과거민감도 스펙트럼을 포괄하는 공정 스케줄링 모델

(A Fair Scheduling Model Covering the History-Sensitiveness Spectrum)

박 경 호 † 황 호 영 †† 이 창 건 ††† 민 상 렬 ††††
(Kyeongho Park) (Hoyoung Hwang) (Changgun Lee) (Sanglyul Min)

요 약 기존의 공정 스케줄링 방법들 중 GPS(generalized processor sharing)는 순간적 관점에서의 공정성을 추구하는 반면에, virtual clock은 장기적 관점에서의 공정성을 추구하는 특성을 지닌다. 이 논문에서는 이들의 차이가 과거의 서비스 정보를 추후의 스케줄링에 반영하는 정도에 있음에 주목하고, GPS와 virtual clock을 포괄하는 스펙트럼 형태의 스케줄링 모델을 제시한다. 이 모델에서 각 응용의 자원 획득 권한은 예치권한이라는 값으로 표현되는데, 예치권한은 각 응용별로 미리 정해진 고유한 비율로 계속 증가하며, 서비스를 받으면 소비된다. 소비되지 않고 누적된 예치권한은 과거에 서비스가 이루어지지 않은 정도를 표현하는 값이라고 볼 수 있으며, 이는 응용의 스케줄링 가능성을 높이므로 이후의 서비스 지연시간을 상대적으로 단축하는 효과를 낸다. 예치권한을 주기적으로 감쇄시키면 과거 정보의 반영 정도를 줄일 수 있으며, 이 때 그 감쇄 정도는 과거행태를 반영하는 정도를 의미한다. 과거의 정보를 전혀 반영하지 않을 경우 GPS의 특성을 나타내게 되며, 모두 반영할 경우 virtual clock의 특성을 보이게 된다. 이러한 스펙트럼 상에서는 평균지연시간과 장기적 공정성 사이에 절충 관계가 존재한다. 이 논문에서는 제시된 모델의 특성을 분석하고 실험을 통해 검증한다.

키워드 : 공정 스케줄링, GPS, virtual clock, 공정성, 장기적 공정성, 지연시간, 스펙트럼, 과거민감도

Abstract GPS(generalized processor sharing) is a fair scheduling scheme that guarantees fair distribution of resources in an instantaneous manner, while virtual clock pursues fairness in the sense of long-term. In this paper, we notice that the degree of memorylessness is the key difference of the two schemes, and propose a unified scheduling model that covers the whole spectrum of history-sensitiveness. In this model, each application's resource right is represented in a value called deposit, which is accumulated at a predefined rate and is consumed for services. The unused deposit, representing non-usage history, gives the application more opportunity to be scheduled, hence relatively enhancing its response time. Decay of the deposit means partial erase of the history and, by adjusting the decaying rate, the degree of history-sensitiveness is controlled. In the spectrum, the memoryless end corresponds GPS and the other end with full history corresponds virtual clock. And there exists a tradeoff between average delay and long-term fairness. We examine the properties of the model by analysis and simulation.

Key words : fair scheduling, GPS, virtual clock, fairness, long-term fairness, delay, spectrum, history-sensitiveness

· 정보통신부 및 정보통신연구진흥원의 'IT신성장동력핵심기술개발사업 [2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]'과 '2007년도 한성대학교 교내연구비' 및 '서울대학교 신임교수 연구정착금'의 지원을 받아 수행되었습니다.

† 학생회원 : 서울대학교 컴퓨터공학부
kalynda@archi.snu.ac.kr
†† 종신회원 : 한성대학교 멀티미디어공학과 교수
hyhwang@hansung.ac.kr
††† 정 회 원 : 서울대학교 컴퓨터공학과 교수
cglee@cse.snu.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터공학과 교수
symin@archi.snu.ac.kr

논문접수 : 2007년 2월 23일
심사완료 : 2007년 4월 17일

1. 서 론

컴퓨터 시스템이나 네트워크용 라우터에서 스케줄러의 역할은 매우 중요하다. 스케줄러는 자원을 배분하고, 응답 시간을 최소화하며, 시스템 자원의 사용률을 극대화하는 등의 종종 서로 상충되는 요구사항들을 만족해야 한다. 특히 서로 경쟁하는 관계에 있는 응용들을 처리하는 경우 공정한 서비스의 제공은 중요한 요구사항 중의 하나이다.

공정성의 의미와 공정한 서비스의 제공을 위한 방법을 정립하기 위해서 기존에 많은 연구가 있었다. 그 중 대표적인 것이 GPS(generalized processor sharing)와 virtual clock인데, 이 두 방법은 서로 다른 관점에서의 공정성을 추구한다. GPS[1,2]는 매 순간적 시점의 대역폭의 공정한 배분을 추구하는 방법으로서, 가용한 자원을 현재 적체된(backlogged) 응용들 간에 이들의 가중치(weight)에 비례하여 나누는 방법이다. 따라서 적체되어 있지 않아 서비스를 받지 않은 응용은 추후에 이에 대한 보상을 기대할 수 없다. 반면에 virtual clock[3][1]은 각 응용이 받은 서비스의 총합이 이들의 가중치에 비례하도록 하는 방법으로, 과거에 받지 못한 서비스를 미래에 전부 보상함으로써 장기적인 관점에서 공정한 자원배분을 추구하는 특성을 지닌다.

