MicroC/OS-II 실시간 운영체제에서의 우선순위 역전현상 해결 방법에 관한 효율적인 연구

전영식*, 허신*
*한양대학교 컴퓨터공학과

e-mail:klocust@hanyang.ac.kr

A Study on Effective Solution for Priority Inversion in MicroC/OS-II Real-time Operation Systems

Young-Sik Jeon*, Shin Heu*
*Dept of Computer Science Engineering, Han-Yang University

요 약

MicroC/OS-II에서는 우선순위 역전 현상에 대한 해결 기법으로 뮤텍스를 사용한 기본적인 우선순위 상속(Basic Priority Inheritance)을 사용한다. 뮤텍스를 구현하려면 리얼타임 커널이 우선순위가 같은 여러 태스크를 지원해야 한다. 하지만 MicroC/OS-II는 우선순위가 같은 여러 태스크를 지원하지 않는다. 이를 해결하기 위해 MicroC/OS-II는 우선순위 예약을 사용할 수밖에 없었으며 이로 인해 불필요한 메모리공간 및 우선순위 자원을 낭비하게 된다. 본 논문에서는 MicroC/OS-II에서의 불필요한 메모리 낭비와 우선순위 낭비를 줄여, 적은 용량의 메모리를 가지는 임베디드 장비에 효율적으로 운영되도록 제안 하고자 한다.

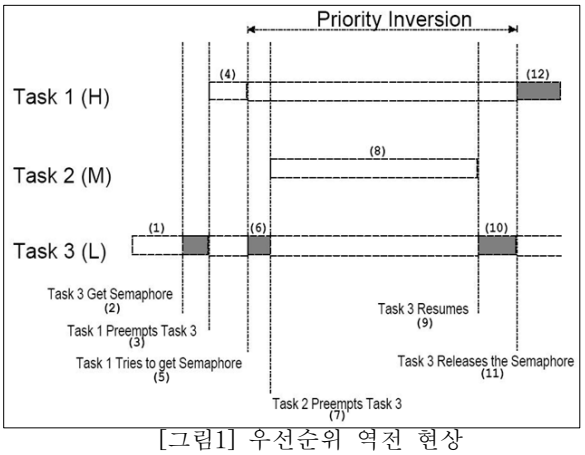
1. 서론

우선순위 역전 현상이란, 실시간 운영체제에서 특정 프로세스가 특정 자원을 필요로 하고, 그 해당되는 자원을 소유한 프로세스보다 높은 우선순위를 가짐에도 불구하고, 할당받지 못해 대기하는 현상을 말한다. MicroC/OS-II에서는 우선순위 역전 현상에 대한 해결 기법으로 뮤텍스를 사용한 기본적인 우선순위 상속(Basic Priority Inheritance)을 사용한다. 뮤텍스를 구현하려면 리얼타임 커널이 우선순위가 같은 여러 태스크를 지원해야 한다. 하지만 MicroC/OS-II는 우선순위가 같은 여러 태스크를 지원하지 않는다. 이를 해결하기 위해 MicroC/OS-II는 뮤텍스를 액세스하고자 하는 최상위 태스크 바로 위에 있는 우선순위를 예약해 놓고, 필요할 때만 낮은 우선순위 태스크의 우선순위를 임시로 올려서 사용하는 방식을 취한다. 이것으로 인해 우선순위 예약을 해야하고, 적은양이지만 불필요한 메모리공간을 낭비하게 된다. 실제로 임베디드 장비에 탑재되어 RAM의 사용량을 최소한으로 유지하려는 MicroC/OS-II의 목적에 위배되는 것이다. 본 논문에서는 MicroC/OS-II에서의 불필요한 메모리 낭비를 줄이고 불필요한 우선순위 예약를 없앰으로써, 적은 용량의 메모리를 가지는 임베디드 장비에 효율적으로 운영되도록 제안 하고자 한다.

2. 관련 연구

2.1. 우선순위 역전현상

그림[1]은 우선순위 역전현상이 발생할 수 있는 상황을 보여준다. 태스크 1은 태스크 2보다 우선순위가 높고, 태스크 2는 태스크 3보다 우선순위가 높다.



[그림1] 우선순위 역전 현상

태스크 3이 이미 소유한 자원을 태스크 1이 액세스하려고 할 때(5), 태스크 1의 우선순위는 사실상 태스크 3의 우선순위로 감소된다. 이 상황은 태스크 2가 태스크 3을 선점할 때 더욱 악화되며, 이로 인해 태스크 1의 실행은 더 지연된다.

