

CAN 기반 분산 제어 시스템의 종단 간 지연 시간 분석과 온라인 글로벌 클럭 동기화 알고리즘 개발

End-to-end Delay Analysis and On-line Global Clock Synchronization Algorithm
for CAN-based Distributed Control Systems

이희배, 김홍렬, 김대원

Heebae Lee, Hongryeol Kim, Daewon Kim

명지대학교 정보제어공학과(전화:(031)330-6755, 팩스:(031)330-6226, E-mail: jin1030@miu.ac.kr)

명지대학교 정보제어공학과(전화:(031)330-6755, 팩스:(031)330-6226, E-mail: museros@miu.ac.kr)

명지대학교 정보공학과(전화:(031)330-6755, 팩스:(031)330-6226, E-mail: dwkim@miu.ac.kr)

Abstract : In this paper, the analysis of practical end-to-end delay in worst case is performed for distributed control system considering the implementation of the system. The control system delay is composed of the delay caused by multi-task scheduling of operating system, the delay caused by network communication, and the delay caused by the asynchronous between them. Through simulation tests based on CAN(Controller Area Network), the proposed end-to-end delay in worst case is validated. Additionally, online clock synchronization algorithm is proposed here for the control system. Through another simulation test, the online algorithm is proved to have better performance than offline one in the view of network bandwidth utilization.

I. 서 론

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 지연 시간과 결합하여 전체 시스템을 실패하게 만드는 요소가 되기도 한다[1][2].

본 논문에서는 CAN(Controller Area Network)을 기반으로 한 분산 제어 시스템 설계를 위해 구현 환경을 고려한 종단 간 지연시간 분석과 온라인 글로벌 클럭(global clock) 동기화 알고리즘을 제안한다. 분산화된 제어 시스템을 구성하는데 있어서 네트워크 프로토콜은 실시간 제어를 가능하게 하는 중요한 요소로서 작용한다. CAN은 고속의 통신 인터페이스를 제공하고 데이터 프레임의 오버헤드(overhead)가 적기 때문에 빠른 응답 특성을 갖고 있다. 또한 식별자(identifier)를 이용한 충돌 방지와 전송 중재(arbitration) 기능을 갖고 있어 실시간 제어 네트워크 프로토콜로서 피드백 제어를 요구하는 시스템 환경에 적합하다[3].

CAN의 경우 기존의 네트워크 지연시간 분석에 관한 연구가 수행된 바 있으나[4], 최근 실시간 시스템의 일반적인 구현 방법인 다중 태스크 운영 체계 환경에 대한 고려가 이루어져 있지 않기 때문에 분석 결과가 구현 환경을 반영하지 못한다는 단점을 갖는다. 따라서 본 논문에서는 대부분의 실시간 시스템이 다중 태스크 환경의 운영 체계를 사용하는 점을 감안하여 네트워크 지연시간의 분석과 함께 운영 체계의 지연시간 분석, 그리고 네트워크 시간 특성과 운영 체계 시간 특성의 비동기성에 의한 지연시간을 고려하여 실질적인 종단 간 최악의 지연시간 분석을 수행한다. 또한 모의 분석을 통해 분석된 결과가 타당한 결과임을 입증한다.

네트워크를 이용한 분산 제어 시스템이 갖는 필연적인 지연시간 특성으로 인해 이러한 지연시간을 보상할 수 있는 기능을 갖춘 제어 시스템의 도입은 필수적이다. 이러한 보상 제어 시스템에 관한 연구로서 고속 전자 제어 시스템을 대상으로 하여 온라인상에서 지연시

간을 측정하는 제어 기시스템에 관한 연구가 수행된 바 있으나[2], 이 연구는 지역 클럭 간의 동기화를 유지하기 위해 과다한 통신 오버헤드가 존재한다는 문제점을 갖는다.

따라서 본 논문에서는 통신 오버헤드를 최소화하기 위해 온라인 글로벌 클럭 동기화 알고리즘을 제안한다. 제안된 글로벌 클럭 동기화는 온라인상에서의 네트워크 트래픽 분석과 각 네트워크 노드에서 실시간 운영 체계의 로컬 클럭 드리프트를 감안하여 이루어진다. 또한 CAN을 기반으로 한 모의 실험을 통하여 기존의 오프라인 글로벌 클럭 동기화 알고리즘과 비교하여 충분한 클럭 정확도를 유지하면서 네트워크 트래픽을 최소화할 수 있음을 입증한다.

