CAN기반 분산 제어시스템의 종단 간 지연시간 분석과 협조 스케줄링 알고리즘 개발

論 文
53D-7-5

Development of Coordinated Scheduling Algorithm and End-to-end Delay Analysis for CAN-based Distributed Control Systems

李 喜 培* · 金 弘 烈** · 金 大 元***
(Heebae Lee · Hongryeol Kim · Daewon Kim)

Abstract - In this paper, a coordinated scheduling algorithm is proposed to reduce end-to-end delay in distributed control systems. For the algorithm, the analysis of practical end-to-end delay in the worst case is performed priory with considering implementation of the systems. The end-to-end delay is composed of the delay caused by multi-task scheduling of operating systems, the delay caused by network communications, and the delay caused by asynchronous timing between operating systems and network communications. Through some simulation tests based on CAN(Controller Area Network), the proposed worst case end-to-end delay analysis is validated. Through the simulation tests, it is also shown that a real-time distributed control system designed to existing worst case delay cannot guarantee end-to-end time constraints. With the analysis, a coordinated scheduling algorithm is proposed here. The coordinated scheduling algorithm is focused on the reduction of the delay caused by asynchronous timing between operating systems and network communications. Online deadline assignment strategy is proposed for the scheduling. The performance enhancement of the distributed control systems by the scheduling algorithm is shown through simulation tests.

Key Words : Coordinated scheduling algorithm, End-to-end delay, CAN, Asynchronous timing, Online deadline assignment

1. 서 론

네트워크 기반의 분산 시스템은 다양한 이기종 시스템의 결합을 보장할 뿐만 아니라 이에 따르는 비용을 최소화할 수 있기 때문에 최근 수많은 관련 연구가 진행되고 있다. 또한 최근에는 네트워크 기반 분산 시스템을 이동로봇과 같은 실시간 시스템에 적용하는 연구가 활발히 진행되고 있다[1].

분산 시스템을 실시간 제어 시스템에 적용할 때 발생하는 가장 큰 문제점은 모든 시스템 구성 요소가 네트워크라는 하나의 매체를 공유하여 정보의 전달을 수행하기 때문에 시스템 지연시간이 상존하게 된다는 것이다. 특히 예측 불가능한 네트워크 지연시간은 각각의 분산 시스템 구성 요소의 내부 자원 사용으로 인한 지연시간과 결합하여 전체 시스템의 운영을 실패하게 만드는 요소가 되기도 한다[2].

분산화된 제어 시스템을 구성하는데 있어서 네트워크 프로토콜은 실시간 제어를 가능하게 하는 중요한 요소로서 작용한다. CAN(Controller Area Network)은 고속의 통신 인터페이스를 제공하고 데이터 프레임의 오버헤드(overhead)가 적기 때문에 빠른 응답특성을 갖고 있다. 또한 식별자(identifier)를 이용한 충돌방지와 전송중재(arbitration) 기능을 갖고 있어 우선순위 기반의 실시간 제어 네트워크 프로토콜에 적합한 특성을 갖는다[3].

종단 간 지연시간 분석은 실시간 분산 제어시스템의 소프트웨어 설계 시에 소프트웨어의 시간특성을 결정하는 기준이 된다. CAN을 대상으로 하여 기존에 네트워크 지연시간 분석에 관한 연구가 수행된 바 있으나[4], 최근 실시간 시스템의 일반적인 구현 방법인 다중 태스크 운영체계 환경에 대한 고려가 이루어져 있지 않기 때문에 분석 결과가 구현 특성을 반영하지 못한다는 단점을 갖는다.

따라서 본 논문에서는 CAN을 기반으로 한 분산 제어 시스템 설계를 위해 구현 특성을 고려한 종단 간 지연시간 모델링을 제안한다. 즉, 본 논문에서는 대부분의 실시간 시스템이 다중 태스크 환경의 운영체계를 사용하는 점을 감안하여 네트워크 지연시간의 분석과 함께 운영체계의 지연시간 분석, 그리고 네트워크 시간 특성과 운영체계 시간 특성의 비동기성에 의한 대기 지연시간을 고려하여 실질적인 종단 간 최악의 지연시간 분석을 수행한다. 또한 모의실험을 통해 분석된 결과가 타당한 결과임을 입증하며, 기존의 최악 지연시간 분석 방식에 의해 설계된 분산 제어 시스템이 다중 태스크 운영체계 환경에서 종단 간 지연시간 제약을 만족할 수 없음을 입증한다.

분산 제어시스템이 갖는 특성으로 인해 연산처리나 네트워크와 같은 자원들에 대해 중앙 집중적인 자원 관리를 수행할 수 없으며, 이렇게 분산화 되고 독립적으로 이루어지는 자원 관리는 각각의 자원 이용 시기가 갖는 비동기성으로

* 學生會員 : 明知大 工大 情報工學科 工學碩士
** 正 會 員 : (株) 캐리어 主任硏究員
*** 正 會 員 : 明知大 工大 情報工學科 敎授 · 工博
接受日字 : 2003年 12月 26日
最終完了 : 2004年 4月 6日

인해 종단 간 지연시간을 증가시키는 원인이 된다. 분산 제어시스템과 같이 복수의 자원 사용 시에 발생하는 비동기성을 분석 또는 개선하려는 연구가 수행된 바 있으나[6], 단일 노드(node) 내에서의 개선을 위한 노력으로 CAN과 같이 메시지의 전송중재가 식별자를 통해 시스템 수준에서 이루어지는 경우에는 적합하지 않다는 단점을 가지며, 구체적인 네트워크 프로토콜을 대상으로 한 연구가 수행된 바 없다.

