선점 임계 스케줄링에서의 실시간 동기화

김세화[0]
서울대학교 전기컴퓨터공학부
ksaehwa@redwood.snu.ac.kr

홍성수
서울대학교 전기컴퓨터공학부
sshong@redwood.snu.ac.kr

김태형
한양대학교 전자컴퓨터공학부
tkim@cse.hanyang.ac.kr

Real-Time Synchronization under Preemption Threshold Scheduling

Saehwa Kim
School of Electrical Engineering and Computer Science
Seoul National University

Seongsoo Hong
School of Electrical Engineering and Computer Science
Seoul National University

Tae-Hyung Kim
Department of Computer Science and Engineering
Hanyang University

요 약

객체 지향 설계 방법론은 현재 소프트웨어 개발에 널리 사용되고 있지만 내장형 실시간 시스템에는 적합한 스케줄링 기법이 존재하지 않기 때문에 제한적으로만 적용되어 왔다. 선점 임계 스케줄링(preemption threshold scheduling: PTS)은 비선점성을 최대한 이용하여 다중 태스킹의 부하를 줄이기 위한 제안되었다. PTS는 전통적인 스케줄링 기법을 대신하여 실시간 객체 지향 설계에 사용되기에 적합하나, 이를 위해서는 실시간 동기화의 문제가 해결되어야 한다. 본 논문에서는 PTS를 위한 실시간 동기화의 필수적인 기반을 제시한다. 구체적으로 PTS를 위한 기본 우선순위 계승 프로토콜과 우선순위 실링(ceiling) 프로토콜을 제시한다. 제시된 동기화 기법은 유효 우선순위의 개념을 사용하여 우선순위를 계승하도록 하며, 선점 임계 실링 대신 우선순위 실링을 사용한다.

1. 서론

객체 지향 설계 방법론은 현재 소프트웨어 개발에 널리 사용되고 있으나 내장형 실시간 시스템 분야에는 여전히 제한적으로만 적용되고 있다. 이는 객체 지향 설계 방법론에 전통적인 우선순위 기반 스케줄링 기법을 곧바로 통합시킬 수 없기 때문이다. 실시간 객체 지향 설계에서는 실시간 시스템을 동시에 수행 가능한 객체들의 집합으로 본다. 따라서 객체를 태스크로 매핑시켜 자동적으로 구현을 생성할 경우 일반적으로 많은 수의 태스크를 수반하게 되고[4], 이는 전통적인 우선순위 스케줄링 기법들을 사용할 경우 과도한 선점 부하를 야기한다는 문제점이 있다. 이러한 다중 태스킹의 부하를 줄이기 위하여, 비선점성을 최대한으로 이용하는 선점 임계 스케줄링(Preemption threshold scheduling: PTS)이 최근 제안되었다[1].

PTS 는 고정 우선순위 선점 스케줄링을 확장한 것으로, 각 태스크는 우선순위에 더하여 선점 임계값(preemption threshold)이라 불리는 스케줄링 속성을 가진다. 태스크의 선점 임계값은 태스크가 디스패치(dispatch)되어 수행을 끝낼 때까지의 우선순위이다. 만약 모든 태스크의 선점 임계값이 자신의 우선순위와 같다면, PTS 는 보통의 선점 스케줄링 모델이 된다. 반면 모든 태스크의 선점 임계값이 시스템의 가장 높은 우선순위와 같다면, PTS 는 비선점 스케줄링 모델이 된다. PTS 는 불필요한 선점을 줄임으로써 스케줄 가능성을 향상시킨다. 또한 선점 임계값의 적용을 통해 비선점적인 관계를 이루는 태스크들이 생기며 이러한 태스크들을 한 단위로 묶을 수 있으므로 구현 수준에서의 태스크(쓰레드 혹은 프로세스)의 숫자를 줄일 수 있다. 결과적으로 PTS 를 통해 실시간 시스템 설계의 확장성(scalability)을 높일 수 있다.

