□특집□

실시간 시스템의 개관

신 현 식† 김 태 웅††

◈ 목 차 ◈

1. 서 론

실시간 시스템은 기존의 컴퓨터 시스템과 달리 시스템 동작의 정확성이 논리적 정확성뿐만 아니라 시간적 정확성에도 좌우되는 시스템을 말한다. 이러한 실시간 시스템의 전형적인 예로서 제어 시스템을 들 수 있다. 제어 시스템은 감지장치(sensor)로부터 입력을 받아들여 이를 정해진 시간 내에 처리하여 작동장치(actuator)로 출력하며 극히 작은 시간적 오차를 허용한다. 실시간 시스템의 응용 분야로는 핵발전소의 제어, 공정 제어, 병원의 감시 장치, 항공기 제어, 무기 체계, 우주선의 운항 및 유도 등의 분야를 들 수 있다. 실시간 시스템에 존재하는 시간 제약 조건은 종료시한(deadline)으로 주어진다. 종료시한은 그것의 엄격성에 의해 세 가지로 분류될 수 있다. 첫째로, 경성(hard) 종료시한은 시스템이 주어진 종료시한을 만족시키지 못한 경우에 막대한 재산적 손실이나 인명의 피해를 주는 경우를 말한다. 둘째로, 연성(soft) 종료시한은 시간 제약 조건을 만족시키지 못하더라도 경성의 경우처럼 치명적이지 않고 종료시한을 넘겨 수행을 마쳐도 계산의 결과가 의미가 있는 경우를 말한다. 연성 종료시한을 갖는 대표적인 시스템은 온라인 트랜잭션 시스템을 들 수 있다. 마지막으로 준경성(firm) 종료시한은 경성과 연성의 중간 형태로 종료시한을 넘겨 수행을 마치는 것은 무의미한 경우를 의미하며 손실이 치명적이지 않은 경우를 말한다.

실시간 시스템이 종료시한을 만족시키기 위해서는 고속의 계산을 요구하게 되지만, 고속의 계산이 실시간 시스템의 요구조건을 만족시키는 것은 아니다. 일반적으로 고속의 계산은 시스템의 평균 응답시간을 최소화하지만, 실시간 시스템에서 요구되는 예측가능성을 보장하지는 않는다[20]. 예측가능성이란 시스템 명세에 정의된 고장이나 작업 부하 조건에서 태스크의 종료시한 만족을 보장하는 것을 의미한다.

실시간 시스템의 제약 조건은 앞에서 설명한 시간 제약 조건외에도 자원, 우선순위 또는 선행관계, 태스크간 통신 및 동기화 제약 조건이 있을 수 있다. 실시간 시스템을 구성하기 위해서 필요한 하드웨어와 소프트웨어는 다음과 특성을 갖는다. 우선 하드웨어는 고신뢰성을 제공하기 위해 결함허용성을 지원해야 하고 확장성과 유연성, 코드의 ROM화, 그리고 기존의 일반 부품을 사용하

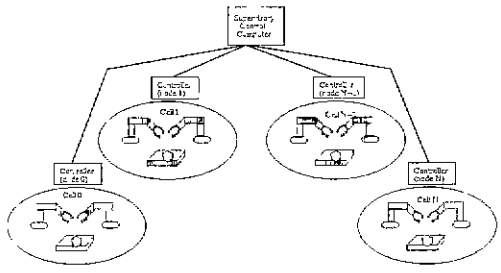
† 정회원 : 서울대학교 컴퓨터공학과 교수
†† 정회원 : 서울대학교 컴퓨터공학과 박사과정

여야 한다. 소프트웨어 부분은 실시간 운영체제 또는 실시간 실행체제등이 요구되며 이들은 태스크의 스케줄링, 태스크간 통신 및 동기화, 인터럽트 처리, 실시간 시계 관리등의 기능을 수행한다. 실시간 시스템의 주요 연구 분야로는 명세와 검증, 설계 방법론, 프로그래밍 언어, 스케줄링 알고리즘, 자원관리를 위한 운영체제의 기능, 실시간 통신 구조, 결함 허용성등을 들 수 있다. 이 글에서는 태스크 스케줄링 정책과 운영체제 그리고 통신 구조 분야를 중심으로 실시간 시스템의 특성에 관하여 검토한다.

