분산 제어 시스템에서의 태스크와 메시지 기반 스케줄링을 이용한 최적 주기와 우선순위 할당

Optimal Period and Priority Assignment Using Task & Message-Based Scheduling in Distributed Control Systems

김 형 육, 이 철 민, 박 홍 성
(Hyoung Yuk Kim, Chul Min Lee and Hong Seong Park)

Abstract : Distributed control systems(DCS) using fieldbus such as CAN have been applied to process systems but it is very difficult to design the DCS while guaranteeing the given end-to-end constraints such as precedence constraints, time constraints, and periods and priorities of tasks and messages. This paper presents a scheduling method to guarantee the given end-to-end constraints. The presented scheduling method is the integrated one considering both tasks executed in each node and messages transmitted via the network and is designed to be applied to a general DCS that has multiple loops with several types of constraints, where each loop consists of sensor nodes with multiple sensors, actuator nodes with multiple actuators and controller nodes with multiple tasks. An assignment method of the optimal period of each loop and a heuristic assignment rule of each message's priority are proposed and the integrated scheduling method is developed based on them.

Keywords : period and priority assignment, real-time, scheduling, DCS

I. 개요

최근 산업현장에서는 Profibus, FIP, CAN, LonWork-s 등의 필드버스와 같은 네트웍을 이용한 분산제어가 가장 중요한 요구사항 중의 하나가 되었으며 이러한 분산 제어 시스템은 필드버스를 이용하여 센서 및 엑츄에이터, 제어기를 연결하는 구조가 일반적이다. 센서, 엑츄에이터, 제어기가 서로 분산되어 있기 때문에 센서에서 제어기, 제어기에서 엑츄에이터로의 데이터 전송이 네트웍 특성에 따라 임의 지연시간을 갖게 된다. 또한 노드의 특성에 따라 각 노드에서 수행되는 태스크의 수행시간 또한 일정치 않다. 그러므로 네트웍 지연 및 각 노드에서의 태스크 수행지연 등을 고려하지 않고 기존의 제어 시스템에 사용하던 방식대로 제어 시스템을 설계하면 문제가 된다. 따라서 네트웍 및 멀티 태스크 시스템을 기반으로 하는 제어 시스템을 설계하기 위해서는 새로운 방식이 필요하다.

이러한 분산 제어 시스템의 설계와 구현은 시간제약 조건들, 태스크들의 수행시간, 각 태스크의 노드할당과 우선순위 부여, 각 태스크와 메시지들의 선행관계(precedence relationship), 메시지의 우선 순위와 응답시간 등을 고려하여야 하므로 매우 복잡하고 어려운 일이다. 이러한 고려사항들은 분산 제어 시스템을 구성하는 각 제어 루프의 실시간 특성이 매우 중요하기 때문이다. 분산 제어 시스템의 실시간 특성은 태스크와 메시지의 주기와 우선순위에 의해 영향 받는 최악응답시간에 의해 결정지어 지는데 예를 들어, 만약 엑츄에이터 노드 i로 메시지를 전송하려고 하는 태스크 A와 시스템 진단 기능을 수행하는 태스크 B가 같은 노드에 존재하며, 태스크 A는 태스크 B보다 우선순위가 낮고, 동시에 태스크 수행이 시작된 경우 태스크 A는 태스크 B의 수행이 끝난 후에야 수행이 될 수 있다. 이와 같이 태스크의 최악응답시간은 태스크의 우선순위에 따라 변화하게 된다. 현재 마이크로 프로세스의 발달로 제어 노드에서 여러 개의 태스크들을 동시에 동작시키고 하나의 네트웍을 통하여 여러 메시지들을 전송하기 때문에 이러한 상황에 대한 연구가 필요하다.

실시간 스케줄링에 관한 연구는 크게 세 가지 부류로 나뉘어질 수 있는데 첫째는 하나의 프로세스 또는 노드 내에서 태스크들의 스케줄링[1]-[7], 둘째는 네트웍의 메시지들을 위한 스케줄링[8]-[12], 셋째는 태스크와 메시지를 모두 고려한 스케줄링[13]-[15]이다.

앞에서 언급한 문제들을 해결하기 위해서는 세 번째 연구 방식이 알맞다. [13]에서는 결정문제 영역에서의 NP-Hard 문제인 분산 제어 시스템 스케줄링을 클러스터링 알고리즘과 유전적 알고리즘을 사용하여 CSMA/ CA와 TDMA 방식의 네트웍으로 구성되는 분산 제어 시스템을 위한 스케줄링 방식을 제안되었지만 태스크간 선행제약을 만족시킬 수 있는 방법이 아니며 특정한 작업을 수행하는 태스크를 어떤 노드에 할당할 지에 대한 제약이 없다는 가정 하에서 제안되었으므로 각 노드가 수행하는 고유 태스크가 존재하는 실제 분산 제어 시스템의 환경에 적용하기 어렵다.

어떠한 태스크도 하나 이상의 메시지를 받지 않는다는 가정 하에서 릴리즈 지터(release jitter)를 이용하여 TDMA를 사용하는 분산 제어 시스템을 위한 휴리스틱(heuristic) 스케줄링 방법[14]이 제안되었다. 그러나 제안된 방법은 여러 개의

논문접수 : 2001. 9. 29., 채택확정 : 2002. 2. 26.
김형육, 이철민 : 강원대학교 제어계측공학과(petrus, jazonsim@control.kangwon.ac.kr)
박홍성 : 강원대학교 전기전자정보통신공학부(hspark@cc.kangwon.ac.kr)

메시지들을 동시에 수신하는 태스크로 구성되는 일반적인 제어 시스템에 적합하지 않고 태스크들과 메시지들간의 선행관계가 다양한 시스템의 경우에 적용하기 어렵다.

태스크 기반의 스케줄링 방법[5][6]을 CAN으로 구성되는 분산 제어 시스템으로 확장한 연구[15]에서는 여러 가지 소거법을 통하여 분산 제어 시스템을 위한 스케줄링 방법을 제안하였는데 일반적인 분산 제어 시스템과 같은 제어 노드 상의 복잡한 태스크를 고려하지 않았고 메시지들의 우선순위 및 최적주기 설정 방법은 제안되지 못했다.