이 두 방법은 공정성에 대한 각기 다른 철학을 지니고 만들어졌으므로 일견 서로 관련이 없는 것처럼 보인다. 그러나 이들의 동작 원리를 살펴보면, 과거의 자원 사용 정보를 유지하고 이후의 스케줄링에 반영하는 방법에 있어서 반대되는 입장을 보인다는 것을 알 수 있다. 즉, GPS의 경우 과거의 대역폭 사용량 정보에 관한 기억을 남기지 않으므로 서비스를 받지 못한 응용들에게 나중에 보상해 주지 않는 반면, virtual clock은 과거에 사용하지 않은 대역폭에 관한 정보를 모두 기억해 두었다가 미래에 모두 보상해 주는 특성을 지닌다. 이 때 이 두 방법을 양 극단으로 하고 과거의 정보를 유지하여 보상해 주는 정도에 따라서 연속성을 지니는 스케줄링의 스펙트럼의 존재를 가정할 수 있다.

이 논문에서는 과거 서비스 대역폭 사용량 정보에 대한 기억 정도를 조절하여 GPS부터 virtual clock까지 하나의 스펙트럼 상에서 표현가능한 스케줄링 모델을 제시한다. 이 모델에서는 자원을 획득할 수 있는 권한을 나타내는 예치권한(deposit)이라는 값이 각 응용에 주어지며, 스케줄러는 예치권한의 크고 작음에 기반하여 서비스할 응용을 선택한다. 이 모델에서는 예치권한이 누적, 감쇄, 소비되는 방법을 정의하고 있으며, 이러한 세부사항의 설정에 의해 전체 모델의 움직임이 결정된다. 특히 예치권한의 주기적인 감쇄 정도가 주요한 매개변수가 되며 이를 조절함으로써 과거 민감도를 결정할 수 있다. 이 모델을 통해서 서로 무관해 보이는 기존의 두 스케줄링 기법 사이에 연관성이 있음을 알 수 있으며, 이들을 양 극단으로 하는 스펙트럼 상에서 장기적 공정성과 평균지연시간 사이의 절충관계를 발견할 수 있다.

1) [3]에는 여러 가지 버전의 virtual clock이 언급되어 있는데, 여기에서는 이 논문의 문맥에 적합한 첫 번째 버전을 가리킨다.

기존에 다양한 공정 스케줄링 알고리즘들을 종합하여 연속된 속성을 지닌 통합된 모델로 설명하려는 시도들은 그리 많지 않다. Zhang[4]은 공정 큐잉에서 사용되는 스케줄러들을 그 특성에 따라 분류, 정리하였으며, Goyal의 연구[5]의 경우, 서비스율을 보장하는 스케줄러의 클래스를 정의하고 몇몇 알고리즘이 이에 포함된다는 것을 보인 바 있다. 그러나 본 논문에서 제시된 것과 같은 관점의 스펙트럼으로 포괄하려는 연구와는 거리가 있다. [6]의 경우 과거에 받은 서비스 대역폭과 지연시간이 상호 관계를 가지는 지연시간-대역폭 정규화에 관한 연구로서, 개념상 본 연구에서 제시된 스펙트럼 상의 한 지점에 대응된다.

이 이후의 논문의 구성은 다음과 같다. 2절에서는 기존의 GPS와 virtual clock 기법의 특성에 대해서 설명한다. 3절에서는 본 논문에서 제안하는 과거정보에 기반한 스펙트럼 형태의 스케줄링 모델을 정의한다. 4절에서는 이 모델에서 제시된 스펙트럼의 양 극단에서 GPS 및 virtual clock과 유사한 특성을 볼 수 있다는 점을 설명한다. 5절에서는 실험을 통해 이 모델이 가지는 특성에 대해 분석한다. 그리고 6절에서 결론을 맺고 추후 연구과제에 대해 언급한다.

2. 기존의 공정 스케줄링 방법

2.1 GPS

GPS는 매 시점에 가용한 서비스 대역폭을 적체된 응용들의 가중치에 비례하여 분배하는 공정 큐잉 방법이다. 가중치가 r_i인 응용 i가 구간(t_1, t_2)에서 받은 서비스의 양을 $S_i(t_1, t_2)$라고 하면, GPS에서는 응용 i가 (t_1, t_2) 구간에서 연속으로 적체되어 있을 때 임의의 j에 대해

$$\frac{S_i(t_1,t_2)}{S_j(t_1,t_2)} \geq \frac{r_i}{r_j}, \quad j = 1,2,\ldots, N \tag{1}$$

을 만족한다[1].

GPS는 순간적인 관점에서 가장 공정한 스케줄링 방법이라고 할 수 있으나, 서버의 용량을 무한히 나누어 여러 응용에 동시에 서비스 가능한 유체(fluid) 모델에만 적용가능하다. 따라서 현실의 스케줄러에 그대로 적용할 수는 없으며, GPS를 패킷 모델로 근사한 여러 알고리즘들이 제안되어 있다. 이의 대표적인 것들로 PGPS (packet-by-packet GPS)[1,2], SFQ(start-time fair queueing)[7], WF^2Q(worst-case fair weighted fair queueing)[8], stride 스케줄링[9], MCF(most credit first)[10] 등이 있다. 각 방법은 기본적으로는 GPS의 원리를 따르고 있으며 패킷 단위로 서비스가 제공되는 서버 모델에서의 쉬운 구현을 염두에 둔 것들로, 이들은 정확도와 구현복잡도 등에서 차이를 보인다.

