CAN 메시지의 Stuffing-bit 최소화

강민구 °*, 박기진*, 신동민**

*아주대학교 공과대학 산업정보시스템공학부
**한양대학교 공학대학 기계정보경영공학부
e-mail : {ozige°, kiejin}@ajou.ac.kr, dmshin@hanyang.ac.kr

Minimizing Stuffing-bit of CAN Message

Minkoo Kang°*, Kiejin Park*, Dongmin Shin**

*Division of Industrial & Information Systems Engineering, Ajou University
**Division of Mechanical and Information Management Engineering, Hanyang University

요 약

CAN (Controller Area Network) 프로토콜은 데이터 링크 계층(Data Link Layer) 에서 여러 가지 결함 허용(Fault-tolerant) 메커니즘을 기본적으로 제공하기 때문에, 고신뢰성, 고안전성이 요구되는 임베디드 제어 시스템(예: 자동차, 산업제어 등) 에서 많이 사용되고 있다. 하지만 제한된 네트워크 대역폭과 이에 따른 응답시간 지연이라는 문제점을 가지고 있으며, 이에 본 논문에서는 전송될 CAN 메시지의 길이를 감소시키기 위해 CAN 메시지의 Stuffing-bit를 최소화하는 메커니즘을 제안하였다. 실험을 통해, CAN 메시지 길이의 축소로 인해 CAN 버스의 부하(Load)를 덜게 되었으며, 네트워크 대역폭 이용률이 증가하는 것을 확인하였다.

1. Introduction

최근 각광을 받고 있는 Controller Area Network (CAN)은 차량 제어 시스템을 포함하여 많은 산업용 임베디드 제어 시스템에서 사용되는 실시간 네트워크의 일종이며, 최고 1Mbit/s의 높은 대역폭(Bandwidth)을 제공하고, 짧은 데이터 길이에 따른 낮은 지연시간(Latency)을 특징으로 한다. 게다가 CAN 컨트롤러 내부에 자체적으로 CAN 메시지의 오류를 감지하는 여러 메커니즘을 가지고 있으며, CAN 네트워크를 구성하는 비용에 비해 높은 수준의 신인도(Dependability)를 보장하는 장점이 있다. CAN 컨트롤러에 내장된 주요한 오류 감지 메커니즘은 비트 오류(Bits Errors), 비트 채움 오류(Bits Stuffing Errors) 및 CRC 오류(Cyclic Redundancy Check Errors) 등이며, CAN 컨트롤러는 감지된 오류 횟수를 기록함으로써 자신의 상태(Error Active State, Error Passive State, Bus-Off State)를 전환 및 관리할 수 있다[1,2].

CAN 메시지 양식은 식별자(Identifier)의 크기가 11비트인 표준형과 29비트인 확장형으로 구분되며, CAN 표준형의 메시지 양식은 그림 1과 같다. 표준형 CAN 메시지의 길이는 전송하고자 하는 데이터의 바이트 수에 대해 가변적이며, 8바이트의 데이터를 전송하는 경우 CAN 프로토콜의 최대 네트워크 대역폭 이용률은 약 58%이다[1]. 하지만 CAN 버스가 메시지 전송만을 위해 100% 가동되지 않으며, 전송하고자 하는 데이터의 길이 또한 0~8바이트로 가변적이므로 실제로는 58%보다 더욱 낮은 네트워크 대역폭 이용률을 보이고 있다.

CAN 컨트롤러에 내장된 여러 오류 감지 메커니즘 중에서 비트 채움(Bit Stuffing) 메커니즘은 같은 극성의 비트가 연속적으로 6개 이상 발생하는 것을 막기 위해, 같은 극성의 비트가 5개 연속으로 발생하게 될 경우 다른 극성의 비트 1개를 삽입하는 방법을 사용하며, 수신 노드에서는 삽입된 비트를 제거하여 CAN 메시지를 처리한다. [3]에서는 비트 채움으로 인해 CAN 메시지의 길이가 최악의 경우 약 22%(111→135bits) 정도 길어지게 되므로, 비트 채움을 하기 이전에 " 101010…" 과 XOR 연산을 수행한 뒤 비트 채움을 하는 기법을 통해 CAN 메시지의 길이를 줄이고자 하였다. 하지만, XOR 연산을 포함하는 비트 채움 메커니즘의 경우에는 우선순위 역전이 발생하게 되므로 [3]에서 제안하는 방법을 CAN 메시지의 식별자 비트 구간에 적용하는 것은 불가능하다.

