Controller Area Network의 메시지 지연시간 계산 및 성능 분석

°조남현* , 김진경** , 김동준* , 김대원*** , 김갑일***
*명지대학교 전기공학과 , **명지대학교 제어계측공학과, ***명지대학교 전기전자공학부

Calculation of Message Latencies and Performance Analysis in the CAN(Controller Area Network)

Cho, Nam Hyun* , Kim, Jin Kyong** , Kim, Dong Jun* , Kim, Dae Won*** , Kim Kab Il***
*Dept. of Electrical Eng., Myong Ji Univ.
**Dept. of Conrol & Instrumentation Eng. Myong Ji Univ.
***Division of Electrical and Electronics Eng. , Myong Ji Univ.

Abstract

This paper presents the worst-case time-delay in CAN(Controller Area Netwokr) system through the mathematical model. Time-delay in the CAN system is devided into a generation delay, a transmission delay and a delivery delay. This paper also measures the performamce represented in terms of the number of station, transmission speed, data size etc. by using the computer simulation.

1. 서 론

생산의 비용을 절감하고 시스템의 성능을 최적화 하려는 경향이 증대됨에 따라 경쟁성을 유지하기 위해 많은 제조업체들은 개방 시스템의 네트웍에 투자를 하고 있다. 이에 1980년대 초에는 생산자동화 환경에 적합한 표준화된 네트웍으로 MAP(Manufacturing Automation Protocol)이 개발되었다. OSI의 기준모델에서 제시하는 7계층을 모두 갖는 MAP은 자동화에 따라 새로운 제어방식을 요구하는 하위 레벨의 센서나 작용기(actuator), 각종 제어기기 등의 비용이나 실시간 특성, 그리고 규격으로 인한 호환성의 문제를 안게되었다. 이러한 문제 해결의 필요성으로 새로운 국제 표준 네트웍이 필요했고 그 결과 하위 레벨의 통신 네트웍으로서 필드버스 시스템이 등장하였다[1].

CAN은 1988년 Bosch와 Intel에서 차량용 네트웍 시스템을 목적으로 개발되었으며 물리 계층과 데이터 링크 계층만으로 구성되어 있다. 일반적으로 CAN은 긴급한 데이터를 전송하는데 효율적이라 생각을 하지만 실제로 그렇지 않은 데이터를 전송하는 경우 그 전송지연시간을 보장하기가 어렵다. CAN에서 사용되는 동적 스케줄링 알고리즘은 실시간 시스템에서 프로세서내의 태스크를 위한 스케줄링 알고리즘과 실질적으로 동일하다. 그러나 이 알고리즘은 CAN에서 주어진 메시지의 최악의 경우 지연시간을 결정하는 문제가 다루어지지 않고 통신 문제에 적용되었다[8]. 따라서 본 논문에서는 이 문제를 CAN에 적용하여 데이터 링크 계층에서 발생될 수 있는 지연시간을 수학적 모델로 만들고, 그 모델을 기초로 CAN의 성능을 분석하고자 한다.

2. CAN 프로토콜의 개요

CAN은 통신을 위해 데이터 프레임(frame), 원격(remote) 프레임, 에러 프레임, 오버로드(overload) 프레임의 4가지 형태의 프레임을 제공한다. 데이터 프레임은 데이터를 전송할 때 CAN 컨트롤러가 전송하는 프레임이며 원격 프레임은 데이터를 요구할 경우에 사용되며 데이터 프레임에서 데이터 영역만 제외된 것이다. 에러프레임은 감지된 에러를 알리는데 사용되며 오버로드 프레임은 연속된 프레임 사이에서 지연시간을 늘리기 위해 사용된다. 그림 1은 데이터 프레임을 나타낸다.

그림1의 데이터 프레임내의 조정(arbitration) 영역에서 식별자(identifier)는 메시지의 타입과 우선순위를 정의한다. CAN은 우선순위버스를 사용한다. 각

