시스템 개발을 위한 특성 분석

김학배

연세대학교 전기공학과

1. 서론

실시간 시스템이란 일반적으로 논리적인 계산결과 뿐만 아니라 그 결과가 산출되어야 하는 시간에도 적시성을 갖는 시스템을 의미한다. 즉, 단순히 데이터 폭주에 대해 빠른 속도로만 반응하여 문제를 해결하는 것이 아닌, 어떠한 상황의 열악한 환경 하에서도 정해진 (요구되는) 시간내에 정확한 출력 결과를 산출해낼 수 있는 시스템을 의미하게 된다. 따라서 일반적 비실시간 시스템과 다르게 생각되는 중요 고려요소는 다음과 같이 분류될 수 있다.

- 시간(time): 실시간 시스템의 가장 중요한 요소로서, 복잡하게 얽혀있으면서도 각각의 성격은 판이한 일들이 분배(allocation & scheduling)와 수행(execution)을 거쳐 정해진 시간(deadline)안에 모든 처리가 끝날 수 있어야 함을 의미한다. 시간적 정확성을 확보하기 위해서는 일의 수행 전에 그 특성과 중요도(priority)들에 대한 사전 정보가 있어야 한다. 따라서, 정확한 시간적 수행능력은 시간 변수가 중요 요소로 포함된 각 작업에 대한 모든 정보를 기초로 하여 수행과 배치를 결정하는 실시간 스케줄링을 통해 확보될 수 있다.
- 신뢰도(reliability): 실시간 시스템은 주로 한번의 실수(failure)가 되돌이킬 수 없는 재앙적 결과(catastrophe)를 초래할 수 있는 환경에서 활용되고 있기 때문에 고장에 대한 강인성과 예측 가능성에 대한 확보가 필수적이다. 즉, 모든 실시간 시스템에 적용되는 정책과 기법들은 시간적 한계 또는 수행과제의 정확성에 대한 예측과 고장 가능성에 대한 대비와 제거 또한 이에 따른 신뢰도 확보를 기본 목표로 하고 있다.
- 환경(environments): 실시간 시스템은 동작되어야 할 환경적 요소를 고려하여야 한다. 예를 들어, 비행시스템의 경우 광범위한 동작온도, 기압, 진동 등을 생각하여야 하며 구름을 지날 때의 전자기 간섭과 같이 모든 예상될 수 있는 상황과 환경에 대한 구체적이고 철저한 대비가 필요하게 된다. 이와 같이 컴퓨터 시스템 그 자체만의 수행조건은 더 이상 의미가 없고 주변의 모든 조건이 통합된 환경의 특성에 대한 정보 또한 시스템 설계 및 평가에 중요한 요소가 된다.

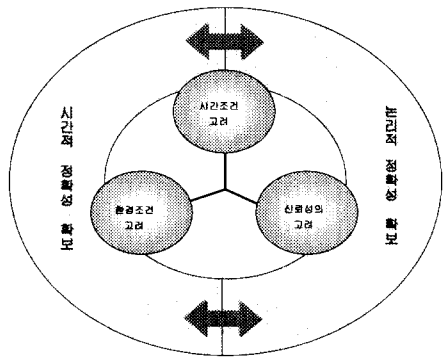


그림 1: 실시간 시스템의 정책

시간적 요소를 고려대상에 포함시키지 않는 비실시간 시스템에 비해 실시간 시스템은 단순히 기존의 비실시간 시스템의 기능에 빠른 결과를 산출하는 기법만이 첨가된 것이 아닌 정확한 결과와 정확한 시간을 어떠한 환경에서도 실행시킬 수 있는 조건을 시스템의 초기 설계부터 고려하여 전혀 다른 정책으로부터 시작해야 함을 알 수 있다. 예를 들어, 비실시간 시스템에서 중요하게 생각하는 호환성의 문제를 약간 희생시키면서 시스템의 예측성을 높이는 등 논리적 정확성과 시간의 적시성에 대한 예측성의 보장을 위해 일반적 시스템의 중요 고려요소를 희생시키는 상호교환(trade-off)적 효과를 고려해야 한다.

