분산 실시간 제어 시스템의 개발

홍성수
서울대학교 전기공학부

1. 서론

최근 고성능의 마이크로 프로세서가 출시함에 따라 대부분의 실시간 제어 시스템은 디지털 제어 시스템으로 대체되었다. 컴퓨터를 이용한 시스템은 유연성(ﬂexibility)과 적응성(responsiveness)이 뛰어난 데다가 도로 복잡한 시스템을 효과적으로 제어할 수 있기 때문이다. 특히 디지털 제어 시스템은 주로 항공, 방위, 산업, 공장 자동화 등 시간적 제약이 중요시되는 분야에서 필수적인 역할을 하고 있다. 이와 같은 시스템을 실시간 제어 시스템이라 하는데, 일반적으로 기능적 제약뿐만 아니라 동시에 시간적 제약을 만족시키는 시스템을 말한다. 다음은 시간적 제약의 몇 가지 예를 보여주고 있다.

* 항공기의 운항 좌표계는 최소한 50ms안에 한번씩 갱신되어야 한다.

* 핵발전소의 방사능 유출은 200ms 안에 감지되어야 하고 차단되어야 한다.

위와 같은 시간적 제약을 만족시키지 못할 경우에는 많은 인명과 재산의 피해를 초래할 수 있으므로 실시간 성의 만족은 필수적이라 할 수 있다. 현재 실시간 시스템 분야에서는 실시간 스케줄링, 실시간 운영체제, 실시간 시스템 구조 이론, 실시간 통신 등의 다양한 분야에서 활발한 연구가 진행되고 있다. 이렇게 세분화된 이론 및 기술들은 체계적으로 통합되어, 실시간 시스템을 설계하고 구현하는데 이용된다. 하지만 이러한 연구 성과가 분산 실시간 제어 시스템을 개발하는데 아직 적용되지 못하고 있는 실정이다. 이는 제어 이론 분야와 실시간 이론 분야의 학계간의 연계가 부족한 덕이다. 실시간 분야에서는 일반적인 실시간 컴퓨터 시스템만을 대상으로 이론이나 기술이 개발되어 제어 시스템을 동작할 수 있지만, 분산 실시간 제어 시스템을 개발하려는 사람들은 주로 제어 이론을 전공한 사람들되어 경험적인 방법에 의존하여 실시간 제어 시스템을 개발하고 있다. 그 결과로 상당수의 분산 실시간 제어 시스템은 기능적 성능의 극대화에만 치중하여 시스템의 시간적 특성을 제대로 고려하지 못하고 있다. 이로 인해 주어진 자원(resources, CPU 혹은 communications bandwidth)을 최적으로 활용하지 못하거나, 심지어는 시간적 제약을 충족시키지 못하기도 한다.

본고에서는 이러한 문제점을 인식하고 실시간 이론을 분산 실시간 제어 시스템에 적용하고자 한다. 단순히 실시간 시스템을 개발하기 위한 설계 이론, 실시간 시스템 소프트웨어의 핵심인 실시간 운영체제, 실시간 통신 및 클록 동기화 등을 통해서, 실시간 제어 시스템을 개발하고자 하는 이들에게 소개하고자 한다.

![그림 1. 분산 실시간 시스템 설계 방법 (PCM)](image)

2. 분산 실시간 시스템 설계 이론

일반적으로 실시간 시스템은 구조적 기법에 따라 설계되는데, 하향식(top down)으로 시스템을 모듈화하여 체계적으로 구축하는 것을 말한다. 실제로 미국 국방성 에서도 이를 표준화하여 사용하고 있다 (DOD-STD-2167). 이 방법은 프로세스(process)를 기반으로 시스템을 계층적으로 분할하여, 프로세스간의 통신 및 독립성을 두고 고려한다. 이를 발전시킨 것으로 DARTS(Design Approach for Real-Time Systems)가 있는데, 이는 General Electric사에서 개발된 기법으로서 데이터 흐름에...
2.2. 태스크의 시간적 속성 유도