본 연구는 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행되었음. (KRF-2006-331-D00438)

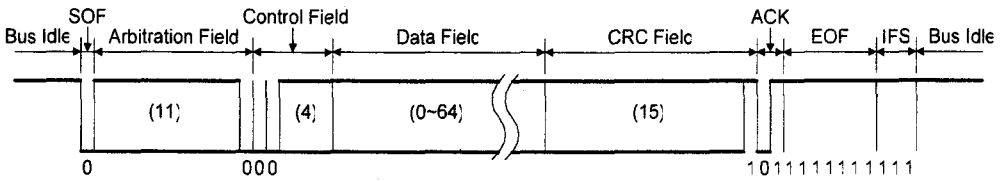

그림 1. 표준형 CAN 메시지 양식

CAN 메시지는 비트 채움 메커니즘에 의해 전송되는 데이터가 더욱 늘어나게 되어, 네트워크 대역폭 이용률이 더욱 감소되고, 응답 시간이 증가하는 단점을 갖고 있다[4,5]. CAN 프로토콜의 단점으로 지적되는 낮은 네트워크 대역폭 이용률을 개선하고, 실시간 임베디드 제어 시스템 응용에서 요구되는 응답 시간 성능 향상을 위해, 전송되는 CAN 메시지의 길이 자체를 줄이는 노력은 의미가 있다고 판단하여, 본 논문에서는 CAN 메시지의 Stuffing-bit를 최소화하는 메커니즘을 제안한다.

2. Advanced Bit Stuffing (ABS) Mechanism

본 논문에서는 우선순위 역전을 막을 수 있는 XOR 연산을 포함한 비트 채움 메커니즘을 제안한다. 현실적으로 CAN 기반 제어 시스템에서는 메시지 ID로 11비트를 모두 필요로 하고 있지 않는 점에 착안하여, CAN 메시지 처리의 중요도를 기준으로 CAN 메시지 ID를 그룹화하여 사용하고자 한다. 예를 들어 총 128개의 메시지 ID가 필요한 경우, CAN 메시지의 중요도에 따라 4개의 그룹으로 나누고, 각 그룹별로 32개의 ID를 사용하는 시스템을 가정하면 그룹을 구분하는 2비트와 각 그룹내의 CAN ID를 표현하는 5개의 비트만 사용하기 때문에, 표준형 CAN의 경우 4개의 비트는 고정적으로 ' 0' 의 값을 가질 수 있다.

일반적으로 표준형 CAN 프로토콜을 사용하는 시스템에서 m 개의 그룹과 한 그룹에서 가질 수 있는 최대 ID 수를 n 개라고 할 때, 고정적으로 ' 0' 의 값을 갖는 비트의 수, k_{std} 는 다음의 수식 (1)과 같다. 그룹을 표현하는 비트는 중재 필드(Arbitration Field)의 MSB(Most Significant Bit) 부분에 할당되고, 각 그룹의 ID를 표현하는 비트는 중재 필드의 LSB(Least Significant Bit) 부분에 할당된다.

$$k_{std} = 11 - \lceil \log_2 m \rceil + \lceil \log_2 n \rceil$$
$$(\text{확장형의 경우, } k_{etd} = k_{std} + 18) \qquad (1)$$

2.1. Mask Generation

CAN 메시지의 우선순위 역전 현상을 피함과 동시에 비트 채움의 수를 줄이기 위한 XOR Mask는 다음에서 설명하는 방법에 의해서 생성된다.

1) 우선 Mask의 전체 크기는 8s+34 (s는 데이터의 바이트 수) 비트이며, " 101010…" 패턴을 이용한다.
2) XOR mask의 상위 12 비트를 대상으로 MSB 부분의 $1+\lceil \log_2 m \rceil$개 비트와 LSB 부분의 $\lceil \log_2 m \rceil$개 비트는 ' 0' 으로 고정한다. 이는 본래 CAN 메시지의 우선순위를 그대로 보호한다는 의미이다 (그림 2 참조).
3) CAN 메시지 그룹의 중요도에 따른 그룹 표시 비트는 우선순위가 가장 높은 그룹을 " 00…" 으로 하고, 우선순위가 가장 낮은 그룹은 " 11…" 순으로 결정한다.

이러한 과정을 통해 CAN 메시지의 우선순위 역전 현상을 피할 수 있고, CAN 메시지의 중재 필드에서 고정적으로 ' 0' 의 값을 갖는 구간에서 발생할 수 있는 Stuffing-bit의 수를 줄일 수 있다.