이러한 장점 때문에 우리는 PTS 를 실시간 객체 지향 시스템 설계에 적합한 스케줄링 기법으로 사용할 수 있다. 본 연구진에서는 PTS 를 적용하여 실시간 객체 지향 설계로부터 실행 가능한 구현을 자동으로 이끌어내는 연구[4, 5]를 수행하였다. 그러나 일반적인 실시간 시스템 설계 구현에 PTS 를 적용하기 위해서는 PTS 에서의 실시간 동기화 문제가 좀 더 포괄적으로 연구되어야 한다. 본 논문에서는 PTS 하에서의 실시간 동기화 문제를 푼다. 구체적으로 PTS 에서의 우선순위 계승 프로토콜인 기본 우선순위 프로토콜과 우선순위 실링(ceiling) 프로토콜을 제시한다.

논문의 나머지 부분은 다음과 같이 구성되어 있다. 2절에서는 태스크 모델을 설명한다. 3절과 4절에서는 각각 PTS에서의 기본 우선순위 계승 프로토콜[6]과 우선순위 실링(ceiling) 프로토콜[6]을 기술한다. 마지막으로 5절에서 결론을 내린다.

2. 태스크 모델

본 논문에서는 [2, 3, 6]에서와 거의 같은 태스크 모델을 사용한다. 구체적으로, 단일 프로세서 환경과 적절하게(properly) 중첩된 뮤택스[6]를 가정한다. 시스템에는 고정된 태스크 집합이 존재하며, 각 태스크는 고정 주기, 알려진 최악 수행 시간, 고정 우선순위와 선점 임계값을 가진다. 우선순위는 숫자가 더 클수록 더 높은 우선순위를 나타낸다. 표 1 에 본 논문에서 사용하는 기반 태스크 모델에 대한 표기법을 나타내었다.

표 1. 기반 태스크 모델의 표기법

표기법	설명
τ_i	태스크
T_i	τ_i의 주기
C_i	τ_i의 최악 수행시간
π_i	τ_i의 고정 우선순위
γ_i	τ_i의 preemption threshold
p_i	τ_i의 수행시 우선순위
M_i	뮤택스 (이진 세마포어)
$P(M_i)$, $V(M_i)$	뮤택스 M_i에 대한 개개의 잠금(lock), 풂(unlock) 시행
$\varphi(M_i)$	뮤택스 M_i의 우선순위 ceiling

PTS 하에서는, 각 태스크들은 정규 우선순위에 더하여 선점 임계값을 갖는다. 각 태스크의 선점 임계값은 불필요한 선점을 통제하는 실행시 우선순위로 사용되기 때문에, 그 태스크의 정규 우선순위보다 작지 않은 값을 할당하는 것이 의미가 있다. 태스크의 실제적인 유효 우선순위는 우선순위 계승과 PTS 에 의한 태스크 디스패칭(dispatching)에 의하여 실행시에 동적으로 변화되기 때문에, 이를 정확하게 정의할 필요가 있다. 개념적으로 태스크의 유효 우선순위는 커널 스케줄러가 디스패치될 태스크를 선택하는데 사용하는 우선순위이다. 이를 스케줄러의 조작적인 측면에서 다음과 같이 정의한다.

- τ_i의 유효 우선순위 p_i =
  (경우 1) 만약 주기내에서 릴리스되었으나, 디스패치되지 않았다면: π_i.
  (경우 2) 그렇지 않으면: $\max(\gamma_i, p_1, p_2, \ldots, p_j)$; 여기에서 $\tau_1, \tau_2, \ldots, \tau_j$ 는 τ_i에 의하여 블록된 태스크들

전통적인 우선순위 기반 선점 스케줄링에서, 태스크는 동기화에 의하여 블록킹을 겪게 된다. PTS 에서는 이에 더하여 새로운 유형의 블록킹이 존재하는데, 본 논문에서 이를 PTS 블록킹이라고 명명한다. 태스크 τ_i 가 π_i 보다 높은 선점 임계값을 가지고 있는 낮은 우선순위의 태스크에 의하여 블록되었을 때, τ_i가 PTS 블록킹에 있다고 말한다.

