論 文
50D-5-2

CAN 프로토콜을 이용한 네트워크 시스템의 성능 분석

Performance Analysis of a Network System using the CAN Protocol

金 大 元* · 崔 煥 洙*
(Dae Won Kim · Hwan Soo Choi)

Abstract - This paper analyses the performance of network system using the CAN(Controller Area Network) protocol. Given messages are assumed to be scheduled by the DMS(Deadline Monotonic Scheduling) algorithm. The mathematical models for time-delay that can be occurred between CAN nodes are defined. The effectiveness of modeling is shown by comparing the difference of time-delay between simulations and practical experiments. We analyse the results according to the variation of factors, such as the number of nodes, the transmission speed, the message size and the number of aperiodic messages through simulation and confirm the real-time performance of lower priority messages. We also investigate the real-time performance of periodic messages when aperiodic message generates.

Key Words : CAN, time-delay, real-time, priority, network system

1. 서 론

최근 국내외에서는 네트워킹과 정보공유를 통해 생산성을 높이고 생산비용을 감소시킬 수 있는 개방형 제어 시스템 구조에 대한 관심이 높아지고 있다. 개방형 제어 시스템의 요소 중 개방형 네트워크 시스템에 대한 관심은 1980년대 필드버스(fieldbus)의 등장으로 시작되어 이제는 기술적 성숙기에 접어들고 있다. 필드버스 시스템은 단일 케이블을 사용하여 배선 작업이 용이하고 설치비용이 적게 들며, 변형이 용이하여 시스템 운용에 유연성을 제공하는 장점이 있다. 이러한 필드버스 장점 때문에 다양한 규격을 갖는 필드버스들이 등장하였으며 현재 Profibus, Fieldbus Foundation, CAN, DeviceNet 등이 주요 시장을 형성하고 있다.

CAN(Controller Area Network)은 1988년 Bosch[1]와 Intel에서 차량용 네트워크 시스템을 목적으로 개발되었으며, 물리계층과 데이터 링크 계층만으로 구성되어 전송속도가 빠르고 각 메시지에 우선순위를 부여해 전송간에 충돌을 방지할 수 있다. CAN은 방직기계, 엘리베이터, 이동로봇 그리고 산업 자동화 시스템의 네트워크 구성에 응용되고 있다. CAN은 구성이 용이하고 가격이 저렴하여 점점 응용분야가 늘어나고 있는 추세이다. 그러나, 많은 경우 네트워크 환경을 고려한 트래픽 분석없이 시스템 구현을 진행함에 따라 효율적 시스템 부하의 분배나 구조 설계가 불가능하다. 따라서 본 논문에서는, CAN 프로토콜을 이용해 네트워크 시스템을 구성할 때 발생될 수 있는 네트워크 트래픽 문제의 분석을 위해 CAN 프로토콜의 모델링과 대상 시스템에 대한 일반적 성능분석을 수행하고자 한다.

먼저 CAN 프로토콜 모델링에 대한 연구를 살펴보면, Tindell[2][3]은 점대점 통신(end-to-end communications)상의 지연시간에 대해 네가지로 정의하였고, 그 중 미디어 접근 지연시간(media access delay)과 전달지연시간(delivery delay)에 대한 분석을 하였다. 또한 토큰 프로토콜(token protocol)과 우선순위 버스(priority bus)을 이용해 지연시간을 분석하였고, 통신 스케줄링을 통해 발생되는 메시지에 대한 실시간성 보장 문제를 다루었다. 또한, Tindell과 Burns[4]는 전송 지연시간과 대기행렬 지연시간에 대한 모델링과 최악의 응답시간을 분석하고, 'SAE' 벤치마크에 적용시켜 CAN을 이용한 네트워크 시스템의 지연시간에 대한 분석을 하였다. 그러나 여기서 발생가능한 모든 지연시간에 대한 분석이 고려되지 않았다. 따라서, 본 논문에서는 CAN 프로토콜을 이용한 네트워크 통신상의 지연시간을 발생지연시간, 대기행렬지연시간, 전송지연시간 그리고 전달지연시간으로 구분하여 모델링 하고자 한다. 메시지 스케줄링에 대해서는 정적인 우선순위 알고리즘으로 RMS(Rate Monotonic Scheduling)와 DMS(Deadline Monotonic Scheduling) 알고리즘들이 제안되었다[5,6]. 그러나, RMS 알고리즘은 모든 메시지가 주기적이어야 한다는 제약조건을 가지고 있으므로 Audsley[6]는 DMS 알고리즘을 이용하여 주기적 메시지와 비주기적인 메시지에 대해 스케줄링하였다. Tindell[7]은 고정된 우선순위를 가지는 하드 실시간 태스크(hard real-time task)에 대해 좀더 유연하고 확장된 스케줄링 방법을 제안하였고, 비주기적 메시지에 대해서 비주기적 메시지의 발생이 제한된 주기내에서 처리된다는 가정하에 스케줄링하였다. 그러나, 일반적인 CAN 통신은 실시간성의 보장 특성이 좋다고 하나 실제로는 낮은 우선순위를 가지는 메시지에 대해서는 실시간성의 보장이 어렵다. 따라서, 이에 본 논문에서는 낮은 우선순위를 가지는 메시지의 실시간성 보장에 대해 다루며, 비주기적 메시지의 실시간성에 대하여 확률 분포를 이용하여 모델링을 한 후 각 메시지의 실시간성 문제를 다룬다.

본 논문에서는 네 가지 종류의 지연시간에 대한 수학적

* 正 會 員 : 明知大 工大 情報制御工學部 副教授 工博
接受日字 : 2001年 3月 22日
最終完了 : 2001年 4月 25日

모델링을 수행하고, 시뮬레이션을 통하여 적정량의 노드와 전송 바이트수, 가능한 전송속도를 알아본다. 메시지 스케줄링의 경우, 주기적 메시지에 대해서는 DMS 알고리즘을 적용하고, 비주기적인 메시지에 대해서는 개선된 DMS 알고리즘을 적용해 스케줄링하며, 포아슨(poisson)분포를 이용하여 모델링한다.

