韓國精密工學會 2000年度 秋季學術大會論文集 pp.717~720 KSPE 00F162

메시지 스케줄링을 이용한 Brake-by-wire 시스템의 Redundancy Management

윤종운*,김기웅,김태열,김재구(부산대 대학원 지능기계공학과),이석(부산대 기계공학부)

Redundancy Management of Brake-by-wire System using a Message Scheduling

J. W. Yune, K. W. Kim, T. Y. Kim, J. G, Kim(Int. Mech. Eng. Dept., PSNU)
S. Lee(Mech. Eng. Dept., PSNU)

ABSTRACT

Event-driven communication protocols such as CAN(Controller Area Network) have inherent packet delays due to the contention process for the use of network medium. These delays are stochastic in nature because most packets arrive at random time instants. The stochastic property of the delay adversely influences the control system's performance in terms of stability, responsiveness and steady-state error. Another problem for safety-critical application such as brake-by-wire systems is the reliability of the communication modules that can fail abruptly. This paper deals with two methods to overcome the above problems : (i) scheduling method that can maintain packet delays under some acceptable level, and (ii) redundancy management of communication modules that prescribes dual-redundancy modules' behavior when one of them fails

Key Words : Brake-by-wire, Message scheduling, CAN(Controller Area Network), Redundancy management

1. 서론

현재까지의 차량용 네트워크 시스템의 적용이 주로 wiring harness의 감소에 초점을 맞춘 반면 최근 진행되고 있는 X-by-wire 기술에 대한 연구는 차량의 기본적인 기능, 즉 구동, 정지, 조향 등의 기능을 하는 기계적인 메커니즘을 물리적인 backup 시스템 없이 네트워크로 연결된 전기, 전자적 장치로 바꾸려는 시도이다. Steer-by-wire, Brake-by-wire system등이 그 대표적인 예이다.

By-wire 시스템을 구성하고 있는 각각의 모듈을 연결하고 실시간 메시지의 처리를 위해서는 적절한 IVN(In-vehicle-network)용 프로토콜이 사용되어야 하는데 CAN(Controller Area Network), TTP/C(time triggered protocol / class C)등이 대표적이다.

CAN[1]은 고성능, 저비용의 직렬통신 프로토콜로 1980년대 독일 Bosch사에서 개발되었다. 높은 데이터 전송률을 기반으로 한 실시간성과 안정성, 확장성을 제공하여 자동차를 위한 실시간 제어용 프로토콜로 현재 가장 많이 쓰이고 있으며 활용영역을 공장 자동화, 빌딩 자동화, 항공기, 자동화 기기등으로 넓히고 있다. CAN에서의 메시지 처리방식은 ID를 이용한 경쟁방식을 통해 목적한 노드에 데이터를 전송하며, 전송될 데이터의 우선 순위도 결정된다. 또한 CAN은 event driven 방식을 이용한 프로토콜로 네트워크 상의 부하가 적은 soft real time 시스템의 구현에는 그 우수성이 입증되었다. 그러나 by-wire 시스템과 같은 hard real time 시스템에는 응답시간이 일정하지 못함으로 인해 네트워크 성능(performance)이 만족스럽지 못한 것으로 보고되고 있다. 이러한 문제를 해결하기 위해 ISO에서는 CAN 프로토콜의 session layer에 2가지 옵션기능을 첨가한 TTC(Time Triggered communication on CAN)를 표준화 (ISO11898-4)시키기 위해 논의 중에 있다. TTC는 CAN 프로토콜의 자동 재전송 기능을 switch-off 시킬 수 있게 했으며 전송된 메시지에 time capturing 기능을 삽입하였다.[2]

CAN을 이용하여 by-wire 시스템의 네트워크 구성을 위해 가장 중요한 것은 메시지의 전달과정에서 생길수 있는 전송지연을 일정 하게하고 jitter를 최소화 하는데 있다. 본 논문에서는 전송하게될 메시지들을 스케줄링을 이용하여 이러한 문제들을 최소화 되게 하였다. 아울러 네트워크 노드를 redundancy화하여 fault로 인한 전체 시스템의 성능 변화를 없게 하였다.

