라운드 로빈 기반 비례지분 스케줄링을 위한 동기화 기법

(Task Synchronization Mechanism for Round Robin based Proportional Share Scheduling)

박 현 희 † 양 승 민 ††
(Hyeon Hui Park) (Seung Min Yang)

요 약 라운드 로빈 기반 비례지분 스케줄링(Round Robin based Proportional Share scheduling, RRPS)은 각 태스크에게 지분(share)을 결정하는 비중(weight)이라는 속성을 정의하고 각 태스크의 비중에 비례하여 CPU 자원을 할당한다. 라운드 로빈 기반 비례지분 스케줄링은 공평성(fairness)을 성능의 척도로 사용하며 스케줄링의 높은 공평성을 목표로 한다. 그러나 태스크들 간의 동기화로 인한 스케줄링의 공평성 문제에 대한 연구는 부족하다.

본 논문에서는 라운드 로빈 기반 비례지분 스케줄링에서 동기화로 인한 스케줄링의 지연이 높은 불공평성을 발생시킴을 보인다. 이를 비중역전(weight inversion)이라는 현상으로 설명한다. 다음, 비중역전을 방지하는 동기화 기법인 비중상속 프로토콜(weight inheritance protocol, WIP)을 제안한다. 또한, 공평성 분석과 시뮬레이션을 통해 비중상속 프로토콜이 불공평성을 감소시킴을 보인다.

키워드 : 라운드 로빈 기반 비례지분 스케줄링, 그룹 비율 라운드 로빈(GR3), 동기화 기법, 세마포어, 비중상속 프로토콜, 비중역전

Abstract Round robin based proportional share scheduling(RRPS) defines weight which determines share for each task and allocates CPU resource to each task in proportional to its respective weight. RRPS uses fairness as the measure of performance and aims at high fairness of scheduling. However, researches for scheduling fairness problem due to synchronization among tasks have been rarely investigated.

In this paper, we discuss that scheduling delay due to synchronization may result high unfairness in RRPS. We explain such a situation as weight inversion. We then propose weight inheritance protocol(WIP), a synchronization mechanism, that prevents weight inversion. We also show that WIP can reduce unfairness using fairness analysis and simulation.

Key words : Round Robin based Proportional Share Scheduling, Group Ratio Round Robin, Synchronization Mechanism, Semaphore, Weight Inheritance Protocol, Weight Inversion

1. 서 론

컴퓨터 시스템은 시스템 자원을 효과적으로 처리할 수 있는 자원관리가 필수적이다. 이러한 자원관리 기법들 중, 간단하면서 효율적인 성능을 보이는 비례지분(proportional share) 자원관리가 있다. 특히, CPU 자원을 관리하는 비례지분 스케줄링(proportional share scheduling)은 각 태스크에게 지분(share)을 나타내는 비중(weight)이라는 속성을 정의하고 비중에 비례하여 태스크에게 CPU 자원을 할당한다. 공평성(fairness)은 태스크가 비중에 비례하여 자원을 정확하게 할당받았는지에 대한 척도로서 사용된다. 여러 비례지분 스케줄링 기법에서, 라운드 로빈 기반 비례지분 스케줄링은 일정한 시

· 이 연구는 2008학년도 숭실대학교 대학연구비의 지원으로 연구되었음

† 학생회원 : 숭실대학교 컴퓨터학과
darklight114@gmail.com
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
smyang@ssu.ac.kr

논문접수 : 2008년 11월 19일
심사완료 : 2009년 4월 18일

정보과학회논문지: 시스템 및 이론 제36권 제4호(2009.8)

간의 양을 나타내는 시간 슬롯(즉, 퀀텀)을 태스크 할당의 단위로 정의한다. 라운드 로빈 기반 비례지분 스케줄링 알고리즘으로는 WFQ[1], Lottery Scheduler[2], EEVDF[3], WF^2Q[4]가 제안되었으며, 이에 발전하여 VTRR[5], Stratified Round Robin[6], GR^3[7], FBPRR[8]등이 연구되었다. 허나 이러한 연구들은 모든 태스크들이 독립적으로 수행됨을 가정하고 있기 때문에, 태스크들 간의 동기화로 인한 스케줄링의 지연문제에 대한 연구는 부족하다.

본 논문에서는 첫째, 라운드 로빈 기반 비례지분 스케줄링 중 GR^3 스케줄링에서 태스크 동기화로 인해 스케줄링이 지연됨으로서 높은 불공평성이 발생함을 보인다. 이를 비중역전(weight inversion)이라 부른다. 비중역전은 낮은 비중을 가진 태스크로 인해 높은 비중을 가진 태스크의 실행이 지연되어 높은 불공평성을 발생시키는 현상이다. 비중역전이 발생할 때의 공평성 분석을 통해, 비중역전의 스케줄링 영향을 파악한다. 둘째, 비중역전을 방지하기 위한 동기화 기법인 비중상속 프로토콜(Weight Inheritance Protocol, WIP)을 제안한다. 비중상속 프로토콜은 태스크의 비중을 일시적으로 상속시켜 비중역전으로 인한 불공평성을 줄인다. 비중상속 프로토콜의 연산들과 비중상속 프로토콜을 위한 GR^3 스케줄링의 확장을 제안하고, 공평성 분석 및 시뮬레이션을 통해 불공평성의 감소를 보인다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로서, GR^3 스케줄링 알고리즘과 기존의 태스크 동기화 기법을 다룬다. 3장에서는 태스크 동기화 상황에서의 GR^3 스케줄링을 설명하고 그 공평성을 분석하기 위한 스케줄링 모델과 가정들을 논의한다. 4장에서는 비중역전을 설명하고 비중역전이 발생할 때의 공평성을 분석한다. 5장은 비중상속 프로토콜을 제안하고 그 공평성을 분석한다. 6장에서는 비중역전과 비중상속 프로토콜이 각각 적용되는 상황에 대한 시뮬레이션을 수행하고 그 결과로 나타나는 공평성을 비교하며, 마지막 7장에서 결론을 맺는다.

2. 관련 연구

2.1 Group Ratio Round Robin(GR^3)

GR^3는 단일프로세서를 위한 라운드 로빈 기반의 비례지분 스케줄링이다[7]. GR^3는 기존의 비례지분 스케줄링보다 나은 공평성을 보장하면서 스케줄링 오버헤드를 최소화하는 것을 목표로 한다. GR^3는 태스크들이 가진 비중(weight)에 따라 태스크들을 그룹(group)으로 묶는다. 이 때 그룹 순서(group order)라는 것을 기반으로 한다. 그룹 순서 d를 가지는 그룹에는 2^d에서부터 (2^{d+1} - 1) 사이의 비중을 가지는 클라이언트가 할당된다. 그룹은 그룹 내 모든 태스크들의 비중을 합한 것인 그룹 비중(group weight)을 가지고, 각 그룹들은 그룹 비중의 내림차순으로 정렬된 그룹 리스트에 포함된다. 만일 동일한 그룹 비중을 가진다면 그룹 순서를 비교하여 정렬된다. GR^3는 두 개의 스케줄링으로 구별된다. 그룹간 스케줄링(intergroup scheduling)은 그룹 리스트 내 그룹 비중에 비례하여 그룹을 선택한다. 그룹간 스케줄링에서 각 그룹 내 태스크에게 할당한 시간 퀀텀을 나타내기 위해 그룹 작업량(group work)을 사용한다. 이 그룹 작업량과 그룹 비중에 대한 조건(condition)을 사용하여 다음 실행될 그룹을 결정한다. 이를 GR^3에서는 잘 정렬된 조건(well-ordering condition)이라 부른다. 잘 정렬된 조건은 다음과 같다. 여기서 W_x는 그룹 작업량이고 Φ_x는 그룹 비중이다. x는 그룹 리스트 내에서 그룹이 위치하는 인덱스를 의미한다.

$$\frac{W_x+1}{\Phi_x} \le \frac{W_{x+1}+1}{\Phi_{x+1}}$$

그룹내 스케줄링(intragroup scheduling)은 선택된 그룹 내 태스크들을 라운드 로빈 방식으로 스케줄링 한다. 이때, 그룹 내에 포함되어있는 서로 다른 비중을 가진 태스크들의 스케줄링을 위해 결손(deficit)이라는 속성을 사용하여 태스크를 선택한다. 그림 1은 이러한 GR^3의 그룹간 스케줄링과 그룹내 스케줄링의 구성 예를 보여

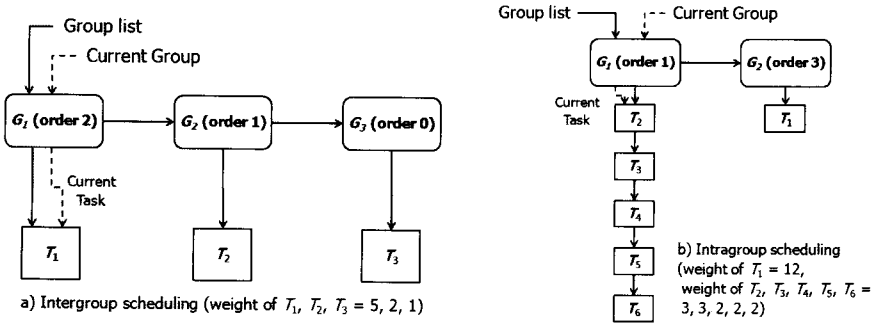

그림 1 (a) GR^3 그룹간 스케줄링과 (b) 그룹내 스케줄링의 구성 예

준다. 그림 1의 (b)에서, 그룹 G_1과 G_2는 각각 12의 동일한 그룹 비중을 가지고 있는 상황으로, 이때 그룹 리스트 내 위치 인덱스는 각 그룹의 그룹 순서를 고려하여 결정되었음을 알 수 있다.

