Starvation Free Protocol in CAN

최준혁*, 정기현*, 최경희**
*아주대학교 전자 공학과
**아주대학교 정보 및 컴퓨터 공학과
e-mail : ddorong@ajou.ac.kr

Starvation Free Protocol in CAN

Jun-Hyuck Choi*, Gi-Hyun Chung*, Kyung-Hee Choi**
*Dept. of Electronics Engineering , Ajou University
**A Professional Graduate School for Information and Communication Engineering, Ajou University

요 약

이 논문은 Controller Area Network(CAN)에서 우선 순위를 가지는 메시지 전송 시 낮은 우선 순위 메시지 들의 starvation 방지 프로토콜을 제안한다. CAN 버스는 많은 부하가 걸리게 되면, 낮은 우선순위를 가지는 메시지는 전송될 기회를 잃어 버리게 되는 문제점을 안고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해서 Starvation Free 프로토콜을 제안하고, 제안된 프로토콜을 verification tool 로 검증한다.

1. 서론

CAN(Controller Area Network)는 마이크로 컨트롤러들 간의 통신을 위해 설계되었다. 산업이 고도로 발달함에 따라 다양한 기기들이 생겨나고 이러한 장치들간에 데이터 및 제어정보의 교환이 필연적으로 필요하게 되었다. CAN 은 이러한 기기들 간에 제어를 용이하게 하고 또한 현재 그 수요가 증가되고 있는 추세이다.

CAN 프로토콜은 message-oriented[3] 프로토콜이다. 즉, 메시지는 메시지를 보내는 노드에 의해서 구분되는 것이 아니라, 메시지 자체로 구분된다. 그리고 CAN 메시지는 우선순위를 가지고 있어서, 동시에 두 개 이상의 메시지가 버스에 접근하게 되면, 우선순위가 높은 메시지가 버스를 차지하게 된다.

이러한 전송 방식 때문에 CAN 에 높은 우선순위를 가지는 메시지의 전송이 많아지게 되면, 상대적으로 낮은 우선순위를 가지는 메시지는 전송할 기회를 잃어버리게 되는 문제점이 발생한다. 즉, 낮은 우선순위를 가지는 메시지에 대한 starvation 현상[3,4]이 나타날 수 있다.

CAN 에서의 starvation 현상을 막기 위한 여러 가지 연구가 있어왔다. 그 중에서 낮은 우선순위를 가지는 메시지의 전송을 확실하게 보장하는 방법으로 CAN 프로토콜을 수정한 medium utilization state tracking (MUST) 프로토콜[4]이 있다. MUST 의 기본 개념은 메시지의 burst 한 전송을 'run'[4]이라고 정의 하고, 이 'run'에서 각 메시지는 단 한번씩만 전송할 수 있도록 하는 것이다. 그리고 이전 메시지보다 높은 우선순위를 가지는 메시지는 그 'run'안에서 보내지 못하게 하는 것이다. 이러한 방법으로 MUST 는 낮은 우선순위를 가지는 메시지도 하나의 'run'안에서 한번은 보낼 수 있게 함으로써 starvation 현상을 막는다. 하지만 MUST 는 CAN 프로토콜 자체를 수정해야 하기 때문에 기존의 CAN 네트워크에 사용할 수 없을 뿐만 아니라, MUST 를 지원하는 CAN 컨트롤러를 사용해야 한다.

이 논문에서 제안하는 CAN starvation free(CSF) 프로토콜은 MUST 프로토콜의 기본 개념을 채용하여 CAN 네트워크상에서 발생할 수 있는 starvation 현상을 해결하기 위한 것이다. 그리고 이 프로토콜은 CAN 프로토콜을 수정하지 않고, CAN 프로토콜 위에 올라가는 레이어 형식의 소프트웨어 프로토콜이다.

이 논문의 내용은 다음과 같다. 제 2 장에서는 CAN 에서의 starvation 현상을 막기 위해서 CSF 프로토콜을 제안한다. 제 3 장에서는 제안한 프로토콜을 모델링하고, verification tool 로 검증한 내용을 다룬다.