PGPS는 응용 i의 k번째 패킷 p_i^k에 시작 태그(start tag) S_i^k와 종료 태그(finish tag) F_i^k를 붙이고 종료 태그의 값이 작은 것을 우선적으로 스케줄링한다. a_i^k, L_i^k를 각각 p_i^k의 도착시간, 길이라고 하고 $V()$가 시스템의 가상시간이라고 할 때, 태그의 계산 방법은 다음과 같다.

$$S_i^k = \max\{ F_i^{k-1}, V(a_i^k)\}$$
$$F_i^k = S_i^k + \frac{L_i^k}{r_i}$$
$$F_i^0 = 0 \qquad (2)$$

다른 대부분의 방법들도 각 요청의 종료 태그를 계산하고 이 순서대로 스케줄링한다는 점에서 유사하다.

PGPS에서는 시스템의 가상시간 $V(a_i^k)$를 구하기 위해 GPS를 에뮬레이션하는 부담이 있는데 반해, SFQ는 시작 태그를 스케줄링 및 가상시간 관리에 사용함으로써 복잡도를 줄였다. 그리고 WF^2Q는 가중치가 높은 응용이 서버를 단기간에 독점하는 현상을 개선한 방법이다. Stride 스케줄링은 네트워크 환경에서 주로 사용되는 GPS를 CPU 스케줄링에 적용한 알고리즘이며, MCF는 GPS의 철학에 기반하고 있으나 알고리즘의 복잡도를 낮춘 방법이다.

2.2 virtual clock

이 논문에서 모델로 하는 virtual clock은 다음과 같은 동작을 수행한다[3].

1. 각 응용은 가중치에 따라 $Vtick_i = 1/r_i$ 를 가진다.
2. 패킷이 도착할 때마다 VC_i += $Vtick_i$ 를 수행한다.
   단, 응용의 최초의 패킷에 대해서는
   VC_i = *real_time* 으로 설정한다.
3. 적체된 응용들의 VC_i 값의 순서대로 스케줄링한다.

이 방법의 경우 현재 시점까지 서비스를 받은 정보가 VC_i에 저장되어 있다고 볼 수 있으며, 이를 토대로 각 응용의 누적된 서비스 양이 가중치에 비례하도록 서비스된다. VC_i는 서비스를 받을 때만 증가하므로 오랜 기간 동안 서비스를 받지 못한 응용은 VC_i가 상대적으로 작은 값을 유지하며, 이런 응용이 서비스 요청을 한꺼번에 몰아서 낼 경우 시스템을 독점할 수 있다.

3. 과거민감도에 기반한 스케줄링 모델

이 절에서는 이 논문에서 제시하는 기본적인 모델에 대해서 설명한다. 이 모델은 과거의 서비스 정보가 추후의 스케줄링에 얼마만큼의 영향을 미치는지를 가장 중요한 매개변수로 삼고 있으며, 과거의 서비스 정보를 스케줄링 권한에 반영하고 이의 감쇄 정도를 제어하는 등

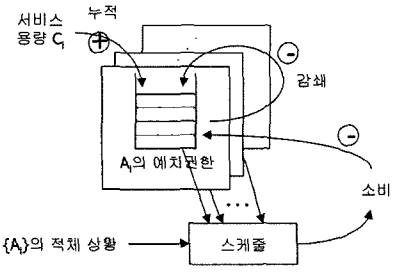

그림 1 기본적인 모델

의 동작을 수행하게 된다. 기본적인 모델을 그림 1에 보였으며, 세부 사항은 아래에서 설명한다.

이 모델에서 가정하는 서버의 특성은 다음과 같다. 자원을 서비스하는 서버의 용량은 편의상 1로 가정하며, 서비스는 시간 슬롯 단위로 스케줄링된다. 그리고 서버는 작업보전형(work-conserving)으로 동작하는데, 이는 수행할 작업이 있는 한 서버는 중단 없이 계속 서비스를 수행함을 의미한다.

3.1 예치권한의 역할

스케줄링은 주어진 철학에 비추어 현재 어느 응용을 서비스하는 것이 가장 타당한가를 결정하는 작업이다. 기본적으로 이 논문의 모델은 과거에 자신에게 주어진 권한에 비해 적게 서비스를 받은 응용을 우대하는 정책에 기반하고 있으므로, 과거에 응용에게 주어졌으나 사용되지 않은 대역폭이 어느 정도인지의 정보를 예치권한(deposit)이라는 하나의 값으로 나타낸다. 따라서 예치권한은 각 응용이 현재의 시점에 자원을 획득할 수 있는 상대적인 권한을 나타내며, 스케줄러는 예치권한이 가장 큰 응용을 선택한다. 예치권한은 아래에 설명하는 바와 같은 규칙들에 의해 누적, 소비, 감쇄되며, 이 변화 규칙을 다르게 설정함으로써 다양한 특성의 스케줄러를 표현할 수 있다.

3.2 예치권한의 누적

응용들은 자신의 가중치에 맞추어 단위시간당 일정한 자원을 서비스받기 위해 사용할 수 있는 권리를 부여받는다. 응용 i는 미리 정해진 고유한 서비스 용량 C_i를 주기적으로 받게 되는데, C_i는 2절에서 언급한 공정 스케줄링 알고리즘들에서의 가중치에 해당하는 값이다.