따라서 본 논문에서는 종단 간 지연시간을 최소화 하는 협조 스케줄링 알고리즘을 제안한다. 제안된 알고리즘은 DMS(Deadline Monotonic Scheduling)[7]를 기반으로 하며, 운영체계의 태스크와 네트워크의 메시지 데드라인을 온라인 상에서 가변하여 자원 이용의 비동기성으로 인한 대기 지연시간을 최소화하는 방식을 통해 전체 종단 간 지연시간을 최소화하는 알고리즘이다. 제안된 알고리즘은 CAN을 기반으로 한 실시간 시스템에 적용한 모의실험을 통해 알고리즘을 사용하지 않은 경우에 비해 종단 간 지연시간을 단축시킴으로써 분산 제어시스템의 성능을 향상시킴을 입증한다.

본 논문의 구성은 다음과 같다. 2장 본론에서 종단 간 지연시간의 구성요소에 대해 서술하고, 각각 요소의 지연시간을 분석한 내용을 서술하며 최종적으로 분석된 내용을 종합하여 최악의 종단 간 응답시간 분석을 수행한다. 또한 분석된 종단 간 지연시간 요소를 기반으로 한 협조 스케줄링 알고리즘의 특징과 수행절차에 대해 서술한다. 3장 모의실험 및 결과에서는 모의실험 결과를 통해 입증된 사실들을 서술하고, 마지막으로 4장 결론에서 결론을 맺고 향후 과제를 밝힌다.

2. 본 론

2.1 종단 간 지연시간의 구성 요소

본 논문의 대상이 되는 전체 시스템의 지연시간을 고려한 구성은 그림 1과 같다. 그림 1의 노드1(node1)과 노드2(node2)는 각각 송신 노드와 수신 노드로 표현되어 있지만, 일반적으로 구현되는 노드는 수신과 송신을 위한 메시지 큐(message queue)를 동시에 가지고 있다. 그림 1에서 ①과 ③은 각각 운영체계의 태스크 스케줄링에 의한 지연시간을 의미하고, ②는 CAN의 메시지 스케줄링에 의한 지연시간을 의미한다. 그리고 ④와 ⑤는 네트워크와 운영체계의 비동기성에 의한 대기 지연시간을 의미한다.

본 논문에서는 네트워크 기반 실시간 시스템의 실질적인 종단 간 지연시간 요소를 운영체계의 태스크 스케줄링에 의해 발생하는 지연시간인 운영체계 지연시간, CAN의 메시지 스케줄링에 의해 발생하는 지연시간인 네트워크 지연시간, 그리고 이 두 가지 자원 사용의 비동기적 시간 특성에 의해 발생하는 대기 지연시간 세 가지로 정의하고 분석을 수행한다. 본 논문에서는 네트워크를 통해 전송되는 메시지와 운영체계에서 수행되는 태스크의 스케줄링은 주기적인 태스크 및 메시지 발생을 기반으로 하여 대표적인 정적 스케줄링 방식인 DMS를 사용한다.

송신 노드의 운영체계에서 주기적인 태스크 수행이 이루어진다 할지라도 DMS의 응답시간은 비주기적인 특성을 가지고 있기 때문에 본 논문에서 구성한 시스템은 그림 1과 같이 발생된 메시지를 바로 메시지 큐에 전달하지 않고 일종의 버퍼를 사용하여 대기시키며, 대기 중인 전송 메시지는 짧은 주기를 갖는 전송 서비스 제공 태스크에 의해 CAN의 메시지 스케줄링 주기에 따라 버퍼에서 메시지 큐로 이동하게 된다. 본 논문에서는 이러한 서비스 제공 태스크의 주기를 CAN의 메시지 최소 전송 주기에 비해 아주 작은 값으로 설정할 수 있음을 감안하여, 버퍼에서 메시지 큐로의 이동에 필요한 지연시간은 발생하지 않는 것으로 가정한다.

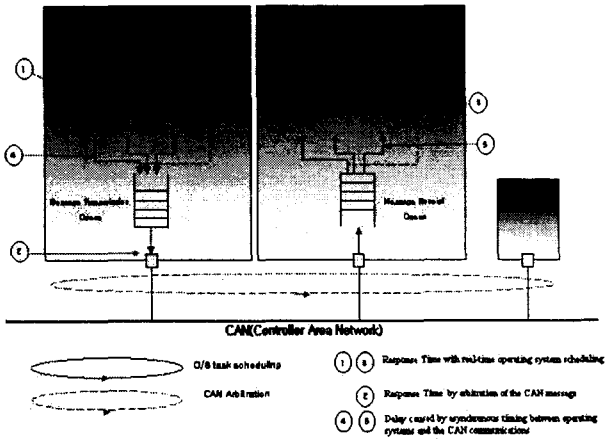

그림 1. 시스템 구성도
Fig. 1. Configuration of system

2.2 운영체계 지연시간

운영체계의 태스크 수행은 각 태스크들의 우선순위에 의해 결정된다. DMS에서의 우선순위 결정 방식은 최소의 데드라인을 갖는 태스크가 최고의 우선순위를 갖는 방식이다. 운영체계에서의 지연시간은 낮은 우선순위의 태스크 수행으로 인한 블록킹 시간(blocking time)과 높은 우선순위 태스크의 선점에 의한 대기시간, 그리고 연산시간으로 구성된다. 따라서 운영체계 최악 지연시간(r_i)은 식(1)과 같다[5].

$$r_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{r_i}{T_j} \right\rceil C_j \tag{1}$$

$$B_i = \underset{\forall K \in lp(i)}{MAX}\, r_k \tag{2}$$

여기서, C_i와 C_j는 최악의 경우 태스크 i, j의 실행 시간을, B_i는 블록킹 시간을 나타내며, T_j는 태스크 j의 주기를, r_i는 최악의 경우 태스크 i의 지연시간을 의미한다. 그리고 $hp(i)$는 태스크 i보다 높은 우선순위를 갖는 태스크의 집합을 의미하며, $lp(i)$는 태스크 i보다 낮은 우선순위를 갖는 태스크의 집합을 의미한다. 식 (2)에서 보는 바와 같이 블로킹 시간은 자원 공유 등에서 나타나는 낮은 우선순위 태스크에 의한 최악의 지연시간을 의미한다.