본 논문은 CAN 기반 분산 제어 시스템에서 다음과 같은 시스템에서 적용할 수 있는 메시지와 태스크를 동시에 고려한 통합 스케줄링 방법을 제안한다. 모든 태스크들은 하나 이상의 메시지를 받을 수 있으며 여러 개의 센서, 엑츄에이터와 다양한 태스크를 선점형으로 수행하는 제어 노드로 구성되는 제어 루프가 여러 개 존재하는 일반적인 분산 제어 시스템이다. 제어 루프의 양극단 시간제약을 만족하면서 가장 짧게 적용될 수 있는 제어 루프의 최적 주기할당 방법을 제안하고 메시지의 우선순위 변화에 따른 제어 루프의 최악응답시간 영향 분석을 통해 메시지 우선순위 할당 규칙을 제안한다. 제안된 방법은 태스크 기반 스케줄링 방법[5][6]에 기초하여 대상 분산 제어 시스템에 적용 가능하도록 메시지와 태스크들을 통합하여 설계된 스케줄링 방법이며, 이를 이용하여 우선순위 할당 규칙과 최적 주기할당 방법을 제안한다.

앞으로 2절에서는 태스크 기반 스케줄링에 대하여 설명하고 3절에서는 본 논문에서 제안된 제어 루프의 최적 주기할당 방법과 메시지 우선순위 할당 방법을 통한 태스크와 메시지 기반 통합 스케줄링 방법을 대상 시스템에 적용한다. 마지막으로 4절에서 결론을 맺는다.

II. 태스크 기반 스케줄링[5][6][15]

1. 태스크 그래프 설계

태스크 기반의 스케줄링 방법은 시스템의 요구사항이 주어지면 이를 만족하는 태스크 그래프를 설계, 태스크를 각 노드에 할당한 후 태스크간 메시지간 제약식을 유도한다. 각 태스크와 메시지에 우선 순위와 주기할당이 이루어지면 태스크와 메시지의 최악응답시간을 구하여 제약식을 풀게 된다. 풀어진 제약식의 결과가 시스템의 양극단 시간제약을 만족시킬 때까지 위의 과정을 반복하게 된다.

두 개의 제어 루프로 구성되는 간단한 분산 제어 시스템의 예와 이를 태스크 그래프로 표현한 것이 그림 1에 나타나 있다. 태스크 그래프 상의 원은 태스크를 나타내며 타원은 태스크간 데이터를 주고받는 통신 포트를 의미하며 점선의 사각형은 노드를 의미한다. 사각형으로 표시되는 센서 1, 2(S1, S2)와 엑츄에이터 1, 2(A1, A2)가 각각 센서 노드와 엑츄에이터 노드에 연결되어 있으며 센서 2는 2 개의 제어 루프의 입력 값으로 사용된다. 첫 번째 제어 루프는 4개의 태스크, 센서 1 태스크, 센서 2 태스크, 제어 1 태스크, 엑츄에이터 1 태스크로 구성되며 각각 $\tau_{S1}, \tau_{S2}, \tau_{C1}, \tau_{A1}$로 표시된다. 두 번째 제어 루프는 3개의 태스크, 센서 2 태스크, 제어 2 태스크, 엑츄에이터 2 태스크로 구성되며 $\tau_{S2}, \tau_{C2}, \tau_{A2}$로 표시된다.

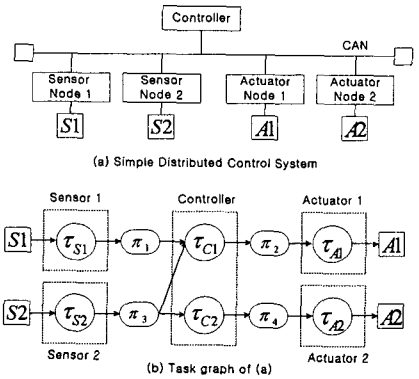


그림 1. 간단한 DCS와 태스크 그래프.
Fig. 1. A simple DCS and its task graph.

태스크 그래프 설계 시 다음과 같은 몇 가지 규칙이 있다. 시스템은 태스크들과 통신포트(메시지)들의 조합으로 구성되며 각 태스크는 입력포트에 씌어진 데이터를 주기적으로 읽고 해당 연산수행 후 출력포트에 쓴다. 하나의 태스크는 여러 입력포트와 여러 출력포트를 가질 수 있으며 하나의 포트는 하나의 태스크만이 쓸 수 있고 여러 태스크가 읽을 수 있다.

2. 스케줄링에서 고려되는 제약

분산 제어 시스템은 시스템 요구사항 및 설계 구조에 의해서 여러 가지 제약을 갖게 되며 태스크 그래프 설계와 태스크의 노드 할당이 이루어진 후 제약식이 유도된다. 제약식을 표현하기 위해 사용되는 표기법으로 i번째 제어 루프의 j번째 태스크, τ_j^i는 $\{e_j^i, T_j^i, d_j^i, p_j^i, \phi_j^i\}$와 같은 다섯 가지 요소로 표현되며 여기서 e_j^i는 태스크 τ_j^i의 수행시간이고 T_j^i는 태스크 τ_j^i의 주기, d_j^i는 태스크 τ_j^i의 데드라인, p_j^i는 태스크 τ_j^i의 우선 순위, ϕ_j^i는 τ_j^i의 최초 시작시간이다.

분산 제어 시스템이 갖는 첫 번째 제약으로는 센서에서 샘플된 데이터가 엑츄에이터에 영향을 미치기까지의 시간인 양극단 시간제약으로 최대 허용가능 지연 시간(MADT : Maximum Allowable Delay Time)으로 표현된다. 만약 i번째 제어 루프의 엑츄에이터 A의 출력 태스크를 τ_A^i라고 하면 $\phi_A^i + d_A^i \le MADT^i$의 조건이 성립된다. 여기서 $MADT^i$는 i번째 제어 루프의 MADT이다.

두 번째 제약으로는 태스크간 생산자와 소비자 관계로 인한 제약을 들 수 있는데, 소비자 태스크 τ_C^i가 단지 하나의 포트로부터 데이터를 읽을 경우 τ_C^i의 주기는 해당 포트로 데이터를 쓰는 생산자 태스크 τ_P^i의 주기와 같아도 무방하므로 이들 두 태스크간의 주기간에는 $T_C^i = T_P^i$라는 관계가 성립될 수 있다. 하지만 소비자인 태스크 τ_C^i가 한 개 이상의 포트로부터 데이터를 읽을 경우 입력 데이터간 동기시간 제약이 존재하므로 τ_C^i의 주기는 생산자 태스크들의 주기와 정수배 관계(Harmonicity Relation)의 제약을 만족해야 한다. 만약 소비자 태스크 τ_C^i가 생산자1 태스크 τ_{P1}^i와 생산자2 태스크

가 쓴 포트로부터 데이터를 읽을 경우 이 태스크들의 주기관계는 $T_{P1}^i | T_C^i$와 $T_{P2}^i | T_C^i$의 제약이 성립된다. 여기서 "A|B"는 B가 A의 정수배임을 의미한다.

