Journal of Korea Multimedia Society Vol. 9, No. 8, August 2006(pp. 1020-1029)

공정 패킷 스케줄러에서 미니빈 기반 구현 복잡도 개선

김태준†, 김황래††

요 약

대용량 QoS(Quality-of-Service) 라우터의 구현을 위해서는 보다 낮은 복잡도의 공정패킷 스케줄러가 요구된다. 타임스탬프기반 공정패킷 스케줄러의 이상적 복잡도는 $O(\log V)$, 여기서 V는 최대 수락 흐름 수, 이지만 최근 빈(bin)개념을 도입하여 $O(1)$으로 대폭 줄였다. 하지만 지연 특성이 악화되고 대역폭 이용도 특성이 저하될뿐만 아니라 엄격한 지연규격의 트래픽 흐름을 수용할 수 없는 문제가 발생할 수 있다. 본 논문에서는 이러한 문제를 해결하기 위해 가변 복잡도를 갖는 MBST(Mini-Bin based Start-Time) 스케줄러를 제안하고 성능특성을 분석하였다. MBST 스케줄러는 대역폭 이용도를 높이기 위해 시작시각 기반 스케줄러의 타임스탬프 계산 방식을 적용하고, 지연 특성을 개선하기 위해 미니빈(mini-bin) 개념을 도입한다. 성능 평가 결과 대역폭 이용도의 저하 없이 시작시각 기반 스케줄러의 복잡도를 1.8 ~ 5 배 낮출 수 있었다.

Mini-Bin Based Implementation Complexity Improvement in Fair Packet Schedulers

Tae-Joon Kim†, Hwang-Rae Kim††

ABSTRACT

Realization of high-capacity quality-of-service router needs fair packet schedulers with a lower complexity. Timestamp based fair packet schedulers have the ideal complexity of $O(\log V)$, where V is the maximum number of admitted flows, but it has been recently reduced to $O(1)$ using bin concept. However, the latency property was deteriorated and the bandwidth utilization was also declined. In addition, traffic flows requiring strong delay bound may not be admitted. To overcome these problems, this paper proposes a Mini-Bin based Start-Time (MBST) scheduler with variable complexity and evaluates its performance. The MBST scheduler uses the timestamp calculation scheme of start-time based schedulers to enhance the bandwidth utilization and also introduces mini-bin concept to improve the latency. The performance evaluation shows that the proposed scheduler can reduce the complexity of the legacy start-time based schedulers by 1.8~5 times without deteriorating the bandwidth utilization property.

Key words: Fair Packet Scheduler(공정 패킷 스케줄러), Complexity(복잡도), Bandwidth Utilization(대역폭 이용도), Latency(레이턴시)

1. 서 론

인터넷 전화, 인터넷 영상회의와 같은 고 수준의 서비스 품질을 요구하는 실시간 멀티미디어 통신 서비스를 수용하기 위해 IETF(Internet Engineering Task Force)에서 품질을 보장할 수 있는 자원예약 기반의 종합 서비스(IntServ)와 확장성이 우수한 차등 서비스(DiffServ) 모델을 제시하였다[1]. IntServ

※ 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 1월 10일, 완료일 : 2006년 4월 25일

† 정회원, 공주대학교 정보통신공학부 부교수
†† 정회원, 공주대학교 컴퓨터공학부 교수
(E-mail : plusone@kongju.ac.kr)

모델에서 품질 보장은 트래픽 흐름의 요구 속도를 보장하고 요구 지연규격을 준수하는 공정 패킷 스케줄러(이하 QoS 스케줄러)에 의해 지원된다. 여기서 트래픽 흐름(이하 흐름)이란 실시간 멀티미디어 통신 서비스에 수반되는 각각의 미디어 스트림(stream), 예로 음성, 영상 스트림 등을 의미하고, 지연규격은 스케줄링 서버, 즉 라우터에서 그 흐름에게 허용되는 최대 통과(transit)시간을 의미한다. 통상 QoS 스케줄러를 탑재한 라우터를 QoS(Quality-of-Service) 라우터라 부른다.

이상적 패킷 스케줄링 알고리즘인 GPS(General Processor Sharing)[2]를 구현하기 위해 다양한 타임스탬프 기반 스케줄러가 연구되었다[3-10]. 이들은 또한 타임스탬프 계산시 사용되는 패킷의 기준시각 관점에서 예상되는 전송 종료시각을 사용하는 종료시각 방식과 예상되는 전송 시작시각을 사용하는 시작시각 방식으로 나눌 수 있다. 본 논문에서는 편의상 전자를 FT(Finish-Time) 방식, 후자를 ST (Start-Time) 방식이라 부른다. FT 방식은 WFQ (Weighted Fair Queuing)[3]에 처음 적용되었는데, WFQ는 우수한 레이턴시(latency) 특성을 제공하나 복잡도가 너무 높은 문제가 있다. SCFQ (Self-Clocked Fair Queuing)[4]에서 자가클럭(self-clock)을 도입하여 이 문제를 해결하였으나 레이턴시 특성이 저하되는 또 다른 문제점이 대두된다. 두 가지 문제점을 동시에 해결할 수 있는 방안이 서버 가상시간 기반의 RPS(Rate Proportional Servers) 이론으로 정립되어[5] 다양한 패킷 스케줄러의 개발에 적용되었다[6-8]. 한편 ST 방식은 SFQ(Starting- time Fair Queuing)[9] 및 QLR(Queuing latency Rate)[10]에 적용되었는데, SFQ는 SCFQ와 마찬가지로 자가클럭을 사용하나 QLR은 자가클럭 대신 RPS 이론에서 제안된 서버 가상시간을 사용한다. 복잡도를 낮추고, 레이턴시 특성을 개선한 그동안의 연구 결과로 이제 서버 가상시간을 사용하는 타임스탬프 기반 QoS 스케줄러는 FT 방식과 ST 방식 모두 이상적 레이턴시 특성과 $O(\log V)$의 이상적 복잡도를 갖는다[11]. 여기서 V는 최대 수용 흐름의 수 이다.