실제 실험을 위한 CAN 통신 노드를 구성하여, 노드간에 메시지를 전송하는데 걸리는 지연시간에 대해 실험을 통해 알아보며, 데이터 링크 계층에서 발생될 수 있는 지연시간에 대한 수학적 모델을 만들고, 그 모델을 기초로 하여 시뮬레이션을 수행하여 실제 CAN 프로토콜의 실시간성에 대한 성능을 비교, 분석한다. 그리고 비주기적 메시지 발생으로 인한 주기적 메시지의 실시간성 보장에 대해서도 시뮬레이션 결과를 통해 분석한다.

2장에서는 CAN 통신 지연시간에 대한 수학적 모델링에 대해 다루고, 비주기적 메시지의 수학적 모델을 만든다. 3장에서는 CAN 통신 모듈 설계 및 통신 실험을 서술하며, 4장에서는 모의 실험 및 결과분석, 그리고 마지막으로 결론에 대해 서술한다.

2. CAN 통신 프로토콜의 모델링

본 장에서는 CAN 프로토콜을 이용한 네트워크 시스템에서 발생할 수 있는 주기적, 비주기적 메시지의 지연시간과 응답시간을 수학적으로 모델링하고 메시지 제어루프를 정의한다. 주기적 메시지의 스케줄링은 고정된 우선순위 선점 방법을 쓰고, 프로세서의 스케줄링 방식과 메시지 스케줄링 방식은 DMS 알고리즘을 사용한다.

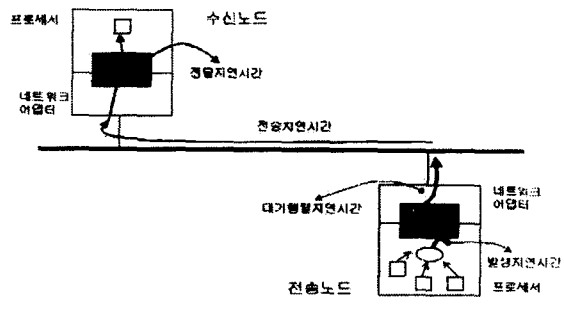

그림 1. CAN 프로토콜을 이용한 통신 구조

Fig 1. Communication architecture using CAN protocol

그러나 비주기적 메시지가 들어오는 경우, 비주기적 메시지에 대해 가장 높은 우선순위가 부여되는 동적 우선순위 할당을 사용한 메시지 스케줄링과 프로세서의 스케줄링을 수행한다. 전체 통신 지연시간은 발생 지연시간, 대기행렬 지연시간, 전송 지연시간과 전달 지연시간으로 이루어지며, 이에 대한 가정으로는 첫째, 낮은 우선권을 갖는 태스크에 대해서는 지터가 없으며, 둘째, 데이터가 주기적이고, 셋째, 모든 태스크는 같은 시점에서 발생된다는 것이다. 그림1은 발생 지연시간, 대기행렬 지연시간, 전송 지연시간, 전달 지연시간 등을 나타낸다.

2.1 발생 지연시간

노드(node)의 외부 장비에서 발생된 프로세스가 일련의 처리 과정을 거치고 네트워크를 통해서 전송할 수 있는 메시지로 변환되고 이를 메모리 큐(queue)까지 전달하는데 걸리는 시간을 발생 지연시간이라 정의하며 이는 식(1)과 같이 표현된다[3,7].

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

여기서, T_j는 태스크 j의 주기, r_i는 최악의 경우 태스크 i의 발생 지연시간을 나타내며, $hp(i)$는 태스크 i보다 높은 우선순위를 갖는 태스크의 집합을, 그리고 C_j와 C_i는 최악의 경우 태스크 i, j의 실행시간을 나타낸다.

동일한 노드의 서로 다른 장비에서 발생하는 프로세스에 대해 DMS 알고리즘을 적용하여 우선 순위가 결정되며, 식(1)의 둘째 항은 각 프로세스가 상위 프로세스로 인한 간섭시간(interference time)에 의해 지연되는 시간을 나타낸다.

2.2 대기행렬 지연시간

메모리 큐에 도달한 메시지가 물리적 전송 매체에 대한 접근 권한을 획득하기까지 걸리는 시간을 대기행렬 지연시간이라고 정의하며 이는 식(2)와 같이 표현된다[4,8].

$$t_m = B + \sum_{\forall j \in hp(m)} \left\lceil \frac{t_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{2}$$

$$B = \max_{\forall K \in lp(m)} (C_k) \tag{3}$$

네트워크를 구성하는 각 노드로부터 발생되는 메시지에 대해서 미리 정의된 우선 순위를 부여하며, 낮은 우선 순위의 노드가 물리적 전송 매체를 점유해서 높은 우선 순위의 노드가 대기하는 시간은 식(2)의 첫째 항인 B로 표현되고, 이를 블록킹 타임(blocking time)이라 정의한다. 이는 식(3)과 같이 표현된다. 여기서, t_m은 대기행렬 지연시간의 응답시간을 나타내고, 식(2)의 둘째 항은 실제 메모리 큐의 메시지의 지연시간을 나타내며, J_j는 프로세스 j의 지터(jitter)이다[2]. 또한, τ_{bit}는 물리적 전송 매체에서의 1비트를 보내는 시간이며, 식(3)의 $lp(m)$은 태스크 m보다 낮은 우선순위를 갖는 태스크의 집합을 나타낸다.

