□특집□

실시간 제어 시스템

임 한 승† 김 학 배††

◈ 목 차 ◈

1. 서 론

제어 시스템의 일부분으로서 컴퓨터 동작을 실시간적으로 이용하려는 최초의 시도는 1950년에 Brown과 Campbell의 논문에 의해 제안되었다. 제어 및 감시를 요하는 시스템(플랜트)은 관련 산업기술의 발전에 따라 그 동작 특성이나 목표 요소가 상당히 복잡해졌고, 이로인해 정밀하고 신뢰도가 높은 시스템이 필요하게 되었다. 컴퓨터의 성능이 향상되고 가격이 저렴해지면서, 기계적 부품이나 아날로그 회로 및 필터 등으로 구현되었던 시스템 제어기는 현대 제어이론의 뒷바침으로 급격히 디지털 컴퓨터에 의해 대체되고 있다[1]. 특히 제어 시스템의 오류동작이 심각한 경제적 손실이나 인명피해 등의 회복할 수 없는 결과를 초래할 수 있는 비행기, 우주선, 핵발전소, 대규모 공정제어 및 발전소 등과 같은 hard 실시간 시스템에는 정밀한 제어 기능과 고신뢰도를 지닌 디지털 컴퓨터의 활용이 필수적이 되었다.

실시간 시스템은 작업의 논리적 결과뿐만 아니라 그 결과가 생성된 시간에따라 정확성(correctness)이 결정되는 시스템이다. 이 시스템은, 물리적인 주변환경에서 입력을 받아 제어 작업을 수행하여 출력을 내는 과정에 적시성(timeliness)이 부여된다. 즉, 데드라인(deadline)이라는 어떤 시간 한계를 넘지 않고 논리적으로 정확한 출력이 나올수 있어야 시스템 동작의 정확성이 보장되어 예기치 않은 결과를 피할 수 있게 된다. 예를 들어 궤환제어 작업인 경우에는 센서(sensor)로부터 입력신호를 수집하여 플랜트의 구동기(actuator)에 적절한 출력신호를 데드라인 이내에 제공해야만 제어작업이 유효하게 된다.

본 논문에선 이러한 실시간 시스템의 기본 특성에 대해 보다 자세하게 살펴본 후, 제어분야에 이용되는 실시간 시스템의 성질과 응용분야 등을 알아보고, 특히 컴퓨터로 작동되는 제어 시스템의 여러 특성들을 고찰한다. 실시간 시스템에서 제어 시스템 설계 및 평가시에 가장 중요하게 고려되는 정보인 시스템의 데드라인을 유도하는 방법도 살펴보고 시스템으로 부터 얻어지는 이득을 살펴보기 위한 성능측정/평가 기법에 대해서도 설명한다.

2. 실시간 제어 시스템의 개요

2.1 실시간 시스템의 특성

† 준회원 : 연세대학교 전기공학과 석사과정
†† 정회원 : 연세대학교 전기공학과 조교수

실시간 시스템이란, 논리적인 계산 결과가 정확해야 할 뿐만 아니라 그 결과가 산출되는 시간이 적시성을 지녀야 하는 시스템이다[2]. 즉, 단순히 데이터 폭주에 대해 빠른 속도로만 반응하여 문제를 해결하는 것이 아니라, 어떤 열악한 환경하에서도 정해진 시간내에 정확한 출력 결과를 산출해낼 수 있는 시스템을 의미한다. 이러한 실시간 시스템은 비실시간 시스템에 비해 다음과 같은 몇 가지 독특한 특성을 지니고 있다[3,5].

① 실시간 시스템에서 정해진 시간(deadline)안에 task가 반드시 처리되어야 한다. 즉, 연산 결과뿐 아니라 결과가 유용하게 되는 시간에 따른 기능적 정확성이 좌우된다. 이 시간의 한계를 넘어서면 수행이 퇴보(degradation)되거나 오동작(malunction)할 수 있다. 여러 제어 작업들이 데드라인을 포함한 시간 조건/정보를 바탕으로 제어기 내에서 배당되거나(assign) 시간상으로 스케쥴링되어야 한다.

② 실시간 시스템은 높은 신뢰도가 요구된다. 시스템 고장(failure)은 재산이나 인명의 막대한 손실을 일으킬 수 있으므로 높은 신뢰도(reliability)가 필수적이다. 물론 failure가 절대로 일어나지 않도록 할 수는 없지만, 가능한 한 데드라인을 어기지 않고 failure로 인한 피해를 줄여야 한다. 높은 신뢰도를 위해 고장을 포용하거나(fault-tolerent) 강인한(robust)시스템이 요구된다. 또한 failure로부터의 회복시간도 짧아야 한다. 반면에, 비실시간 시스템들에게는 시스템의 신뢰도보다는 성능(performance) 인자가 중요시된다.

③ 실시간 시스템은 동작되어야 할 환경적 요소도 고려되야 한다. 컴퓨터 시스템 그 자체만의 수행조건은 큰 의미가 없고, 주변의 모든 조건이 통합된 환경에 대한 정보가 시스템 설계 및 평가에 중요한 요소가 된다. 비실시간 시스템에서의 컴퓨터는 컴퓨터 자체의 절대적인 기능을 위주로 설계되었다. 그러나 실시간 시스템에서는 주변환경 속에서 하나의 유동적(dynamic)요소로서 시스템 전체의 성능 및 신뢰도에 영향을 미치는 특성을 고려해야 한다. 따라서 컴퓨터의 사양 선정, 설계, 및 평가 시에 주변 환경(environments, 또는 좁은 의미로 플랜트를 지칭)에 대한 명확한 정보 수집 및 분석이 선행되어야 한다. 실시간 시스템의 이러한 특성을 고려할 때 제어컴퓨터는 연산장치로서 그 자체만의(stand- alone) 성능 분석은 더 이상 의미가 없다. 시스템의 환경을 시스템 내에 하나의 기능적 요소로서 고려하여 설계하고 평가해야 한다. 이러한 면에서 실시간 시스템을 embedded system이라 부르기도 한다[4].