2. 실시간 시스템의 예

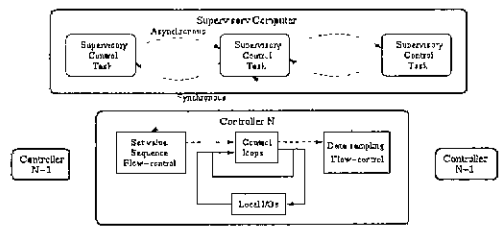
(그림 1)은 전형적인 공장 자동화 시스템을 보여 주고 있다. 이 시스템은 다수의 셀 제어기와 감시 제어부로 구성된다. 각 셀 제어기는 감시 제어부와 네트워크로 연결되어 있고 압력, 각도, 속도등을 제어하기 위해 일정 주기마다 각각에 대하여 제어 루프를 수행한다. 제어 루프는 감지장치와 같은 입력 장치에서 읽은 정보를 이용하여 작동장치에 출력을 보낸다. 입력에서 출력까지의 응답시간은 각 공정의 안정성과 밀접한 관련이 있으므로 일반적으로 제어 루프는 경성 실시간 태스크로 분류된다. 한편 감시 제어부는 각 셀 제어기로부터 데이타를 수집하여 사용자에게 디스플레이하거나 전체적인 공정 흐름을 조정한다. 또한 셀 제어기의 제어 루프의 기준값(setpoint value)을 제공하기도 한다. 이러한 상태 정보의 디스플레이와 같은 태스크는 사용자의 요구에 따라 비동기적으로 생성되며 시스템의 동작에 치명적이지 않기 때문에 연성 실시간 태스크로 분류된다.

감시 제어부에서 셀 제어기로의 명령을 처리하는 관리 태스크는 셀 제어기와 감시 제어부에서 서브태스크로 분할되어 수행된다. 예를 들어 셀



(그림 1) 공장 자동화 시스템

제어기에서 데이타를 읽는 서브태스크, 감시 제어부에서의 서브태스크, 감시 제어부로부터 수신된 데이타를 처리하여 제어 루프에 전달되는 서브태스크로 나눌 수 있다. 따라서, 관리 태스크의 응답시간은 이러한 각 서브태스크의 응답시간의 합으로 주어지며 응답시간이 작을수록 시스템의 성능 향상을 꾀할 수 있다. 공장 자동화 시스템에서의 태스크들간의 관계는 (그림 2)와 같다. 셀 제어기의 각 제어 루프는 하나의 주기적인 경성 실시간 태스크로 구성되며 관리 태스크의 서브태스크들은 동기적인 통신 프리미티브를 사용하며 직렬적으로 수행된다. 따라서, 이러한 공장 자동화 시스템은 경성 및 연성 실시간 태스크로 구성되며 각 태스크의 종료시한을 보장하는 스케줄링 기법이 요구된다. 한편, 네트워크를 통해 전송되는 메시지를 적절히 스케줄하는 기법이 요구된다. 현재 FDDI, 토큰링, Ethernet, FieldBus, CAN등의 네트워크들을 이용하여 경성 실시간 통신에 적용할 수 있다[3, 4, 8, 21, 23].



(그림 2) 공장 자동화 시스템의 태스크 예

3. 실시간 태스크의 스케쥴링

3.1 실시간 태스크의 특성

실시간 태스크는 도착 형태에 따라 주기적 태스크와 비주기적 태스크로 분류될 수 있다. 주기적 태스크는 일정 시간마다 활성화되어 자기자신의 종료시한내에 작업을 수행하여야 한다. 일반적으로 주기적 태스크는 경성 종료시한을 가진다고 간주한다. 비주기적 태스크는 외부로부터 발생되는 비동기적 사건의 처리를 위해 사용되는데, 사건의 중요도에 따라 경성과 연성 종료시한을 가질 수 있다. 실시간 태스크는 종료시한이라는 시간 제약 조건외에 선행 관계, 자원 요구 등의 조건을 가질 수 있다. 선행관계는 태스크들간의 수행순서에 관한 제약 조건으로서 태스크 T_i가 T_j이전에 수행이 끝나야 한다면 T_i는 T_j에 선행한다고 말한다. 태스크의 선행 관계 그래프는 DAG(directed acyclic graph)로 표현된다. 또한 태스크들은 프로세서이외의 자원, 즉 버퍼나 자료구조 등에 대한 경쟁이 있을 수 있다. 자원에 대한 경쟁은 잠금(lock)과 같은 메카니즘을 필요로 하는데 잠긴 자원을 사용하려는 태스크는 반드시 대기하여야 한다. 우선순위에 기반한 스케쥴링 정책을 사용하는 경우, 우선순위 역전(priority inversion) 현상이 발생할 수 있다. 우선순위 역전은 높은 우선순위를 갖는 태스크가 낮은 우선순위를 갖는 태스크에 의해 수행이 지연되는 현상을 말한다. 이러한 우선순위 역전 시간의 상한이 존재하지 않는 경우는 태스크의 종료시한 만족을 보장할 수 없다. 따라서 PCP(priority ceiling protocol)나 SRP(stack resource policy)와 같은 우선순위 역전 시간을 한정할 수 있는 자원 관리 정책을 사용함으로써 예측가능성을 높일 수 있다[2, 16].

