■ 2008년도 학생논문 경진대회 수상작

CANopen 네트워크 이용률 감소를 위한 PDO 패킹 메커니즘

(PDO Packing Mechanism for Reducing CANopen Network Utilization)

강 민 구 † 박 기 진 †† 김 종 철 †††
(Minkoo Kang) (Kiejin Park) (Jongcheol Kim)

요 약 최근 각광받고 있는 차량 내부 네트워크(In-Vehicle Network)의 일종인 CANopen 프로토콜은 다양한 벤더의 하드웨어 특성에 의존적인 CAN(Controller Area Network) 기반 응용 프로그램 개발의 문제점을 해결하고자 제안되었으며, 프로파일링(Profiling) 개념을 사용하여 CAN과 이의 응용 계층인 CAL(CAN Application Layer)에서 동작하는 모든 하드웨어 장치를 지원함에 따라 CAN 기반 응용 시스템의 개발 기간의 단축이 가능하다. 메시지 처리 성능(예: 최악 응답 시간)을 높이기 위해서는 CANopen 네트워크 이용률(Utilization)을 감소시킬 필요성이 있으며, 이를 위해 가능한 많은 메시지를 패킹(Packing)하여 전송함으로써, 메시지 전송 시 발생하는 메시지 프레임의 오버헤드를 줄이는 것이 바람직하다. 이에 본 논문에서는 CAN의 응용 계층에서 동작하는 CANopen의 OD(Object Dictionary) 및 PDO (Process Data Object) 통신 서비스를 이용하는 PDO 패킹 메커니즘을 제안하였다. SAE(The Society of Automotive Engineers)에서 제공하는 벤치마크(Benchmark) 자료를 이용하여, 본 논문에서 제안한 메커니즘의 성능을 평가하였으며, 선행 연구에 비해 CANopen 네트워크 이용률이 약 10% 가량 감소하는 것을 확인하였다.

키워드 : CAN, CANopen, PDO, 네트워크 이용률, 최악 응답 시간

Abstract CANopen which is one of the in-vehicle network (IVN) protocols is adopted to solve the hardware dependency problem of the CAN-based application. CANopen makes different CAN devices interoperable each others. By the advantage of the device profiling concept, it can make the period of developing CAN-based application system shorten. The utilization of CANopen network must be reduced to improve the communication performance (e.g. worst-case response time). For reducing network utilization, messages need to be packed as many as possible so that message frame overhead can be decreased. In this paper, we suggested a PDO packing mechanism using object dictionary (OD) and process data object (PDO) communication service in CANopen. Through experiments, the performance of the mechanism is evaluated with SAE benchmark. As a result, network utilization is decreased about 10% compared to the result of the previous works.

Key words : Controller Area Network(CAN), CANopen, Process Data Object(PDO), Network Utilization, Worst-Case Response Time(WCRT)

· 본 연구는 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-331-D00438)

† 학생회원 : 아주대학교 산업공학과
ozige@ajou.ac.kr
†† 종신회원 : 아주대학교 산업정보시스템공학부 교수
kiejin@ajou.ac.kr
††† 비 회 원 : 아주대학교 산업공학과
kbellfe@ajou.ac.kr

논문접수 : 2008년 4월 24일
심사완료 : 2008년 8월 18일

정보과학회논문지: 시스템 및 이론 제36권 제2호(2009.4)

1. 서 론

차량 내부 네트워크(In-Vehicle Network)의 일종인 Controller Area Network(CAN)은 1980년대 BOSCH에서 개발된 이후, 현재에 이르기까지 차량 제어 시스템을 포함한 많은 산업용 임베디드 제어 시스템에서 성공적으로 사용되고 있는 네트워크 프로토콜로써, 최고 1Mbit/s의 대역폭과 최대 8 Bytes의 비교적 짧은 데이터 길이에 따른 낮은 지연시간을 특징으로 한다. 또한 CAN 컨트롤러는 CAN 메시지의 여러 가지 오류(i.e. Acknowledgement Error, Bit Error, Bit Stuffing Error, Cyclic Redundancy Checksum(CRC) Error, Form Error)를 감지하는 메커니즘을 데이터 연결 계층(Data Link Layer)에서 자체적으로 제공하며, 감지된 오류 횟수를 기록함으로써 자신의 상태(Error Active State, Error Passive State, Bus-Off State)를 전환 및 관리할 수 있다. 게다가 CAN 메시지의 전송 방식인 CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance)으로 인해 대역폭의 손실 없이 CAN 버스 상에서의 메시지 충돌을 방지할 수 있는 장점으로 인해[1,2], 최근 자동차 분야에서 그 사용이 급격히 증가하고 있다[3-5].

CAN 메시지 프레임 양식은 식별자(Identifier)의 길이에 따라 표준형(Standard 2.0A, 11-bit identifier)과 확장형(Extended 2.0B, 29-bit identifier)으로 구분되며, CAN 메시지의 전송 목적에 따라 4가지 프레임 양식(Data Frame, Remote Frame, Error Frame, Overload Frame)이 있다. 표준형 CAN 메시지의 데이터 프레임 양식은 그림 1에서 보는 바와 같이 Start-Of-Frame (SOF) Bit, Arbitration Field, Control Field, Data Field, CRC Field, Acknowledgement Field, End-Of-Frame(EOF) Flag 및 Inter Frame Spacing(IFS)으로 구성되어 있다[6].