④ 제어기의 작업(입력수집, 연산/통신, 출력변환)수행이 예측가능(predictable)해야 한다. 즉, 시스템 내에서 어떠한 가정(예를 들어, 주어진 시간내에 가능한 시스템 요소 고장의 수가 threshold가 되는 어떤 수보다 작다는)이 만족된다면 해당 응용 작업들에 대한 모든 시간적 제한이 확실히 지켜져야 한다. 물론 예측에 대한 100% 확신을 위해서는 사전에 모든 제어 작업들에 대한 정보(갯수, 데드라인, 연산 자원, 작업간 선행 관계 등)와 작업수행 시에 발생할 수 있는 가능한 전체 환경 변화에 대한 정보들을 알고 있어야 한다. 그러나 유동적인 환경의 특성과 stochastic한 제어 작업의 속성들을 비추어 볼 때, 이것은 단순히 제어작업이 주기적으로 반복되는 경우와 같은 특수한 경우를 제외하고는 비현실적이다. 따라서 보다 현실적인 의미로 "확률적(probabilistic)" 또는 "실행중 결정적(run-time deterministic)"인 의미로 predictability를 해석할 수 있다. "확률적"이라는 것은 전체 작업중 일부분이 확실히 데드라인을 지키거나, 각각의 작업이 어떤 확률값을 갖고 데드라인을 지키는 것이다. "실행중 결정적"이라는 것은 한 작업이 시작 전에 그 때의 시스템의 부

하상황을 감안하여 작업이 데드라인내에 완료될 수 있을 지의 여부를 검사하여 작업의 시작이나 포기를 결정하는 것이다. 제어기의 설계시에는 이러한 예측이 불가능하나 시스템 운용 중에는 각 작업에 대한 예측이 동적으로 가능하므로 수시로 변화하는 비주기 작업이나 동적 부하공유(load sharing)에 유용하다.

실시간 시스템에서의 제어 작업들에 대해 이러한 시간이나 예측가능성에 관한 특성 외에도 고려해야 할 또 다른 제약성(constraint)이 있다. (비실시간 시스템에서도 성능개선을 위한 설계 및 평가에 다음의 제한조건 정보가 필요함)

- 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 하위작업간에 순서에 주의
- 동시성(concurrency)제약 : 만약 여러 작업들이 하나의 자원을 동시에 이용하는 경우(예, 동일한 센서값을 여러 작업들이 수집)에는 일관성(consistency) 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

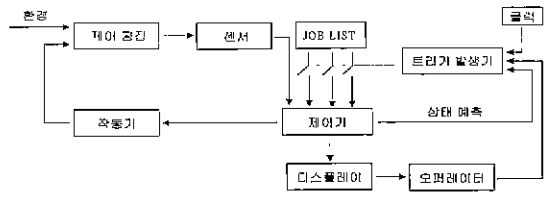
이러한 실시간 시스템과 작업들의 특성 및 제약성들 중 가장 중요한 것은 역시 시간에 대한 제약성이고 이는 플랜트, 즉 주변환경의 특성에 의해 결정된다.

2.2 실시간 제어시스템

실제로 이용되는 실시간 제어 시스템의 예를 통해 실시간 제어시스템의 특성을 살표보겠다. 그림 1은 실시간 제어시스템의 전형적인 예를 들어 보인 것이다[5]. 제어 공정에 대한 데이터를 제공하는 센서와 환경으로부터 제어 컴퓨터에 입력이 수집된다. 이 입력은 규칙적인 간격으로 제어 컴퓨터에 들어간다. 데이터가 들어오는 비율(data rate)은 낮은데, 일반적으로 초당 20워드를 넘지 않는다. JOB LIST는 제어 컴퓨터의 실행될 task나 job들의 집합으로서 모든 제어 소프트웨어는 미리 정해져 각각의 job에 분할된다.

Trigger generator는 하나 이상의 중대한 task의 실행을 시작시킨다. 이 trigger generator는 다음과 같이 세 가지로 분류된다.

- Time-generated trigger: 일정한 간격으로 발생되는데, 이에 상응하여 제어 컴퓨터 job이 일정한 간격으로 초기화된다. 제어 용어로 open-loop trigger이다
- State-generated trigger: closed-loop trigger로서 시스템이 특정한 상태에 있을 때 발생하게 된다. 주위 온도에 따라 on/off 되는 자동온도 조절장치가 그 예가 된다.
- Operator-generated trigger: 오퍼레이터는 자동 시스템을 무시할 수 있는데, trigger를 마음대로 발생시키거나 취소시킬 수가 있다.