한편 QoS 라우터의 고속화, 대형화를 위해서는 보다 낮은 복잡도의 QoS 스케줄러가 요구된다. 최근에 BSFQ(Bin Sort Fair Quing)[12]에서 빈(bin) 개념을 도입하여 FT 방식의 복잡도를 $O(\log V)$에서 $O(1)$으로 대폭 낮추었다. 하지만 빈에 의해 레이턴시 특성이 크게 악화되어 엄격한 지연규격의 서비스를 수용할 수 없을 뿐만 아니라 대역폭 이용도가 크게 저하되는 문제가 발생한다. 본 논문에서는 이러한 문제를 해결하기 위해 $O(1)$에서 $O(\log V)$ 사이의 가변 복잡도를 갖고, 대역폭 이용도가 우수한 MBST(Mini-Bin based Start-Time) 스케줄러를 제안하고 성능 특성을 평가하고자 한다.

본 논문의 구성은 다음과 같다: 2장에서 관련 연구의 소개와 더불어 기존 방식의 문제점을 살펴보고, 3장에서 제안된 방식을 기술하며, 4장에서 제안된 방식의 성능 특성을 분석한다. 그리고 5장에서 제안된 방식의 성능 특성을 평가하며 6장에서 결론을 맺는다.

2. 관련연구

QoS 스케줄러의 성능은 주로 레이턴시와 복잡도 지표에 의해 평가된다[2-12]. 복잡도는 스케줄링 알고리즘을 동작시키는 컴퓨팅 능력 측면의 제약 정도를 평가하는 것으로 타임스탬프 계산 복잡도와 패킷 전송 복잡도에 의해 결정된다[11]. 전자는 스케줄러마다 다르나 후자는 최대 V개 흐름 큐의 선두 패킷중 타임스탬프가 가장 앞선 패킷을 찾는 작업에 기인하므로 $\log V$ 번의 연산 동작을 의미하는 $O(\log V)$의 값을 갖는다[11].

먼저 FT 방식 관련 연구결과를 살펴보자. WFQ에서 임의 흐름 i의 레이턴시 D_i^{FT} 는 다음과 같이 계산된다[3].

$$D_i^{FT} = \frac{L_i}{r_i} + \frac{L_{\max}}{C}, \tag{2.1}$$

여기서 r_i는 흐름 i의 속도, L_i는 흐름 i의 최대 패킷 크기, $L_{\max}$는 모든 흐름의 최대 패킷크기, 그리고 C는 스케줄러 대역폭이다. 참고로 스케줄러 대역폭이란 출력 링크의 전송 속도를 의미하며, 본 연구에서 사용되는 두 가지 용어인 속도와 대역폭은 같은 의미를 갖는다. WFQ의 레이턴시는 대응 GPS 서버에서의 패킷 서비스 시간을 의미하는 L_i / r_i항과 출력 링크에서 최대 패킷 전송시간을 의미하는 $L_{\max} / C$항으로 구성되는 이상적인 값을 갖는다. 하지만 패킷이 도착할 때 마다 모든 흐름, 즉 최대 V개의 흐름 상태를 파악해야 하므로 WFQ의 타임스탬프 계산 복잡도는 $O(V)$가 된다. SCFQ 방식에서 자가클럭 (전송 중인

패킷의 타임스탬프)을 도입하여 타임스탬프 계산 복잡도를 $O(1)$로 대폭 줄였으나 $(L_i / r_i + VL_{\max} / C)$로 계산되는 레이턴시[4]는 (2.1)에 비해 $(V-1)L_{\max} / C$ 만큼 크게 저하된다.

RPS 이론[5]에서 서버 가상시간을 도입하여 WFQ의 레이턴시 특성을 유지하면서 복잡도를 줄이는 방법을 제시하였다. 서버 가상시간이란 서버 활성구간 동안, 즉 서버가 지속적으로 패킷을 전송하는 구간의 시작시점부터 현재까지 전송된 전체 트래픽량을 나타내는 시간의 함수로서 스케줄러의 기준시간으로 사용된다. 이 이론은 대부분의 FT 방식 기반 스케줄러 (이하 FT 스케줄러), 예로 SPFQ(Starting Potential Fair Queueing)[6], MSPFQ(Medium Starting Potential Fair Queueing)[7] 및 NSPFQ (New Starting Potential Fair Queuing)[8] 등에 적용되었다. 이들은 모두 서버 가상시간의 정확성을 높이고 운영상의 복잡성을 줄이는데 주안점을 두고 개발되었다. 그 결과 이제 FT 스케줄러는 $O(\log V)$의 이상적 복잡도[11]와 WFQ급 레이턴시 특성을 갖게 되었다.

한편 ST 방식을 사용하는 SFQ(Start-time Fair Queuing)와 QLR(Queuing Latency Rate)에서 흐름의 레이턴시 D^{ST}는 다음과 같이 계산된다[9,10].

$$D^{ST} = \sum_{i=1}^{V} \frac{L_i}{C} \tag{2.2}$$

타임스탬프 계산 복잡성은 자가 클럭과 서버 가상시간과 같은 스케줄러 기준시간의 운영 복잡성에 기인하므로[11] 타임스탬프 계산시 사용되는 패킷 기준시각의 선택과 무관하다. 따라서 자가 클럭을 사용하는 SFQ는 SCFQ와 동일한 복잡도를 갖고, 서버 가상시간을 사용하는 QLR은 RPS 기반 스케줄러와 동일한 복잡도를 갖는다.

QoS 라우터가 고속화 될수록, 수용 흐름의 수가 많을수록 복잡도에 의해 라우터의 처리능력이 제한되므로 복잡도의 중요성이 강조되고 있다. 빈(bin) 개념을 도입하여 복잡도를 $O(1)$로 대폭 줄인 BSFQ [12]에서는 출력 버퍼를 가상 시간 폭이 Δ 인 빈의 집합으로 구성하고, 도착하는 패킷을 그의 타임스탬프 값에 대응하는 빈에 저장한다. 여기서 Δ는 수용되는 흐름의 정규화된 패킷길이 (흐름속도에 의해 정규화된 패킷길이)의 최대값과 같거나 더 큰 값을 갖는다. 각 빈들에 대한 서비스는 가상시간이 앞선 빈을 먼저 서비스하는 우선순위 기반이고, 빈내 패킷들에 대한 서비스는 FCFS(First-Come First-Service)으로 수행한다. BSFQ 방식에서 흐름 i의 레이턴시 D_i^B 다음과 같이 주어진다[12].