CANopen 프로토콜은 다양한 벤더의 하드웨어 특성에 의존적인 개발 과정을 따르는 CAN 기반 응용 시스템 개발의 문제점을 해결하고자 제안되었으며, CAN과 CAL(CAN Application Layer)에서 동작하는 모든 하드웨어 장치를 지원하기 위해 프로파일링(Profiling) 개념을 사용하였고, 이로 인해 CAN 기반 응용 시스템의 소프트웨어 개발 기간을 단축시킬 수 있다. 이러한 프로파

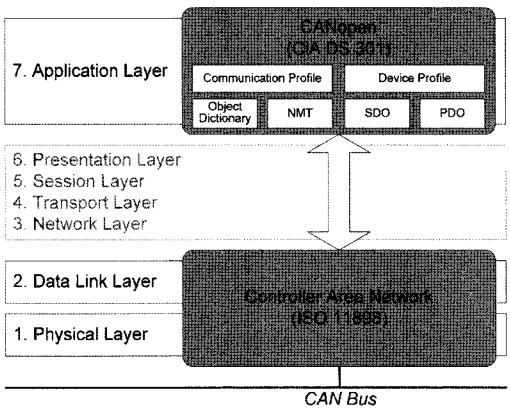

그림 2 CANopen의 프로파일링

일링 개념을 구현하기 위해 CANopen에서는 OD(Object Dictionary), SDO(Service Data Object) 및 PDO (Process Data Object)가 정의되었다[7](그림 2 참조).

OD는 네트워크를 구성 및 운영하기 위한 각종 파라미터로 이루어진 일종의 참조(Look-up) 테이블이며, 표 1에서 보는 바와 같이 32 비트 인덱스 범위로 나누어져 있고, EDS(Electronic Data Sheet) 또는 DCF(Device Configuration File)의 형태로 각 CANopen 노드에 저장되어 있다. OD 인덱스 범위 0001h-0FFFh는 Data Types을 정의하기 위해 사용되며, 1000h-1FFFh에서는 CANopen 통신을 위한 파라미터가 정의되고, 2000h-5FFFh에서는 CANopen 표준이 아닌 하드웨어 장치의 데이터 저장 등을 위해 사용된다. SDO는 각 노드에 저장되어 있는 OD의 내용을 읽고 쓰는 작업을 통해 네트워크 초기화 등을 수행하기 위해 사용되지만, 프레임의 구조적인 특징 때문에 일반적인 데이터 전송 작업을 수행하기에는 적합하지 않다. 이를 해결하기 위해 정의된 PDO는 네트워크 운영 상태에서 효율적인 데이터 전송 작업을 수행한다. 한편 응용 계층(Application Layer)에서 동작하는 SDO와 PDO 같은 CANopen 메시지는 궁극적으로 데이터 연결 계층에서 동작하는 CAN 메시지로 변환되어 처리된다. 또한 수신된 메시지의 수취 여부를 결정함에 있어서 CAN에서는 CAN 메시지 ID를 사용하는 반면, CANopen에서는 PDO의 ID 값으로 OD 인덱스를 참조한 뒤에 데이터의 수취 여부를 결정한다[8].

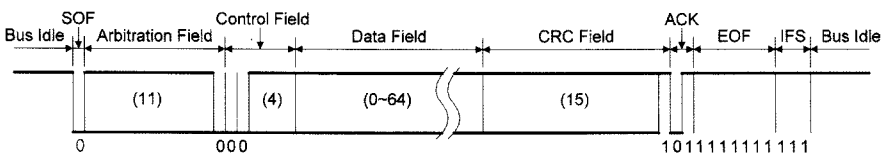

그림 1 표준형 CAN 메시지의 데이터 프레임 양식

표 1 Object Dictionary 인덱스 범위

인덱스 범위	설명
0000h	Reserved
0001h - 0FFFh	Data Types
1000h - 1FFFh	Communication Entries
2000h - 5FFFh	Manufacturer Specific
6000h - 9FFFh	Device Profile Parameters
A000h - FFFFh	Reserved

CAN과 CANopen 프로토콜은 Event Triggered 방식의 대표적인 예로써 Time Triggered 방식과 비교하였을 때, 네트워크의 유연성(Flexibility)이 좋다는 장점이 있는 반면에 메시지의 응답 시간(Response time)을 결정적으로(Deterministic) 보장할 수 없다는 단점을 가진다[4]. 이러한 단점을 해결하기 위해 FPNS (Fixed-priority non-preemptive scheduling) 기법을 적용하여 CAN 메시지의 최악 응답 시간(Worst-case response time)을 계산한 연구가 선행되었으며[9,10], 이들은 현재까지 CAN 기반 응용 시스템의 실시간 처리 시간 분석을 위해 가장 많이 적용된 연구 결과이다.