2. CAN Starvation Free 프로토콜

2.1 Bitwise arbitration in CAN 프로토콜

CAN 에서 메시지의 전송은 프레임[1,3]단위로 일어난다. CAN 에서 모든 프레임은 우선순위를 가지게 된다. 프레임의 우선순위는 Arbitration 필드내의 11bit 의 IDENTIFIER 필드에 의해 결정되고, IDENTIFIER 필드의 값이 작을수록 높은 우선순위를 가진다. 일반적인 프레임의 형태는 그림 1[1,3] 과 같다.

그림 1 CAN 데이터 프레임의 구조

CAN 버스는 'dominant' 와 'recessive' 의 두 가지 신호 레벨을 가진다. 만약 'dominant' 와 'recessive' 비트가 동시에 버스에 접근하게 되면, 버스의 신호 레벨은 'dominant' 레벨이 된다. 이러한 수법을 bitwise arbitration[1,3]이라고 한다. CAN 버스가 idle 인 상태이면, 네트워크상의 모든 노드는 메시지를 전송하기 위해서 버스에 접근할 수 있다. 둘 이상의 노드가 동시에 버스에 접근할 때 일어나는 버스 충돌은 bitwise arbitration[1,3]에 의해 해결된다. 메시지를 전송하는 모든 전송 노드는 전송하고 있는 비트의 레벨과 버스에서 측정되는 레벨을 비교한다. 만약 비교한 결과가 서로 다르다면 다음 비트부터는 전송을 중지하고, 서로 같다면 계속해서 다음 비트를 전송한다. 이와 같은 방식을 CAN 프로토콜에서는 bitwise arbitration 이라고 한다. IDENTIFIER 필드를 전송하는 동안 bitwise arbitration 이 일어나고, 최종적으로 하나의 노드만인 버스 접근권한을 획득해서 데이터를 보낼 수 있게 된다. 그림 2 는 bitwise arbitration 의 한 가지 예를 보여 준다. Bitwise arbitration 을 사용하기 때문에 CAN 에서는 메시지의 손실이 일어나지 않고, 효율적인 메시지 전송이 가능하다.

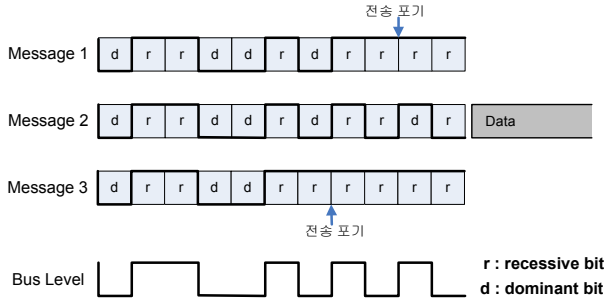

그림 2 CAN 프로토의 Bitwise arbitration

2.2 Starvation Free 프로토콜의 기본 개념

CAN Starvation Free(CSF) 프로토콜의 기본 개념은 메시지 전송을 일정한 규칙에 따라 제한함으로써 starvation 현상이 일어나는 것을 막는 것이다. CSF 프로토콜은 CAN 프로토콜 위에 올라가는 레이어 형식을 가지고, CAN 프로토콜 레이어와 어플리케이션 레이어 사이에서 메시지전송을 관리하게 된다.

CSF 프로토콜은 기본적으로 CAN 프로토콜의 메시지 전송 규칙을 따르면서, 메시지 전송을 관리하기 위한 추가적인 규칙을 가진다.

첫 번째로, CSF 프로토콜이 적용된 CAN 네트워크에서의 CAN 메시지들은 그룹을 가진다. 그룹들은 서로 다른 우선순위를 가지고, 같은 그룹 안의 메시지들도 각기 구분된다. CAN 메시지 그룹을 나누기 위해서 IDENTIFIER 필드를 'c_priority' 와 'id' 의 두 부분으로 나누어서 사용한다. 그림 2 는 IDENTIFIER 필드를 c_priority 와 id 로 나눈 모습을 보여준다.