(그림 1) 전형적인 실시간 제어시스템

제어 컴퓨터의 출력은 구동기(actuator)나 디스플레이 패널(display panel)로 들어간다. 구동기는 기계

적 장치이고 디스플레이는 인간과의 접촉이므로 데이터가 들어오는 비율은 낮다. 또한 정확한 출력이 데드라인(deadline)이라는 시간 한계값을 넘지 않아야만 시스템 동작에 정확성이 보장되어 예기치 않은 결과를 피할 수 있게 된다. 그림과 같은 궤환제어 작업인 경우 센서(sensor)로 부터 입력신호를 수집하여 플랜트의 구동기(actuator)에 적절한 출력신호를 데드라인 이내에 제공해야만 제어작업이 유효하게 된다.

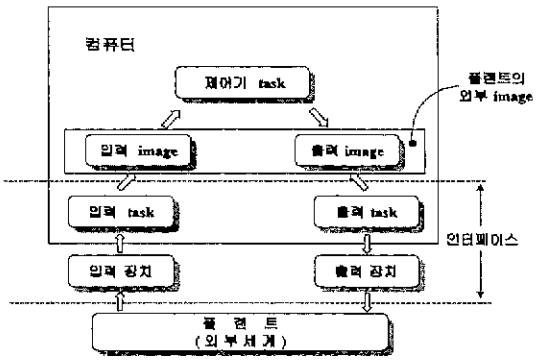
2.3 실시간 제어 시스템의 이용

제어 시스템의 일부분으로서 컴퓨터 동작을 실시간적으로 이용하는 최초의 시도는 1950년에 Brown과 Campbell의 논문에 의해 제안되었다. 이 논문에서 제안된 컴퓨터 동작에는 아날로그 연산요소가 이용되었다. 실시간 제어로서 디지탈 컴퓨터가 사용 된 곳은 항공 분야로서, 1954년엔 Digitrac이라 불리는 디지탈 컴퓨터가 자동항법과 무기 제어시스템에 이용되었다. 제어 및 감시를 요구하는 시스템은 관련 산업 기술의 발전과 더불어 그 동작 특성이나 목표 요소가 상당히 복잡해지고 있기 때문에, 기계적 부품이나 단순한 아날로그 회로 및 필터 등으로 구현되었던 시스템 제어기는 디지탈 컴퓨터에 의해 대체되고 있다. 특히 시스템의 잘못된 동작이 심각한 경제적인 손실이나 인명피해 등의 결과를 초래할 수 있는 비행기, 우주선, 핵발전소, 대규모 공정제어, 및 발전소 등과 같은 시스템에는 정밀한 제어 기능과 고 신뢰도를 지닌 디지털 컴퓨터의 활용이 필수적이 되었다. 예를 들어, 21세기에 등장할 고도의 연료절감형 비행기(fly-by-wire)는 조종사에 의한 수동 조작이나 기계적 제어기에 의존할 수 없게 되고, 빠르고 정확한 제어동작이 요구될 것으로 예상된다. 설계시에 연료의 효율성이라는 목표를 추구하다 보면 불가피하게 비행기 시스템의 동적 특성이 불안정하게 된다. 시스템의 안정성 유지 및 비상신호에 대해 효과적인 대응을 위해서는 보다 신속하고 정확하며 일관된 제어 작업이 요구된다. 물론 이러한 극단적인 예 이외에도, 많은 응용 분야에서 컴퓨터에 의한 실시간(real-time) 제어 작업이 필수화된 시스템들이 늘어나고 있다.

3. 컴퓨터 제어의 특성

3.1 컴퓨터 제어의 유용성

컴퓨터 제어는, 이것이 사용되지 않는다면 제대로 작동할 수 없는 대규모의 복잡한 공정이 필요한 곳에 사용되는 추세에서 관련기술의 발전과 경제성 등의 이유로 제어기능의 다양성 및 효율성이 요구되는 많은 곳에 그 응용분야를 확장하고 있다. 컴퓨터의 사용은 약품 제조와 같은 연속공정에 반복성(repeatability)을 증가시켜 주었다. 기존의 시스템에서는 다양한 물품을 생산하기 위한 시퀀싱 절차를 변경하기가 어려웠다. 그러나 컴퓨터 제어로 인해 적응성(flexibility)이 증가하여, 생산품의 다양화로 인한 생산과정의 빈번한 변화에도 대응할 수 있게 되었다. 또한 컴퓨터 제어로 인해 공정의 특성을 이해하는데 큰 도움을 주었다. 이로 인해 관리(supervisory) 시스템으로 하여금 작동 포인트가 원하는 방향으로 근접하도록 설비들을 작동시키는 일이 용이해졌다[1]. 또한 마이크로프로세서를 사용함으로써 아날로그 유니트를 사용 할 때 보다 훨씬 설치 및 유지 비용이 적게 들게 되었다. 이로 인해, 컴퓨터 제어를 이용한 시스템에서 컴퓨터 하드웨어의 비용 보다는 시스템 디자인이나 소프트웨어에 드는 비용이 더 크게 되었다. 따라서 시스템 디자인과 소프트웨어의 표준화가 더욱 절실히 요구되고 있다.



(그림 2) 일반화된 컴퓨터 제어

3.2 컴퓨터 제어 시스템의 분류

컴퓨터는 다양한 방법으로 제어 분야에 이용이 되고 있다. 이러한 응용들을 살펴보면 공통적인 특징들이 발견되는데 이를 공유하는 기본 제어 시스템을 다음과 같이 분류할 수 있다[1].