2.3 네트워크 지연시간

CAN에서 DMS를 이용하여 메시지를 전송할 때 발생되

는 네트워크 지연시간은 대기행렬 지연시간, 그리고 전송 지연시간으로 구분된다[4].

최악의 네트워크 지연시간(r_m)은 식(3)과 같이 표현된다.

$$r_m = B + \sum_{\forall j \in hp(m)} \left\lceil \frac{r_m + J_j}{T_j} \right\rceil C_m \tag{3}$$

$$B = \underset{\forall K \in lp(m)}{MAX} c_k \tag{4}$$

여기서, C_m은 메시지 m의 실질적인 전송 시간을 의미하며, B 는 메시지 m의 블록킹 시간, T_j는 메시지 j의 발송 주기, $hp(m)$는 메시지 m보다 우선순위가 높은 메시지의 집합, 그리고 $lp(m)$는 메시지 m보다 우선순위가 낮은 메시지의 집합을 의미한다. 식 (4)에서 보는 바와 같이 블로킹 시간은 낮은 우선순위 메시지에 의한 최악의 지연시간을 의미한다.

식 (3)에서 전송 지연시간(C_m)은 메시지 전송 큐의 메시지가 네트워크를 점유하고 목적 노드의 메시지 수신 큐까지 도달하는데 걸리는 시간으로 정의하며, 이는 식(5)와 같이 표현된다[7].

$$C_m = \left(\left\lfloor \frac{34 + 8S_m}{5} \right\rfloor + 47 + 8S_m \right) \tau_{bit} + \rho \tag{5}$$

여기서, C_m은 버스 상에서 메시지가 물리적으로 전송되는 시간을 나타내며, S_m은 메시지의 바이트 크기를 의미한다. 그리고 τ_{bit} 는 네트워크에서 1비트를 보내는 시간이며, ρ 는 네트워크의 전기적 특성을 고려한 지연시간이며 네트워크 특성에 따라 정수로 표현된다.

2.4 운영체계 수행과 네트워크 전송의 비동기성에 의한 대기 지연시간

그림 1에서 송신 노드와 수신 노드 상에 존재하는 종단의 송신 태스크와 수신 태스크는 각각의 지역 운영체계에서 각각의 주기를 가지고 운영이 된다. 또한 송신 노드에서 수신 노드로 전송되는 메시지는 전체 분산 네트워크 시스템의 메시지 구성에 따른 자체 주기를 가지고 전송이 수행된다. 각 수행부에서 발생하는 이러한 주기의 비동기성은 각각의 최악 응답시간 특성에 추가하여 대기 지연시간을 발생시키게 된다[8].

그림 2에서 보는 바와 같이 수행의 순서에 있어 선행 수행부의 지연시간이 후행 수행부의 수행 시작 시간과 차이가 나기 때문에 대기 지연시간이 발생한다. 그림 2에서 메시지의 운영체계 계층으로부터 네트워크 계층으로 전송 시에는 송신 노드의 운영체계가 선행 수행부, 네트워크가 후행 수행부가 되며, 메시지의 네트워크 계층으로부터 운영체계 계층으로 수신 시에는 네트워크가 선행 수행부, 수신 노드의 운영체계가 후행 수행부가 된다. 따라서 대기 지연시간은 송신 노드에서 발생한 메시지가 네트워크로 전송 대기될 때 발생하며, 네트워크에서 수신된 메시지가 수신 노드에서 처리 대기될 때 발생한다.

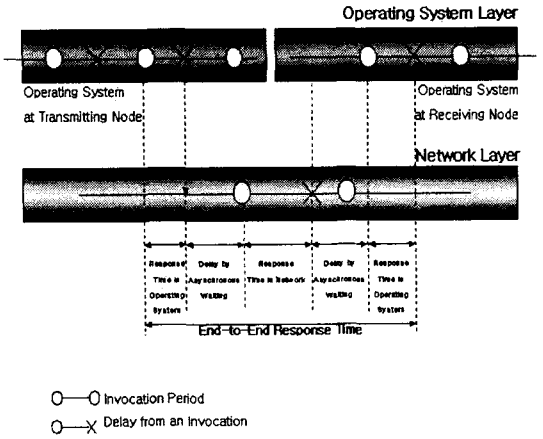

그림 2. 종단 간 지연시간의 구성
Fig. 2. Composition of end-to-end delay

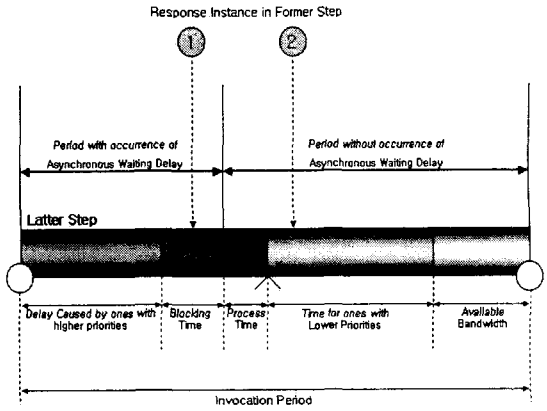

그림 3. 비동기성에 의한 지연시간의 발생 구역
Fig. 3. Delay boundary caused by asynchronous timing

운영체계 수행과 네트워크 전송의 비동기성에 의한 대기 지연시간은 그림 3에서 보는 바와 같이 선행 수행부의 지연시간이 후행 수행부의 어느 시점까지 나타나는지에 따라 발생하게 된다. 선행 수행부의 지연시간이 후행 수행부의 수행 주기에서 ②의 경우와 같이 수행 시작 시간보다 길게 나타날 경우 비동기성에 의한 대기 지연시간이 발생한다. 만약 선행 수행부의 지연시간이 ①의 경우와 같이 후행 수행부의 수행 시작 시간보다 짧게 나타난다면 비동기성에 의한 대기 지연시간은 발생하지 않는다.