CAN 메시지의 최악 응답 시간과 CAN 버스 이용률(Utilization)을 개선하기 위해, CAN 메시지 오버헤드를 최소화하여 메시지의 전송 시간(Transmission time)을 줄이는 방법이 있다. 여기서 데이터 전송 시 발생하는 오버헤드는 CAN 메시지 프레임에서 전송하고자 하는 데이터(그림 1의 Data field)를 제외한 나머지 부분을 의미한다. [11]에서는 CAN 메시지 프레임의 비트 삽입(Bit stuffing)을 최소화하기 위해, 비트 삽입 이전에 "010101…" 패턴의 비트 마스크(Mask)와 CAN 메시지 프레임을 XOR 연산하는 메커니즘이 제안되었다. 하지만 CAN 메시지 프레임의 Arbitration Field와 "01010 1…" 패턴의 비트 마스크를 XOR 연산하는 경우, 우선순위 역전(Priority Inversion)이 발생하는 치명적인 단점이 있다. [12]에서는 이러한 우선순위 역전을 방지함과 동시에 비트 삽입을 최소화하는 ABS(Advanced Bit Stuffing) 메커니즘이 제안되었다. 그러나 특정 비트 마스크를 이용하여 비트 삽입을 최소화하는 메커니즘은 특정 비트 패턴을 가지는 CAN 메시지 프레임에 대해서만 좋은 결과를 보인다는 단점과 해당 메커니즘을 지원하는 CAN 컨트롤러 사이에서만 작동 가능하다는 문제점이 있다.

CAN 메시지 프레임의 오버헤드를 최소화하기 위한 다른 방법으로 한 프레임에 가능한 많은 데이터를 패킹(Packing)하여 전송하는 방법이 있다. 그러나 데이터 연결 계층에서 정의 및 구현되는 CAN 메시지를 직접적으로 패킹하는 것은 각각의 수신 노드마다 수취할 메시지 ID 목록을 갱신해야 하며, 패킹된 데이터를 분리할 별도의 메커니즘이 필요하게 되므로 이는 매우 비효율적이다. 하지만 CANopen에서는 OD 내에서 텍스트의 삽입, 삭제, 수정 작업만을 이용하여 전송하고자 하는 데이터를 PDO에 효율적으로 패킹할 수 있으며, 패킹된 PDO는 데이터 연결 계층에서 자동적으로 CAN 메시지 프레임으로 변환되는 장점이 있다. 이에 본 논문에서는 CAN의 응용 계층에서 동작하는 CANopen의 OD 및 PDO 통신 서비스를 이용하는 PDO 패킹 메커니즘을 제안하였다. 본 논문의 2장에서는 시스템 모델을 통해 PDO 패킹 메커니즘을 제안하였으며, 3장에서는 제안한 메커니즘의 성능을 분석하였고, 마지막으로 4장에서는 결론을 내렸다.

2. PDO 패킹 메커니즘

PDO 패킹 메커니즘은 CANopen 네트워크 이용률 감소를 목표로 하되, 시스템이 스케줄 가능하다는 조건하에서 이루어져야 한다. 이를 위해 CAN 메시지의 최악 응답 시간(Worst-case Response Time)을 계산하여 스케줄가능성(Schedulability)을 보장해야 한다. CAN 메시지의 최악 응답 시간을 계산하기 위해 1) 모든 CAN 메시지는 주기적으로 발생되고, 2) CAN 메시지의 우선순위는 (Deadline-Jitter)-monotonic 알고리즘으로 결정되며, 3) 노드에서의 지연 시간, 즉 지터(Jitter)가 발생 가능하고, 4) CAN 메시지 전송 오류는 없다고 가정하였다[9]. CAN 메시지 m의 최악 응답 시간(R_m)은 지터(J_m), 전송 대기 시간(Queuing delay, t_m) 및 전송 시간(Transmission delay, C_m)의 합으로 계산된다[9,10](식 (1) 참조).

$$R_m = J_m + t_m + C_m \tag{1}$$

C_m은 전송하고자 하는 CAN 메시지 프레임 양식(e = 0(표준형), e = 1(확장형)), 데이터의 바이트 수(b_m) 및 1비트 전송 시간(τ_{bit})에 의해 결정되며, 비트 삽입으로 인한 추가적인 오버헤드를 최대라고 가정하면 다음 식 (2)에서 보는 바와 같이 계산된다.

$$C_m = (55 + 25e + 10b_m)\tau_{bit} \tag{2}$$

t_m은 다른 메시지가 이미 CAN 버스를 점유함에 따른 지연 시간(Blocking time, B_m)과 m 보다 높은 우선순위를 갖는 CAN 메시지(i.e. $\forall j \in hp(m)$)의 전송에 따른 간섭 시간(Interference)의 합으로 계산될 수 있다. B_m은 자신보다 낮은 우선순위를 갖는 CAN 메시지(i.e. $\forall k \in lp(m)$) 중에서 프레임의 길이가 가장 긴 경우로 계산된다(i.e. $B_m = \max_{k \in lp(m)} (C_k)$). 좌변과 우변에 모두 t_m이 존재하고, t_m은 단조 증가(Monotonically increas-

ing)하므로 재귀 관계(Recurrence relation)로 계산할 수 있다(식 (3) 참조).