GR^3는 공평성 측정을 위해, Generalized Processor Sharing(GPS)[9]에 기반을 둔 완전한 공평성(perfect fairness)을 사용한다. 완전한 공평성은 $W_T = W_{all}\frac{\phi_T}{\Phi_{all}}$로 나타내며, 태스크가 자신의 비중에 비례하여 완전히 할당받았음을 의미한다. 여기서, W_T는 태스크의 작업량(workload 혹은 work)으로, 태스크가 CPU 자원을 퀀텀 단위로 할당 받은 수를 의미한다. W_{all}은 모든 태스크가 수행한 작업량의 합이고 ϕ_T는 태스크의 비중이며, Φ_{all}은 모든 그룹 비중의 합으로, 즉 모든 태스크들의 비중 합이다. 완전한 공평성을 기반으로, GR^3에서는 $W_T - W_{all}\frac{\phi_T}{\Phi_{all}}$를 서비스시간에러(service time error)인 E_T로 정의한다. 만일 서비스시간에러가 양수라면 해당 태스크는 자신의 비중보다 많이 할당 받은 것이며, 서비스시간에러가 음수라면, 자신의 비중보다 덜 할당 받았음을 의미한다. GR^3에서 각 그룹 비중의 비율이 모두 자연수로 나타나는 경우 즉, $\frac{\Phi_x}{\Phi_{x+1}} \in \mathbb{N}$($\Phi_x$는 그룹 리스트 내 인덱스 x에 대한 그룹 비중)에서, GR^3의 서비스시간에러는 $-4 < E_T < g+3$(g는 그룹의 개수)이며, 그렇지 않을 때, $-\frac{g(g-3)}{2} - 3 < E_T < g+3$이다[7]. 그러나 [7]에서는 모든 태스크가 블록되거나 스스로 대기하지 않는, 즉 태스크의 실행이 독립적이라는 가정을 가지며 태스크 동기화는 전혀 고려되지 않는다.

2.2 우선순위 역전현상

태스크들 간 동기화 문제로서 널리 알려진 것으로, 우선순위 역전(priority inversion)이 있다. 특히, 제한없는 우선순위 역전(unbound priority inversion)을 해결하기 위해 제안된 동기화 기법으로 우선순위 상속 프로토콜(priority inheritance protocol, PIP)이 있다[10]. PIP는 낮은 우선순위 태스크에게 높은 우선순위 태스크의 우선순위를 일시적으로 상속(inheritance)하여 중간 우선순위 태스크에게 선점당하지 않게 함으로서 높은 우선순위 태스크의 실행 지연을 방지하는 프로토콜이다. PIP에서는 두 가지 유형의 블록킹(blocking)이 존재한다. 하나는 높은 우선순위 태스크가 낮은 우선순위 태스크에 의해 블록 되는 direct 블록킹과 나머지 하나는 낮은 우선순위 태스크가 높은 우선순위 태스크의 우선순위를 상속받아 실행될 때, 중간 우선순위 태스크가 이로 인해 블록 되는 push-through 블록킹이다.

3. 공평성 분석을 위한 GR^3 스케줄링 모델

이 장에서는 태스크 동기화가 발생하는 GR^3 스케줄링을 설명하기 위한 스케줄링 모델과 공평성 분석을 위한 가정들을 설명한다. 공평성 분석이란 서비스시간에러의 상위 경계(upper bound)와 하위 경계(lower bound)를 결정함을 의미한다[7]. 따라서 GR^3 스케줄링에 대한 불공평성(unfairness)은 기본적인 GR^3 스케줄링이 가진 서비스시간에러에서 태스크 동기화로 인해 그 범위가 증가할 때 발생한다고 말한다. [7]에서는 태스크를 클라이언트(client)라는 용어로 사용하나 본 논문에서는 태스크로 표기한다.

3.1 스케줄링 모델

본 논문에서 스케줄링 모델을 위해 사용되는 기본적인 표기들은 [7]과 [10]에 기반을 둔다. 표 1은 이러한 표기들을 나타낸다. 작업량은 퀀텀을 단위로 하는, 태스크가 CPU를 할당받아 실행되는 양을 의미하므로 GR^3에 의해 태스크가 한번 선택될 때마다 해당 태스크의 작업량은 1만큼 증가한다.

임의의 임계영역 구간에 대한 수행시간이 퀀텀보다 적으며, 해당 임계영역이 퀀텀과 퀀텀 사이에 걸쳐있지 않는, 즉 임계영역이 퀀텀에 완전히 내포되어 있는 경우, GR^3는 해당 임계영역에 대한 태스크 동기화가 필요하지 않다. 이는 한 퀀텀동안 하나의 태스크만이 실행됨을 GR^3에서 보장하기 때문이다. 이에 본 논문의 스케줄링 모델에서는 태스크 동기화를 위한 lock() 연산의 시점을 태스크가 스케줄러에 의해 선택되고 바로 수행되는 퀀텀의 시작으로, unlock() 연산의 시점을 퀀텀의 끝으로 적용한다. 따라서 임계영역 구간에 대한 수행시간은 작업량으로서 표현할 수 있다.

표 1 외에, 본 논문에서는 다음과 같은 표기들을 사용한다. 다음의 표기들은 모두 하나의 심벌(symbol)로서 사용되며 괄호안의 표기는 해당 표기의 약칭으로서 사용된다.

- $W(Z_{i,k})$ ($W_{Zi,k}$): 임계영역 $Z_{i,k}$의 길이를 작업량으로 나타낸 것이다. 즉, 임계영역 구간에 대한 작업량이며, 이를 임계영역 구간 작업량이라 부른다.
- $D_{Zi}W_j$ (DW_j) : 태스크 T_j가 임계영역 집합 Z_i에 대한 획득을 실패하여 블록킹될 때, $D_{Zi}W_j$는 블록킹 시 작업량을 포함하여 태스크 T_j가 깨어날 때까지 할당받았어야할 작업량의 길이이며 이를 지연작업량이라 부른다. 즉, $D_{Zi}W_j$는 T_j가 블록된 시점 이후로, GR^3 스케줄링에 의해 T_j가 선택되어야 할 시점마다 1씩 증가하는 가상의 작업량이다.

그림 2는 본 논문에서 사용하는 표기들의 설명과 함께, GR^3 스케줄링에서 임의의 임계영역 $Z_{i,k} = Z(W_Z =$

표 1 기본적인 표기들

T_i	태스크 i ($i \in N$, $1 \leq i \leq$ n , n은 태스크의 개수)
ϕ_i	T_i의 비중(weight) (모든 i에 대해, $\phi_i > 0$인 정수)
Φ_{all}	모든 태스크의 비중 합. 즉, $\Phi_{all} = \sum_{i=1}^{n} \phi_i$
g	그룹의 개수
G_x	정렬된 그룹 리스트에서 리스트 인덱스 x에 대한 그룹
Φ_x	그룹 G_x의 비중으로 그룹 내 태스크들의 비중 합
W_i	태스크 T_i의 작업량(workload 혹은 work)
W_{Gx}	그룹 G_x의 작업량
W_{all}	모든 그룹들의 전체 작업량
Z_i	T_i가 가지는 모든 임계영역의 집합
$Z_{i,k}$	T_i의 k번째 임계영역
$\beta_{i,j}$	T_i를 블록킹시킬 수 있는 T_j의 임계영역 집합
$S_{i,k}$	$Z_{i,k}$를 잠그고 해제하는 이진 세마포어(binary semaphore)

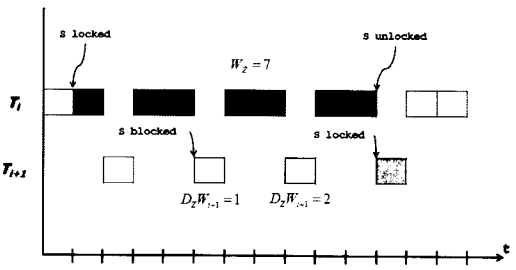

그림 2 태스크 동기화 상황에서의 GR^3 스케줄링 예

7)에 대한 세마포어 $S_{i,k} = S$를 T_i과 T_{i+1}이 접근하는 태스크 간 동기화 상황을 나타낸다. 여기서, $\phi_i = 2$이며, $\phi_{i+1} = 1$이다. 이에, GR^3 스케줄러는 각 태스크의 비중에 비례하여 스케줄링을 수행하여, T_i가 2의 작업량을 할당 받을 때마다 T_{i+1}에게 1의 작업량을 할당하여 실행시킨다. T_{i+1}이 실행중에 T_i가 이미 획득한 임계영역 Z에 접근하기 위해 lock(S)를 수행할 때, 세마포어 S는 이미 T_i가 획득한 상태이므로 T_{i+1}은 블록된다. 이 때, $D_Z W_{i+1}$은 1이 되며, 이후 T_i 의 unlock(S) 연산에 의해 T_{i+1}이 깨어날 때까지 이 $D_Z W_{i+1}$은 GR^3 스케줄링에 의해 T_{i+1}가 선택되었어야 할 때마다 1씩 증가 된다. 그림 1에서 주의 할 점은 $D_Z W_{i+1}$은 실제로는 존재하지 않는다는 것이다. 즉, 실제 스케줄링에서 T_{i+1}는 블록된 상태이므로, T_i는 임계영역 구간을 계속하여 수행한다.