2.3 전송 지연시간

메모리 큐의 메시지가 물리적 전송 매체를 점유하고 목적 노드의 메모리 큐까지 도달하는데 걸리는 시간을 전송 지연시간이라고 정의하며 이는 식(4)와 같이 표현된다[4,8].

$$C_m = \left(\left\lfloor \frac{34 + 8s_m}{5} \right\rfloor + 47 + 8s_m \right) \tau_{bit} + \rho \tag{4}$$

여기서, s_m은 메시지의 바이트 크기를 나타내며, C_m은 버스상에서 메시지가 물리적으로 전송되는 시간을 나타낸

다. 그리고, ρ는 물리적 전송 매체의 전기적 특성을 고려한 지연시간이며 매체의 특성에 따라 정수로 표현된다.

식(4)의 첫째 항은 CAN 프로토콜 특성상 전송된 메시지의 오류를 검사하기 위한 방법으로 동일한 비트가 연속적으로 5비트가 이어지는 경우 반대되는 신호를 추가하며, 이를 스터프(stuff) 비트라고 정의한다. 그리고, 상수 47은 CAN 메시지의 실제 데이터를 제외한 부분에 대한 전체 오버헤드(overhead)이고, 34는 비트 스터핑(bit-stuffing)에 의해 발생되는 오버헤드이다.

2.4 전달 지연 시간

전달 지연 시간은 기존의 정의를 확장시켜서, 메시지 큐에 도착한 메시지가 노드의 프로세서에 의해서 일련의 처리 과정을 거쳐서 장비까지 전달되는 제어신호를 출력하는데 걸리는 시간을 전달 지연시간이라고 정의하며, 식(5)와 같이 표현된다.

$$D_H = C_H + I_H \tag{5}$$

여기서, D_H는 전달 지연시간이고, C_H는 직접 소프트웨어에 의해 태스크로 전달되는 시간을 말하며, I_H는 다른 태스크들에 의해서 간섭받는 시간을 나타낸다. 노드에 메시지가 도달하면 프로세서는 인터럽트(interrupt) 방식에 의해서 이를 처리한다. 식(5)의 첫째 항은 ISR(Interrupt Service Routine)에 의해서 처리되는 시간이며, 둘째 항은 노드에서 발생한 프로세스를 처리하기 위해서 지연되는 시간을 나타내며, 식(6)과 같이 표현된다.

$$I_H = \sum_{\forall j \in hp(H)} \left\lceil \frac{D_H + B_j}{T_j} \right\rceil C_j + \sum_{\forall m \in im(p)} P_m C_I \tag{6}$$

여기서, B_j는 프로세스 j가 임의적으로 실행이 지연되는 최대 시간이며, P_m은 메시지의 패킷(packet)의 수를 나타내고, C_I는 ISR의 실행 시간을 나타낸다. 식(6)의 첫째 항은 노드에서 발생하는 상위 우선 순위를 갖는 프로세스에 의해서 지연되는 시간이며, 둘째 항은 메시지의 패킷이 메모리 큐에 도달할 때마다 ISR의 실행을 고려한 항이다. 위에서 정의된 모든 식 (1), (2)와 (6)은 양변에 동일한 변수를 가지고 있음으로 이를 효과적으로 계산하기 위해서는 초기값을 0으로 설정하고 반복(iteration) 방법에 의해서 계산이 수행되어져야 한다[12].

2.5 메시지 제어루프(Message Control-Loop)

메시지 제어루프는 한 노드에서 발생한 메시지가 목적한 노드에 도착하여 원하는 태스크를 수행하고 그에 대한 응답 메시지가 메시지를 발생한 노드까지 도착했을 때까지를 말한다. 이 메시지 제어루프를 수학적으로 모델링하는 경우, 먼저 수식화된 발생 지연시간과 대기행렬 지연시간, 전송 지연시간, 전달 지연시간을 이용하여 수식을 만든다. 또한, 이 메시지 제어루프의 제한 조건은 첫째, 지터가 없으며 둘째, 모든 메시지는 같은 시점에 발생을 한다는 것이다. 그리고 각 노드에 대한 우선권은 발생지연시간에서 얻어진 우선권을 대기행렬지연시간과 전송지연시간에서는 계승을 받아서 사용하게 된다. 그리고 목적한 노드에 도착하여 원하는 태스크를 수행하는 모델은 큐잉 이론을 이용하며[9], 식(7)과 같다.

$$t_p = \frac{S}{C} \tag{7}$$

여기서, t_p는 수행된 시간이고, S는 태스크의 수행할 크기, 그리고 C는 태스크가 수행될 용량을 나타낸다. 그러나, 여기서는 각 노드별 프로세서를 1개로 가정하고, C는 상수 1로 정의하여 사용한다. 따라서, 메시지 제어루프를 수식화하면, 다음과 같다.

$$t_{mcl} = (r_i + C_m + t_m + D_H) \times 2 + t_p \tag{8}$$

식(8)의 첫째 항에 2를 곱한 이유는 메시지가 전송되었다가 다시 수신되는 루프를 형성하기 때문이며, 둘째 항은 전송 받은 노드에서 태스크를 실행하는 시간인 t_p를 추가시킨 것이다. 메시지 제어루프의 경우, 점대점 방식으로 모델링했기 때문에 2개 이상의 메시지에 대해서는 위의 수식이 이용될 수 없다.