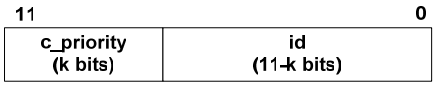

그림 3 CSF 프로토콜의 IDENTIFIER 필드 구조

두 번째, CAN 네트워크에 참여한 모든 노드들은 버스상에 어떠한 메시지가 전송되었는지를 표시할 수 있는 트랙킹 테이블(tracking table)을 가지고, 이 트랙킹 테이블에 표시된 메시지는 더 이상 전송하지 못하게 한다. 그림 3 은 트랙킹 테이블의 모양을 보여준다. 이 규칙을 적용하면 모든 메시지를 평등하게 전송할 수 있지만, CAN 프로토콜의 장점중의 하나인 우선순위에 따른 전송을 하지 못하게 된다.

어느 정도의 평등한 전송을 보장하면서, 우선순위에 따른 전송을 하기 위해서 세 번째 규칙을 가진다.

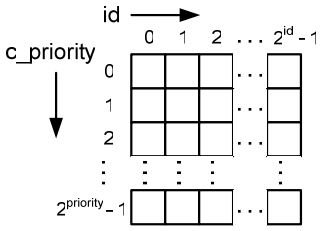

그림 4 트랙킹 테이블

세 번째, 버스에 메시지가 전송 될 때마다, 전송된 메시지의 c_priority 보다 작은 c_priority 를 가지는 트랙킹 테이블은 클리어된다. 예를 들면 c_priority 가 2 인 메시지가 전송되었다면, c_priority 가 0 인 트랙킹 테이블과 1 인 트랙킹 테이블은 모두 클리어된다. 즉, 이미 한번 전송되어 트랙킹 테이블에 표시된 메시지라고 하더라도, 버스상에 이 메시지보다 우선순위가 낮은 그룹의 메시지가 전송되었다면, 트랙킹 테이블이 클리어되어 다시 보낼 수 있게 된다. 이와 같은 규칙을 적용함으로써 같은 그룹의 메시지들(c_priority 가 같은)은 동일한 전송 횟수를 가지지만, 높은 우선순위를 가지는 그룹의 메시지일수록 더 많은 전송 횟수를 보장받는다. 그리고 우선 순위가 가장 낮은 그룹의 메시지라도 일정 시간 안에 반드시 전송할 수 있다.

만약 모든 트랙킹 테이블이 표시되면, 더 이상 어떠한 메시지도 전송될 수 없다. 또한 특정 메시지가 전송 되지 않기 때문에, 다른 메시지들이 전혀 전송할 수 없는 경우도 생길 수 있다. 예를 들면 c_priority 가 0 과 1 인 트랙킹 테이블이 모두 표시 되었다면, c_priority 가 1 보다 큰 메시지가 전송되기 전 까지는 더 이상 c_priority 0 과 1 인 그룹의 메시지를 보낼 수 없다. 이런 상황에서 c_priority 1 보다 큰 메시지들 중 보낼 메시지가 없다면, 어떠한 메시지도 전송될 수 없

다. 이러한 데드락(deadlock) 현상이 일어날 수 있기 때문에 네 번째 규칙이 필요하다.

네 번째, 일정시간 동안 버스상에 전송되는 메시지가 없다면, 즉, bus idle time 이 일정시간 지속되면, 트랙킹 테이블을 모두 클리어 시킨다. 네 번째 규칙을 적용하기 위해서는 버스가 idle 인 시간을 측정해야 하고, CAN 네트워크상의 모든 노드들의 트랙킹 테이블을 클리어시키기 위한 메시지가 필요하다. 그래서 이러한 역할을 하는 마스터 노드를 둔다. 마스터 노드는 버스의 상태를 계속 모니터링하고, bus idle 인 시간이 일정시간을 넘게 되면 'Tracking Table Clear(TTC)' 메시지를 보낸다. TTC 메시지는 CAN 네트워크상에 전송되는 메시지들 중 가장 높은 우선순위를 가지도록 해서, 다른 메시지의 전송 때문에 TTC 메시지가 전송되지 못하는 일이 일어나지 않도록 한다.

