실시간 서버 시스템에서 우선 순위 반전현상을 감소하기 위한 모델

최 대 수† · 임 종 규†† · 구 용 완†††

요 약

실시간 시스템에서 다양한 실시간 활동들의 엄격한 시간 요구를 만족시키기 위해 종종 시스템 실행시간 활동을 조정하는 특별한 방법을 요구한다. 무제한 블록킹은 높은 우선 순위 활동이 낮은 우선 순위 활동을 선점할 수 없을 때 발생한다. 이러한 상태를 우선 순위 반전 현상이라고 한다. 우선 순위 반전은 실시간 시스템에서 쓰레드의 마감시한내 수행 보장을 방해하는 문제들 중 하나이다. 커널에서 우선 순위 반전 문제를 제거하고 동시에 쓰레드에 대한 최악 블록 시간을 제한하는 것은 어렵다. 쓰레드란 데이터와 스택을 접근할 수 있는 실행코드이다.

본 연구에서는 우선 순위 반전 문제를 감소하기 위한 새로운 서버 모델을 제안한다. 본 모델은 서버에서 우선 순위 반전 기간을 최소화하는 실시간 서버 모델이다. 제안한 서버 모델은 우선 순위 반전 기간을 최소화 할 뿐만 아니라 보다 높은 우선 순위 쓰레드들의 마감시한 실패율을 감소하는 향상된 서버 구조를 만들기 위한 기틀을 제공한다.

A Model for Reducing Priority Inversion in Real Time Server System

Dae-Soo Choi† · Jong-Kyu Im†† · Yong-Wan Koo†††

ABSTRACT

Satisfying the rigid timing requirements of various real-time activities in real-time systems often requires some special methods to tune the systems run-time behaviors. Unbounded blocking can be caused when a high priority activity cannot preempt a low priority activity. In such situation, it is said that a priority inversion has occurred. The priority inversion is one of the problems which may prevent threads from meeting the deadlines in the real-time systems. It is difficult to remove such priority inversion problems in the kernel at the same time to bound the worst case blocking time for the threads. A thread is a piece of executable code which has access to data and stack.

In this paper, a new real-time server model, which minimizes the duration of priority inversion, is proposed to reduce the priority inversion problem. The proposed server model provides a framework for building a better server structure, which can not only minimize the duration of the priority inversion, but also reduce the deadline miss ratio of higher priority threads.

1. 서 론

실시간 시스템(real-time system)이란 외부의 비동기적인 사건에 대하여 예측 가능한 시간 내에 응답을 보장할 수 있는 시스템이다. 이러한 엄격한 시간 제약(timing constraints)을 만족하기 위해서는 실시간 시스템이 예측 가능하고(predictable) 신뢰성 있는(reliable) 환경을 제공해야만 한다. 실시간 작업 스케줄링에서 선점 스케줄링 기법을 사용하는 경우에는 높은 우선순

† 준 회 원 : 수원대학교 대학원 전자계산학과
†† 준 회 원 : 수원대학교 자연과학연구소 객원연구원
††† 종신회원 : 수원대학교 전자계산학과 교수 및 전자계산소장
논문접수 : 1999년 1월 29일, 심사완료 : 1999년 9월 27일

위 작업이 낮은 우선순위 작업을 선점할 수 있도록 하여야 할 것이다. 그러나 실시간 작업들이 공유자원을 사용할 경우 이들의 동기순서화를 위하여 사용되는 도구들에 의해 높은 우선 순위 작업이 낮은 우선 순위 작업에 의해 블록 되는 경우가 발생할 수 있다. 이러한 현상을 우선 순위 반전(priority inversion)이라 한다[1]. 우선순위 반전의 기간이 비결정적(nondeterministic)으로 길어지는 무제한 우선 순위 반전(unbounded priority inversion)은 실시간 시스템의 스케줄 가능성(schedulability) 및 예측 가능성(predictability)을 떨어뜨린다. 따라서, 이러한 우선 순위 반전기간을 최소화하여 실시간 시스템의 성능을 높이는 연구가 필요하다.

최근까지 연구된 실시간 시스템을 위한 서버 모델들은 예측 가능한 서비스 시간을 보장하기 위한 기반을 제공한다. 서버들의 구조와 우선 순위 관리는 서버 모델에서 중요하다. 서버 모델은 서버에서 쓰레드들의 수와 요구들을 처리하기 위한 쓰레드 생성시간에 따라 단일 쓰레드 서버 모델(single thread server model), 작업자 모델(worker model) , 동적 서버 모델(dynamic server model)로 구분된다[2][3][4]. 또한 최근에 연구된 연관 우선 순위 작업자 모델도 포함될 수 있다[5]. 이러한 서버 모델들이 갖는 문제점 중 하나는 실시간 처리에 있어서 우선 순위 반전 현상인데, 이를 해소하기 위한 연구가 행해졌지만 부분적인 우선 순위 반전 허용의 문제를 여전히 내포하고 있다[5][6].