2.6 비주기적 메시지(Aperiodic Message)

비주기적 메시지의 시간적 특성은 메세지의 발생간격이 주기적이지 않다는 것이다. 이러한 시간적 특성으로 인하여 비주기적 메시지의 처리는 실시간성의 구현에 있어서, 어려운 문제중에 하나이다. 따라서, 본 논문에서는 비주기적 메시지가 갖는 시간적 특성을 고려하여 확률적 접근 방법을 통하여 이를 처리하고자 한다[10]. 즉, 메시지 발생의 비주기적인 특성을 이와 유사한 시간적 특성의 사건에 적용되는 포아슨 분포를 이용하여 식(9), 식(10)과 같이 모델링한다.

$$f_x(x) = e^{-b} \sum_{k=0}^{\infty} \frac{b^k}{k!} \delta(x-k) \tag{9}$$

$$b = \lambda T \tag{10}$$

여기서, λ는 비주기적 메시지의 발생 비율을, k는 비주기적 메시지의 발생 횟수를, $f_x(x)$는 비주기적 메시지의 발생 확률을 나타낸다. 비주기적 메시지의 발생 확률 값을 계산하고자 하는 시간 구간 T는 주기적 메시지 주기의 최소 공배수로 결정을 하게 되며, 이에 대한 제한 조건은 첫째, 지터가 없으며 둘째, 주기적 메시지와 비주기적 메시지의 발생 시점은 동일하다는 것이다[11]. 그리고 비주기적 메시지의 스케줄링 알고리즘은 DMS 알고리즘을 사용하지 않고, 개선된 DMS 알고리즘을 사용한다. 개선된 DMS 알고리즘은 동적 우선순위 개념을 이용하여 발생된 비주기적 메시지는 다른 메시지들 보다 높은 최상위 우선 순위를 갖는다는 것이다. 비주기적 메시지가 발생할 경우, 메시시의 최악의 응답시간을 구하는 방법은 식(11)과 같다.

$$R_{ap} = (r_i + C_m + t_m + D_H) + t_{ap} \tag{11}$$

식(11)의 첫째 항은 메시지가 전송되는 시간이며, 둘째 항은 비주기적 메시지가 발생하여 실행되는 시간이다. 둘째 항을 직접 합하는 이유는 개선된 DMS 알고리즘을 이용하기 때문에 비주기적 메시지가 발생하면 최우선 순위가 되어 실행되기 때문이다.

3. CAN 통신 모듈 설계

CAN 통신을 구현하기 위해서는 크게 하드웨어 모듈과 소프트웨어 모듈이 필요하다. 하드웨어 모듈은 독립 CAN보드(independent CAN board)와 CAN ISA 인터페이스 보드(또는 CAN PCI 인터페이스 보드)로, 소프트웨어 모듈은 통신 프로토콜 처리를 위한 펌웨어와 통신 상태를 확인 할 수 있는 CAN 네트워크 모니터링 프로그램 등으로 구성된다.

본 논문에서는 상용화된 모듈을 사용하지 않고 CAN 통신 모듈을 직접 설계하고, 제작하여 실험에 사용하였다. 그 이유는 수학적으로 모델링된 지연시간의 모의실험 결과와 실제 구현한 CAN 통신 모듈간 실험결과를 좀더 상세하게 비교해 보기 위해서이다. 그림2는 제작된 독립 CAN 보드와 CAN ISA 인터페이스를 보여준다.

그림 2. 설계된 실험용 CAN 보드
Fig 2. CAN boards designed for experiments

3.1 CAN 통신을 위한 하드웨어 설계

CAN 통신을 위한 하드웨어는 센서의 입력을 받아 스마트센서의 기능을 구현하기 위한 독립 CAN 보드와 개방형 제어기의 CAN 통신 기능 구현을 위한 ISA 슬롯에 장착되는 PC 인터페이스 보드의 두 가지 종류로 구성된다.

3.1.1 독립 CAN 보드와 CAN ISA 인터페이스 보드

독립 CAN 보드를 구성하기 위해서는 CPU, CAN 제어기, ROM, RAM 그리고 CAN 전송기를 필요로 한다. 다음 그림3은 독립 CAN 보드의 내부구조를 보여준다.

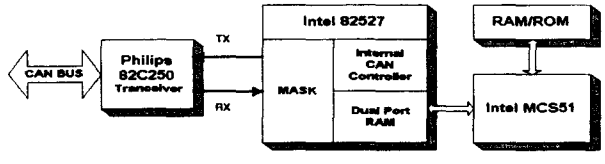

그림 3. CAN 통신 노드 내부 구조
Figure 3. The internal architecture of CAN communication nodes

그림3에서 제일 핵심이 되는 부분이 CAN 제어기이다. 이것은 물리 계층과 응용 계층사이의 중계 역할을 하며, 데이터가 중간에 충돌이 일어나지 않도록 하게 하는데, 이 기능은 82527에 내장된 쌍방향성 램(Dual Port RAM)이 있기 때문이다.

또한 CAN 제어기는 필터링(filtering)하는 기능이 있는데, 이 역할을 담당하는 부분이 쌍방향성 램 앞의 마스크(mask)라는 부분이다. 이 마스크 부분은 CAN 식별자(identifier)를 판단하여 자신이 갖고 있는 식별자와 동일한 데이터만을 취하도록 되어 있다.

CAN ISA 인터페이스 보드는 개방형 제어기의 CAN 통신 기능 구현을 위해 PC의 ISA 슬롯에 장착되는 보드로 CAN의 물리계층과 PC의 응용 프로그램을 중계하는 역할을 한다.

3.2 CAN 통신을 위한 응용계층 설계

표1은 DeviceNet의 CAN 식별자 영역을 나타내고 있는데, 그 요소는 다음과 같다. 표1은 상용으로 개발된 DeviceNet의 응용계층의 개념이며, 본 논문에서의 식별자에 대한 할당은 DeviceNet 식별자 할당과 동일한 방식을 이용한다.