세 번째로는 태스크간 선행 제약으로 생산자 태스크의 수행이 완료되기 전에 소비자 태스크가 수행되면 안되므로 소비자 태스크는 생산자 태스크의 데드라인만큼 지연되어 수행된다. 만약 두 개의 태스크, τ_P^i와 τ_C^i가 생산자와 소비자의 관계에 있다면 $\phi_P^i + d_P^i \le \phi_C^i$의 제약 조건이 성립된다.

3. 태스크 기반 스케줄링의 적용

시스템 모델을 태스크 그래프로 설계한 뒤 2.2절에서 언급된 태스크간 제약을 고려하여 해당 시스템의 제약들을 유도한다. 그림 1 (b)의 태스크 그래프에 대한 제약식을 유도하면 주기에 대한 제약식들과 데드라인과 초기시작시간에 제약식들로 나누어지며 주기에 관한 제약식 집합은 다음과 같이 유도된다. 여기서는 표시를 단순하게 하기 위하여 제어 루프의 번호를 생략한다.

- 센서 노드에서 메시지 : $T_{S1} = T_{m1}$, $T_{S2} = T_{m3}$
- 메시지에서 제어 노드 : $T_{m1} | T_{C1}$, $T_{m3} | T_{C1}$, $T_{m3} = T_{C2}$
- 제어 노드에서 메시지 : $T_{C1} = T_{m2}$, $T_{C2} = T_{m4}$
- 메시지에서 엑츄에이터 노드: $T_{m2} = T_{A1}$, $T_{m4} = T_{A2}$

여기서 T_{mk}는 각 통신 포트(k=1,2,3,4,)의 지연시간을 계산하기 위해서 필요한 메시지 주기를 의미한다.

초기 시작시간과 데드라인에 대한 중간 제약식은 다음과 같다.

- 양극단 시간제약식

- $\phi_{A1} + d_{A1} - \min(\phi_{S1}, \phi_{S2}) \le MADT^1$

- $\phi_{A2} + d_{A2} - \phi_{S2} \le MADT^2$

- 센서 노드에서 제어 노드

- $\phi_{C1} \ge \max(\phi_{S1} + d_{S1} + d_{m1}, \phi_{S2} + d_{S2} + d_{m3})$

- $\phi_{C2} \ge \phi_{S2} + d_{S2} + d_{m3}$

- 제어 노드에서 엑츄에이터 노드

- $\phi_{A1} \ge \phi_{C1} + d_{C1} + d_{m2}$, $\phi_{A2} \ge \phi_{C2} + d_{C2} + d_{m4}$

여기서 d_{mk}는 각 통신 포트(k=1,2,3,4)에 대한 최대 지연시간을 의미한다.

제약식 유도 후 태스크와 메시지의 주기와 우선 순위를 할당하고 태스크와 메시지의 최악응답시간을 계산한 후 이를 제약식에 대입하여 제어 루프의 양극단 시간 제약이 만족되는지 검사한다. 시스템에 적용할 태스크와 메시지의 주기와 우선순위 집합은 무수히 많기 때문에 양극단 시간제약을 만족하는 집합을 찾는 것이 매우 중요하며 찾아진 집합 내에서도 어떤 집합이 최적인지를 결정해야 한다. 기존에 제안된 방법으로는 최적화 알고리즘을 통해 모든 경우의 수를 계산하는 방법[13]과 소거법을 통한 적용 가능한 집합을 줄여나가는 방법[5][6][15]이 있다. 본 논문에서는 3절에서 제안된 메시지 우선순위 할당 규칙과 제어 루프의 최적 주기할당 방법을 통하여 제약식을 풀어나간다.

III. 태스크와 메시지 기반 통합 스케줄링 방법

1. 시스템 모델과 태스크 모델

본 논문에서 예로서 고려되는 분산 제어 시스템은 그림 2에서 보여지는 바와 같이 여러 개의 제어 루프로 구성되어 있으며, 각 제어 루프는 CAN을 통해 여러 개의 센서 노드와 엑츄에이터 노드, 하나의 제어 노드로 구성되어 있다. 총 4개의 루프가 고려되었는데 각 루프는 두 개의 센서 노드와 한 개의 제어 노드, 한 개의 엑츄에이터 노드로 구성된다. 센서 노드 2와 5의 샘플된 데이터는 각각 두 개의 제어 노드로 동시에 전송된다. 첫 번째 제어 루프는 센서 노드 1과 2, 제어 노드 1, 엑츄에이터 노드 1로 구성된다. 두 번째 제어 루프는 센서 노드 2와 3, 제어 노드 2, 엑츄에이터 노드 2로 구성된다. 세 번째 제어 루프는 센서 노드 4와 5, 제어 노드 3, 엑츄에이터 노드 3으로 구성된다. 네 번째 제어 루프는 센서 노드 5와 6, 제어 노드 4, 엑츄에이터 노드 4로 구성된다. 각 노드에서 수행되는 태스크들로는 센서 노드 상의 샘플링 태스크와 제어 노드 상의 제어 태스크와 그 외의 세 개의 태스크, 엑츄에이터 노드 상의 출력 태스크가 있다.

그림 2의 대상 시스템의 태스크 그래프가 그림 3에 나타나 있으며 태스크 그래프에 사용된 표기법이 표 1에 나타나 있다. Network으로 표시된 점선 사각형은 네트웍을 의미하며 네트웍 태스크(τimj)는 통신포트와 1대1로 매핑된다. 하나의 노드 상에서의 태스크간 통신포트는 매우 짧은 시간이 걸리는 메모리 읽기, 쓰기이므로 생략이 가능하지만 다른 노드에 존재하는 태스크간 통신포트는 네트웍의 종류와 전송속도, 전송량, 우선순위에 따라 전송시간이 결정되므로 이를 고려하기 위해 가상의 네트웍 태스크를 사용한다.

통합 스케줄링 방법을 적용할 그림 3의 태스크 그래프는 하나의 루프에 여러 개의 노드가 사용되고 하나의 노드 상에 여러 개의 태스크가 존재하는 그리고 하나의 데이터가 여러 노드에 의해 사용될 수 있는 실제적인 분산 실시간 제어 시스템 모델이다.

2. 제약식 유도

이 절에서는 각 제어 루프의 양극단 시간 제약을 만족하는 태스크들의 주기, 초기시작시간, 데드라인을 산출하기 위해 각 제어 루프의 태스크와 메시지의 관계식인 제약식을 태스크 그래프를 이용하여 유도한다.

제약식을 유도하는데 있어 기존 연구[5][6][15]에서 고려하지 못한 제약사항으로서 시스템이 그림 4와 같이 동작할 경우 엑츄에이터의 출력 태스크가 샘플링 태스크 1, 2의 다음 샘플링 시점 이후에 수행되기 때문에 정확하지 않은 샘플링 데이터로 인해 시스템의 성능을 저해할 수 있다. 즉, 제어신호가 플랜트에 영향을 미치기 전에 데이터가 샘플 되기 때문에 샘플된 데이터의 정확도가 떨어지게 되며 이러한 영향은 반복적으로 계속 발생하게 된다. 이러한 데이터 샘플링 적시성을 만족하는 스케줄링을 위해 다음과 같은 제약조건이 필요하다.

