분산 제어시스템에서 3가지 형태의 실시간 데이터를 고려하는 양극단 스케줄링 방법

End-to-End Scheduling Method Considering 3-type RT-Data in Distributed Control Systems

김 형 육*, 박 홍 성**
(Hyoung Yuk Kim and Hong Seong Park)

*강원대학교 통신및멀티미디어공학과(전화:033)251-6501, 팩스:033)242-2059, E-mail:petrus@control.kangwon.ac.kr)
** 강원대학교 전기전자정보통신공학부(전화:033)250-6346, 팩스:033)242-2059, E-mail : hspark@cc.kangwon.ac.kr)

Abstract : In recent years, 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 considering aperiodic, periodic and non-real-time message and task simultaneously. 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.

Keywords : End-to-End Scheduling, Real-Time, DCS.

Ⅰ. 서 론

현재의 제어시스템은 내장형 프로세스가 장착된 스마트 센서 및 구동기가 Profibus, FIP, CAN, LonWorks 등과 같은 필드버스를 통해 제어알고리즘을 수행하는 제어노드 또는 제어기와 연결되는 구조가 일반적이다. 그러므로 점대점 연결방식을 통해 구성되던 제어시스템과는 달리 센서, 구동기, 제어기가 서로 분산되어질 수 있기 때문에 센서에서 제어기, 제어기에서 구동기로의 데이터 전송이 네트웍 특성에 따라 임의 지연시간을 갖게 된다. 또한 노드의 특성 및 노드에서 수행되는 태스크 수와 태스크 간의 관계에 따라 각 노드에서 수행되는 태스크의 기동에서 수행완료까지의 지연시간 또한 일정치 않다. 그러므로 네트웍 지연 및 각 노드에서의 태스크 수행지연 등을 고려하여 시스템을 설계하여야 한다[1-3].

이러한 분산 제어시스템의 설계와 구현은 시간제약조건들, 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 제어시스템을 구성하는 각 제어루프의 실시간 특성이 매우 중요하기 때문이다.

이러한 문제를 해결하기 위한 연구로서, [4] 에서는 어떠한 태스크도 두 개 이상의 메시지를 받지 않는다는 가정 하에서 TDMA를 사용하는 분산 제어시스템을 위한 휴리스틱(heuristic) 스케줄링 방법이 제안되었다. 또한, 태스크 기반의 스케줄링 방법[5]을 CAN으로 구성되는 분산 제어시스템으로 확장한 연구[6]에서는 여러 가지 소거법을 통하여 분산 제어 시스템을 위한 스케줄링 방법을 제안하였다. [7] 은 클러스터링 알고리즘과 유전적 알고리즘을 사용하여 CSMA/CA와 TDMA 방식의 네트워크로 구성되는 분산 시스템을 위한 스케줄링 방식을 제안하였다. 그러나 이러한 연구들은 주기실시간 태스크 및 주기실시간 메시지에 대해서만 스케줄링 방법을 제안함으로써 실제로 분산제어시스템 상에 존재하는 비주기 실시간 태스크 및 메시지와 비실시간 태스크 및 메시지가 함께 고려되어야 할 경우에 적용에 문제점을 갖게 된다.

본 논문에서는 분산 제어 시스템의 메시지 우선순위 설정 방법과 태스크 주기할당 방법을 제안하고 양극단에서 스케줄링을 연구한 [8]의 연구를 보완하여 비주기 실시간 태스크, 주기 실시간 태스크, 비실시간 태스크와 비주기 실시간 메시지, 주기 실시간 메시지, 비실시간 메시지를 모두 함께 고려하여 분산 제어시스템을 스케줄링하는 방법을 제안하고 시뮬레이션을 통해 제안된 방법을 검증한다.

2절에서는 3가지 실시간 데이터를 고려하여 스케줄링하기 위한 방법을 제안하며 3절에서는 대상 시스템에 제안된 방법을 적용하여 결과를 보이고 끝으로 4절에서 결론을 맺는다.