2. 본 론

2.1 종단 간 지연시간의 구성 요소

본 논문의 배경이 되는 전체 시스템의 구조는 그림 1과 같다. 그림 1에서 ①과 ③은 각각 운영 체계의 태스크 스케줄링에 의한 지연시간을 나타내고 ②는 CAN의 메시지 스케줄링에 의한 지연시간을 나타낸다.

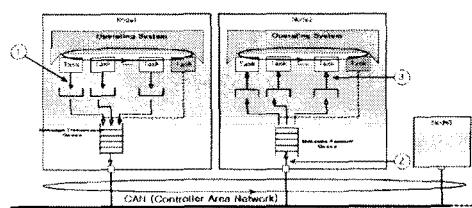


그림 1. 시스템 구성도

본 논문에서는 네트워크 기반 실시간 시스템의 실질적인 종단 간 지연시간 요소를 운영 체계의 태스크 스케줄링에 의해 발생하는 지연시간인 운영 체계 지연시간, CAN의 메시지 스케줄링에 의해 발생하는 지연시

간인 네트워크 지연시간, 그리고 이 두 가지 지연 시간 특성의 비동기성에 의해 발생하는 대기 지연시간이 세 가지로 정의하고 분석을 수행한다.

본 논문에서는 네트워크를 통해 전송되는 메시지와 운영체계에서 수행되는 태스크의 스케줄링은 대표적인 정적 스케줄링 방식인 RMS(Rate Monotonic Scheduling)[5]를 사용한다. 송신 노드에서 운영체계를 주기적인 스케줄링 방식인 RMS를 사용한다 할지라도 RMS의 응답 시간은 비주기적인 특성을 가지고 있기 때문에 본 논문에서는 그럼 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.

2.2 운영체계 지연시간

운영체계의 태스크 수행은 각 태스크들의 우선순위에 의해 수행된다. RMS에서의 우선순위는 최소의 주기를 갖는 태스크가 최고의 우선순위를 갖게 된다. 운영체계에서의 지연시간은 낮은 우선순위의 태스크 수행으로 인한 블록킹 시간(blocking time)과 높은 우선순위 태스크의 선점율 포함하는 대기 시간, 그리고 연산 시간으로 구성된다. 따라서 운영체계 지연시간의 최악응답시간(r_i)은 식(1)과 같다[5].

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

$$B = \underset{\forall K \in lp(i)}{\text{MAX}} (r_i) \quad (2)$$

여기서, C_i 와 C_j 는 최악의 경우 태스크 i, j 의 실행 시간을, B_i 는 블록킹 타임을 나타내며, T_j 는 태스크 j 의 주기를, r_j 는 최악의 경우 태스크 i 의 지연시간을, 그리고 $hp(i)$ 는 태스크 i 보다 높은 우선순위를 갖는 태스크의 집합을 나타낸다. 식 (2)에서 보는 바와 같이 블록킹 타임은 자원 공유 등에서 발생하는 낮은 우선순위 태스크에 의한 최악의 지연시간을 의미한다.

2.3 네트워크 지연시간

CAN에서 RMS를 이용하여 메시지를 전송할 때 발생되는 네트워크 지연시간은 대기행렬 지연시간, 그리고 전송 지연시간으로 구분한다[4].

첫째, 메모리 큐에 도달한 메시지가 네트워크에 대한 접근권한을 획득하기까지 걸리는 시간을 대기행렬 지연시간이라고 정의하며 최악의 대기행렬 지연시간(r_m)은 식(3)과 같이 표현된다.

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

$$B = \underset{\forall K \in lp(m)}{\text{MAX}} (C_k) \quad (4)$$

여기서, 첫째 항인 B 은 블록킹 타임을 나타내며 식 (4)와 같이 표현된다. 이 역시 낮은 우선순위의 메시지가 네트워크를 점유해서 높은 우선순위의 메시지가 대기하는 시간을 말한다.

식(3)의 둘째 항은 실제 메모리 큐의 지연시간을 나타내며, J_j 는 프로세스 j 의 jitter(jitter)이고, τ_{bu} 는 네트워크에서 1비트를 보내는 시간이다.