제약조건 : 제어 루프 i에서 엑츄에이터 노드의 출력 태스크의 초기시작 시간과 데드라인의 합은 해당 루프에 포함된 샘플링 태스크들의 주기 중 가장 큰 값보다 작거나 같아야 한다.

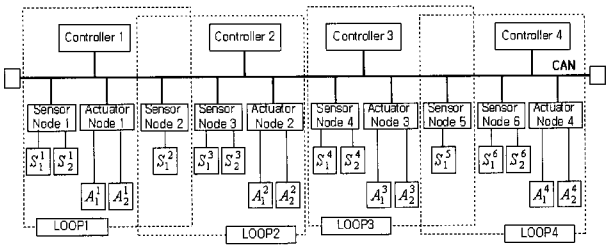


그림 2. 다양한 루프를 갖는 대상 시스템.
Fig. 2. Target system with multiple control loops.

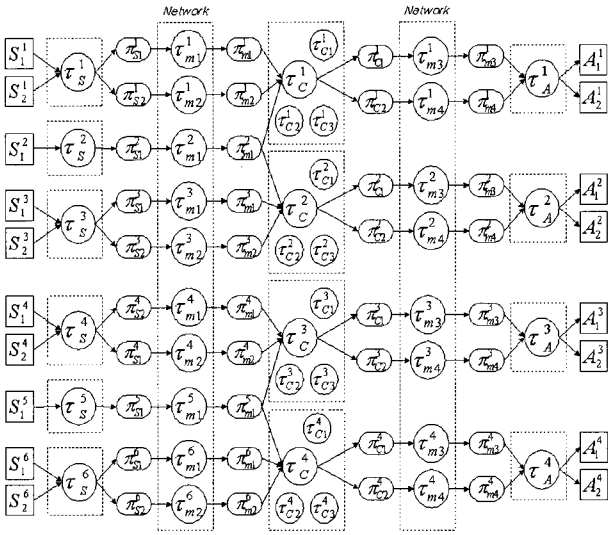


그림 3. 그림 2의 태스크 그래프.
Fig. 3. Task graph of Fig. 2.

$$\phi^i_{actuator} + d^i_{actuator} \le \max_{\forall j \in SetOfSample(i)}(T^i_j)$$

여기서 SetOfSample(i)는 제어 루프 i에 포함되어 있는 모든 샘플링 태스크들의 집합이다.

시스템 요구사항 및 설계 구조에 의해서 유도되는 여러 가지 제약들은 주기에 대한 제약식과 데드라인과 초기시작 시간에 대한 제약식 두 집합으로 이루어지며 제어 루프 1의 주기에 대한 제약식 집합은 다음과 같다.

- 센서 노드에서 메시지 : $T^1_S = T^1_{m1}$, $T^1_S = T^1_{m2}$, $T^2_S = T^2_{m1}$
- 메시지에서 제어 노드 : $T^1_{m1} | T^1_C$, $T^1_{m2} | T^1_C$, $T^2_{m1} | T^1_C$
- 제어 노드에서 메시지 : $T^1_C = T^1_{m3}$, $T^1_C = T^1_{m4}$
- 제어 노드에서 엑츄에이터노드: $T^1_{m3} | T^1_A$, $T^1_{m4} | T^1_A$

제어 루프 1의 초기 시작시간과 데드라인에 대한 제약식 집합은 다음과 같다.

- 양극단 시간 제약 :

$$- \phi^1_A - \phi^1_S + d^1_A \le MADT^1$$

$$- \phi^1_A - \phi^2_S + d^1_A \le MADT^1$$

$$\Rightarrow \phi^1_A - \min(\phi^1_S, \phi^2_S) + d^1_A \le MADT^1 \quad (1)$$

표 1. 그림 3의 표기법.
Table 1. Notations in Fig. 3.

S^i_j	The j-th sensor in the i-th sensor node (i=1 ~ 6, j=1,2)
τ^i_s	The sampling server task in the i-th sensor node (i=1 ~ 6)
π^i_{sj}	Port for transmitting the j-th sensor value in i-th sensor node (i=1 ~ 6, j=1,2)
τ^i_{mj}	Virtual communication task for π^i_{sj}
π^i_{mj}	Port for the j-th message from i-th sensor node (j=1,2, i=1~6) Port for the j-th message from i-th control node (j=3,4, i=1~4)
τ^i_c	The control task in the i-th control node (i=1~4)
τ^i_{c1}	Sporadic task in the i-th control node (i=1~4)
τ^i_{cj}	Periodic task in the i-th control node (i=1~4, j=2,3)
π^i_{cj}	Port for transmitting the j-th control value in i-th control node (i=1~4, j=1,2)
τ^i_A	The actuator task in the i-th actuator node (i = 1~4)
A^i_j	The j-th actuator in the i-th actuator node (i, j =1~4)

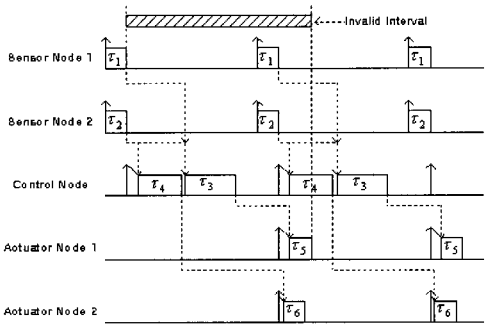


그림 4. 샘플된 데이터의 적시성.
Fig. 4. Timeliness of sampled data.

- 샘플링 적시성 제약 :

$$- \phi^1_A + d^1_A \le \max(T^1_S, T^2_S) \quad (2)$$

- 센서 노드에서 제어 노드 :

$$- \phi^1_C \ge \phi^1_S + d^1_S + \max(d^1_{m1}, d^1_{m2})$$

$$- \phi^1_C \ge \phi^2_S + d^2_S + d^2_{m1}$$

$$\Rightarrow \phi^1_C = \max(\phi^1_S + d^1_S + \max(d^1_{m1}, d^1_{m2}), \phi^2_S + d^2_S + d^2_{m1}) \quad (3)$$

- 제어 노드에서 엑츄에이터노드 :

$$- \phi^1_A = \phi^1_C + d^1_C + \max(d^1_{m3}, d^1_{m4}) \quad (4)$$

이들 식에서 미리 주어지는 값들은 제어 루프의 양극단 시간제약 $MADT^i$와 모든 태스크들의 수행시간, e^i_j 뿐이며 주어진 값들과 3.3절과 3.4절에서 제안된 방법을 통해 모든 ϕ^i_j와 d^i_j를 구하여야 한다. 2, 3, 4번째 제어 루프의 제약식 집합들도 위와 같은 방법으로 유도된다.