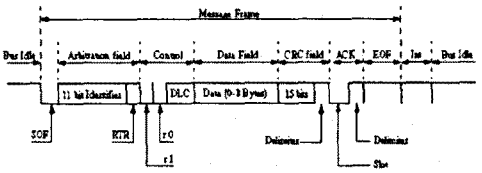

그림 1 CAN 데이터 프레임 형식

메시지에 우선순위를 나타내는 식별자를 부여하여

전송하는 경우 다른 메시지가 나타나는 경우 조정 기능을 통해 메시지의 충돌을 방지한다. 그리고 수신 모드의 CAN 컨트롤러는 메시지의 식별자와 자신의 식별자가 일치하면 메시지를 받아들인다. 또한 어떤 정보에 대해서는 수신 모드로 동작하는 CAN 컨트롤러가 특정 데이터의 RTR 비트를 high로 한 원격 프레임을 네트웍으로 보냄으로써 특정 데이터의 전송을 요구할수도 있다. 또한 송신모드에 있는 CAN컨트롤러는 각각의 메시지에 대해 CRC를 생성시켜 CRC 영역에 담아 네트웍으로 전송하면 수신모드에 있는 CAN 컨트롤러는 CRC를 체크하여 에러가 발생하였는 지를 감지한다[3,5].

3. CAN 데이터링크 계층의 수학적 모델

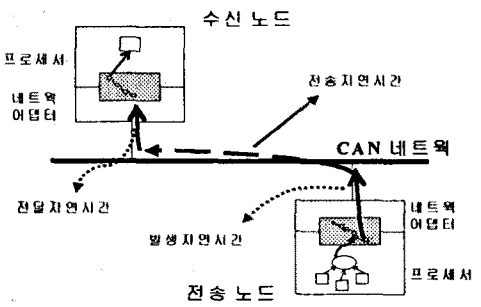

그림 2. CAN의 네트웍 모델

그림2는 CAN의 시스템 모델을 나타낸다. 그림2와 같이 하나의 버스에 여러 노드가 CAN 인터페이스를 통해 연결되어 있다. 한 노드의 프로세서와 네트웍 어댑터 사이에는 공유 메모리를 가진다. 프로세서 내에서 스케줄링 된 태스크에서 발생된 메시지는 이 메모리를 통해 버스로 전송하게 된다. 이 때 버스가 다른 메시지의 전송을 하지 않는다면 노드는 이 메시지의 전송을 준비한다. 만약 이미 전송중이거나 전송중이 아니더라도 다른 노드에서 메시지의 전송 시도에 의해 충돌이 일어난 경우엔 조정 과정을 통해 메시지의 스케줄링이 되어 우선순위가 높은 메시지가 전송된다. 메시지를 전송하여 수신하는데까지 걸리는 응답 시간을 구하기 위해서는 이러한 메시지 스케줄링과 전송 과정을 거치는 동안 최악의 경우에 나타나는 지연시간을 구하여 응답시간을 보장한다. 메시지의 스케줄링과 전송 문제를 풀기 위해 고정된 우선순위 선점 스케줄링 기법을 사용한다[6, 7, 9].

네트웍에 나타나는 문제중 하나가 지터(jitter)인데 이 지터의 발생은 두가지로 볼 수 있다. 먼저 메시지의 실행 주기보다 메시지의 도착 주기가 작을 때 즉, 한 메시지의 실행이 끝나기도 전에 다음 주기의 메시지가 도착한 경우에 발생하며 디지털 신호에 따르는 각 메시지의 tick 값의 차이가 생기므로 지터가 발생한다.

(1) 발생 지연 시간

프로세서 내에서 태스크가 메시지를 발생하여 메모리의 큐(queue)까지 전달하는데 걸리는 시간을 발생 지연시간이라 정의한다[6]. 이는 최악의 경우 태스크 i의 발생지연 시간은 다음으로 나타낼수 있다.

$$r_i = \max_{q=0,1,2,3,\ldots} (w_i(q) + J_i - qT_i) \qquad (1)$$

여기서 T_i는 최악의 경우 태스크 i의 주기를 의미하며 $w_i(q)$는 qT_i번째 태스크가 발생하기 전의 레벨 i 지연 구간이며 J_i는 태스크 i의 지터이다. 여기서 level i busy 구간은 태스크 i가 연속적으로 발생하여 각 메시지가 큐에 도달하는 시간의 합을 말한다. 이것은 고정된 우선순위 스케줄링 이론의 주요 개념인데 CAN 네트웍에서는 프로세서 내에서 이루어지는 태스크 스케줄링과 관련된 processing busy 구간과 버스내에서 이루어지는 메시지 스케줄링과 관련하여 communication busy 구간으로 나뉜다. processing busy 구간의 시간 즉 w는 다음과 같다.