3.2 공평성 분석을 위한 가정들

공평성 분석의 복잡도를 줄이고 분석의 의미를 명확히 하기위해 다음을 가정한다. 첫째, GR^3 스케줄링 알고리즘과 동일하게 단일프로세서만을 고려한다[7]. 둘째, 모든 태스크는 스스로 대기하거나 블록되지 않는다(예. I/O 대기). 따라서 모든 태스크는 임계영역에 접근하기 위해 수행되는 세마포어 연산에 의해서만 블록될 수 있다. 셋째, 특별한 언급이 없는 한, 공평성 분석 시에 모든 인접한 태스크들 간의 비중 비율이 자연수로 동일한, 즉, $\frac{\phi_i}{\phi_{i+1}} = r \in \mathrm{N}$, $1 \leq i < n$ (n은 태스크 개수)인 경우에서 동기화가 발생할 때의 공평성 분석을 수행한다. 이는 기본 GR^3 스케줄링 알고리즘의 공평성 비교를 위해서이다. 4.5절에서, 일반적인 경우에 대한 공평성을 언급한다. 넷째, 공평성 분석에서, GR^3의 스케줄링과 세마포어의 각 연산에 대한 수행시간은 없으며, 각 세마포어 연산 lock()과 unlock()은 원자적(atomic)이다. 다섯째, 교착상태(deadlock)를 피하기 위해 임계영역들은 적절히 내포(properly nested)되어 있다[10]. 따라서 임계영역은 다른 임계영역에 완전히 포함되거나 서로 겹치지 않아야 한다. 또한 교착상태 회피(avoidance)나 방지(prevention)와 같은 외부적인 방법이 적용되어 있다.

위의 가정들에 덧붙여, 공평성 분석의 단위로서 동기화 스케줄링 사이클(synchronization scheduling cycle)을 사용한다. 이 동기화 스케줄링 사이클은 VTRR에서 사용하는 스케줄링 사이클(scheduling cycle)[5]에서 유도된 것으로서, 한 태스크가 임계영역에 처음 접근한 시점에서부터 해당 임계영역을 공유하는 모든 태스크들이 임계영역의 실행을 완료하는 시점을 한 사이클로 정의한다. 단, 처음 접근된 임계영역을 제외하고 다른 태스크에 의해 다른 임계영역을 접근하지 않아야 한다. 공평성 분석은 하나의 동기화 스케줄링 사이클 내에서 수행된다. 4.5절에서 동기화 스케줄링 사이클이 아닌 경우에 대해 언급한다.

4. 비중 역전(Weight Inversion)

비중역전은 동기화가 발생될 때 낮은 비중의 태스크에 의해 높은 비중의 태스크의 실행이 지연되는 현상을 말한다. 이는 우선순위 역전(priority inversion)과 유사하다. 낮은 비중의 태스크가 획득한 임계영역에 높은 비

중의 태스크가 접근하고자 할 때, 높은 비중의 태스크는 블록킹되고 대기하게 된다. 따라서 높은 비중의 태스크는 자신의 비중에 비례하여 실행되지 못하게 되어 불공평성이 증가하게 된다. 문제는 일반적인 블록킹으로 인한 불공평성보다 비중역전으로 인한 불공평성이 상당히 높다는 점이다.

본 논문에서의 공평성 분석의 목적은 동기화 상황에서 모든 발생 가능한 스케줄링에 대한 분석이 아니라, 비중역전으로 인한 불공평성과 스케줄링 요소 간에 어떠한 관계가 있는지 파악하는데 있다. 본 논문의 공평성 분석에서 핵심적인 척도는 지연작업량인 DW이다. 태스크 동기화가 발생된 스케줄링 결과에서, 태스크 T의 작업량에 각 태스크의 DW_T가 포함되면 동기화가 발생되지 않은 상황에서의 GR^3 스케줄링이 가지는 공평성과 동일하게 된다. 예를 들어, $\frac{\phi_i}{\phi_{i+1}} \in N,\ 1 \le i < n$인 경우에 대한 GR^3 스케줄링이 가지는 태스크 T에 대한 서비스 시간에러를 E_T라고 한다면, 다음과 같이 나타낼 수 있다.

$$-4 < (W_T + DW_T) - (W_{all} + DW_T)\frac{\phi_T}{\Phi_{all}} < g+3$$

$$-DW_T(1-\frac{\phi_T}{\Phi_{all}}) - 4 < W_T - W_{all}\frac{\phi_T}{\Phi_{all}} < g+3-DW_T(1-\frac{\phi_T}{\Phi_{all}})$$

$$-DW_T(1-\frac{\phi_T}{\Phi_{all}}) - 4 < E_T < g+3-DW_T(1-\frac{\phi_i}{\Phi_{all}})$$

$$-DW_T - 4 < E_T < g+3$$

여기서, 모든 태스크 T에 대해서 $0 < 1-\frac{\phi_T}{\Phi_{all}} < 1$이다. 이는 태스크 동기화로 인한 지연작업량이 발생하려면 전체 태스크의 개수가 반드시 2보다 크거나 같아야 하기 때문이다. 결국, $\frac{\phi_i}{\phi_{i+1}} \in N,\ 1 \le i < n$인 경우의 GR^3 스케줄링에서, 동기화로 인해 발생되는 DW중에 가장 큰 DW_{max}가 존재한다면, 전체 스케줄링에 대한 서비스시간에러 E는 다음과 같다.

$$-DW_{\max} - 4 < E < g+3\ ,\ (DW_{\max} > 0)$$

이때의 DW_{max}를 최대지연작업량이라 하며, 또한 스케줄링이 최대지연작업량을 가진다고 말한다. 공평성 분석은 GR^3 스케줄링에서 비중역전이 발생하지 않는 경우와 비중역전이 발생하는 경우로 나눈다. 각 경우에 공통적으로 다음과 같은 상황을 둔다.

- GR^3 스케줄링에서 실행가능한 모든 태스크들인 T_i ($1 \le i \le n$)가 존재하고 각각에 대해 $\phi_1 \ge \phi_2 \ge \phi_3 \ge \dots \ge \phi_{n-1} \ge \phi_n$이다. 즉, T_1의 비중이 가장 크며 T_n의 비중이 가장 작다.
- 단일 동기화 스케줄링 사이클 내 모든 i에 대해, 인접한 태스크 간의 비중 비율이 자연수로 동일하다. 즉, $\frac{\phi_i}{\phi_{i+1}} = r \in N,\ 1 \le i < n, r \ge 2$.
- 임계영역 $Z_{i,k}$에 대해, 항상 $W_{Zi,k} \ge 2$이며 태스크의 개수인 n은 항상 $n \ge 2$이다.

4.1 비중역전이 발생하지 않는 경우의 공평성

단일 동기화 스케줄링 사이클에서 비중역전이 발생하지 않는 경우는 가장 높은 비중의 태스크가 먼저 임계영역을 획득하고 이어 낮은 비중의 태스크들이 임계영역에 접근할 때이다. 이러한 경우에서, 최대지연작업량이 발생되려면 임계영역을 획득한 가장 높은 비중의 태스크를 제외한 나머지 모든 태스크들이 해당 임계영역을 높은 비중 순으로 접근해야 한다.

그림 3은 이러한 스케줄링 환경을 그림으로 나타낸 것이다. 그림에서 볼 수 있듯이, 가장 높은 비중의 태스크를 제외한 모든 태스크들이 높은 비중 순으로 lock()을 바로 수행할 때 최대지연작업량이 발생할 수 있다. 이 때, 태스크의 지연작업량은 자신보다 높은 비중의 태스크들이 가진 지연작업량이 누적되기 때문에, 최대지연작업량을 가지는 태스크는 마지막에 임계영역을 수행하는 태스크인 T_n이다. 따라서 T_n의 지연작업량을 구하면 비중역전이 발생하지 않는 경우의 발생 가능한 최대지연작업량을 알 수 있다.

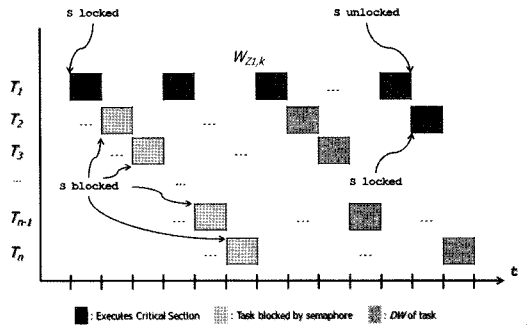

그림 3 비중역전이 발생하지 않는 상황에서의 스케줄링 환경

보조정리 1. 단일 동기화 스케줄링 사이클 내에서 비중역전이 발생하지 않는 경우, GR^3 스케줄링이 가지는 최대지연작업량 DW_n은 다음과 같다($Z = Z_{1,k}$는 모든 태스크가 공유하는 임계영역, W_Z는 임계영역 구간 작업량, $W_Z \ge 2$, $n \ge 2$).

$$DW_n \le (n-1)W_Z$$

증명. T_1이 임계영역을 획득한 태스크이므로, DW_1은 0 이다. $2 \le m \le n$인 m에 대하여, DW_m는 인접한 자신보다 바로 높은 태스크인 T_{m-1}가 임계영역의 구간

