□특집□

분산 실시간 제어 시스템의 설계 방법

홍 성 수†

◈ 목 차 ◈

1. 서 론

최근 저가의 고성능 마이크로 프로세서가 출현함에 따라 대부분의 실시간 제어 시스템은 디지털 제어 시스템으로 대체되었다. 컴퓨터를 이용한 시스템은 유연성(flexibility)과 적응성(responsiveness)이 뛰어날 뿐만 아니라 고도로 복잡한 시스템을 효과적으로 제어할 수 있기 때문이다. 특히 디지털 제어 시스템은 우주 항공, 방위 산업, 공장 자동화 등 시간적 제약의 만족이 중요시되는 분야에서 필수적인 역할을 하고 있다. 이러한 실시간 제어 시스템은 다음과 같은 시간적 제약을 갖는다 [1, 2].

* 항공기의 운항 좌표계는 최소한 50ms만에 한번은 갱신되어야 한다.
* 핵발전소의 방사능 유출은 200ms안에 감지되어야하고 차단되어야 한다.

현재 실시간 시스템 분야에서는 위와 같은 시간 제약을 만족시키기 위해 실시간 스케쥴링, 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 아직까지는 이들을 포괄하여 분산 실시간 제어 시스템을 설계할 수 있는 이론이 정립되지 못한 실정이다. 따라서 대부분의 실시간 제어 시스템의 개발은 주로 경험적인 방법에 의존하고 있는 실정이다. 이러한 방법은 기능적 성능의 극대화에만 치중되기 쉬운 반면에 시스템의 시간적 특성을 제대로 고려하기 어렵다. 이로 인해 주어진 자원(resources, CPU 혹은 network)을 최적으로 활용하지 못하거나 심지어는 시간적 제약을 충족시키지 못하기도 한다 [3].

본 고에서는 분산 실시간 제어 시스템의 설계를 위한 이론적인 방법론과 함께 구현상에서 고려해야 할 점들을 알아본다. 구체적으로 실시간 시스템의 분석 및 모델링, 실시간 운영체제, 실시간 통신 및 클록 동기화 등을 소개하고자 한다.

2. 분산 실시간 시스템의 설계 이론

일반적으로 실시간 시스템은 구조적 기법에 따라 설계되는데, 이는 하향식(top down)으로 시스템을 모듈화하여 체계적으로 구축하는 것을 말한다. 실제로 미국 국방성에서도 이를 표준화하여 사용하고 있다 (DOD-STD-2167). 이러한 구조적

† 정회원 : 서울대학교 공과대학 전기공학부 조교수

기법은 (1) 제약 조건의 구체화, (2) 시스템 구성 요소의 구현, (3) 시스템 통합, (4) 검증의 4단계로 구성된다. 이는 주어진 기능적, 시간적 요구사항으로부터 제약 조건들을 파악한 후, 시스템을 프로세스(process) 단위로 세분화하여 프로세스간의 통신 및 동기화 등을 고려하여 구현하는데 중점을 둔 방법이다. 이를 발전시킨 것으로 DARTS (Design Approach for Real-Time Systems)가 있는데 [4], 이는 General Electric사에서 개발된 기법으로서 데이터 흐름에 역점을 두고 설계하는 방법이다. DARTS에서는 시스템과 외부 환경과의 관계를 정의하는 것을 시작으로 데이터와 제어의 흐름을 중심으로 시스템을 기술한다. 이로부터 행위 모델을 구축하여 시스템을 병렬적인 태스크로 구성한다.

이외에도 객체 지향 방법에 따른 Selic의 ROOM(Real-Time Object Oriented Modeling)이 있다 [5]. 이는 event-driven 방식에 기반하여 시스템의 데이터 흐름을 기술하는 메시지 순서 도면(message sequence chart)을 통해 시스템을 모델링한다. 시스템의 구성 요소인 행위자(actor)는 메시지에 의한 요청에 따라 반응하고 새로운 메시지를 생성하여 다른 행위자에게 전해준다. 이러한 객체 모델은 시스템을 구성하는 태스크 모델과 동일한 것으로 간주될 수 있다.