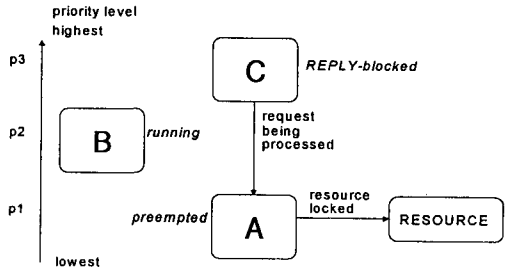
본 연구에서 제안한 서버 모델은 각 작업자 쓰레드에 우선 순위 그룹을 새로이 부여하여 기존 서버 모델에서 발생했던 우선 순위 반전 현상을 줄여 보다 많은 요구들이 마감시한을 만족하고자 하며, 특히 높은 우선 순위 요구들이 마감시한내에 작업을 완료함으로서 실시간성을 높이고자 한다. 높은 우선 순위 요구를 담당하는 쓰레드의 우선 순위 범위는 좁게 할당하고 낮은 우선 순위 요구를 담당하는 쓰레드의 우선 순위 범위는 상대적으로 넓게 할당하여 요구를 처리한다. 이러한 우선 순위 할당 기법은 높은 우선 순위 요구에 보다 높은 비중을 두어 가능한 빨리 요구가 처리되도록 하고자 한 것이다.

본 논문은 2장 관련연구로서 우선 순위 반전 현상과 기존에 제안된 여러 서버 모델들에 관하여 기술한다. 3장에서는 제안한 서버 모델과 관련 알고리즘에 관하여 기술한다. 4장은 제안한 서버 모델을 기존 서버 모델과 모의 실험을 통하여 서로 비교하였으며 5장은 결론 및 향후 연구 방향에 대해 기술함으로 본 논문을 마감하고자 한다.

2. 관련연구

2.1 우선 순위 반전

우선 순위 반전 현상은 프로세스의 우선 순위 결합과 프로세스들 사이의 자원 공유에서 발생한다. 예를 들어, (그림 1)에서 자원은 3개의 프로세스들 중에서 가장 낮은 우선 순위(p1)를 갖는 프로세스 A에 의해 사용 중일 수 있다. 이때, 가장 높은 우선 순위(p3)를 갖는 프로세스 C는 인터럽트를 얻어 즉각적인 자원에 대한 접근을 요구하기 위해 A에게 요구를 보낸 후 응답은 블록된다. 그후, A는 스케줄을 얻어서 수행을 시작한다. A가 수행하던중 C로부터의 요구를 미처 처리 완료하지 못한 상황에서 중간 우선 순위(p2)를 갖는 프로세스 B가 인터럽트를 받아 A를 선점하고 수행을 시작할 수 있다. 이것은 B가 A보다 높은 우선 순위를 갖기 때문이다. 결과적으로 C는 A가 B에 대한 요구를 끝낼 때까지 기다려야만 한다. 사실상 높은 우선 순위 프로세스 C는 낮은 우선 순위 B 또는 A에 의해 블록된다. 이러한 상황은 우선 순위 반전의 한가지 예이다. 이러한 경우 프로세스 B가 얼마나 수행시간이 소모되는지 알 수 없기 때문에 임계 프로세스(C)가 작업을 완료하는데 필요한 시간을 예측할 수 없게 된다. 이것은 어플리케이션의 시간 요구사항을 만족하기 어렵게 만든다. 이러한 문제를 피하기 위하여 실시간 운영체제(real time operating systems)는 일반적으로 우선 순위 상속(priority inheritance)의 개념을 이용한다.



(그림 1) 우선 순위 반전

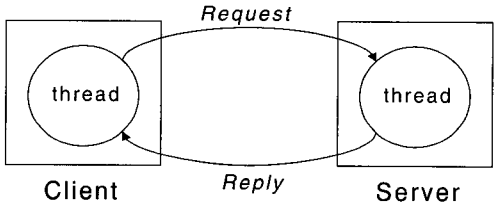
2.2 실시간 서버 모델

각 서버들의 구조와 우선 순위 관리는 실시간 서버

모델에서 중요한 포인트이다. 본 분류는 서버에서 쓰레드의 수와 요구를 처리하기 위한 쓰레드를 만드는 시간, 쓰레드 우선순위 할당방법에 의해 이뤄진다 [3][4][6][7][8]. 쓰레드란 데이터와 스택을 접근할 수 있는 실행코드이다.

2.1.1 단일 쓰레드 서버 모델
(single thread server model)

모델의 첫 번째 분류는 단일 쓰레드 서버 모델이다. 본 모델은 (그림 2)에서처럼 하나의 쓰레드가 들어오는 모든 요구들을 처리한다. 새로운 요구는 서버에서 수행중인 이전요구의 수행을 선점할 수 없다. 따라서, 새로운 요구는 이전요구가 종료될 때까지 블록된다. 새로운 요구는 송신자 쓰레드(sender thread)와 동일한 우선 순위를 할당받으며, 서버에서 쓰레드의 우선 순위는 요구를 받을 때 우선 순위에 기초하여 동적으로 변화한다. 만약 쓰레드가 새로운 요구를 기다릴 필요가 없다면 새로운 요구는 우선 순위화된 큐에 삽입되고, 요구의 우선 순위가 쓰레드의 우선 순위보다 높다면 쓰레드의 우선 순위는 요구의 우선 순위 수준까지 올라간다. 본 모델은 구현하기 쉽지만 요구의 수행시간이 길어질 경우, 우선 순위 반전기간은 길어진다.