표 1 . DeviceNet의 CAN 식별자 영역

Table 1. DeviceNet's use of the CAN identifier field

IDENTIFIER BITS											HEX	IDENTITY
10	9	8	7	6	5	4	3	2	1	0	RANGE	USAGE
0	Group 1 Message ID				Source MAC ID						000-3ff	Message Group 1
1	0	MAC ID						Group 2 Message ID			400-5ff	Message Group 2
1	1	Group 3 Message ID			Source MAC ID						600-7bf	Message Group 3
1	1	1	1	1	Group 4 Message ID (0-2f)						7c0-7ef	Message Group 4
1	1	1	1	1	1	1	x	x	x	x	7f0-7ff	Invalid CAN Identifier

DeviceNet 식별자 할당은 CAN의 식별자 영역을 사용하는데 이는 11개의 CAN 식별자 비트로 된 4개의 메시지 그룹으로 이루어진다. 메시지 그룹1은 보통 전송을 하며, 네트워크상의 버스 억세스의 가장 높은 우선권을 갖고 있다. 메시지 그룹2는 전송과 수신을 할당한다. 메시지 그룹3은 전송에 사용하며, 메시지 그룹4는 특별한 메시지를 보낼 때 사용한다. 본 논문에서는, 메시지 그룹1과 메시지 그룹2를 사용하여 구현한다. 다음 그림 4는 식별자의 구성도이다.

노드 안에 있는 이진수는 각각의 정의된 식별자들을 나타낸다. 여기서, 독립 CAN 보드는 메시지 그룹1을 사용하고, CAN 프로토콜 분석기는 그룹2를 사용하여 구현한다. 그리고, 독립 CAN 보드#1이 가장 높은 우선권을 갖는다. 그렇기 때문에 긴급상황들이 발생할 경우, 독립 CAN 보드#1에서 메시지를 처리하는 것이 가장 실시간성이 좋게 된다.

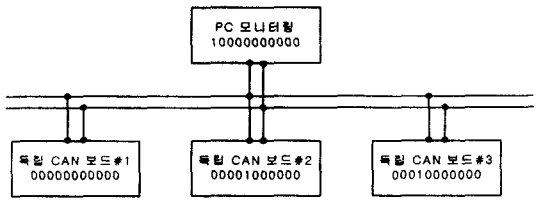

그림 4 식별자의 구성도

Fig 4. Configuration of identifier

메시지의 필터링, 전송 및 전달을 담당하는 메시지 오브젝트(Intel사 82527의 내부 레지스터의 이름)의 구성은 다음과 같다. 독립 CAN 보드의 메시지 오브젝트의 경우는 CAN 네트워크를 모니터링하는 노드로만 전송을 하고 그 노드로부터 데이터를 전달받기 때문에, 전송과 수신을 위한 2개의 메시지 오브젝트가 할당되고, CAN 네트워크를 모니터링하는 노드의 메시지 오브젝트의 경우는 독립 CAN 보드 3개의 노드에 각각 전송을 하고 데이터를 수신하기 위한 메시지 오브젝트를 포함해서 모두 4개의 메시지 오브젝트가 할당된다.

4. 모의실험 및 결과분석

본 장에서는 모의 실험을 통해 얻은 값을 실제 실험한 값과 비교하여 수학적 모델링의 유효성을 확인한 후 CAN 프로토콜의 성능분석을 수행하고자 한다. 또, 주기적 메시지의 전송 노드수와 전송속도, 전송바이트에 따라 응답시간의 변화와 실시간성의 보장을 확인하고, 비주기적 메시지가 발생했을 경우에 대한 주기적 메시지의 실시간성 보장을 살펴보고자 한다.

4.1 실제 실험과 모의실험 결과

먼저 실제로 수행한 하드웨어적 실험결과와 비교하기 위하여 모의실험에서는 하나의 노드에서 다른 노드로 전송되는데 걸리는 지연시간에 대해 2장에서 언급한 수학적 모델링을 이용하여 최악의 응답시간을 구한다.

다음 그림5는 10개의 노드를 갖는 모의실험을 위한 네트워크 구조를 나타내며 CAN의 일반적인 특성상 노드간의 거리는 최대 40m내의 거리에서 구성이 된다고 가정하고 전송지연시간에 영향을 주는 물리적 전송 시간은 1Mbit/sec를 기준으로 해서 한 비트를 전송하는데 걸리는 시간은 1μs이다.

또한, 표2는 노드 10개의 경우의 모의실험 결과이며 기본 단위는 μs이다. 위의 결과는 전달할 데이터의 크기를 1바이트로 고정시킨 후 노드수와 전송속도를 바꾸어주면서 결과를 얻은 것이다.

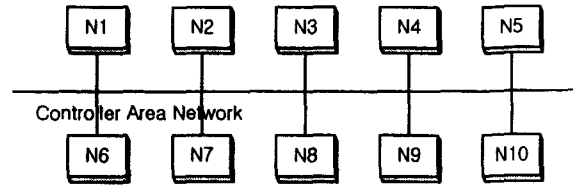

그림 5. 10개의 노드로 구성된 CAN 통신 네트워크

Fig 5. CAN communication network architecture with 10 nodes

표2의 결과를 보면 전송속도가 느려지면 그에 따른 응답시간이 느려짐을 알 수 있다. 먼저 우선순위가 가장 높은 8번 노드를 보면 1Mbit/sec일 때와 125Kbit/sec일 때의 응답시간은 약 5배의 차이가 나는 것을 볼 수 있는데, 그 이유는 물리적 전송 매체를 통해 메시지를 전송할 때 전송속도가 빠를수록 한 비트를 전송하는데 걸리는 시간이 짧기 때문에 지연시간 요소 중에 전송지연시간에서의 차이 때문이다. 또한, 우선순위에 따라 응답시간이 증가하고 있는데 이는 같은 시점에서 메시지를 전송하기를 원하지만 결정된 우선순위에 따라 메모리 큐에서 처리되기를 기다리고 있기 때문에 지연되는, 즉 대기행렬 지연시간의 증가, 현상으로 인해 응답시간이 계속적인 증가를 나타내고 있다.

특히, 모의 실험을 통하여 하나의 노드에서 다른 노드로 보내는 응답시간은 표2의 최상위 우선권을 갖는 메시지가 다른 노드로 전송하는 경우와 같기 때문에 표2의 최상위 우선순위 즉, 우선순위 1을 가지는 노드의 1Mbit/sec 속도에서의 응답시간과 같은 결과인 92.3μs이다.