3. 우선순위 역전 문제의 방지

동기화 프리미티브(primitive)를 사용하는 실시간 시스템에서 발생할 수 있는 심각한 우선순위 역전 문제는, 우선순위 계승 프로토콜을 사용하지 않고서는 막을 수가 없다. 기본 우선순위 계승 프로토콜(Basic Priority Inheritance Protocol: BPI)은 우선순위 역전 문제를 결과적으로 초래하는 선점을 막는다. 이 절에서는 PTS 에서의 우선순위 역전 문제를 식별하고, PTS 에서의 BPI 를 정의하고, 그 특성들을 기술한다.

3.1. PTS 에서의 우선순위 역전 문제

전통적인 우선순위 기반 스케줄링에서 우선순위 역전 문제는 중간 우선순위의 태스크가 자신보다 높은 우선순위의 태스크를 블록하고 있는 자신보다 낮은 우선순위의 태스크를 선점할 때 발생한다. PTS 하에서는 태스크의 우선순위를 유효 우선순위로 대치시킴으로써 이러한 우선순위 역전 문제를 '유효 우선순위 역전 문제'로 변환시킬 수 있다. 이러한 변환은 PTS 에서 우선순위가 유효 우선순위로 대치되기 때문에, 명백히 자명하다. 즉 τ_L 에 의하여 블록된 태스크 τ_i 에 대하여 PTS 하에서의 우선순위 역전은 다음과 같은 두 가지 경우에 발생한다: (1) $\gamma_L < \pi_M < \pi_i$ 의 관계를 만족하는 중간우선순위의 태스크 τ_M이 τ_L을 선점할 때와 (2) $\pi_i < \pi_H \le \gamma_i$ 의 관계를 만족하는 더 높은 우선순위의 태스크 τ_H 가 τ_L 을 선점할 때이다.

3.2. PTS 에서의 BPI

PTS 에서의 BPI 프로토콜은 다음과 같이 정의한다.

Protocol 1. PTS 에서의 BPI.

- *유효 우선순위의 상속.*
  태스크 τ_H가 $\pi_H > \gamma_L$인 τ_L에 블록 당할 때, 태스크 τ_L은 태스크 τ_H의 p_H를 계승한다
- *유효 우선순위의 회복.*
  태스크 τ_L이 임계 구역을 빠져 나올 때, 태스크 τ_L은 임계 구역에 들어가지 전에 가졌던 유효 우선순위 p_L을 회복한다.

이러한 BPI 의 정의가 우선순위 역전 문제를 발생시키는 선점을 막는다는 것을 쉽게 볼 수 있다. 그림 1 에 Protocol 1 을 따르는 예를 나타내었다. 태스크 τ_M 과 τ_H 모두 τ_i 를 선점할 수 없기 때문에, 우선순위 역전 문제가 방지된다.