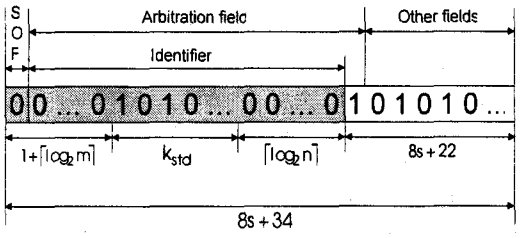

그림 2. ABS 메커니즘을 따르는 XOR Mask

CAN 프로토콜을 사용하는 시스템의 메시지 그룹의 수(m)와 한 그룹에서 가질 수 있는 최대 ID의 수(n)가 결정되면 그림 2와 같은 XOR Mask (음영으로 표시된 부분)를 생성할 수 있다. 새로운 Mask를 이용하게 되면 ID 비트 구간의 연속적인 ' 0' 비트에 대해 " 101010…" 패턴의 비트로 Masking 함에 따라 비트 채움의 수를 줄일 수 있으며, 우선순위를 결정하는 비트 구간을 보호할 수 있게 되어 우선순위 역전 현상을 피할 수 있다.

3. Performance Evaluation

본 논문에서 제안하는 비트 채움 메커니즘의 성능 평가에 사용된 가정은 다음과 같다. 1) CAN 메시지에서 특정 비트 값을 고정적으로 가지지 않는 부분에 대해 0과 1의 비트 값을 가질 확률은 동일하며, 다른 비트의 값에 대해 독립이다. 2) CAN 메시지의 데이터 필드와 CRC 필드의 비트 값들은 특정 비트 값을 고정적으로 가지지 않기 때문에 기존의 기법[3]과 제안하는 기법에 대한 성능의 차이가 없으므로, 성능 평가의 대상에서 제외한다. 그러므로 성능 평가의 대상은 CAN 메시지의 상위 부분인 SOF, 중재 필드 및 제어 필드(Control Field)이다.

3.1. Case Study: 4-group

CAN 메시지 식별자의 그룹 수(m)와 한 그룹에서 가질 수 있는 최대 식별자의 수(n)를 기준으로 XOR Mask를 생성하고, Original 메시지와 XOR Masking 메시지의 Stuffing-bit의 수를 비교하는 방법을 사용하여 제안하는 메커니즘의 성능을 평가하였다. 예를 들어 $\lceil \log_2 m \rceil$=2, $\lceil \log_2 n \rceil$=5 인 경우, CAN 메시지의 중재 필드에서 고정적으로 0의 값을 가지는 구간은 4비트(=11-2-5)이며, 이때의 XOR Mask는 0x0A055 이다. XOR Masking 메시지는 Original 메시지의 우선순위를 그대로 보호하며(그림 3의 a, b 참조), 고정적으로 ' 0' 의 값을 가지는 구간이 " 101010…" 으로 바뀐 비트 값을 가진다. 제어 필드의 DLC(Data Length Code, 그림 3의 c) 값은 " 0001" 에서 " 1000" 까지 8 종류의 값을 가지며, XOR Mask의 대상이 된다(그림 3 참조).

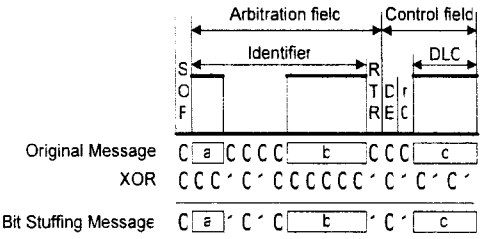

그림 3. Bit Stuffing 메시지

그림 3의 전체 19 비트 CAN 메시지에서는 SOF를 포함한 상위 7비트에서 Stuffing-bit 가 발생하지 않으며, 1~8 바이트 데이터에 따르는 모든 DLC값에 대해 하위 7 비트에서도 Stuffing-bit 가 발생하지 않는다. 나머지 5 비트(그림 3의 b 참조)에 대해 Stuffing-bit 1개가 발생하는 경우는 전체 32개 경우의 수 중에서 4가지(00000, 00001, 01111, 11111)뿐이며, 그 외의 경우에는 Stuffing-bit 가 발생하지 않는다.

3.2. Probabilistic Analysis

성능 평가의 대상이 되는 EOF, 중재 필드 및 제어 필드 구간에 대해 $\lceil \log_2 m \rceil$=2, $\lceil \log_2 n \rceil$=1,2,…,9 인 경우, 발생 가능한 모든 XOR Masking 메시지에 대해 Stuffing-bit 의 발생 확률($P_m(X)$) 및 기대값($E_m(X)$)은 표 1과 같으며, 같은 조건에서 Original 메시지의 경우에는 표 2와 같다.