2. Brake-By-wire 시스템

Brake-by-wire 시스템은 운전자의 조작을 센서로 입력받아 네트워크를 통해 브레이크 제어모듈의 액추에이터를 작동시켜 차량이 정지 혹은 감속하게 만드는 기술이다. 즉 브레이크 제어모듈은 바퀴에 인가된 하중과 속도를 센서를 통해 입력받아 운전자가 원하는 방향과 속도로 차량을 운행할 수 있도록 각각의 구동륜에 다른 제동력을 인가하게 된다. 전통적인 브레이크 시스템이 유압작동기를 기반으로 한 기계메커니즘에 기반을 둔 반면 By-wire 시스템은 센서와 액츄에이터, ECU등을 이용한 메카트로닉스 시스템이다. Brake-by-wire 시스템은 부가적인 하드웨어 없이 다음과 같은 여러 가지 기능과 장점을 제공한다.[3][4]

- ABS, TCS, ESP등과 같은 기능을 소프트웨어로 대체
- 중량경감
- 유지보수 및 고장진단이 용이
- 안정성 향상
- 장기적인 관점에서 적은 비용.

3. 네트워크 시스템과 제어기

3.1 네트워크 시스템

CAN 프로토콜에서 메시지의 우선순위 지정방식은 각각의 메시지가 가지는 정보의 중요도와 응답 deadline등을 고려하여 설정하게 된다. Hard real-time 시스템과 같이 메시지의 발생주기가 짧고 발생빈도가 빈번한 경우 CAN과 같은 경쟁방식을 쓰는 프로토콜은 우선순위가 높은 메시지는 deadline내에 전송이 가능하나 우선순위가 낮은 메시지는 최악의 경우 deadline을 지키지 못하거나 전송을 실패하는 경우가 발생할 수 있다. 그러므로 CAN 프로토콜을 기반으로 하는 제어 시스템의 설계에 있어 이러한 문제점을 극복할 수 있는 방안이 요구되는데, scheduling이 하나의 대안이 될 수 있다. 즉 각각의 제어기에서 발생되는 메시지를 순차적으로 발생 시킴으로서 충돌로 인해 발생될 수 있는 지연과 jitter를 제거 할 수 있다. Fig. 1에서 보는바와 같이 각각의 메시지는 중앙의 제어모듈에 가상적으로 동기화 되어 있으며 시간경과에 따라 연속적으로 데이터를 전송하게 된다. 전체 메시지 전송주기는 3ms이다.

메시지의 전송시 생길수 있는 영향을 비교하기 위해 두방식을 사용하여 DC모터를 비례-미분제어기를 통해 제어하는 실험을 하여 제어기의 성능을 알아보았다. CAN 프로토콜로 연결된 5개의 노드로 구성되어 있으며 전송지연과 jitter로 생길수 있는 제어기의 영향을 쉽게 확인하기 위해 메시지의 발생주기를 5ms로 주기적으로 발생하게 하였으며, 전송속도를 낮게하여(125Kbps) 메시지들간의 충돌 확률을 높였다.

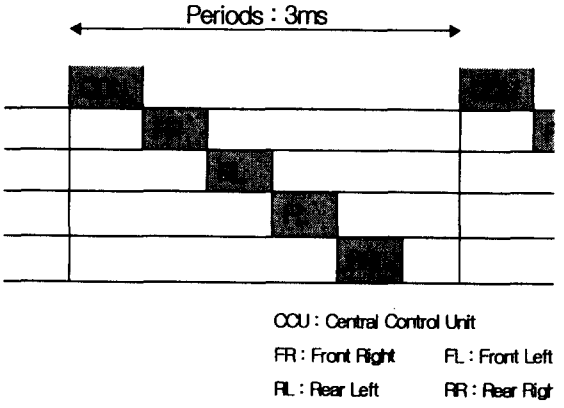

Fig.1 Message scheduling

Fig. 2에 보는바와 같이 경쟁방식의 경우 제어기가 원하는 값에 수렴하지 못하고 발산함을 관찰할 수 있다. 그러나 스케줄링의 경우 overshoot가 상당히 크고 약간의 진동이 있지만 제어가 어느 정도 이루어지고 있음을 알 수 있다.