3.2 실시간 태스크 스케쥴링

3.2.1 주기적 태스크

실시간 스케쥴링 알고리즘의 기능은 주어진 태스크 집합에 대하여 태스크의 시간, 선행, 자원 요구 조건을 만족하는 스케쥴이 존재하는지를 결정하고(feasibility analysis) 만약 태스크의 요구 조건을 만족하는 스케쥴이 존재하면 그러한 스케쥴을 생성하는 것이다(schedule construction). 실시간 태스크의 스케쥴링 알고리즘은 다음과 같이 세 가지로 분류할 수 있다.

- 테이블에 기반한 실행전 스케쥴링
- 고정 우선순위에 기반한 스케쥴링
- 가변 우선순위에 기반한 스케쥴링

Xu와 Parnas가 제안한 테이블에 기반한 실행전 스케쥴링은 주어진 태스크 집합에서 종료시한, 선행, 상호배제등의 제약 조건을 만족시키는 스케쥴을 생성한다음 태스크의 시작과 끝시간을 테이블에 저장하여 테이블의 내용에 기초하여 태스크를 수행시킨다[24]. 이 방법은 높은 예측가능성을 제공하지만 태스크가 변하게 되면 테이블 전체를 다시 생성해야 하기 때문에 유연성이 떨어진다.

고정 우선순위에 기반한 스케쥴링은 RMS(rate monotonic scheduling)[11]과 DMS(deadline monotonic scheduling)[1]이 널리 사용된다. 이 알고리즘들은 각각 주기적 태스크들의 스케쥴가능성을 검사하는 조건이 증명되어 있다[1, 10]. RMS는 주기가 작은 태스크에 높은 우선순위를 할당하는 방법을 취하고 DMS는 종료시한이 작은 태스크에 높은 우선순위를 할당한다. 이 두 알고리즘은 최적의 고정 우선순위 스케쥴링 알고리즘으로 증명되어 있다. 최적의 고정 우선순위 스케쥴링 알고리즘이란 임의의 고정 순위 스케쥴링 알고리즘이 주어진 태스크의 집합을 스케쥴할 수 있으면, 최적 스케쥴링 알고리즘은 항상 실행가능한 스케쥴을 생성할 수 있다는 것을 의미한다. 즉, 최적 알고

리즘으로 스케줄될 수 없는 태스크 집합에 대해서는 다른 어떤 고정 우선순위 스케줄링 알고리즘으로도 스케줄할 수 없다는 것을 말한다.

가변 우선순위에 기반한 스케줄링은 EDF (earliest deadline first)[11]와 LSF(least slack first)[13]등의 정책을 사용한다. EDF는 종료시한이 가까운 태스크에 높은 우선순위를 동적으로 할당하고, LSF는 여유시간(slack)이 작은 태스크에 높은 우선순위를 할당한다. 태스크의 여유시간은 종료시한에서 태스크의 남은 수행 시간을 뺀 값으로, 만약 여유시간이 0이면 즉각 수행하지 않으면 종료시한을 넘길 수 있다. EDF와 LSF를 사용하는 태스크 집합의 프로세서 이용률이 1이하이면 종료시한 만족을 보장할 수 있다. 이들 두 알고리즘도 각각 가변 우선순위를 사용하는 경우 최적 스케줄링 알고리즘으로 증명되어 있다.

3.2.2 비주기적 태스크

실시간 시스템에서 연성 종료시한을 가지는 비주기적 태스크는 종료시한보다는 응답시간에 초점을 맞춰 스케줄하게 된다. 즉, 경성 주기적 태스크들의 종료시한을 만족시키면서 비주기적 태스크의 응답시간을 줄이는 것이 성능상의 목표이다. 그러나, 경성 종료시한을 갖는 비주기적 태스크는 경성 주기적 태스크와 마찬가지로 종료시한을 보장해 주어야 한다.

연성 비주기적 태스크를 스케줄하는 방법 중 백그라운드 처리는 주기적 태스크를 우선적으로 수행하고 남는 시간에 비주기적 태스크를 수행한다. 이 방법을 사용하면 비주기적 태스크의 응답시간이 길어지게 된다. 응답시간을 개선하기 위해 주기적으로 수행되는 서버를 두는 대역폭 보존 알고리즘으로는 DS, PE, SS, TBS등의 알고리즘이 있다[6, 18, 19]. 복잡한 계산을 요구하지만 최소 응답시간을 제공하는 알고리즘으로는 여유시간 도용 알고리즘이 있다[9].