응용이 항상 적체되어 있고 서비스가 유체 모델 형태로 진행된다면, 각 응용은 자신에게 주어지는 이 권리를 즉시 완전히 소비할 것이다. 그러나 일반적으로는 실제 서비스 진행 과정에서 응용은 적체 상태와 휴지 상태를 반복하게 되며, 설사 여러 응용이 동시에 적체되어 있더라도 한 시점에는 한 응용만이 패킷 단위로 서비스될 수 있다. 따라서 발생 즉시 사용되지 않은 권리는 예치권한에 누적되며, 이렇게 예치권한이 증가한 응용은 다

음번 스케줄링에서 선택될 가능성이 상대적으로 높아지게 된다.

권리가 사용되지 않고 계속 누적되기만 한다면, 시간 t에 응용 i의 예치권한 $D_i(t)$는

$$D_i(t) = D_i(t-1) + C_i$$
$$D_i(0) = 0 \quad (3)$$

으로 표현된다. 주어진 권리가 사용되지 않았다면 그 이유는 응용이 휴지 상태여서 서비스 요구를 내지 않았기 때문이거나, 또는 적체 상태였지만 다른 응용과의 경쟁에 졌기 때문이다. 이 때 예치권한은 응용이 사용할 수 있도록 과거에 부여받은 권리를 사용하지 않고 보관해 놓은 것이라고 해석할 수 있다. 예치권한이 상대적으로 크다는 것은 자원을 획득하기 위한 경쟁에서 유리하기 때문에 지연시간을 줄일 수 있다는 의미이며, 동시에 권리의 미사용분에 대해 그만큼의 보상을 받을 수 있다는 의미가 된다. 결국 이러한 모델에서는 자원의 사용량과 지연시간이 상호 상관관계를 가지게 된다.

3.3 예치권한의 소비

응용이 자원을 사용한 경우 서비스의 비용에 해당하는 만큼을 예치권한에서 차감하는데, 세부적인 것은 개별적인 스케줄러의 특성에서 언급한다.

3.4 예치권한의 감쇄

본 모델에서는 예치권한에 과거의 자원을 사용하지 않은 정도에 대한 정보가 누적되어 있으며, 이를 통해 추후의 스케줄링시에 보상해 주는 효과를 낼 수 있음을 앞에서 언급하였다. 예치권한을 주기적으로 감쇄시키면, 이런 과거의 정보를 부분적으로 망각하는 효과를 기대할 수 있다. 그리고 이 감쇄 정도에 변화를 주면 시간의 경과에 따른 과거 정보의 반영 정도를 조절할 수 있으며, 결과적으로는 서비스 지연시간도 이에 영향 받게 된다. 극단적으로 이 예치권한의 값을 바로 없애서 사라지게 한다면 응용은 과거에 받지 못한 서비스에 대한 보상을 전혀 받을 수 없고 지연시간 측면에서 이득을 누릴 수 없지만, 다른 개별 응용들은 지연시간을 보장받을 수 있을 것이다. 반대로 예치권한의 값이 사라지지 않고 모두 유지된다면 장기간 서비스를 받지 않아 예치권한이 많이 누적된 응용은 사용하지 않은 권리를 나중에 모두 보상받음으로써 지연시간의 이득을 누릴 수 있으나, 이로 인해 다른 응용들은 지연시간을 보장받지 못하는 상황이 발생할 수 있다.

단위시간당 예치권한의 감쇄 정도를 λ라는 상수로 표현한다면 예치권한의 감쇄 모델은 다음의 식으로 표현 가능하다.

$$D_i(t) = \lambda D_i(t-1), \quad 0 \le \lambda \le 1 \quad (4)$$

이 때 λ는 과거의 기억을 유지하는 정도를 결정한다. 식 (3)을 감안할 경우 다음과 같이 정리할 수 있다.

$$D_i(t) = \lambda D_i(t-1) + C_i, \quad 0 \le \lambda \le 1 \quad (5)$$

위 식에서 표현된 예치권한에는 과거에 사용되지 않은 권리의 정보가 누적되어 있으며, 시간 t에서 멀리 떨어진 과거의 정보일수록 여러 번의 감쇄를 거치기 때문에 전체 예치권한 값에 기여하는 정도가 작음을 알 수 있다. 예를 들어, 시점 t-5, t-3, t-2에 예치권한이 사용되지 않고 누적되었다면 식 (5)에 의한 시점 t에서의 예치권한은 다음과 같을 것이다.

$$D_i(t) = (\lambda^5 + \lambda^3 + \lambda^2 + 1)C_i$$

이 값은 예치권한의 소비에 의한 감소분을 감안하지 않고 감쇄만을 반영한 것이며, 특정시점에서의 정확한 예치권한은 소비 모델까지 감안하여 결정된다.

그림 2에 λ의 변화에 따라 예치권한의 증가 형태가 달라짐을 보였다(단, 계속 서비스는 받지 않았다고 가정했을 때).