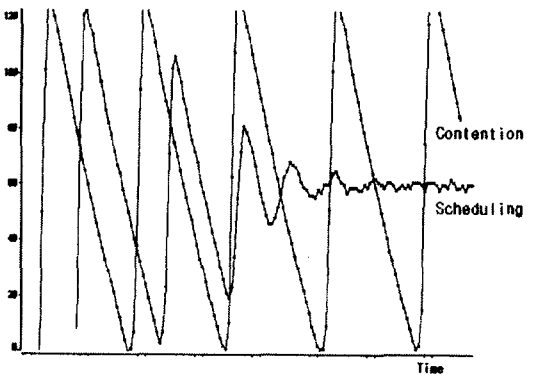

Fig.2 Comparison of scheduling and contention

3.2 제어기의 구성

3.2.1 하드웨어 및 소프트웨어

제어기의 redundancy화를 위해 제어노드에는 2개의 동일한 기능- 2개의 FSU(fault silent unit)가 하나의 FTU(fault tolerant unit)을 이룬다 -을 하는 모듈로 구성되어 있으며, 바퀴(Wheel)에 해당하는 노드는 1개의 모듈로 구성하였다. 이들 노드는 Fig. 3과 같이 CAN 프로토콜을 이용하여 하나의 공

유매체에 접속되어 있다.

각각의 스테이션 모듈은 Infineon사의 C167CR [6]마이크로 컨트롤러로 version 2.0B를 지원하는 CAN 프로토콜 컨트롤러를 내장하고 있다. DC모터의 제어실험을 위해서 PWM(pulse width modulation), 타이머와 CAN등을 이용하였으며 정확한 동작을 위해 인터럽트(interrupt)로 모든 루틴(routine)을 처리하였다.

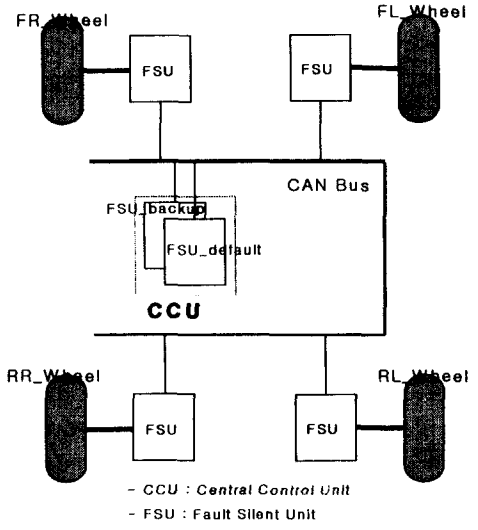

Fig.3 Brake-by-wire system structure

중앙의 제어노드에서는 각각의 바퀴에서 수신된 속도신호를 비례-미분 알고리즘을 이용하여 적절한 제어값을 2byte씩 총 8Byte의 정보를 전송하게 된다. 바퀴에서는 DC 모터에 연결된 엔코더(encoder)의 펄스를 계측하여 전송하고 제어노드로부터 수신된 제어값을 PWM신호로 바꾸어 DC 모터 드라이버를 구동하게 된다. 전체 프로그램은 C를 이용하여 구성하였다.

3.2.2 Redundancy management를 위한 구조

Redundancy화된 제어노드의 management를 위해서는 몇가지 고려해야할 점이 있다.