① batch 시스템: 생산품을 산출하기 위한 일련의 동작들이 수행되는 시스템이다. 생산품의 명세(specification)와 정확한 구성은 수행이 달라짐에 따라 변하게 된다. batch 생산의 중요한 측정요소는 set-up(charge-over)시간이다. 이것은 다음 batch 생산을 위한 장비를 준비하는데 걸리는 시간을 나타낸다. 즉 아무것도 생산되지 않는 동안 낭비된 시간이다. 작동 시간과 set-up 시간의 비율은 batch 사이즈를 알맞게 결정하는데 중요하다. NC(Numerically Controlled)기계 기구의 출현으로 set-up시간이 짧아져 batch 사이즈가 크게 줄어들었다.

② 연속 시스템(continuous system): 도중에 중단없이 긴 시간동안 생산이 유지되는 시스템을 연속 시스템이라 한다. 다른 부분의 비율이 변화하더라도 공정을 중단하지 않고 수행된다. 연속시스템 또한 경우에 따라서는 batch 모드의 생산을 할 수 있다. 공정을 연속적인 작업으로 변화하는 경우가 있는데, 전체 공정을 다 바꾸지는 않고 부분적인 공정만 연속적으로 바꾸는 것이다. 그러나 이 혼합된(hybrid mode) 시스템의 경우 특정한 batch를 항상 확인할 수 있어야 한다.

③ Laboratory 시스템: 복잡한 실험 테스트를 제어하기 위한 시스템으로서 주로 operatorinitiated 타입이다. 결과를 분석하고 필요한 수정을 계산해 내는 역할을 한다.

논리적으로 실시간 컴퓨터 시스템을 다음과 같이 두 가지 영역으로 나눌 수도 있다[6].

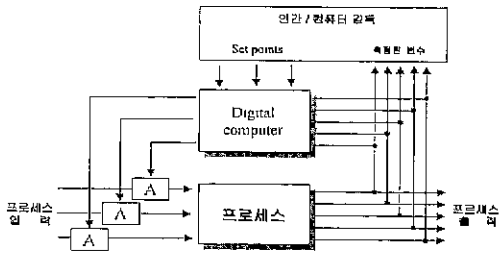
① 주변 영역 (peripheral area): 이 영역은 센서, 작동기, 디스플레이, 중심영역에 들어가도록 미리 조사하거나 포맷하는데 사용되는 공정과 관련된 요소, 그리고 중심영역으로부터 작동기 나 디스플레이로 빠져나오는 데이터들로 이루어져 있다.

② 중심 영역 (central area): 이 영역은 프로세서와 모든 high-level computing-task들이 발생하는데 연관된 하드웨어들이다. 대개 어려운 설계 문제들은 이 영역에서 발생한다.

또한 제어 방식에 따라 다음과 같은 여러 가지 분류가 가능하다[1].

① 순차 제어 (sequence control): 순차제어 시스템은 대부분의 시스템의 일부분에 적용되는데 종종 batch 시스템을 포괄한다. 그러므로 이 시스템은 batch 시스템으로 설명될 수 있다. batch 시스템은 식품 공정이나 화학 공정에서 원료를 섞고 실행하는 공정과 관계되어 있다. 일련의 공정을 실행하는데 있어 timing과 실행은 S/W에 의해 진행된다. 대규모의 화학 공장 설비에선 시퀀스가 매우 길고 크기 때문에 일부 시퀀스들은 병렬로 진행된다. 많은 시퀀스 동작들은 매우 복잡해지기 때문에 언제 시퀀스를 시작할지가 어려운 문제가 된다. 이러한 결정은 인간이나 컴퓨터에 의한 동작감시로 이루어지는데 이후에 소개될 감시제어(supervi- sory control)가 바로 그것이다.

② 루프 제어(loop/direct-digital control): 대부분의 batch 시스템은 시퀀스 제어에 연속적인 피드백제어가 더해지는데 이것이 루프제어다. 그리고 이러한 루프제어는 DDC(Direct Digital Control)에 의해 수행된다. DDC에서 컴퓨터가 피드백 루프를 구성하는데, 이는 (그림 3)과 같다. 궤환(feedback)루프에 의한 컴퓨터의 시퀀스는 신뢰도가 요구되는 중요한 구성요소에 이용된다.



(그림 3) Direct Digital Control

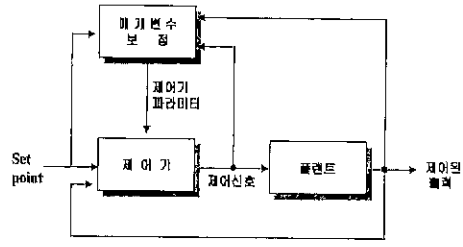
아날로그 제어에 비해 DDC가 갖는 장점은 다음과 같다. 첫째, 비용이 싸다. 둘째, DDC는 단순한 구현이 가능하기 때문에 제어기의 정확도를 높여주고 드리프트(drift)를 감소시켜준다. 셋째, mean-time-between-failure를 늘려주어 시스템의 안정성을 증가시켜준다.

③ PID 제어: PID 제어의 일반적인 알고리즘은 다음과 같다.

$$m(t) = K_p[\, e(t) + 1/T \int_0^t e(t)dt + T_{de(t)}/dt\,]$$

여기서 $e(t) = r(t) - c(t)$ 이고 , $c(t)$는 측정값이고, $r(t)$는 기준값 또는 설정값이고, $e(t)$는 에러(error)다. K_p는 전체 제어기의 이득이고, T_i는 적분시간이고 T_d는 미분시간이다. 희망하는 값과 실제 얻어지는 출력값 사이의 오차에 비례하여 제어 신호가 만들어 지도록 한다. 제어신호와 에러신호 사이의 비율은 상수값인 K_p에 의해 보정될 수 있다. K_p를 크게 설정하면 steady-state 에러가 작아지고 응답속도가 빨라지는 반면, 응답이 진동하기 때문에 많은 어플리케이션에 적당치 못하게 된다. 반대로 K_p를 작게 설정하면 응답이 느리고 정상상태 오류가 커진다. 여기에 적분 요소를 둠으로써 이러한 오류를 제거할 수 있다. 미분요소는 에러신호의 변화를 감지하여 이를 줄여주는 역할을 한다.