$$D_i^B = \Delta + \frac{L_i}{r_i} + \frac{VL_{\max}}{C} . \tag{2.3}$$

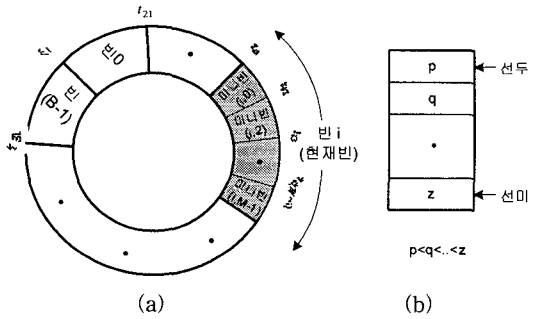
(2.3)은 FT 방식의 레이턴시, 즉 (2.1)과 비교시 $(\Delta + (V-1)L_{\max} / C)$ (이하 레이턴시 바이어스(bias)) 만큼 더 큰 값을 갖는다. 이로 인해 레이턴시 바어어스보다 엄격한 지연규격을 요구하는 서비스를 수용할 수 없는 근본적인 문제가 발생한다. 요구 지연규격이 레이턴시 바이어스를 초과하지 않을 경우 과도한 바이어스 때문에 레이턴시를 지연규격보다 작게 하기 위해 (2.3)에서 r_i값, 즉 예약속도를 더 크게 높여야 한다. 하지만 이로 인해 예약속도와 요구속도의 차이만큼의 대역폭 손실(이하 예약손실)이 발생하여 대역폭 이용도가 크게 저하될 수 있다.

3. 제안방식

본 논문에서는 낮은 복잡도와 높은 대역폭 이용도 특성을 얻기 위해 두 가지 측면에서 기존의 스케줄러를 개선한다: 첫째 BSFQ에 적용된 빈 개념을 예약손실이 없어 대역폭 이용도 특성이 우수한 ST 방식 기반 스케줄러 (이하 ST 스케줄러) 에 적용하여 대역폭 이용도를 높이고, 둘째 미니빈 개념을 도입하여 복잡도 특성을 희생하는 대신 레이턴시 특성을 개선한다. 제안되는 스케줄러는 미니빈(mini-bin)과 ST 방식에 기초하므로 이를 MBST(Mini-Bin based Start-Time) 스케줄러라 칭한다.

3.1 미니빈 구조

전송 대기 중인 패킷을 보관하는 출력버퍼는 일반적으로 그림 1 a)와 같이 링 버퍼 형태를 갖는데, 이를 B개의 빈으로 분할하고, 모든 빈을 각각 M개의 미니빈으로 분할한다. 빈은 모두 가상시간 측면에서 Δ의 동일한 길이를 갖고, 미니빈은 모두 가상시간 측면에서 δ의 동일한 길이를 갖는다. 여기서 빈 길이 Δ 수용되는 흐름의 정규화된 패킷길이의 최대값으로 한다. 참고로 빈과 미니빈의 메모리 크기는 각각 $C\Delta$비트와 $C\delta$비트가 되고, $\delta = \Delta / M$이다. 각 미니빈의 번호는 소속 빈 번호인 a와 소속빈내 미니빈 번호인 b의 쌍,



(a) (b)

그림 1. 미니빈 기반 출력 버퍼 구조 : a) 미니빈 기반 출력버퍼, b) 빈 I의 활성목록

즉 (a,b)로 표기한다. 현재 서비스 중인 빈과 미니빈을 각각 현재 빈(current bin)과 현재 미니빈(current mini-bin)이라 하고, 현재빈의 번호를 가르키는 빈 포인터를 운용한다. 빈 포인터는 서버 기동시에 0의 값으로 설정된다.

MBST 스케줄러는 서버 가상시간 $P(t)$를 운용하는데, 이는 시간 t에서 현재 미니빈의 시작 시점의 가상 시간과 같다. 모든 빈은 대기중인 패킷을 갖고 있는 미니빈, 즉 활성 미니빈(active mini-bin)의 소속빈내 미니빈 번호, 즉 미니빈 번호 (a,b)의 b값이 나열된 활성목록을 갖는다. 활성목록내 번호는 우선순위 순서, 즉 오름차순으로 나열한다. 따라서 가장 작은 번호가 활성목록의 선두(head)에 위치하고, 가장 큰 번호가 선미(tail)에 위치한다. 현재 미니빈에 대기중인 패킷을 모두 전송 완료하면 활성목록의 선두 번호가 제거되고, 그 다음에 나열된 번호를 선두에 위치하게 한다. 현재 빈의 활성목록의 선두에 위치한 번호의 미니빈이 바로 현재 미니빈이 되는데, 현재 미니빈이 바뀔 때 마다 서버 가상시간을 갱신한다. 활성목록내 나열된 번호가 없으면, 즉 모든 활성 미니빈이 모두 서비스 완료되면 다음 빈을 현재 빈으로 설정하기 위해 빈 포인터를 다음 빈을 가르키도록 갱신한 후 마찬가지의 방법으로 현재 빈내 패킷을 서비스 한다. 참고로 빈 길이의 정의로부터 활성 흐름은 현재 빈내에 최소한 하나 이상의 패킷을 갖게되므로 서버 활성구간동안 현재 빈으로 편입되는 빈은 항상 하나 이상의 패킷을 갖고 있는 빈, 즉 활성 빈이 된다.

3.2 스케줄링 알고리즘

먼저 패킷 수신 알고리즘을 살펴보자. MBST 스케줄러에 패킷이 도착하면 다음과 같이 타임스탬프를 계산한 후 해당 미니빈에 패킷을 저장하고 필요시 활성목록을 갱신한다:

i) 임의 흐름 i 의 k 번째 패킷 p_i^k가 1023시점에 도착시 이의 가상 시작시간, 즉 타임스탬프 S_i^k를 다음과 같이 계산한다.

$$S_i^k = \max(F_i^{k-1}, P(t)), \tag{3.1}$$

여기서 $F_i^{k-1} = S_i^{k-1} + l_i^{k-1}/r_i$, l_i^{k-1}는 패킷 p_i^{k-1}의 길이, 그리고 $F_i^0 = 0$이다.

ii) 계산된 타임스탬프로 미니빈의 번호 (a,b)를 다음과 같이 구하여 현재 빈으로부터 a번째 떨어진 빈의 b번째 미니빈에 p_i^k를 저장한다. 만약 $(aM+b) > BM$ 이면 출력 버퍼의 범위를 벗어나므로 에러처리 한다.

$$a = \left\lfloor \frac{S_i^k - (P(t) - h\delta)}{\Delta} \right\rfloor \quad \text{및} \quad b = \left\lfloor \frac{S_i^k - (P(t) - h\delta) - a\Delta}{\delta} \right\rfloor. \tag{3.2}$$

여기서 h는 활성목록의 선두번호이고, $(P(t) - h\delta)$는 현재 빈의 시작 시점의 가상 시간을 의미한다.

iii) 미니빈 (a,b)가 처음으로 패킷을 보관, 즉 비활성 상태에서 활성상태로 천이할 경우 빈 a의 활성목록에 미니빈 (a,b)의 b 값을 등록한다. 이때 활성목록에 등록된 번호 리스트를 오름차순으로 재배열 한다.