논문은 이러한 면을 고려하여 실시간 시스템 개발(설계 및 성능 평가)시에 고려해야 할 몇 가지 기본적인 특성들을 기술해 본다.

2. 실시간 시스템의 하드웨어

2.1 실시간 하드웨어 구조의 특징

실시간 컴퓨터는 일반적인 목적의 컴퓨터와 두 가지

중요한 다른 점이 있다. 하나는, 실시간 컴퓨터들이 그들의 응용분야에 더욱 종속적이며 특징적이라는 것이고, 둘째는 실시간 컴퓨터는 고장의 결과가 재앙적 결과를 초래할 수 있는 환경에 주로 사용된다는 것이다. 실시간 컴퓨터는, 예를 들어, 비행기의 제어와 같은 특수성을 갖는 작업에 기초하여 일반적 컴퓨터와는 다른 구조 및 특성을 갖게 된다. 다시 말해, 실시간 컴퓨터는 특별한 목적의 응용을 위해 설계된다. 따라서 일반적인 목적의 범용 컴퓨터와는 달리 비교적 사전에 응용상의 특징과 작동 환경에 대한 정보를 파악할 수가 있어서 최적의 결과를 위한 수행환경을 정확히 조절할 수 있는 점도 있지만, 고장의 결과는 일반적인 목적의 컴퓨터보다 운용되는 환경이 열악하고 비결정적이기 때문에 실시간 시스템에서 더 심각할 수 있다는 문제가 있다. 이와 같이, hard 실시간 시스템은 일반적으로 특정한 목적으로 활용되기 때문에 과거에는 여기에 적용되는 하드웨어적 구조 또한 적용될 시스템의 목적에 따라 모든 조건이 달랐다. 그러나 최근에 이루어진 VLSI 기술의 발달과 정량화된 이론의 발전으로 극도로 엄격한 실시간 요소의 추구를 목표로 하는 경우를 제외하고, 호환성과 같은 기존의 일반적 컴퓨터에서 실현할 수 있는 요소를 포함하면서도 실시간 규준을 지킬 수 있는 architecture를 사용하는 추세이다.

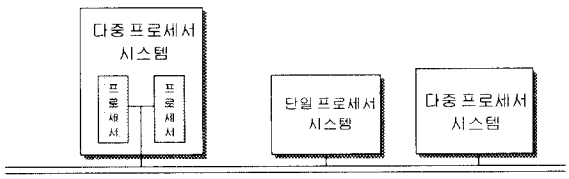


그림 2: 실시간 시스템의 네트워크 구조

전체적인 입장에서의 실시간 시스템 하드웨어는 일반적으로 복수 개의 컴퓨터가 서로 연결되어 데이터를 주고받으면서 동작하여야 하므로 네트워크의 구성이 중요한 요소로 등장하는데, 일반적 목적의 시스템 하드웨어와는 달리 (i) 다중프로세서 사이의 네트워크, (ii) 단일 프로세서 사이의 네트워크, 그리고 (iii) 다중프로세서와 단일프로세서 사이의 네트워크로 구성되어 있다. 이러한 구조는 데이터 통신의 속도를 높여줄 뿐만 아니라 신뢰성 확보도 가능하게 하며, 이에 덧붙여 고장포용(fault-tolerance)의 기법을 사용함으로써 좀더 높은 수준의 신뢰성과 예측가능성을 추구하게 된다. 고장포용은 시스템 설계 시작단계부터 시간적 제한조건을 가지고 하드웨어나 소프트웨어를 통해 고려되어야만 하고, 높은 데이터 속도와 엄격한 시간적 제약 때문에 많은 상황에서 고장포용 설계는 정적이어야만 한다는 점 때문에 현재까지 다양한 이론이 나와 있지만 일반적인 사양의 개발에 따른 상용화 시스템의 적용은 어려운 실정이다.

