}로 표기하자. 그리고 그림 1에서 D2BT() 와 B2DT() 알고리즘이 없는 원시 LOFQ를 생각하자. 그러면 원시 LOFQ에서 $h\le s$가 된다. 원시 LOFQ 스케줄러에서의 최대 수용 흐름의 수를 N^{LOFQN}으로 표기하면 $N^{LOFQ}\ge N^{LOFQN}$이 된다.

보조정리 9: LOFQ 스케줄러는 RR 흐름에 대해 WFQ 스케줄러 보다 열악하지 않은 대역폭 이용도를 보장하고 잉여 지연자원을 제공한다.

증명: RR흐름의 정의로부터 $r>r^C$ 이므로 (2.2)로부터 $s=r$가 되고, $h\le s$이므로 $h\le r$가 된다. 따라서 다음 식이 성립한다.

$$N^{LOFQ}\ge N^{LOFQN}=\max\{\text{정수 } k|\ \textstyle\sum_{i=1}^{k} h_i\le C^B\}$$
$$\ge N^{WFQ}=\max\{\text{정수 } k|\ \textstyle\sum_{i=1}^{k} r_i\le C^B\}. \quad (4.6)$$

$r>r^C$ 이므로 (2.3)으로부터 임의 RR 흐름 i에 대해 $(M^{max}/C^B+M_i/r_i)<b_i$가 된다. 이 흐름에 대한 최적 레이턴시 바이어스 D_i^B를 다음과 같이 정의하자: $D_i^B\equiv(b_i-M_i/r_i)>M^{max}/C^B$. 그러면 (2.4)로부터 $(D^B+\beta_i M_i/r_i)=(D_i^B+M_i/r_i)$가 되므로 β_i는 다음과 같이 표현된다.

$$\beta_i=1-(D^B-D_i^B)r_i/M_i. \quad (4.7)$$

(2.6)으로부터 스케줄러의 점유 DW D는 다음과 같이 전개된다.

$$D = \frac{\sum_{i=1}^{N^{LOFQ}}((D^B - D_i^B)r_i/M_i)M_i}{C^B} = D^B - \frac{\sum_{i=1}^{N^{LOFQ}} D_i^B r_i}{C^B} < D^B - \frac{M^{max}}{C^B}. \quad (4.8)$$

$N^{LOFQ} \geq N^{WFQ}$이고 $D \leq D^B$이므로 LOFQ 스케줄러는 최소한 WFQ 스케줄러와 동일한 대역폭 이용도를 보장하고, M^{max}/C^B보다 큰 지연자원을 제공한다.

보조정리 10: LOFQ 스케줄러는 DR 흐름에 대해 WFQ 스케줄러 보다 열악하지 않은 대역폭 이용도를 보장한다.

증명: DR 흐름의 정의에 의해 $r<r^C$ 이므로 (2.2)로부터 $s=r^C$가 되고, $h \leq s$이므로 $h \leq r^C$가 된다. 따라서 다음 식이 성립한다.

$$N^{LOFQ} \geq N^{LOFQN} = \max\{\text{정수 } k | \sum_{i=1}^{k} h_i \leq C^B\}$$
$$\geq N^{WFQ} = \max\{\text{정수 } k | \sum_{i=1}^{k} r_i^C \leq C^B\}. \quad (4.9)$$

$s=r^C$ 이므로 (2.4)와 (2.3)으로부터 $(D^B+\beta_i M_i/r_i^C)=(M^{max}/C^B+M_i/r_i^C)$가 된다. 따라서 β_i는 다음과 같이 표현된다.

$$\beta_i = 1-(D^B - D_i^B)r_i^C/M_i. \quad (4.10)$$

(2.6)으로부터 $D=(D^B-M^{max}/C^B)<D^B$가 된다. $N^{LOFQ} \geq N^{WFQ}$이고 $D \leq D^B$이므로 LOFQ 스케줄러는 최소한 WFQ 스케줄러와 동일한 대역폭 이용도를 보장한다.