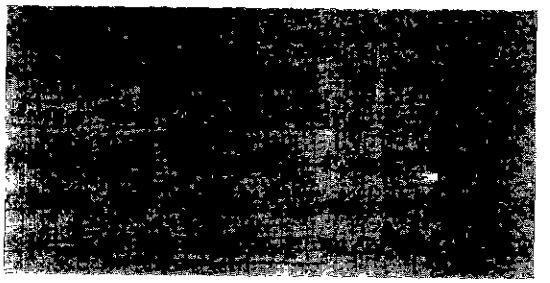
최근에는 Gerber [1]등이 더욱 발전된 설계 이론인 PCM(Period Calibration Method)이라는 방법론을 제안하였다. 이는 시스템을 태스크(task 혹은 process) 단위로 세분하여 각 태스크에 개별적인 시간적 제약(주기, 종료시한 등)을 부과함으로써 시스템 전체에 주어진 시간적 제약을 만족시키도록하는 방법이다. 사실, 앞서 기술된 방법론들은 시스템의 시간적 제약을 명확히 기술하지 못하여 실시간 시스템이 아닌 일반 시스템의 설계 방법론과 크게 다르다고 할 수 없다. 반면, PCM은 시간적 제약의 기술은 물론 이를 만족시키기 위한 방법까지 제안하고 있어 실제적인 유용성이 크다 하겠다. 한 편, Kopetz [6]는 이상적인 구성요소(component)를 정의하여 시스템 설계, 구현 및 유지를 구성적으로 가능하게 하였다. Kopetz는 특히 고장 허용(fault tolerant) 실시간 시스템을 위해 시간장벽(temporal firewall)의 개념을 도입하였으며, 이를 통해 하부 시스템간의 시간적 오류의 전파를 배제할 수 있도록 하였다.

Gerber와 Kopetz가 제안한 설계 이론은 시스템을 구성적인(composable) 방법에 의해 분석하고 설계한다는데 큰 의미를 가진다. 일반적으로 시스템의 시간적 특성은 기능적 특성과 밀접하게 연관되어 그 설계 및 구현을 아주 어렵게 만든다. 하지만 이들의 방법은 시스템의 기능적 특성으로부터 시간적 특성만을 가능한 분리하고, 또한 각 구성 요소간의 시간적인 상호 간섭을 배재한다. 따라서 시스템의 분석과 모델링은 물론 설계를 용이하게 하여 복잡한 실시간 제어 시스템 개발에 적합하다.

이 절에서는 Gerber가 제안한 PCM의 설계 방법을 간략히 소개하고자 한다. PCM에서 시스템의 기능적 특성은 DARTS의 태스크 구조 다이어그램(task architecture diagram)과 유사한 태스크 그래프(task graph)로서 기술된다. 지면 관계상, 여기서는 시스템에 부과된 시간적 제약을 기술하고 이를 만족시키는 방법을 위주로 소개하겠다.

PCM에서 실시간 시스템의 설계는 시스템의 기능적 특성을 기술하는 태스크 그래프와 태스크들을 각 프로세서에 할당한 정보, 시스템의 시간적 특성을 기술하는 양극단 시간 제약(end-to-end timing constraints)으로부터 시작한다. 여기서 양극단 시간 제약이라 함은 서론에서 언급된 예와 같이 시스템에 부과된 시간적 제약을 말한다. 이와 같은 양극단 시간 제약으로부터 각 태스크의 주

기(period), 종료시한(deadline), 위상(phase)과 같은 시간적 속성이 유도된다. 이러한 개개의 태스크의 시간적 속성은 실시간 스케쥴링에 의해 만족될 수 있으며, 이를 통해 처음에 부과된 양극단 시간 제약은 자동적으로 만족된다.



(그림 1) 분산 실시간 제어 시스템의 설계 (PCM)

태스크 그래프는 생산자/소비자(producer/consumer) 모델을 기초로 한다. 그림 2는 입력 데이터가 각 태스크들을 거쳐 최종적인 출력을 생성되기 까지의 과정을 간략히 보여주고 있다. X, Y는 각각 입력과 출력 데이터, T는 태스크, p는 태스크간의 통신을 위해 사용되는 통신 포트를 의미한다. 그림 2의 모델은 태스크의 선행 관계, 데이터의 흐름 등, 시스템의 기능적 특성을 잘 기술해 준다. 한 편 시스템의 시간적 특성은 양극단 시간 제약으로 기술된다. 일반적으로 양극단 시간 제약은 시스템의 입력과 출력 (X, Y)간의 시간적 관계를 제약하는 것으로 입력 동기화 제약(Input Correlation Constraint), 최대 유효 허용 시간(Maximum Allowable Validity Time)와 최대 출력 주기(Maximum Loop Processing Period)의 형태로 주어진다.