(그림 2) 단일 쓰레드 서버 모델

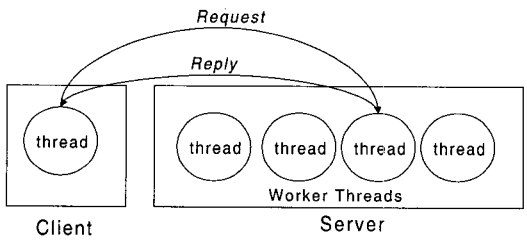
2.1.2 작업자 모델(worker model)

두 번째 분류는 (그림 3)에서처럼 여러 쓰레드들이 요구들을 협력하여 처리하는 모델이다. 이러한 모델을 작업자 모델이라 부르며, 요구들을 처리하는 쓰레드들은 작업자 쓰레드(worker thread)라 한다. 본 모델은 두 가지 유형으로 다시 나눌 수 있다.

첫 번째, 정적 우선 순위 작업자 모델(static priority worker model)은 서버의 각 쓰레드에게 다른 우선 순위를 부여한다. 새로운 요구가 서버에 도착할 때, 서버는 요구의 우선 순위와 같은 우선 순위를 갖는 쓰레드를 선택한다. 만약 해당 쓰레드가 수행중이면 요구는 블록되어지며, 쓰레드가 이전 요구를 완료할 때까지 우선 순위 큐에 입력된다. 본 모델은 단일 쓰레드 서버 모델보다 좋은 선점 가능성을 제공한다.

본 모델은 정해진 개수의 서버 쓰레드로 작업자 그룹을 구성하고 각 쓰레드에 각기 다른 한정된 우선 순위 요구만을 서비스하도록 함으로서 낮은 우선 순위 요구들에 의한 서버들의 독점을 방지하는 방식이다. 그러나 이는 우선 순위 단계를 모두 파악해야 하며 우선 순위 단계가 많아질 경우 상대적으로 작업자 쓰레드의 개수가 많아져야 하는 오버헤드가 있다.

둘째, 동적 우선 순위 작업자 모델(dynamic priority worker model)은 요구를 받은 쓰레드는 요구의 우선 순위를 상속받는다. 만약 모든 쓰레드들이 요구들을 처리하고 있다면 새로운 요구는 쓰레드 중에서 하나가 완료하기를 기다려야만 한다. 쓰레드는 새로운 요구를 처리하도록 선택된다. 그리고 요구는 선택된 쓰레드가 완료하기를 기다리기 위해 우선 순위 큐에 삽입된다. 만약 요구의 우선 순위가 선택된 쓰레드의 현재 우선 순위보다 높다면 쓰레드의 우선 순위는 새로운 요구의 우선 순위로 올라간다. 본 모델의 단점은 모든 작업자 쓰레드들이 낮은 우선 순위 요구들에 의해 사용 중이라면 우선 순위 반전이 발생한다.

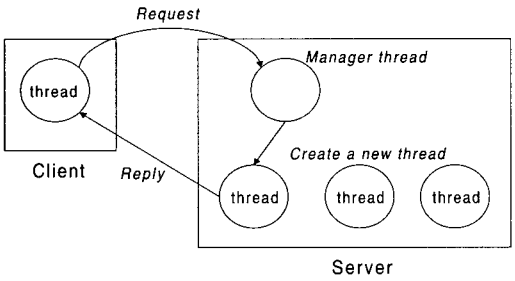


(그림 3) 작업자 모델

2.1.3 동적 서버 모델(dynamic server model)

본 모델은 (그림 4)에서처럼 새로운 요구가 도착될 때마다 새로운 쓰레드가 생성되어진다. 새롭게 생성된 쓰레드는 요구의 우선 순위를 상속받는다. 본 모델은 가장 좋은 선점 가능성을 제공한다. 이것은 요구가 다른 모델과 달리 서버에서 쓰레드들의 완료를 기다릴 필요가 없기 때문이다. 반면, 커널 수준에서 지원되는 쓰레드로 구현되어진다면 새로운 쓰레드를 만드는 비용은 증가될 것이다. 보통의 IPC(InterProcess Communication)는 요구가 들어왔을 때 새로운 쓰레드를 만들도록 지원하

지 않기 때문에, 관리자 쓰레드(manager thread)가 쓰레드들을 만들도록 요구한다. 이는 여분의 문맥 교환(context switching)이 필요하며 가장 심각한 문제는 쓰레드가 동적으로 생성되어진다는 점으로서 서버에서 최대 자원 사용량을 측정하기가 어렵다. 따라서, 실시간 서버에는 적합하지 않다.