비동기성에 의한 최악의 대기 지연시간은 선행 수행부의 지연시간이 후행 수행부의 수행 시작 시간보다 길게 나타나고, 후행 수행부의 높은 우선순위 작업 수행에 의한 지연시간과 블로킹 시간으로 구성되는 대기시간이 최소가 될 때이다. 후행 수행부의 최소 지연시간은 식(6)과 같이 표현되며, 최소 지연시간을 가질 때의 최악의 대기 지연시간은 식(7)과 같이 표현된다[11].

$$r_{i\_min} = \sum_{\forall j \in (hp(i) \cup harm(i))} C_j \quad (6)$$

$$A_i = T_i - r_{i\_min} \quad (7)$$

여기서, *harm(i)*는 태스크 i, 혹은 메시지 i의 주기의 멱(power)을 주기로 갖는 태스크 혹은 메시지의 집합이다. 즉 태스크 i, 혹은 메시지 i와 동시에 수행 대기되는 최소의 태스크 혹은 메시지 집합을 의미한다. 따라서 식(6)과 (7)을 고려한 전체 분산 제어 시스템의 종단 간 최악 지연시간은 식(8)과 같이 표현된다[11].

$$E_i = r_i(trans) + A_i(net) + r_m + c_m + A_i(recv) + r_i(recv) \quad (8)$$

여기서, E_i는 종단 간 최악의 지연시간을 의미하고, $r_i(trans)$는 송신부 운영체계 계층에서의 최악 지연시간을, $A_i(trans)$는 네트워크 계층에서의 비동기성으로 인한 최악의 대기 지연시간을 의미한다. r_m은 네트워크 최악의 대기 행렬 지연시간을 의미하고, C_m은 네트워크 전송 지연시간을 의미한다. 또한$A_i(recv)$는 수신부 운영체계 계층에서의 비동기성으로 인한 최악의 대기 지연시간을 의미하며, $r_i(recv)$는 수신부 운영체계 계층에서의 최악 지연시간을 의미한다.

2.5 종단 간 지연시간 최소화를 위한 협조 스케줄링 알고리즘

식 (8)에서 표현된 바와 같이 전체 분산 시스템의 종단 간 최악 지연시간은 각 수행부에서의 최악의 지연시간과 각 수행부의 비동기적인 수행으로 인한 대기 지연시간으로 구성된다. 이러한 구성 요소 중 대기 지연시간은 동기화를 통해 최소화가 가능하다. 본 논문에서는 이러한 대기 지연시간의 최소화를 통해 전체 분산 시스템의 종단 간 지연시간을 최소화하는 협조 스케줄링 알고리즘을 제안한다.

협조 스케줄링 알고리즘을 위해 태스크 및 메시지 집합의 시간 특성 요소를 다음과 같이 재정의 한다.

① 주기(T_i)

태스크 및 메시지의 실시간성 보장 조건으로서 주기 내의 수행 완료 보장을 실시간성이 만족되는 것으로 규정한다.

② 시작시간($R_i(k)$)

주기성을 갖는 태스크 및 메시지가 휴지 상태에서 수행대기 된 상태를 의미하며 다음의 식 (9)와 같이 정의된다.

$$R_i(k) = (R_0 + k \times T_i) \quad (9)$$

여기서, R_0는 최초의 시작 시간을 나타내고, k는 반복 수행 횟수(0,1,2,..)를 의미한다.

③ 수행시간(C_i 또는C_m)

태스크 및 메시지의 수행시간을 규정한다.

④ 데드라인($D_i(k)$)

태스크 및 메시지의 데드라인으로서 DMS의 일반적인 개념인 하드(hard) 데드라인이 아닌 소프트(soft) 데드라인이다. 태스크 및 메시지의 데드라인 할당은 스케줄러에 의해 온라인상에서 동적으로 이루어진다. 데드라인의 가변 범위는 다음의 식 (10)과 같다.

$$0 < D_i(k) \le T_i \quad (10)$$

⑤ 수행가치(V)

태스크 및 메시지의 수행 가치를 규정하며, 수행 가치의 상대적인 비교를 위해 수행 가치가 높은 순으로 높은 값을 부여한다.

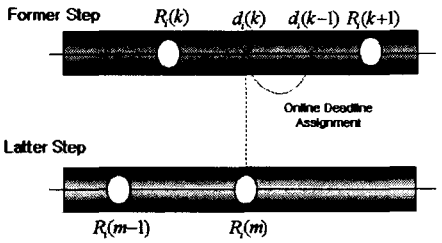

그림 4. 협조 스케줄링 알고리즘의 데드라인 온라인 할당
Fig. 4. Online deadline assignment of coordinated scheduling algorithm

본 논문에서 제안하는 협조 스케줄링 알고리즘은 선행 수행부 태스크 혹은 메시지의 시작 시간 $R_i(k)$에서 데드라인 $D_i(k)$의 동적인 할당을 통해 이루어지며, 데드라인의 할당 값은 그림 4와 같이 후행 수행부에서 $R_i(m-1) < R_i(k) < R_i(m)$을 만족하는 $R_i(m) - R_i(k)$의 값으로 결정된다. 데드라인 할당을 위한 우선순위는 선행 수행부의 태스크 혹은 메시지의 수행가치 v_i에 의해 결정된다. 수행가치에 의한 우선순위 할당에 관한 연구는 기존에 QoS(Quality of Service) 분야에서 활발히 이루어진 바 있다[10].