첫째 FSU_backup 모듈에서는 전원이 켜짐과 동시에 메시지를 전송할 수 없으므로 우선 FSU_default가 발생시키는 메시지를 모니터하고 있어야 한다. 즉 일정한 시간 간격으로 발생하는 인터럽트 신호를 처리하는 루틴내에서 모니터 된 메시지를 카운트하여 정해진 값 이하가 되면 FSU_default가 정상적인 작동을 하고 있지 못하다고 판단하게 된다. 이는 인터럽트 루틴시간 간격과 새로운 메시지의 발생까지 지연이 생기게 됨을 의미한다.

둘째 FSU_default가 외부의 물리적인 충격으로 전기적인 손상이 생겨 더 이상 정상적인 기능을 할 수 없을 때는 문제가 되지 않으나 watchdog 타이머 등에 의해 리셋(reset) 되었을 때 무조건 적인 메시지 생성은 FSU_backup에서 발생하는 메시지와 ID가 같게 되므로 충돌이 발생할 수 있다. 그러므로 리셋이 되거나 전원이 최초로 켜졌을 때 자신이 발생시킬 메시지와 같은 ID를 가진 메시지가 네트워크 상에 존재하는지 모니터해야 한다.

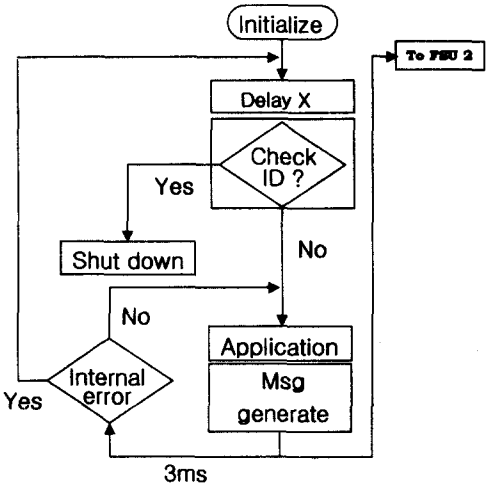

Fig.4 FSU_default logic flow chart

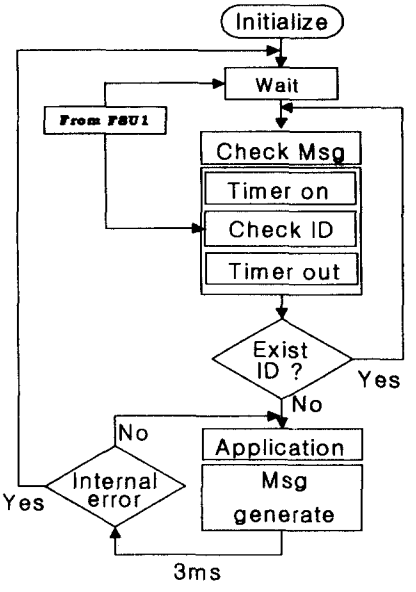

Fig.5 FSU_backup logic flow chart

FSU_default는 일정한 시간동안 전송하게될 ID와 같은 메시지가 네트워크상에 통신이 이뤄지고 있는지를 검색하고 존재하고 있으면 shut down모드로 들어가며 그렇지 않은 경우 정상적인 동작을 하게

된다.

FSU_backup은 FSU_default에서 전송하는 제어신호를 수신하게 되면 타이머에 의해 발생하는 일정한 시간 간격마다 수신된 메시지를 카운트하여 자신이 메시지를 발생시켜야 될지를 판단하게 된다. 모든 FSU들은 watchdog 타이머에 의해 내부 오류가 발생하게 되는 경우 자동적으로 reset 명령이 소프트웨어적으로 발생되도록 하였다.

4. Redundancy Management

4.1 Redundancy Management를 위한 실험방법

중앙 제어모듈과 각 바퀴들은 CAN 프로토콜을 이용하여 1Mbps의 속도로 송수신을 행한다. 제어신호는 3ms의 주기로 발생하며 바퀴내의 encoder 신호도 3ms의 주기로 갱신하게 하였다.