2.2. BPI(Basic Priority Inheritance) 프로토콜

BPI 프로토콜은 자원을 소유하고 있는 프로세스에게, 해

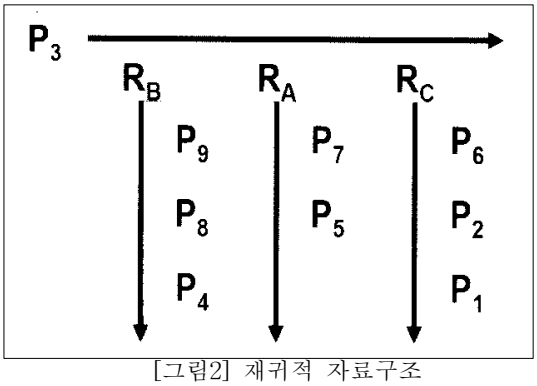
당 자원을 요청하고 대기하는 프로세스들 중에서 가장 높은 우선순위를 상속받아 해당 프로세스가 선점되지 않고 실행을 완료하도록 한 후, 자원을 반환할 때 원래의 우선순위로 복귀하는 방식이다.

2.3. 우선순위 최고 한도(Priority Ceiling) 프로토콜

세마포어별로 획득한 태스크를 실행하는 우선순위(ceiling 값)을 고정적으로 결정하는 방식이다. 이 방식에서는 세마포어가 고정된 실행 우선순위를 갖기 때문에 최적의 실행 순서를 스케줄링 할 수 있는 가능성이 있다.

2.4. RPI(Recursive Priority Inheritance) 프로토콜

RPI 프로토콜은 재귀적인 형태의 자료구조를 사용함으로써 복잡한 우선순위 역전 현상을 해결한다.



[그림2] 재귀적 자료구조

그림[2]에서 보이는 바와 같이 P_3의 재귀적 자료구조는 R_A, R_B, R_C를 소유하고 있으며, 각 자원들은 해당 자원을 요청한 프로세스 대기 큐를 가진다. 또한 각 자원별로 대기 중인 프로세스는 우선순위에 의해 정렬이 되어있고, 재귀적 자료구조를 가지는 P_3는 소유 중인 자원들을 각 자원의 첫 번째 대기 프로세스의 우선순위에 의해 정렬한다. 따라서 첫 번째 자원의 첫 번째 대기 프로세스가 항상 가장 높은 우선순위를 갖도록 정렬하고, 이 우선순위를 재귀적 자료구조를 갖는 프로세스가 상속받도록 한다. 즉, P_3는 R_B의 첫 번째 프로세스인 P_9의 우선순위를 상속받게 된다.

2.5. MicroC/OS-II 실시간 커널의 특징적 구조

a) 각 태스크는 같은 우선순위를 가질 수 없다.

MicroC/OS-II에서는 우선순위 역전 현상을 해결하기 위해 뮤텍스를 사용한다. 뮤텍스를 구현하려면 리얼타임 커널이 우선순위가 같은 여러 태스크를 지원해야 하는데, MicroC/OS-II에서는 우선순위가 같은 여러 태스크를 지원하지 않는다. 이 문제를 피하기 위해서 뮤텍스를 액세스하고자 하는 최상위 태스크 바로 위에 있는 우선순위를 예약해 놓고, 필요할 때만 낮은 우선순위 태스크의 우선순위를 임시로 올리는 방식을 사용한다.

b) 각 태스크는 오직 하나의 자원을 할당 받을 수 있다.

MicroC/OS-II에서 태스크 생성은 태스크 컨트롤 블록(TCB)을 할당함으로 이루어지며, 이 구조체 안에 세마포어용 이벤트 컨트롤 블록(ECB)을 가리키는 포인터 변수 OSTCBEVentPtr를 두어 태스크에 자원을 할당한다.

c) MicroC/OS-II에 우선순위 해결 기법의 적용

관련 연구를 비롯하여 우선순위 역전을 방지하는 방법은 오래전부터 연구가 진행되어 왔다. 하지만 이들을 MicroC/OS-II에 그대로 적용하기에는 무리가 있다. BPI 프로토콜의 경우 같은 우선순위를 가지지 못하는 MicroC/OS-II에는 곧바로 적용이 불가능 하다. 태스크당 다수의 자원을 가졌을 때 발생하는 우선순위 역전현상을 고려한 Priority Ceiling, RPI 프로토콜 또한 MicroC/OS-II에서는 불필요하다.