보조정리 11: LOFQ 스케줄러는 OP 흐름에 대해 WFQ 스케줄러 보다 열악하지 대역폭 이용도를 보장한다.

증명: OP 흐름의 정의에 의해 $r=r^C$ 이므로 (2.2)로부터 $s=r$가 되고, $h \leq s$이므로 $h \leq r$가 된다. 따라서 다음 식이 성립한다.

$$N^{LOFQ} \geq N^{LOFQN} = \max\{\text{정수 } k | \sum_{i=1}^{k} h_i \leq C^B\}$$
$$\geq N^{WFQ} = \max\{\text{정수 } k | \sum_{i=1}^{k} r_i \leq C^B\}. \quad (4.11)$$

$s=r$이므로 (2.4)와 (2.3)으로부터 $(D^B+\beta_i M_i/r_i)=(M^{max}/C^B+M_i/r_i)$가 된다. 따라서 β_i는 다음과 같이 표현된다.

$$\beta_i = 1-(D^B - D_i^B)r_i/M_i. \quad (4.12)$$

(2.6)으로부터 $D=(D^B-M^{max}/C^B)<D^B$가 된다. $N^{LOFQ} \geq N^{WFQ}$이고 $D \leq D^B$이므로 LOFQ 스케줄러는 최소한 WFQ 스케줄러와 동일한 대역폭 이용도를 보장한다.

참고로 DR 및 OP 흐름의 경우도 잉여 지연자원을 제공하나 그 양이 무시할 수 있을 정도로 미미하다.

정리 1: LOFQ 스케줄러는 WFQ 스케줄러와 같거나 우수한 대역폭 이용도를 보장한다.

증명: RR, DR 및 OP 흐름의 수를 각각 I, J 및 K라 하자. 그러면 보조정리 9, 10 및 11로부터 $N^{LOFQ} \geq N^{LOFQN} \geq N^{WFQ}$가 된다. 여기서 N^{WFQ}는 $(\sum_{i=1}^{I} r_i + \sum_{j=1}^{J} r_j^C + \sum_{k=1}^{K} r_i) \leq C^B$를 만족하는 최대 정수 값 I, J 및 K의 합이다. (4.7). (4.10) 및 (4.12)를 (2.6)에 대입하면 스케줄러의 점유 DW인 D는 다음과 같이 전개된다.

$$D = \frac{\sum_{i=1}^{I}(0,1-\beta_i)M_i + \sum_{j=1}^{J}(0,1-\beta_j)M_j + \sum_{k=1}^{K}(0,1-\beta_k)M_k}{C^B}$$
$$= \frac{\sum_{i=1}^{I}(D^B-(b_i-M_i/r_i))r_i + \sum_{j=1}^{J}(D^B-M^{max}/C^B)r_j^C + \sum_{k=1}^{K}(D^B-M^{max}/C^B)r_k}{C^B}$$
$$\leq D^B - \frac{\sum_{i=1}^{I}(M^{max}/C^B)r_i + \sum_{j=1}^{J}(M^{max}/C^B)r_j + \sum_{k=1}^{K}(M^{max}/C^B)r_k}{C^B}$$
$$= D^B - \frac{M^{max}}{C^B}. \quad (4.13)$$

$N^{LOFQ} \geq N^{WFQ}$이고 $D<D^B$이므로 LOFQ 스케줄러는 최소한 WFQ 스케줄러와 동일한 대역폭 이용도를 보장한다. 보조정리 8로부터 DR 흐름은 D2B 변환가능이고 스케줄러에 M^{max}/C^B 보다 더 많은 가용 DW가 있으므로 DR 흐름에 대한 D2B 변환을 통해 스케줄러의 점유 BW를 줄여서 더 많은 흐름을 수용할 수 있다. 따라서 정리가 증명된다.