(그림 2) 태스크 그래프

* 입력 동기화 제약 - 다수의 입력 데이터들의 시간적 상관 관계를 제약한다.
* 최대 유효 허용 시간 - 입력으로부터 출력이 생성되기 까지 걸리는 최대 시간값을 의미한다.
* 최대 출력 주기 - 출력이 생성되는 주기의 최대값을 의미한다.

위와 같은 방법으로 얻어진 양극단 시간 제약은 태스크의 시간적 속성을 유도하기 위한 중간 시간 제약으로 변환된다. 이 때 분산 시스템상의 프로세서간의 통신에 의한 시간 지연은 태스크로 간주된다. 중간 시간 제약은 통신을 포함한 태스크들의 주기, 종료시한, 위상 등의 변수로 표현되는 일련의 부등식과 주기 조화성, 그리고 목적 함수를 포함하는 비선형 최적화 문제가 된다. 중간 시간 제약이 유도되면, 주어진 범위내에서 목적 함수를 최적화시키도록 각 태스크의 속성을 할당한다. 본 논문에서는 지면 관계상 구체적인 방법을 기술하지 못하니 주기값 유도는 [1]을 종료시한 및 위상값의 유도는 [13]을 참조하기 바란다.

3. 실시간 운영체제

앞 절에서 소개된 태스크는 제어 알고리즘을 실제 수행 가능한 코드로 구현된 것으로 이들을 수행시키기 위해서 적합한 실시간 운영 환경이 필요하다. 이에 따라 오늘날에는 실시간 특성을 요구하는 분산 제어 시스템의 개발을 위해 상용의 실시간 운영체제를 많이 이용한다. 이 절에서는 기존의 범용 운영체제에 대한 지식을 전제로 실시간 운영체제가 갖추어야 할 특징을 알아 보고 현재 많이 사용되고 있는 상용 실시간 운영체제들을 소개한다.

첫째, 실시간 운영체제는 선점가능한(preemp-

tible) 멀티쓰레드(multithread)를 지원할 수 있어야 한다. 멀티쓰레드는 시스템의 병렬성(concurrency)을 지원할 뿐만 아니라 자원의 효율적 관리를 가능하게 하며, 문맥 교환(context switching) 시간을 줄여 시스템의 응답성을 향상시켜준다. 이러한 쓰레드는 구현 방식에 따라, 쓰레드가 커널의 기능 자체로 구현된 커널 레벨 쓰레드(kernel-level thread)와 유저의 라이브러리 함수(library functions)로 구현된 유저 레벨 쓰레드(user-lever thread)로 구분된다. 커널 레벨 쓰레드는 쓰레드 스케줄링이나 시그날링(signaling)을 단순화하는 장점이 있으나 오버헤드(overhead)가 크다는 단점이 있다. 반면에 유저 레벨 쓰레드는 쓰레드 스케줄링을 어렵게 하지만 오버헤드가 적다는 장점이 있다. 이에 따라 최근에는 이 둘을 혼합한 형태의 쓰레드 모델들이 제안되기도 하였다. 멀티 쓰레드 모델은 현재 실시간 운영체제에만 국한되는 것이 아니라 성능 향상을 위해 기존의 많은 범용 운영체제에서도 도입되었다. 선 마이크로시스템(Sun Microsystems)의 솔라리스(Solaris), 마이크로소프트(Microsoft)의 윈도우즈(Windows) NT, Mach 등에서도 멀티쓰레드를 지원하도록 개발되었다.

둘째, 실시간 운영체제는 예측 가능한 실시간 스케줄링과 동기화(synchronization) 메카니즘을 제공하여야 한다. 실시간 스케줄링은 실시간 분야에서 그 동안 가장 활발히 연구되어온 분야로서 현재 많은 이론들이 제안되었다. 특히 Liu와 Layland가 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[12]. 이들은 현재 가장 많이 사용되기도 하는 알고리즘으로서 실시간 운영체제는 이러한 실시간 스케줄링을 지원하여야 시스템의 자원을 효과적으로 관리하면서 주어진 시간 제약을 만족시킬 수 있다. 또한 실시간 운영체제는 우선순위계승 프로토콜(priority inheritance protocol)을 지원할 수 있어야 한다. 이는 공유 자원의 사용에 있어서 블로킹 시간(blocking time)을 제한하여 쓰레드 동기화를 예측 가능케 한다.