표 1. XOR Masking 메시지의 Stuffing-bit 발생 확률($P_m(X)$) 및 기대값($E_m(X)$)

$\lceil \log_2 n \rceil$	Stuffing-bit 발생 확률, $P_m(X)$			기대값 $E_m(X)$
	X=0	X=1	X=2	
1	1	0	0	0
2	1	0	0	0
3	0.875	0.125	0	0.125
4	0.875	0.125	0	0.125
5	0.875	0.125	0	0.125
6	0.844	0.156	0	0.156
7	0.754	0.234	0.012	0.258
8	0.728	0.254	0.018	0.291
9	0.728	0.254	0.018	0.291

표 2. Original 메시지의 Stuffing-bit 발생 확률($P_o(X)$) 및 기대값($E_o(X)$)

$\lfloor \log_2 n \rfloor$	Stuffing-bit 발생 확률, $P_o(X)$				기대값
	X=0	X=1	X=2	X=3	$E_o(X)$
1	0	0.234	0.594	0.172	1.938
2	0	0.305	0.609	0.086	1.781
3	0	0.324	0.633	0.043	1.719
4	0.086	0.402	0.477	0.035	1.461
5	0.157	0.477	0.341	0.025	1.234
6	0.199	0.509	0.268	0.024	1.117
7	0.251	0.561	0.176	0.012	0.949
8	0.260	0.572	0.159	0.009	0.917
9	0.260	0.572	0.159	0.009	0.917

표 1과 2에서 보는 바와 같이 Original 메시지에 비해 XOR Masking 메시지의 Stuffing-bit가 적게 발생하는 것을 알 수 있다. XOR Masking 메시지의 Stuffing-bit 감소 효과를 정량적으로 평가하기 위해 표 1와 2의 마지막 행에서는 Stuffing-bit 수의 기대값을 계산하였으며, 각 9개 기대값의 평균값을 비교하면 XOR Masking 메시지의 Stuffing-bit의 기대값은 0.152으로, Original 메시지의 기대값 1.337의 약 11.4%로 감소하였음을 알 수 있다.

4. Conclusion

본 연구에서는 CAN 프로토콜의 최대 단점인 낮은 네트워크 이용률과 CAN 메시지 응답시간을 향상시키고, 기존 연구에서 문제점으로 지적되는 CAN 메시지 우선순위 역전현상을 피하기 위해 CAN 메시지의 Stuffing-bit를 최소화할 수 있는 XOR Masking 메커니즘을 개발하였으며, 이를 위해 Stuffing-bit의 발생 확률과 Stuffing-bit의 기대값을 정의하였다. CAN 메시지의 상위 부분인 SOF, 중재 및 제어 필드에 대해 Original 메시지와 제안된 메커니즘을 적용한 XOR masking 메시지의 Stuffing-bit 발생 확률 및 기대값을 비교하는 실험을 통해 CAN 메시지의 길이가 줄어들었으며, 이에 따라 네트워크 대역폭 이용률이 증가하였다. 또한 본 논문에서 제안하는 메커니즘은 우선순위 배정을 위해 사용되는 비트 수에 비해 CAN 메시지의 수가 적은 임베디드 시스템에서 큰 성능 향상을 보이고 있음을 확인하였다. 추후에는 확장형 CAN 메시지 및 CAN 응용 계층에도 적용 가능한 Stuffing-bit 최소화 연구를 수행할 예정이다.

참고 문헌

[1] M. Farsi, K. Ratcliff, and M. Barbosa, " An Overview of Controller Area Network," Computing & Control Engineering Journal, Vol. 10, pp. 113-120, June 1999.

[2] Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed Communication. International Standards Organisation (ISO). ISO Standard-11898, Nov 1993.

[3] T. Nolte, H. Hansson, and C. Norstrom, " Minimizing CAN Response-Time Jitter by Message Manipulation," *Eighth IEEE Real-Time and Embedded Technology and* Applications Symposium (RTAS'02), pp. 197-206, Sep. 2002.

[4] T. Nolte, H. Hansson, and C. Norstrom, " Probabilistic worst-case response-time analysis for the controller area network," *Real-Time and Embedded Technology* and Applications Symposium, pp. 200-207, May 2003.

[5] K. W. Tindell, A. Burns, and A. J. Wellings, " Calculating Controller Area Network (CAN) Message Response Times," Control Engineering Practice, Vol. 3(8), pp. 1163- 1169, 1995.