국지적인 입장에서의 실시간 시스템 하드웨어는 (i) 센서로부터 데이터 수집, (ii) 데이터 프로세싱, 그리고 (iii) 구동기기(actuator)나 디스플레이로의 결과 출력의 세 가지 단계의 pipeline으로 나눌 수 있다. 정확한 시간조건의 만족과 신뢰성의 확보를 위해서는 집중형 구조로는 한계가 존재하며 이를 보완하는 분산형의 구조를 갖게 된다. 즉, 각각 의 작업 수행 노드는 다중 프로세서

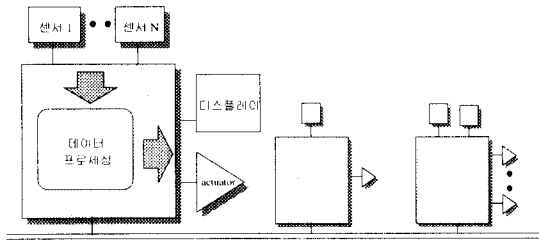


그림 3: 분산형 실시간 시스템 H/W 구조

가 되기도 하며 네트워크 프로세서와 같이 특별한 목적의 프로세서를 독립적으로 존재시키는 정책과 같이 분산형 구조가 필요하게 된다. 그림 4는 이를 간단히 보여주고 있다. 실시간 시스템의 하드웨어 구조는 각각의 작업 요소들이 높은 성능을 가지도록 보조할 수 있게 설계되어야 한다. 즉, 데이터 수집과 actuator들을 위해서 하드웨어 구조는 확장된 I/O 능력을 제공해야 하며, 데이터 프로세싱을 위해서는 고속이고, 적시성을 보장하면서 고신뢰도를 확보할 있게 작동하도록 설계 및 구현되어야 한다.

전통적인 실시간 architecture들은 특정한 목적을 위한 하드웨어와 소프트웨어에 기반을 두고 있으며, architecture도 일반적으로 응용이 달라질 때마다 병행하여 변화하였다. 하지만 그러한 하드웨어 구조는 비용 면에서나 활용도면에서도 효과적이지 못하다. VLSI 기술의 발전과 더불어 여러 방면의 실시간 응용분야에 새로운 분산형 구조의 발달이 가능하게 되었다. 이처럼 실시간 시스템을 위해 개발된 다양한 하드웨어 구조들로부터 특별한 목표와 기능의 구현을 위한 설계시의 많은 일반적인 개념과 원리들이 도출되었는데, 그 일부는 다음과 같다.

- 일반적인 off-the-shelf 요소 및 부품들로 특수한 목

적의 configuration을 개발

- 하드웨어에 맞추기 위해 문제를 바꾸지 않음
- 처음부터 고장허용(fault-tolerance) 및 실시간용 특성을 위주로 설계
- 다중버스(bus)의 필요성 이행
- 시스템의발달의 한계는 모듈이 추가될 때의 over-head 의 증가에 의존
- 열악한 주변환경에 대한 민감성 감소를 위해 중요 코드의 ROM에의 저장
- On-line testability기능 제공
- 기능상의 분할(functional partitioning)을 실행
- 응용 task들을 방해로부터 보호하기 위해 시스템 프로세서들에서 시스템 overhead를 덜어냄
- Private 메모리의 읽기전용코드과, global 메모리의 공유데이터 저장을 위한 활용
- 동적으로 load된 메모리와 캐쉬(cache)의 시간제한성하의 응용에서의 문제점발생의 어려움

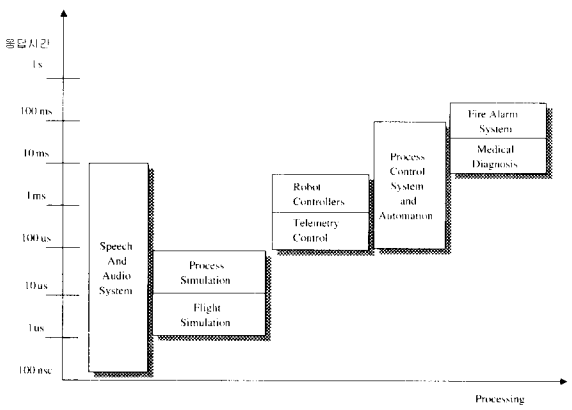


그림 4: 산업별 응답시간

또한, 위의 규칙들을 바탕으로, 일반적으로 다중프로세서로 구성된 실시간 하드웨어의 설계시에는 각 노드의 개별 프로세서들에 대한 예측 가능한 명령어 실행, 메모리 접근 및 context 스위칭 등을 통해 외부와의 상호작용, 인터럽트 조절, 실시간 작업 실행시의 예측성 및 속도가 보장되어야 하고, 노드간의 통신이나 고장포용 기법들도 실시간 분산제어기의 예측성을 보장하기 위해 중요하게 다루어지어야 한다. 이러한 하드웨어 구조의 중요한 연구분야는 interconnection topology, inter-process 통신, 그리고 operating- system과 고장포용(fault-tolerance) 기능의 지원 등의 다양한 분야가 있다.