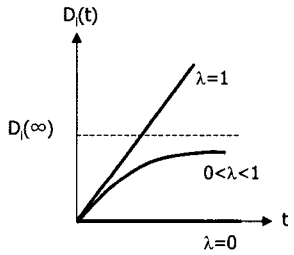

그림 2 λ에 따른 예치권한의 변화

λ=0일 때 예치권한은 누적되지 못하므로 증가할 수 없다. 반면에 λ=1일 때는 예치권한은 무한히 증가하게 된다. 0<λ<1일 때는 예치권한이 증가함에 따라 감쇄되는 양도 많아져서 특정한 값에 수렴하게 된다. 이 때 예치권한의 수렴하는 값은 다음과 같이 계산된다.

$$D_i(\infty) = \lambda D_i(\infty) + C_i$$
$$D_i(\infty) = \frac{C_i}{1-\lambda} \quad (6)$$

개념상 λ=0인 경우는 GPS에 해당하는 특성을 가지고, λ=1인 경우는 virtual clock에 해당하는 특성을 가진다고 볼 수 있다. 그리고, 0<λ<1인 경우는 그 중간적인 특성을 지니게 된다.

식 (6)에 따르면 λ의 변화에 따라 각 응용의 최대 예치권한 값이 결정되며, 이는 서비스 지연시간에 영향을 준다. 일반적으로 한 응용의 예치권한이 다른 응용의 예치권한과 차이가 클수록, 예치권한이 작은 응용의 지연시간 또한 증가할 것이다. 따라서, λ가 0일 경우 지연시간을 보장받으나, λ가 증가할수록 평균지연시간은 증가한다. 한편 과거의 정보를 오래 유지할수록 장기적 관

점에서의 응용들 간의 서비스 양에 공정성을 추구할 수 있으므로, 그림 3에 보인 바와 같이 평균지연시간과 장기적 공정성 간의 절충관계가 성립함을 알 수 있다.

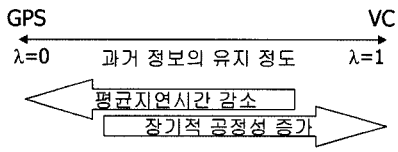

그림 3 장기적 공정성과 평균지연시간의 절충관계

4. 모델의 특성 분석

이 절에서는 λ 값의 변화에 따른 이 모델의 특성을 고찰해 보고, 기존의 스케줄링 기법인 GPS, virtual clock과의 유사성을 검토한다.

4.1 λ=0인 경우: GPS

GPS는 가용한 용량을 적체되어 있는 응용들 사이에 가중치의 비율대로 나누어 가지는 모델이므로 적체되어 있지 않고 휴지 상태인 응용들이 받지 못한 서비스는 보상되지 않는다. 따라서 본 모델에서 개념적으로 λ=0인 경우에 해당한다. 그리고 서비스를 받은 각 응용은 자신이 현 시점에 받은 예치권한을 전량 소비하게 된다.

그러나 현실적으로 GPS를 구현하는 알고리즘들의 경우 유체 모델인 아닌 패킷 단위의 서비스를 가정하여 변형된 형태를 제시하고 있으므로, 본 모델도 다음과 같은 변경을 통해 패킷 서버에의 적용이 가능하다.

(1) 패킷 단위의 서비스를 수행할 때, 응용이 요청을 내지 않아서 받지 못한 서비스에 대한 보상은 해 주지 않지만, 적체되어 있었음에도 불구하고 다른 응용과의 경쟁에서 밀려서 사용하지 못하고 누적된 예치권한은 스케줄링으로 인해 발생하는 오차라고 볼 수 있으므로 이에 대해서는 보상할 필요가 있다. 이를 위해서는 응용이 적체구간에 받는 예치권한은 그대로 누적시키면 된다. 즉 적체구간에 한해서 λ=1을 적용한다. 이를 감안할 때, 식 (5)를 다음과 같이 변형하여 적용하면 된다.

$$D_i(t) = \lambda D_i(t-1) + C_i$$
$$\lambda = \begin{cases} 0: \text{응용 } i \text{의 휴지구간} \\ 1: \text{응용 } i \text{의 적체구간} \end{cases} \quad (7)$$

(2) GPS에서는 같은 양의 서비스는 동일한 가치를 가진다고 볼 수 있으므로, 원칙적으로 서비스를 받은 응용의 예치권한 소비분은 항상 일정한 값을 적용해야 할 것이다. 그러나, 휴지상태에 있는 응용에 의해 사용되지 못하고 바로 소멸되는 예치권한으로 인해 적체된 응용들의 예치권한이 음의 방향으로 발산하는 현상이 발생할 수 있다. 예를 들어, C_1=C_2=0.5인 두 응용 A_1, A_2가 있을 때 A_1만이 적체되어 있다면 A_1은 독점적으로 서비스를 받을 것이다. 만일 서비스로 인한 소비분을 1로 고정하면 A_1은 단위시간당 0.5의 예치권한을 누적하지만 1을 소비하게 되므로, 결과적으로 A_1의 예치권한은 매 단위시간당 0.5씩 무한히 감소하게 된다. 이 결과, A_2가 나중에 서비스 요청을 할 경우 A_1은 무한 대기 상태에 처할 소지가 있는데 이는 GPS의 동작원리에 맞지 않는다.

이 문제를 해결하기 위해 예치권한의 전체증감분이 균형을 맞추도록 흐름 균형화가 이루지게 한다. 즉 현재 적체 상태에 있는 응용에게 주어지는 예치권한만이 실제로 유효하게 사용될 가능성이 있는 것으로 간주하여, 이들의 합을 서비스 받은 응용에게 부과되는 소비분으로 삼는다. 이를 식으로 표현하면

$$\text{흐름균형화를 감안한 예치권한의 소비분} = \sum_{j \in Q} C_j \quad (8)$$

이다. 이 때 Q는 적체되어 있는 응용들의 집합이다.