출력 버퍼에 대기중인 패킷의 전송에 있어 각 빈들에 대한 서비스는 빈 포인터기반 우선순위, 빈내 미니빈들에 대한 서비스는 활성목록 기반 우선순위, 그리고 미니빈내 패킷들에 대한 서비스는 FCFS (First-Come First-Service) 방식으로 수행한다. 서버 활성구간의 시작 또는 매 패킷의 전송 완료시 동작하는 패킷 전송 알고리즘을 구체적으로 살펴보면 다음과 같다:

i) 현재 미니빈내에 패킷이 대기하고 있으면 FCFS 방식으로 패킷을 전송한다.

ii) i)에서 대기중인 패킷이 없으면 활성목록의 선두 번호를 제거한 후 활성목록에 등록된 번호가 있을 경우 그 다음 번호를 선두로 위치하고 다음과 같이 서버 가상시간 $P(t)$를 갱신한 다음 i)의 과정을 반복한다.

$$P(t) = P(t) + (h^+ - h^-)\delta, \tag{3.3}$$

여기서 h^-는 최근에 제거된 활성목록의 선두 번호이

고 h^+는 새로 활성목록의 선두에 위치하는 번호이다.

iii) ii)에서 활성목록에 등록된 번호가 없지만 스케줄러가 여전히 활성상태이면 빈 포인터를 다음 빈을 가르키도록 갱신하고 서버 가상시간 $P(t)$를 다음과 같이 갱신한 후 i)의 과정을 반복한다.

$$P(t) = P(t) + (M - h^- + h^0)\delta, \quad (3.4)$$

여기서 h^0는 현재빈으로 설정되는 빈의 활성목록의 선두 번호이다.

4. 성능 특성 분석

먼저 MBST 스케줄러의 레이턴시 특성을 분석하고, 대역폭 이용도를 구한 후 BSFQ방식의 대역폭 이용도와 비교한다. 그리고 MBST 스케줄러의 복잡도를 살펴본다.

4.1 레이턴시 특성

정리 1: MBST 스케줄러에서 임의 흐름 105 의 레이턴시 D^M는 다음과 같다.

$$D^M = \delta + \frac{\sum_{j=1}^{V} L_j}{C}. \quad (4.1.1)$$

증명: 흐름 i를 제외한 V-1개의 흐름이 이미 활성화된 상태에서 흐름 i가 t_1 시점에 패킷 P_i의 도착에 의해 활성화 된다고 가정하자. 그러면 (3.1)과 (3.2)에 의해 P_i는 현재 미니빈에 저장된다. 한편 이미 활성화되어 있는 임의 흐름 j의 경우 현재 미니빈의 종료시점의 가상시간과 동일한 값의 타임스탬프를 갖는 패킷도 현재 미니빈에 보관되므로 현재 미니빈에 보관될 수 있는 트래픽의 량 W_j^{cur}은 다음과 같이 계산된다.

$$W_j^{cur} \le r_j\delta + L_j. \quad (4,1.2)$$

이로부터 P_i의 서비스가 시작되기 이전에 전송될 수 있는 현재 미니빈의 트래픽 W_{-i}^{cur}은

$$W_{-i}^{cur} \le \sum_{j=1, j\neq i}^{V} W_j^{cur} \le \delta(C - r_i) + \sum_{j=1, j\neq i}^{V} L_j. \quad (4.1.3)$$

따라서 MBST 스케줄러에서 P_i의 서비스 시작시점 τ^S는

$$\tau^S \le t_1 + \frac{W_{-i}^{cur}}{C}. \quad (4.1.4)$$

MBST 스케줄러는 패킷 P_i을 서비스할 때 C의 속도로 서비스하므로 서비스 종료시점 τ^F는

$$\tau^F = \tau^S + \frac{L_i}{C} \le t_1 + \frac{W_{-i}^{cur}}{C} + \frac{L_i}{C} \le t_1 + \delta + \frac{\sum_{j=1}^{V} L_j}{C}. \quad (4.1.5)$$

(4.1.5)로부터 (4.1.1)이 증명된다. □

MBST 스케줄러의 레이턴시, 즉 (4.1.1)은 미니빈의 길이에 큰 영향을 받는다. 미니빈의 길이가 증가할수록 FCFS에 근접하고 길이가 짧아질수록 SFQ에 근접한다. 참고로 빈 길이 Δ를 0으로 할 경우 BSFQ 스케줄러의 레이턴시((2.3))는 FT 방식의 레이턴시 ((2.1))에 미치지 못하는 반면 MBST 스케줄러의 레이턴시 ((4.1.1))은 ST 방식의 레이턴시 ((2.2))와 동일해진다.

4.2 대역폭 이용도

QoS 스케줄러의 대역폭 이용도는 할당된 대역폭 중 트래픽이 요구하는 속도뿐만 아니라 요구 지연규격을 모두 만족시키면서 실제 트래픽 전송에 사용된 대역폭의 비율을 의미한다. 복잡성을 피하기 위해 하나의 지연규격 Q만 고려하고, 도착흐름의 속도를 $[r_{\min}, r_{\max})$내 분포하는 랜덤변수 R로 정의한다.

먼저 MBST 스케줄러의 성능비교 대상인 BSFQ 스케줄러의 대역폭 이용도를 구하자. 대역폭 이용도는 할당된 대역폭 자원 중 실제 트래픽 처리에 이용된 자원의 비율로 정의한다. 요구속도가 r_i인 임의 흐름 i에 대해 그의 요구지연규격을 만족시키기 위해 필요한 최저속도를 흐름 i의 임계속도라 부르고, 흐름 i의 임계속도와 예약속도를 각각 r_i^{crt}와 r_i^{res}로 표기하자. 흐름의 요구속도가 임계속도보다 빠르면 요구속도로 예약하고, 느리면 임계속도로 예약해야 하므로 $r_i^{res} = \max(r_i, r_i^{crt})$가 된다. 따라서 요구속도가 임계속도보다 작은, 즉 $r_i < r_i^{crt}$인 경우 $(r_i^{crt} - r_i)$ 만큼의 예약손실이 발생한다.