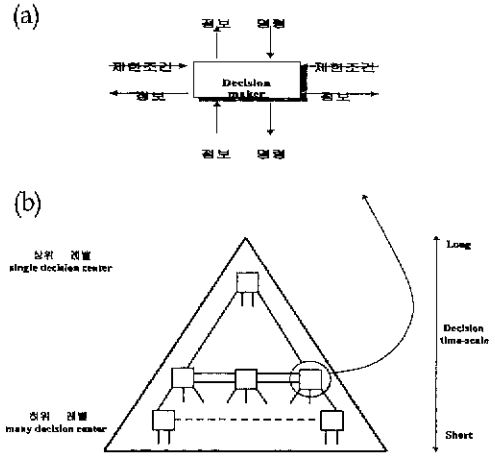
④ 적응제어 시스템 (adaptive control system): '적응제어 시스템'이란 프로세스의 동적 변화와 외부간섭의 의한 영향에 대해 반응하여 그 작용을 수정할 수 있는 제어 시스템을 말한다[7]. 적응제어 시스템은 두가지 루프(loop)를 지니고 있는데, 하나는 궤한 루프(feedback loop)이고 다른 하나는 파라미터 보정 루프(parameter adjustment loop)이다. 이 시스템을 이용하여 퍼포먼스와 기능성이 향상된 궤환제어 시스템을 설계할 수 있다. 적응제어에는 programmed 적응제어(gain scheduled control), self-tuning 적응제어, model-reference 적응제어 등 여러 종류가 있다.



(그림 4) 적응제어 시스템

⑤ 계층적 시스템(hirarchical system): 전형적인 company decision-making 구조로서 가장 자연스럽게 발전된 형태다. 다음 그림에서 보듯이 피라미드 구조로 되어있다. 각각의 decision 요소들은 상위 레벨로 부터 명령을 받아 다시 그 레벨로 정보를 되돌

려 준다. 또한 하위레벨이나 동등 레벨로 명령을 전달한다. 이러한 구조는 서로 다른 레벨의 시간 응답 요구조건들에의해 생산 프로세스의 자연적인 분할로 이루어 진다. 피라미드의 밑바닥에서는 응답 시간이 빠르고 간단한 문제들이 요구되어 진다. 프로세스가 사위 계층으로 올라 갈수록 허용된계산의 복잡성이 증대되고 응답 시간이 느려진다.



(그림 5) 계층적인 decision-making
(a) decision-making 기능
(b) decision-making 구조

⑥ 분산 시스템 (distributed system): 분산 시스템은 다음과 같은 가정에 의해 이루어진 시스템이다. 첫째, 각각의 유닛은 다른 모든 유닛들과 유사한 task들을 수행한다. 둘째, 특정한 유닛에 대한 failure나 과부하가 생길 때, 모든 또는 어떤 작업은 다른 유닛으로 전달된다. 즉, 계층적 시스템과는 달리, 작업들이 기능에 의해 특정한 컴퓨터에 할당되지 않는다. 모든 작업은 일단 분할 된 뒤, 몇몇 컴퓨터들에 할당된다. 이러한 시스템을 H/W나 S/W로 구현하는데 있어서, 컴퓨터들 사이에 task를 할당하는 것이 동적으로 이루어져야 한다는 어려움이 있다. 즉, 수행해야 할 작업에 접근할 수 있는 메카니즘이 있어야 하고, 이 작업들을 할당하도록 부하를 각각의 컴퓨터에 제시할 수 있는 메카니즘이 있어야 한다. 또 각각의 컴퓨터는 모든 정보에 접근할 필요가 있으므로 high-bandwidth data highways가 필요하다. 이 highways의 발전에 상당한 진행이 이루어지고 다양한 종류가 나타나게 되었다. 또한 multi-processor 컴퓨터 시스템에 대한 개발로 완전히 분산된 시스템도 선보이게 될 것이다.

4. 제어 작업의 데드라인 정보

실시간 시스템에서의 시간 제약성, 즉 데드라인 정보는 그 중요성에도 불구하고 제어기설계시에 미리 결정된 값으로 가정되거나 간단한 모의 장비를 이용한 원시적 실험을 통해 부적절하게 측정되어 왔다[3]. 그러나 이런 유도방법들은 제어기를 효율적으로 설계나 시스템신뢰도를 정확히 평가하는 데 장애가 되었다. 데드라인유도의 분석적 방법에 있어서, 제어기의 제어법칙계산 지연시간이 시스템안정성(stability)에 미치는 영향을 분석하는 연구가 주로 수행되었다. 또한 특수한 정적상황에서 이동로보트제어 시스템의 안정성을 유도하는 연구도 진행되었다. 그러나 이러한 모든 기존의 유도방법들은 현실적인 계산을 가능하도록 하기 위해, 컴퓨터 제어기의 지연시간이 기본 시간단위인 샘플링 주기의 한두 배일 때의 상황만을 고려하거나 랜덤(stochastic)변수인 데드라인을 고정된 상수로 취급해 버렸다. 물론, 컴퓨터 프로그램을 이용한 시뮬레이션을 통해 항공기의 착륙 시에 제어기의 데드라인을 수치적으로 완벽하게 유도한 연구도 있으나[3], 이 또한 사용된 가정의 제한성 때문에 데드라인유도의 일반적 방법이라고 보기 어렵다.