$$t_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{t_m^n + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \quad (3)$$

CAN 메시지 m이 스케줄 가능하다는 것은 최악 응답 시간이 마감 시간(Deadline, D_m)보다 작거나 같음($R_m \le D_m$)을 의미하므로 전송 지연 시간, t_m은 다음의 식 (4)를 만족해야 한다.

$$t_m \le D_m - J_m - C_m \quad (4)$$

식 (4)의 범위 내에서 t_m이 수렴하는 경우(i.e. $t_m^{n+1} = t_m^n$) 또는 식 (4)의 범위를 벗어나는 경우(i.e. $t_m^{n+1} > D_m - J_m - C_m$)에는 식 (3)을 계산을 종료한다. CAN 메시지 m의 스케줄 가능성 s_m은 식 (4)의 범위 내에서 t_m의 수렴 여부에 따라 0 또는 1의 값으로 정의하였다(식 (5) 참조).

$$s_m = \begin{cases} 1 & ,if\ t_m^{n+1} = t_m^n \le D_m - J_m - C_m \\ 0 & ,if\ t_m^{n+1} > D_m - J_m - C_m \end{cases} \quad (5)$$

결국, 전체 시스템의 스케줄 가능성 s_{system}은 다음의 식 (6)에서 보는 바와 같이 모든 CAN 메시지의 스케줄 가능성을 누적 곱하여 산출할 수 있다.

$$s_{system} = \prod_{\forall m} s_m \quad (6)$$

2.1 PDO 통신 서비스

PDO 통신 서비스는 여러 개의 데이터를 하나의 PDO에 패킹하여 전송하는 기능을 제공하며 PDO는 Transmit PDO(TPDO)와 Receive PDO(RPDO)로 구분된다. CANopen에서는 최대 127개의 노드(e.g. 자동차용 ECU)를 지원하며, 각 노드에서는 최대 4개의 TPDO와 4개의 RPDO를 정의하여 사용할 수 있다. 만약 하나의 노드에 5개 이상의 TPDO 또는 RPDO가 요구되는 문제가 발생한다면, 최대 지원 가능한 노드의 수를 줄이고 각각의 PDO에 수동적으로 CAN ID를 할당하는 방법으로 해결이 가능하다. PDO 통신 서비스를 제공하기 위한 파라미터는 통신(Communication) 파라미터와 매핑(Mapping) 파라미터가 있으며, OD 인덱스 범위 1400h-1BFFh에 기록되어 있다(표 2 참조). 예를 들어 5번 노드의 TPDO 매핑 파라미터와 7번 노드의 RPDO 통신 파라미터는 OD 인덱스 1A04h와 1406h에 각각 기록되어 있다. 통신 파라미터에는 PDO의 ID를 포함한 통신 설정을 위한 값들이 기록되어 있으며, 매핑 파라미터에는 PDO에 패킹된 데이터의 정보가 기록되어 있다[8]. 결국 PDO 패킹을 수행한다는 것은 OD 내에서 PDO 매핑 파라미터를 삽입, 삭제, 수정하는 작업을 의미한다.

매핑 파라미터의 내용에 따라 PDO 패킹을 수행하는 예제(총 5개의 데이터를 1개의 프레임에 패킹하여 전송함.)는 그림 3에서 보는 바와 같다. 5번째 TPDO의 매핑 파라미터는 OD 인덱스 1A04h에 저장되어 있으며, 서브인덱스 0에는 데이터 매핑 정보의 개수(i.e. 5)가 저장되어 있고, 서브인덱스 1에서 5사이에는 5개의 데이터 매핑 정보가 기록되어 있다. 각각의 데이터 매핑 정보는 32 비트 길이이며, 그 중 상위 16 비트와 다음 8 비트는 데이터가 기록되어 있는 OD 인덱스와 서브인덱스이고, 나머지 8 비트는 데이터의 길이를 의미한다. 결국 5번째 TPDO는 OD 인덱스 2013h에서 각각 1 바이트의 'Hours', 'Minutes', 'Seconds' 데이터와 OD 인덱스 5010h에서 각각 2 바이트의 'Temperature', 'Pressure' 데이터가 패킹되어 총 7 바이트의 데이터를 전송하게 된다.

표 2 PDO 통신 서비스를 위한 파라미터의 OD 인덱스 범위

인덱스 범위	설명
1400h - 15FFh	RPDO 통신 파라미터
1600h - 17FFh	RPDO 매핑 파라미터
1800h - 19FFh	TPDO 통신 파라미터
1A00h - 1BFFh	TPDO 매핑 파라미터

2.2 시스템 모델

PDO 패킹 메커니즘을 위한 네트워크 시스템 모델은 그림 4에서 보는 바와 같이 p개의 CANopen 노드(N_1, $N_2, \cdots, N_p$)와 1개의 CAN 버스로 구성되어 있으며, 모델링 시에 각 노드의 고장(Failure)은 발생하지 않고, CAN 버스에서의 전송 오류는 없으며, 각각의 노드는 전송하고자 하는 1개 이상의 데이터를 가지고 있다고 가정하였다.