이 모두 실행될 때까지 자신의 비중만큼 수행되어야 하는 작업량이다. 따라서 DW_m는 T_{m-1}이 가졌던 지연작업량에 대한 자신의 비중만큼의 작업량과 T_{m-1}이 깨어나서 임계영역을 수행하여 unlock()을 할 때까지인, 임계영역의 구간 W_Z에 대한 자신의 비중만큼의 작업량을 합한 것이 된다. 따라서 모든 m에 대한 DW_m는 다음과 같다.

$$DW_1 = 0$$

$$DW_2 = W_Z \frac{\phi_2}{\phi_1}$$

$$DW_3 = DW_2 \frac{\phi_3}{\phi_2} + W_Z \frac{\phi_3}{\phi_2} = W_Z \frac{\phi_2}{\phi_1}\frac{\phi_3}{\phi_2} + W_Z \frac{\phi_3}{\phi_2} = W_Z \frac{\phi_3}{\phi_1} + W_Z \frac{\phi_3}{\phi_2}$$

...

$$DW_m = DW_{m-1}\frac{\phi_m}{\phi_{m-1}} + W_Z \frac{\phi_m}{\phi_{m-1}} = \left(W_Z \frac{\phi_{m-1}}{\phi_1} + W_Z \frac{\phi_{m-1}}{\phi_2} + \ldots \right.$$

$$\left. + W_Z \frac{\phi_{m-1}}{\phi_{m-2}} \right) \frac{\phi_m}{\phi_{m-1}} + W_Z \frac{\phi_m}{\phi_{m-1}} = W_Z \frac{\phi_m}{\phi_1} + W_Z \frac{\phi_m}{\phi_2} + \ldots + W_Z \frac{\phi_m}{\phi_{m-1}}$$

...

$$DW_n = W_Z \frac{\phi_n}{\phi_1} + W_Z \frac{\phi_n}{\phi_2} + W_Z \frac{\phi_n}{\phi_3} + \ldots + W_Z \frac{\phi_n}{\phi_{n-1}}$$

여기서, $1 \le m \le n-1$에 대해서, $\frac{\phi_n}{\phi_m} \le 1$이므로,

$$DW_n \le (n-1) W_Z$$

보조정리 1에서, $DW_n = (n-1) W_Z$인 경우는 모든 태스크의 비중이 동일할 때이다. GR^3 스케줄링에서 실행가능한 모든 태스크의 비중이 동일하다면, 일반적인 라운드 로빈 스케줄링과 동일하게 동작한다. 따라서 이 경우에는 어떠한 태스크가 임계영역을 획득하더라도, 나머지 다른 태스크들은 모두 블록하게 된다. 허나, 최대지연작업량은 $(n-1) W_Z$를 넘지 않는다.

4.2 비중역전이 발생하는 경우의 공평성

단일 동기화 스케줄링 사이클에서 비중역전이 발생하는 경우는 낮은 비중의 태스크가 먼저 임계영역을 획득한 이후에 보다 높은 비중의 태스크가 해당 임계영역에 접근할 때이다. 그러므로 비중역전에 의해 최대지연작업량이 발생하려면 가장 낮은 비중을 가진 태스크가 획득한 임계영역을 나머지 모든 태스크들이 낮은 비중 순으로 접근해야 한다. 그림 4는 이러한 스케줄링 환경을 그림으로 보여준다.

비중역전이 발생하지 않는 경우와 동일하게, 최대지연작업량을 가지는 태스크는 마지막으로 임계영역을 수행하는 T_1이 될 것이다. 비중역전이 존재하는 스케줄링의 공평성을 분석하기 위해, 가장 낮은 비중의 태스크가 임계영역 구간 작업량 W_Z를 2만큼 실행했을 때, 각 태스크들에 대한 지연작업량을 최초지연작업량, FDW라 한다.

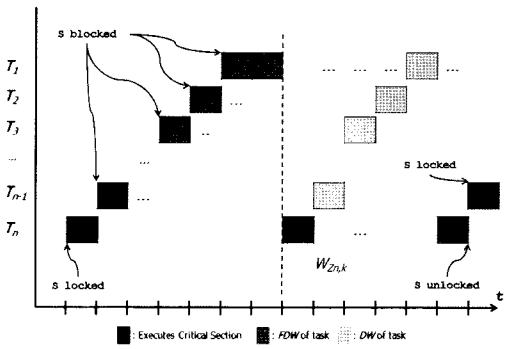

그림 4 비중역전이 발생하는 경우에서의 스케줄링 환경

보조정리 2. 단일 동기화 스케줄링 사이클 내에서 비중역전이 발생하는 경우, GR^3 스케줄링이 가지는 최대지연작업량 DW_1은 다음과 같다($Z = Z_{n,k}$는 모든 태스크가 공유하는 임계영역, W_Z는 임계영역의 구간 작업량, $W_Z \ge 2$, $r = \frac{\phi_i}{\phi_{i+1}} \in \mathrm{N}$, $1 \le i < n$, $r \ge 2$).

$$DW_1 = (W_Z - 1)\left(\frac{r^n - 1}{r - 1} - 1\right)$$

증명. T_n는 임계영역을 획득한 태스크이므로, FDW_n는 0 이다. $m = n-1$인 m에 대하여, T_{n-1}은 다른 태스크들 보다 먼저 lock() 연산을 수행한다. 따라서 FDW_{n-1}은 T_n이 1의 임계영역 구간 작업량을 실행한 것에 자신의 비중에 비례하여 수행되어야하는 작업량이다. 그러므로 FDW_{n-1}은 다음과 같다.

$$FDW_{n-1} = 1 \times \frac{\phi_{n-1}}{\phi_n} = \frac{\phi_{n-1}}{\phi_n}$$

$m = n-2$인 m에 대하여, 비중역전만이 발생하는 상황에서의 조건을 만족하기 위해서는 반드시 T_{n-1}이 블록된 후에 T_{n-2}가 임계영역을 획득하려고 해야 한다. 이를 반대로 생각한다면, T_{n-2}는 T_{n-1}이 블록될 때까지 실행중이라는 의미가 된다. 그러므로 T_{n-2}에 대한 FDW_{n-2}는 T_{n-1}이 블록되기 전까지 자신이 수행한 작업량을 배제해야 한다. T_{n-1}이 블록되기 전까지 T_{n-2}가 수행한 작업량은 $\frac{\phi_{n-2}}{\phi_{n-1}}$가 된다. 따라서 T_{n-2}에 대한 FDW_{n-2}는 T_n이 수행한 작업량에 비례한 자신의 작업량에서 T_{n-1}이 블록되기 전까지 T_{n-2}가 수행한 작업량을 뺀 것이다. 결과적으로, FDW_{n-2}는 다음과 같다.

$$FDW_{n-2} = \frac{\phi_{n-2}}{\phi_n} - \frac{\phi_{n-2}}{\phi_{n-1}}$$

따라서 DW_{n-2}는 FDW_{n-2}에 나머지 임계영역 구간인 $(W_Z - 2)$만큼과, 이전 수행된 태스크들에 대해 누적된 지연작업량을 더한 것이 된다. 나머지 $1 \le m \le n-3$

인 m에 대해서도 동일하게 적용하여 각 DW를 구하면 다음과 같다.

$$DW_n = 0$$

$$DW_{n-1} = FDW_{n-1} + (W_Z - 2)\frac{\phi_{n-1}}{\phi_n} = \frac{\phi_{n-1}}{\phi_n} + (W_Z - 2)\frac{\phi_{n-1}}{\phi_n} = (W_Z - 1)\frac{\phi_{n-1}}{\phi_n}$$

$$DW_{n-2} = FDW_{n-2} + \left((W_Z - 2)\frac{\phi_{n-1}}{\phi_n}\right)\frac{\phi_{n-2}}{\phi_{n-1}} + W_Z\frac{\phi_{n-2}}{\phi_{n-1}}$$

$$= \left(\frac{\phi_{n-2}}{\phi_n} - \frac{\phi_{n-2}}{\phi_{n-1}}\right) + (W_Z - 2)\frac{\phi_{n-2}}{\phi_n} + W_Z\frac{\phi_{n-2}}{\phi_{n-1}} = (W_Z - 1)\frac{\phi_{n-2}}{\phi_n} + (W_Z - 1)\frac{\phi_{n-2}}{\phi_{n-1}}$$

...

$$DW_m = FDW_m + \left((W_Z - 2)\frac{\phi_{m+1}}{\phi_n} + W_Z\frac{\phi_{m+1}}{\phi_{n-1}} + \ldots + W_Z\frac{\phi_{m+1}}{\phi_{m+2}}\right)\frac{\phi_m}{\phi_{m+1}} + W_Z\frac{\phi_m}{\phi_{m+1}}$$

$$= \frac{\phi_m}{\phi_n} - \left(\frac{\phi_m}{\phi_{n-1}} + \frac{\phi_m}{\phi_{n-2}} \ldots + \frac{\phi_m}{\phi_{k+1}}\right) + (W_Z - 2)\frac{\phi_m}{\phi_n} + W_Z\frac{\phi_m}{\phi_{n-1}} + \ldots + W_Z\frac{\phi_m}{\phi_{m+2}} + W_Z\frac{\phi_m}{\phi_{m+1}}$$

$$= (W_Z - 1)\frac{\phi_m}{\phi_n} + (W_Z - 1)\frac{\phi_m}{\phi_{n-1}} + \ldots + (W_Z - 1)\frac{\phi_m}{\phi_{m+2}} + (W_Z - 1)\frac{\phi_m}{\phi_{m+1}}$$

...

$$DW_1 = (W_Z - 1)\frac{\phi_1}{\phi_n} + (W_Z - 1)\frac{\phi_1}{\phi_{n-1}} + \ldots + (W_Z - 1)\frac{\phi_1}{\phi_2}$$

여기서, $\phi_1 = r\phi_2 = r^2\phi_3 = \ldots = r^{n-1}\phi_n$ 임을 이용하면,

$$DW_1 = (W_Z - 1)(r^{n-1} + r^{n-2} + \ldots + r^2 + r)$$

$$= (W_Z - 1)\left(\frac{r^n - 1}{r - 1} - 1\right)$$

보조정리 2를 통해, 비중역전이 발생하는 경우에서 최대지연작업량은 태스크들 간의 비중 차이와 태스크의 개수에 대해 증가함을 알 수 있다. 이제, 단일 동기화 스케줄링 사이클에서 비중역전으로 인한 공평성을 파악할 수 있다.