- 프로세서와 I/O를 위한 interconnection topology : 실시간 응용에서는 높은 속도의 데이터 프로세싱과 확장된 I/O의 필요하기 때문에, 프로세서와 I/O 양쪽 모두를 위한 integrated interconnection topology를 발전시키는 것이 대단히 중요하다.
- 고장처리의 구조적 지원
- 실시간 스케쥴링의 구조적 지원
- 실시간 오퍼레이팅 시스템의 구조적 지원
- 실시간 언어의 구조적지원

2.2 요구되는 응답시간과 응용 실시간 시스템

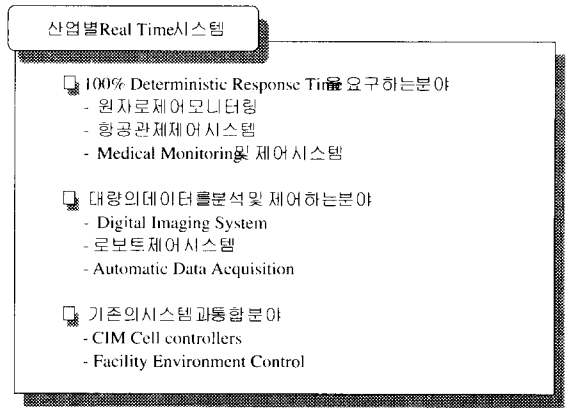


그림 5: 산업별 Real Time 시스템

실시간 시스템은 응용 산업분야에 따라서 그 특성이 달라지게 되는데 그 중에서도 요구되는 응답시간에 대한 고려를 통해 시스템의 구조를 개발할 필요가 있다. 여기서 산업별 응답시간은 단순 평균적 시간 지표가 아닌 확정적 시간응답을 의미하는 것으로 그림 6과 같다.

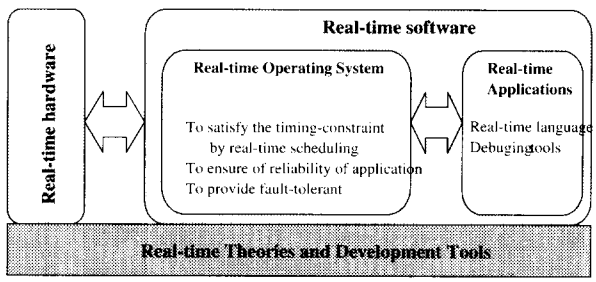


그림 6: 실시간 소프트웨어의 구조

또한, 주요 요구사항이 시간적 정확성과 논리적 정확성, 또는 일반 목적의 시스템과의 호환성등 산업별로 다르게 나타나기 때문에 이에 따른 분류가 그림 5와 같이 가능하다.

3. 실시간 시스템의 소프트웨어

실시간 시스템에 있어서 실시간적 요소를 고려하여 동작하게 하는 소프트웨어로는 크게 오퍼레이팅 시스템(platsform)과 프로그래밍 언어(개발용)를 들 수가 있다. 실시간 오퍼레이팅 시스템(RT O/S)은 컴퓨터의 핵심이며 기반으로서 기존의 비실시간 오퍼레이팅 시스템이 수행하는 프로세스 관리와 동기화, 메모리 관리, 프로세스간의 통신 및 I/O 제어 등의 기능에 시간 제한성(constraints)을 부여하고 예측성을 강화하는 측면을 갖는다. 이러한 소프트웨어적 기초인 실시간 오퍼레이팅 시스템상에서 실제의 작업을 위한 프로그래밍에 활용되는 프로그래밍 언어는, 하드웨어와의 인터페이스는 오퍼레이팅 시스템에 맞기고 기존에 정량화되어 있는 분석기법이나 도구를 이용하여 예측성을 확보하고 실시간 기준을 확보하게 된다. 따라서 하드웨어의 선정과 함께 소프트웨어의 최적화된 선택을 위해 이 두 가지의 활용과 특성을 알아볼 필요가 있다.