(3) 서비스를 받은 직후 응용의 예치권한은 감소하지만 계속 적체 상태를 유지할 경우 서서히 양의 방향으로 회복된다. 반면 응용이 휴지 상태로 바뀌면 식 (7)에 정의하였듯이 λ=0이 적용되므로 예치권한도 0으로 재설정된다. 그런데 만일 서비스를 받은 직후에 응용의 예치권한이 음수였고 이 응용이 짧은 시간 동안의 휴지상태를 거쳐 적체상태로 바뀐다면, 계속 적체되어 있었던 경우에 비해 오히려 더 빨리 예치권한을 회복하는 현상이 발생할 수 있다. 즉 짧은 휴지기간을 자주 거치는 이런 응용은 지속적으로 적체되어 있는 다른 응용에 비해 과도한 서비스를 누리는 불공정성이 발생할 수 있다. 이를 해결하기 위해서는 휴지 상태에서 무조건 λ=0을 적용하지 않고 예치권한이 음수인 구간 동안에 한해 λ=1을 적용한다. 따라서 식 (7)을 다음과 같이 개선한다.

$$D_i(t) = \lambda D_i(t-1) + C_i$$
$$\lambda = \begin{cases} 1: \text{응용 i의 휴지구간 또는 예치권한<0인 구간} \\ 0: \text{나머지 구간} \end{cases} \quad (9)$$

위와 같은 설정을 적용할 때 각 응용이 움직일 수 있는 예치권한의 범위는 한정되어 있고, 어떤 응용이 적체되어 서비스를 기다리고 있는 한 그 응용의 예치권한은 식 (7)에 의해 감쇄 없이 계속 증가하므로 한정된 시간 내에 반드시 서비스를 받을 수 있다. 즉 각 요청의 지연시간은 한정되어 있다.

4.2 λ=1인 경우: virtual clock

virtual clock은 응용이 사용하지 않은 권리를 모두 예치권한에 누적했다가 보상하는 모델이므로 λ=1에 해

당한다. 또한 서비스는 예치권한의 크기에 무관하게 동일한 가치를 지니므로 서비스시의 소비량은 항상 고정된 값을 적용한다. virtual clock은 과거에 받지 못한 서비스에 대해서 요청이 있으면 전부 보상해 주는 방식이므로, 오랜 기간 동안 서비스를 받지 않아 예치권한이 크게 증가한 응용이 서버를 독점할 경우 다른 응용들은 기아 상태에 빠지므로 무한대의 최악 지연시간을 가질 수 있음을 직관적으로 알 수 있다.

4.3 $0<\lambda<1$인 경우

$0<\lambda<1$의 경우는 위의 양 극단의 중간에 해당하는 특성을 지닌다. 서비스를 장기간 받지 않더라도 감쇄 효과에 의해 예치권한이 무한히 증가하지는 못하므로 식 (6)에서 계산한 $D_i(\infty)$를 넘지는 못한다.

예치권한의 소비분의 책정에는 여러 가지 정책이 고려될 수 있는데, $\lambda=1$인 경우에서처럼 고정된 값을 적용하면 전체적인 흐름 균형이 이루어지지 않는 문제가 생길 수 있다. 이의 해결을 위해서 여러 가지 정책이 고려될 수 있겠으나, 본 논문에서는 일단 예치권한의 소비분을 현재 시점의 예치권한의 값에 단순히 비례해서 적용하는 방법을 채택한다. 이를 수식으로 구체화하면, 서비스에 따른 예치권한의 소비분은 $\mu D_i(t)$로 표현된다. (단, $0<\mu<1$) 즉 작은 예치권한을 가지는 응용이 서비스를 받을수록 그 서비스에 대한 가격을 낮게 책정한다. 작은 예치권한을 가지는 응용이 스케줄러에 의해 선택되었다는 것은 오랜 기간 서비스를 받지 않아 예치권한이 많이 누적된 다른 응용이 없었다는 의미이며, 이는 일반적으로 그만큼 경쟁이 덜한 상황이었음을 암시하는 것이어서 서비스의 가격을 적게 책정하는 것으로 해석할 수 있다.

5. 실 험

이 절에서는 시뮬레이션을 통해 본 모델에서 λ의 변화가 평균지연시간에 미치는 영향과 그 의미를 검토하고, 부가적으로 시스템 사용률에 미치는 영향에 대해서도 언급한다.

이 실험에서는 두 개의 응용이 동시에 수행된다고 가정하였는데, 응용 A_1은 CPU 요청만을 연속적으로 내는 반면, 응용 A_2는 CPU 요청과 I/O 요청을 무작위로 발생시킨다. 각 응용이 내는 요청은 모두 시간 슬롯 단위의 동일한 길이의 요청으로 가정하였다. 각 응용의 서비스 용량 C_1, C_2는 각각 0.5씩으로 가정하였으며, μ는 0.3으로 설정하였다. 그리고 λ를 0.1부터 0.9까지 0.1 단위로 변화시키고, A_2의 전체 요청에서 CPU 요청이 차지하는 비율을 10%에서 90%까지 10% 단위로 변화시키면서 10,000 단위시간씩 동안 실행한 결과를 관찰하였다.