정리 1. 모든 태스크 i에 대해, 인접한 태스크의 비중 비율이 자연수로 동일한 경우, 즉 $\frac{\phi_i}{\phi_{i+1}} = r \in \mathbb{N}, r \geq 2$ 인 GR^3 스케줄링이 존재한다. 이 때, 단일 동기화 스케줄링 사이클에서 비중역전이 존재하면, 서비스시간에러 E는 다음과 같다($Z = Z_{n,k}$는 모든 태스크가 공유하는 임계영역, W_Z는 임계영역 구간 작업량, $W_Z \geq 2$, n 은 태스크의 개수, n $\geq$ 2, g는 그룹의 개수).

$$-(W_Z - 1)\left(\frac{r^n - 1}{r - 1} - 1\right) - 4 < E < g + 3$$

증명. 보조정리 1과 보조정리 2에서 최대지연작업량이 동일한 경우는 $r = 2$, $n = 2$, $W_Z = 2$인 경우이다. 이를 제외하고는 항상 $(W_Z - 1)\left(\frac{r^n - 1}{r - 1} - 1\right) > (n - 1)W_Z$이다. 따라서 비중역전이 존재한다면 발생 가능한 최대지연작업량은 어떠한 경우라도 $(W_Z - 1)\left(\frac{r^n - 1}{r - 1} - 1\right)$ 보다 클 수 없다.

5. 비중상속 프로토콜

이 장에서는 비중역전으로 인한 불공평성을 감소하는 동기화 기법인 비중상속 프로토콜(Weight Inheritance Protocol, WIP)을 제안한다. WIP의 목적은 비중역전이 스케줄링 내에서 발생하지 않도록 방지하는데 있다. WIP은 낮은 비중의 태스크가 획득하고 있는 임계영역에 높은 비중의 태스크가 접근하려 할 때, 자신의 비중을 낮은 비중의 태스크에게 상속시킨다. 따라서 높은 비중의 태스크는 임계영역을 획득하는 시간을 앞당길 수 있게 되어, 비중역전으로 발생될 수 있는 불공평성을 감소시킨다. WIP를 명확하게 설명하기 위해, WIP에서는 카운트를 1로 설정한 카운팅(counting) 세마포어를 사용한다.

5.1 WIP를 적용한 세마포어 연산

WIP는 세마포어 연산인 lock()과 unlock()에 각각 적용된다(그림 5). 상속을 판단하기 위해서, 세마포어는 해당 세마포어의 소유자 태스크를 나타내는 *Holder*를 가진다. 현재 실행중인 태스크($T_{current}$)가 블록될 때, *Holder*의 비중과 자신의 비중을 비교한다. 만일 자신의 비중이 크다면 *Holder*에게 자신의 비중을 상속한 다음 블록된다. 이 때, *Holder*는 자신이 속한 그룹에서 상속된 비중에 대한 그룹으로 이동한다. 이러한 이동은 GR^3 스케줄러에 의해 수행된다. 만일 세마포어를 $T_{current}$가 획득한다면, 비중을 세마포어의 진입비중($\phi^{Entered}$)에 저장한다. 진입비중은 unlock()에서 사용된다.

해당 임계영역을 벗어나기 위해 $T_{current}$(*Holder*)가 unlock()을 수행하면, $T_{current}$는 해당 세마포어의 큐에서 블록된 태스크를 깨운다. 만일 세마포어의 큐가 비어 있지 않으면서 깨어난 태스크(T_{wakeup})의 비중이 $T_{current}$의 비중보다 적다면, T_{wakeup}는 현재 태스크의 비중을 상속받는다. 이후 $T_{current}$는 자신의 원래 비중($\phi^{orig}{}_{current}$)과 비교하여 크다면, 상속된 상태를 의미하므로 $\phi^{orig}{}_{current}$로 돌아가 수행을 계속한다. 이 때, 해당 세마포어가 다른 세마포어에 의해 내포된 상태라면 원래 비중이 아닌 이전 세마포어에 의해 상속된 비중으로 조정되어야 하므로, 해당 세마포어를 lock()했을 때 저장된 진입 비중인 $\phi^{Entered}$로 조정되어 수행을 계속한다. 위의 설명한 것처럼, unlock() 시 상속된 태스크는 원래의 비중으로 돌아가야 하지만, 그림 5에서는 포함되어 있지 않다. 이는 5.3절에서 다룰 스케줄링 확장을 위한 것이다. 또한, 설명하지 않은 상속작업량(Inheritance Work, *IW*)과 지연결손(Delay Deficit, *DD*)도 5.3절에서 다룬다.

그림 5에서 세마포어의 큐(queue)와 스케줄러가 가지는 대기큐(waitqueue)의 연산인 push()와 pop()은 스택(stack)과 다르게 사용된다. push()는 해당 큐의 마지막에 태스크를 위치시키는 연산이고 pop()은 큐에서 첫 번째로 위치하는 태스크를 가져오는 연산이다.

5.2 일시적인 공평성과 서비스 시간의 증가

```
count = 1
lock()
{
  count--
  if (count < 0){
    if(φ_Current > φ_Holder){
      if(φ_Holder == φ_Holder^orig)
        IW_Holder = 0
      /* ← : inherit from */
      φ_Holder ← φ_Current
    }
    queue.push(T_current)
    Block T_current
  }
  Holder = Current
  φ^Entered = φ_Current
}
```

```
unlock()
{
  count++
  if (count ≤ 0) {
    T_Wakeup = queue.pop()
    if (queue is not empty and φ_Wakeup < φ_Current)
      φ_Wakeup ← φ_Current

    Wakeup T_Wakeup

    if(φ_Current > φ_Current^orig) {
      if(Current has nested semaphore include S)
        φ_Current = φ^Entered
      else {
        DD_Current = -IW_Current
        Scheduler.waitqueue.push(T_Current)
      }
    }
  }
}
```

그림 5 WIP가 적용된 세마포어 lock()연산(좌), unlock()연산(우)

WIP를 적용하면 비중역전으로 인한 불공평성을 줄일 수 있다. 그러나 WIP가 적용됨으로서, GR^3 스케줄링 상황에서 다음의 현상이 발생한다. 첫째는 일시적인 불공평성(temporary unfairness)이고 둘째는 서비스 시간의 증가이다. 일시적인 불공평성은 낮은 비중의 태스크가 높은 비중의 태스크의 비중을 상속할 때, 낮은 비중이 그동안 수행하던 작업량에 비해, 높은 비중의 태스크에 해당하는 작업량을 가지지 못함으로서, 낮은 비중의 태스크가 불공평성을 가지는 현상이다. 그러나 일시적인 불공평성은 임계영역을 수행할 때까지만 나타나며, 가장 중요한, 높은 비중을 가진 태스크에 대한 공평성에 전혀 영향을 끼치지 않는다.

일시적인 불공평성 현상과 비슷하게, 서비스 시간의 증가는 WIP가 적용될 때 낮은 비중의 태스크에게 나타난다. 서비스 시간의 증가 문제는 낮은 비중의 태스크가 자신의 원래 비중으로 돌아와서 수행되는, 임계영역 구간의 실행완료 시점에서 발생한다. 낮은 비중의 태스크가 비중을 상속받아 임계영역을 수행하는 동안 자신의 작업량을 상속받은 비중에 비례하여 증가시키기 때문에, 자신의 비중으로 돌아오는 시점에서, 낮은 비중의 태스크는 자신의 비중보다 더 많이 실행한 것이 된다. 이는 낮은 비중의 태스크의 서비스시간에러에 대해 음수가 아닌 양수 값이 증가되는 불공평성을 발생시키게 된다. 따라서 이를 방지하는 방법이 필요하다.

5.3 WIP를 위한 GR^3 스케줄링 확장

서비스 시간의 증가를 방지하기 위해서, 비중을 상속받았던 태스크에 대한 지연(delay) 방법을 사용한다. 서비스 시간의 증가는 임계영역에서 빠져나간 이후에 발생되기 때문에, 세마포어 연산인 lock()이나 unlock()에서 모두 해결할 수 없다. 이런 이유로, GR^3 스케줄링 알고리즘을 확장한다. 이를 위해, 태스크는 각각 상속작업량(Inheritance work, IW)과 지연결손(Delay Deficit, DD)이라는 속성을 갖는다. 상속작업량과 지연결손의 사용은 그림 2에 포함되어 있다. lock()에서, 만일 태스크가 비중을 처음 상속받는다면 상속작업량은 0으로 초기화 되며, 이후 임계영역을 수행할 때 마다 1만큼 증가된다. 태스크가 마지막 임계영역을 빠져나갈 때, 태스크는 지연결손을 상속작업량의 음수로서 설정하고 스케줄러가 가진 대기 큐(wait queue)에 들어간다. 이때 해당 태스크를 지연태스크(delayed task, $T_{delayed}$)라 한다.