PDO 패킹을 이용하여 네트워크 이용률을 감소하기 위해서는 가능한 모든 PDO 패킹 조합(Configuration)에 대해 1) 시스템 스케줄 가능성 분석과 2) 네트워크 이용률을 계산한 다음, 시스템이 스케줄 가능하다는 조건하에서 네트워크 이용률이 최소가 되는 PDO 패킹 조합을 찾아야 한다.

2.2.1 PDO 패킹 조합의 수

n개의 데이터를 i개의 PDO에 패킹하는 조합의 수를 k_i^n라고 정의하면, 1 이상의 모든 양의 정수 n에 대해 다음의 식 (7)이 성립한다.

$$k_1^n = k_n^n = 1 \quad (7)$$

다음으로 k_2^n는 n개의 데이터 중에서 $j(1 \le j \le n-1)$개를 뽑아서 2개의 PDO 중 하나에 할당하는 경우의 수이므로 $k_2^n = \frac{1}{2}\left\{\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n-1}\right\}$이고, 정리하면 식 (8)과 같다.

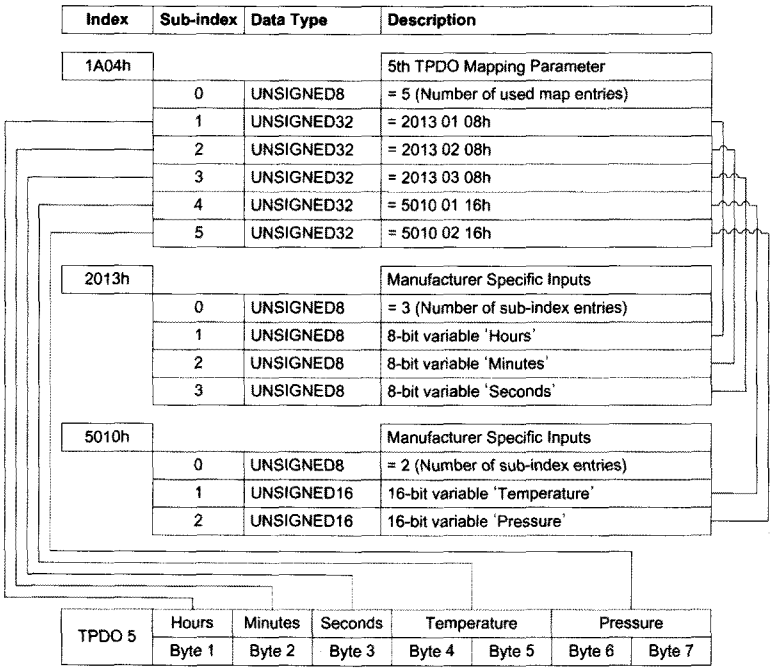

그림 3 PDO 패킹을 위한 매핑 파라미터의 설정

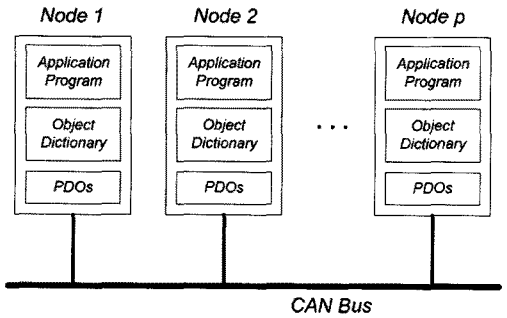

그림 4 CANopen 네트워크 시스템 모델

$$k_2^n = 2^{n-1} - 1, \quad n \geq 2 \tag{8}$$

k_3^n는 식 (8)에서 도출된 $k_2^1, k_2^2, \cdots, k_2^{n-1}$를 이용하여 계산되며, k_4^n 또한 $k_3^1, k_3^2, \cdots, k_3^{n-1}$를 이용하여 계산될 수 있다. 이를 일반화하면 다음의 식 (9)와 같다.

$$k_i^n = \frac{1}{i}\sum_{j=1}^{n-i+1}\binom{n}{j}k_{i-1}^{n-j}, \quad 2 \leq i \leq n-1 \tag{9}$$

식 (7)과 (9)에 의해 n개의 데이터를 PDO에 패킹하는 조합의 수 k^n은 다음의 식 (10)에서 보는 바와 같다. 결과적으로 전체 PDO 패킹의 조합의 수는 데이터의 수에 대한 다항식의 형태로 표현하기가 어렵다. 따라서 가능한 모든 PDO 패킹 조합에 대해 시스템 스케줄 분석과 네트워크 이용률을 계산하여 최적의 PDO 패킹 조합을 찾는 것은 NP-complete 문제이다.

$$k^n = k_1^n + k_n^n + \sum_{i=2}^{n-1} k_i^n = 2 + \sum_{i=2}^{n-1} k_i^n \tag{10}$$

2.2.2 PDO 패킹 메커니즘의 흐름도

본 논문에서 제안하는 PDO 패킹 메커니즘은 우선적으로 시스템 초기화를 수행한 다음, 더 이상 PDO 패킹을 수행할 수 없을 때까지 1) PDO 패킹 조건을 만족하는 2개의 PDO를 선택하여 2) 1개의 PDO로 패킹하는 작업을 반복한다(그림 5 참조). PDO 패킹 메커니즘을 수행하는 동안 모든 PDO는 세 가지 상태(패킹 가능 상태, 패킹 불가능 상태, 일시적 패킹 불가능 상태) 중에 하나에 있다고 정의하였다. 초기 설정으로는 모든 PDO가 패킹 가능 상태에 있으며, 모든 PDO가 패킹 불가능 상태가 되면 PDO 패킹 메커니즘이 종료된다.