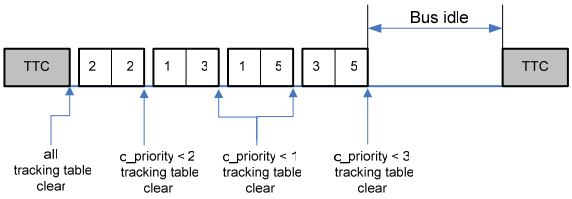

그림 5 CSF 프로토콜에서의 메시지 전송

3. CSF 프로토콜 모델링 & Verification

3.1 CSF 프로토콜 모델링

CSF 프로토콜이 적용된 노드를 CAN 트랜시버와 CPU 부분으로 나누어 finite-state model(FSM)[7,9,10]으로 모델링 하였다. CAN 트랜시버 부분은 CAN 프로토콜을, CPU 부분에는 CSF 프로토콜을 적용해서 이 두 부분이 서로 동시에 동작을 하게끔 모델링 하였다. 그리고 CPU 부분은 일반 노드와 마스터 노드로 구분해서 모델링 하였다.

그림 8 은 CAN 의 트랜시버를 모델링한 그림이다. CAN 트랜시버는 CAN 버스와 동기화 되어 동작하기 때문에 따로 CAN 버스는 모델링 하지 않았다. 그리고 CAN 프로토콜의 arbitration 기능만을 모델링 하였다.

CAN 트랜시버는 Idle, SOF, Arbitration, Tx, Rx 의 5 가지 상태를 가진다. Idle 상태는 트랜시버가 아무런 동작도 하지 않는 상태이다. 즉, bus idle 인 상태이다. SOF 는 다른 CAN 트랜시버와의 동기화를 맞추기 위해 설정한 가상적인 상태이다. 트랜시버에서 메시지를 전송하고자 하면, SOF 상태로 가고 그 다음 상태에 Arbitration 상태가 된다. 다른 트랜시버들은 SOF 상태인 트랜시버가 있다면, 다음 상태에서 바로 Arbitration 으로 간다. 이런 방식으로 CAN 네트워크상의 모든 트랜시버는 동시에 Arbitration 상태로 갈 수 있다. Arbitration 에서 이긴 트랜시버는 Tx 상태로 가게 된다. Tx 상태는 버스 접근권한을 획득해서 데이터를 보내는 상태를 말한다. Arbitration 에서 진 다른 모든 트랜시버는 Rx 상태로 가서, Tx 상태인 트랜시버가 보내는 데이터를 받는다.

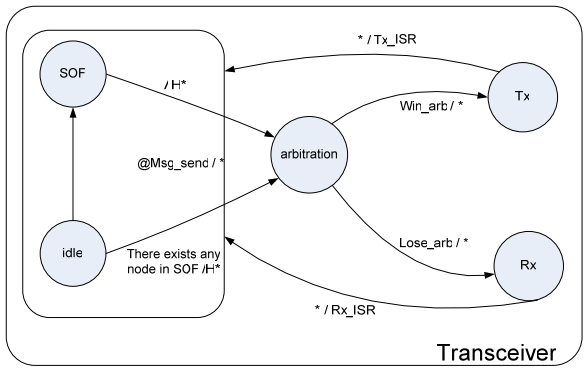

그림 6 CAN 트랜시버 모델링

그림 9 는 일반 노드의 CSF 프로토콜을 모델링한 것이다. CSF 프로토콜과 인터럽트 서비스 루틴(ISR)만을 모델링 하였다. message ready 상태는 어플리케이션에서 메시지를 받아 전송을 준비를 하는 상태이다. TT check 상태는 전송하고자 하는 메시지가 전송될 수 있는지 트랙킹 테이블을 검사하는 상태이다. 전송할 수 있는 메시지라면 Tx Buffer Check 상태로 가서 버퍼의 상태를 검사한다. 여기에서 버퍼는 CPU 와 트랜시버간의 메시지 버퍼이다. 버퍼가 빌 때까지 대기 했다가 Setting for Message 상태로 가게 된다. Setting for Message 상태는 트랜시버에게 메시지를 전달하는 상태이다. Rx ISR 상태와 Tx ISR 상태는 트랜시버에서 메시지 수신/송신 완료 후에 들어가게 되는 인터럽트 서비스 루틴 상태이다. 인터럽트 서비스 루틴에서는 트랙킹 테이블에 버스에 전송된 메시지를 표시한다.