본 논문에서 제안한 협조 스케줄링 알고리즘의 수행절차는 수행가치 v_i에 따른 우선순위에 의해 탐욕(greed) 방식으로 데드라인 할당을 수행한다. 제안된 수행절차는 송신 노드의 스케줄러와 메시지 스케줄러에서 수행된다. 송신 노드에서 수행될 경우에 선행 수행부는 송신부의 운영체계 스케줄러이며, 후행 수행부는 메시지 스케줄러이다. 그리고 메시지 스케줄러에서 수행될 경우에 선행 수행부는 메시지 스케줄러이며, 후행 수행부는 수신부의 운영체계 스케줄러이다. 수행 알고리즘은 다음과 같다.

Step 1:
태스크 혹은 메시지 집합을 수행가치가 높은 차례부터 *i=1,2...n*의 순서로 내림차순 정렬한다.

Step 2:
*i=1*부터 *i=n*의 순서로 다음을 수행 한다.

Step 2-1:
$R_i(m-1) < R_i(k) < R_i(m)$을 만족하도록 선행 수행부의 k 번째에 후행 수행부의 m을 구한다.

Step 2-2:
$D_i(k) = R_i(m) - R_i(k)$을 새로운 데드라인으로 하여 허용제어(admission control)를 수행한다.

Step 2-3:
허용 가능하면 Step2를 반복하고, 허용 불가능하면 Step 3을 진행한다.

Step 3:
*j=n*부터 *j=i-1*의 순서로 다음을 진행한다. j=i-1 이후 Step 3이 반복되면 $D_i(k) = D_i(k-1)$로 하고 Step2를 진행한다.

Step 3-1:
$D_j(k) = T_j$로 하여 허용제어를 수행한다.

Step 3-2:
허용 가능하면 Step2를 반복하고, 허용 불가능하면 Step 3을 진행한다.

상기 절차 중의 허용 제어는, 태스크의 경우는 *i=1,2...n*의 모든 태스크 집합에 대해 식 (1)의 r_i가 $r_i < T_i$를 만족하는 경우 허용 가능한 것으로 판단하며, 메시지의 경우는 *i=1,2...m*의 모든 메시지 집합에 대해 식 (3)의 r_m이 $r_m < T_i$를 만족하는 경우 허용 가능한 것으로 판단하여 전체 시스템의 실시간성을 보장한다.

3. 모의실험 및 결과

본 논문에서 제안한 식 (8)의 타당성을 입증하기 위해 그림 5의 환경에서 표 1의 시간 특성 조건에 따른 종단 간 응답시간의 모의실험을 수행한다. 표 1의 시간 특성은 다중 태스크 처리를 지원하는 대표적인 운영체계인 표준 리눅스(Linux)의 소프트웨어 타이머 인터럽트 주기가 10*mS*임을 감안하여 주기, 데드라인, 그리고 연산시간을 10*mS*의 배수로 설정하였다. 표 1에서 송신노드 태스크는 송신노드의 운영체계에서 수행되는 태스크를 의미하며, 네트워크 메시지는 네트워크에서 전송되는 메시지를 의미한다. 그리고 수신노드 태스크는 수신노드 운영체계에서 수행되는 태스크를 의미한다.

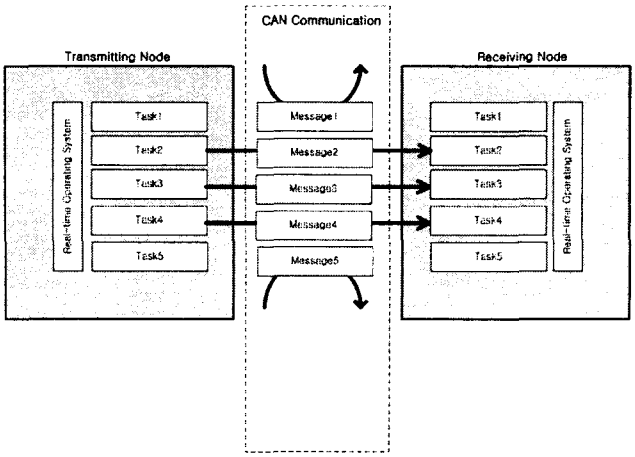

그림 5. 모의실험 환경
Fig. 5. Simulation Environment

본 논문에서는 송신노드 태스크에서 네트워크 메시지를 통해 수신노드 태스크로 전달되는 종단 간 경로를 실시간 분산 제어채널로 정의한다. 예를 들어 아래의 표 1에서 고유번호 2번의 송신노드 태스크에서 고유 번호 2번의 메시지를 통해 고유번호 2번의 수신 태스크로 전달되는 종단 간 경로는 분산 제어채널 2번이 된다.