따라서 기존 MicroC/OS-II의 우선순위 역전현상 해결 알고리즘은 뮤텍스를 이용한 BPI를 적용하였으며, 우선순위 상속을 위해 뮤텍스를 액세스하고자 하는 최상위 태스크 바로 위에 있는 우선순위를 예약해 놓고, 필요할 때만 낮은 우선순위 태스크의 우선순위를 임시로 올려서 사용하는 방식을 취한다.

3. 설계

3.1. 우선순위의 일시적 교환 프로토콜 (Temporary Priority Swap Protocol)

본 논문에서는 효율적인 우선순위 상속 프로토콜을 구현하기 위해 이미 선점되있는 자원을 얻고자 하는 태스크 우선순위와 우선순위는 작지만 자원을 소유한 태스크의 우선순위를 서로 교환(swap)하는 방법을 사용한다.

이를 설계하기 위해서는 특정 프로세스가 이미 선점된 자원을 요청하는 경우와 자원 사용을 완료한 후 뮤텍스 반환시 우선순위 교환이 일어났을 경우 원래의 우선순위 값으로 되돌리는 경우도 생각해야 한다. 또한, 우선순위 역전 현상이 중첩적으로 일어났을 경우와 자원 사용을 완료한 후에 우선순위를 되돌리는 경우도 생각해야 한다.

이렇게 함으로써 기존의 BPI 프로토콜이 가지는 추가적인 ECB할당과 우선순위를 할당해야하는 문제, ECB에 대한 별도의 초기화작업을 수행해야하는 문제, RAM 공간 낭비 문제를 해결 할 수 있게 된다. 이것은 결국, 실제로 사용되는 RAM공간의 절약을 최대 목표로하여 실시간성을 유지하려는 MicroC/OS-II에 적합한 방법이다.

3.1. 관련 자료 구조

```
typedef struct os_tcb {
    INT16U          OSTCBId;     /* 태스크의 ID */
    OS_EVENT        *OSTCBEventPtr;
    /* 태스크가 소유한 이벤트 컨트롤 블록(ECB)   */
    INT8U           OSTCBPrio; /* 태스크 우선순위 */
    /* 이하 생략 */
} OS_TCB;
```

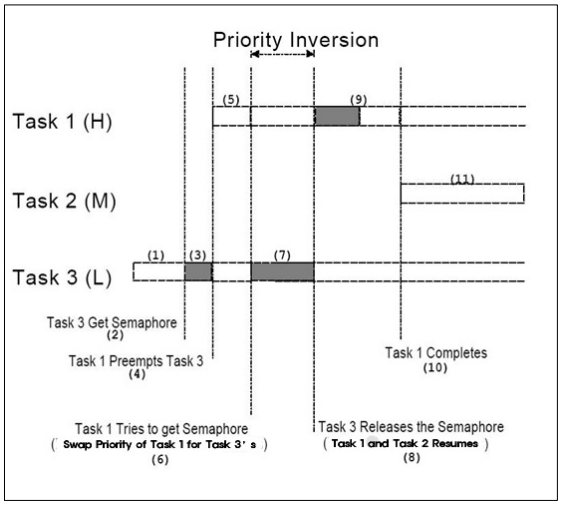
[표1]. 태스크의 정보를 저장하는 TCB 자료구조

```
typedef struct os_event {
    INT8U    OSEventType;
    /* OS_EVENT_TYPE_MUTEX  */
    void     *OSEventPtr;
    /* Pointer to message or queue structure */
    INT16U   OSEventCnt;
    /* 소유 태스크의 우선순위 및 Mutex 사용가능여부 */
    INT16U   OSEventTbl[OS_EVENT_TBL_SIZE];
    /* Mutex를 할당 받기 위해 대기중인 태스크들 */
    /* 이하 생략 */
} OS_EVENT;
```

[표2]. Mutex의 정보를 담아두는 ECB 자료구조

3.2. 알고리즘의 설계

그림[3]은 우선순위 역전현상이 발생할 수 있는 상황을 보여준다. 태스크 1은 태스크 2보다 우선순위가 높고, 태스크 2는 태스크 3보다 우선순위가 높다.