• 시스템 초기화

시스템 초기 상태에는 각각의 노드(N_i)가 전송해야 하는 데이터의 수(n_i)만큼 PDO를 가지고 있으며, 이러한 PDO는 노드의 구분 없이 $m_i (1 \leq i \leq n_{all} = \sum_{i=1}^{p} n_i)$라고 정의하였다. i는 PDO의 우선순위를 의미하며 i가 작을수록 높은 우선순위를 갖는다. 즉 m_1의 우선순위가 가장 높고 $m_{n_{all}}$의 우선순위가 가장 낮다. PDO의 우선

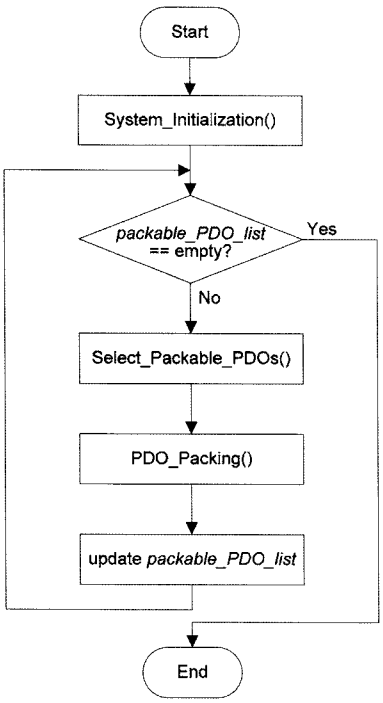

그림 5 PDO 패킹 메커니즘 흐름도

순위를 할당하기 위해 (D-J)-monotonic 알고리즘을 적용하였으며, 이는 모든 메시지의 마감 시간이 주기보다 작거나 같을 경우, (D-J)-monotonic 알고리즘을 적용하면 최적화된 우선순위 할당이 가능하기 때문이다[13]. m_i의 주기는 T_i, 마감 시간은 D_i, 데이터의 바이트 수는 b_i라고 정의하였으며, 2장에서와 마찬가지로 1) 모든 PDO는 주기적으로 발생되고, 2) 노드에서의 지연 시간이 발생 가능하며, 3) CAN 버스에서의 전송 오류는 없다고 가정하였다.

• PDO 패킹 조건

시스템 초기 상태의 PDO는 반복적인 PDO 패킹 작업을 통해 그 수를 점점 줄일 수 있으며, 이러한 PDO 패킹 작업은 다음에서 설명하는 네 가지의 조건을 만족하는 경우에만 수행된다.

- (C1) PDO 패킹은 동일 노드의 PDO만을 대상으로 한다.
- (C2) PDO에 패킹되는 데이터의 총 길이는 8 바이트 이하이어야 한다.
- (C3) 모든 PDO의 최악 응답 시간은 그것의 마감 시간보다 작거나 같아야 한다.
- (C4) PDO 패킹 이전과 비교하여 CANopen 네트워크 이용률이 감소되어야 한다.

C3 조건의 만족 여부를 검사하기 위해 전체 PDO에 대해 최악 응답 시간을 계산하는 것은 매우 비효율적이다. 이는 우선순위 i와 $j(1\le i\le j\le n_{all})$를 가지는 m_i와 m_j의 PDO 패킹 작업 이후에도 m_i 보다 높은 우선순위를 갖는 PDO의 최악 응답 시간은 계산 과정의 특성상 변함이 없으므로 검사 대상에서 제외하는 것이 효율적이기 때문이다.

C4 조건의 만족 여부를 검사하기 위해서는 패킹 이전과 이후의 네트워크 이용률을 계산하여 비교해야 한다. 특정 PDO의 네트워크 이용률은 단위 시간 동안 해당 PDO 전송을 위해 CAN 버스를 점유한 시간으로 계산되며, 예를 들어 $T_x \le T_y$인 2개의 PDO m_x와 m_y를 패킹하는 경우, 각 PDO의 데이터 바이트 수를 b_x, b_y라고 하고 전송 시간을 C_x, C_y라고 하면 패킹 이전의 네트워크 이용률(U_{prev})은 식 (11)에서 보는 바와 같다.

$$U_{prev} = \frac{C_x}{T_x} + \frac{C_y}{T_y} = \frac{\{55(T_x+T_y)+10(b_xT_y+b_yT_x)\}\tau_{bit}}{T_xT_y} \tag{11}$$

m_x와 m_y의 PDO 패킹 이후, 데이터 바이트 수는 b_x+b_y이 되고 주기는 $T_x(=\min(T_x,T_y))$이며 패킹 이후의 네트워크 이용률(U_{next})은 다음의 식 (12)에서 보는 바와 같다.