경성 비주기적 태스크를 스케줄하는 가장 간단한 방법은 비주기적 태스크의 최소 도착간 시간(minimum interarrival time)을 주기로 갖는 주기적 태스크로 변환하여 종료시한의 만족을 보장한다[13]. 그러나, 이러한 방법은 비주기적 태스크의 작업 부하를 과대 평가하였기 때문에 시스템에서 수용할 수 있는 태스크의 수가 감소하게 된다. 이러한 단점을 개선하기 위해 주기적 태스크의 여유시간을 계산하여 경성 비주기적 태스크의 수용가능성 검사하는 알고리즘들로는 Schwan의 슬롯 리스트, Chetto의 연구, 여유시간 도용 알고리즘등이 있다[5, 15, 17].

3.2.3 부정확한 계산

부정확한 계산은 일시적인 장애나 과부하동안 시스템이 일정 수준의 응답성을 보장하고 점진적인 성능저하를 제공하기 위해 사용된다. 부정확한 계산의 기본 아이디어는 실행동안 항상 결과를 리턴할 수 있는 알고리즘이 존재한다는 것이다. 즉, 이미지 처리나 추적(tracking)과 같은 응용이나 반복 알고리즘에서는 수행시간이 길어짐에 따라 정확한 결과를 생성하지만 만약 시스템 장애나 과부하시에 계산을 미리 종료하더라도 그 때까지 생성한 결과는 수용가능한 계산의 결과일 수 있다는 것이다. 따라서 어느 정도의 오차까지 계산을 수행한 후 시간이 없을 때는 그때까지의 결과를 리턴한다. 이때 생성된 결과는 정확하지는 않지만 수용가능한 결과이다. 이러한 방법을 이용하여 부정확한 시스템에서는 결과의 질과 시간을 맞바꿈으로써 태스크 스케줄의 융통성을 제공한다. 부정확한 시스템에서는 하나의 태스크가 반드시 수행해야할 중요한 태스크인 필수적 태스크와 덜 중요한 선택적 태스크로 분리되어 있다. 부정확한 시스템에서의 스케줄링은 모든 필수적 태스

크는 종료시한내에 수행을 마칠 수 있도록 스케줄하고 선택적 태스크는 정상적인 경우에는 정확한 결과를 얻기 위해 스케줄하지만 과부하시에는 전체 에러가 최소화되도록 스케줄한다. 필수적 태스크의 종료시한을 만족하면서 전체 에러를 줄이는 최적화 알고리즘은 모두 EDF의 변형인 ED알고리즘을 이용한다[12].

4. 실시간 운영체제

최근 운영체제는 컴퓨터의 성능을 향상시키는데 많은 도움이 되고 있다. 그러나, 앞에서 설명한 바와 같이 실시간 시스템에서의 성능은 단지 평균 실행시간에 의해서만 측정되지는 않는다. 시스템의 성능을 향상시키는데 많은 도움이 되는 범용 운영체제의 특징들이 실시간 응용에서는 많은 문제점을 나타낼 수도 있다. 본 글에서는 일반적인 운영체제와 구별되는 실시간 운영체제(이하 RTOS)의 특징과 설계상의 고려사항에 대해 기술한다.

4.1 RTOS의 특징

4.1.1 결정성

결정성은 명확하게 정의된 시간동안에 특정 동작이 수행되는 시스템의 성향을 말한다. 완전 결정성 시스템은 주위 환경에 관계없이 항상 일정한 시간 내에 동작이 완료된다. 역으로 완전 비결정성 시스템은 동작되는 시간의 상한이 없는 경우를 말한다. 그러나, RTOS를 포함한 모든 운영체제는 완전 결정성 시스템을 제공하고 있지는 않다. RTOS의 중요한 성능 척도중의 하나가 바로 결정성 동작수행 정도이다.

4.1.2 응답성

응답성은 어떠한 사건에 대해 빨리 응답하는 시스템의 능력을 말한다. 동기적인 사건은 시스템의 성능에 중요한 영향을 미치는 수행속도를 가진 연속적인 동작을 발생시키므로, 동기적인 사건에 대한 응답성은 보통 실시간 시스템과 그렇지 않은 시스템간의 중요한 차이는 아니다. 그러나, I/O 인터럽트와 같은 비동기적인 사건은 외부 사건에 대해 즉각적으로 반응해야 하는 실시간 시스템에서 중요한 요소이다.