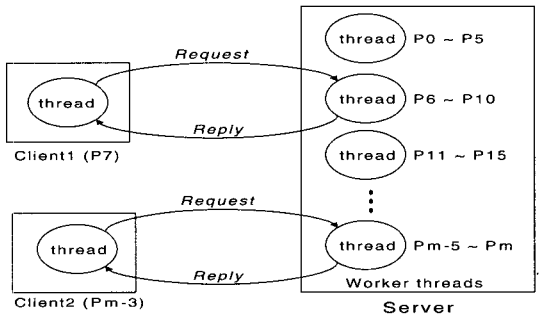


(그림 4) 동적 서버 모델

2.1.4 연관 우선 순위 작업자 모델
(associative priority worker model)

선점가능성과 오버헤드간의 상충관계(tradeoff)를 최적화 하고자 정적 우선 순위 작업자 모델과 동적 우선 순위 작업자 모델의 특성을 결합한 모델이다[5]. 본 모델은 각 작업자 쓰레드가 처리해야 할 우선 순위 단계 범위를 갖으며, 새로운 요구가 들어왔을 때 요구의 우선 순위가 속한 범위의 작업자 쓰레드가 요구를 처리하고자 하는 방법이다. 이때 해당 작업자 쓰레드가 다른 요구를 처리하고 있다면, 새로운 요구는 대기큐에 대기하면서 해당 작업자 쓰레드의 우선순위가 자신보다 낮다면 자신의 우선 순위를 작업자 쓰레드에게 상속하게 된다. 이러한 방법으로 낮은 우선 순위의 요구들이 모든 쓰레드를 선점하는 경우를 방지할 수 있으며, 제일 높은 우선 순위의 쓰레드가 제일 낮은 우선 순위 쓰레드의 처리를 기다리게 되는 경우도 없다. 결과적으로 새로운 요구에게 보다 높은 선점가능성을 제공하고 우선 순위 반전시간을 감소하여 요구에 대한 응답시간을 줄일 수 있다.

상기 모델의 단점은 클라이언트 요구들의 우선 순위가 하나의 서버 작업자 쓰레드 우선 순위 범위에 집중적으로 요구될 경우 그 범위를 갖는 쓰레드에 요청된 요구들은 계속적으로 사상(mapping)되어 큐길이를 증가시킨다. 반면 다른 우선 순위 범위를 갖는 쓰레드는 요구가 없어서 계속적으로 유휴상태로 될 경우가 발생한다. 본 모델의 문제점은 기다리고 있는 우선순위 범위의 쓰레드가 suspend 되었을 때, 자기보다 높은 우선순위를 갖는 요청들이 기다리고 있음에도 불구하고 쓰레드를 할당받지 못했기 때문에 낮은 우선순위 범위에 있는 쓰레드가 수행된다는 점이다.



(그림 5) 연관 우선 순위 작업자 모델

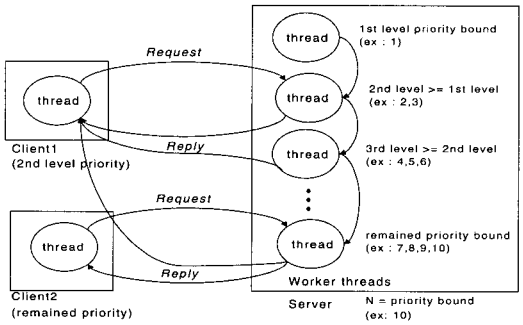
3. 서버 모델 설계

3.1 다단계 서버 모델

본 연구에서는 (그림 6)에서 보인 것처럼 기존의 서버 모델에서 발생했던 우선 순위 반전시간을 감소시켜 보다 좋은 실시간성을 갖고자 하며, 연관 우선 순위 작업자 모델의 문제점인 하나의 우선 순위 범위에 요구들이 많아질 경우 유휴 쓰레드와 바쁜 쓰레드가 동시에 존재하는 것을 감소시켜 보다 많은 요구들이 마감시한을 만족하도록 모델을 설계하였다.

제안한 모델은 각 쓰레드마다 우선 순위 범위를 다르게 부여하여 처리하게 된다. 즉, 높은 우선 순위 요구를 담당하는 쓰레드의 우선 순위 범위는 좁게 할당하고 낮은 우선 순위 요구를 담당하는 쓰레드의 우선 순위 범위는 상대적으로 넓게 할당하여 요구를 처리한다. 이러한 우선 순위 할당 기법은 높은 우선 순위 요구에 보다 높은 비중을 두어 가능한 빨리 요구가 처리되도록 하고자 한 것이다. 반면 낮은 우선 순위 요구는 실시간 처리의 비중이 적기 때문에 하나의 쓰레드에서 많은 범위의 요구들을 처리하게 된다.

본 기법은 높은 우선 순위를 담당하는 쓰레드에 요구가 집중된다면 연관 우선 순위 작업자 모델에서와 같은 바쁜 쓰레드에는 큐잉이 계속되고 유휴 쓰레드는 계속유휴상태에 있는 문제가 발생될 것이다. 따라서, 이미 작업자 그룹내 해당하는 쓰레드가 사용중인 경우,



(그림 6) 다단계 서버 모델

자신보다 우선 순위가 하위레벨에 있는 작업자 쓰레드가 유휴한 상태인가를 판단하여 유휴한 경우 그곳에 요구를 할당하게 한다. 이때 작업자 그룹의 쓰레드 우선 순위는 우선 순위 상속 규약에 따라서 요구의 우선 순위를 일시적으로 상속받게 된다. 자신의 작업자 그룹 쓰레드가 아닌 하위 쓰레드에서 처리된 요구 완료 시 작업자 그룹의 쓰레드는 원래의 우선 순위로 환원된다. 이러한 처리는 연관 우선 순위 작업자 모델의 취약점을 해결하여 스케줄 가능성 및 마감시한을 더욱 만족할 수 있다.