$$w_i(q) = (q+1)C_i + B_i + \sum_{\forall j \in hp(i)} I_j \qquad (2)$$

이 구간은 자신의 실행 시간과 높은 우선순위의 태스크에 의해 지연된 시간을 더하여 자신의 주기보다 클경우에 계속 이어지며 다음의 조건이 될 때 level i busy 구간은 멈추게 된다.

$$w_i(q) \le (q+1)T_i \qquad (3)$$

hp(i)는 태스크 i 보다 우선순위가 높은 태스크의 집합을 말하며 I_j는 프로세서가 $w_i(q)$ 구간동안 태스크 j를 실행하는데 걸리는 시간을 의미하며 다음과 같이 나타낸다.

$$I_j = \left\lceil \frac{w_i(q) + J_j}{T_j} \right\rceil C_j \qquad (4)$$

C_j는 최악의 경우 태스크 j의 실행 시간이며 T_j는 태스크 j의 주기이다. B_i는 최악의 경우 낮은 우선순위의 태스크에 의해 지연된 시간을 의미한다[7].

(2) 전송 지연 시간

큐에 도달한 메시지가 목적 노드까지 전송되는데 지연되는 시간이다. 개념은 발생 지연 시간에서 구한 것과 유사하지만 버스에 메시지를 전송할 때 나타나는 전기적인 전파지연시간과 전송속도에 따른 실행시간의 변화가 추가된다. 최악의 경우 메시지 i의 최대 응답시간은 다음과 같다.

$$r_i = \max_{q=0,1,2,\ldots}(w_i(q) + J_i - qT_i + \delta) \qquad (5)$$

T_i는 최악의 경우 메시지 i의 전송주기이며 $w_i(q)$는 qTi번째 메시지가 발생하기 전의 level i busy 구간, 그리고 J_i는 메시지 i의 지터이며 δ는 전기적 전파지연시간을 나타낸다. level i busy 구간 역시 발생지연시간인 경우와 같이 메시지 i보다 우선순위가 높은 메시지가 자신의 노드내에 발생하거나 다른 노드의 메시지가 생길 경우에 증가하게 된다.

$$w_i(q) = (q+1)C_i + B_i + \sum_{\forall j \in hp(i)} I_j \qquad (6)$$

hp(i)는 메세지 i 보다 우선순위가 높은 메시지의 집합을 말하며 I_j는 $w_i(q)$ 구간동안 메시지 j를 전송하는데 걸리는 시간을 의미하며 다음과 같이 나타낸다.

$$I_j = \left\lceil \frac{w_i(q) + J_j}{T_j} \right\rceil C_j \qquad (7)$$

C_j는 최악의 경우 메세지 j의 전송 시간이며 T_j는 메세지 j의 주기이다. B_i는 최악의 경우 낮은 우선순위의 메세지에 의해 지연된 시간을 의미한다. 위의 식에서 나타나는 전송시간 C는 다음과 같이 쓸 수 있다.

$$C = \left(\left\lfloor \frac{34 + 8s_m}{5} \right\rfloor + 47 + 8s_m \right) \tau_{bit} \qquad (8)$$

여기서 s_m은 메시지의 길이를 나타내며 바이트의 단위를 갖는다. τ_{bit}은 버스의 비트시간이다. 이는 예를들어 버스가 1Mbit/sec으로 동작할 때 1μs 값을 갖는다[8]. $w_i(q)$의 중단 조건 역시 다음과 같다.

$$w_i(q) \le (q+1)T_i \qquad (9)$$

(3) 전달 지연 시간

이제까지 프로세서 내의 태스크에서 메시지가 발생하여 목적 노드의 프로세서까지 도착하는 과정에 관한 수학적인 모델을 알아보았다. 도착한 메시지는 프로세스 내의 소프트웨어에 의해 태스크로 전달되는데 이 전달과정중에 나타나는 지연시간을 계산하여 최악의 경우 전달 지연 시간의 경계값을 구하고자 한다.