실제 실험은 설계된 노드를 이용하여 수행했으며, 각각을 CAN 노드로 설정하여 점대점 방식으로 하나의 노드에서 다른 노드로 전송하는 시간을 측정하였다. 모의실험과 동일하게 전송바이트는 1바이트로 고정하고, 전송속도는 1Mbit/sec로 하여 실험하였다. 1000회 이상 실험한 결과 하나의 노드에서 다른 노드로 전송되는 시간은 평균 81μs의

표 2 노드 10개의 경우 모의실험 결과

Table 2. The simulation result when the number of nodes are 10

Node	Size (byte)	Priority	T(period) (μs)	D(Deadline) (μs)	Data Rate 125Kbit/sec(μs)	Data Rate 250Kbit/sec(μs)	Data Rate 500Kbit/sec(μs)	Data Rate 1Mbit/sec(μs)
1	1	6	100000	99990	3054.8	1542.8	786.8	408.8
2	1	4	100000	99000	2046.2	1038.2	534.2	282.2
3	1	10	1000000	1000000	5072.0	2552.0	1292.0	662.0
4	1	5	100000	99900	2550.5	1290.5	660.5	345.5
5	1	9	1000000	999999	4567.7	2299.7	1165.7	598.7
6	1	8	100000	99999	4063.4	2047.4	1039.4	535.4
7	1	3	5000	4999	1541.9	785.9	407.9	218.9
8	1	1	5000	4990	533.3	281.3	155.3	92.3
9	1	2	5000	4993	1037.6	533.6	281.6	155.6
10	1	7	100000	99990	3559.1	1795.1	913.1	472.1

결과를 얻었다. 표3에서 모의실험과 실제 실험 결과를 비교하였다.

표 3. 모의실험과 실제 통신실험과의 결과 비교

Table 3. Comparison of results between the simulations and the practical experiments

Simulation result	Real practical experiment
92.3 μ s	81 μ s

통신 모듈간 실험결과와 모의실험의 결과가 비슷하게 나오는 것을 볼 수가 있었고, 약간의 차이는 모의실험에서는 모두 최악의 경우에 대한 모델링을 했고, 실험의 경우는 정상적인 상태에서의 응답속도를 구했기 때문이다.

4.2 주기적 메시지의 모의실험

다음 그림6의 결과는 노드수 40개일 경우 전송속도 변화에 따른 전송 실패율을 나타낸 것이다. 여기서, 세로 좌표축인 MD(Missed Deadline)는 식(12)와 같이 정의한다. 발생하는 메시지들의 데드라인은 표2에 나타나 있고, DMS 알고리즘에 의해 우선순위를 할당하여 스케줄링 하였다.

$$MD=\frac{\text{데드라인을넘어가는최악의응답시간}}{\text{각전송바이트에대한최악의응답시간}} \quad (12)$$

그림6에서 보면, 전송바이트가 많아지고, 전송속도가 느려지면 실시간성 보장이 나빠짐을 알 수 있는데 그 이유는 메시지 크기의 증가와 전송속도의 느림으로 인해 대기행렬 지연시간과 전송 지연시간에 영향을 주어 그에 따른 응답시간이 길어지기 때문이다. 가장 느린 전송속도인 125Kbit/sec일 경우에는 데드라인을 지키는 경우가 없음을 알 수 있다.

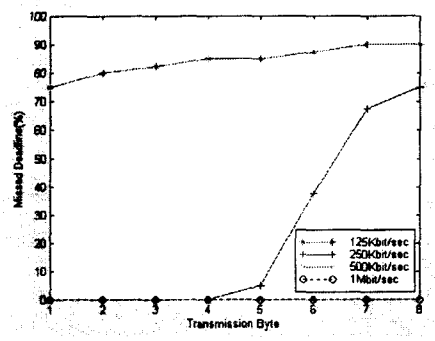

그림 6 노드 40개일 경우의 메시지 전송 실패율

Fig 6. Percentage of transmission failure when the number of nodes are 40.

그림7은 전송속도 125Kbit/sec인 경우 노드수의 변화에 따른 전송실패율을 보여준다. 다음의 결과에서 보듯이 전송노드가 느리면 느릴수록 실시간성이 나빠지는 것을 볼 수 있다. 결과적으로 전송속도가 느릴수록, 전송바이트수가 증가할수록 그리고 전송노드가 많아질수록 실시간성이 보장 안됨을 알 수 있다.

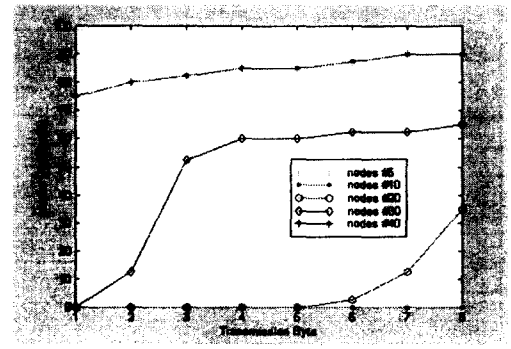

그림 7. 125Kbit/sec인 경우의 전송 실패율

Fig 7. Percentage of transmission failure when the transmission speed is 125Kbit/sec.

4.3 비주기적 메시지의 모의실험

주기적 메시지와 비주기적 메시지가 동시에 발생하는 경우 실시간성의 보장이 어려워진다. 비주기적 메시지는 주기의 구간이 없고 발생하는 시점도 예상할 수 없기 때문이다. 이러한 문제를 분석하기 위하여 비주기적 메시지와 유사한 특성을 갖는 확률분포인 포아슨 분포를 이용하여 모델링하고, 실시간 통신이 가능한 CAN 프로토콜을 이용한 네트워크상에 적용시킴으로써, 기존의 실시간성이 보장된 주기적 메시지들이 비주기적 메시지의 발생으로 인해 데드라인 내에 처리될 수 있는지의 여부를 모의실험을 통하여 분석하고자 한다. 표4는 비주기적 메시지 모의실험을 위해 가정한 주기적 메시지의 구성을 나타낸다. 모두 주기적 메시지이며 데드라인에 의해서 우선순위가 결정되며, 전송속도는 1Mbit/sec로 가정한다.