$$U_{next} = \frac{\{55+10(b_x+b_y)\}\tau_{bit}}{T_x} \tag{12}$$

PDO 패킹 이후의 네트워크 이용률 감소량을 확인하기 위해 $U_{prev} - U_{next}$를 계산하면 다음의 식 (13)에서 보는 바와 같다.

$$U_{prev} - U_{next} = \frac{\{55T_x+10b_y(T_x-T_y)\}\tau_{bit}}{T_xT_y} \tag{13}$$

• PDO 패킹 수행

PDO 패킹 메커니즘에서 패킹 작업의 대상이 되는 두 개의 PDO를 선택하고 PDO 패킹을 수행하는 절차는 그림 6에서 보는 바와 같다. PDO 패킹의 대상은 'PDO1'과 'PDO2'라고 정의하였으며, PDO1에는 패킹 가능한 PDO 중에서 가장 우선순위가 높은 것이 선택되고, PDO1과 패킹하기에 적합한 다른 PDO를 찾아 PDO2로 지정한다. PDO2를 선택하는 기준은 C1~C4 조건의 만족 여부이다.

만약 조건을 만족하는 PDO2가 없는 경우에는 PDO1을 패킹 불가능 상태로 전환한 뒤, 다른 PDO1을 선택한다. 복수 개의 PDO가 조건을 만족하는 경우에는 네트워크 이용률 감소량을 최대화하는 PDO를 선택하여 PDO2로 지정한다. 여기에서 PDO1과 같은 주기를 갖는 PDO는 그것의 데이터 바이트 수와 관계없이 네트워크 이용률 감소량이 $55\tau_{bit}/T_x$으로 최대가 되므로(식 (13)에서 $T_x = T_y$인 경우에 해당함.), 주기가 다른 PDO 보

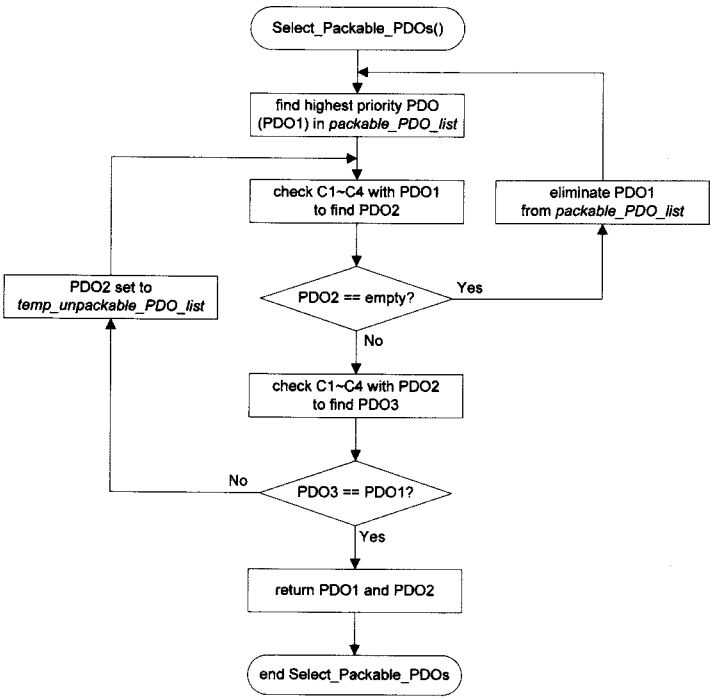

그림 6 PDO 패킹 수행 절차

다 우선권을 가진다. PDO1과 주기가 같은 PDO를 선택하는 또 다른 이유는 그림 3의 예제에서 'Hours', 'Minutes', 'Seconds'와 같이 실질적으로 동시에 사용되는 데이터를 우선적으로 패킹하기 위함이다. 한편, PDO1과 주기가 같은 PDO가 복수 개일 경우에는 우선순위가 가장 높은 PDO를 선택하고, 주기가 같은 PDO가 없는 경우에는 주기가 다른 PDO 중에서 네트워크 이용률 감소량이 큰 것을 선택한다.

PDO2는 PDO1의 입장에서 최선의 패킹 대상이지만 PDO2의 입장에서는 PDO1이 최선의 패킹 대상이 아닐 가능성이 있다. 그러므로 PDO2와 패킹하기에 적합한 PDO를 찾아 'PDO3'이라고 지정한 다음, 만약 PDO3과 PDO1이 같다면 PDO 패킹을 수행하고, 그렇지 않다면 PDO2를 일시적 패킹 불가능 상태로 전환한 뒤, 다른 PDO2를 찾는다. 일시적 패킹 불가능 상태는 PDO1이 이미 선택된 PDO2를 제외하고 다른 PDO2를 찾기 위해 정의되었다.