둘째, 메모리 큐의 메시지가 네트워크를 점유하고 목적 노드의 메모리 큐까지 도달하는데 걸리는 시간을 전송 지연시간이라고 정의하며 이는 식(5)와 같이 표현된다[8].

$$C_m = \left(\left\lfloor \frac{34 + 8s_m}{5} \right\rfloor + 47 + 8s_m \right) \tau_{bu} + \rho \quad (5)$$

여기서, C_m 은 버스 상에서 메시지가 물리적으로 전송되는 시간을 나타내며, s_m 은 메시지의 바이트 크기를 나타낸다. 그리고 ρ 는 네트워크의 전기적 특성을 고려한 지연시간이며 네트워크 특성에 따라 정수로 표현된다.

2.4 네트워크 시간특성과 운영체계 시간특성의 비동기 성에 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. 이러한 주기의 비동기성은 각각의 최악 응답 특성에 추가하여 대기 지연시간을 발생시키게 된다.

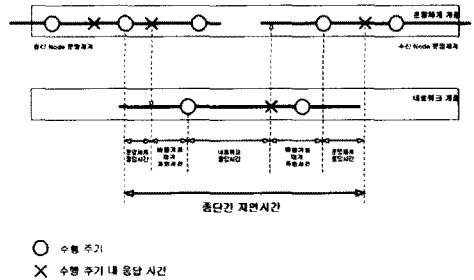


그림 2. 종단 간 지연시간의 구성

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

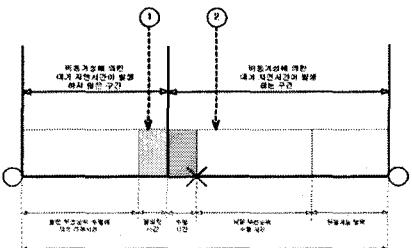


그림 3. 비동기성에 의한 지연 시간의 발생 구역

비동기성에 의한 지연 시간은 그림 3에서 보는 바와 같이 선행 수행부의 응답 시간이 후행 수행부의 수행 주기에서 ②와 같이 수행 시간 이후에 나타날 경우 발생한다. 만약 선행 수행부의 응답 시간이 ①과 같이 수행 시간 이전에 나타난다면 비동기성에 의한 지연 시간은 발생하지 않는다. 비동기성에 의한 최악의 대기 지연시간은 선행 수행부의 응답 시간이 수행 시간

이후에 나타나고, 또한 후행 수행부의 높은 우선순위 작업 수행에 의한 지연 시간과 블로킹 시간으로 구성되는 대기시간이 최소가 될 때이다. 후행부의 최소 지연시간은 식(6)과 같고, 최소 지연시간을 가질 때의 최악의 대기 지연시간은 식(7)과 같다.

$$D_{min_i} = \sum_{\forall j \in (hp(i) \cap harm(i))} C_j \quad (6)$$

$$W_{max_i} = T_i - D_{min_i} \quad (7)$$

여기서, $harm(i)$ 는 태스크 i , 혹은 메시지 i 주기의 역(power)을 주기로 갖는 태스크 혹은 메시지의 집합이다. 즉 태스크 i , 혹은 메시지 i 와 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 같다.

$$D_{ele_i} = r_i(trans) + W_{max_i}(net) + r_m + c_m + W_{max_i}(recv) + r_i(recv) \quad (8)$$

여기서, D_{ele_i} 는 종단 간 최악 지연시간을 의미하며, $r_i(trans)$ 은 송신부 운영체계 계층에서의 최악 응답시간, $W_{max_i}(net)$ 은 네트워크 계층에서의 비동기성으로 인한 최악의 대기 지연시간, $W_{max_i}(recv)$ 는 수신부 운영체계 계층에서의 비동기성으로 인한 최악의 대기 지연시간, 그리고 $r_i(recv)$ 는 수신부 운영체계 계층에서의 최악 응답시간을 의미한다.

2.5 글로벌 클럭 동기화 알고리즘

네트워크 절대시간인 글로벌 클럭은 마스터 클럭(master clock)을 갖는 제어부에서 발생된다. 주기적으로 발생되는 마스터 클럭 메시지에 의해서 각 노드의 슬레이브로 커널 클럭(slave local clock)은 지속적인 갱신이 되어 네트워크 상의 전체 노드가 동일한 글로벌 클럭을 갖게 된다.