5. 성능평가

성능평가에 사용할 흐름의 규격 F(r,b,M)에서 요구속도 r과 지연규격 b의 분포를 고찰해보자. 인터넷에서 유통되는 미디어는 8Kbps급의 저속 음성부터 1Mbps급의 고속 영상까지 다양하므로 [8,1000]Kbps 내 균등하게 분포하는 균등분포와 8Kbps급 음성 흐름, 128Kbps급 오디오 흐름 및 1Mbps급 비디오 흐름이 각각 70%, 20% 및 10%의 비율로 구성되는 VoIP(Voice over Internet Protocol) 분포의 두 가지 요구속도 분포를 고려한다. 널리 사용되는 QoS 서비스인 VoIP에서 이용자의 만족을 위해서 단방향 종단간 지연시간이 300ms 이내가 되어야 하는데[10], 본 연구에서는 각 라우터에 할당되는 지연규격이 [1,11] ms에 분포한다고 가정한다. 그러면 대부분의 경우

종단간 라우터의 수가 20개를 넘지 않을 것이므로 종단간 큐잉 지연은 약 [20,220]ms가 되고, 종단간 전송지연은 약 [80, 280]ms가 된다. 그런데 스케줄러에는 다수의 흐름이 수용되므로 널리 알려진 중심극한정리(Central Limit Theorem)에 의해 지연규격은 N(6ms, σ^2), 여기서 N()은 정규(Normal)분포, 6ms는 평균 지연규격 그리고 σ^2 는 지연규격의 분산을 의미함, 를 따르게 된다. 참고로 분산이 줄어들수록 지연규격의 분포가 평균값에 집중되는 델타분포로 근접하고, 분산이 증가할수록 전 구간에 골고루 분포하는 균등분포로 근접한다. 시뮬레이션 방법을 이용하여 WFQ 및 LOFQ1과의 비교 측면에서 LOFQ2의 성능을 평가하는데, 시뮬레이션은 SMPL(Simulation Model Programming Language)[11]에 공평 패킷 스케줄러 루틴을 추가하여 수행한다.

평가결과는 그림 3과 4에 도시되어 있다. 그림 3은 패킷 크기 증가에 따른 WFQ, LOFQ1 및 LOFQ2의 대역폭 이용도를 보여준다. LOFQ1는 [12]에서 조사된 일반적인 패킷 크기 범위인 1~3Kbits에서 WFQ에 비해 우수한 이용도 특성을 보여주나 그 정도는 패킷크기에 영향을 받는다. 이에 반해 LOFQ2는 새로 추가된 점유자원 최적화의 결과로 패킷 크기에 상관 없이 항상 최적의 개선효과를 얻을 수 있는데, WFQ에 비해 약 35% 정도, LOFQ1 대비 20 ~ 30% 정도의 이용도 개선이 관찰된다. 그리고 균등분포에 비해 VoIP 분포가 이용도 특성이 낮은데, 이는 저속이면서 엄격한 지연규격을 요구하는 흐름이 상대적으로 많기 때문이다. 그림 4는 정규분포를 따르는 지연규격에 대해서 분산을 증가시키면서 대역폭 이용도를 살펴본 것이다. 이 결과 분산이 증가할 수록 WFQ와 LOFQ1의 경우 이용도 특성이 감소한 후 수렴하는데, LOFQ의 경우 점유자원 최적화의 결과로 큰 변화 없이 일정한 이용도 특성을 보여준다. 참고로 [9]에서 LOFQ1의 대역폭 이용도 특성이 다소 과다하게 나타났는데, 이는 일부 매개변수의 잘못에 기인한 잘못된 결과이므로 본 논문을 통해서 바로 잡는다.