인터럽트 응답 시간은 몇가지 중요한 구성요소를 가진다. 첫째는, 인터럽트의 승인이 지연될 수 있는 최대 시간이다. 둘째 요소는 최초로 인터럽트를 처리하고 ISR(interrupt service routine)을 수행시키기 시작하는데 요구되는 총 시간이다. 이 시간은 ISR의 종류에 따라 다르다. 만약 ISR이 사용자 작업과 문맥교환을 포함하면, 최대 지연시간은 중요한 응답시간의 요소가 된다. 셋째 요소는 실제로 인터럽트를 서비스하는데 필요로 하는 총 시간이다. 네 번째 요소는 시스템 응답시간에 중첩된 인터럽트가 미치는 영향이다. 시스템이 인터럽트의 중첩을 허용하고 처리하는 정도는 결정성 작용과 응답성에 영향을 미치기 마련이다.

응답시간은 실시간 시스템에서 매우 중요하다. RTOS는 I/O를 통해 외부 시스템과 상호작용을 한다. 그러므로, I/O 응답성과 I/O 동작의 결정성은 실시간 계산 시스템에서 중요한 특징이다.

4.1.3 사용자 제어

범용 시분할 운영체제에서의 사용자 제어는 공정성과 응답성의 특징에 맞게 서비스를 하도록 설계되어 있기 때문에 제한적이다. 그러나, RTOS는 시스템 동작의 최종적 제어를 사용자에게 부여한다. 이러한 사용자 제어는 어떠한 작업 수행에 대해 고정된, 사용자에 의해 정의된 우선순위의 개념에서 시작된다. 이것은 실시간 시스템에서 제공하는 스케줄링 정책일 뿐만 아니라, 매우 일반적인 개념이다. 시스템이 사용자에게 작업의 상

대적 우선순위를 부여하도록 하는 것은 사용자가 처리율과 응답성을 정할 수 있게 하였다.

사용자 제어는 작업 우선순위를 정하는 것에서 끝나지 않는다. 이것은 시스템의 모든 영역으로 확장된다. 실시간 시스템은 사용자에게 페이징의 사용 혹은 프로세스 스와핑, 어떤 프로세스가 메모리에 상주해야 하는지, 어떤 디스크 전송 알고리즘이 사용되어야 하는지등과 같은 특징을 정하도록 할 것이다.

때때로, 실시간 시스템은 사용자에게 특권을 부여한다. 말하자면, 사용자 프로세스를 특권 모드로 이동시킬 수 있는 권리를 부여하는 것이다. 이것은 사용자 프로세스에게 프로세서에 대한 제어를 상실하지 않고 시간이 임박한 동작이 수행되는 것을 보장하기 위해 전체 인터럽트 시스템을 정지시키고, 고성능 데이타 전송을 위해 메모리 보호 절차를 위반하는 것과 같은 행동을 할 수 있게 하였다. 사용자 프로세스가 특권 모드로 진입하는 권리를 부여한 것은 비실시간 시스템에서는 존재하지 않는 중요한 특징중의 하나이다.

4.1.4 신뢰성

RTOS에서의 신뢰성은 장기간동안 시스템이 오류 없이 계속 동작할 수 있는 것을 의미한다. 실시간 시스템은 시간을 다루는 실세계 시스템을 제어하므로, 재부팅을 요구하는 실시간 시스템에서의 오류는 막대한 금전적 손해나 인간의 생명을 위협하는 문제를 야기할 수 있다. RTOS에서의 신뢰성은 또한 필요한 모든 자원을 미리 할당함으로써 경성 실시간 프로세스가 수행될 수 있도록 보장하는 것을 의미한다.

4.2 RTOS의 설계

예측가능한 RTOS를 설계하는데 두 가지의 접근 방법으로서 사건 구동형 구조와 시간 구동형 구조가 있다. 사건 구동형 RTOS는 모든 시스템의 동작이 특정 사건에 대한 응답으로 기동되는 시스템을 의미하고, 시간 구동형 RTOS는 시스템의 동작이 미리 정해진 동기화된 시간에 의해 기동되는 시스템을 의미한다. RTOS의 예측가능성은 각 태스크의 수행전에 자원에 대한 요구 조건을 만족시킬 수 있는지 검사함으로써 얻어진다. 사건 구동형 시스템은 자원에 대한 요구와 자원의 가용성이 시간에 따라 변화하기 때문에 동적으로 자원 요구 조건에 대한 검사를 수행한다. 반면에 시간 구동형 시스템은 이러한 요구조건에 대한 검사를 오프라인에 수행하여 자원에 대한 경쟁을 해결한다. 따라서, 사건 구동형 시스템은 다양한 동적 환경에 대처해야 하기 때문에 복잡도가 증가하는 단점이 있고 시간 구동형 시스템은 유연성이 떨어지는 단점이 있다. 그러나, 어떤 접근 방법을 사용하더라도 자원에 대한 요구와 가용성에 대한 검사는 응용의 시간 제약조건을 고려하여 수행되어야 한다. 따라서 시스템의 시간적 동작을 결정짓는 시간에 대한 관리가 RTOS의 설계에 있어 가장 중요한 요소이다.