이러한 기존의 데드라인정보 유도방법의 한계

를 극복하기 위해 새로운 연구가 수행되었다. 즉, 플랜트의 동적 성질과 제어기 반응지연의 주된 원인인 제어법칙계산시간 및 제어기의 내적/외적 고장 등의 발생특성(빈도 및 지속주기)을 고려하여 플랜트의 데드라인을 유도하는 방법이 제안되었다[9,10]. 이 방법은 회귀(recursive)법을 응용하여 수치 해석적으로 접근하는 방식이다. 실시간 제어시스템의 가장 단순한 모델인 선형불변(LTI) 시스템의 궤환(feedback)제어로부터, 실제적 모델인 인공위성궤도제어, 로켓트발사궤적제어, 그리고 이동로보트 충돌 방지 제어까지, 각각의 데드라인을 시간 및 상태의 함수로서 유도함으로써, 제시된 방법의 유용성이 검증되었다.

앞에서 언급한 바와 같이 실시간시스템에 있어 데드라인은 주어진 작업의 결과가 효력을 갖기 위해 반드시 지켜야하는 시간의 한계치다. 이 값을 유도하기 위해 먼저 작업의 무효화나 비정상적인 결과에 의해 발생하는 시스템의 고장 및 극한 상황에 대해 구체적으로 정의되어야 한다[10]. 먼저, 수학적으로 다루기 쉬운 형태로서, 시스템 안정성(stability)의 유지조건을 생각할 수 있다. 선형불변(LTI)인 경우 시스템 전달함수나 상태방정식을 통해 유도된 고유치(eigenvalue)를 검사함으로써 제어플랜트의 안정성 유지여부를 확인할 수 있다. 이 경우에 시스템 전달함수나 상태방정식을 적절하게 변환하여 제어작업의 무효화로 인한 영향을 고려하여 공식화하고, 그로 인해 변환된 시스템의 동적 특성을 관찰하여 시스템의 안정성을 검사한다. 데드라인에 해당하는 변수 D의 값을 변화/증가시키면서 --- 물론, 이 경우 변화된 데드라인에 따라 제어작업의 무효화정도도 변화하게 된다 --- 결과적으로 얻어진 시스템 전달함수나 상태방정식의 안정성을 위의 방법으로 연속해서 검사하여 안정성을 잃게 하는 최소치의 변수 $D_{\min}$를 구한다. 여기서 구해진 $D_{\min}$가 시스템의 작업에 대한 데드라인이 된다. 이를 수학적으로 공식화해 표현하면 아래 식과 같이 데드라인이 정의된다.

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

여기서 k, X, U, D 는 각각 시간, 시스템 상태, 입력, 데드라인을 나타내는 변수들이고 U_A는 가능한 입력 공간, 그리고 $\lambda_{\max}$는 k, X, U, D 등에 좌우되는 시스템의 최대 고유치이다. 위의 정의를 바탕으로 다음과 같은 단계를 거쳐 플랜트의 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 안정성손실시까지 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 수정되어야 한다. 시스템 안정성의 손실 못지 않게, 시스템의 허용공간 이탈이 데드라인유도를 위한 시스템의 고장 또는 극한상황으로 정의될 때가 있다. 예를 들어, 착륙하는 항공기의 피치(pitch)각 등의 중요 각도들이 허용범위 내에 머물러야 지면과의 충돌을 피할 수 있으며, 자동화된 공장의 물품이동을 위한 로봇은 산재해있는 장애물들을 피하기 위해서는 역시 허용공간을 항상 유지해야 한다.

$$D(X(k_0)) = \sup_{U(k-N)\in U_A} \{ N \mid \Phi(k, k_0, X(k_0), U(k-N)) \in X_A \}$$

위 식에서 X_A는 시스템상태의 허용공간으로 정의되고 시간 k_0에서 출발한 시스템의 궤적은 지연시간이 N일 때 아래 식에 의해 전개된다.

$$X(k_0) = \Phi(k, k_0, X(k_0), U(k-N))$$

이 경우의 데드라인은 같은 방법에 의해 위 식(D(X))과 같이 정의된다. 앞에서의 유도과정에서 네 번째 단계(④)의 안정성검사 대신 '허용공간 이탈여부'가 매 주기당 검사되는 것 이외는 같은 절차 및 방식이 적용된다. 보다 구체적으로 이 경우에 매 주기당의 시스템의 궤적을 주어진 동적 특성 방정식으로부터 계산하고 이의 허용공간으로부터의 이탈여부를 각 주기에 대해 조사하는 방법이 가능하다. 그러나, 경우에 따라 허용공간이 쉽게 얻어질 수도 있지만 대부분의 경우 해석적인 방법에 의한 유도는 불가능하다. 따라서 허용공간의 공식적인 유도절차 또한 관심을 끄는 연구과제가 되고 있다.

5. 퍼포먼스 측정(Performance measure)

실시간 시스템이 점점 복잡해짐에 따라 시스템을 안전하게 제어하는 문제가 점점 어려워지고 있다. 항공기, 핵발전 등 time-critical 시스템의 컴퓨터 제어에 있어서 엄중한 성능에 대한 요구조건들이 중요하게 된다. 이러한 조건들은 적절한 퍼포먼스 측정 및 평가 없이는 충분히 설명될 수 없다. 이러한 관점에서 실시간 제어 컴퓨터의 퍼포먼스 측정은 다음 두가지를 만족해야 한다[6].