우선 태스크 i의 발생 이전에 $w_i(q)$ 내의 계산시간을 구한다. 목적 노드의 큐로 전달되는 메시지는 비주기적으로 전달되지만 최소한의 주기를 갖는데. 큰 외부주기(T_i)와 작은 내부주기(t_i)로 나뉘어 연속적으로 발생한다. 이 경우 내부주기의 합이 외부주기보다 커서는 안된다.

$$n_i t_i \le T_i \qquad (10)$$

M_i와 m_i는 외부주기와 내부주기의 번호를 의미하며 다음과 같이 주어진다.

$$M_i = \left\lfloor \frac{q}{n_i} \right\rfloor \qquad (11)$$

$$m_i = q - M_i n_i \qquad (q = 0,1,2,3\ldots) \qquad (12)$$

현재의 태스크 i의 실행은 $m_i t_i + M_i T_i - J_i$에서 시작하며 이 태스크가 발생하기 이전까지 총 실행시간의 합은 다음과 같다.

$$M_i n_i C_i + m_i C_i \qquad (13)$$

이제부터는 $w_i(q)$ 내에서 높은 우선순위를 가지는 하나의 태스크 j에 의해 지연되는 시간을 구해보자. $w_i(q)$내에 존재하는 태스크 j의 외부주기 개수를 구하는 식은 아래와 같다.

$$F_j = \left\lfloor \frac{J_j + w_i(q)}{T_j} \right\rfloor \qquad (14)$$

높은 태스크에 의해 지연된 시간을 뺀 $w_i(q)$의 시간은 $J_j + w_i(q) - F_j T_j$와 같고 이 나머지 시간에 높은 우선순위의 태스크 발생 개수의 한계를 정하면 아래식으로 쓸 수 있다.

$$\left\lceil \frac{J_j + w_i(q) - F_j T_j}{t_j} \right\rceil \qquad (15)$$

높은 우선순위의 태스크는 $w_i(q)$내에 n_j만큼 발생할수도 있으므로 나머지 시간동안의 한계값은 n_j로 정의할수도 있다. 따라서 식(15)는 다음과 같이 쓴다.

$$\min\left[n_j, \left\lceil \frac{J_j + w_i(q) - F_j T_j}{t_j} \right\rceil \right] \qquad (16)$$

결국 우선순위가 높은 모든 태스크에 의해 지연되는 시간은 아래와 같다.

$$\sum_{\forall j \in hp(i)} \left[\min\left[n_j, \left\lceil \frac{J_j + w_i(q) - F_j T_j}{t_j} \right\rceil \right] + F_j n_j \right] C_j$$

(17)

따라서, 태스크 i의 최악의 경우 응답시간은 식(18)과 같이 주어지고

$$r_i = \max_{q=0,1,2,3....}(w_i(q) + J_i - m_i t_i - M_i T_i) \qquad (18)$$

여기서 $w_i(q)$는 식(19)와 같다.

$$w_i(q) = (M_i n_i + m_i + 1)C_i + B_i + \sum_{\forall j \in hp(i)} \left[\min\left[n_j, \left\lceil \frac{J_j + w_i(q) - F_j T_j}{t_j} \right\rceil \right] + F_j n_j \right] C_j \qquad (19)$$

$w_i(q)$의 중단조건은 $w_i(q) \le M_i T_i + m_i t_i - J_i$이다.

4. 시뮬레이션 및 결과

본 시뮬레이션에서는 16개의 노드에서 주기적/비주기적 메시지를 발생시킨다. 주기적 메시지의 평균 발생주기는 30ms이며 데이터 길이는 1~8바이트까지 변화한다. 또한 비주기적 메시지는 지수 분포로 발생한다고 가정하였고 평균발생빈도 1/5ms, 1/10ms, 1/20ms인 세가지의 경우를 다루었으며 데이터의 길이는 1byte로 가정하였다. 그리고 전송속도는 1Mbps, 500Kbps, 250Kbps, 125Kbps로 변화한다고 가정하였다.

그림 3과 4는 1Mbps와 125Kbps의 경우에 나타나는 시스템 지연 시간을 나타낸다. 시스템 지연시간은 메시지가 발생하여 큐에 삽입되어 목적 노드까지 전송하는데까지의 지연시간이다.

그림3과 4를 보면 예상했던대로 전송속도가 높을수록 시스템지연시간은 자연히 감소하며 전송 데이터가 길어질수록 지연시간이 증가함을 알수 있다. 또한 비주기 메시지의 발생빈도가 높아질수록 지연시간도 증가함을 볼 수 있다.

그림5는 비주기적 메시지의 평균발생빈도가 1/10ms인 경우 조정한 후 각 전송속도에 따라서 전체의 메시지 중에 데드라인을 만족시키지 못하여 전송되지 못한 메시지의 확률을 나타낸다.
그림5에서와 같이 1Mbps, 500Kbps의 경우에는 시스템지연시간이 작으므로 전송 실패율은 데이터의 크기에 관계없이 0이다. 그러나 125Kbps인 경우에는 시스템 지연시간이 커지므로 데드라인을 만족시키지 못한 경우가 많아져 전송실패율 또한 증가함을 볼 수 있다.