3.1 실시간 오퍼레이팅 시스템

실시간 시스템의 하드웨어에서 응용분야까지 모든 분야를 망라하는 통합적 성격을 갖는 것이 실시간 오퍼레이팅 시스템이다. 즉, 기존의 오퍼레이팅시스템과 같은 시스템에 대한 다양한 제어를 맡고 있지만 이들 제어를 시간적 제한 조건에 대한 고려와 예측성의 확보가 가능하게 설계되어진다. 비록 상용화나 구체적인 목표에 따라 다양하게 분류할 수 있으나 컴퓨터를 정확한 시간과 출력결과로서 운영하고자 하는 점에서는 실시간 오퍼레이팅 시스템은 공통적으로 일반적 오퍼레이팅 시스템과는 다른 다음과 같은 고려요소를 갖게 된다.

- 실시간 스케쥴링 알고리즘에 의한 시간적 제약조건 만족
- 실시간 소프트웨어에 대한 예측성
- 고장에 대한 강인성 제공

3.1.1 실시간 스케쥴링

실시간 시스템에 있어서 가장 중요한 고려요소는 시간이다. 즉, 모든 task에 주어진 중요도가 시간에 관한 함수로 나타나며 이 시간 조건이 만족되어질 수 있는 자원의 분배, task의 처리순서, CPU의 처리여부 등을 결정짓는 과정이 필요하게 되고 이를 스케쥴링(scheduking)이라고 한다. 따라서 오퍼레이팅 시스템에 적용될 실시간 스케쥴링 알고리즘의 종류와 특성이 시스템의 특성을 결정지을 수 있는 요소이다. 스케쥴링 알고리즘의 선정과정 중에 고려해야 할 문제로는 시간적 제한 조건인데 작업의 도착/발생시간, 준비시간, the worst case computation-time, deadlines 등이 있으며 시스템이 수행할 작업의 종류에 따라서 하나이상의 기준이 적절하게 정해지게 된다. 이와 같은 시간 기준에 의해 실시간 시스템의 시간 제한 조건을 만족시키고 시스템이 예측가능하고 보수유지가 쉽게 하기 위하여 잘 정의되어 있고, 이해될 수 있는 알고리즘에 의한 시스템 자원의 스케쥴링이 필요하다.

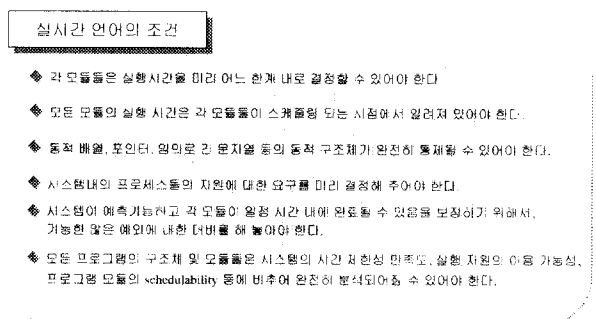
3.1.2. 실시간 커널

실시간 O/S는 기본적 실시간적 제한조건을 만족시켜야함은 물론 고장포용(fault-tolerance)과 분산시스템에서의 시간 제한조건을 고려하고 통합시켜야 한다. 또한 자원에 대한 배분, 센서처리와 통신에 대한 스케쥴, CPU, 메모리, 그리고 다양한 입출력에 대한 제어를 관장하는 광범위한 처리에 대한 책임을 지니고 있다. 이러한 작업들은 O/S가 인터럽트를 처리하고 자원에 대한 분배를 지시하며, task의 상태를 천이시키고 동기화시켜 실제 시스템이 동작을 할 수 있게 하는 통합적 성격을 띄어야 함을 나타낸다. 이 모든 일들을 처리하는 핵심적 요소로 커널을 생각할 수 있다.