5. 실시간 통신

(그림 1)와 같이 실시간 시스템은 서로 다른 처리기에서 수행되는 태스크들이 자신의 계산 결과를 메시지로 전송함으로서 서로 통신할 수 있다. 시스템의 시간 제약 조건을 만족시키기 위해서 이러한 태스크간 통신은 제한된 시간내에 이루어져야한다. 통신에서 발생되는 지연시간은 통신 매체를 통하여 메시지를 전송하는데 걸리는 시간과 목적지에서 메시지가 목적 태스크에게 전달되기까지 걸리는 시간으로 구성된다. 따라서 실시간 통신에서는 매체 접근 지연시간과 메시지 전달 지연시간의 상한이 정해져야 한다. 매체에

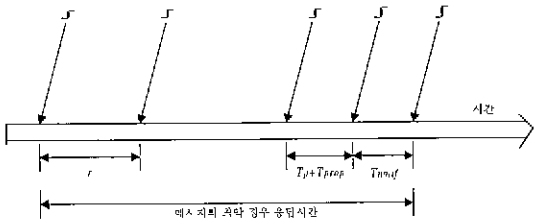
대한 접근 시간을 한정하기 위해서 실시간 시스템의 통신에서는 토큰 링이나 FDDI와 같이 토큰을 전달하는 기법이나 TDMA 방식등의 매체 접근 방식을 주로 사용한다. 메시지 전달 시간은 목적지에서의 태스크 스케줄링 정책에 의존적이다.

5.1 종점간 통신 지연시간 분석

한 태스크에서 다른 태스크로의 메시지 전달 시간을 종점간 통신 지연시간이라고 부르며, 이 지연시간은 다음의 네 가지 요소로 분할될 수 있다[22].

- 생성 지연시간(generation delay)
- 큐 지연시간(queuing delay)
- 전송 지연시간(transmission delay)
- 전달 지연시간(delivery delay)

생성 지연시간은 응용 태스크가 메시지를 생성하고 메시지를 통신 매체의 전송 큐에 삽입할 때까지 걸리는 시간으로서, 송신 태스크가 도착해서 메시지가 큐잉되는 시간까지의 최대값을 의미한다. 큐 지연시간은 메시지가 통신 매체에 접근하기 전까지 큐에서 대기한 시간을 말한다. 지점간 통신 링크에서 한 메시지는 동일 처리기에서 보내는 다른 메시지와 경쟁하여야하고, 공유 통신 링크에서 메시지는 다른 처리기에서 보내는 메시지들과 경쟁하여야 한다. 전송 지연시간은 통신 매체를 통하여 메시지를 전송하는데 걸리는 시간을 의미하고, 전달 지연시간은 목적지에서 도착된 메시지가 수신 태스크에게 전달되기까지 걸리는 시간을 의미한다. 전달 지연시간은 패킷의 헤더의 해석, 다중 패킷으로 구성된 메시지의 재조립, 메시지 데이타를 버퍼로 복사, 그리고 메시지의 도착을 스케줄러에게 알리는 작업을 포함한다. 종점간 통신 지연시간에서 전달 지연시간은 중요한 위치를 차지한다. 왜냐하면 수신 태스크가 메시지의 도착을 기다리며 블럭되어 있기 때문이다. 이 과정을 (그림 3)에 나타낸다.



1	송신 태스크의 도착
2	메시지가 큐에 들어가는 가장 늦은 시간
3	메시지의 마지막 패킷이 큐에서 제거되는 순간, 즉 마지막 패킷의 전송이 시작되는 시간 T_p는 패킷을 전송하는데 걸리는 최악 시간 T_{prop}는 전기적인 전파 지연시간
4	마지막 패킷이 목적지에 도착
5	목적지 태스크의 수행이 시작되는 시점

(그림 3) 종점간 통신 지연시간

5.2 실시간 통신용 매체 접근 제어(MAC)

5.2.1 토큰 전달 기법

실시간 시스템의 통신에 사용되는 MAC 프로토콜 중 토큰을 전달 기법은 토큰 링과 FDDI가 있다. 토큰 전달 방식에서는 토큰을 가지고 있는 노드만이 메시지를 전송할 수 있으므로, 모든 메시지들의 종료시한을 만족시키기 위해서는 한 노드가 토큰을 가지고 있는 시간이 제한되어야 한다. 이러한 토큰 전달 방식을 timed token 방식이라 부른다.