일반적으로 디지털 제어 시스템을 구축하는 방법은 크게 두 가지로 나눌 수 있다. 하나는 연속 시간장에서 모델링한 후 이를 컴퓨터 상에서 이산화 하여 구현하는 방법으로서 (1) 'continuous time design and then digitization'으로 불리기도 한다. 또 다른 방법으로는 이산 시간장에서 직접 설계 및 구현하는 (2) 'direct digital design'이다. 이 두 가지 방법은 디지털 제어 시스템 설계의 전형적인 방법들이지만 이러한 방법으로는 시스템의 한정된 자원(CPU 혹은 network)을 제대로 활용하기 힘들다. 예를 들어, 제어 루프의 샘플링 주기에(sampling period)는 그 성능 요구 조건에 따라 시간 동안, 외란, 패 라미터에 대한 민감성 등을 고려하여 경험적인 방법으로 결정된다. 하지만 샘플링 주기는 제어신호를 활용 하는 정도를 나타내기 때문에 시스템의 구현과 밀접한 관계를 갖는다. 따라서 시스템의 성능은 물론 가능할 수 있는 자원을 동시에 고려할 수 있어야만 시스템 개발을 최적화할 수 있다. 

2.1. 시스템 모델

본 논의 시간적 제어 시스템은 생산자/소비자(procer/consumer) 모델을 기초로 한다. 그림 2는 입력 데이터가 각 태스크들을 거쳐 최종적인 출력을 생성하기까지의 과정을 간략히 보여주고 있다. X, Y는 각각 입력과 출력 데이터, P는 태스크, π는 태스크간의 통신을 위해 사용되는 통신 포트를 의미한다. 그림 2의 모델은 태스크의 실행 관계, 태스크를 각 프로세서에 할당한 정보, 데이터의 호출 등, 시스템의 기능적 특성을 잘 기술해 준다. 한편 시스템의 시간적 특성은 양극단 시간 제약으로 기술된다. 일반적으로 양극단 시간 제약은 최대 유용한용 시간(Maximum Allowable Validity Time)과 최대 출력 주기(Maximum Loop Processing Period)의 형태로 주어진다.

* 최대 유용한용 시간 - 입력으로부터 출력이 생성되기까지 걸리는 최대 시간값을 의미한다. 이를 간략히 MAVT(Maximum Allowable Validity Time)로 나타낸다.
* 최대 출력 주기 - 출력이 생성되는 주기의 최대값을 의미한다. MLPF(Maximum Loop Processing Period)로 나타낸다.

그림 2. 분산 실시간 시스템 모델

2.2. 태스크의 시간적 속성 유도

일반적으로 디지털 제어 시스템을 구축하는 방법은 크게 두 가지로 나눌 수 있다. 하나는 연속 시간장에서 모델링한 후 이를 컴퓨터 상에서 이산화 하여 구현하는 방법으로서 (1) 'continuous time design and then digitization'으로 불리기도 한다. 또 다른 방법으로는 이산 시간장에서 직접 설계 및 구현하는 (2) 'direct digital design'이다. 이 두 가지 방법은 디지털 제어 시스템 설계의 전형적인 방법들이지만 이러한 방법으로는 시스템의 한정된 자원(CPU 혹은 network)을 제대로 활용하기 힘들다. 예를 들어, 제어 루프의 샘플링 주기에(sampling period)는 그 성능 요구 조건에 따라 시간 동안, 외란, 패 라미터에 대한 민감성 등을 고려하여 경험적인 방법으로 결정된다. 하지만 샘플링 주기는 제어신호를 활용하는 정도를 나타내기 때문에 시스템의 구현과 밀접한 관계를 갖는다. 따라서 시스템의 성능은 물론 가능할 수 있는 자원을 동시에 고려할 수 있어야만 시스템 개발을 최적화할 수 있다. PC의 이론을 적용하기 위해 다음과 같은 두 가지 가정을 세운다.

* 제어 알고리즘은 'continuous-time design and then digitization'의 방법에 의해 고안된다.
고안된 알고리즘은 주어진 목적합수(objective function)를 만족시키는 최적 알고리즘이다.