II. 3가지 메시지 및 태스크를 위한 스케줄링

1. 추가된 가정

기존 논문[8]에서의 가정 및 환경은 센서노드와 구동노드 상에 주기태스크만 수행되며 제어기 상에 제어 알고리즘을 수행하는 주기태스크와 기타 비주기 태스크가 수행된다는 가정에서 서술되었다. 또한 태스크 수행시간 분석을 단순화하기 위해 태스크간 공유자원의 사용을 배제하였다. 네트웍 상으로 전송되는 메시지는 주기 메시지들만 존재한다고 가정하였고 모든 노드 또는 스테이션들은 preemptive 기반으로 동작한다고 가정하였다. 비주기 실시간 데이터 및 태스크, 주기 실시간 데이터 및 태스크, 비실시간 데이터 및 태스크의 3가지 경우를 고려하여 스케줄링을 제안하는 본 논문에서는 센서 노드와 구동노드에도 비주기 및 비실간 태스크가 존재하며 네트웍 상의 메시지도 비주기 실시간 메시지와 비실시간 메시지도 존재한다고 가정한다.

2. 태스크 그래프 수정

비주기 실시간 및 비실시간 태스크를 고려하여 스케줄링하기 위해서는 태스크 그래프에 이를 반영하도록 하여야 하므로 아래와 같은 방법으로 노드별로 태스크가 추가될 수 있다.

2.1 센서노드

센서노드 자신에게 연결된 다수의 센서들을 차례로 샘플링한다. 이를 위해 주기적으로 수행되는 주기 실시간 형태의 샘플링 태스크가 존재한다. 이때, 샘플링은 주기적으로만 수행된다고 가정한다. 어떤 경우에서는 센서들을 항상 주기적으로 샘플링하지 않고 제어기 또는 임의의 소비자가 해당 센서의 값을 알고자 원할 때만 샘플링하는 비주기적인 샘플링 태스크도 존재할 수 있다. 즉, 주기적인 샘플링 태스크와 비주기적인 샘플링 태스크가 공존할 수 있다. 그러나 이러한 경우는 모델을 단순화하기 위해서 배제한다. 샘플링 태스크는 샘플링된 데이터를 제어기에게 전송한다. 이 메시지는 주기 실시간 메시지 형태가 된다. 샘플링 태스크는 센서의 값이 시스템의 Critical Fail을 유발하거나 유발할 수 있는 범위라면 이에 대한 Alarm 신호를 보내준다. 즉, 샘플링 태스크는 Normal한 경우의 주기 실시간 메시지와 Critical한 경우의 비주기 실시간 메시지를 전송하게 된다. 센서노드는 Watch Dog Timer와 같은 자신의 시스템 오동작을 방지하기 위한 메커니즘을 제공한다. 그러므로 Watch Dog Time out 시에는 자신의 오동작에 대한 정보를 네트웍으로 전송해야 한다. 이를 담당하는 태스크는 비주기 실시간 태스크이다. 해당 센서의 오동작에 관련된 메시지를 수신하는 노드는 제어기만이라고 가정한다. 제어기는 이 메시지를 수신해서 적절한 에러 처리루틴을 수행하고 구동기로 메시지를 전송한다. 근래의 스마트 센서들은 센서노드의 환경정보 및 업데이트가 원격에서 가능하다. 이러한 기능은 로컬에서 1:1 연결 인터페이스를 통해 이루어질 수도 있지만 원격에서도 이루어질수 있으므로 Ethernet과 같은 네트웍 인터페이스를 통해 제어기로 명령이 전달되고 제어기는 분산제어스템을 연결하고 있는 필드버스를 이용하여 센서노드의 정보를 읽어오게 된다. 이러한 작업을 담당하는 태스크는 비주기 비실시간 태스크이다. 위와 같은 센서노드의 동작환경을 가정할 때 그림1과 같이 센서노드에 대한 태스크 그래프를 나타낼 수 있다.