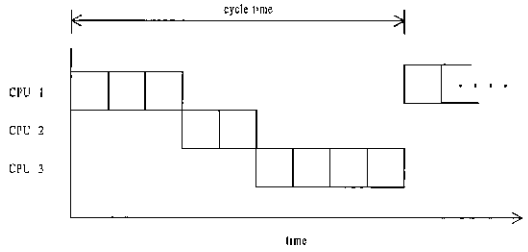
토큰 링은 메시지들의 우선순위에 기반한 매체 접근을 지원하기 위해 예약 방식에 기반하고 있다[21]. 그리고 각 노드는 토큰당 한 패킷을 전송한다. 토큰에 존재하는 우선순위 필드와 예약 필드를 이용하여 메시지간의 경쟁을 중재한다. 예약 필드를 통해서 토큰이 링을 한바퀴 돌고 나면 현재 대기 중인 패킷 중 가장 우선순위가 높은 패킷을 가진 노드가 토큰을 얻게 된다.

FDDI는 timed token 방식의 하나로서 실시간 통신을 위해 많은 연구가 진행되었다. FDDI는 동

기적 메시지와 비동기적 메시지를 구분하여 전송하는데 동기적 메시지는 일반적으로 실시간 메시지를 전송하는데 사용된다. FDDI의 각 노드는 토큰을 수신하면 동기적 메시지를 할당된 대역폭만큼 보낸다. 비동기적 메시지는 비주기적 메시지나 비실시간 메시지를 보내는데 사용된다. 실시간 메시지의 종료시한을 만족시키기 위해 각 노드에 할당되는 동기적 메시지용 대역폭을 할당하는 문제가 많은 사람들에 의해 연구되었다[4].

5.2.2 TDMA 프로토콜

비엔나 대학의 MARS 프로젝트[7]에서 주로 연구된 TDMA 프로토콜은 처리기간의 통신 매체에 대한 경쟁을 다음과 같이 중재한다. 한 노드내의 지역 메시지는 우선순위 큐로 순서를 정하고, 다른 노드의 메시지들과의 경쟁은 라운드로빈을 통해 해결한다. TDMA하에서는 노드는 자신의 슬롯에 정해진 수의 패킷을 실어 보낼 수 있다. 슬롯은 고정된 크기를 가지고 보낼 패킷이 없는 경우는 슬롯이 끝날때까지 통신 매체가 유휴상태에 놓이게 된다. 각 처리기는 서로 다른 고정된 슬롯 크기를 가질 수 있다. 한 슬롯이 끝나면 또 다른 노드가 패킷 전송을 시작하게 된다. 모든 노드들이 접근 기회를 가진 후 위의 동작을 반복하게 된다. TDMA는 패킷의 전송이 충돌이 일어나지 않게 하기 위해 노드간 시계 동기화(clock synchronization)를 필요로 한다. 또한 패킷의 전송 시간은 변하지 않고 상수 시간으로 가정한다.



6. 결 론

이상에서 실시간 시스템의 특징과 태스크 스케쥴링 정책, 실시간 운영체제, 실시간 통신에 대해 알아보았다. 미래의 실시간 시스템은 규모가 커지고 복잡해지고 분산 환경과 동적 환경에서 동작할 것이다. 그리고 전문가 시스템 요소를 포함하게 되고 서로 다른 단위의 시간을 포함하는 복잡한 시간 제약조건을 가지게 된다. 더욱이 이러한 시간 제약조건을 만족시키지 못하면, 경제, 인명, 생태학적 재난을 초래할 것이다. 대규모 실시간 시스템을 개발하는 해결책은 현재의 OR, 데이타베이스, 스케쥴링, 운영체제등의 이론만으로는 부족하다. 실시간 시스템의 연구 분야인 명세와 검증, 설계 방법론, 프로그래밍 언어, 스케쥴링 알고리즘, 운영 체제 기능 설계, 통신 아키텍쳐등의 연구 결과를 잘 조정하고 확장시킴으로써 대규모 실시간 시스템의 개발을 앞당길 수 있을 것이다.

참고문헌

[1] N. Audsley, A. Burns, M. Richardson, and A. Wellings, "Hard Real-Time Scheduling: The Deadline-Monotonic Approach", Proceedings of IEEE Workshop on Real-Time Operating Systems and Software", pp. 133-137, 1991.

[2] T. Baker, "A Stack-based Resource Allocation Policy for Real-Time Processes", Proceedings of Real-Time Systems Symposium, pp. 191-200, 1990.

[3] S. Cavalieri, A. Stefano, and O. Mirabella, "Pre-run-time Scheduling to Reduce Schedule Length in FieldBus Environment", IEEE Trans. on Software Engineering, Vol. 21, No. 11, pp. 865-880, 1995.