위와 같은 방식 하에서 생성령 주파수는 별도적인 시스템의 역학을 충분히 반영할 수 있기 위해 보통 closed-loop bandwidth의 몇 배 이상이 되어야 한다. 실제 수치는 주로 경험적으로 결정되는데, 예를 들어 외란을 고려하고 한다면 약 20배 이상이 되어야 한다. 따라서 이는 생성령 주기의 최대값을 제한하는데 MLPP 제약 값으로 볼 수 있다. 한편, 임의의 센서값으로부터 최종 제어 출력값은 오늘날 기준의 웅덩이식 또한 최대 값을 갖는다. 이는 MAVT로서 표현하는데 이 제약을 만족시키기 못하면 시스템의 안정성(stability) 등의 문제 가 생긴다. 이는 Shin 등에 의해 분석된 바 있는데, [12]에서는 시스템의 안정성을 유지하기 위한 종료시점을 유도하기도 하였다.


3. 실시간 운영체제

앞절에서 소개된 테스크는 제어 알고리즘을 실제 수행 가능한 코드로 구현한 것으로 이를 수행시키기 위해서 적합한 실시간 운영 환경이 필요하다. 이에 따라 오늘날에는 실시간 특성을 요구하는 분산 제어 시스템의 개발을 위해 사용하는 실시간 운영체제가 많이 이용한다. 이 점에서는 기존의 범용 운영체제에 대한 지식을 전제로 실시간 운영체제가 갖추어야 할 특성을 알아보고 현재 많이 사용되고 있는 실시간 운영체제들을 소개한다.

앞서, 실시간 운영체제는 선택 가능한(preemptible) 멀티쓰레드(multithread)를 지원할 수 있어야 한다. 멀티쓰레드는 시스템의 병렬성(concurrency)을 지원할 뿐만 아니라 자원의 효율적 관리를 가능하게 하며, 병목 교환(context switching) 시간을 줄여 시스템의 응답성을 향상시켜준다. 이러한 쓰레드는 구현 방식에 따라, 쓰레드가 커널의 기능 자체에 구현된 커널 레벨 쓰레드(kernel-level thread)와 유저의 라이브러리 함수(library functions)로 구현된 유저 레벨 쓰레드(user-level thread)로 구분된다. 커널 레벨 쓰레드는 쓰레드 스케줄링이나 시그널링(signaling)을 단순화하는 장점이 있으나 오버헤드(overhead)가 크다는 단점이 있다. 반면에 유저 레벨 쓰레드는 쓰레드 스케줄링을 어렵게 하지만 오버헤드가 적다는 장점이 있다. 이들의 장단점을 보완하기 위해 최근에는 이 두를 혼합한 형태의 쓰레드 모델이 제안되고 있다. 멀티 쓰레드 모델은 현재 실시간 운영체제에 어떤 한계도 제한된 것이 아니라 성능 향상을 위해 기존의 많은 범용 운영체제에서도 도입되었다. 산마이크로시스템(Sun Microsystems)의 솔라리스(Solaris), 마이크로소프트(Microsoft)의 윈도우스(Windows) NT, Mach 등에서도 멀티쓰레드를 지원하도록 개발되었다.

둘째, 실시간 운영체제는 예측 가능한 실시간 스케줄링과 동기화(synchronization) 메커니즘을 제공하여야 한다. 실시간 스케줄링은 실시간 분야에서 그 동안 가장 활발히 연구되어온 분야로서 현재 많은 이론들이 제안되었다. 특히 Liu와 Layland 제안한 우선순위 기반(priority-based) 알고리즘인 RMA(rate monotonic assignment)와 EDF(earliest deadline first) 알고리즘은 각각 정적 우선순위(fixed priority)와 동적 우선순위(dynamic priority) 알고리즘 중 최적 알고리즘으로 알려져 있다. 이들은 현재 가장 많이 사용되며 하는 알고리즘으로서 실시간 운영체제는 이러한 실시간 스케줄링을 지원하여 시스템의 안정을 효과적으로 관리하면서 주어진 시간 제약을 만족시킬 수 있다. 또한 실시간 운영체제는 우선순위 계층 프로토타입(priority inheritance protocol) [13]을 지원할 수 있어야 한다. 이는 공유 자원의 사용에 있어서 블로킹 시간(blocking time)을 제한하여 쓰레드 동기화를 예측 가능하게 한다.

마지막으로, 운영체제의 시간적 특성을 명확히 파악할 수 있어야 한다. 예를 들면, 인터럽트가 발생한 시점에서부터 테스크가 수행을 재개할 때까지 걸리는 인터럽트 결여기간(interrupt latency)을 알 수 있어야 한다. 따라서 사용 프로그램의 수행이 예측 가능하며 스케줄링 분석이 용이해진다. 이 밖에 시스템 콜(system call)의 최대 시간이나 인터럽트가 마스크되는 시간 등을 알 수 있어야 한다.