평가결과는 그림 3과 4에 도시되어 있다. 그림 3은 패킷 크기 증가에 따른 WFQ, LOFQ1 및 LOFQ2의 대역폭 이용도를 보여준다. LOFQ1는 [12]에서 조사된 일반적인 패킷 크기 범위인 1~3Kbits에서 WFQ에 비해 우수한 이용도 특성을 보여주나 그 정도는

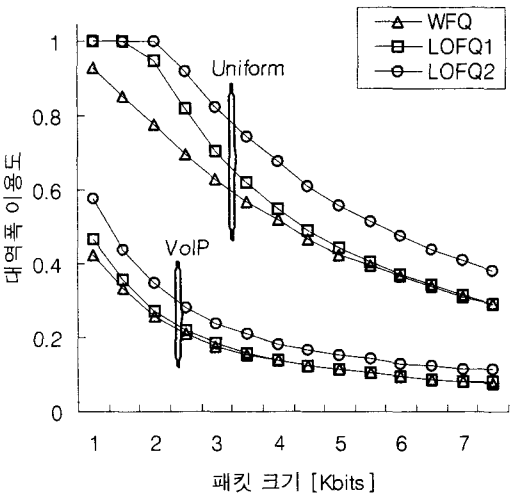


그림 3. 패킷크기에 따른 대역폭 이용도

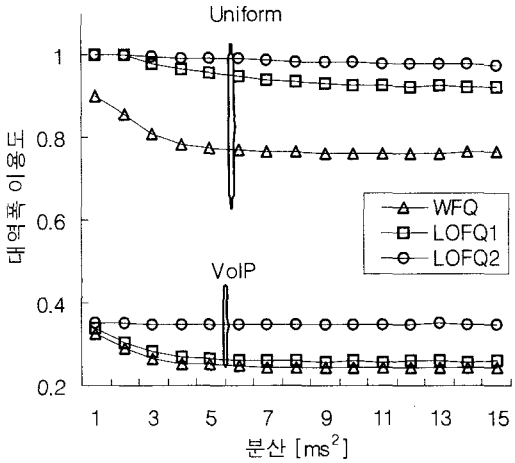


그림 4. 지연규격의 분산에 따른 대역폭 이용도

패킷크기에 영향을 받는다. 이에 반해 LOFQ2는 패킷 크기에 상관 없이 항상 최적의 개선효과를 얻을 수 있는데, 구체적으로 살펴보면 WFQ에 비해 약 35% 정도, LOFQ1 대비 20 ~ 30% 정도의 이용도 개선이 관찰된다. 이는 새로운 흐름을 수용하기 위해 요구되는 가용 BW와 DW가 없을 경우 LOFQ1은 더 이상 흐름을 수락할 수 없지만 LOFQ2의 경우 점유자원 최적화를 통해 가용 BW와 DW를 확보하여 더 많은 흐름을 수용할 수 있기 때문이다. 그리고 균등분포에 비해 VoIP 분포가 이용도 특성이 낮은데, 이는 저속이면서 엄격한 지연규격을 요구하는 흐름이 상대적으로 많기 때문이다. 그림 4는

정규분포를 따르는 지연규격에 대해서 분산을 증가시키면서 대역폭 이용도를 살펴본 것이다. 분산이 증가할수록 WFQ와 LOFQ1의 이용도 특성은 감소한 후 수렴하는데, 이는 델타분포에서 균등분포로 변해감에 따라 엄격한 지연규격을 요구하는 흐름이 상대적으로 많아지기 때문이다. 반면 LOFQ의 경우 점유자원 최적화를 통해 항상 최적의 대역폭 이용도를 제공하기 때문에 일정한 값을 보여준다. 참고로 [9]에서 LOFQ1의 대역폭 이용도 특성이 다소 과다하게 나타났는데, 이는 일부 매개변수의 잘못에 기인한 잘못된 결과이므로 본 논문을 통해서 바로 잡는다.

6. 결 론

널리 이용되고 있는 WFQ의 대역폭 이용도 저하 문제를 해결하기 위해 도입된 LOFQ를 개선하였다. LOFQ의 초기버전인 LOFQ1은 새로운 흐름의 수락 요청시 BW와 DW가 모두 부족할 경우 수락을 거절하는데, 본 연구에서 제안된 LOFQ2는 스케줄러의 점유자원 최적화 기능을 도입하여 더 많은 흐름을 수락하도록 하였다. 그리고 LOFQ1은 D2B 변환과 B2D 변환시 변환될 자원 량을 알 수 없어서 미소량 단위로 반복적으로 변환동작을 수행하는데 이로 인해 자원변환의 수행 복잡도가 너무 높다는 단점이 있다. 이에 반해 LOFQ2는 변환될 수 있는 최대 자원 량을 정확하게 산정하여 자원변환의 동작 횟수를 최소화 하였다. 또한 수학적 기법을 이용하여 WFQ와 LOFQ의 대역폭 이용도를 비교하였는데, LOFQ의 이용도 특성이 WFQ의 그것과 같거나 더 우수함을 증명하였다.