그림 6는 125Kbps의 전송속도인 경우 3가지의 비주기적 메시지 평균발생빈도에 대하여 전송메시지의 실패확률을 보여준다. 비주기 메시지의 발생빈도와 데이터의 크기가 높아짐에 따라 전송실패율은 증가함을 알 수 있다.

5. 결론

본 논문에서는 CAN의 데이터링크 계층의 수학적 모델과 시뮬레이션을 통한 성능 분석을 수행하였다. 먼저 CAN 시스템 모델의 동작을 분석하여 최악의 경우 나타나는 발생지연시간, 전송지연시간, 전달지연시간을 구하였다. 또한, 본 논문의 CAN 모델을 이용하여 주어진 CAN 통신 환경에서 주기적 메시지 발생주기, 비주기적 메시지 전송속도, 데이터 크기, 노드수 등의 변화를 고려한 네트웍의 성능분석을 위해 시뮬레이션을 수행하였다.

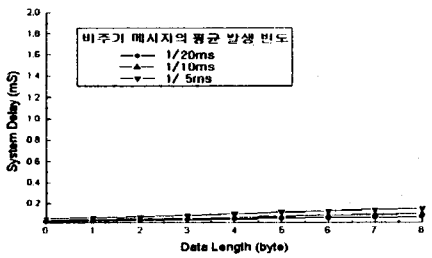

그림 3. 1Mbps인 경우 시스템 지연

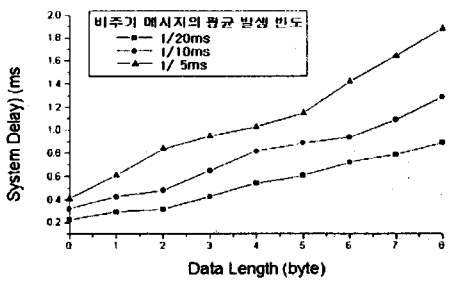

그림 4. 125Kbps인 경우 시스템 지연

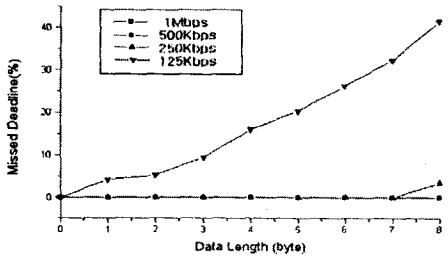

그림 5. 전송속도에 따른 메시지 전송 실패 확률

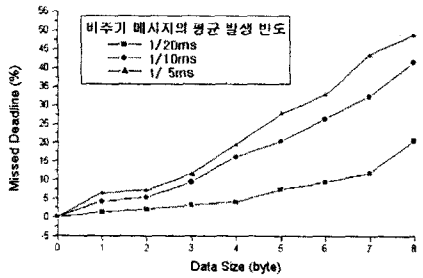

그림 6. 125Kbps인 경우 전송실패확률

6. 참고문헌

[1] J.D.Decotignie, P.Pleinevaux, "A survey on industrial communication networks," *Annales des telecommunications*, Vol.48, No. 9-10, pp. 435-448, 1993.

[2] C. Demartini and G. Cena, "Real-time communication in the Factory automation," *IECON 92*, pp. 772-777, Sep. 1992

[3] INTERNATIONAL STANDARD ISO 11898 :1993

[4] Tindell, K., Hansson, H., & Wellings, A.J. "Analyzing Real-Time Communications", Real-Time Systems Symposium, San Juan, Puerto Rico, December 1994

[5] CAN Specification, BOSCH, Part A, 1991

[6] Tindell, K., "Analysis of Hard Real-Time Communications", YCS 222, Department of Computer Science, University of York(January 1994)

[7] Tindell, K., Burns, A., and Wellings, A., "An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks," *Real-Time Systems* 6(2) pp.133-151

[8] Tindell, K. and A. Burns, "Guaranteed Message Latencies on Control Area Networks", First International CAN Conference, Mainz, Germany, Sep. 1994

[9] Burns, A., Tindell, K. Wellings, A., "Fixed Priority Scheduling with Deadlines Prior to Completion", *Proceedings Sixth Euromicro Workshop on Real-time Systems*, IEEE Computer Society Press (June 1994)