총 예약속도가 스케줄러 대역폭 C를 초과할 수 없으므로 BSFQ 스케줄러가 수용할 수 있는 최대 흐름 수 V^B는

$$V^B = \{k \mid \sum_{i=1}^{k} r_i^{res} \le C \ \ and \ \ \sum_{i=1}^{k+1} r_i^{res} > C\}. \quad (4.2.1)$$

복잡성을 피하기 위하여 모든 흐름의 최대 패킷크기는 $L_{\max}$로 동일하다고 가정하자. 그러면 V^B는

$\lfloor C / r^{res} \rfloor$로 계산되고, (2.3)으로부터 모든 흐름은 모두 똑 같은 임계속도를 갖게 되는데, 이를 r^{crt}로 표기하면

$$r^{crt} = CL_{\max} /(CQ - VL_{\max} - C\Delta). \quad (4.2.2)$$

이제 도착하는 흐름의 예약속도는 $[r^{crt}, r_{\max})$내 분포하는 새로운 랜덤변수 $\max(R, r^{crt})$로 표현할 수 있으며, 이의 평균값 r^{res}는 평균값 정의에 의해 다음과 같이 계산된다.

$$r^{res} = E[\max(R, r^{crt})] = r^{crt} F_R(r^{crt}) + \int_{r^{crt}}^{r_{\max}} k f_R(k) dk, \quad (4.2.3)$$

여기서 f_R과 F_R은 랜덤변수 R의 pdf(probability density function)와 cdf(cumulative distribution function)이다. 정의에 의해 BSFQ 스케줄러의 대역폭 이용도 ρ_{BW_B}는 다음과 같이 계산된다.

$$\rho_{BW\_B} := \frac{1}{C}\sum_{i=1}^{V^B} r_i = \frac{r^{req}}{C}\left\lfloor \frac{C}{r^{res}} \right\rfloor. \text{ 여기서 } r^{req} = E[R]. \quad (4.2.4)$$

MBST 스케줄러에서 최대 흐름 수 V^M은 예약속도 수용조건으로부터 구해지는 V^B와 달리 예약속도 수용과 지연규격 수용의 두 가지 조건으로부터 구해진다. 예약속도 수용측면에서 흐름 수의 상한치 V_R^M은 (4.2.1)의 V^B와 마찬가지의 방법에 의해 다음과 같이 표현된다.

$$V_R^M = \{k \mid \sum_{i=1}^{k} r_i \le C \ \ and \ \ \sum_{i=1}^{k+1} r_i > C\} = \left\lfloor \frac{C}{r^{req}} \right\rfloor. \quad (4.2.5)$$

한편 지연규격 수용측면에서 흐름 수의 상한치 V_Q^M는 (4.1.1)로부터 다음과 같이 구해진다.

$$V_Q^M = \{k \mid \sum_{i=1}^{k} L_i \le C(Q-\delta) \ \ and \ \ \sum_{i=1}^{k+1} L_i > C(Q-\delta)\} = \left\lfloor \frac{C(Q-\delta)}{L_{\max}} \right\rfloor. \quad (4.2.6)$$

V^M은 V_R^M과 V_Q^M 중 작은 값이 되므로 (4.2.5)와 (4.2.6)으로부터

$$V^M = \min(V_R^M, V_Q^M) = \min\left(\left\lfloor \frac{C(Q-\delta)}{L_{\max}} \right\rfloor, \left\lfloor \frac{C}{r^{req}} \right\rfloor\right). \quad (4.2.7)$$

따라서 MBST 스케줄러의 대역폭 이용도 ρ_{BW_M}는 정의에 의해 다음과 같이 계산된다.

$$\rho_{BW\_M} := \frac{1}{C}\sum_{i=1}^{V^M} r_i = \frac{r^{req}}{C} V^M = \frac{r^{req}}{C}\min\left(\left\lfloor \frac{C(Q-\delta)}{L_{\max}} \right\rfloor, \left\lfloor \frac{C}{r^{req}} \right\rfloor\right). \quad (4.2.8)$$

이제 MBST 스케줄러의 대역폭 이용도를 BSFQ 스케줄러의 그것과 비교해보자.

정리 2: 동일한 조건, 즉 동일한 지연규격, 동일한 패킷크기 및 동일한 빈 길이($\Delta = \delta$) 하에서 MBST 스케줄러의 대역폭 이용도는 BSFQ 스케줄러의 대역폭 이용도 와 같거나 더 높다. 즉,

$$\rho_{BW\_B} \le \rho_{BW\_M}.$$

증명: BSFQ와 MBST 스케줄러에서 최대 수용 흐름의 수, 즉 V^B와 V^M을 비교해보자. V^M가 예약속도 수용조건에 의해 결정되는 경우 정의에 의해 $r_i^{res} > r_i$이므로 (4.2.1)과 (4.2.5)로부터

$$V^B = \{k \mid \sum_{i=1}^{k} r_i^{res} \le C \ \ and \ \ \sum_{i=1}^{k+1} r_i^{res} > C\} \le \{k \mid \sum_{i=1}^{k} r_i \le C \ \ and \ \ \sum_{i=1}^{k+1} r_i > C\} = V^M. \quad (4.2.9)$$