① 시스템으로 부터 얻어지는 이득을 나타내야 한다.

② 이 이득을 얻기위해 소비되는 비용을 나타내야 한다.

이득과 비용이 동시에 표현되어야 순수한 이득이 계산될 수 있다. 그러나 아직까지는 컴퓨터 특성을 완벽하게 한가지로 나타내줄 수 있는 방법이 없다. 모든 측정 방법들은 시스템의 어느 영역에 대한 성능을 부분적으로 평가해주는 데 불과하다. 그러므로 컴퓨터의 다양한 특성에 대해 부여되는 성능인자의 선택이 중요하다. 이러한 퍼포먼스의 측정은 다음을 만족시켜야 한다.

- 과련된 정보에 대한 효과적인 encoding을 나타내야 한다.
- 주어진 어플리케이션에서 후보 제어기의 우선순위에 대한 객관적인 근거가 있어야 한다.
- 설계에 대한 객관적인 최적화 영역이 있어야 한다.
- 검증할수 있는 사실을 나타내야 한다.

컴퓨터는 응용분야의 배경(context)을 충분히 고려해서 설계, 평가 및 선정되어야 한다. 이러한 배경 및 환경에 대한 심각한 고려없이는 모든 성능평가가 무의미해 진다. 설계자는 응용분야의 요구조건들을 형식화/공식화하여 나타낼 수 있는 정보를 필요로 한다. 이러한 컴퓨터 시스템의 측정 가능한 특성들은 스칼라 함수로서 나타내지는데, 일반적으로는 다음과 같은 세가지 방법으로 측정할 수 있다[6].

① 선형 결합(linear conbination): 컴퓨터의 측정가능한 특성들의 선형 결합으로 스칼라 함수가 나타난다. $a_1, a_2, \ldots, a_N$를 제어 공정의 상대적 또는 절대적 중요성을 나타내는 가중치(weights)라 하고, 이 가중치들이 시스템의 속성(attribute)인 $x_1, x_2, \ldots, x_N$에 할당되었다고 하자. 그렇다면 전체 퍼포먼스는 $\sum_{i=0}^{N} a_i x_i$ 이 된다. 여기에서 각각의 가중치 자체를 객관적으로 얻기가 어렵지만, 중요도에 따라 속성을 순위매기면(prioritization) 객관성이 어느정도 확보된다. 더군다나 속성들을 하부속성(sub-attribute)들로 이루어 진 것으로 생각할 수 있다. 따라서 상황에 따라 하부 속성들을 순위매길 수 있다. 여기에서 순위매겨진 아이

템들은 가장 우선권이 가장 낮다. 각각 속성의 weight는 하부 속성들과 연관된 숫자의 평균값이 된다.

② 퍼포머빌러티(performability): payoff는 컴퓨터 효율성을 측정하는데 있어서 유일한 방법이다. 즉 특정한 응용과 연관되어 A의 payoff가 B의 payoff 보다 크다면, 컴퓨터 A가 컴퓨터 B보다 더 우수하다는 것이다. payoff는 사용자의 입장에서 보면 응용 시스템이 얼마나 잘 수행되는가를 나타내 준다. 사용자를 하나 이상의 등급으로 나눈다면 이에 대응하는 하나 이상의 값이 나온다. 사용자의 입장은 수행 레벨(accomplish level)을 부여한다. 응용 시스템에서 제어된 공정의 performability는 다음과 같이 벡터함수로 표현된다.

$$p_{s(A)} = [(p_s(a_1), \ldots, p_s(a_N)],$$

($p_s(a)$ = 수행 레벨 $a \in A$ 에서 응용시스템이 수행될 확률)

이제 컴퓨터의 특성을 응용 시스템의 특성에 연결시키면 된다. 이것은 컴퓨터 시스템의 Markov 모델을 만들어 각각 상태의 궤도를 수행 레벨과 연관시키면 된다. 예를 들어 다음 수행 집합은 항공기-제어 컴퓨터의 performance 레벨을 정의하기 위한 것이다.

- a_0 : 경제적 penalty 없음, 작동 penalty 없음, 임무 profile 변화 없음, 재난 없음
- a_1 : 경제적 penalty, 임무 profile 변화 없음, 재난 없음
- a_3 : 임무 profile 변화 , 재난 없음
- a_4 : 재난

다음으로, 컴퓨터 시스템의 performability는 각각의 수행레벨에서 항공기가 성공적으로 mission flight를 수행하는 확률이 된다.

③ 비용함수(cost function): 비용함수에 대한 접근 방식은, 여러가지 computational task에 대한 컴퓨터의 응답시간(response time)에 초점을 맞추게 된다. 응답시간은 task가 시작되는 시간과 결과가 작동기나 디스플레이로 내보내는 시간과의 간격이다. 제어 컴퓨터는 궤환루프로 되어있으므로 응답시간은 궤환지연(feedback delay)을 유발한다. 궤환 지연이 증가함에 따라 제어 시스템의 안정도가 떨어진다. 특정한 응답시간을 넘어서면 불안정성에 들어서게 되어 큰 사고가 일어날 수도 있다[8]. 불안정성에 들어서지 않는다 하더라도 performance 등급이 낮아진다. 또한 컴퓨터의 응답시간은 컴퓨터 architecture, 오퍼레이팅 시스템, failure handling 메카니즘, 소프트웨어 등에 의존하기 때문에, 응답 시간의 함수를 컴퓨터 구조를 조정하기 위한 최적의 영역으로 사용하는 것이 가능하다. 응답시간 ξ과 관련된 컴퓨터 task i의 비용은 다음과 같이 생각할 수 있다.