[그림3] TPSP알고리즘의 예

태스크 1이 태스크 3이 소유한 자원에 접근하려 할 때(6), 태스크 1과 태스크 3의 우선순위는 서로 교환된다. 커널은 태스크 1을 뮤텍스의 대기 목록에 올린 뒤 자원을 계속 사용 하도록 태스크 3을 재실행한다(7). 자원을 다 사용한 뒤(8), 태스크 1과 태스크 3의 우선순위를 원래 상대로 되돌려 놓는다.

3.3. 관련 함수 설명

```
void OSMutexPend(OS_EVENT* pevent,
                 INT16U timeout,
                 INT8U* err){
/* Mutex를 소유하고 있는 태스크의 우선순위가 Mutex를 요청한 태스크의 우선순위 보다 낮을 경우, 두 태스크의 우선순위 값을 교환한다. */
}
```

[표3]. 변경된 OSMutexPend 함수

```
INT8U OSMutexPost( OS_EVENT* pevent){
/* Mutex를 소유하고 있는 태스크의 우선순위 값이 변경되었을 경우, 변경되지 않은 태스크를 만날 때 까지 반복하여 OSMutex_RdyAtPrio 함수 호출 */
}
```

[표4]. 변경된 OSMutexPost 함수

```
OS_EVENT  *OSMutexCreate (INT8U *perr){
/* 우선순위를 할당 받아 ECB에 삽입하는 부분 제거 */
}
```

[표5]. 변경된 OSMutexCreate 함수

```
OS_EVENT  *OSMutexDel (OS_EVENT *pevent,
                       INT8U opt, INT8U *perr){
/* 할당받은 우선순위를 반환하는 부분 제거 */
}
```

[표6]. 변경된 OSMutexDel 함수

```
static  OS_TCB*  OSMutex_RdyAtPrio (OS_TCB *ptcb){
/* 원래의 우선순위 값을 가진 태스크와 서로 교환한다. */
}
```

[표7]. 변경된 OSMutex_RdyAtPrio 함수

4. 결론 및 향후 과제

우선순위 역전현상을 해결하는 3가지 프로토콜을 살펴보았다. 그리고 실시간 운영체제인 MicroC/OS-II에서 우선순위 역전현상 해결을 하기위해 사용하는 방법을 살펴보았으며, 기존의 MicroC/OS-II가 가지는 문제를 보완하기 위한 방법을 모색해 보았다. 하지만, MicroC/OS-II가 가지는 특수성 때문에 일반적인 PIP, PCP, RPI 프로토콜을 적용하는데 어려움을 겪었음을 발견하였다.

그러므로 MicroC/OS-II와 같은 특성을 가지는 경우에 적용될 수 있는 새로운 기법을 제안하게 되었다. 이렇게 함으로써 기존의 BPI 프로토콜을 구현하기 위해 추가적인 ECB를 할당해야하고 별도의 초기화작업을 수행해야하는 RAM 공간 낭비 문제를 절약할 수 있게 되었다. 또한, 우선순위 수에 제한이 있는 MicroC/OS-II에 우선순위 예약으로 인한 낭비를 해결할 수 있었다.

향후에는 우선순위 역전현상이 중첩적으로 일어났을 경우 효율적으로 해결하는 방안을 고려하고, 각 태스크 간 우선순위 중복 문제 해결, 여러 자원의 소유 가능, 보다 향상되고 이식성 높은 우선순위 역전 현상 해결 알고리즘을 적용하는 것이 필요할 것이다.

참고문헌

[1] Jean J. Labrosse, “MicroC/OS-II 실시간 커널 2판”, 에이콘, 2005년

[2] 강성구, 경계현, 고광선, 엄영익. “실시간 운영체제의 우선순위 역전현상 해결을 위한 프로토콜 설계 및 구현”. 한국정보처리학회, v.13A, no.5, pp.405-412, 2006

[3] L. Sha, R. Rajkumar, and J. P. Lehoczky, ″Priority Inheritance Protocols: An Approach to Real-Time Synchronization″, IEEE Transactions on Computers Vol. 39 No. 9, Sep. 1990, pp. 1175-1185.

[4] J. B. Goodenough and L. Sha, The Priority Ceiling Protocol, Special Report CMU/SEI-88-SR-4, Mar. 1998

[5] 영정 정무, “임베디드 시스템을 위한 RTOS″, 홍릉과학출판사. 2006. ISBN 8972835269.