3.2 알고리즘

서버에 대한 요구들은 스케줄러에 의해서 해당 쓰레드에 우선적으로 할당을 의뢰한다. 해당 쓰레드가 다른 요구에 의해 처리중이라면 하위레벨의 쓰레드를 조사하여 유휴한 쓰레드에 할당하게 된다. 만약, 유휴한 쓰레드가 없다면 마지막 쓰레드의 우선 순위화된 큐에 삽입되어 처리된다. 구체적인 알고리즘은 다음과 같다.

```
Schedule_Request(Request_thread)
{
  Switch(Request_thread.priority) {
    case first_priority_bound : if(first_level_thread .EQ. idle) {
                  assign Request_thread to first_level_thread;
                  break; }
    case second_priority_bound : if(second_level_thread .EQ. idle) {
                  assign Request_thread to second_level_thread;
                  break; }
    case third_priority_bound : if(third_level_thread .EQ. idle) {
                  assign Request_thread to third_level_thread;
                  break; }
                  :
    default : assign Request_thread to last_level_thread;
  }
  return OK;
}
```

(그림 7) 다단계 알고리즘

4. 성능 평가

4.1 모의 실험 모델

실시간 환경을 지원하는 서버 모델은 먼저 다른 요구에 비해 실시간 요구의 우선적 처리를 보장해 주어야 하며, 우선 순위가 낮은 요구와 비교해 보다 긴급한 높은 우선 순위 요구에 대해 우선적 처리를 보장하여야 한다.

실험 과정에서 앞서 소개한 모든 서버 모델들과 새롭게 제안한 서버 모델을 비교/평가하여야 하겠으나 단일 쓰레드 서버 모델, 정적 우선 순위 작업자 모델과 동적 서버 모델은 각각 한정된 경우이기 때문에 다른 모델과 비교/평가한다는 것은 별 의미가 없다. 따라서, 본 모의 실험에서는 동적 우선 순위 작업자 모델과 연관 우선 순위 작업자 모델, 제안한 모델만을 다루었다.

4.2 입력 매개 변수

실험에서 사용된 프로그램은 모의 실험 모델링 도구인 ARENA로서 PC상에서 구현되었다[9]. 본 도구에서 전체 요구 수는 150개로 가정하여 해당 쓰레드를 스케줄링 하였으며, 각 작업의 실행시간은 평균 실행시간 10에 분산이 5인 정규분포를 따른다[1]. 요구들은 4.258의 지수분포에 따라 생성되어진다[10]. 각 요구는 가장 높은 순위 1에서부터 가장 낮은 우선 순위 10까지의 우선 순위를 가진다. 또한 작업자 쓰레드의 개수를 2에서 5개까지 하나씩 늘려가면서 모의 실험하였다. 우선 순위 범위가 10이기 때문에 쓰레드의 개수가 6개 이상에서는 모델 구분의 의미가 적어지므로 모의 실험에서 제외하였으며, 또한 쓰레드 개수가 1개일 경우에는 단일 쓰레드 서버 모델과 같은 성능을 가지게 되므로 제외하였다.

연관 우선 순위 작업자 모델은 쓰레드가 2개일 경우 우선 순위 범위가 (1~5), (6~10)로 부여하고 쓰레드가 3개일 경우 (1~3), (4~6), (7~10)으로 부여하고 쓰레드가 4개일 경우 (1~2), (3~4), (5~7), (8~10)로 부여하고 쓰레드가 5개일 경우 (1~2), (3~4), (5~6), (7~8), (9~10)로 범위를 부여하여 실험하였다. 제안한 모델의 우선 순위 범위는 쓰레드가 2개일 경우 (1~3), (4~10)으로 부여하고 쓰레드가 3개일 경우 (1), (2~4), (5~10)로 부여하고 쓰레드가 4개일 경우 (1), (2~3), (4~6), (7~10)으로 부여하고 쓰레드가 5개일 경우 (1), (2),

(3~4), (5~7), (8~10)로 범위를 부여하여 실험하였다. 동적 우선 순위 작업자 모델은 우선 순위 범위의 구분 없이 요구가 할당되므로 우선 순위 범위가 필요 없다. 또한 각 쓰레드마다 큐의 용량을 무한대로 가정한다.

각 우선 순위별 쓰레드 발생 분포는 <표 1>과 같이 3가지 경우로 변화를 주었다. [경우 1]에서는 모든 요구들이 동일한 발생 확률로 생성될 때 쓰레드의 수를 2~5개로 증가시켜가면서 실험하였으며 [경우 2]와 [경우 3]은 쓰레드의 수를 4개로 한정해서 우선 순위에 따라 다른 발생 분포를 가지고 모델을 실험하였다.