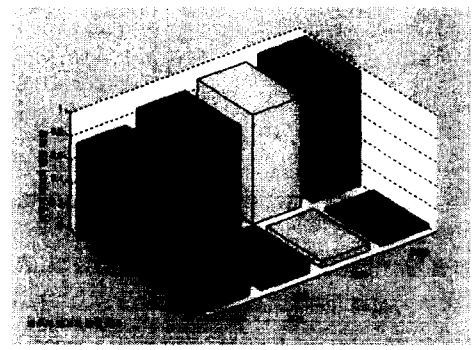

그림 8. 비주기적 메시지가 1번 발생할 경우의 확률 분포

Fig 8. Random distribution when one aperiodic message is generated.

그림 8은 비주기적 메시지가 시간 구간에서 1번 발생 할 확률 분포를 나타낸 것이다. 그림과 같은 확률 분포를 갖는 비주기적 메시지는 시간이 증가함에 따라 발생하는 확률이 작아짐을 알 수 있다.

표 4. 가정한 주기적 메시지의 구성
Table 4. Configuration of periodic messages assumed

Node	Size(byte)	Period(μs)	Deadline(μs)
1	1	800	700
2	2	3200	3000
3	1	6400	6000
4	3	1600	1000
5	3	3200	2900
6	1	800	750
7	6	1600	1200
8	4	6400	5900
9	8	1600	1300
10	2	800	790
11	3	3200	3100
12	5	6400	6100
13	7	1600	1450
14	8	6400	6400
15	7	3200	3200
16	5	1600	1600
17	1	6400	5500
18	3	800	690
19	4	1600	1400
20	4	3200	2700
21	3	800	800
22	3	6400	5000
23	5	1600	1500
24	4	3200	2500
25	3	800	710
26	4	3200	2100
27	7	6400	900
28	3	800	500
29	5	3200	2670
30	4	1600	1100

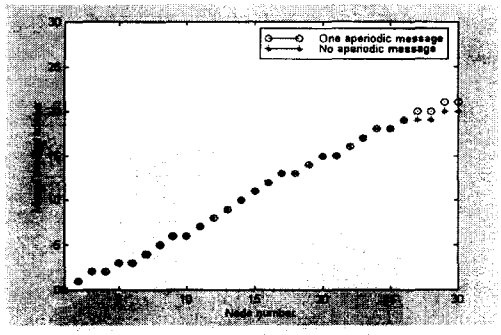

그림 9. 비주기적 메시지가 1개일 경우의 전송 실패 개수

Fig 9. Number of transmission failure when the number of aperiodic message is one.

그림9는 표4의 주기적 메시지가 데드라인을 넘는 모의실험 결과와 하나의 비주기적 메시지에 의해 데드라인을 넘는 경우에 대하여 나타낸 그림이다. 비주기적 메시지가 발생할 경우, 최악의 경우 응답시간을 계산하기 때문에, 주기적인 메시지와 비주기적인 메시지가 모두 같은 시점에서 발생되고, 비주기적 메시지는 가장 높은 우선권을 갖는다는 가정을 한다. 그리고, 이 모의실험에서의 비주기적 메시지의 크기는 8바이트이며 실행시간은 0.2ms로 가정한다.

그림9의 결과에서 나타난 것처럼 비주기적 메시지가 하나일 경우 노드수, 즉 메시지 수를 늘려가며 실험한 결과 26개의 노드까지는 주기적인 메시지에 대해 비주기적인 메시지의 발생이 데드라인을 넘기는 경우에 대해 영향을 주지 않지만 노드수가 늘어나면 영향을 미치는 걸 볼 수 있다. 노드의 수를 30개까지 하여 구한 이유는, 노드수가 작은 경우에는 발생된 비주기적 메시지 처리에 대한 스케줄링이 가능하여 제한시간을 넘지 않기 때문이다. 노드가 많을수록 전송할 크기가 클수록 전송속도가 느릴수록 제한시간을 넘는 것은 앞의 모의실험에서 확인했다.

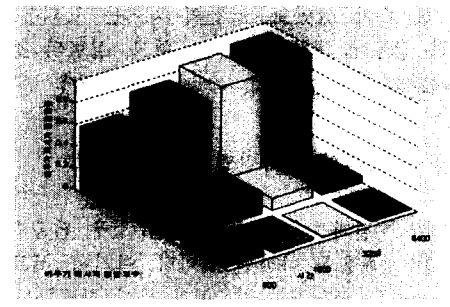

그림 10. 비주기적 메시지가 2번 발생할 경우의 확률분포

Fig 10. Random distribution when two aperiodic messages are generated.

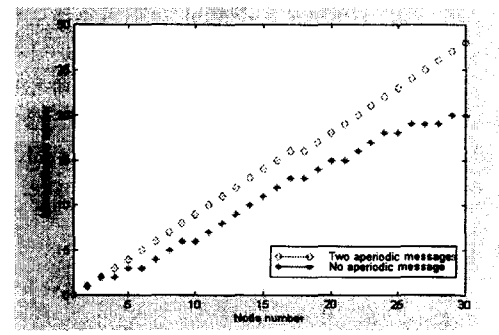

그림 11. 비주기적 메시지가 2개일 경우의 전송 실패 개수

Fig 11. Number of transmission failure when the number of aperiodic messages are two.