$$g_i(x, \xi) = \Omega_i(x, \xi) - \Omega_i(x, 0)$$

여기서 $\Omega_i(x, \eta)$는 task의 응답시간이 η이고 시스템의 state가 x라고 할 때 task i의 퍼포먼스 인덱스의 contribution이 된다. 즉, $g_i(x, \eta)$는 제어컴퓨터의 시간지연 ξ에 의한 성능인자의 증가부분이다. 만일 ξ가 매우 커서 심각한 failure가 발생한다면, 이에 따른 비용은 매우 크다고 가정한다. 만일 개개의 task에 대한 contribution을 분리할 수 없다면, joint 비용함수로서 나타내면 된다. 이제 비용함수와 관련된 측정이 가능해진다. 예를 들어, dynamic failure(p_{dyn})가 발생할 확률은 컴퓨터의 제어공정이 치명적으로 실패할 확률이 된다. 비용의 기대치는 임무 수행 중 failure가 발생하지 않는 기간동안 발생한 평균 비용이 된다.

6. 결 론

지금까지 실시간 제어시스템에 대해 기본 개

념과 원리, 그리고 실제 응용분야까지 다양하게 살펴 보았다. 실시간 제어시스템은 기존의 제어 시스템이 갖는 한계점들을 극복하며 계속 새로운 가능성을 제시해 주고 있다. 또한 디지털 컴퓨터가 널리 사용됨에 따라, 실시간 제어 시스템에 대한 연구는 중요한 학문 및 실용기술 개발 분야로 자리잡고 있다. 이것은 플랜트의 유동적 특성에 대한 정보/지식 습득을 위한 제어 공학뿐만 아니라, 최적의 제어 컴퓨터 설계 및 평가를 위한 컴퓨터 공학에까지 중요한 분야로 여겨지고 있다. 그러나 이 분야의 연구에 문제가 전혀 없는 것은 아니다. 이러한 기본적인 연구 이론이 해석학적인 방법과 컴퓨터 시뮬레이션을 통해 증명되고 검증되어 왔으나, 실제 응용 분야에 적용하여 검증된 것은 많지 않다. 이 시스템의 연구는, 여타의 새로운 연구분야와 마찬가지로, 실험용 testbed 또는 prototype 제작을 통해 내재해 있는 가정의 유효성을 입증하면서, 개발된 기법과 실제 상황의 해법 사이의 차이를 줄여 나가야 한다. 또한 실시간 제어의 응용 분야 자체도 검증된 결과와 수집된 아이디어를 활용해서 리엔지니어링 되어야 한다.

참고문헌

[1] S. Bennett, "Real-Time Computer Control", 2nd ed. Prentice Hall, 1994

[2] J. Stankovic, "Misconceptions about real-time: A serious problem for next-generation systems", *IEEE Comput,*. pp.10-19, Oct. 1988

[3] K. Shin, C. Krishna, and Y. Lee, "A unified method fro evaluating real-time computer controller and its application," *IEEE Trans. on Automat. Contr.,* vol. AC-30, no.4, pp.357-366, Apr. 1985

[4] W. Halang, "Contemporary Computers Considered Inappropriate for Real-time Control", *IFAC Algorithms and Architectures Real-time Conttol*, Seoul, Korea, 1992

[5] K. Shin and P. Ramanathan, "Real-time Computing: A new Discipline of Computer Science and Engineering", *Proceedings of the IEEE*, no.1, vol.82, pp.6-24, Jan. 1994

[6] C. Krishna and K. Shin, "Performance Measure for Control Computers", *IEEE Trans. on Automat. Contr.,* vol.AC-32, pp.467-473 , June. 1987

[7] K. Astrom and B.Wittenmark, *Adaptive Control*, 2nd ed. Addison-Wesley publishing Company, 1994

[8] C. Krichna "Processor Tradeoffs in Distributed Real-Time System", *IEEE Trans. on computers*, vol.C-36, no4, pp.1030-1040, Sep. 1987

[9] H. Kim and K. Shin "On the Maximum Feedback Delay in a Linear/Nonlinear Control Systems with Input Disturbances Caused by Controller Computer Failures", *IEEE Trans. on Control Systems Technology*, vol.2, no.2, pp.110-122, June 1994

[10] K. Shin and H. Kim "Derivation and Application of Hard Deadlines for Real-Time Control Systems," *IEEE Trans. on Sys., Man., and Cybernetics*, vol.22, no.6, pp.1403-1413, November/December 1992

임 한 승

1998년 연세대학교 전기공학과 (공학사)
1998년-현재 연세대학교 전기공학과 석사과정
관심분야 : 실시간 생산제조 시스템, fault-tolerance 이론

김 학 배

1988년 서울대학교 전자공학과 (공학사)
1990년 미시간대학(The Univ. of Michigan) EECS (공학석사)
1994년 미시간대학(The Univ. of Michigan) EECS (공학박사)
1994년-1996년 NASA Langley 연구센터 미국립 NRC 계약연구원
1996년-현재 연세대학교 전기공학과 조교수
관심분야 : 실시간 시스템, fault-tolerant 시스템, 생산제조시스템 자동화, 신뢰도 평가기법