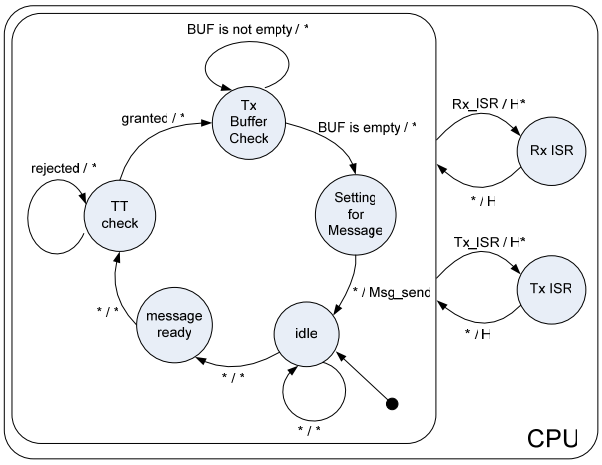

그림 7 CSF 프로토콜이 적용된 일반 노드의 CPU 모델링

그림 10 은 마스터 노드의 CSF 프로토콜을 모델링한 것으로, 일반 노드의 CPU 모델링과 거의 유사하다. 차이점은 line_busy_check 상태에서 버스의 상태를 항상 감시하고, bus idle 상태가 일정시간 이상 지속되면 TTC 메시지를 보낸다는 것이다.

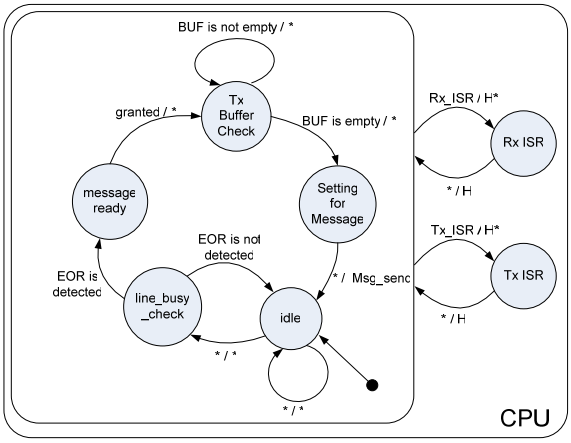

그림 8 CSF 프로토콜이 적용된 마스터 노드의 CPU 모델링

3.2 프로토콜 Verification

3.2.1 Cadence SMV

FSM 을 verification 하기 위해서 Cadence SMV 를 사용하였다. Cadence SMV 는 symbolic model checking[8] 기술을 기반으로 하고 있는 formal verification system 으로 Cadence Berkeley Lab.에서 개발되었다.[7,8] 그리고 시스템 모델을 기술하기 위해서 CTL model checking 을 지원하는 Verilog hardware description language 를 사용한다.[8] Cadence SMV 에서 검증할 특성은 temporal logic 으로 기술되어서 신호들 사이의 시간관계를 정확하게 기술할 수 있게 해 준다. 또한 finite state model 과 combinational logic 의 특성을 자동적으로 검증하는 데 유용한 verification tool 이다.

3.2.2 Verification 특성

CSF 프로토콜이 정확한 동작을 하는지를 검증하기 위한 특성은 아래와 같다.