3.2 실시간 프로그래밍 언어

실시간 오퍼레이팅 시스템과 컴퓨터 하드웨어에 기반하여 실제의 과업을 수행하려고 할 때, 실시간 컴퓨터 언어의 개발은 이 두 가지에 비해서 최근까지 상대적으로 그 성과가 많이 이루어져 있지는 않은 상태이다. 하드웨어적 컴퓨터 구조와 직접 연결되는 low-level의 프로그래밍은 지금까지 개발된 실시간 기법이나 도구들의 사용을 어렵게 하고, 하드웨어적 플랫폼에 변화가 있을 경우에 전체적 코드의 변화를 의미하기 때문에, 하드웨어와는 별개인 high-level 실시간 컴퓨터 언어의 개발이 절대적으로 필요하게 되었다.

실시간 시스템의 기본적 요소를 고루 갖추어야 하는 실시간 프로그래밍 언어의 최소한의 구비 조건은 다음과 같다.



이러한 기능들이 구현되는 방법에 의해 실시간 언어가 분류될 수 있는데, 최소한의 기능들을 지니며 필요한 경우 확장 가능한 Moulea-2와 같은 단순한 clean 언어가 있고, 프로그래머가 선택할 수 있는 다양한 기능을 지닌 Ada, Conic 등의 보다 복잡하며 표준화된 언어들이 있다.

4. 실시간 task의 특성 및 요구조건

실시간 시스템에서 적절한 시스템을 구현하기 위한 task 특성에 대한 파악은 곧 시스템 개발 정책과 연관성을 갖게 되므로 중요한 요소이다. 이러한 실시간 시스템과 작업들의 특성 및 제약성들 중 가장 중요한 것은 시간에 대한 제약성이고 이는 플랜트, 즉 주변환경의 특성에 의해 결정된다. 여기서는 실시간 task들이 가져야 되는 여러 가지 요소에 대하여 먼저 알아보고, 이러한 제약 조건을 갖는 실시간 task들을 여러 가지 측면에서 구분한 후 실시간 task들을 위한 가장 중요한 요소인 deadline을 유도하는 방법을 간략히 설명하여 본다.

4.1 실시간 task의 특성 및 제약

시간 제한조건(timing constraint)을 특징으로 하는 실시간 시스템에서 무엇보다도 deadline이 중요한 요소라고 할 수 있다. Deadline은 특성과 중요도에 따라서 hard deadline 및 soft deadline으로 분류되고, relative deadline에 대한 파악과 soft deadline을 가지는 task의 performability에 대한 파악이 요구된다. 또한 실시간 시스템에서의 제어 작업들에 대한 시간이나 예측가능성에 관한 특성과 제약조건 외에도 아래에 설명된 고려해야 할 다른 제약성(constraint)이 있다. (비실시간 시스템에서도 성능개선을 위한 설계 및 평가에 다음의 제한조건 정보가 필요하다.)

- 자원(resource)제약성 : 작업실행을 위해 제어기내의 프로세서(CPU), 입출력(I/O) 디바이스, 통신 네트웍, 파일, 및 데이터 베이스/구조 등의 자원이 필요
- 선행(precedence)제약성 : 하나의 복잡한 작업은 복수개의 하위 작업(subtask)들로 나누는 경우 각 하위 작업간에 순서에 주의
- 동시성(concurrency)제약 : 만약 여러 작업들이 하나의 자원을 동시에 이용하는 경우(예, 동일한 센서값을 여러 작업들이 수집)에는 일관성(consis-tency)문제에 유의
- 통신 요구 : 연산 속도 향상을 위해 분산 작업인 경우 제어기 네트웍 구조를 바탕으로 자료교환이 필요
- Dependability와 성능 제약 : 각 작업에 대해 신뢰도, availability, 및 성능인자에 대해 기준/목표치 필요

2 실시간 Task의 분류

Task의 특성을 파악하기 위해서 중요한 요소들로는 release time 및 execution time, size, deadline, 그리고 주기성 등으로 나눌 수 있다. 또한, task의 미수행시에 시스템에 주는 영향에 대해서 criticality에 대한 파악과 task 사이의 precedence 관계 및 priority 또한 중요하다.