그림 4는 위와 같은 설정에서 두 응용이 내는 CPU 요청들이 경쟁할 때 겪는 평균지연시간의 전체 추이를 비교해서 보인 것이다. 평균지연시간을 계산할 때 각 응용이 내는 모든 CPU 요청을 포함시키면 한 응용만이 CPU에 적체된 시점의(즉 다른 응용과의 경쟁 없이 바로 CPU를 획득할 수 있는 시점의) 요청들까지 모두 포함되므로, CPU를 적게 사용한 응용이 CPU 경쟁시에 얻는 지연시간상의 이득을 왜곡하여 표현할 우려가 있다. 그래서 동시에 적체되어 경쟁하는 요청들만의 지연시간을 평균에 반영하였다.

일단 과거정보 민감도를 나타내는 λ의 변화에 따른 평균지연시간의 추이를 살펴보면, A_2의 CPU 요청 비율에 상관없이 λ의 값이 커짐에 따라 A_1은 평균지연시간이 증가하는데 반해 A_2의 평균지연시간은 감소하는 것을 볼 수 있다. 이것은 λ의 값이 증가할수록 과거의 서비스 대역폭 사용량에 대한 기억을 하는 정도가 증가하고 이로 인해 A_2의 미사용 서비스에 대한 보상을 하는 경향이 더욱 증가하기 때문이다.

한편 A_2의 CPU 요청 비율이 작을수록 A_1에 비해 평

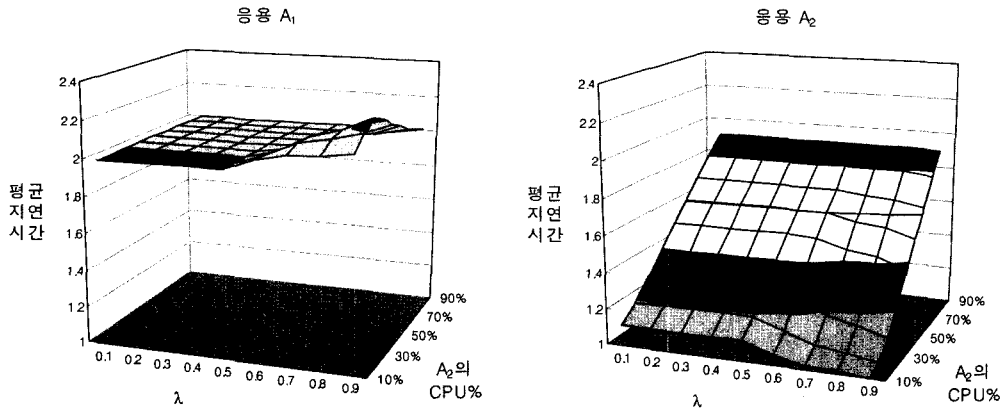

그림 4 경쟁하는 CPU 요청들의 평균지연시간

균지연시간이 더 짧아지는 것을 분명하게 볼 수 있다. 이는 A_2의 CPU 사용량이 적을수록 예치권한이 더 많이 누적되므로, CPU 요청의 스케줄링시에 우선권을 가지는 경향이 있기 때문이다. 그런데 A_2의 CPU 요청 비율이 30%인 부근에서 A_1의 평균지연시간이 약간 증가하는 특이 현상이 관찰된다. 그 이유는 A_2의 CPU 요청 비율이 10%에서 30%로 증가하는 동안, 연속으로 도착하는 A_2의 요청들과의 경쟁에서 A_1이 연속하여 지는 경우가 증가하기 때문이다. 그러나 A_2의 CPU 요청 비율이 30% 이상이 되면서 A_2의 예치권한의 증가세 둔화가 심해지기 때문에, A_2의 요청이 연속으로 도착하더라도 연속으로 경쟁에 이기는 비율은 줄어들고 이에 따라 A_1의 평균지연시간도 감소 추세로 전환된다.

그림 5는 A_2의 CPU 요청 비율이 50%일 때 λ의 변화에 따른 I/O 서버의 이용률 변화를 관찰한 것이다. λ가 증가함에 따라 I/O 서버의 이용률도 증가함을 볼 수 있는데, 이는 I/O 위주 작업의 CPU 요청에 우선순위를 줄수록 이 작업의 CPU에서의 평균지연시간이 줄어들고 결과적으로 I/O 서버쪽으로의 요청을 더 많이 낼 수 있기 때문이다. 이 실험에서는 CPU가 병목서버이므로 I/O 서버의 이용률이 증가하면 시스템 전체의 이용률도 자연히 증가한다.

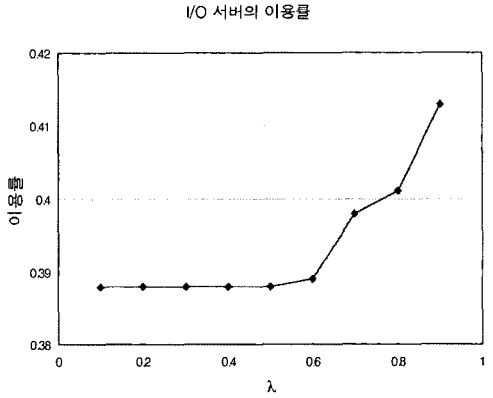

그림 5 I/O 서버의 이용률

6. 결론 및 향후 연구 방향

이 논문에서는 기존의 GPS와 virtual clock의 차이가 과거의 서비스 정보를 추후의 스케줄링에 반영하는 정도에 있음을 지적하고, 이를 양 극단으로 하는 스펙트럼 형태의 스케줄링 모델을 제시하였다. 이 모델에서는 각 응용이 자원을 획득할 수 있는 권한을 예치권한이라는 값으로 나타내며, 예치권한의 누적, 소비, 감쇄 방법을 정의한다. 사용되지 않아 누적된 예치권한을 주기적으로 감쇄시킴으로써 과거 정보의 망각이 이루어지는데, 그 감쇄 정도가 과거 정보의 반영 정도에 영향을 미치게 된다. 이 모델을 통해 서로 무관해 보이는 GPS와 virtual clock의 관계를 파악할 수 있었고, 이 스펙트럼 상에 평균지연시간과 장기적 공정성의 절충관계가 존재함을 규명하였다.