(1) CAN 버스의 arbitration에서 우선 순위가 높은 메시지가 선택되는가?
(2) 모든 노드는 모든 경우에 정상적인 방법으로 arbitration에 참여하는가?
(3) 한 노드가 송신상태(Tx)일 때 다른 나머지 노드들은 모두 수신상태(Rx)인가?
(4) 두 개 이상의 노드가 동일한 시점에서 송신상태(Tx)는 아닌가?
(5) Tracking table에 표시된 메시지가 전송되지는 않는가?
(6) 마스터 노드는 버스가 일정시간 동안 idle상태이면 TTC 메시지를 전송하는가?
(7) TTC 메시지를 받은 노드들은 tracking table을 클리어 하는가?
(8) 각 노드는 버스에 전송된 메시지보다 높은 우선순위의 트랙킹 테이블을 클리어 하는가?

Cadence SMV 로 위의 특성들을 검증 했을 때 모두 참인 결과가 나왔다.

4. 결론

이 논문은 CAN 네트워크에서의 starvation 현상을 막기 위한 CAN Starvation Free(CSF) 프로토콜을 제안하였다. CSF 프로토콜은 CAN 프로토콜 위에 레이어 형태로 적용되기 때문에 기존의 CAN 네트워크에도 적용할 수 있다. CSF 프로토콜은 CAN 메시지를 그룹화해서, 그룹의 우선순위에 따라 메시지 전송 횟수를 조절한다. 이러한 방법으로 CAN 에서 발생할 수 있는 낮은 우선순위를 가지는 메시지의 starvation 현상을 막을 수 있다.

그리고 제안한 프로토콜의 동작을 검증하기 위해서 CSF 프로토콜을 모델링하고, 이 모델을 verification tool 을 사용해 검증하였다. 검증한 결과 CSF 프로토콜이 유효하다는 것을 확인할 수 있었다.

현재는 CSF 프로토콜의 실제 성능 측정과 CSF 프로토콜의 네 번째 규칙인 'bus idle time 이 일정시간 지속되면, 트랙킹 테이블을 모두 클리어시킨다.'에서 bus idle time 이 어느 정도 지속되었을 때 TTC 메시지를 보낼 것인가에 대한 고려를 하지 못했다. 향후에는 CSF 프로토콜을 실제 시스템에 구현해 버스 로드(load)에 따른 성능을 평가해 볼 것이다.

참고문헌

[1] Bosch. CAN specification version 2.0. Published by Robert Bosch GmbH, September 1991.

[2] Road Vehicles – Interchange of Digital Information – Controller Area Network for High-Speed Communication, ISO 11898, Nov. 1993.

[3] K. Etschberger, " Controller Area Network – Basics, Protocols, Chips and Applications", IXXAT Press, 2001.

[4] G. Cena and A. Valenzano, "Achieving Round-Robin Access in Controller Area Networks", IEEE Transaction on Industrial Electronics, Vol. 49, No. 6, December 2002.

[5] T. Nolte, H. Hansson, L. L. Bello, "Implementing Next Generation Automotive Communications", Embedded Real-Time Systems Implementation Workshop in conjunction with the 25th IEEE International Real-Time Systems Symposium, December 2004.

[6] Khawar M. Zuberi and Kang G. Shin, "Design and Implementation of Efficient Message Scheduling for Controller Area Network", IEEE Transaction on Computers, Vol. 49, No. 2, February 2000.

[7] M. van Osch and S. A. Smolka, "Finite-State Analysis of the CAN Bus Protocol", Proceedings of Sixth IEEE International Symposium on High Assurance Systems Engineering, IEEE Press, October 2001.

[8] K. McMillan, " Getting Started with SMV, User's Manual", Cadence Berkeley Laboratories, 1998.

[9] K. McMillan, " Symbolic Model Checking" Kluwer Academic Publishers, 1993.

[10] A. A. Mir, S. Balakrishnan and S. Tahar, " Modeling and Verification of Embedded Systems Using Cadence SMV", Electrical and Computer Engineering, Vol. 1, March 2000.

[11] M. D. Natlae, " Scheduling the CAN Bus with Earliest Deadline Techniques", In Proceedings of the 21st IEEE Real-Time System Symposium, December 2000.