표 1. 종단 간 지연시간 모의실험을 위한 태스크 및 메지지 집합의 정의
Table 1. Task and message sets for the simulation of end-to-end delay

고유 번호		1	2	3	4	5
송신노드 태스크	주기	300mS	500mS	500mS	700mS	700mS
	데드라인	300mS	500mS	500mS	700mS	700mS
	연산시간	20ms	20ms	50ms	50ms	100ms
네트워크 메시지	주기	600mS	600mS	600mS	600mS	600mS
	데드라인	600mS	600mS	600mS	600mS	600mS
	전송바이트	8Bytes	8Bytes	8Bytes	8Bytes	8Bytes
수신노드 태스크	주기	500mS	700mS	700mS	1000mS	1000mS
	데드라인	500mS	700mS	700mS	1000mS	1000mS
	연산시간	20ms	20ms	50ms	50ms	100ms

그림 5에서 보는 바와 같이 표 1의 고유번호 1번과 5번의 송신노드 태스크와 수신노드 태스크는 분산 제어 시스템의 메시지 통신과 무관한 운영체계의 독립 태스크이며, 고유번호 1번과 5번의 네트워크 메시지 또한 제어 시스템 정보와 무관한 독립적인 메시지이다. 따라서 표 1의 분산 제어 시스템 상에서는 고유번호 2번에서 4번까지의 송신노드 태스크가 각각 고유번호 2번에서 4번까지의 네트워크 메시지를 이용하여 고유번호 2번에서 4번까지의 수신노드 태스크로 제어 정보를 전달하는 실시간 분산 제어채널이 2번부터 4번까지 존재한다.

본 논문의 모의실험에서 CAN의 메시지 전송 속도는 125*kbps*로 가정하며, 네트워크 메시지의 지터는 0으로 가정한다. 또한 비동기성에 의한 대기 지연시간의 명확한 도시를 위해 낮은 우선순위 태스크 혹은 메시지에 의한 블로킹 시간은 항상 최대가 되는 것으로 가정한다.

그림 6은 표 1의 조건에서 실시간 분산 제어채널 4번의 종단 간 지연시간의 발생을 256회 동안 모의실험한 결과이다. 그림 6의 최악의 종단 간 지연시간은 실시간 분산 제어채널 4번에 대해 식(8)을 이용하여 구한 값으로서 2084mS의 값을 갖는다. 기존의 종단 간 최악 지연시간 분석 연구 결과[4]의 경우, 운영체계 수행과 메시지 전송의 비동기성에 의한 대기 지연시간은 고려가 되어있지 않기 때문에 최악의 종단 간 지연시간은 484mS의 값을 갖게 되고, 모의실험 된 종단 간 지연시간의 결과 중 최대 값이 2070mS이기 때문에 시스템의 시간 특성을 반영할 수 없음을 알 수 있다. 그림 6에서 보는 바와 같이 실시간 분산 제어 채널 4번의 종단 간 지연 시간은 25회를 기준으로 반복적으로 나타나는 추세를 갖는다. 따라서 모의실험에서 수행한 횟수 이상에서의 종단 간 지연시간 또한 그림 6에서와 같이 본 논문에서 제안한 최악의 종단 간 지연시간을 넘지 않음을 예측할 수 있다.

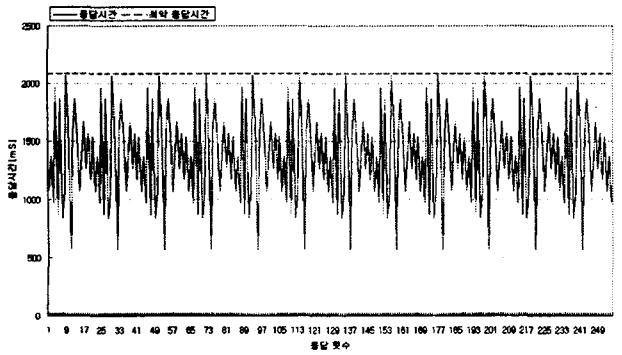

그림 6. 표1의 실시간 분산 제어채널 4의 종단 간 지연시간
Fig. 6. End-to-end delay of real-time distributed control channel 4 in Table 1.

기존의 종단 간 최악 지연시간 분석 연구 결과에 의하면, 실제로 표1의 실시간 분산 제어채널 4번의 결과와 유사한 최악 지연시간 값을 갖는 태스크 및 메시지 집합의 예는 표 2와 같다. 기존의 연구 결과는 수신노드의 태스크에 대한 고려가 이루어져있지 않기 때문에 표2의 수신노드 태스크를 위해 표1과 동일한 수신노드 태스크의 시간 특성 집합을 사용한다.

그림 7은 그림 5의 동일한 환경에서 표 2의 조건에서 실시간 분산 제어채널 4번의 종단 간 지연시간의 발생을 256회 동안 모의실험한 결과이다. 그림 7에서와 같이 표 2의 실시간 분산 제어채널 4번의 종단 간 지연시간은 최대 값이 2140mS로 예측된 최악의 지연시간 이상의 값을 갖게 된다. 따라서 기존의 종단 간 최악 지연시간 분석 연구 결과에 의해 예측된 최악의 지연시간을 기반으로 하여 분산 제어 시스템을 설계할 경우, 실제로 온라인상에서는 종단 간 실시간성을 위배할 수 있음을 알 수 있다. 이것은 기존의 연구 결과에 운영체계의 수행과 네트워크의 전송 시기의 비동기성으로 인한 대기 지연시간이 반영되어 있지 않기 때문이다.