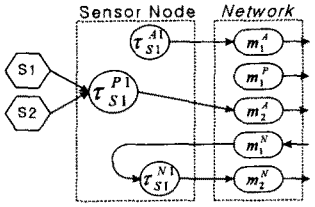

그림 1. 센서노드 모델

Fig. 1. Sensor Node Model

2.2 구동노드

구동노드는 자신에게 연결된 다수의 구동기를 구동한다. 이를 위한 구동태스크가 존재한다. 구동기를 구동하기 구동태스크는 메시지가 수신되면 해당 I/O로 값을 쓰게 된다. 이 작업은 주기적으로 메시지의 수신 버퍼를 검색하여 수행될 수도 있으나 빠른 응답을 위해 메시지 도착 시 수행되는 비주기적으로 수행된다고 가정한다. 또한, 알람 메시지와 같은 비주기 실시간 메시지에 대한 처리를 빠르게 수행하기 위해서는 비주기 실시간 형태의 태스크가 알맞다. 그림 2(a) 에서 처럼 알람 메시지 처리를 위한 비주기 실시간 태스크가 따로 존재한다면 주기 실시간 메시지에 따른 구동기 조작을 방지하기 위한 태스크간 동기화 문제를 고려해야 하므로 구현이 복잡해질 수 있다. 그러므로 구동태스크를 비주기 실시간 태스크로 수행시키고 이 태스크에서 알람 메시지까지 같이 처리한다면 구현뿐 아니라 해석이 단순화되게 된다. 구동노드는 센서노드처럼 Watch Dog Timer와 같은 자신의 시스템 오동작을 방지하기 위한 메커니즘을 제공하며 원격에서 환경정보 읽기 및 업데이트가 가능해야 한다. 위와 같은 구동노드의 동작환경을 가정할 때 그림 2(b)와 같이 구동노드에 대한 태스크 그래프를 나타낼 수 있다.

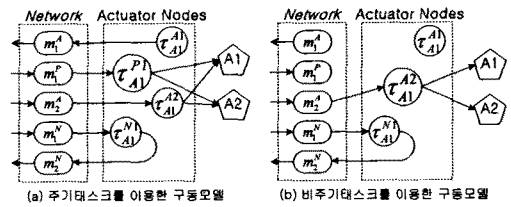

그림 2. 구동노드 모델

Fig. 2. Actuator Node Model

2.3. 제어노드 모델

제어노드는 센서노드로부터 다수의 센서 데이터를 수신하여 제어알고리즘을 수행한 후 다수의 구동노드로 제어값을 전송한다. 제어알고리즘 수행 태스크는

주기 실시간 태스크이다. 제어노드는 센서노드로부터 알람 메시지 수신하여 처리한 후 적절한 에러 대응을 위해 구동노드로 알람 메시지를 전송한다. 이 작업을 수행하는 태스크는 비주기 실시간 태스크이며 전송하는 메시지 또한 비주기 실시간 메시지이다. 제어노드는 센서노드 또는 구동노드의 시스템 오동작으로 인한 메시지를 처리한다. 센서노드와 제어노드의 Watch Dog Timer의 timeout과 같은 오동작 발생 시 각 노드는 이를 제어노드에 알리며 제어노드는 이 메시지를 수행하여 적절한 에러 대응 루틴을 수행한다. 이 작업을 수행하는 태스크는 비주기 실시간 태스크이다. 제어노드는 자신의 환경정보 및 업데이트가 원격에서 가능할 뿐만 아니라 센서노드 또는 구동노드의 환경정보 읽기 및 업데이트를 GW 역할을 한다. 이밖에 제어노드에는 다양한 태스크들이 존재한다. 위와 같은 제어노드의 동작환경을 가정할 때 그림 3와 같이 제어노드에 대한 태스크 그래프를 나타낼 수 있다.

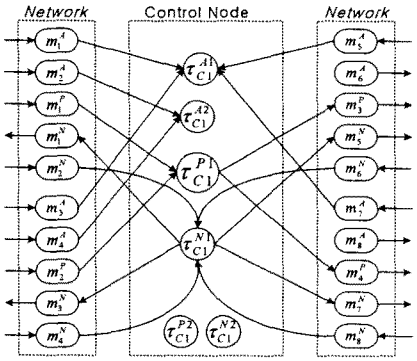

그림 3. 제어노드 모델

Fig. 3. Controller Node Model

3. 양극단 스케줄링

기존 연구를 기반으로 3가지 형태의 태스크와 메시지를 고려하여 스케줄링하기 위해서 기존에 제안된 방법은 주기메시지와 태스크만을 고려한 방법이므로 태스크와 메시지 우선순위 설정 알고리즘과 주기설정 알고리즘에 보완이 필요하다.

주기설정 방법은 태스크의 경우 비주기 태스크와 비실시간 태스크는 주기를 갖고 있는 않으므로 분석을 위해서는 가상의 주기를 설정해야 한다. 이를 위해서 비주기 실시가 태스크의 경우 데드라인을 주기로 설정한다. 비실기간 태스크는 정해지는 데드라인이 없으므로 임의의 값을 설정하도록 한다. 태스크의 주기가 설정되면 메시지의 주기는 해당 메시지의 생산자 태스크의 주기와 같기 때문에 비주기 실시간 메시지나 비실시간 메시지의 주기 또한 설정이 가능해 진다.