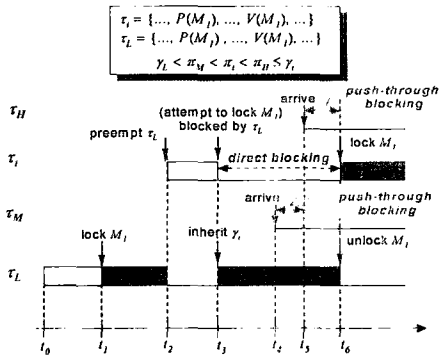

그림 1. PTS에서의 BPI를 통한 우선순위 역전의 방지

3.3. PTS 에서의 BPI 의 특성

PTS 에서의 BPI 는 [6]에서의 원래의 BPI 와 유사한 특성을 가지고 있다. 원래의 프로토콜에서, 태스크는 (1) 직접(direct) 블록킹, (2) 상속(push-through) 블록킹, (3) 추이(transitive) 블록킹의 세 가지 블록킹 유형 중 하나에 의하여 블록된다. 우리의 프로토콜에서는, 태스크는 이에 더하여 PTS 블록킹에 의하여 블록된다. 직접 블록킹은 보다 높은 우선순위의 태스크가 보다 낮은 우선순위의 태스크에 의하여 잠겨 있는 뮤택스를 잠그려는 시도를 할 때 발생한다. 이는 비선점적인 공유 자원의 일관성을 보전하기 위한 것이다. PTS 에서의 상속 블록킹은 중간 우선순위의 태스크가 자신보다 높은 우선순위의 태스크를 블록하고 있는 자신보다 낮은 우선순위의 태스크를 선점하려는 시도를 할 때 발생한다. 그림 1 에서 태스크 τ_M 과 τ_H 은 각각 (t_4, t_5)와 (t_5, t_6)의 구간에서 상속 블록킹에 의하여 블록되었다.

추이 블록킹은 태스크 τ_H가 태스크 τ_M에 의하여 블록되고, 차례로 τ_M 이 또 다른 태스크 τ_H에 의하여 블록되는 경우를 말한다. 또한 태스크는 연쇄 블록킹을 당할 수 있다. 연쇄 블록킹(또는 블록킹의 연쇄)은 태스크가 임계 구역을 들어갈 때 반복적으로 블록되는 경우를 말한다. 연쇄 블록킹은 블록킹의 유형이 아니라, 태스크가 한 번 이상 블록되는 경우를 가르킨다는 데 주의해야 한다.

한편 BPI 는 그 자체로서는 교착상태를 막을 수 없다. 교착상태는 여러 개의 태스크가 순환적인 방식으로 중첩된 뮤택스를 접근하려는 시도를 할 때 발생한다.

4. 교착상태, 추이 블록킹 및 연쇄 블록킹의 방지

BPI 는 상속 블록킹을 통하여 우선순위 역전문제를 방지하지만, 3.3 절에서 논한 바와 같이 추이 블록킹과 연쇄 블록킹으로 인한 긴 블록킹 지연과 교착상태를 발생 시킬 수 있다. 우선순위 실링(ceiling) 프로토콜(Priority Ceiling Protocol: PCP)은 이러한 문제들을 해결하기 위하여 제안되었다. [6]. 이 절에서는 PTS 에서의 PCP 를 정의하고, 이에 대한 이론적 근거를 기술한다.

4.4. PTS 에서의 PCP

PCP 알고리즘의 기반 아이디어는 각 태스크에 대하여, 그 태스크 이상의 우선순위를 가진 태스크들이 접근할 수 있는 모든 뮤택스들이 잠겨 있지 않을 경우에만 뮤택스를 잠그는 것을 허용하게 하는 것이다. 만약 그렇지 않다면, PCP 는

그러한 뮤택스들이 풀릴 때 까지 그 태스크를 기다리게 한다. 이러한 아이디어를 실현하기 위하여, Rajkumar 등은 우선순위 실링의 개념을 제안하고, 이를 각 뮤택스와 연관시켰다[6].

본 논문에서는 [7]에서의 정의 방식과 유사한 방식으로, 오프라인 실링 프로토콜과 온라인 잠금(locking) 프로토콜의 조합으로서 PTS 에서의 PCP 를 정의한다.

Protocol 2. PTS 에서의 PCP.

- *오프라인 실링 프로토콜*

  CP. 각 뮤택스 M_i 는 우선순위 실링값을 다음과 같이 할당받는다: $\varphi(M_i) = max\{\pi_i | \tau_i \in \Psi_i\}$; 여기에서 Ψ_i는 뮤택스 M_i 를 사용할 수 있는 태스크들의 집합.