마지막으로, 운영체제의 시간적 특성을 명확히 파악할 수 있어야 한다. 예를 들면, 인터럽트가 발생한 시점에서부터 태스크가 수행을 재개할 때까지 걸리는 인터럽트 잠복기(interrupt latency)를 알 수 있어야 한다. 따라서 응용 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 운선순위(priority)를 제공하지 못하여 실시간 스케줄링에 적합하지 않다. 또한 우선순위 계승 프로토콜(priority inheritance protocol)을 제공하지 못하며, 운영체제의 시간적 특성이 명확히 파악되지 않는다. 예를 들면, Windows NT는 우수한 하드웨어 인터페이스를 제공하지만 Pentium Power Management 인터럽트가 예측할 수 없는 시간만큼 시스템을 선점(preempt)한다. 따라서 응용 프로그램의 시간적 분석을 어렵게 함은 물론 신뢰할 수 있는 시스템을 개발할 수 없다. 마지막으로 Windows NT는 programmable 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 표준안으로는 현

재 POSIX 1003.1b가 나와 있다. 최근 서울대학교 실시간 운영체제 연구실에서는 ARX/ULTRA라는 실시간 커널을 개발한 바 있다. ARX는 동적 스택 바인딩(dynamic stack binding)과 업콜(upcall)을 통해 user-level상에서 멀티쓰레드를 효과적으로 지원한다. 또한 운영체제의 유연성(flexibillity)을 향상시키기 위해 user-level I/O를 지원한다.

4. 실시간 통신 및 클록 동기화

4.1 실시간 통신의 특성

분산 실시간 시스템의 개발은 기본적으로 공유 메모리가 없다는 문제점을 안고 있다. 이에 따라 분산된 각 노드들은 서로 데이터를 주고 받는데, 이러한 메시지 전송 또한 태스크와 마찬가지로 실시간 특성을 갖추어야 한다. 메시지 전송은 선택한 통신 프로토콜에 따라 시간적 특성을 달리하는데 실시간 통신 프로토콜이 일반 프로토콜과 특히 구분되어야 할 특성은 다음과 같다.

첫째, 최대 전송 시간(protocol latency)과 지터(latency jitter)가 작아야 한다. 일반적인 의미에서는 평균 전송 시간이 작을수록 좋은 통신 프로토콜이 된다. 하지만 실시간 시스템에서는 최악의 상황을 전제로 분석 및 개발이 이루어지기 때문에 최대 전송시간이 작을수록 좋다. 또한 전송시간의 변화폭을 지터라 하는데 이 또한 작을수록 좋다. 지터는 센서 입력이나 제어 출력의 전송을 비정규적(irregular)으로 만들어 실제 제어 성능에도 영향을 끼치는 요소이다. ATM(asynchronous transfer mode)은 광대역 통신망상에서 지터를 최소화하기 위해 개발되어 현재 널리 보급중이다.

둘째, 실시간 통신은 신뢰성(reliability)을 가져야 한다. 대개의 제어 시스템의 경우 열악한 환경에 적용되므로 통신망 자체가 강인하여 고장이 쉽게 나지 않아야 한다. 또한 여러 가지 오류에도 감내할 수 있어야 시스템의 신뢰성을 얻을 수 있다. 예를 들어 메시지가 전송이 안되면 이는 즉시 감지(detect)되고 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 감지 되고 복구되어야 한다.

4.2 필드버스(fieldbus) 프로토콜

앞서 기술된 특성을 갖춘 통신 프로토콜이라면 실시간 제어 시스템 응용에 무리없이 사용될 수 있는데, 여기에서는 제어 시스템의 필드 기기 (센서 및 액츄에이터)등을 연결하는 필드버스(fieldbus) 프로토콜을 소개하고자 한다. 필드버스는 원래 특정 분야의 요구 조건을 만족시키기 위해 다수의 산업체에서 개발되었으나 현재는 다양한 프로토콜들을 표준화하려는 노력이 적극적으로 추진되고 있다. 필드버스는 OSI 표준안인 7계층 모델에서 응용층(application layer), 데이터 연결층(data link layer), 물리층(physical layer)의 3계층으로 축소된 모델로서 자동차 응용에 사용되는 CAN, VAN 등이 있고, 원거리 입출력을 위한 Bitbus, 독일에서 제안된 PROFIBUS, 프랑스에서 제안된 FIP 등이 있다.