Task의 도착시간의 예측가능성에 따라서 periodic, sporadic, 그리고 aperiodic task로 분류되고, 중요도에 따라서 critical, non-critical task로 분류할 수 있다. 이와 같은 task의 특성에 따라서 자원에 대한 요구(resource require- ment)가 다르게 되고, 또한 시스템 개발 정책도 바뀌어지게 된다. 즉, 다양한 특성을 가지는 task들을 실시간 시스템에서 수행할 경우에 다양한 정책 및 자원들을 필요로 하게 되므로, task들을 적절한 분류 및 특성 파악은 실시간 시스템에서 기본적인 작업중에 하나라고 볼 수 있다.

4.2.1 Periodic과 Aperiodic tasks

센서를 통해 매 100ms마다 그 정보를 입수하는 것과 같이 실시간 시스템에는 주기적으로 행해지는 task가 상당수 존재하게 된다. 이러한 주기적 task는 시스템 설계시에 알려져 있기 때문에 미리 스케줄링될 수 있다. 반면에 상황에 따라 때때로 발생 가능한 aperiodic task는 예측이 불가능하기 때문에 정확한 시간 내에 이러한 작업을 수행하기 위해서는 충분한 계산 능력을 미리 확보해 놓아야 할 필요가 있다. 이중 어느 한계 내의 기간 안에는 도착 가능한 task는 sporadic task라고 불린다.

4.2.2 Critical과 noncritical tasks

정확한 시간 내에 실행되지 않을 경우의 결과의 중요도에 따라 실시간 task는 critical task와 noncritical task로 나뉘어진다.

Critical task는 그 시간적 중요도가 결정적인 경우로서, 만일 deadline을 놓치게 될 경우 재앙적 결과가 일어날 수 있다. 예를 들어 비행기의 제어가 이러한 경우이다. 이러한 중요도에 불구하고, critical task가 모든 실행주기마다 주어진 deadline안에 성공적으로 실행되어야 하는 것은 아니다. Critical task를 기본적으로 필요한 속도보다 높은 빈도로 실행하는 여분을 줌으로써 그 시스템을 정상적으로 실행 가능하게 하기도 한다. 여기서의 실행주기나 deadline은 응용분야와 task의 속성, 주어진 기간에 task가 수행되는 횟수에 따라서 다르게 된다.

Noncritical task는 이름대로 결정적이지 않은 것을 의미하지만 시간에 따라 달라지는 데이터의 처리를 담당하고 있으므로 만일 deadline안에 수행되어지지 않는다면

소용없는 일이 된다. 따라서 이러한 noncritical task의 스케줄링에 있어서의 목표는 deadline 안에 성공적으로 수행될 일들의 비율을 최대화하는데 있다.

4.3 실시간 task의 데드라인 정보 유도

실시간 시스템에서의 시간 제약성, 즉 데드라인 정보는 그 중요성에도 불구하고 시스템 설계시에 미리 결정된 값으로 가정되거나, 간단한 모의 장비를 이용한 원시적 실험을 통해 부적절하게 측정되어 왔는데 이런 유도방법들은 효율적인 시스템의 설계나 정확한 시스템 신뢰도의 평가에 장애가 되어왔다. 데드라인 유도의 분석적 방법으로는 시스템의 수행작업계산 지연시간의 시스템안정성(stability)에의 영향을 분석하는 연구가 주로 수행되었다. 또한, 특수한 상황에서의 시스템 안정성을, 제한된 가정 하에서 유도하는 연구도 진행되었다. 그러나 이러한 모든 기존의 유도방법들은 현실적인 계산 가능을 위해 단순히 시스템의 지연시간이 기본시간 단위인 샘플링 주기의 한두 배일 때의 상황만을 고려하거나, 기본적으로 랜덤(stochastic)변수인 데드라인을 고정된 상수로 취급하는 비합리적인 가정 하에서만 유효하다. 물론 컴퓨터 프로그램을 이용한 시뮬레이션을 통해 항공기의 착륙 시에 제어기의 데드라인을 수치적으로 완벽하게 유도한 연구도 있으나, 이 또한 사용된 가정의 제한성 때문에 데드라인 유도의 일반적 방법이라고 보기 어렵다. 이러한 기존의 데드라인정보 유도방법들의 한계를 극복하기 위해 시스템이 적용될 응용분야의 동적 성질과 시스템 반응지연의 주된 원인인 수행작업의 계산시간 및 시스템의 내적/외적 고장 등의 발생특성(빈도 및 지속주기)이 고려되며 응용분야의 데드라인을 유도하는 회귀(recursive)법을 응용한 수치적 방법이 제안되었다. 실시간 시스템의 실제적 모델인 인공위성궤도제어, 로켓트발사궤적제어, 그리고 이동로보트충돌방지제어 등에서 단순화과정을 거쳐 각각의 데드라인을 시간 및 상태의 함수로서 구하는 예로써, 제시된 방법의 유용성이 검증되었다.