본 논문에서는 글로벌 클럭 메시지 전송으로 인해 발생되는 통신 오버헤드를 최소화하기 위해 마스터 클럭 메시지의 주기를 온라인상에서 가변하는 온라인 글로벌 클럭 동기화 알고리즘을 제안한다. 제안된 알고리즘은 마스터 클럭 메시지 주기를 메시지 실시간성을 보장하는 네트워크 이용률(utilization)과 시스템이 허용하는 최대오차를 파악해 결정한다.

마스터 클럭 발생 노드가 네트워크의 사용 대역폭을 확인하기 위해 본 논문에서는 확장 식별자 필드(extended ID field)를 사용한다[6]. 확장 식별자 필드의 구성은 그림4와 같다.

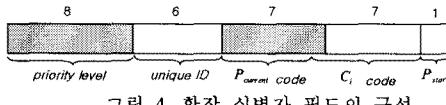


그림 4. 확장 식별자 필드의 구성

확장 식별자 필드는 5가지 정보를 갖는다. 'priority level'은 버스점유 우선순위를 나타내고, 'unique ID'는 메시지를 구별하는 유일한 비트정보이다. ' C_i code'는 메시지 전송시간을 나타내기 위한 비트정보이다. 따라서 ' C_i code'의 값은 1~8의 값을 가진다. 끝으로 ' P_{start} '값은 $kP+t$ ($t=0$)일 때 중재에 나선 메시지는 비트 1을 갖고, 그 후의 메시지는 비트 0을 갖는다. CAN 물리 계층의 특성상 전송데이터는 모든 노드에 브로드캐스팅(broadcasting)되므로 항상 각 노드는 자신의 정보를 유지할 수 있게 된다.

네트워크 이용률(U)은 식(9)와 같이 $P_{current}$ 값과 $B_{current}$ 값에 의해 알 수 있다. 여기서 $P_{current}$ 는 현재까지 발생한 전체 메시지 주기의 최소공배수이고, $B_{current}$

는 $P_{current}$ 에서 전송시간으로 사용될 것으로 예상되는 총 네트워크 점유시간이다. 따라서 이 두 값을 구하면 네트워크 이용률을 알 수 있다. $P_{current}$ 와 $B_{current}$ 값을 구하는 과정은 다음과 같다.

- 1) 현재의 $P_{current}$ 와 자신의 주기 T_i 의 값을 비교.
- 2) $T_i \geq P_{current}$ 이면, T_i 의 값과 C_{icode} 값을 식별자 필드에 넣어 전송.
- 3) 버스점유 후 $T_i \geq P_{current}$ 이면, $P_{current}$ 에서의 총 전송시간인 $B_{current}$ 는 $(T_i/P_{current})B_{current} + C_i$ 값으로 변경

$$U = \frac{B_{current}}{P_{current}} \quad (9)$$

$$UB = n\left(\frac{1}{n} - 1\right) \quad (10)$$

$$T \geq \frac{C}{UB - \frac{B_{current}}{P_{current}}} \quad (11)$$

RMS의 네트워크 이용률 한계치(UB: Utilization Bound)는 식(10)과 같고, 이를 적용하면 마스터 클럭 메시지의 최소주기는 식(11)과 같다.

마스터 클럭 메시지의 최대주기 설정은 시스템이 허용하는 최대오차를 벗어날 수 없게 하는 것이다. Linux와 같은 현대의 운영체계는 절대적인 시간 측정의 수단으로 부정확한 실시간 클럭(real time clock)을 사용하지 않고 비교적 정확도가 뛰어난 오실레이터(oscillator)와 같은 시스템 클럭을 사용하여 소프트웨어적인 인터럽트로 내부의 절대시간 값을 생성한다.