3. 우선순위 할당

유도된 각 루프의 제약식 집합을 풀기 위해서는 모든 태스크와 메시지에 우선순위를 할당하여야 한다. 왜냐하면 태스크와 메시지의 우선 순위는 각 제어 루프의 최악응답시간에 영향을 주기 때문이다. 본 논문에서는 제어 노드 상의 태스크 우선순위를 표 3에서와 같이 임의로 설정하였다. 만약 이러한 설정값을 기반으로 제어 루프의 양극단 시간 제약을 만족하는 스케줄링 결과를 찾을 수 없다면 제어 노드 상의 태스크 우선순위는 휴리스틱한 방법으로 조정되어야 한다.

메시지의 우선순위 경우에는 특정 메시지의 우선순위 변화로 인한 제어 루프의 양극단 최악응답시간의 영향을 분석하여 이를 통해 적절한 메시지 우선 순위 할당 규칙을 있음을 알 수 있었다. 우선 순위 변화대상의 메시지로는 제어 태스크가 수행되기 전에 앞서서 제어 태스크에 송신되어야 할 메시지 A(τ^1m1)와 제어 태스크가 수행된 뒤에 엑츄에이터로 송신되는 메시지 B(τ^1m3), 두 가지 이상의 제어 루프에 영향을 미치는 메시지 C(τ^2m1)로 총 3가지의 서비스 특성을 갖는 메시지에 대하여 분석하였다.

첫 번째의 경우 그림 5와 같이 메시지 A의 우선 순위가 증가함에 따라 메시지 A를 포함하는 루프의 응답시간이 증가함을 알 수 있다. 즉, 메시지의 우선 순위가 낮아짐에 따라 제어 루프의 양극단 최악응답시간이 증가하게 된다.

그림 6에서 보는 바와 같이 두 번째 경우도 첫 번째 경우와 마찬가지로 메시지 B의 우선순위가 낮아지면 제어 루프의 양극단 최악응답시간도 증가됨을 알 수 있으나 영향의 시점이 우선순위 수준 8에서 지연되어 나타남을 알 수 있다.

표 2. 메시지 우선순의 할당 결과.
Table 2. Message priority assignment result.

LOOP	MADT	100Kbps				500Kbps				1Mbps			
		β =1		β =2		β =1		β =2		β =1		β =2	
		T	W	T	W	T	W	T	W	T	W	T	W
1	60	40	36	45	40	25	24	30	25	25	23	25	23
2	80	60	57	60	53	35	32	35	31	30	28	30	28
3	100	80	77	90	79	40	37	40	37	35	33	35	33
4	120	115	98	100	89	40	39	40	39	35	33	35	33

* α=0.1, γ=0

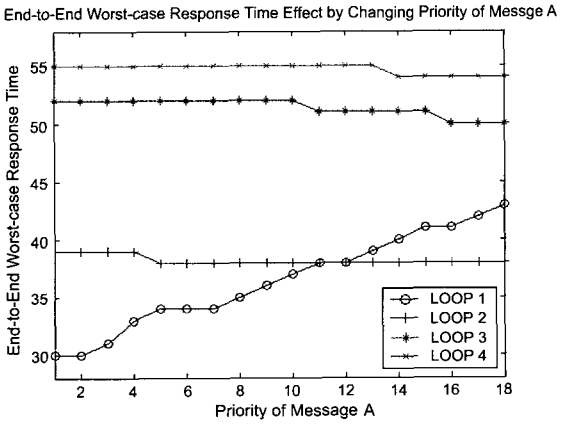


그림 5. 메시지 A의 경우.
Fig. 5. Case of message A.

세 번째의 경우인 그림 7에서는 메시지 C의 우선 순위 변화가 메시지 C를 공유하는 양 루프의 응답시간을 모두 증가시키는 것을 알 수 있다. 또한 세 경우에 대하여 메시지 A를 사용하지 않는 제어 루프의 양극단 최악응답시간 특성은 메시지 A의 우선 순위에 거의 영향을 받지 않는다.

이러한 결과를 통해서 다음과 같이 3가지 휴리스틱한 우선순위 할당 규칙을 제안하며 이에 대한 알고리즘이 Algorithm 1에 나타나 있다.

1) 제어 루프의 주기가 짧을수록 해당 루프에 포함된 메시지의 우선순위를 높게 설정한다.

2) 제어 태스크의 입력 메시지들의 우선 순위는 제어 태스크의 출력 메시지들의 우선 순위 보다 높아야 한다.

3) 여러 루프에 중첩되어 사용하는 메시지의 경우 이를 사용하는 루프에 모두 영향을 미치므로 우선 순위를 높게 설정해야 한다.

제안된 메시지 우선순위 할당 알고리즘은 3.4절의 주기할당과 제약식 풀기의 과정 후에나 할당된 메시지 우선순위가 제어 루프의 양극단 응답시간에 어떤 영향을 줄 수 있는지 알 수 있다. 표 2에는 α, β, γ 상수값에 따라 설정된 메시지 우선순위가 3.4절의 과정을 마친 후에 각 제어 루프의 적용 가능한 최소 주기와 양극단 최악 응답시간 계산 결과가 나타나 있다. 표 2상에는 CAN의 전송속도에 따라 결과가 나누어

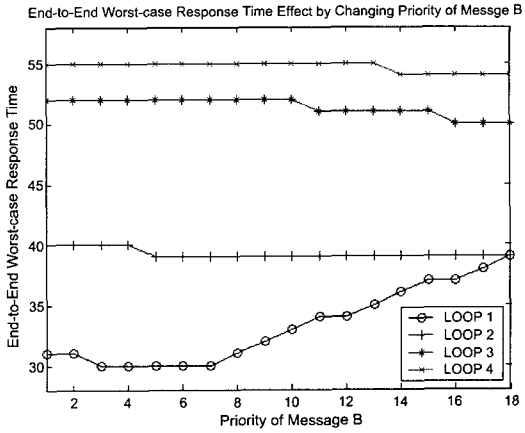


그림 6. 메시지 B의 경우.
Fig. 6. Case of message B.

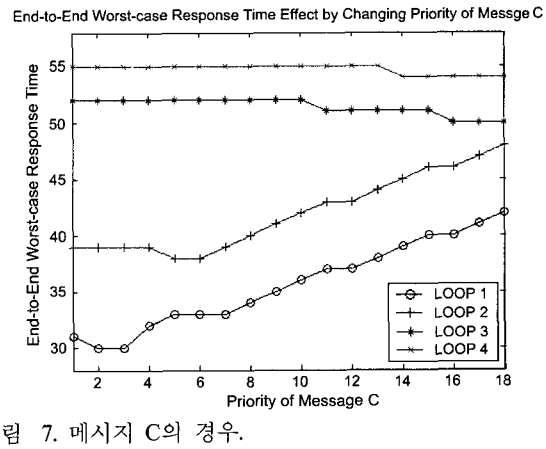


그림 7. 메시지 C의 경우.
Fig. 7. Case of message C.