스케줄링이 이루어질 때마다, 대기 큐 내에 지연태스크에 대한 실행 여부를 검사한다. 이러한 지연태스크의 검사 알고리즘은 그림 6에서 나타난다. 기본적인 개념은 지연태스크가 GR^3 스케줄링에 의해 선택될 수 있는 조건에 만족한다면 지연태스크의 지연결손을 증가시키고, 지연결손이 0에 도달할 때, 지연태스크를 깨우는 것이다. 이를 위해, 해당 태스크에 대한 가상작업량(pseudo work, W^{pseudo})을 계산하여 가상작업량이 스케줄링의 조건에 만족할 때마다 지연결손을 증가시킨다. 이 스케줄링 조건을 가상조건(pseudo condition)이라 한다. 그림 7

```
check_delayed_task()
{
  while(each T_delayed)  {
    if(G(T_delayed) is not empty)  {
      if(next group is G(T_delayed) and pseudo_condition)  {
          Δ_delayed = Δ_delayed + (φ_delayed / Φ^G_min − 1)
          if( Δ_delayed ≥ 1)
              DD_delayed = DD_delayed + 1
          else
              DD_delayed = 1 / (N_G + 1)
      }
    } else {
      if(Φ_Current > φ_delayed > Φ_next and pseudo_condition)
          DD_delayed = DD_delayed + 1
    }

    if(DD_delayed ≥ 0)
      Wakeup T_delayed
  }
}
```

그림 6 지연태스크 검사 알고리즘

은 가상조건을 보여준다. 가상조건은 GR^{3}의 그룹간 스케줄링에서 유도된 것이다[7]. 지연태스크에 대한 그룹($G(T_{delayed})$)의 존재 유무에 따라 지연결손의 증가량을 다르게 설정한다. 그룹이 존재하지 않는 지연태스크는 지연결손을 1씩 증가시키고 그룹이 존재하는 경우는 1 혹은 (1/(그룹 내 태스크 수(N_G)+1))만큼 증가시킨다. 이를 구별하기 위해 현재 스케줄링 상에서 그룹이 존재하는 지연태스크들에 대해 각각 공제(credit, Δ_{delayed})를 설정한다. 공제는 ((지연태스크의 비중($\phi_{delayed}$)/그룹 내 최소 비중(Φ^G_{min})) − 1)만큼 증가한다. 이는 GR^{3}의 그룹 내 스케줄링에 기반을 둔 것이다[7].

결과적으로, 지연태스크 검사를 수행하여 태스크를 지연시키면, 낮은 비중의 태스크는 자신의 비중에 맞게 수행된다. 게다가 지연태스크들은 자신의 상속작업량만큼 지연되기 때문에 해당 임계영역 구간 이상 지연되지 않는다. 또한, 지연태스크 검사 알고리즘은 기존 GR^3 스케줄링 알고리즘에는 전혀 영향을 주지 않는다.

5.4 WIP의 공평성 분석

보조정리 2에서의, 비중역전이 발생할 때, WIP를 적용시켜 최대지연작업량을 파악한다. 이에, 비중역전 시 발생되는 최대지연작업량을 얼마나 줄일 수 있는지 알 수 있다.

보조정리 3. 단일 동기화 스케줄링 사이클 내에서 비중역전이 발생할 때 비중상속 프로토콜을 적용한다면, 최대지연작업량 DW_1은 다음과 같다(n은 태스크의 개수, $Z = Z_{n,k}$ 는 모든 태스크가 공유하는 임계영역, W_z는 임계영역 구간 작업량, $W_z \geq 2$).

$$DW_1 = (n-1)W_Z - 1$$

증명. 보조정리 2에서 WIP가 적용될 때, T_n을 제외한 나머지 태스크들에 대한 FDW는 동일 비중의 태스크 간에 발생하는 FDW가 되므로 모두 1이다. 또한, 모든 태스크가 블록된 후 T_n이 가지는 비중은 가장 높은 비중인 ϕ_1과 같아지므로 나머지 임계영역 구간에 대해서 지연작업량은 ϕ_1에 대한 지연작업량으로 계산된다. 이후, 나머지 태스크들은 모두 ϕ_1을 상속받아 실행된다. 결과적으로, $1 \leq m \leq n$ 인 m에 대해서, 지연작업량 DW_m 및 최대지연작업량인 DW_1을 구하면 다음과 같다.

$$DW_n = 0$$

$$DW_{n-1} = FDW_{n-1} + (W_Z - 2)\frac{\phi_{n-1}}{\phi_n} = 1 + (W_Z - 2)\frac{\phi_{n-1}}{\phi_1}$$

$$DW_{n-2} = FDW_{n-2} + (W_Z - 2)\frac{\phi_{n-2}}{\phi_1} + W_Z\frac{\phi_{n-2}}{\phi_1} = 1 + (W_Z - 2)\frac{\phi_{n-2}}{\phi_1} + W_Z\frac{\phi_{n-2}}{\phi_1}$$

...

$$\begin{aligned} DW_m &= FDW_m + \left((W_Z - 2)\frac{\phi_{m+1}}{\phi_1} + W_Z\frac{\phi_{m+1}}{\phi_1} + \ldots + W_Z\frac{\phi_{m+1}}{\phi_1}\right)\frac{\phi_m}{\phi_{m+1}} + W_Z\frac{\phi_m}{\phi_1} \\ &= 1 + (W_Z - 2)\frac{\phi_m}{\phi_1} + W_Z\frac{\phi_m}{\phi_1} + \ldots + W_Z\frac{\phi_m}{\phi_1} + W_Z\frac{\phi_m}{\phi_1} \\ &= 1 + (W_Z - 2)\frac{\phi_m}{\phi_1} + (n-m-1)W_Z\frac{\phi_m}{\phi_1} \end{aligned}$$

...

$$\begin{aligned} DW_1 &= 1 + (W_Z - 2)\frac{\phi_1}{\phi_1} + W_Z\frac{\phi_1}{\phi_1} + W_Z\frac{\phi_1}{\phi_1} + W_Z\frac{\phi_1}{\phi_1} + \ldots + W_Z\frac{\phi_1}{\phi_1} \\ &= 1 + (W_Z - 2) + W_Z + W_Z + W_Z + \ldots + W_Z \\ &= 1 + (W_Z - 2) + (n-2)W_Z \\ &= (n-1)W_Z - 1 \end{aligned}$$

$$pseudo\_condition = \begin{cases} \dfrac{W_G + 1}{W^{pseudo}_{G_{delayed}} + 1} > \dfrac{\Phi_i}{\phi_{delayed}} \quad \left(W^{pseudo}_{G_{delayed}} = \left\lfloor W_T \dfrac{\phi_{delayed}}{\Phi_{all}} \right\rfloor\right) & \textbf{if } G(T_{delayed}) \textbf{ was empty} \\ \dfrac{W_G + 1}{W^{pseudo}_{G_{delayed}} + 1} > \dfrac{\Phi_i}{\Phi_{G_{delayed}}} \quad \left(W^{pseudo}_{G_{delayed}} = \left\lfloor W_{G_{delayed}} \dfrac{\Phi^{pseudo}_{G_{delayed}}}{\Phi_{G_{delayed}}} \right\rfloor, \Phi^{pseudo}_{G_{delayed}} = \Phi_{G_{delayed}} + \phi_{delayed}\right) & \end{cases}$$

그림 7 가상조건(pseudo condition)

비중상속 프로토콜은 비중역전으로 발생할 수 있는 최대지연작업량을 비중역전이 발생하지 않는 경우에 발생할 수 있는 최대지연작업량에 가깝도록 줄인다. 따라서 비중역전으로 발생하는 불공평성을 상당히 감소시킨다.

정리 2. 모든 태스크 i에 대해, 인접한 태스크의 비중 비율이 자연수로 동일한 경우, 즉 $\frac{\phi_i}{\phi_{i+1}} \in \mathrm{N}$인 GR^3 스케줄링 환경이 존재한다. 이 때, 단일 동기화 스케줄링 사이클에서 비중상속 프로토콜을 적용하면 서비스시간에러 E는 다음과 같다(n은 태스크의 개수이며 n ≥ 2, $Z = Z_{n,k}$는 모든 태스크가 공유하는 임계영역, W_Z는 임계영역 구간 작업량, $W_Z \geq 2$, g는 그룹의 개수).

$$-(n-1)W_Z - 4 < E < g+3$$

증명. 보조정리 3에 의해, 비중상속 프로토콜이 적용된 단일 동기화 스케줄링 사이클에서 발생 가능한 최대지연작업량은 $(n-1)W_Z$보다 클 수 없다. 만일 비중상속 프로토콜이 적용된다면 비중역전이 발생한다는 의미이므로 적어도 하나의 태스크에 의해 발생될 것이다. 따라서 비중역전이 발생되지 않는 태스크들의 최대지연작업량 또한 $(n-1)W_Z$보다 항상 작다. 또한 보조정리 1에 의해, 비중역전이 전혀 발생되지 않는다면 최대지연작업량은 $(n-1)W_Z$이다. 따라서 모든 경우, 최대지연작업량은 $(n-1)W_Z$보다 클 수 없다.