시스템 클럭의 최대 허용 오차가 $\pm \rho$, 시간(t)의 클럭값이 $H(t)$, 시간(s)의 클럭값이 $H(s)$ 라면, 시간 간격[s, t]에서 클럭의 부정확도에 의한 왜곡 범위는 식(12)와 같다[7].

$$(1 - \rho)(t - s) \leq H(t) - H(s) \leq (1 + \rho)(t - s) \quad (12)$$

여기서, 시간 간격[s, t]에서의 최대 오차는 $2\rho \times (t - s)$ 와 같게 된다.

$$T \leq \frac{t - s}{2\rho + G} \quad (13)$$

소프트웨어적인 인터럽트로 절대시간을 생성하는 경우 필연적으로 발생하는 요소는 인터럽트 지연시간(interrupt latency)이다. 이러한 인터럽트 지연 요소 G 와 식(12)을 고려하여 시간 간격[s, t]의 정확도를 유지하기 위한 마스터 클럭 메시지의 최대주기는 식(13)과 같다.

따라서 글로벌 클럭 동기화를 위한 마스터 클럭 메시지의 주기는 식(14)의 범위 안에서 가변적으로 결정되어 진다.

$$\frac{C}{UB - \frac{B_{current}}{P_{current}}} \leq T \leq \frac{t - s}{2\rho + G} \quad (14)$$

III. 모의실험 및 결과

본 논문에서 제안한 식(8)의 타당성을 입증하기 위해 표 1의 시간 특성 조건에 따른 모의 종단 간 응답 시간의 모의실험을 수행한다. 표 1에서 1번과 5번 태스크는 분산 제어 시스템의 메시지 통신과 무관한 운영체계의 독립 태스크이며, 메시지 1번과 5번 또한 제어 시스템 정보와 무관한 독립적인 메시지이다. 분산 제

이 시스템 상에서 송신 노드의 태스크2~4는 각각 메시지 2~4를 이용하여 수신노드의 태스크2~4로 제어 정보를 전달하게 된다. 본 논문에서는 각각의 송신 노드에서 네트워크 메시지를 통해 수신 노드로 전달되는 종단 간 경로를 실시간 분산 제어 채널로 정의한다. CAN의 메시지 전송 속도는 125kbps로 가정하였으며, 네트워크 메시지의 지터는 0으로 가정하였다. 또한 비동기성에 의한 대기 지연시간의 명확한 도시를 위해 낮은 우선순위 태스크 혹은 메시지에 의한 블로킹 시간은 항상 0이 되는 것으로 가정하였다.

표 1. 종단 간 응답 시간 모의실험을 위한 태스크 및 메시지 짐합의 시간 특성

No.	1	2	3	4	5
송신노드	주기	100mS	200mS	300mS	500mS
태스크	연산시간	20mS	30mS	50mS	50mS
네트워크	주기	100mS	300mS	500mS	700mS
메시지	전송바이트	8Byte	8Byte	8Byte	8Byte
수신노드	주기	100mS	200mS	400mS	400mS
태스크	연산시간	20mS	50mS	50mS	50mS

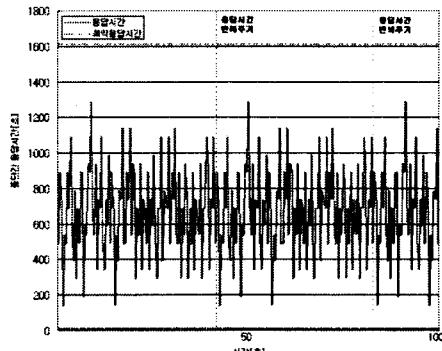


그림 5는 표 3의 실시간 분산 제어 채널 4에 대해 102,990mS 동안 각각의 실시간 분산 제어 채널의 응답 시간을 모의 실험한 결과이다. 그림 5에서 보는 바와 같이 응답 특성은 42,590mS 간격의 주기적인 형태로 응답 시간의 발생 형태가 반복적으로 변화한다. 따라서 100초 이후에도 각 분산 제어 채널의 종단 간 응답 시간은 본 논문에서 제안한 최악의 종단 간 지연시간을 넘지 않음을 알 수 있다.