Algorithm 1. Message Priority Assignment

/* W^i_{mj} is the priority weight of τ^i_{mj} */

/* α, β, γ are constants to be adjusted */

/* $loop(\tau^i_{mj})$ is the set of loops including τ^i_{mj} */

/* $NumsOfTasks(\tau^i_{mj})$ is the number of tasks that use τ^i_{mj} */

$$W^i_{mj} = 0$$

$$W^i_{mj} = W^i_{mj} + \alpha \times \{\max_{\forall k}(MADT^i) - \min_{\forall l \in loop(\tau^i_{mj})}(MADT^l)\}$$

if τ^i_{mj} is the input message of the control task

$$W^i_{mj} = W^i_{mj} + \beta$$

$$W^i_{mj} = W^i_{mj} + \gamma \times NumsOfTasks(\tau^i_{mj})$$

Set the priority of τ^i_{mj} according to W^i_{mj}

Algorithm 2. Optimal Period Assignment

/* $cl(\tau^i_j)$ is the set of loops including a cross task(τ^i_j) */

/* $\underset{\forall k \in cl(\tau^i_j)}{GCD}(T^k)$ is the greatest common divisor of the loops' periods related to a cross task(τ^i_j) */

/* T^l_{granul} is a given time granularity for period harmonicity */

$$T^l_L = \sum_{\forall \tau_j \in Loop\_l} e_j$$

$$T^i_U = MAVT^i$$

while ($T^i_L - T^i_U > \varepsilon$) { /* if result is not optimal period */

$$T^i_B = \left\lfloor \frac{T^i_L + T^i_U}{2} \right\rfloor$$

if ($\frac{T^i_{granul}}{2} \le T^i_B - \left\lfloor \frac{T^i_B}{T^i_{granul}} \right\rfloor \times T^i_{granul}$) then $T^i = (1 + \left\lfloor \frac{T^i_B}{T^i_{granul}} \right\rfloor) \times T^i_{granul}$

else $T^i = \left\lfloor \frac{T^i_B}{T^i_{granul}} \right\rfloor \times T^i_{granul}$

if (T^i_j is cross task) then $T^i_j = \underset{\forall k \in cl(\tau^i_j)}{GCD}(T^k)$

else $T^i_j = T^i$

/* Solving_Constraints() : */
/* Using (5)&(6), get deadlines of all tasks & messages. */
/* Applying them to (3)&(4), get all initial phase times. */
/* If (1)&(2) are satisfied, then TRUE else FALSE */

schedule = Solving_Constraints( )

if (schedule = TRUE) then $T^i_U = T^i$

else $T^i_L = T^i$

}

져 있으며 T는 제어 루프에 적용된 주기를 나타내며 W는 제어 루프의 양극단 최악 응답시간을 의미한다.

4. 주기할당과 제약식 풀기

각 제어 루프의 제약식 유도와 태스크와 메시지에 우선순위가 할당되면 3.2절에서 유도된 각 제어 루프의 주기에 대한 제약식 집합을 만족하는 태스크 주기를 할당한다. 이러한 태스크 주기할당은 분산 시스템 스케줄링에 있어서 풀어야 할 중요한 문제 중의 하나로서 기존의 연구로는 주기조화성과 단위소거 방법을 통해 탐색영역을 줄여나가는 방식[5][6][15]이 있었으나 줄여진 탐색영역에서 최적의 값을 찾는 방법은 제안되지 못했다.

본 논문에서는 제어 루프의 주기가 짧아질수록 시스템 성능이 좋아질 수 있다는 일반적인 사실에 기초하여 다음의 제어 목적을 달성하는 최적 주기할당 알고리즘을 제안하며 Algorithm 2에 나타나 있다. 제어 목적은 모든 제어 루프들의 양극단 시간제약을 만족하면서 가장 짧은 제어 루프 주기를 구하는 것이다.

제안한 주기할당 알고리즘은 각 제어 루프에 적용 가능한 최대, 최소 주기를 가지고 이진 탐색과 주기에 대한 제약을 만족하는 주기를 찾아내어 각 태스크에 주기를 할당하며 제약식 풀기-Solving_Constraints()-를 수행하여 찾아진 주기가 스케줄 가능한지 판단하게 된다. 이러한 과정의 반복 계산을 통해 적용 가능한 최소주기를 탐색하다.

제약식 풀기의 과정은 태스크의 주기할당 후에 초기 시작시간과 데드라인에 관한 제약식을 풀기 위하여 태스크와 메시지의 데드라인을 구하게 된다. 태스크 기반 스케줄링 방법은 제어 루프를 구성하는 모든 태스크와 메시지의 데드라인의 합이 양극단 시간제약을 만족하는지 검사한다. 기존에는 태스크의 데드라인을 태스크의 주기로 설정하였지만 태스크의 데드라인이 큰 값이 될수록 스케줄링 분석 시에 양극단 최악응답시간이 커지게 되므로 양극단 시간제약을 만족하는 스케줄링 결과를 얻을 확률이 적어지게 된다. 그러므로 태스크와 메시지의 데드라인을 좀 더 정확하게 구할 수 있는 방법이 필요하다. 본 논문에서는 태스크와 메시지의 정확한 데드라인을 구하기 위해서 아래의 (5)와 (6)을 사용한다.

태스크간 독립적이고 고정된 우선 순위 사용하는 선점형 스케줄링 시스템의 경우 제어 루프 i의 j번째 태스크 τij의 최악응답시간, Rij는 다음과 같다[1].

$$R^i_j = B^i_j + \sum_{\forall k \in hp(\tau^i_j)} \left\lceil \frac{R^i_j}{T^i_k} \right\rceil \times e^i_k + e^i_j \tag{5}$$

여기서 B^i_j는 태스크 τ^i_j가 자신보다 낮은 우선 순위를 갖는 태스크의 공유자원 사용으로 인해 블록킹되는 최대 시간이며 $hp(\tau^i_j)$는 태스크 τ^i_j 보다 높은 우선 순위를 갖는 태스크들의 집합이다.

CAN 상에서 메시지 i의 최악응답시간, M_i는 다음과 같다[12].

$$M_i = B_{mi} + \sum_{\forall j \in hp(m_i)} \left\lceil \frac{M_i}{P_{mj}} \right\rceil \times C_{mj} + C_{mi} \tag{6}$$

여기서 B_{mi}는 메시지 i가 자신보다 낮은 우선 순위를 갖는 메시지에 의해 지연되는 최대시간이고 P_{mj}는 메시지 j의 주기, C_{mi}는 메시지 i의 전송시간, $hp_{(mi)}$는 메시지 i보다 높은 우선 순위 갖는 메시지 집합이다.