5.5 일반적인 경우의 공평성

5.5.1 일반적인 GR^3 스케줄링 환경에서의 공평성

인접한 태스크간의 비중 비율이 자연수가 아닌 일반적인 비중들을 가진 태스크 집합에 대한 GR^3 스케줄링 환경을 논의한다. 보조정리 1과 비슷하게, 비중역전이 발생하지 않는 경우, 태스크 m이 각각 가질 수 있는 최대지연작업량 DW_m는 다음과 같다.

$$DW_m = W_Z \sum_{h=1}^{m-1} \frac{\phi_m}{\phi_h},$$

$$\left(1 < m \leq n,\ \phi_1 \geq \phi_2 \geq \ldots \geq \phi_{m-1} \geq \phi_m \geq \phi_{m+1} \geq \ldots \geq \phi_n\right)$$

일반적인 GR^3 스케줄링에서는 모든 태스크들 중에 최대지연작업량을 갖는 태스크는 마지막에 깨어나는 태스크인 T_n이 아닐 수 있다. 보조정리 1의 경우에서, 비중 간 비율이 1이 아니라면 모든 태스크의 비중이 다르지만 일반적인 경우에서는 중간 비중이 동일한 태스크들의 부분 집합이 존재할 수 있기 때문에, 그 보다 낮은 비중의 태스크가 가지는 지연작업량보다 더 많은 지연작업량을 가질 수 있다. 이는 동일한 중간 비중을 가진 태스크들의 개수와, 해당 비중보다 낮은 비중간의 비율에 따라 결정된다. 따라서 최대지연작업량을 DW_{max}라 한다면, 다음과 같다.

$$DW_{\max} = \max\left\{DW_m \mid DW_m = W_Z \sum_{h=1}^{k-1} \frac{\phi_k}{\phi_h}\right\}$$

그러나 어떠한 경우라도, 모든 태스크의 비중이 동일한 경우에 나타나는 최대지연작업량인 $(n-1)W_Z$을 넘지 못하기 때문에, 일반적인 GR^3 스케줄링 환경에서 공평성인 서비스시간에러는 다음과 같다.

$$-(n-1)W_Z - \frac{g(g-3)}{2} - 3 < E < g+3$$

보조정리 2의, 비중역전이 발생하는 경우, 태스크 m의 FDW_m이 보조정리 2에서의 FDW_m을 갖지 않을 수 있다. 이는 태스크들의 비중 분포에 따라 높은 비중을 가진 태스크보다 낮은 비중을 가진 태스크가 먼저 수행될 수 있기 때문이다. 그러나 어떠한 경우라도, FDW_m는 자신의 비중에 비례한 값 보다 높을 수 없다. 그러므로 태스크 m이 가질 수 있는 지연작업량 DW_m는 다음과 같다.

$$DW_m = \frac{\phi_m}{\phi_n} + (W_Z - 2)\frac{\phi_m}{\phi_n} + W_Z \frac{\phi_m}{\phi_{n-1}} + \ldots + W_Z \frac{\phi_m}{\phi_{k+1}}$$

$$DW_m = (W_Z - 1)\frac{\phi_m}{\phi_n} + W_Z \sum_{l=m+1}^{n-1} \frac{\phi_m}{\phi_l},$$

$$\left(1 \leq m < n,\ \phi_1 > \ldots > \phi_{m-1} > \phi_m > \phi_{m+1} > \ldots > \phi_n\right)$$

또한, 비중역전을 일으키는 태스크들의 집합 내 동일한 비중을 갖는 태스크는 존재할 수 없기 때문에, 최대지연작업량은 DW_1이 된다. 그러므로 서비스시간에러는 다음과 같다.

$$-(W_Z - 1)\frac{\phi_1}{\phi_n} - W_Z \sum_{l=2}^{n-1} \frac{\phi_1}{\phi_l} - \frac{g(g-3)}{2} - 3 < E < g+3$$

만일 비중상속 프로토콜이 적용된다면, 태스크들의 비중 분포에 관계없이 스케줄링이 가지는 최대지연작업량은 $(n-1)W_Z - 1$이므로 서비스시간에러는 다음과 같다.

$$-(n-1)W_Z - \frac{g(g-3)}{2} - 3 < E < g+3$$

5.5.2 단일 스케줄링 사이클이 아닌 환경에서의 공평성

단일 스케줄링 사이클이 아닌 일반적인 경우는 크게 두 가지 상황으로 구별할 수 있다: 1) 단일 스케줄링 사이클이 여러 번 발생하는 경우 혹은 2) 임의의 스케줄링 구간에서 여러 임계영역을 태스크가 획득하려고 하는 경우. 1)의 경우는 모든 태스크들이 주기적으로 해당 임계영역을 접근하려고 한다면 발생할 수 있다. 이 때 발생되는 최대지연작업량은 단일 스케줄링 사이클에서 발생하는 최대지연작업량에 주기를 곱한 값이 최대지연작업량이 될 것이다. 2)의 경우, 임의의 스케줄링 구간에서 여러 임계영역을 태스크가 획득할 때 태스크가 획

득하려는 임계영역의 구간 작업량인 W_Z에 의해 결정될 수 있다. 따라서 이러한 경우는 각각의 임계영역에 대한 지연작업량을 분석해야만 최대지연작업량을 파악할 수 있다. 그러나 2)의 경우도 단일 스케줄링 사이클로 분리하여 분석하는 것이 가능하다.

5.6 비중상한 프로토콜(Weight Ceiling Protocol, WCP)

WIP는 비중역전으로 인한 불공평성을 감소시키지만, 두 가지의 단점을 가진다. 첫 번째는WIP 자체로서 교착상태(deadlock)를 방지할 수 없다는 것이고 두 번째는 연속된 블록킹(chained blocking)으로 인해 태스크가 가지는 지연작업량이 증가할 수 있다는 것이다. 이러한 단점을 해결하기 위하여, 우선순위 상한 프로토콜[10]과 비슷한 비중상한 프로토콜(Weight Ceiling Protocol, WCP)을 생각할 수 있다. WCP는 임계영역마다 태스크의 비중에 따른 상한(ceiling)값을 정의하고 태스크가 임계영역에 접근할 때 다른 잠겨있는 임계영역에 대한 상한 값을 비교하여 해당 상한 값보다 크지 않다면 태스크를 블록시킴으로서, 교착상태와 연속된 블록킹을 일으키지 않게 한다. 그러나 이 WCP는 우선순위 상한 프로토콜과 비슷하게 상한 블록킹(ceiling blocking)이라는 비관적인(pessimistic) 블록킹을 발생시킬 수 있다[10]. 이러한 상한 블록킹으로 인해 특정 태스크 집합에 대해서는 WCP를 적용하는 것이 WIP를 적용한 것보다 더 높은 불공평성을 일으킬 수도 있다. WCP가 WIP보다 더 효율적인 프로토콜이 되기 위해서는 각 태스크의 임계영역에 대한 정보들을 실행 시(run-time)에 검사하여 불필요한 상한 블록킹이 발생하지 않도록 하는 상한 검사 메커니즘이 필요할 것이다.

6. 시뮬레이션 측정

본 논문에서 시뮬레이션은 보조정리에 의해 증명된 상황뿐만 아니라 실제 수행될 수 있는 스케줄링 환경을 적용하여 비중역전으로 인한 불공평성이 얼마나 심각한가와 WIP로 인해 불공평성이 얼마나 감소되는 가를 확인하기 위해 실험한다. 이에 시뮬레이션에서는 인접한 태스크간의 비중 비율이 자연수가 아닌 태스크 집합과 태스크가 여러 세마포어를 획득하는 GR^3 스케줄링 환경을 설정한다.

시뮬레이션에는 기본적인 GR^3 스케줄링과 더불어 태스크 동기화를 위한 세마포어가 포함된다. 이 세마포어는 기본 세마포어와 WIP 및 스케줄링 확장이 적용된 세마포어로 각각 구현되어 있다. 이러한 시뮬레이션 환경에서, 태스크 입력 데이터는 태스크 이름과 태스크 비중을 기본으로 하고, 획득하려하는 세마포어의 이름, 세마포어를 획득하는 시점을 나타내는 작업량, 세마포어가 나타내는 임계영역의 구간 작업량이 포함될 수 있다. 태스크 집합에서는 임계영역을 획득하지 않는 태스크도 포함되어 있다. 예를 들어, "T1 10 S1 3 10 S2 20 10"은 10의 비중을 가지는 T1이라는 태스크가 세 번째로 CPU를 할당받을 때 S1이라는 세마포어를 획득하기 위해서 lock(S1)을 수행하고 만일 S1을 획득한다면 S1에 대한 임계영역 구간 작업량인 10만큼 수행하며, 이후 T1의 작업량이 20일 때 세마포어 S2에 대한 임계영역을 획득하려고 하며 이때 임계영역의 구간 작업량은 10이라는 의미를 가진다. 태스크의 개수는 10에서 200까지 변화하며, 각각의 태스크 집합 내에서 태스크가 가지는 비중은 최대 2000으로 제한한다. 한 태스크 집합에 대한 시뮬레이션 측정 시간을 제한하기 위해 하나의 태스크가 획득할 수 있는 세마포어의 개수는 최대 5개로 설정한다.

이러한 환경과 태스크 집합 입력 데이터를 사용하여, 비중역전이 발생할 때 기본 세마포어가 수행된 스케줄링 결과에 대한 서비스시간에러와, 동일한 집합에서 WIP 및 스케줄링 확정이 적용된 세마포어가 수행 시의 서비스시간에러를 측정한다. 각 집합에 대해서 임계영역 구간 작업량 W_Z가 10에서 100을 가질 때에 대해 각각 측정한다. 표 2는 태스크 개수가 각 N개인 집합에서 시뮬레이션 결과의 일부이다. 표 2는 서비스시간에러에서 최소인 음수 값과 최대인 양수 값이 포함되어 있다.