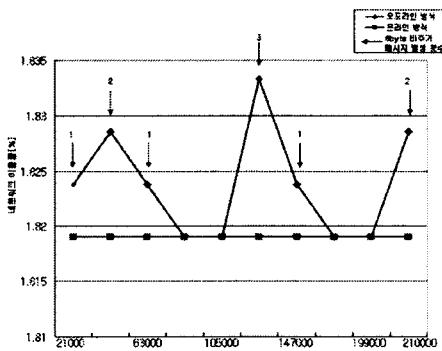


그림 6. 오프라인 방식과 온라인 방식의 글로벌 클럭 동기화 기법에 따른 네트워크 이용률의 비교

본 논문에서 제안한 온라인 글로벌 클럭 동기화 알고리즘의 효율성을 입증하기 위해 표1의 메시지 짐합

에 대해, 주기 100mS와 8 바이트의 전송 크기를 갖도록 오프라인 상에서 고정된 글로벌 클럭 메시지를 이용한 방법과 제안된 방법의 유용 자원 확보 성능을 모의실험을 통해 비교한다.

모의실험은 일반적인 실시간 Linux의 기본 스케줄링 인터럽트가 10mS인 것을 감안하여, 5mS에서의 정확도를 유지하기 위해 $(t-s)$ 값을 5mS로 설정하였다. 또한 기존의 실시간 Linux 성능 평가[8]에 의해 인터럽트 지연 시간을 150uS로 가정하였으며, 일반적인 시스템 클럭의 정확도에 따라 ρ 를 200uS로 가정하였다. 또한 그림 6에서 보는 바와 같이 순간적인 비주기 메시지의 발생을 고려하였다.

그림 6에서와 같이 오프라인 방식의 글로벌 클럭 동기화 기법을 사용하는 경우 비주기 메시지의 발생시에 네트워크 이용률이 증가하는데 반하여, 온라인 글로벌 클럭 동기화 기법을 사용하는 경우 네트워크 이용률이 일정함으로써 일정한 가용 네트워크 대역폭의 확보가 가능하게 된다. 온라인 글로벌 클럭 동기화 기법의 이러한 성능은 제어 시스템에 예상하지 못한 부하가 발생하였을 경우, 충분한 처리 능력을 보장하게 된다.

3. 결 론

본 논문에서는 네트워크 기반 실시간 시스템의 실질적인 종단 간 지연시간 요소를 운영 체계의 태스크 스케줄링에 의해 발생하는 지연시간인 운영체계 지연시간, CAN의 메시지 스케줄링에 의해 발생하는 지연시간, 네트워크 지연시간, 그리고 이 두 가지 지연시간 특성의 비동기성에 의해 발생하는 대기지연 세 가지로 정의하고, RMS 방식의 운영체계 스케줄링과 네트워크 스케줄링을 대상으로 하여 최악 지연시간의 분석을 수행하였다.

또한 본 논문에서는 통신 오버헤드를 최소화하기 위해 온라인 글로벌 클럭 동기화 알고리즘을 제안하였다. 제안된 글로벌 클럭 동기화 알고리즘은 온라인상에서의 네트워크 트래픽 분석과 각 네트워크 노드에서 실시간 운영체계의 로컬 클럭 드리프트를 감안하여 이루어진다. CAN을 기반으로 한 모의실험을 통하여 기존의 오프라인 글로벌 클럭 동기화 알고리즘과 비교하여 충분한 클럭 정확도를 유지하면서 네트워크 이용률을 최소화할 수 있음을 입증하였다.

참 고 문 헌

- [1] Bjorn Wittenmark, Ben Bastian, and Johan Nilsson, "Analysis of Time Delays in Synchronous and Asynchronous Control Loops", Decision and control, 1998. Proceedings of the 41st IEEE Conference on, Vol. 4, pp. 4637-4642, 2002.
- [2] 김홍렬, 곽권천, 김대원, "CAN 기반 피드백 시스템의 한국형 고속전철 여객시스템 적용", 제어자동화시스템 공학회 논문지, Vol.9, No.11, pp.948-953, 2003
- [3] M. Farsi, K. Ratcliff and Manual Barbosa, "An overview of Controller Area Network", COMPUTING & 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] John Lehozczky, Lui sha, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", Proc. of IEEE Real-Time Systems Symposium, 1989.
- [6] 이병훈, 김홍렬, 김대원, "CAN기반 실시간 시스템을 위한 확장된 EDS 알고리즘 개발", 대한전기학회논문집, Vol.51D No.5, pp.294-302, 2002.
- [7] Johan Nilsson, "Real-Time Control Systems with Delays" Phd Thesis, Lund Institute of Technology, 1998.
- [8] Benjamin Ip, "Performance Analysis of VxWorks and RTLinux", <http://contest.kesic.or.kr/>.