(5)와 (6)에 의해 구해진 최악응답시간은 태스크나 메시지가 수행 또는 전송되는데 있어서 최대로 지연되어 완료되는 시간이므로 태스크와 메시지의 데드라인으로 사용할 수가

있다. 그러므로 R^i_j를 태스크 τ^i_j의 데드라인 d^i_j으로, M_i를 해당하는 가상 네트웍 태스크의 데드라인으로 사용한다.

(5)와 (6)을 통해 구해진 태스크와 메시지의 데드라인을 이용하여 3.2절의 (3)과 (4)를 풀게 되고 얻어진 결과가 (1)과 (2)를 만족한다면 이때 적용된 태스크와 메시지의 주기는 스케줄 가능한 주기값으로 판단된다. 이러한 주기할당과 제약식풀기의 과정을 통하여 양극단 시간제약을 만족하면서 적용 가능한 가장 짧은 제어 루프 주기가 될 때까지 반복적으로 재 계산되어 최종적인 스케줄링 결과를 얻게 된다.

본 논문의 3.1절부터 3.4절에 걸쳐 제안된 통합 스케줄링 방법의 구조가 그림 8에 나타나있다. 대상 시스템을 구성하는 CAN의 전송속도가 100Kbps와 500Kbps일 때를 고려하여 제안된 스케줄링 방법을 적용하여 얻어진 각 제어 루프의 최적주기 할당의 결과가 각각 그림 9와 그림 10에 나타나있다. 보여지는 결과에서처럼 제안된 방법은 네트워의 전송속도에 관계없이 6번의 반복계산 이내에 제어 루프의 주기가 최적이 되도록 스케줄할 수 있음을 알 수 있다. CAN의 전송속도가 500Kbps일 때, 메시지 우선순위 할당을 위한 비례 상수가 α=0.1, β=2, γ=0일 때, 각 제어 루프의 주기는 30, 35, 40, 40으로 최적화되었고 최종적으로 스케줄된 태스크들과 메시지들의 파라미터 값들이 표 3에 나타나 있다.

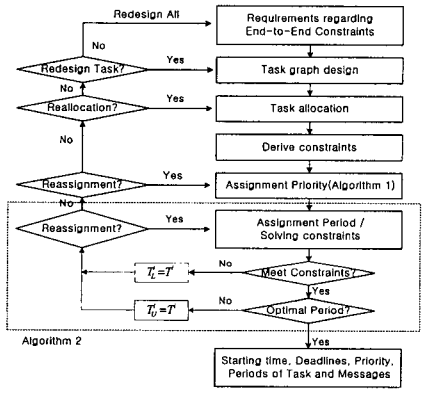


그림 8. 통합 스케줄링의 구조.
Fig. 8. Architecture of integrated scheduling.

표 3. 제안된 스케줄링 적용 결과(500Kbps).
Table 3. Result of proposed scheduling(500Kbps).

Loop 1 (MADT: 60)									Loop 2 (MADT: 80)									Loop 3 (MADT: 100)									Loop 4 (MADT: 120)								
Task	Priority T	Priority M	e	T	R	D	ϕ	Task	Priority T	Priority M	e	T	R	D	ϕ	Task	Priority T	Priority M	e	T	R	D	ϕ	Task	Priority T	Priority M	e	T	R	D	ϕ
τ^1_S	1		2	30	2	2	0	τ^2_S	1		3	35	3	3	0	τ^3_S	1		3	40	3	3	0	τ^4_S	1		3	40	3	3	0
τ^1_{M1}		1	0.2	30	0.4	1	2	τ^2_{M1}		4	0.3	35	0.9	1	3	τ^3_{M1}		10	0.1	40	1.9	2	3	τ^4_{M1}		13	0.1	40	2.5	3	3
τ^1_{M2}		2	0.1	30	0.6	1	3	τ^2_{M2}		5	0.2	35	1.1	2	4	τ^3_{M2}		11	0.2	40	2.1	3	5	τ^4_{M2}		14	0.1	40	2.6	3	6
τ^1_T	1		1	5	1	1	0	τ^2_T	1		1	5	1	1	0	τ^3_T	1		1	40	1	1	0	τ^4_T	1		1	40	1	1	0
τ^1_{M3}		3	0.2	5	0.8	1	1	τ^2_{M3}		3	0.2	5	0.8	1	1	τ^3_{M3}		12	0.2	40	2.3	3	1	τ^4_{M3}		12	0.2	40	2.3	3	1
τ^1_{C1}	1		4	20	4	4	0	τ^2_{C1}	1		4	20	4	4	0	τ^3_{C1}	1		4	20	4	4	0	τ^4_{C1}	1		4	20	4	4	0
τ^1_C	2		11	30	15	15	4	τ^2_C	2		13	35	17	17	6	τ^3_C	2		15	40	19	19	8	τ^4_C	2		14	40	18	18	9
τ^1_{C2}	3		6	50	25	25	0	τ^2_{C2}	3		6	50	27	27	0	τ^3_{C2}	3		6	50	29	29	0	τ^4_{C2}	3		6	50	28	28	0
τ^1_{C3}	4		8	60	33	33	0	τ^2_{C3}	4		8	60	35	35	0	τ^3_{C3}	4		8	60	37	37	0	τ^4_{C3}	4		8	60	36	36	0
τ^1_{M3}		6	0.2	30	1.3	2	19	τ^2_{M3}		8	0.2	35	1.7	2	23	τ^3_{M3}		15	0.1	40	2.8	3	27	τ^4_{M3}		17	0.2	40	3.1	4	27
τ^1_{M4}		7	0.1	30	1.3	2	21	τ^2_{M4}		9	0.1	35	1.8	2	25	τ^3_{M4}		16	0.1	40	2.9	3	30	τ^4_{M4}		18	0.2	40	3.1	4	31
τ^1_A	1		2	30	2	2	23	τ^2_A	1		4	35	4	4	27	τ^3_A	1		4	40	4	4	33	τ^4_A	1		4	40	4	4	35

IV. 결 론

본 논문에서는 분산 제어 시스템을 위한 태스크와 메시지 기반의 통합 스케줄링 방법을 제안하였고 예제를 통하여 제안된 방법에 대한 유효성을 보였다. 제안된 통합 스케줄링 방법은 CAN 기반 하에서 여러 개의 제어 루프로 구성되고 각 제어 루프는 여러 개의 센서 노드, 엑츄에이터 노드와 여러 종류의 태스크가 수행되는 제어 노드로 구성되는 일반적인 분산 제어 시스템에 적용하였으며, 제안된 통합 스케줄링 방법을 통하여 양극단 시간제약 및 선행 제약을 만족하는 초기시작시간, 태스크 주기, 메시지 우선순위, 태스크와 메시지 데드라인을 쉽게 찾아낼 수 있었다.