〈표 1〉 우선 순위별 타스크 발생률

case \ priority	1	2	3	4	5	6	7	8	9	10
경우 1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
경우 2	0.13	0.12	0.11	0.1	0.1	0.1	0.1	0.09	0.08	0.07
경우 3	0.15	0.14	0.13	0.12	0.11	0.09	0.08	0.07	0.06	0.05

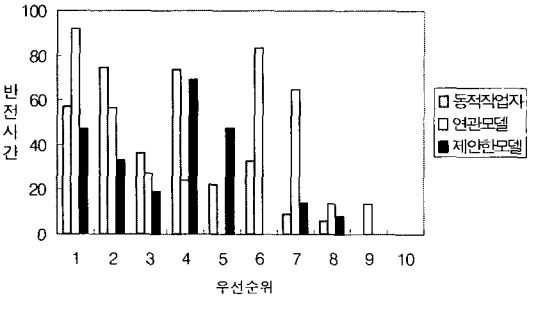
4.3 평 가

본 모의 실험은 <표 1>과 같이 3가지 경우로 가정하여 우선 순위 반전시간을 우선 순위별로 구분하여 평가하였다.

4.3.1 경우 1

(1) 쓰레드가 2개일 경우

모든 우선 순위 요구들이 균일하게 생성되고 작업자 쓰레드의 수를 2개로 가정하였을 때 결과이다. (그림 8)에서 우선 순위 반전시간을 보면 우선 순위 1~3의 경우 제안한 모델이 가장 적은 반전시간을 보이는데, 이는 높은 우선 순위 요구의 보다 신속한 처리가 되었음을 알 수 있다.



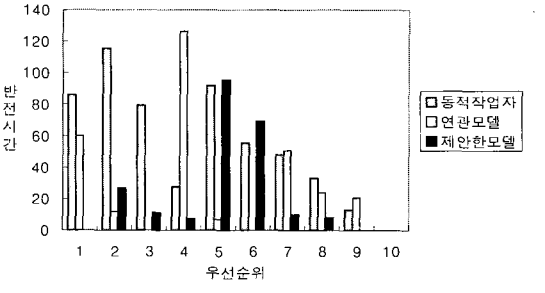
(그림 8) 우선 순위 반전시간 (단위:ms)

반면, 우선 순위 4~5에서 제안한 모델의 우선 순위 반전 시간이 다른 모델에 비해 많이 발생한 이유는 쓰레드 2에서 처리되어야 할 우선순위(4~5)인 작업이 낮은 우선 순위에 의해 상대적으로 큐에서 대기하는 시간이 길어졌기 때문인 것으로 판단된다. 전체 누적된 우선 순위 반전시간을 비교해보면 동적 작업자 모델은 312.18ms, 연관 우선 순위 작업자 모델은 375.51ms, 제안한 모델은 238.98ms로서 높은 우선 순위 요구의 반전시간 감소는 전체 우선 순위 반전시간 감소에도 영향을 준다는 것을 알 수 있다.

(2) 쓰레드가 3개일 경우

(그림 9)에서는 높은 우선 순위 1에서의 우선 순위 반전시간이 제안한 모델에서는 없다. 높은 우선 순위 요구의 즉각적인 처리 확률이 높아진다는 의미이다. 그러므로 높은 우선 순위 작업의 마감시한내 완료 확률도 높아 질 것으로 예측할 수 있다. 반면, 우선 순위 5~6에서 제안한 모델의 우선 순위 반전 시간이 다른 모델에 비해 많이 발생한 이유는 서버 스레드에 할당하는 우선순위 처리 범위에서 그 원인을 찾을 수 있을 것 같다. 연관 우선순위 작업자 모델은 (1~3), (4~6), (7~10)으로 부여하는데 비해, 제안한 모델에서는 (1), (2~4), (5~10)으로 할당하기 때문에, 세 번째 쓰레드에서 반전시간이 연관 우선순위 기법보다는 크게 발생할 수 있음을 알 수 있다.

전체적인 반전시간을 비교해 보면 동적 작업자 모델은 550.40ms, 연관 우선 순위 작업자 모델은 299.88ms 이었으며, 제안한 모델은 228.74ms로 제일 적은 우선 순위 반전시간을 보였다.

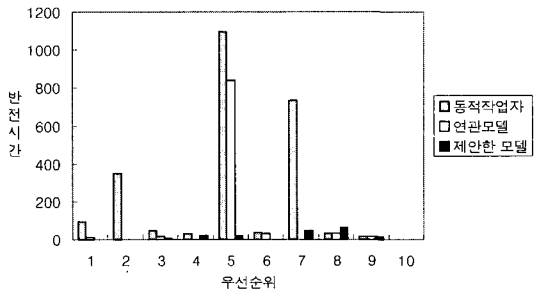


(그림 9) 우선 순위 반전시간 (단위:ms)

(3) 쓰레드가 4개일 경우

우선 순위 반전시간을 보면 (그림 10)에서 볼 수 있

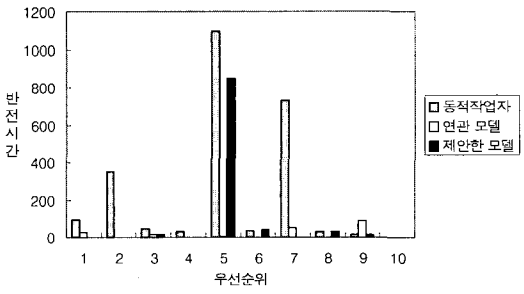
듯이 탁월한 성능을 보였다. 전체 우선 순위 반전시간을 보면 동적 우선 순위 작업자 모델은 2440.50ms, 연관 우선 순위 작업자 모델은 941.43ms, 제안한 모델은 171.65ms로 제일 좋은 결과를 갖는다. 특히 가장 높은 우선 순위 1과 2에서는 우선 순위 반전이 전혀 발생되지 않고 높은 우선 순위 요구들의 즉각적인 처리가 보장되었다.