흥미로운 점은 $0<\lambda<1$인 경우의 실험 결과에서 관찰되는 특성이 UNIX 계열 운영체제에서 많이 사용되는 decay usage 알고리즘[11]과 유사한 측면을 보인다는 것이다. 개념적인 측면에서 decay usage 알고리즘과 본 모델의 방법은 과거의 서비스 정보를 주기적으로 감쇄시킨다는 점이 유사하다. 일반적으로 decay usage 알고리즘은 크게 다음과 같은 두 가지 특성을 지닌다.

(1) 최근 CPU를 적게 사용한 응용에 높은 우선순위를 줌으로써 I/O 위주 작업의 지연시간을 개선한다.

(2) I/O 위주의 응용이 CPU에서 빨리 서비스를 받으므로 I/O 쪽의 이용률이 높아지고 아울러 시스템의 전체적인 이용률을 높일 수 있다.

위 그림 4의 결과는 특성 (1)에 부합되며, 그림 5의 결과는 특성 (2)에 부합된다. 두 방법은 적용되는 매개변수의 종류가 다르고, decay usage의 구현 편의성 추구로 인해 발생하는 양자화 오류 등으로 상호간의 직접적인 변환은 어렵지만, 그 특성상의 유사성에 대해서는 고찰해 볼 가치가 있다.

참 고 문 헌

[1] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. of ACM SIGCOMM '89*, pp. 1-12, 1989.

[3] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. of ACM SIGCOMM '90*, pp. 19-29, 1990.

[4] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. of IEEE*, vol. 83, no. 10, pp. 1374-1396, October 1995.

[5] P. Goyal and H. M. Vin, "Generalized Guaranteed Rate Scheduling Algorithms: A Framework," *IEEE/ACM Trans. on Networking*, vol. 5, no. 4, pp. 561-571, August 1997.

[6] 이주현, 황호영, 민상렬, "지연-대역폭 정규화 관점에서의 출력링크 서비스 알고리즘", *한국정보과학회 제33회 추계학술발표회 논문집*, pp. 259-262, 2006.

[7] P. Goyal, H. M. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks,"

IEEE/ACM Trans. on Networking, vol. 5, no. 5, pp. 690-704, October 1997.

[8] J. C. R. Bennett and H. Zhang, "WF^2Q: Worst-case Fair Weighted Fair Queueing," *IEEE INFOCOM '96*, pp. 120-128, 1996.

[9] C. A. Waldspurger and W. E. Weihl, "Stride Scheduling: Deterministic Proportional-Share Resource Management," *Tech. Rep. MIT/LCS/TM-528*, MIT, 1995.

[10] D. Pan and Y. Yang, "Credit Based Fair Scheduling for Packet Switched Networks," *IEEE INFOCOM 2005*, pp. 843-854, 2005.

[11] J. L. Hellerstein, "Achieving Service Rate Objectives with Decay Usage Scheduling," *IEEE Trans. on Software Engineering*, vol. 19, no. 8, pp. 813-825, August 1993.

박 경 호

1991년 서울대학교 컴퓨터공학과 학사 1993년 서울대학교 컴퓨터공학과 석사 1993년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 스케줄링, 멀티미디어시스템 등

황 호 영

1993년 서울대학교 컴퓨터공학과 공학사 1995년 서울대학교 컴퓨터공학과 공학석사. 2003년 서울대학교 전기컴퓨터공학 공학박사. 2003년~2007년 안양대학교 디지털미디어학부 조교수. 2007년~현재 한성대학교 멀티미디어공학과 조교수. 관심분야는 정보통신, 무선 및 이동통신망, 멀티미디어시스템 등

이 창 건

1991년 서울대학교 컴퓨터공학과 학사 1993년 서울대학교 컴퓨터공학과 석사 1998년 서울대학교 컴퓨터공학과 박사 1998년~2000년 LG전자 선임연구원 2000년~2002년 University of Illinois, CS Dept., Postdoctoral Research Associate. 2002년~2006년 Ohio State University, ECE Dept. Assistant Professor. 2006년~현재 서울대학교 컴퓨터공학부 조교수. 관심분야는 실시간시스템, 내장형시스템, QoS 관리, 무선 ad hoc 네트워크 등

민 상 렬

1983년 서울대학교 전자계산기공학과 공학사. 1985년 서울대학교 전자계산기공학과 공학석사. 1989년 워싱턴대학교 전산학과 박사. 1989년~1990년 IBM Watson 연구소 객원 연구원. 1990년~1992년 부산대학교 컴퓨터공학과 조교수. 1992년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 컴퓨터구조, 컴퓨터성능평가, 실시간시스템, 내장형시스템 등