초기 시작시간을 사용하는 기존의 태스크 기반 스케줄링에서 발생 가능한 샘플링과 엑츄에이터 구동간의 동기화 문제를 해결하였으며 분산 제어 시스템을 구성하는 각 제어 루프가 양극단 시간제약을 만족하면서 가장 짧은 제어 루프 주기를 가질 수 있도록 태스크의 최적 주기할당 알고리즘을 제안하였다. 또한 메시지의 우선순위 변화에 따른 각 제어 루프의 최악응답시간 영향을 분석을 통하여 메시지 특성에 따른 우선순위 할당 규칙을 제안하였다. 최적 주기할당 방법과 우선순위 할당 규칙을 통하여 분산 제어 시스템을 구성하는 각 제어 루프의 양극단 최악응답시간을 최소화할 수 있는 최적 주기와 우선순위를 구할 수 있었다.

앞으로 연구해야할 내용으로는 제안된 스케줄링 알고리즘을 다양하고 복잡한 시스템에 적용하는 것과 Optimality를 보장할 수 있는지를 검증하는 것이 있다.

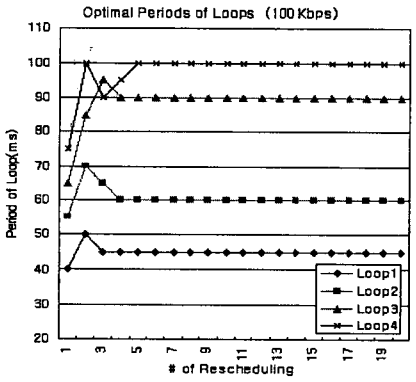


그림 9. 알고리즘 1의 결과(100Kbps).
Fig. 9. Result of algorithm 1(100Kbps).

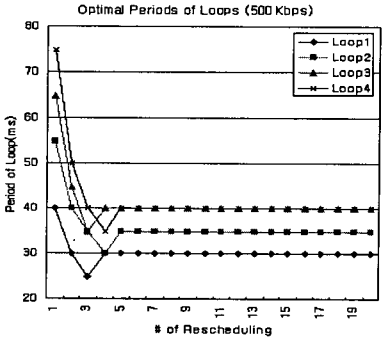


그림 10. 알고리즘 1의 결과(500Kbps).
Fig. 10. Result of algorithm 1(500Kbps).

참 고 문 헌

[1] A. Burns, "Preemptive priority based scheduling : An appropriate engineering approach in Principles of real-time systems," *Prentice Hall*, 1994.

[2] K. Ramamritham and J. A. Stankovic, "Scheduling algorithm and operating systems support for real-time systems," *Proc. of IEEE*, pp. 55-67, Jan., 1994.

[3] N. C. Audsley, A. Burns and A.J. Wellings, "Deadline monotone scheduling theory and application," *Control Eng. Practice*, vol. 1, no. 1, pp. 71-78, 1993.

[4] J. Xu and D. Parnas, "Scheduling processes with release times, deadlines, precedence and exclusion relations," *IEEE Tr. on Software Engineering*, pp. 360-369, March, 1990.

[5] R. Gerber and S. S. Hong, "Guaranteeing real-time requirements with resource-based calibration of periodic processes," *IEEE Tr. on Software Engineering*, 21(7), July, 1995.

[6] 홍성수, 최종호, 박홍성, "주기조정과 커널 자동 생성을 통한 다중 루프 시스템의 구현," 제어 · 자동화 · 시스템공학 논문지, 제3권, 제2호, pp. 187-196, 4, 1997.

[7] J. Y.-T Leung and J. Whitehead, "On complexity of fixed-priority scheduling of periodic real-time tasks," *Performance Evaluation*, 2(4), pp. 237-250, December, 1982.

[8] F. Vasques and G. Juanole, "Pre-run-time schedulability analysis in fieldbus networks," *IEEE*, 1994.

[9] P. Lorenz and Z. Mammeri, "Real-time software architecture : Application to FIP filedbus," *Proc. of AARTC*, pp. 415-423, 1995.

[10] M. J. Johnson, "Proof that timing requirements of the FDDI token ring protocol are satisfied," *IEEE Tr. on Communications*, COM-35, no. 6, June, 1987.

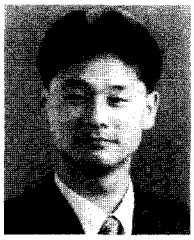
[11] B. Chen and W. Zhao, "Properties of the timed token protocol," *Technical Report 92-038, Oct., Dept. of Computer Science, Taxas A&M Univ.*, 1992.

[12] K. Tindell, H. Hansson, and A. Wellings, "Analyzing real-time dommunications : Controller area network," *IEEE Real-time Systems Symposium*, 1994.

[13] S. Faucou, A.-M. Deplanche and J.-P. Beauvais, "Heuristic techniques for allocating and scheduling communicating," *WFCS-2000*, pp. 257-265, 2000.

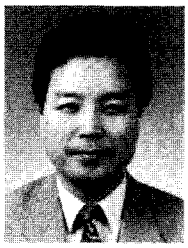
[14] K. Tindell, "Holistic schedulability analysis for distributed hard real-time systems," *Technical Report, YCS-197, Dept. of Computer Science Univ. of York*, Nov., 1994.

[15] J. W. Park, Y. S. Kim, S. S. Hong, M. Saksena, S. H. Noh and W. H. Kwon, "Network conscious of distributed real-time systems," *Journal of System Architecture*, pp. 131-156, 1998.



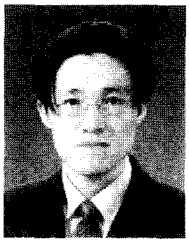
김 형 육

1973년 7월 16일생. 1999년 강원대 제어계측공학과 졸업. 동 대학원 석사(2001). 2001년~현재 강원대 제어계측공학과 박사과정. 관심분야는 실시간 스케줄링, 필드버스, 무선데이터 통신 등.



박 홍 성

1961년 3월 16일생. 1983년 서울대 제어계측공학과 졸업. 동 대학원 석사(1986). 동 대학원 박사(1992). 1992년~현재 강원대 전기전자정보통신공학부 부교수. 관심분야는 실시간 네트웍, 무선 네트웍 등.



이 철 민

1968년 12월 2일생. 1993년 강원대 제어계측공학과 졸업. 동 대학원 석사(1997). 1997년~현재 강원대 제어계측공학과 박사과정.