표 2. 기존의 연구 결과에 따른 표1의 시간 특성 변경 예
Table 2. An example of modified time characteristics from Table 1 according to previous study

송신노드 태스크	주기	100mS	100mS	100mS	1500mS	500mS
	데드라인	100mS	100mS	100mS	1500mS	500mS
	연산시간	20ms	20ms	50ms	50ms	100ms
네트워크 메시지	주기	600mS	600mS	600mS	600mS	600mS
	데드라인	600mS	600mS	600mS	600mS	600mS
	전송바이트	8Bytes	8Bytes	8Bytes	8Bytes	8Bytes
수신노드 태스크	주기	500mS	700mS	700mS	1000mS	1000mS
	데드라인	500mS	700mS	700mS	1000mS	1000mS
	연산시간	20ms	20ms	50ms	50ms	100ms

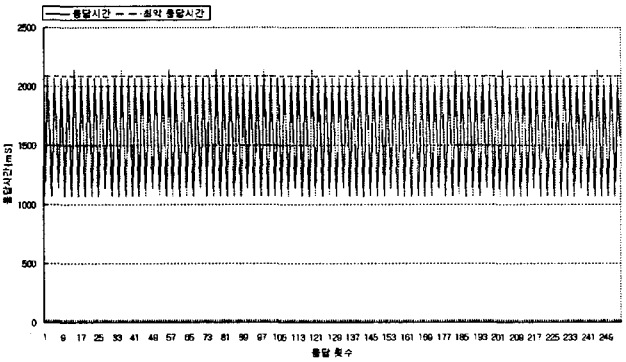

그림 7. 표2의 실시간 분산 제어채널 4의 종단 간 지연시간
Fig. 7. End-to-end delay of real-time distributed control channel 4 in Table 2.

본 논문에서 제안한 협조 스케줄링 알고리즘을 사용한 경우, 종단 간 지연시간 감소를 통해 분산 제어시스템의 성능을 향상시킬 수 있음을 입증하기 위해 표 1의 조건에서 협조 스케줄링 알고리즘을 사용한 경우와 그렇지 않은 경우에 대해 종단 간 지연시간의 모의실험을 수행한다. 표 1의 데드라인은 협조 스케줄링의 온라인 데드라인 할당 기법에서 초기값으로서 사용된다.

온라인 데드라인 할당 기법의 기준이 되는 각 실시간 분산 제어채널의 수행가치는 실제로는 주위 환경 및 시스템 동작 상태에 따라 동적으로 변화가 이루어 질 수 있지만, 본 논문에서는 모의실험 결과의 명확한 도시를 위해 실시간 분산 제어채널 번호와 동일한 고정된 수행가치 값을 각 분산 제어채널에 할당한다.

그림 8은 표 1에 대해 협조 스케줄링 알고리즘을 사용한 경우와 사용하지 않은 경우, 시스템의 채널 4번의 종단 간 지연시간을 모의실험을 통해 비교한 결과이다. 그림에서 DMS 스케줄링이 협조 스케줄링 알고리즘을 사용하지 않은 경우이다. 그림에서 보듯이 협조 스케줄링 알고리즘을 사용하지 않은 경우의 시스템 평균 응답시간은 1372mS이고, 반면에 협조 스케줄링 알고리즘을 사용한 경우의 시스템 평균 응답시간은 899mS이다. 그림 8의 결과와 같이 협조 스케줄링 알고리즘을 사용한 경우, 그렇지 않은 경우와 비교하여 평균 지연시간이 약 35% 감소함을 알 수 있다.

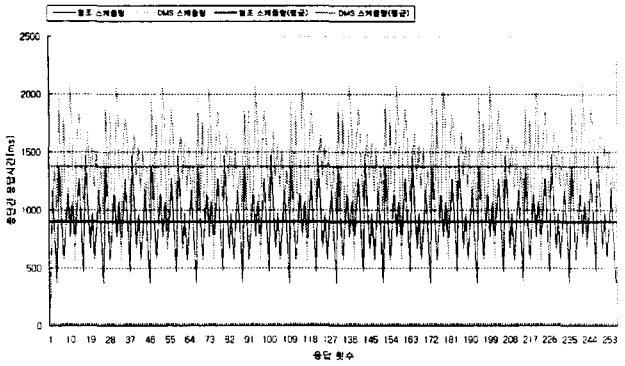

그림 8. 실시간 분산 제어채널 4번의 종단 간 지연시간 비교
Fig. 8. Comparison with end-to-end delay of real-time distributed control channel 4

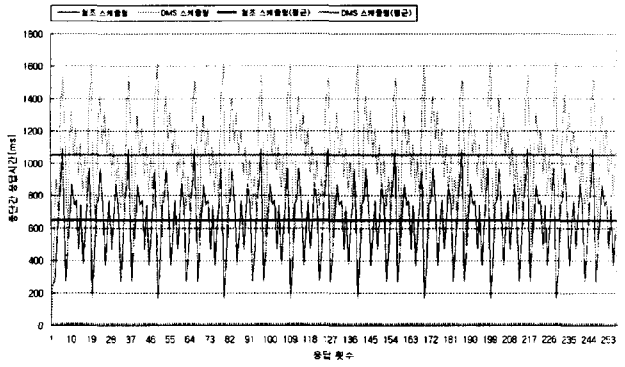

그림 9. 실시간 분산 제어채널 2번의 종단 간 지연시간 비교
Fig. 9. Comparison with end-to-end delay of real-time distributed control channel 2

그림 9는 표 1의 분산 제어 시스템의 채널 2번의 종단 간 지연시간을 모의실험을 통해 비교한 결과이다. 그림에서 DMS 스케줄링이 협조 스케줄링 알고리즘을 사용하지 않은 경우이다. 그림 9에서 보듯이 협조 스케줄링 알고리즘을 사용하지 않은 경우 시스템 평균 지연시간은 1051mS이고, 협조 스케줄링 알고리즘을 사용한 경우 시스템 평균 지연시간은 643mS이다. 그림 9의 결과에서 채널 2번의 경우도 평균 지연시간이 약 39% 감소함을 알 수 있다. 그러므로 전체 채널의 양 끝 채널을 모의실험 함으로써 협조 스케줄링 알고리즘을 사용할 경우 모든 채널의 평균 지연시간이 감소한다는 것을 알 수 있다.

이렇듯 제안된 협조 스케줄링 알고리즘을 사용할 경우, 사용하지 않는 경우와 비교하여 시스템의 성능이 향상됨을 입증하였다.