현재 자동차 응용 뿐만 아니라 다른 자동 항법, 의료 기기에서 초대형 광학 망원경에 이르기까지 그 시장을 넓혀 가고 있는 CAN은 1996년도에 1000만개의 칩이 판매되기도 하였다. CAN은 CSMA/CD방식을 채택하여 지정자(identifier)를 통한 버스 중재(bus arbitration)를 한다. 각 메시지는 고유의 지정자를 갖는데, 지정자는 메시지의 우선순위 역할을 하여 우선순위에 기반한 실시간 스케줄링을 가능하게 한다. CAN에 대한 실시간 스케줄링 분석은 Tindell의 [10]과 Shin의 [11]에서

논문에서 찾아 볼 수 있다. CAN에 대한 분석은 CAN controller 구현 방식에 따라 약간 다른데, [10]에 따르면 Queue-oriented model을 따르는 Basic-CAN 보다 DP(Dual-Ported) RAM을 이용한 Full-CAN이 더 좋은 실시간성을 가진다. DP RAM을 이용할 경우 우선순위 역전(priority inversion)을 줄여 우선순위가 높은 메시지가 낮은 메시지에 의해 블로킹되는 시간을 줄일 수 있기 때문이다. 현재 CAN의 국제 표준안은 ISO 11898과 ISO 11519에 등재되어 있다.

한편, 프랑스에서 제안된 FIP는 개방형 구조로서 가장 인정받고 있는 필드버스중의 하나이다. FIP는 중앙집중형의 제어 방식을 따르며 실행전 스케쥴링(pre-run-time scheduling)을 채택한다. 이는 순환실행체제(cyclic executive) 스케쥴링 방식과 거의 동일하여 스케쥴링이 결정적(deterministic)이어서 시스템의 예측성(predictability)를 증대시킨다. 이러한 스케쥴링 방식은 스케쥴링 알고리즘을 사용자의 의도에 따라 자유롭게 선택할 수 있다는 장점이 있지만 비선점(nonpreemptible)형이기 때문에 스케쥴링의 효율성이 떨어진다는 단점도 있다.

4.3 클록 동기화(clock synchronization)

분산 실시간 제어 시스템의 개발에 있어서 또 하나의 문제점은 기준 클록(reference clock)이 없다는 것이다. 기준 클록이 없으므로 각 노드들은 제각기 자신의 클록에 따라 동작한다. 만약 모든 클록이 완벽하다면 문제가 없겠지만 물리적으로 클록은 주변 환경(예를 들면 온도나 습도)에 따라 클록 드리프트(clock drift)가 발생하여 정확하지 않게 된다. 이로 인해 분산된 태스크들의 수행이나 메세지 전송과 같은 이벤트(event)들의 순서가 뒤바뀌거나, 심지어는 시스템의 동작이 시간적 제약을 만족하는지를 확인해 볼 도리가 없는 것이다. 따라서 분산된 각 클록들은 기준 시간에 따라 동기화(synchronize)되어야 한다. 클록 동기화는 여러 가지 방법이 있으나 Lamport의 CNV나 COM [8], Kopetz의 FTA [9]등이 대표적이다. 이 알고리즘들은 각 클록들의 값을 반영하여 일정한 간격으로 분산된 클록들의 값을 갱신한다. 클록 동기화는 어느 레벨에서 수행되느냐에 따라 그 정확도가 다른데, 응용 소프트웨어 레벨에서는 약 500 μsec 에서 5msec, 운영체제 레벨에서는 약 10 μsec 에서 100 μsec, 하드웨어 controller 레벨에서는 10 μsec 이하의 정확도를 얻을 수 있다.