[4] B. Chen, G. Agrawal, and W. Zhao, "Optimal Synchronous Capacity Allocation for Hard Real-Time Communications with the Timed Token Protocol", Proceedings of Real-Time Systems Symposium, pp. 198-207, 1992.

[5] H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm", IEEE Trans. on Software Engineering, Vol. 15, No. 10, 1989.

[6] T. M. Ghazalie and T. P. Baker, "Aperiodic Servers in a Deadline Scheduling Environment", Real-Time Systems, Vol. 9, No. 1, pp. 31-67, 1995.

[7] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger, "Distributed Fault Tolerant Real-Time Systems: MARS Approach", IEEE Micro, pp. 25-40, Feb. 1989.

[8] J. Lee and H. Shin, "A variable bandwidth allocation scheme for Ethernet-based real-time communication," Proc. of First Int'l Workshop on Real-Time Computing Systems and Applications, pp. 28-32, 1995.

[9] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft- Aperiodic Tasks in Fixed-Priority Preemptive Systems", Proceedings of Real-Time Systems Symposium, pp. 110-123, 1992.

[10] J. Lehoczky, L. Sha, and Y. Ding, "The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", Proceedings of Real-Time Systems Symposium, pp. 166-171, 1989.

[11] C. Liu and J. Layland, "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment", The Journal of the ACM, Vol. 20, No. 1, pp. 46-61, 1973.

[12] J. W. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise Computations", Proceedings of IEEE, Vol. 82, No. 1, pp. 83-94, 1994.

[13] A. K. Mok, "Fundamental Design Problem of Distributed Systems for the Hard Real-Time Environment", Ph.D Thesis, MIT, 1983.

[14] K. Ramamritham and J. A. Stankovic, "Scheduling Algorithms and Operating Systems Support for Real-Time Systems", Proceedings of the IEEE, Vol. 82, No. 1, pp. 55-67, 1994.

[15] K. Schwan and H. Zhou, "Dynamic Scheduling of Hard Real-Time Tasks and Real-Time Threads", IEEE Trans. on Software Engineering Vol. 18, No. 8, pp. 736-748, 1992

[16] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Trans. on Software Engineering, Vol. 39, No. 9, pp. 1175-1185, 1990.

[17] S. Ramos-Thuel and J. P. Lehoczky, "On-line Scheduling of Hard Deadline Aperiodic Tasks in Fixed Priority Systems", Proceedings of Real-Time Systems Symposium, pp. 160-171, 1993.

[18] B. Sprunt, L. Sha, and J. P. Lehoczky, "Aperiodic Task Scheduling for Hard Real-Time Systems", Real-Time Systems, Vol. 1, pp. 27-69, 1989.

[19] M. Spuri and G. C. Buttazzo, "Efficient Aperiodic Service under Earliest Deadline Scheduling", Proceedings of Real-Time Systems

Symposium, pp. 2-11, 1994.

[20] J. Stankovic, "Misconceptions about Real-Time Computing", IEEE Computer, Vol. 21, No. 10, pp. 10-19, 1988.

[21] J. K. Strosnider, T. Marchok, and J. P. Lehoczky, "Advanced Real-Time Scheduling Using the IEEE 802.5 Token Ring", Proceedings of Real-Time Systems Symposium, pp. 42-52, 1988.

[22] K. Tindell, A. Burns, and A. J. Wellings, "Analysis of Hard Real-Time Communications", Real-Time Systems, Vol. 9, No. 2, pp. 147-171, 1995.

[23] K. Tindell, H. Hansson, and A. Wellings, "Analysing Real-Time Communications: Controller Area Network(CAN)", Proceedings of Real-Time Systems Symposium, pp. 259-263, 1994.

[24] J. Xu and D. Parnas, "Scheduling Processes with Release Times, Deadlines, Precedence and Exclusion Relations", IEEE Trans. on Software Engineering, Vol. 16, No. 3, pp. 360-369, 1990.

신 현 식

1973년 서울대학교 응용물리학과 (공학사)
1980년 미국 텍사스대학교 의공학 (석사)
1985년 미국 텍사스대학교 컴퓨터 공학과 (박사)
1986년-1990년 서울대학교 컴퓨터공학과 조교수
1990년-1996년 서울대학교 컴퓨터공학과 부교수
1996년-현재 서울대학교 컴퓨터공학과 교수
관심분야 : 분산 시스템, 실시간 시스템

김 태 웅

1993년 서울대학교 컴퓨터공학과 (공학사)
1995년 서울대학교 컴퓨터공학과 (석사)
1995년-현재 서울대학교 컴퓨터 공학과 박사과정
관심분야 : 실시간 스케줄링, 실시간 통신, 분산 시스템