다음은 V^M가 지연규격 수용조건에 의해 결정되는 경우에 대해 살펴보자. BSFQ 스케줄러의 경우 (2.3)으로부터 $r_i^{crt} = CL_i /(CQ - VL_{\max} - C\Delta)$로 계산되고, 정의에 의해 $r_i^{res} \ge r_i^{crt}$가 되며, 또한 예약속도 수용조건에 의해 $\sum_{i=1}^{V^B} r_i^{res} \le C$가 된다. 따라서 $\sum_{i=1}^{V^B} r_i^{crt} \le \sum_{i=1}^{V^B} r_i^{res} \le C$가 성립하므로

$$\sum_{i=1}^{V^B} L_i \le (CQ - VL_{\max} - C\Delta). \quad (4.2.10)$$

한편 MBST 스케줄러의 경우 (4.1.1)로부터

$$\sum_{i=1}^{V^{SD}} L_i \le C(Q-\Delta). \quad (4.2.11)$$

(4.2.10)과 (4.2.11)로부터

$$V^B = \{k \mid \sum_{i=1}^{k} L_i \le (CQ - VL_{\max} - C\Delta) \ and \ \sum_{i=1}^{k+1} L_i > (CQ - VL_{\max} - C\Delta)\} \le \{k \mid \sum_{i=1}^{k} L_i \le C(Q-\Delta) \ and \ \sum_{i=1}^{k+1} L_i > C(Q-\Delta)\} = V^M. \quad (4.2.12)$$

(4.2.9)와 (4.2.12)로부터 항상 $V^B \le V^M$관계가 성립한다. 따라서 (4.2.4)와 (4.2.8)의 대역폭 이용도 정의에 의해 정리가 증명된다.□

4.3 복잡도

스케줄러의 복잡도는 스케줄링 알고리즘 수행상의 제약을 평가하기 위한 것으로 하나의 패킷을 처리하는데 요구되는 연산과정의 반복횟수로 표시한다. 그리고 가장 복잡한 내부장치의 복잡도에 의해 스케줄러의 전체의 복잡도가 결정된다[11]. MBST 스케줄러의 복잡도를 살펴보자. 3.2장의 스케줄링 알고리즘에서 먼저 M=1 인 경우를 생각해보자. 패킷 도착시 (3.1), (3.2) 및 $O(1)$의 재배열 동작이 한번만 일어나므로 패킷 수신 알고리즘의 복잡도는 $O(1)$이 된다. 한편 빈 길이의 정의로부터 현재빈의 서비스 완료시 스케줄러가 계속 활성상태이면 다음 빈이 항상 활성빈 상태가 되므로 한번의 동작으로 새로운 현재빈을 찾을 수 있다. 따라서 패킷 전송 알고리즘의 복잡도 역시 $O(1)$이 된다. 다음으로 M이 1보다 큰 경우를 생각해보자. 패킷 도착시 도착한 패킷이 저장되는 미니빈이 활성상태로 변할경우 그 미니빈의 번호를 활성목록에 등록하면서 활성목록을 오름차순으로 재배열한다. 이때 최대 log$M$번의 분류동작이 일어날 수 있으므로 패킷 수신 알고리즘의 복잡도는 log$M$이 된다. 패킷 전송 알고리즘의 복잡도는 오름차순으로 재배열되어 있는 활성목록에서 선두 번호의 미니빈에서 패킷을 FCFS로 찾아서 전송하므로 $O(1)$이 된다. 따라서 MBST 스케줄러의 복잡도는 $O(\log M)$이 된다.

5. 성능 평가

먼저 성능평가 환경을 생각해보자. 대표적 품질보장 서비스인 인터넷 전화의 종단간 지연은 150ms 이내로 정의되어 있다[13]. 종단간 지연은 전송선로에서의 전파지연과 각 노드에서의 전달지연 및 단말과 게이트웨이에서의 처리지연으로 구성된다. 근거리용(인트라넷 및 시내전화 등)과 원거리용(시외 및 국제전화 등)의 두 가지 인터넷 전화 유형에 대해 종단간 전파지연/처리지연/ 홉수가 각각 20ms/30ms/10개와 100ms/30ms/20개인 환경을 가정하자. 각 노드에서의 지연은 동일하다고 가정하면 각 노드에서 요구되는 지연규격은 1ms ~ 10ms가 된다. 한편 H.323 기반 인터넷 전화에서 패킷의 크기는 음성의 경우 100 ~ 150바이트, 영상의 경우 50 ~ 150바이트 정도로 분석되어 있다[14]. 본 연구에서는 500비트, 1000비트 및 1500비트의 3가지 고정된 패킷 크기 L을 고려하고, 2Mbps급 영상, 200Kbps급 오디오 및 20Kbps급 음성의 세가지 속도의 트래픽 흐름이 동일한 확률로 발생하는 것으로 가정한다. 그리고 라우터의 출력 링크의 속도는 10Gbps로 가정한다.

MBST 스케줄러의 비교 대상이 되는 BSFQ 스케줄러의 경우, 최소 빈 길이가 500/20000=25ms가 되므로 1 ~ 10ms의 요구 지연규격을 만족시킬 수 없어 사용할 수 없다. 다행히도 BSFQ 스케줄러에 미니빈 개념을 도입하면 MBST 스케줄러와 마찬가지로 복잡도를 희생하는 대신 레이턴시 특성을 개선할 수 있다. 개선된 BSFQ 스케줄러를 편의상 BSFQ+스케줄러라 부르자. 그러면 BSFQ+ 스케줄러의 레이턴시는 BSFQ 스케줄러의 레이턴시인 (2.3)에서 빈 길이 Δ를 미니빈 길이 δ로 대체한 형태가 되고, 이의 복잡도는 MBST 스케줄러의 그것과 마찬가지로 O(log(M))이 된다.

이제 MBST, BSFQ+, FT 및 ST의 4가지 스케줄러에 대해 대역폭 이용도와 복잡도 특성을 비교해보자. 먼저 L=1000비트와 Q=1ms 하에서 빈내 미니빈 수, 즉 M의 증가에 따른 성능 특성 비교 결과를 그림 2에 도시하였다. ST와 FT 스케줄러의 경우 복잡도가 약 13 정도로 높지만 우수한 대역폭 이용도 특성을 보인다. 반면 BSFQ+와 MBST 스케줄러는 대역폭 이용도가 낮으나 M의 증가로 복잡도를 희생하는 대신 대역폭 이용도를 개선할 수 있음을 볼 수 있다. M의 증가시 MBST 스케줄러는 ST 스케줄러의 성능에 근접하나 BSFQ+ 스케줄러는 아무리 M을 높여도 FT 스케줄러의 성능을 좇아가지 못한다. 또한 이 그림으로부터 ST 스케줄러가 FT 스케줄러보다 대역폭 이용도 특성이 우수함을 확인할 수 있는데, 이는 예약손실이 없기 때문이다.