4. 결 론

본 논문에서는 CAN을 기반으로 한 분산 제어 시스템 설계를 위해 구현 특성을 고려한 종단 간 지연시간 모델링을 제안하였다. 본 논문에서는 대부분의 실시간 시스템이 다중 태스크 환경의 운영체계를 사용하는 점을 감안하여 네트워크 지연시간의 분석과 함께 운영체계의 지연시간 분석, 그리고 네트워크 시간 특성과 운영체계 시간 특성의 비동기성에 의한 대기 지연시간을 고려하여 실질적인 종단 간 최악의 지연시간 분석을 수행하였다. 또한 모의실험을 통해 분석된 결과가 타당한 결과임을 입증하였으며, 기존의 최악 지연시간 분석 방식에 의해 설계된 분산 제어 시스템이 다중 태스크 운영체계 환경에서 종단 간 지연시간 제약을 만족할 수 없음을 입증하였다.

또한 본 논문에서는 종단 간 지연시간을 최소화하는 협조 스케줄링 알고리즘을 제안하였다. 제안된 알고리즘은 DMS를 기반으로 하며, 운영체계 태스크와 네트워크 메시지의 데드라인을 온라인상에서 가변하여 자원 이용의 비동기성으로 인한 대기 지연시간을 최소화하는 방식을 통해 전체 종단 간 지연시간을 최소화하는 알고리즘이다. 제안된 알고리즘을 CAN을 기반으로 한 실시간 시스템에 적용한 모의실험을 통해 알고리즘을 사용하지 않은 경우에 비해 종단 간 지연시간을 단축시킴으로써 분산 제어시스템의 성능을 향상시킴을 입증하였다.

향후 연구과제로는 대표적인 실시간 시스템인 이동 로봇을 대상으로 한 지연시간 분석과 실험을 수행할 예정이며, 지연시간을 최소화하기 위한 협조 스케줄링 알고리즘의 구현과 온라인 클럭 동기화 기법을 포함한 지연 보상 제어기의 개발을 수행할 예정이다.

참 고 문 헌

[1] M. Mock and E. Nett, "Integrating Hard and Soft Real-Time Communication in Autonomous Robot Systems", IECE/IEEE Joint Special Issue on Autonomous Decentralized Systems of the IEICE Transactions on Communications, vol. E83-B, no.5, pp. 1067-1074, May 2000.

[2] Johan Nilsson, "Real-Time Control Systems with Delays" Phd Thesis, Lund Institute of Technology, 1998.

[3] M. Farsi, K. Ratcliff and Manual Barbosa, "An overview of Controller Area Network", Computing and Control Engineering Journal, pp. 113-120, June 1999.

[4] 전종만, 김대원, "메시지 지연시간을 고려한 CAN 기반 피드백 제어시스템의 응답특성 분석", 대한전기학회논문집, Vol. 51D, No5, pp. 190-196, 2002.

[5] N.C Audesly, "Deadline Monotonic Scheduling", YCS146, Dept. Computer Science, University of York, 1990. [online] available at http//cs.york.ac.uk/rts/.

[6] A. Mehra, A. Indiresan, and K. G. Shin, "Resource management for real-time communication: Making theory meet practice," in Proc. of IEEE RTAS, pp. 130-138, 1996.

[7] Tindell, K., Burns, A., "Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks", Technical report YCS 229, Department of Computer Science, University of York, England, May 1994.

[8] Bjorn Wittenmark, Ben Bastian, and Johan Nilsson, "Analysis of Time Delays in Synchronous and Asynchronous Control Loops", Decision and control, Proceedings of the 41st IEEE Conference on, Vol. 4, pp. 4637-4642, 1998.
[9] G. F. Franklin, J. D. Powell, and A. Emani-Naeini., "Feedback Control of Dynamic Systems", Addison-Wesley, Reading, Massachusetts, third edition, 1994.
[10] G.Buttazzo, M.Spuri, F.Sensini, "Value Vs. Deadline Scheduling in Overload Conditions", IEEE Proc. on Real Time Systems Symposium, IEEE Computer Society Press., 1998.
[11] 이희배, 김홍열, 김대원, "CAN 기반 분산 제어시스템의 종단 간 지연시간 분석과 온라인 글로벌 클럭 동기화 알고리즘 개발", 정보 및 제어 학술회의 논문집, pp. 677-680, 2003.

저 자 소 개

이 희 배 (李 喜 培)

1972년 2월 1일생. 2002년 명지대학교 전기정보제어공학부 졸업. 2004년 동대학원 졸업. 2004년~현재 (주) 시넵스정보통신 재직중. 관심분야는 실시간 분산시스템, 이동통신 중계기 등.

Tel : (02) 2634-0038
Fax : (031) 330-6226
E-mail : aspilin1030@hanmail.net

김 홍 열 (金 弘 烈)

1973년 11월 25일생. 1996년 명지대학교 전기공학과 졸업. 1998년 동대학원 졸업. 2004년 박사과정 수료. 1997년~현재 (주)캐리어 기술연구소 주임연구원. 관심분야는 실시간 분산시스템, 행위기반 로봇 등.

Tel : (031) 330-6755
Fax : (031) 330-6226
E-mail : hr.kim@carrier.co.kr

김 대 원 (金 大 元)

1960년 2월 15일생. 1984년 서울대학교 제어계측공학과 졸업. 1986년 동대학원 졸업. 1987년~1992년 (주)대우중공업 중앙연구소 선임연구원. 1992년~현재 명지대학교 정보제어공학과 교수. 관심분야는 실시간 분산시스템, 퍼스널로봇, 웹기반 응용 등.

Tel : (031) 330-6755
Fax : (031) 330-6226
E-mail : dwkim@mju.ac.kr