- *온라인 잠금 (locking) 프로토콜*

  LP1. 시스템 스케줄링 속성으로서 '시스템 실링'이 시스템에 잠겨 있는 모든 (현재 수행중인 태스크에 의해 잠긴 것을 제외한) 뮤택스들의 우선순위 실링 값의 최대값으로 동적으로 설정된다.

  LP2. 태스크 τ_i 가 임계 구역에 들어가기 위해서는 γ_i가 시스템 실링 값보다 더 커야 한다.

  LP3. *유효 우선순위의 계승*

  태스크 τ_H가 $\pi_H > \gamma_L$인 τ_L에 블록 당할 때, 태스크 τ_L은 태스크 τ_H의 p_H를 계승한다.

  LP4. *유효 우선순위의 회복*

  태스크 τ_L이 임계 구역을 빠져 나올 때, 태스크 τ_L은 임계 구역에 들어가지 전에 가졌던 유효 우선순위 p_L을 회복한다.

이 정의에서 (**LP3** 와 **LP4** 에 명세되어 있듯이) 유효 우선순위가 우선순위 계승에 사용되고, (**CP** 에 명세되어 있듯이) 정규 우선순위가 뮤택스의 실링값에 대하여 사용되었다. 우선순위 계승에 유효 우선순위를 사용하는 것은 BPI 의 채용으로서 당연하다. 그러나 뮤택스의 실링값을 정함에 있어서 선점 임계값을 사용하지 않고, 정규 우선순위를 사용한 것은 직관적이지 않다. 이에 대한 이유는 다음 절에서 설명한다. 이제 Protocol 2 의 PCP 정의가 교착상태와 추이 및 연쇄 블록킹을 방지함을 보인다.

잠금 규칙. 태스크 τ_c 가 뮤택스를 잡기 위해서는 다음의 조건이 만족되어야 한다: $\pi_H \geq \gamma_c$ 인 태스크 τ_H 가 사용할 수 있는 모든 뮤택스들이 잠겨 있지 않다.

정리 1. 잠금 규칙을 따르면 교착상태와 천이 및 연쇄 블록킹을 방지하는 것이 보장된다.
증명. 지면 관계상 생략. □

정리 2. Protocol 2 의 PTS 에서의 PCP 는 잠금 규칙을 따른다.
증명. 지면 관계상 생략. □

4.5. 우선순위 실링 대 선점 임계 실링

정리 2 에 따라 PTS 를 위한 PTS 는 우선순위 실링을 통하여 정의될 수 있으나, 뒤에 나오는 정리 3 과 같이 우선순위 실링 대신 선점 임계 실링을 사용하여서도 정의될 수 있다. 이 절에서는 선점 임계 실링을 사용하지 않고, 우선순위 실링을 사용하여 PCP 를 정의한 이유에 대하여 설명한다.

정리 3. 선점 임계 실링을 사용하는 PCP 는 잠금 규칙을 따른다.
증명. 지면 관계상 생략. □

정리 4. 임의의 태스크 τ_i 에 대하여 τ_i 가 임계 구역에 들어가고자 할 때 잠겨 있을 경우 블록킹을 발생시키는 뮤택스의 집합을 ζ_i 라고 하자. 우선순위 실링을 사용하는 PCP (PCP-Type 1)에서의 ζ_i는 우선순위 임계 실링을 사용하는 PCP (PCP-Type 2)의 ζ_i의 부분집합이다.
증명. PCP-Type 1 에서의 ζ_i 를 ζ^{Type1}_i 라 하고, PCP-Type 2 에서의 ζ_i를 ζ^{Type2}_i라 하자. ζ^{Type1}_i는 $\pi_H \geq \gamma_i$인 태스크 τ_H에 의하여 잠길 수 있는 뮤택스들의 집합이다. ζ^{Type2}_i는 $\gamma_H \geq \gamma_i$인 태스크 τ_H 에 의하여 잠길 수 있는 뮤택스들의 집합이다. 임의의 태스크 τ_H 에 대하여 $\gamma_H \geq \pi_H$ 인 관계가 성립하므로, ζ^{Type1}_i는 ζ^{Type2}_i의 부분집합이다. □