DC 모터를 1200rpm의 속도로 제어하기 위해 3ms마다 계측된 펄스수를 기준- 60회, 분해능 1000펄스-으로하였다. .

전체 시스템의 동작과 발생할 수 있는 문제들에 따라 2가지의 작동방법으로 나누었다. 즉 FSU_default 모듈이 fault로 정상적인 동작을 할 수 없을때 이를 대신하여 역할을 수행하게될 FSU_backup 모듈의 작동을 불러일으킬 수 있는 상황들이다.

첫째, 내부 연산 오류로 인한 watchdog 타이머의 작동을 시뮬레이션 하기 위해 167CR 보드에 있는 리셋 스위치를 작동 시켰다.

둘째, 외부의 물리적인 충격으로 인한 하드웨어의 작동정지를 위해서 테스트 보드의 전원을 off 시키는 것으로 대신하였다.

4.2 실험 결과

CANalyzer를 통하여 관찰된 제어된 DC 모터의 속도를 Fig. 6에서 보여주고 있다. fault로 인한 영향을 시뮬레이션하기 위해 시스템이 작동하고 있는 동안 리셋 스위치를 작동시키거나 시스템의 전원을 off 시켰다. 그러나 전체 시스템의 성능에는 영향을 주지 않음을 알 수 있다. 실제 CANalyzer를 통해 관찰된 FSU_default와 FSU_backup사이의 전환시간은 3ms 정도로 전체 시스템의 제어성능에 영향을 주어야하나 바퀴 모듈이 이전 제어정보를 이용하여 계속적으로 갱신하는 과정을 거치기 때문에 예상보다 영향이 적음을 알수 있다.

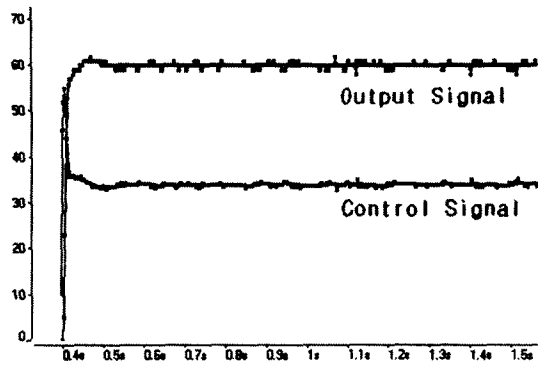

Fig.6 DC motor speed control using PD controller

5. 결론

Brake-by-wire 시스템 구축을 위하여 한 개의 redundancy 모듈과 4개의 single 모듈간의 운영에 관하여 실험을 통하여 실제 작동과 문제점에 대해 논하였다. CAN을 이용하여 by-wire 시스템의 네트워크를 구성시 backup 모듈의 작동시 나타나는 지연시간이 문제가 됨을 알수 있다. 또한 제어된 모터의 속도가 완전히 수렴하지 않고 미세한 양이나 떨림이 있는 것을 관찰 할 수 있다.

스케줄링 기법이 hard real-time 시스템의 제어기용 네트워크에 근간을 이루어 구현을 하는 것이 적합함을 실험을 통해 확인할 수 있었다.

참고문헌

1. W. Lawrenz, "CAN System Engineering From Theory to Practical Applications", Springer, 1997

2. CiA(CAN in automation), "CAN Newsletter", marketing holger aeltwanger. pp. 46, June, 2000

3. K. Holiding "Brake By Wire", IEE Colloquium, 1990

4. H. Kopetz, "Automotive Electronics", IEEE, 1999

5. B. Hedenetz, R. Belschner, "Brake-by-wire without Mechanical Backup by Using a TTP-Communication Network", SAE Technical paper, 1998

6. Infineon, "C167 Derivatives : 16bit CMOS single-chip microcontroller user manual", Infineon, 1996