하지만 앞서 설명된 CAN이나 FIP는 프로토콜상에서 응용 프로그램과 하위의 데이터 연결층(data link layer)간의 동기화를 지원하지 않고 있다. 이에 따라 분산된 응용 소프트웨어간의 클록 동기화를 직접적으로 얻을 수 없다. 따라서 하드웨어적으로나 소프트웨어적으로 별도의 수단이 요구된다. CAN의 경우, Rodd는 [7]에서 클록 동기화만을 위해 고안된 TICK 칩을 이용하여 CAN 네트워크 상에서 동기화하는 방법을 제안하였다. 한편, FIP 상에서는 FIP controller가 발생시키는 send/receive 인터럽트를 이용하여 소프트웨어적으로 클록을 동기화할 수 있다.

5. 결 론

본 고에서는 실시간 시스템 이론의 새로운 응용 분야로 부상하고 있는 분산 실시간 제어 시스템을 위한 설계 이론, 실시간 운영체제, 필드버스를 이용한 실시간 통신 및 클록 동기화 등에 대해 간략히 알아보았다. 이들은 분산 실시간 제어 시스템에 관련된 연구의 일부분이지만 핵심적이고 기초가 된다는 점에서 그 중요성이 크다고 하겠다. 더욱 효과적인 시스템의 개발을 위해서는 태스크 할당을 포함하는 분산 실시간

스케줄링, 고장 허용성, 분산 메모리 동기화 등의 다양한 문제에 대한 고찰이 필요할 것이다. 국내에서도 분산 실시간 제어 시스템에 대한 관심과 필요가 급격히 증대될 것으로 기대되는 바 본 고가 독자에게 좋은 지침이 되었으면 하는 바이다.

참고문헌

[1] R. Gerber, S. Hong, and M. Saksena. Guaranteeing real-time requirements with resource-based calibration of periodic processes. IEEE Trans. on Software Engineering, 21(7):579-592, July 1995.

[2] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin. Visual assessment of a real-time system design: A case study on a CNC controller. IEEE Real-Time Systems Symposium, Washington DC, USA, December 1996.

[3] M. Ryu and S. Hong. Toward automatic synthesis of schedulable real-time controllers. Journal of Integrated Computer Aided Engineering, to appear in 1998.

[4] H. Gomma. Software design method for real-time systems. Communications of the ACM, 27(9):938-949, September 1984.

[5] B. Selic, G. Gullekson, and J. McGee, I. Engelberg. ROOM: An object-oriented methodology for developing real-time systems. CASE'92 Fifth IWCASE, July 1992.

[6] H. Kopetz. Component-based design of large distributed real-time systems. IFAC Workshop on Distributed Computer Control Systems, July 1997.

[7] M. G. Rodd, K. Dimyati, and L. Motus. The design and analysis of low cost real-time fieldbus systems. IFAC Workshop on Distributed Computer Control Systems, July 1997.

[8] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. Journal of the ACM, 32(1):52-78, January 1985.

[9] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. IEEE Trans. on Computers, C-36(8):933-940, August 1987.

[10] K. Tindell, H. Hanssen, and A. Wellings. Analysing real-time communications: Controller area network (CAN). In Proceedings of IEEE Real-Time Systems Symposium, December 1994.

[11] K. M. Zuberi and K. G. Shin. Non-preemptive scheduling of messages on controller area network for real-time control applications. In Proceedings of IEEE Real-Time Technology and Application Symposium, June 1995.

[12] C. Liu and J. Layand. Scheduling algorithm for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1)46-1, January 1973.

[13] M. Ryu and S. Hong. End-to-end design of distributed real-time systems. Control Engineering Practice, to appear in 1998. (also appeared in Proceedings of IFAC Workshop on Distributed Computer Control Systems, July 1997.)

[14] M. Ryu, S. Hong, and M. Saksena. Streamlining real-time controller design: From performance specifications to end-to-end timing constraints. IEEE Real-Time Application and Technology, Montreal, Canada, June 1997.

홍 성 수

1986년 서울대학교 컴퓨터공학과 (학사)
1988년 서울대학교 컴퓨터공학과 (석사)
1994년 University of Maryland at College Park, Dept. of Computer Science (박사)
1988년-1989년 한국전자통신연구소 연구원
1994년-1995년 Faculty Rescarch Associate (Unive. of Maryland)
1995년 Member of Technical Staff Silicon Graphics Inc.
1995년-1997년 서울대학교 공과대학 전기공학부 전임강사
1997년-현재 서울대학교 공과대학 전기공학부 조교수
관심분야 : 실시간 시스템, 분산제어 시스템, 소프트웨어 엔지니어링, 실시간 운영체제