다음은 대용량 라우터에서 요구되는 낮은 복잡도 범위에서 MBST 스케줄러의 성능 특성을 BSFQ+ 스케줄러의 그것과 비교 해보자. 1000비트의 고정된 패킷크기 L과 1.5ms, 5ms 및 10ms의 3가지 대표적인 지연규격 Q에 대해 성능 특성을 살펴본 결과를 그림 3, 4 및 5에 도시하였다. 이들 그림으로부터 MBST 스케줄러는 2.5 ~ 8 정도의 낮은 복잡도로 14 정도의 높은 복잡도를 갖는 ST 스케줄러의 대역폭 이용도 특성을 얻을 수 있다. 그리고 요구되는 복잡도는 지연규격이 느슨할수록 낮아짐을 볼 수 있다. 반면 BSFQ+ 스케줄러는 엄격한 지연규격하에서는

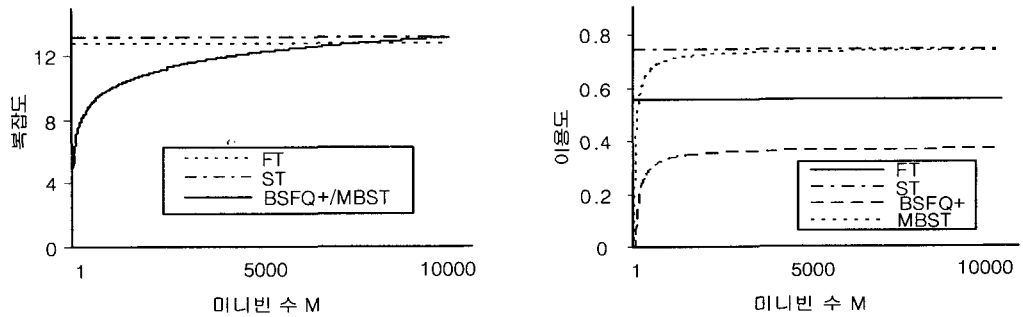


그림 2. L=1000비트 및 Q=1ms 경우 : a) 복잡도 비교, b) 대역폭 이용도 비교

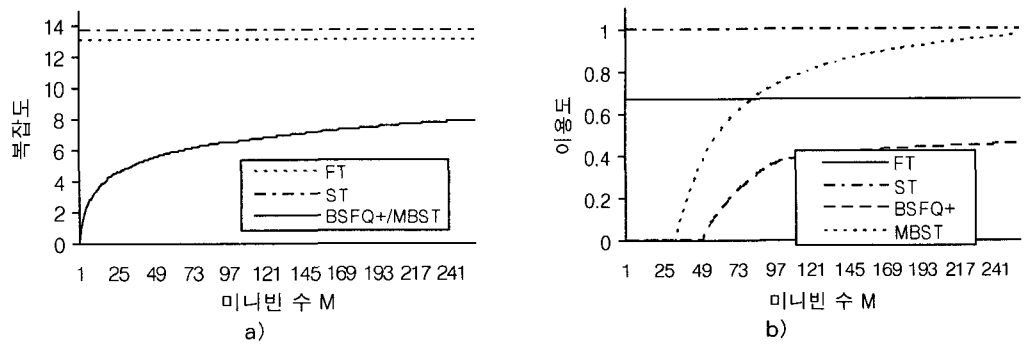


그림 3. L=1000비트 및 Q=1.5ms 경우 : a) 복잡도 비교, b) 대역폭 이용도 비교

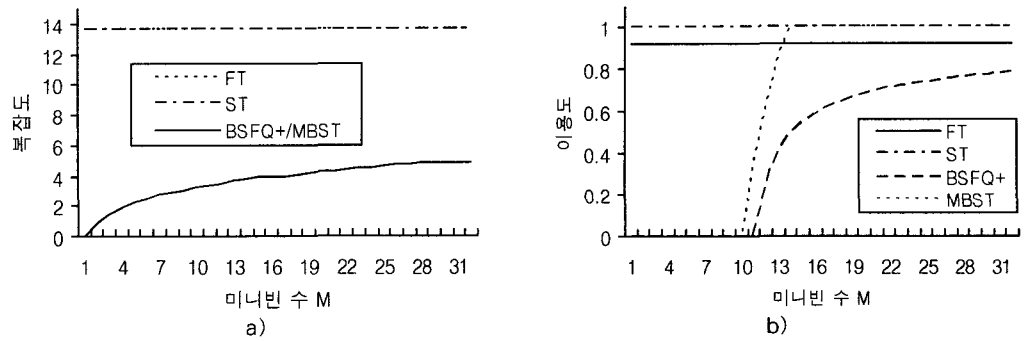


그림 4. L=1000비트 및 Q=5ms 경우 : a) 복잡도 비교, b) 대역폭 이용도 비교

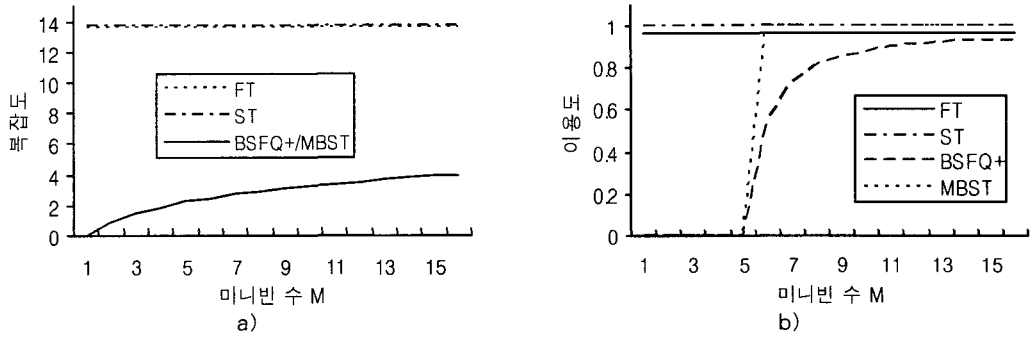


그림 5. L=1000비트 및 Q=10ms 경우 : a) 복잡도 비교, b) 대역폭 이용도 비교

아무리 복잡도를 높여도 FT 스케줄러의 대역폭 이용도 특성을 따를 수 없다. 하지만 느슨한 지연규격, 예로 Q=10ms인 경우 4 정도의 다소 높은 복잡도로 FT 스케줄러에 근접하는 대역폭 이용도를 얻을 수 있음을 볼 수 있다. MBST 스케줄러와 기존 BSFQ를 개선한 BSFQ+ 스케줄러의 비교시 MBST 스케줄러가 모든 지연규격, 모든 복잡도 범위에서 우수한 대역폭 이용도 특성을 보임을 확인할 수 있다. 한편 그림 4, 즉 Q=5ms인 경우에 대해 패킷의 크기를 가변하였을 때 대역폭 이용도 특성의 변화를 그림 6에 도시하였다. 이 그림으로부터 패킷크기를 줄일수록 대역폭 이용도가 개선됨을 확인할 수 있다. 이는 (2.1), (2.2), (2.3) 및 (4.1.1)의 레이턴시 식에서 레이턴시가 패킷크기에 비례하기 때문이다. 역시 패킷크기에 상관없이 MBST 스케줄러가 BSFQ+ 스케줄러보다 훨씬 우수한 대역폭 이용도 특성을 보임을 확인할 수 있다. 참고로 패킷 크기를 줄이면 패킷의 오버헤드의 비율이 그만큼 늘어나므로 유료부하 이용도는 나빠질 수 있다.

6. 결 론

시작시각 기반 타임스탬프 계산방식을 적용하여 대역폭 이용도를 높이고, 미니빈(mini-bin) 개념을 도입하여 복잡도를 희생하는 대신 레이턴시 특성을 개선할 수 있는 MBST 스케줄러를 제안하였다. 제안된 MBST 스케줄러의 레이턴시, 대역폭 이용도 및 복잡도 특성을 분석하였고, 이의 대역폭 이용도는 기존 BSFQ 스케줄러의 그것보다 항상 우수함을 증명하였다.

근거리 및 원거리 인터넷 전화를 대상으로 BSFQ 스케줄러에 미니빈 개념을 도입한 BSFQ+ 스케줄러와 성능 특성을 비교 평가한 결과 다음과 같은 두 가지 결과를 얻었다: 첫째 BSFQ+ 스케줄러는 M을 아무리 높여도 FT 스케줄러의 대역폭 이용도를 쫓아갈 수 없으나 MBST 스케줄러의 경우 ST 스케줄러보다 1.8 ~ 5배 낮은 복잡도로 ST 스케줄러의 대역폭 이용도 특성(동일한 복잡도하에서 FT 스케줄러보다 우수함)을 얻을 수 있다. 둘째 지연규격과 패킷크기에 상관없이 MBST 스케줄러는 항상 BSFQ+ 스케줄러보다 우수한 대역폭 이용도와 낮은 복잡도 특성을 갖는다.

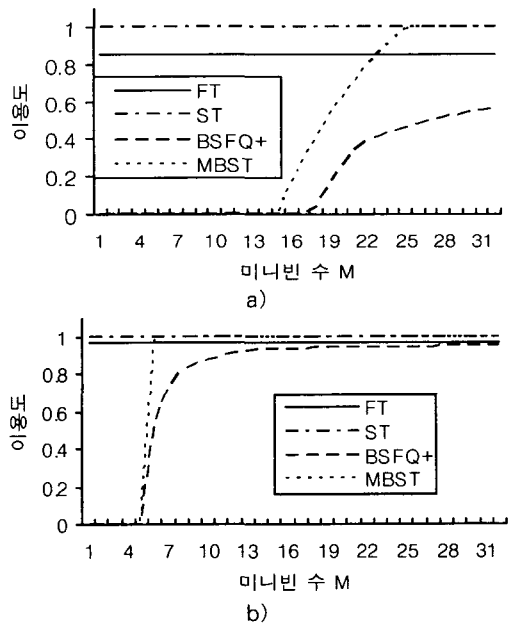


그림 6. 패킷크기 L 에 따른 성능 특성 (Q=ms) : a) L=1500비트, b) L=500비트