마지막으로 균등분포와 VoIP 분포의 요구속도 및 정규분포의 지연규격 트래픽 환경하에서 시뮬레이션을 통해 LOFQ2의 이용도 특성을 WFQ 및 LOFQ1의 그것과 비교한 결과 WFQ 대비 35% 정도, LOFQ1 대비 20 ~ 30% 의 이용도 개선이 확인되었다.

마지막으로 균등분포와 VoIP 분포의 요구속도 및 정규분포의 지연규격 트래픽 환경하에서 시뮬레이션을 통해 개선된 LOFQ 방식의 이용도 특성을 WFQ 및 LOFQ1과 비교하였다. 비교 결과 WFQ 대비 35% 정도, LOFQ1대비 20 ~ 30% 의 이용도 개선이 확인되었다.

참 고 문 헌

[1] X. Xiao and L. M. Ni, "Internet QoS: A Big Picture," *IEEE Network*, Vol. 13, No. 2, pp. 8-18, 1999.

[2] A.K. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," *PhD dissertation*, Massachusetts Institute of Technology, 1992.

[3] A. Demers, S. Keshav, and S. Shenker, "Design and analysis of a fair queuing algorithm," *Proc. ACM SIGCOMM'89*, pp. 1-12, 1989.

[4] S. Shenker, C. Partridge, and R. Guerin, *Specification of Guaranteed Quality of Service*, RFC 2212, IETF, 1997.

[5] M. Baldi and F. Risso, "Efficiency of Packet Voice with Deterministic Delay," *IEEE Comm. Mag*, pp. 170-177, 2000.

[6] A. Francini and F.M.Chiussi, "A Weighted Fair Queuing Scheduler with Decoupled Bandwidth and Delay Guarantees for the Support of Voice Traffic," *Proc. GLOBECOM*, Vol. 3, 2001.

[7] Do-Sung Jun, Jinwoo Choe, and Alberto Leon-Garcia, "Credit-based Processor Sharing for Decoupled Delay and Bandwidth Allocation," *IEEE Comm. Letters*, Vol. 5, No. 4, pp. 178-180, 2001.

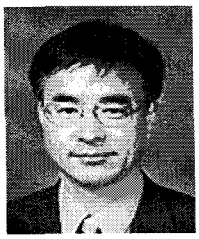
[8] Jens Schmitt. "Optimal Network Service Curves under Bandwidth-Delay Decoupling," *IEE Electronics Letters*, Vol. 38, No. 6, pp. 297-299, 2002.

[9] 김태준, 김황래, 공평 패킷 스케줄러의 대역폭 이용 효율 개선에 관한 연구, 정보처리학회논문지 제 13권-C 제3호, pp. 331-338, 2006.

[10] ITU-T Recommendation G.114: One-way transmission time, 2003.

[11] M. H. MacDougall, *Simulating Computer Systems, Techniques and Tools*, MIT Press, 1987.

[12] Hani ElGebaly, "Characterization of Multimedia Streams of an H.323 Terminal," *Intel Technology Journal*, Vol. 2 Issue 2, 1998.



김 태 준

1980년 2월 경북대학교 전자공학과 졸업
1982년 2월 한국과학기술원 전자공학 석사
1999년 8월 한국과학기술원 전자공학 박사
1982년 3월 한국전자통신연구원
1996년 3월 천안공업대학
2005년 3월~현재 공주대학교 정보통신공학부 부교수
관심분야 : 고속통신망, VoIP, 트래픽제어