실시간 시스템에 있어 데드라인은 주어진 작업의 결과가 그 효력을 갖기 위해 반드시 지켜야하는 시간의 한계치이고, 이 값을 유도하기 위해 먼저 작업의 무효화나 비정상적인 결과에 의해 발생하는 시스템의 고장 및 극한 상황에 대한 정의가 구체화되어야 한다.

먼저 수학적으로 다루기 쉬운 형태로 시스템의 안정성(stability) 유지조건이 거론될 수 있다. 안정성은 선형불변(LTI)인 경우 시스템 전달함수나 상태방정식을 통해 유도된 고유치(eigenvalue)를 검사함으로써 간단히 안정성 유지여부를 확인할 수 있다. 이 경우에 시스템 전달함수나 상태방정식을 적절하게 변환하여 제어작업의 무효화로 인한 영향이 내포되도록 공식화하고 그로 인해 변환된 시스템의 동적 특성을 관찰하며 시스템의 안정성을 검사한다. 데드라인에 해당하는 변수 L의 값을 변화/증가시키면서, 그 결과로 얻어진 시스템 전달함수나 상태방정식의 안정성을 위의 방법으로 연속해서 검사해 안정성을 잃게 하는 최소치의 변수 $D_{\min}$가 시스템의 해당 작업에 대한 데드라인이 된다. 이를 수학적으로 공식화해 표현하면 식(1)과 같이 데드라인이 정의되며 이를 구하는 기본과정은 아래와 같이 정리된다.

$$D(X) = \sup_{U(k-N)\in U_A} \{ N : \lambda_{\max}(k, X, U(k-N)) < 1 \} \quad (1)$$

위 식에서 k, X, U, L 는 각각 시간, 시스템 상태, 입력, 데드라인을 나타내는 변수들이고 U_A는 가능한 입력 공간, 그리고 $\lambda_{\max}$는 k, X, U, L 등에 좌우되는 시스템의 최대 고유치이다. 위의 정의를 바탕으로 다음과 같은 단계를 거쳐 응용분야의 동적 특성에 좌우되는 제어 작업의 데드라인을 유도할 수 있다.

① 정상적인 제어작업시의 시스템의 상태방정식 모델링
② 데드라인 변수의 정의 및 최소시간단위로의 가정
③ 데드라인 간과시의 작업무효화내포를 위한 상태방정식 변환
④ 변환된 상태방정식에 대한 고유치 유도를 통한 안정성 검사
⑤ 안정성 손실시까지 ②부터 ④까지의 단계의 반복
⑥ 안정성을 잃게 하는 최소치의 변수값을 데드라인으로 확정

위의 과정은 해당 응용시스템에 따라 구체화될 수 있으며 만약 시스템에 대한 기본가정이 변한다면 이에 따르는 단계들도 적절하게 수정되어야한다.

5. 결론

실시간 제어 시스템은 여타의 새로운 연구분야와 마찬가지로 실험용 testbed 또는 prototype 제작을 통해 내재해 있는 가정의 유효성을 입증하면서 개발된 기법/해법과 실제 상황과의 해법 사이의 차이를 줄여 나가고, 실시간 응용 분야 자체도 검증된 state-of-the-art 결과와 수집된 아이디어를 활용해서 리엔지니어링 되어야 한다. 이러한 면에서 실시간 시스템의 기본 축이 되는 H/W, S/W 및 task에 대한 기본 특성 및 개발시의 요구 사항 및 사항들에 대해 간략하게 살펴보았다.