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*, pp. 1 - 12, 1989.

[4] S.J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications," *Proc. IEEE INFOCOM*, pp. 636-646, 1994.

[5] D. Stiliadis and A. Varma, "Rate Proportional Servers: A Design Methodology for Fair Queueing Algorithms," *IEEE/ACM Trans. Networking*, Vol. 6, No. 2, pp. 164-174, 1998.

[6] D. Stiliadis and A. Varma, "Efficient Fair Queuing Algorithms for Packet-Switched Networks," *IEEE/ACM Trans. Networking*,

Vol. 6, No. 2, pp. 175-185, 1998.

[7] Dong-Yong KWAK, Nam-Seok KO, and Hong-Shik PARK, "Medium Starting Potential Fair Queueing for High-Speed Networks," *IEICE Trans. Communications*, Vol. E87-B, No. 1, pp. 188-198, 2004.

[8] Dong-Yong Kwak, Nam-Seok Ko, Bongtae Kim, and Hong-Shik Park,"A New Starting Potential Fair Queuing Algorithm with O(1) Virtual Time Computation Complexity," *ETRI Journal*, Vol. 25, No. 6, pp. 475-488, 2003.

[9] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, Vol. 5, No. 5, pp. 690-704, 1997.

[10] H. Zhou, *Real-Time Services over High Speed Network*, Ph.D Dissertation, Curtin University of Technology, Australia, 2002.

[11] J. Xu and R. J. Lipton, "undamental Tradeoffs between Delay Bounds and Computational Complexity in Packet Scheduling Algorithms," *ACM SIGCOMM'2002*, pp. 279-292, 2002.

[12] S. Y. Cheung and C. S. Pencea, "BSFQ: Bin Sort Fair Queuing," *Proc. IEEE INFOCOM*, 2002.

[13] 이은곤, "인터넷 전화 국내외 제도화 현황 및 정책적 시사점," 정보통신정책, 제16권, 제17호, pp. 1-17, 2004.

[14] H. ElGebaly, "Characterization of Multimedia Streams of an H.323 Terminal" *Intel Technology Journal* Vol. 2 No. 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 교수
2005년 3월～현재 공주대학교 정보통신공학부 부교수
관심분야 : 고속통신망, VoIP, 트래픽제어

김 황 래

1982. 중앙대학교 공과대학 컴퓨터공학과 /공학사
1991. 중앙대학교 대학원 컴퓨터공학과/공학석사
1994. 중앙대학교 대학원 컴퓨터공학과/공학박사 수료
1983～1994 한국전자통신연구원 선임연구원
1994～현재 공주대학교 컴퓨터공학부 교수
관심분야 : 컴퓨터네트워크, 네트워크 보안, 컴퓨터 시뮬레이션