우선순위 설정은 하나의 노드 상에서 다수의 태스크가 존재하는 경우 RM 기반의 우선순위 설정 방법으로 매우 간단히 적용할 수 있다. 메시지의 우선순위는 기존에 제안된 메시지 우선순위 설정 방법을 통해 할당한다.

Ⅲ. 대상 시스템 적용

본 절에서는 2절에서 언급한 기존 연구 확장 내용을 대상 시스템에 적용한다. 그림 4과 같이 시스템 모델을 선정하고 이 시스템에 대한 태스크 그래프는 2절에서 서술된 가정에 따라 그림5와 같이 유도된다.

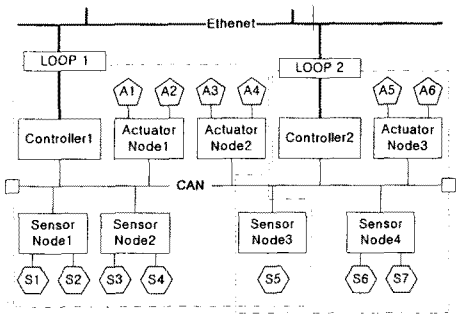

그림 4. 대상 시스템 모델

Fig. 4. Target System Model

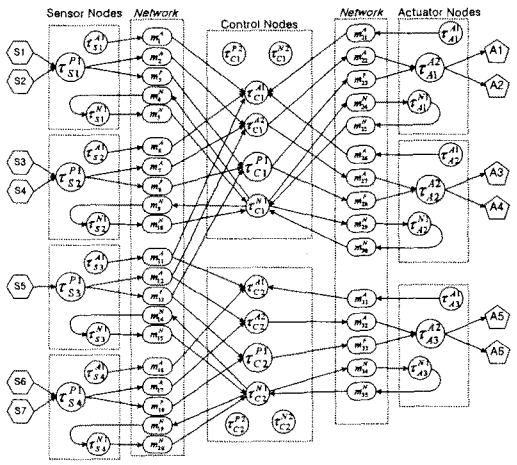

그림 5. 시스템의 태스크 그래프

Fig. 5. Task Graph of Target System

태스크 그래프 유도를 통해 아래와 같은 제약식을 유도할 수 있다.

1) 제어루프 1

- 양극단 응답시간 제약

$$\max(\phi_{S1}^{P1}+d_{S1}^{P1}+d_{m_1^P},\phi_{S2}^{P1}+d_{S2}^{P1}+d_{m_2^P},\phi_{S3}^{P1}+d_{S3}^{P1}+d_{m_3^P})$$
$$+d_{C1}^{P1}+\max(d_{m_5^P}+d_{A1}^{A2},d_{m_6^P}+d_{A2}^{A2})\le MADT^1$$

- 주기에 관한 제약

$$T_{S1}^{P1}=T_{m_1^P}\,|\,T_{C1}^{P1},\qquad T_{S2}^{P1}=T_{m_2^P}\,|\,T_{C1}^{P1},\qquad T_{S3}^{P1}=T_{m_3^P}\,|\,T_{C1}^{P1},$$

$$T_{C1}^{P1}=T_{m_5^P}=T_{m_6^P}$$

- 태스크간 선행제약

$$\max(\phi_{S1}^{P1}+d_{S1}^{P1}+d_{m_1^P},\phi_{S2}^{P1}+d_{S2}^{P1}+d_{m_2^P},\phi_{S3}^{P1}+d_{S3}^{P1}+d_{m_3^P})\le\phi_{C1}^{P1}$$

2) 제어루프 2

- 양극단 응답시간 제약

$$\max(\phi_{S3}^{P1}+d_{S3}^{P1}+d_{m_3^P},\phi_{S4}^{P1}+d_{S4}^{P1}+d_{m_4^P})+d_{C1}^{P1}+d_{m_1^P}+d_{A3}^{A2}\le MADT^2$$

- 주기에 관한 제약

$$T_{S3}^{P1}=T_{m_3^P}\mid T_{C1}^{P1},\ T_{S4}^{P1}=T_{m_4^P}\mid T_{C2}^{P1},\ T_{C2}^{P1}=T_{m_1^P}$$

- 태스크간 선행제약

$$\max(\phi_{S3}^{P1}+d_{S3}^{P1}+d_{m_3^P},\phi_{S4}^{P1}+d_{S4}^{P1}+d_{m_4^P})\le\phi_{C2}^{P1}$$

유도된 제약식을 이용하여 기존 주기할당 알고리즘을 통하여 주기태스크 및 메시지의 주기를 동적으로 할당하고 나서 태스크와 메시지의 우선순위를 할당한다. 이 후 태스크와 메시지의 최악응답시간을 계산하여 각 제어루프의 양극단 최악 응답시간을 얻어낸다.