(그림 10) 우선 순위 반전시간 (단위:ms)

(4) 쓰레드가 5개일 경우

모든 우선 순위 요구들이 균일하게 생성되고 쓰레드를 5개로 가정했을 때 결과이다. 이 부분에서 제안한 모델의 탁월한 결과를 볼 수 있다.

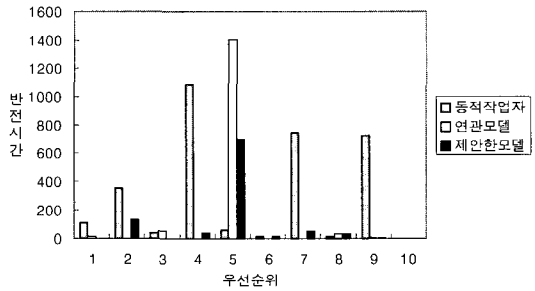


(그림 11) 우선 순위 반전시간(단위:ms)

(그림 11)의 우선 순위 반전시간을 비교해 보면 동적 작업자 우선 순위 모델이 2520ms, 연관 우선 순위 작업자 모델이 208ms 그리고 제안한 모델이 985ms로 나타났다. 제안한 모델의 경우 다른 모델과 비교해 이전 실험과 마찬가지로 높은 우선 순위 (1~4)에서 적게 발생했다. 즉 높은 우선 순위 요구의 적은 반전시간은 높은 우선 순위 요구의 마감시한내 완료될 확률이 높다. 마찬가지로 우선 순위 5에서의 반전시간이 크게 발생한 모습은 쓰레드 4에서 다른 하위 작업에 의해 많이 큐잉되었음을 알 수 있다.

4.3.2 경우 2

전체 쓰레드의 수를 4개로 있을 <표 1>의 [경우 2]와 같이 우선 순위에 따라 발생률을 다르게 부여하여 [경우 1]보다 높은 우선 순위 요구가 상대적으로 많이 발생되게 가정하였을 때 결과를 비교하였다. (그림 12)의 우선 순위 반전시간은 이전 경우와 비슷한 성향을 나타내었다. 특히 제안한 모델의 경우 우선 순위 1에서는 반전문제가 발생하지 않았음을 보여주고 전체 우선 순위 반전시간에서도 동적 우선 순위 작업자 모델이 3147.28ms, 연관 우선 순위 작업자 모델은 1516.11ms, 그리고 제안한 모델은 978.86ms로 가장 적게 나타났다.



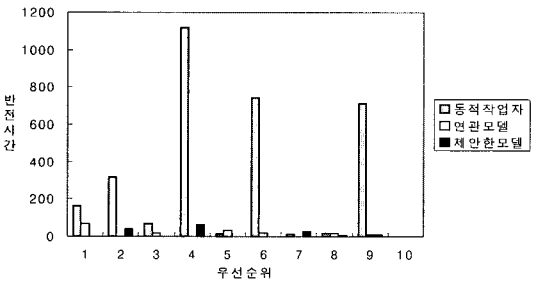
(그림 12) 우선 순위 반전시간 (단위:ms)

4.3.3 경우 3

전체 쓰레드 수를 4개로 고정하여 <표 1>의 [경우 3]과 같이 우선 순위에 따라 발생 분포를 다르게 부여하여 [경우 2]보다도 높은 우선 순위 요구에 대해서 상대적으로 많이 발생되게 가정하였을 때 결과를 비교하였다. 반전시간을 비교해 보면 역시 좋은 결과를 보여준다. 총 반전시간은 동적 우선 순위 작업자 모델은 3171.14ms, 연관 우선 순위 작업자 모델은 159.11ms, 그리고 제안한 모델은 146.28ms로 나타났다

(경우 2) 와 (경우 3) 실험에서, 우선순위 2인 경우를 보면 연관 우선순위 작업자 모델에 비해 제안한 모델에서 반전 현상이 발생한 원인은 작업자 쓰레드에게 처리할 수 있는 우선순위를 할당할 때 연관 우선순위에서는 쓰레드 1에 우선순위 범위를(1~2)를 할당하는데 비해, 제안한 모델에서는 쓰레드 2에 우선순

위 범위 (2~3)을 할당하므로 우선순위 3번 쓰레드에 의해 2번 쓰레드가 우선순위 반전이 발생하기 때문이다.