위와 같은 기준에 따르면 Windows NT 같은 기존의
범용 운영체제는 실시간 운영체제가 될 수 없다. 우선 Windows NT가 멀티 쓰레드를 지원하지만 충분한 우선 순위(priority)를 제공하지 못하여 실시간 스케줄링에 적합하지 않다. 또한 우선순위 계층 프로토콜(priority inheritance protocol)을 제공하지 못하며, 운영체제의 시간적 특성이 명확히 파악되지 않는다. 예를 들면, Windows NT는 우수한 하드웨어 인터페이스를 제공하지만 Pen- tium Power Management 인터럽트가 예측할 수 없는 시간만큼 시스템을 정지(preempt)한다. 따라서 응용 프로그램의 시간적 분석을 어렵게 함은 물론 신뢰할 수 있는 시스템을 개발할 수 없다. 마지막으로 Windows NT는 programable timer를 제공하지 않아 실시간 클록(clock)을 사용할 수 없다는 기본적인 문제를 갖고 있다.

현재 널리 사용되는 실시간 운영체제로는 CHORUS/OS, IRIX, LynxOS, OS-9, p-SOS, QNX, RT-mach, SORIX 386/486, VRTX, VxWorks 등이 있으며 실시간 운영체제에 대한 표준으로는 현재 POSIX 1003.1b가 나와 있다. 한편 국내에서는 최근 서울대학교 실시간 운영체제 연구실에서 실시간 커널인 ARX가 개발되었다. ARX는 기존의 유저 레벨 쓰레드가 갖는 장점을 유지하면서 그 문제점에 대한 명확하고 효율적인 방법을 제공한다. 처음, 커널에서 보호(block)된 쓰레드가 프로세스 전체를 보호할 수 있도록 동적 스택 바인딩(dynamic stack binding) 기법을 제공한다. 둘째, 응용 프로그램이 원하는 방식에 따라 쓰레드를 스케줄링할 수 있도록 커널의 이벤트를 효율적으로 유지 스케줄러에 전달하기 위한 방 법으로 스케줄링 이벤트 upcall 기법을 제공한다. upcall로 의해 발생하는 오버헤드를 줄이기 위해 커널은 유지 스케줄러를 lock-free하게 작성할 수 있는 방법과 기존 모델을 제공한다.

4. 실시간 통신 및 클록 동기화

4.1 실시간 통신의 특성

분산 실시간 시스템의 개발은 기본적으로 공유 메모리가 없다는 문제점을 고용하고 있다. 이에 따라 분산된 각 노드들은 서로 데이터를 주고받는데, 이러한 메시지 전송 또한 대.tx와 복잡해지며 실시간 특성을 갖추어야 한다. 메시지 전송은 선택한 통신 프로토콜에 따라 시간적 특성을 달리 하는데 실시간 통신 프로토콜은 일반 프로토콜과 특허 구분되어야 할 특성은 다음과 같다. 첫째, 최대 전송시간(maximum protocol latency)과 지터(lateny jitter)가 작아야 한다. 일반적인 의미에서는 평균 전송 시간이 작을수록 좋은 통신 프로토콜이 된다. 하지만 실시간 시스템에서는 최악의 상황을 전체로 분석 및 개발이 이루어지기 때문에 최대 전송시간이 작을수록 좋다. 또한 전송시간의 변화폭을 지키려 하면 이 또한 작을수록 좋다. 지터는 센서 입력이나 제어 출력의 전송을 비정규격(average)으로 만들며 실제 제어 성능에도 영향을 끼치는 요소이다. [14]. ATM (asynchronous transfer mode)는 이러한 지터를 최소화하기 위해 광대역 통신망에서 개발되어 널리 보급중이다.