표 2에서, GR^3 스케줄링에서 비중역전이 발생하는 경우 태스크의 개수와 임계영역의 길이에 따라 증가함을 볼 수 있다. 표 2에서 태스크의 개수가 150이고 임계영역의 구간 작업량이 100일 때 발생한 서비스시간에러의 최소는 -212155이다. 이는 퀀텀의 길이가 10 마이크로초라면 해당 태스크 집합에서 비중역전에 의해 태스크가 최대 약 2.1초의 할당을 비중에 비례하여 받지 못함을 의미한다. 이는 비중간의 차이가 매우 큰 태스크 간에 비중역전으로 인해 발생함을 알 수 있다. 특히 위의 경우, 높은 비중의 태스크가 가진 내포된 임계영역 구간들에 대해서 각각 비중역전이 발생하는 연쇄 블록킹으로 인해 서비스시간에러의 범위가 더욱 더 증가한 결과를 보였다. 반면, WIP가 적용된 동일한 집합에서는 서비스시간에러의 최소가 -7097로, WIP에 의해 약 2초가 감소됨을 보인다. 이는 WIP가 적용됨으로서 비중역전이 발생하지 않아 결국 동일한 비중의 태스크 간 동기화로 인한 작업량 손실만이 발생한 결과이다. 그러나 WIP가 적용되더라도 해당 경우와 같은 연쇄 블록킹은 여전히 발생하며 이로 인한 작업량의 손실은 줄일 수 없었다.

또한, 표 2를 통해 WIP를 위한 GR^3 스케줄링 확장이 적용되어 있기 때문에 WIP가 적용되더라도 서비스시간에러의 최대는 거의 변하지 않음을 볼 수 있다. 만일 GR^3 스케줄링 확장이 적용되지 않으면 서비스시간에러

표 2 시뮬레이션 측정 결과

n	임계영역 구간 작업량(W_Z)											
	10				50				100			
	GR^3		GR^3+WIP		GR^3		GR^3+WIP		GR^3		GR^3+WIP	
10	-270	0.95	-31	0.95	-1390	0.95	-151	0.95	-2790	0.95	-301	0.95
30	-806	1.68	-177	1.68	-4100	1.68	-447	1.68	-8216	1.68	-897	1.68
60	-1457	1.56	-101	1.63	-7325	1.68	-501	1.63	-14657	1.74	-1001	1.63
70	-4278	1.55	-279	1.62	-21442	1.8	-1399	1.62	-42918	1.55	-2799	1.62
90	-6953	1.7	-369	1.7	-34741	1.7	-1849	1.7	-69533	1.7	-3699	1.7
100	-9892	1.49	-469	1.54	-49460	1.56	-2349	1.54	-98992	1.49	-4699	1.54
150	-21235	1.62	-707	1.62	-106073	1.66	-3547	1.62	-212155	1.62	-7097	1.62
200	-24393	1.66	-757	1.65	-121824	1.71	-3797	1.65	-222853	1.64	-7597	1.65

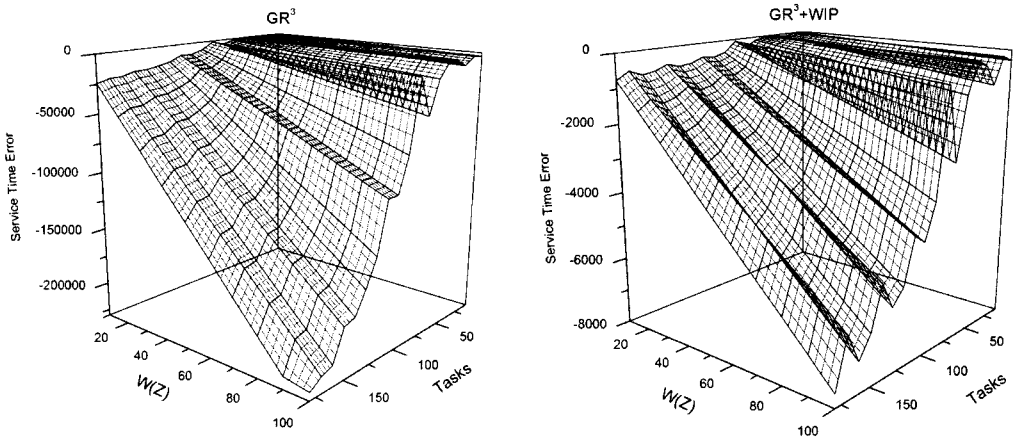

그림 8 시뮬레이션 측정 결과 그래프 좌) GR3, 우) GR3+WIP

의 최대는 WIP에 의해 감소된 서비스시간에러의 값만큼 증가된다. 예를 들어, n=10이고 W_Z=10일 때 GR^3 스케줄링 확장이 적용되지 않은 경우, 최소는 -31이고 최대는 241이 측정되는 결과를 얻는다. 그림 8은 시뮬레이션 측정 결과를 그래프로 나타낸 것이다. 그림 8의 좌측 그래프는 기본 GR^3 스케줄링의 서비스시간에러이며, 우측그래프는 WIP가 적용된 스케줄링의 서비스시간에러이다. 표 2에서 알 수 있는 것처럼, 서비스시간에러의 최대는 거의 변하지 않기 때문에 그림 8의 그래프는 서비스시간에러의 최소만을 표시한다.

7. 결론 및 향후 연구 방향

본 논문에서는 라운드 로빈 기반 비례지분 스케줄링인 GR^3에서 발생할 수 있는 동기화 문제인 비중역전을 설명하고 그 공평성을 분석하여 비중역전이 불공평성을 상당히 증가시킴을 보였다. 이러한 비중역전을 해결하기 위해 비중상속 프로토콜과 이를 위한 GR^3 스케줄링의 확장을 제안하였다. 또한, 비중상속 프로토콜의 공평성을 분석하여 비중상속 프로토콜이 비중역전에서 발생할 수 있는 불공평성을 감소시킴을 보였다. 본 논문의 기대효과는 다음과 같다. 첫째, 다른 여러 비례지분 기반 스케줄링에서의 동기화 문제들의 연구에 기반이 될 수 있다. 둘째, 제안된 프로토콜을 적용시킴으로서 다른 비례지분 기반 스케줄링에서도 발생할 수 있는 불공평성을 줄일 수 있다. 향후 연구로는 5.6절에서 언급한 비중상속 프로토콜의 단점인 교착상태와 연쇄 블록킹을 방지할 수 있는 프로토콜인 비중상한 프로토콜(Weight Ceiling Protocol, WCP)을 연구한다. 이는 비중상속 프로토콜과 동일하거나 더 나은 공평성을 가짐을 목적으로 한다.

참 고 문 헌

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

[2] C. A. Waldspurger, "Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management," PhD Thesis No. MIT/LCS/TR-667, Dept. of Electrical Eng. and Computer Science, Massachusetts

Inst. of Technology, 1995.
[3] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and C. Plaxton, "A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems," *Proc. IEEE Real-Time Systems Symp.*, p.288, Dec. 1996.
[4] J. Bennett and H. Zhang, "WF2Q: Worst-case Fair Weighted Fair Queueing," in *Proceedings of INFOCOM '96*, San Francisco, CA, Mar. 1996.
[5] J. Nieh, C. Vaill, and H. Zhong, "Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler," *Proc. General Track: 2002 USENIX Ann. Technical Conf.*, pp.245-259, June 2001.
[6] S. Ramabhadran and J. Pasquale, "Stratified Round Robin: A Low Complexity Packet Scheduler with Bandwidth Fairness and Bounded Delay," *Proc. ACM SIGCOMM*, pp.239-249, 2003.
[7] J. Nieh, C. Vaill, and H. Zhong, "Group Ratio Round-Robin: An O(1) Proportional Share Scheduler," *Proc. General Track: 2004 USENIX Ann. Technical Conf.*, pp.245-259, June 2004.
[8] A. Sarkar, P. P. Chakrabarti, and R. Kumar, "Frame-Based Proportional Round-Robin," *IEEE Transactions on computers*, vol.55, no.9, pp.1121-1129, Sep. 2006.
[9] L. Kleinrock, "Queueing Systems, Volume II: Computer Applications," New York: John Wiley & Sons, 1976.
[10] L. Sha, R. Jajkumar, and J. P. Lechoczky, "Priority Inheritance protocols: An Approach to real-time synchronization," *IEEE Transactions on Computers*, vol.39, no.9, Sept. 1990.

박 현 희
2000년 숭실대학교 컴퓨터학부 졸업(학사). 2003년 숭실대학교 대학원 컴퓨터학과 졸업(석사). 2009년 현재 숭실대학교 대학원 컴퓨터학과 졸업(박사). 관심분야는 실시간 시스템, 내장 시스템, 리눅스 커널

양 승 민
1978년 서울대학교 공과대학 전자공학과 학사. 1978년~1981년 삼성전자㈜ 연구원. 1983년 미국Univ. of South Florida 전산학 석사. 1986년 미국 Univ. of South Florida 전산학 박사. 1987년 미국 Univ. of South Florida 조교수. 1988년~1993년 미국 Univ. of Texas at Arlington 조교수. 1993년~현재 숭실대학교 컴퓨터학부 교수. 1996년~1998년 국회도서관 정보처리국장 파견