이에 대한 결과로서 그림6에는 태스크의 우선순위 할당 결과가 그림 7에는 메시지 우선순위 할당결과가 나타나 있다. 그림8에는 주기할당 알고리즘을 통해 설정되는 태스크와 메시지 주기에 따라 제어루프의 양극단 주기와 최악 응답시간이 나타나 있다.

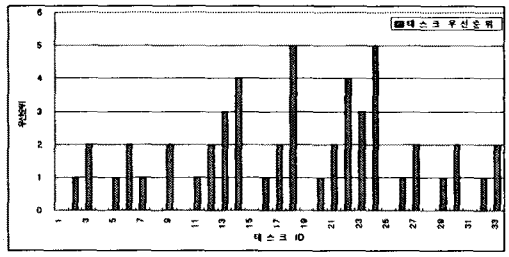

그림 6. 태스크 우선순위 할당 결과

Fig. 6. Allocated priorities of tasks

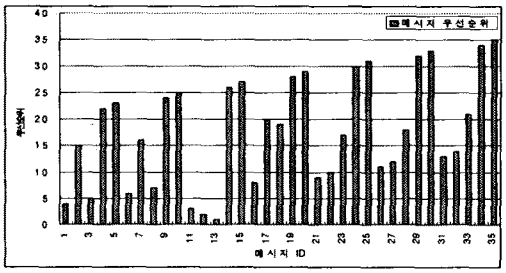

그림 7. 메시지 우선순위 할당 결과

Fig. 7. Allocated priorities of messages

Ⅳ. 결 론

본 논문에서는 다수의 센서와 구동기, 제어기를 통해 여러 개의 제어루프로 구성되는 실제적인 제어시스템을 위한 스케줄링 방법을 제안하였다. 제안된 방법은 기존연구[8]를 보완하여 센서에서 제어기, 제어기에서 구동기로 이어지는 양극단의 태스크와 메시지의 우선순위와 주기를 모두 고려하여 스케줄링하며 비주기 실시간 데이터, 주기 실시간 데이터, 비주기 비실시간 데이터로 나누어지는 3가지 형태의 메시지를 모두 고려하여 스케줄링한다.

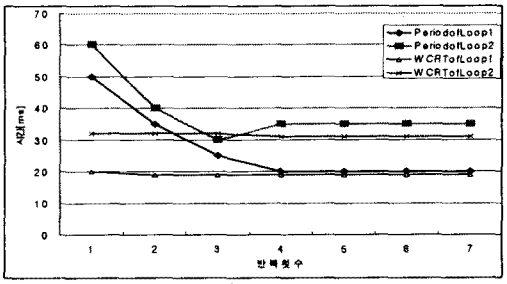

그림 8. 제어루프의 양극단 최악응답시간

Fig. 8. End-to-end worst-case response time of control loops

참고문헌

[1] A. Burns, "Preemptive priority based scheduling: An appropriate engineering approach in Principles of Real-Time Systems," Prentice Hall, 1994.

[2] 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.

[3] 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.

[4] K. Tindell, "Holistic Schedulability Analysis for Distributed Hard Real-time Systems," Technical Report, YCS-197, Dept. of Computer Science Univ. of York, NOV., 1994.

[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] 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.

[7] S. Faucou, A.-M. Deplanche and J.-P. Beauvais, "Heuristic Techniques for Allocating and Scheduling Communicating," WFCS-2000, pp. 257-265, 2000.

[8] 김형욱, 이철민, 박홍성 "분산 제어시스템에서의 태스크와 메시지 기반 스케줄링을 이용한 최적 주기와 우선순위 할당", 제어,자동화,시스템공학 논문지, 제8권, 제6호, pp.506-513, 2002.6

[9] K. Tindell, H. Hansson, and A. Wellings, "Analyzing Real-time Communications: Controller Area Network," IEEE Real-time Systems Symposium, 1994.