둘째, 실시간 통신은 신뢰성을(reliability)을 가져야 한다. 대개의 제어 시스템의 경우, 암호화 된 환경에서 적용되므로 통신망 자체가 안전하게 고장이 쉽게 나지 않아야 한다. 또한 여러 가지 오류에도 감내할 수 있어야 시스템의 신뢰성을 얻을 수 있다. 예를 들어 메시지가 손실되며 감지(detected)되며 재전송(retransmission) 되어야 한다. 통신상에서 감지될 수 있는 오류는 메시지가 손실되는 전송 오류와 임의의 노드(node)가 고장나거나 통신에 참여하지 않은 노드 오류가 있다. 이러한 오류들은 통신 프로토콜 자체에서 감지되고 복구되어야 한다.

4.2. 필드버스(fieldbus) 프로토콜

앞서 기술한 특성을 갖춘 통신 프로토콜이라면 실시간 제어 시스템을 구성할 수 있는데, 여기에서는 제어 시스템의 필드 기기(센서 및 액추에이터) 등을 연결하는 필드버스(fieldbus) 프로토콜을 소개하고자 한다 [8]. 필드버스는 원래 특정 분야의 요구 조건을 만족시키기 위해 다수의 산업체에서 개발되었으나 현재는 다양한 프로토콜들을 표준화하려는 노력이 적극적으로 추진되고 있다. 필드버스는 OSI 표준의 7계층 모델에서 응용 오류(application layer), 데이터 연결(data link layer), 물리적(layer)의 3계층으로 구분된 모델로서 자동차 용송에 사용되는 CAN [5], VAN 등의 있고, 원거리 입력을 위한 Bitbus, 독립적으로 제안된 PROFIBUS [7], 프로그램에서 제안된 FIFO [6] 등이 있다.

현재 자동차 용송뿐 아니라 다른 자동 장비, 의료 기기에서 초고속 광학 망원경이 어려와서 그 시장을 넘어가고 있는 CAN은 1996년도에 1000만개의 칩이 판매되었기도 하였다. CAN은 CSMA/CD 방식을 채택하여 정지(identifier)를 통한 버스 중재(bus arbitration)를 한다. 각 메시지는 고유의 정지자를 갖는데, 정지자는 메시지의 우선순위 역할을 하여 우선순위에 기반한 실시간 스케줄링을 가능하게 한다. CAN에 대한 실시간 스케줄링은 Tindell과 Shin에 의해 분석된 바 있다. CAN의 실시간 성능은 CAN controller 구현 방식에 따라 약간 다를 수 있다. [5]에 따르면 Queue-oriented model을 따르는 Basic-CAN보다 DP(Dual-Ported) RAM을 이용한 Full-CAN이 더 좋은 실시간성을 가진다. DP RAM을 이용할 경우 우선순위 역전(priority inversion)을 줄여 우선순위가 높은 메시지
지가 낮은 메시지에 의해 불로경되는 시간을 줄일 수 있기 때문이다. 현재 CAN의 국제 표준안은 ISO 11898
과 ISO 11519에 등재되어 있다.

한편, 프랑스에서 제안된 FIP는 개방형 구조로서 가
장 인정받고 있는 필드버스중 하나이다. FIP는 중앙집
중형의 제어 방식을 따르며 실행할 스케줄링(pre-run-time
scheduling)을 체택한다. 이는 순환실행체계(cyclic execu-
tive) [9] 스케줄링 방식과 거의 동일하게 스케줄링이 결
정적(deterministic)이어서 시스템의 예측성(predictability)을
증대시킨다. 이러한 스케줄링 방식은 스케줄링 알고리즘
을 사용자의 의도에 따라 자유롭게 선택할 수 있다는
장점이 있지만 비선점(non-preemptible)형이기 때문에 스
케줄링의 효율성이 떨어진다는 단점도 있다.

4.3. 클록 동기화(clock synchronization)