(그림 13) 우선 순위 반전시간 (단위:ms)

5. 결 론

실시간 시스템은 서버와 같은 공유자원들에 대해서 보다 높은 우선 순위 활동의 최악 블록 시간(worst case blocking time)을 결정해야 한다. 본 연구에서는 실시간 시스템에서 실시간성을 향상시키기 위해 기존 모델에서 발생했던 우선 순위 반전시간을 더욱 감소시키는 서버 모델을 제시하였다. 실험 결과, 실시간 시스템에서 가장 중요한 고려사항 중 하나인 마감시한 만족을 위한 기본 요소인 우선 순위 반전시간이 제안한 서버 모델이 높은 우선 순위에서 가장 많이 만족하는 것으로 나타났다. 이는 더욱 긴급한 처리를 요하는 작업에 대해 훌륭한 스케줄링을 할 수 있다는 것을 의미하며 특히 높은 우선 순위 요구에 대한 발생률을 증가시켰을 경우에 다른 모델에 비해 더욱 좋은 성능을 보여 실시간성을 요하는 응용에 적합한 모델이라 하겠다. 하지만, 낮은 우선순위를 갖는 쓰레드들은 그들이 할당될 수 있는 서버 쓰레드가 높은 우선 순위 쓰레드에 비해 적게 한정되기 때문에 전체적인 서버의 이용률 측면에서 다른 모델에 비해 떨어질 수 있다는 단점을 갖는다.

향후 연구과제로 본 연구에서 사용된 작업부하는 매우 단순화된 모델이다. 보다 정확한 실험을 위해서는 실시간 시스템의 작업 특성에 대한 연구가 필요하고 제안한 모델에 있어서 각 서버 쓰레드에게 처리가능한 우선 순위 할당을 위한 최적 알고리즘을 마련하는 것과 각 유휴한 쓰레드를 할당하는 과정에서 하위레벨에서 상위레벨로 할당해 줌으로써 서버의 이용률을 향상시킬 수 있는 방안도 연구해 볼 가치가 있다.

참 고 문 헌

[1] 이기웅, "실시간 시스템에서의 순위반전 대책 규약들의 성능분석," 한국과학기술원 석사학위논문, 1994.

[2] T. Nakajima, T. Kitayama, and H. Tokuda, "Experiments with Real-Time Servers in Real-Time Mach," In Proceeding of USENIX 3rd Mach Symposium, 1993.

[3] H. Tokuda and T. Nakajima, "Evaluation of Real-Time Synchronization in Real-Time Mach," In Proceeding of USENIX 2nd Mach Symposium, 1991.

[4] H. Tokuda and T. Nakajima, "Priority Inversion in Real-Time Communication," In Proceeding of 10th IEEE Real-Time System Symposium, Dec., 1989.

[5] 천경아, 박홍진, 김영찬, 전성익, "우선 순위 계승 프로토콜을 이용한 실시간 서버의 모델 연구," 한국정보과학회 추계학술회의 논문집, 제24권 2호, 1997.

[6] 김종훈, "실시간 Mach 시스템 하에서의 서버 모델의 문제점과 개선안," 한국외국어대학교 석사학위논문, 1995.

[7] T. Kitayama, T. Nakajima, H. Arakawa, and H. Tokuda, "Integrated Management of Priority Inversion in Real-Time Mach," IEEE Real-Time Systems Symposium, December, 1993.

[8] R. Rashid, R. Baron, A. Forin, D. Julin, D. Orr, R. Sanzi, D. Golub, M. Jones, "Mach : A Foundation for Open Systems," a Position Paper, In Proceeding of the 2nd Workshop on Workstation Operating Systems. IEEE, September, 1989.

[9] W.D. Kelton, R.P. Sadowski, D.A. Sadowski, "Simulation with Arena," WCB/McGraw-Hill, 1998.

[10] H. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach : Towards a Predictable Real-Time System," In Proceeding of USENIX 1st Mach Symposium, October, 1990.

최 대 수

e-mail : rtlinux@lycos.co.kr

1997년 호원대학교 전자계산학과 졸업(이학사)

1999년 수원대학교 대학원 전자계산학과 졸업(이학석사)

1999년~현재 수원대학교 대학원 전자계산학과 박사과정

관심분야 : 운영체제, RTOS, 실시간 통신, 분산 시스템

임 종 규

e-mail : jkim6@hanmail.net

1989년 목원대학교 컴퓨터공학과 졸업(학사)

1994년 수원대학교 전자계산학과 졸업(석사)

1997년 수원대학교 전자계산학과 재학중(박사)

1997년~현재 수원대학교 자연과학연구소 객원연구원

관심분야 : 운영체제, 개방형 분산 시스템, 전자상거래 시스템, 실시간 리눅스

구 용 완

e-mail : ywkoo@mail.suwon.ac.kr

1976년 중앙대학교 전자계산학과 졸업(학사)

1980년 중앙대학교 전자계산학과 졸업(석사)

1988년 중앙대학교 전자계산학과 졸업(박사)

1983년~현재 수원대학교 전자계산학과 정교수 및 전자계산소장

관심분야 : 분산 운영체제, 실시간 시스템, 시스템 네트워크 관리, 멀티미디어 시스템