우선순위 임계 실링을 사용하는 PCP 는 우선순위 실링을 사용하는 PCP 에는 없는 불필요한 블록킹이 발생한다. 구체적으로 이는 태스크 τ_i가 임계 구역에 들어가려고 할 때, $\pi_H < \gamma_i$ 이면서 $\gamma_H \geq \gamma_i$ 인 태스크 τ_H에 의하여 잠길 수 있는 뮤택스들이 잠겨 있는 경우이다. 이러한 불필요한 블록킹은 PTS 에서의 최대 우선순위 임계 할당 정책에 의하여 태스크들의 우선순위 임계값이 가능한 한 높은 값을 가지게 되어 매우 빈번하게 발생할 수 있다. 각 블록킹은 태스크의 문맥 전환 횟수를 두 번 더 발생시키며, 블록당한 높은 우선순위의 태스크의 응답 시간을 블록킹 시간만큼 길어지게 한다. 따라서 우선순위 임계 실링 대신 우선순위 실링을 사용하는 것이 바람직하다.

5. 결론

본 논문에서는 선점 우선순위 스케줄링(preemption threshold scheduling: PTS)에서의 실시간 동기화 문제를 다루었다. 구체적으로 PTS 에서의 기본 우선순위 계승 프로토콜(BPI)과 우선순위 실링(ceiling) 프로토콜(PCP)을 제시하였다. 또한 제안된 프로토콜을 통해 PTS 에서 선점 불가능한 공유 자원을 사용할 때의 우선순위 역전 문제 및 교착상태, 긴 블록킹 지연시간을 막을 수 있음을 보였다.

PTS 하에서의 각 태스크는 정규 우선순위에 더하여 선점 임계값을 부가적으로 갖기 때문에, 우선순위 계승을 적용할 때와 실링값을 정함에 있어서 어떤 스케줄링 속성을 사용할 지가 직관적이지 않다. 이를 명시적으로 해결하기 위하여 본 논문에서는 유효 우선순위의 개념을 도입하였다. 제안된 프로토콜은 우선순위 계승에 있어서는 유효 우선순위를 사용하고, 뮤택스들의 실링값을 결정하는데 있어서는 선점 임계값이 아닌 정규 우선순위를 사용한다.

참고 문헌

1. W. Lamie, Preemption-Threshold, White Paper, Express Logic Inc., Available at http://www.threadx.com/wppreemption.html
2. M. Saksena and Y. Wang. Scalable real-time system design using preemption thresholds, In Proceedings of IEEE Real-Time Systems Symposium, pp. 25 -34, 2000.
3. Y. Wang and M. Saksena. Scheduling fixed priority tasks with preemption threshold. In Proceedings of IEEE Real-Time Computing Systems and Applications Symposium, pp. 328-335, 1999.
4. S. Kim, S. Cho, and S. Hong. Schedulability-aware mapping of real-time object-oriented models to multi-threaded implementations. In Proceedings of International Conference on Real-Time Computing Systems and Applications, pp. 7-14, 2000.
5. S. Kim, S. Hong, and N. Chang, Scenario-based implementation architecture for real-time object-oriented models, In Proceedings of IEEE International Workshop on Object-oriented Real-time Dependable Systems, 2002.
6. R. Rajkumar, L. Sha and J. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. In IEEE Transactions on Software Engineering, vol. 39, pp. 1175-1185, 1990.
7. M. Chen and K. Lin. Dynamic priority ceilings: A concurrencly control protocol for real-time systems. In Real-Time Systems Journal, vol. 2, pp. 325-346, 1990.