분산 실시간 시스템의 개발에 있어서 또 하나의 문제
점은 기준 클록(reference clock)이 없다는 것이다. 기준
클록이 없으므로 각 노드들은 제각기 자신의 클록에 따
라 동작한다. 만약 모든 클록이 정확하다면 문제가 없겠
지만 불일치적으로 클록은 주변 환경에 따라 변동되거나
또한 드리프트(drift)가 발생하여 정확화
지 않게 된다. 이로 인해 분산된 하드웨어의 수행이나
메시지 전송과 같은 이벤트(event)들의 순서가 위배되거나,
상상치는 시스템의 동작이 시간적인 제약을 만족하는
지를 확인해 볼 도리가 없는 것이다. 따라서 분산된 각
클록들은 기준 시간에 따라 동기화(synchronize)되어야
한다. 클록 동기화는 여러 가지 방법이 있으나 Lamport
의 CNV나 COM이 대표적이다 [10]. 이 알고리즘들은 시
스템의 각 클록들의 값을 반영하여 일정한 간격으로 분
산된 클록들의 값을 갱신하는데, 전체 클록의 3분의 1이
고장나도 동기화가 가능하도록 해준다. 클록 동기화는
어느 레벨에서 수행되느냐에 따라 그 정확도가 다를뿐,
응용 소프트웨어 레벨에서는 약 500 μsec 에서 5msec,
운영체계 레벨에서는 약 10 μsec 에서 100 μsec , 하드
웨어 controller 레벨에서는 10 μsec 이하의 정확도를 얻
을 수 있다.

하지만 앞서 설명한 CAN이나 FIP는 프로토콜 상에서
응용 프로그램과 하위의 데이터 연결층(data link layer)간
의 동기화를 지원하지 않고 있다. 이에 따라 분산된 응
용 소프트웨어간의 클록 동기화를 직접적으로 얻을 수
없다. 이에 따라 하드웨어로나 소프트웨어로 별
도의 클록 동기화를 고려해야 한다. CAN의 경우,
을 이용하여 CAN 내트워크 상에서 동기화하는 방법을
제안하였다. 한편, FIP 상에서는 FIP controller가 발생시
키는 send/receive 인터럽트를 이용하여 소프트웨어적으
로 클록을 동기화할 수 있다.

5. 결론

앞서 언급한 바와 같이 제어 분야와 실시간 분야간에
하는 핵심적 문제로, 실시간 제어 시스템을 개발하는
이들은 필수적인 실시간 시스템을 고려하였다. 주로 시스템
설계를 위한 이론, 실시간 운영체계, 필드버스를 중심으로
하는 실시간 통신 및 클록 동기화에 대해 기술하였는데
다소 생소한 감도 없지 않다. 하지만 앞으로 실시간 제
어 시스템에 대한 관심과 필요가 급격히 증대될 것으로
기대되므로 본 고가 독자에게 좋은 자극이 되었으면 하
는 바이다.

참고 문헌

Real-Time Requirements with Resource-Based Cali-
bration of Periodic Processes", IEEE Transactions on
[2] H. Kepetz, "Component-Based Design of Large
Distributed Real-Time Systems", Workshop on
Distributed Computer Control Systems, pages 171-
177, July 1997.
Algorithm for Embedded Digital Controllers", Tech-
nical Report SNU-EE-TR-96-1, School of Electrical
Engineering, Seoul National University, October 1996.
Distributed Real-Time Systems", Workshop on
Distributed Computer Control Systems, pages 227-
227, July 1997.
Real-Time Communications: Controller Area Network
(CAN)", IEEE Real-Time Systems Symposium,
December 1994.
[7] K. Bender, "PROFIBUS - The Fieldbus for Industrial
Automation", Carl Hanser Verlag & Prentice Hall,
1993.
[8] P. Pleinevaux and J. D. Decotignie, "Time Critical
Communication Network: Field Buses", IEEE Network

'98 년도 학회지 기술특집

❤ 아래와 같이 기술 특집 주제를 확정하였습니다. 관련된 논문이나 보고서가 있으면 보내주시기 바랍니다.

원고는 학회지 편집부 김종원 교수((Tel. 880-7138, Fax. 883-1513, E-mail: mejwkim@asri.snu.ac.kr)에 게 보내주시면 편집하여 게재토록 하겠습니다.

- 아래 -

<table>
<thead>
<tr>
<th>일</th>
<th>제 목</th>
</tr>
</thead>
<tbody>
<tr>
<td>1월</td>
<td>실시간 제어시스템</td>
</tr>
<tr>
<td>3월</td>
<td>DSP 활용 기술</td>
</tr>
<tr>
<td>5월</td>
<td>정보처리기기 기술</td>
</tr>
<tr>
<td>7월</td>
<td>반도체 공정 자동화 시스템</td>
</tr>
<tr>
<td>9월</td>
<td>센서기술 응용</td>
</tr>
<tr>
<td>11월</td>
<td>보완계측 기술</td>
</tr>
</tbody>
</table>