그림10은 비주기적 메시지가 2개 발생할 때의 확률 분포이다. 그림10에서는 시간이 갈수록 발생할 횟수가 커질수록 비주기적 메시지의 발생확률이 작아짐을 볼 수 있고, 그림11은 비주기적 메시지가 2번 발생할 경우의 메시지 전송 실패 개수이다. 예상했던 바와 같이 비주기적 메시지가 늘어날수록 실시간성의 보장이 나빠짐을 볼 수 있다. 노드수가 4개 이상부터 데드라인을 넘기는 경우가 주기적인 메시지만 있을 때 보다 증가함을 볼 수 있다. 결과적으로 비주기적 메시지가 증가함에 따라 노드 수 증가에 따른 주기적 메시지의 실시간 보장에 대한 영향을 나타내고 있다.

5. 결 론

본 논문에서는 CAN 프로토콜을 이용한 네트워크 시스템의 성능에 영향을 줄 수 있는 여러 요소들, 즉 노드 수, 전송바이트, 전송속도 등의 변화를 고려해 봄으로써, 시스템의 성능과 실시간성 보장에 대한 분석을 수행하였다. CAN 통신상의 발생되는 지연시간에 대해 수학적으로 모델링하여 메시지의 최악의 경우에 대한 응답시간을 결정하고, 모의실험을 통해 성능 분석을 실시하였다. 그리고, 실제 CAN 통신 시스템을 구현하여 CAN 노드간에 발생되는 지연시간과 응답시간에 대해 모의실험 결과를 통해 비교함으로써 모델링의 유효성을 입증하였다.

본 논문에서 가정한 CAN 프로토콜 통신 모델을 이용해 주기적인 메시지에 대해 여러 요소들의 변화를 고려한 네트워크 시스템의 성능을 모의실험을 통해 알아보았다. 또한, 비주기적 메시지 발생으로 인한 주기적 메시지의 실시간성 보장에 대한 분석을 수행하였다. 모의실험의 분석 결과를 통해 알 수 있듯이 CAN을 이용한 시스템을 구성할 때 전송속도가 500Kbit/sec이상이라고 가정한다면 최대 40개의 노드까지 추가 가능한 사실로 인해 시스템 기능 향상에 기여를 할 수 있다. 또한, 제어루프를 구성하는 시간 요소 중에서 응용 계층에서의 지연 시간을 줄임으로 인해 빠른 응답 특성을 보였다. 본 논문에서의 시뮬레이션 분석 결과는 통신 프로토콜이 정해진 네트워크 기반 시스템 설계 시 고려되어야 할 메시지의 크기나 노드의 확장, 전송속도의 결정, 그리고 비주기적 메시지 처리 등에 필요한 기초자료를 제공할 수 있기 때문에 반드시 필요한 과정으로 생각된다.

향후 과제로는 수학적 모델링의 가정조건을 줄이는 것이며, 실제 제어 대상을 선정해서 적용해 봄으로써, CAN 프로토콜을 이용한 네트워크 시스템의 효율성과 유효성을 확인하는 것이다.

참 고 문 헌

[1] BOSCH, CAN Specification, Part A, 1991.

[2] Ken Tindell, "Analysis of Hard Real-Time Communications", YCS222, Department of Computer Science, University of York, Jan. 1994.

[3] K.W.Tindell, A.Burns, A.J.Wellings, "Guaranteeing Hard Real Time End-to-End Communications Deadline", Dec 1991.

[4] Ken Tindell, Alan Burns, "Guaranteeing Message Latencies On Controller Area Network(CAN), Preceeding 1st International CAN conference, Mainz, Germany, Sep 1994.

[5] J.P. Lehoczky, L. Sha and Y. Ding. "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", In Proceedings of IEEE Real-Time Systems Symposium, pages 166-171, Dec. 1989.

[6] N. Audsley, "Deadline Monotonic Scheduling", Sep. 1991.

[7] K.W. Tindell, "An Extendible Approach for Analysing Fixed Prority Hard Real-Time Task", Real-Time Systems 6(2), pp.133-151, 1992.

[8] K.Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network(CAN) Message Response Times" Proceedings 1994 IFAC workshop on Distributed Computer Control Systems(DCCS), Toledo, Spain, Sep 1994.

[9] Modeling and Analysis, ADDISON-WESLEY pp.95-100, 1978.

[10] P. Z.Peebles, Jr. Probability Random Variables and Random Signal Principles, McGRAW-HILL, 1993, USA.

[11] Robert Davis and Alan Burns, "Optimal Priority Assignment for Aperiodic Tasks with Firm Deadlines in Fixed Priority Pre-emptive Systems", Real-Time Systems Research Group, Department of Computer Science, University of York.

[12] N. Audsley, A. Burns, M. Richardson, K. Tindell, A. J. Wellings, "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling", Department of Computer Science, University of York.

저 자 소 개

김 대 원 (金 大 元)

1960년 2월 15일 생. 1984년 서울대 제어계측공학과 졸업. 1986년 동대학원 졸업(석사). 1990년 동대학원 졸업(공박). 1987년~1992년 대우중공업(주) 중앙연구소 선임연구원. 1992년~현재 명지대 정보제어공학부 부교수. 관심분야: 자동화 네트워크, 실시간 시스템, 웹기반 응용

Tel : 031-330-6472, Fax : 031-330-6226

E-mail : dwkim@mju.ac.kr

최 환 수 (崔 煥 洙)

1961년 12월 25일 생. 1984년 서울대 제어계측공학과 졸업. 1986년 Univ. of Washington 전기공학과 졸업(석사). 1990년 Univ. of Washington 전기공학과 졸업(공박). 1990~1992년 (주)금성사 컴퓨터연구소 선임연구원. 1992년~현재 명지대 공과대학 정보제어공학부 부교수. 관심분야: 영상처리 및 컴퓨터비젼, 신호처리

Tel : 031-330-6363, Fax : 031-330-0271

E-mail : hschoi@mju.ac.kr