2.3 PDO 패킹 메커니즘의 구현

PDO 패킹 메커니즘은 크게 오프라인(Off-line)과 온라인(On-line) 용도로 각각 구분되어 구현될 수 있다. 오프라인 PDO 패킹 메커니즘은 시스템 설계 국면(Design phase)에서 동작하기 때문에 구현이 용이하다는 장점이 있는 반면, 온라인 PDO 패킹 메커니즘은 시스템 운영 국면(Operation phase)에서 동작하기 때문에 네트워크 유연성이 좋다는 장점이 있다.

오프라인 PDO 패킹 메커니즘은 시스템에서 요구되는 데이터의 전송 특성(e.g. 지터, 주기, 마감시간)을 이용하여 PDO 패킹 작업을 수행하며, 3.1에서 설명한 바와 같이 각각의 노드가 가지고 있는 OD 내에서 PDO 매핑 파라미터를 삽입, 삭제 및 수정하는 작업을 통해 구현되며, 이는 결국 EDS 또는 DCF를 자동적으로 생성, 삭제 및 수정하는 것을 의미한다.

온라인 PDO 패킹 메커니즘은 CANopen 네트워크 시스템 내에 새로운 노드 또는 데이터가 추가될 경우에 마스터(Master) 노드에서 자동적으로 수행되며, 이때 마스터 노드를 제외한 모든 노드는 NMT(Network Management) 메시지에 의해 그림 7에서 보는 바와 같

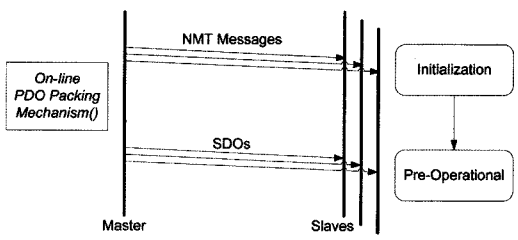

그림 7 온라인 PDO 패킹 메커니즘의 수행 절차

이 초기화(Initialization) 상태로 전환된다. 이후 각 노드는 운영 이전(Pre-operational) 상태로 전환되면서, 마스터 노드로부터 수신된 SDO를 통해 PDO 패킹 메커니즘의 결과(EDS 또는 DCF)를 수신한다.

3. 성능 평가

본 논문에서 제안하는 PDO 패킹 매커니즘의 성능을 평가하기 위해 SAE(The Society of Automotive Engineers)에서 제공하는 벤치마크(Benchmark) 자료[9,10]를 이용하였다. 벤치마크 자료에는 7개의 노드에서 전송해야 할 총 53개 데이터의 길이, 지터, 주기, 마감시간 등을 제공한다(표 3 참조).

[7,8]에서는 'piggyback' 방법을 적용하여 53개 데이터를 17개의 CAN 메시지로 패킹하였고, 125Kbit/s의 대역폭에서의 최악 응답 시간은 식 (1)을 이용하여 계산되었으며, 모든 CAN 메시지의 최악 응답 시간은 각각의 마감 시간을 모두 만족하였고, CAN 버스 이용률은 84.44% $\left(=\sum_{i=1}^{17}\frac{C_i}{T_i}\right)$이다(표 4 참조).

표 4의 둘째 행에서 8, 9번 데이터의 지터는 각각 0.1ms과 0.2ms이며, 패킹 이후에는 작은 값을 취하게 된다. 또한 23, 24, 25, 28번 데이터의 행을 보면 각 데이터는 바이트 단위가 아니라 비트 단위로 패킹되었다는 것을 알 수 있다. 이는 PDO 패킹 메커니즘과는 다르게 데이터 연결 계층에서 별도의 메커니즘이 필요하다는 것을 의미한다.

본 논문에서 제안하는 PDO 패킹 메커니즘을 이용하여 최종적으로 패킹된 PDO는 총 17개이며, 125Kbit/s의 대역폭에서 모든 PDO의 최악 응답 시간은 수식 (1)을 이용하여 계산되었으며, 각각의 마감 시간을 만족하였다(표 5 참조). PDO 패킹 메커니즘의 CAN 버스 이용률은 74.54%로써, 'piggy-back'을 이용한 선행 연구에 비해 약 10% 가량 감소되었다. 그림 8은 표 5의 CAN 메시지와 표 5의 패킹된 PDO의 최악 응답 시간을 비교한 그래프이며, 본 논문에서 제안하는 PDO 패킹 메커니즘의 최악 응답 시간이 선행 연구에 비해 대체적으로 비슷하거나 작은 값을 가지는 것을 확인할 수 있다.

표 3 The SAE Benchmark

#	Size (bits)	Jitter (ms)	Period (ms)	Deadline (ms)	Node	#	Size (bits)	Jitter (ms)	Period (ms)	Deadline (ms)	Node
1	8	0.6	100	100	Battery	28	1	0.5	50	20	Battery
2	8	0.7	100	100	Battery	29	8	0.3	10	10	V/C
3	8	1	1000	1000	Battery	30	8	0.4	10	10	V/C
4	8	0.8	100	100	Battery	31	2	0.5	50	20	V/C
5	8	1.1	1000	1000	Battery	32	8	0.1	5	5	V/C
6	8	0.9	100	100	Battery	33	1	1.6	1000	1000	V/C
7	8	0.1	5	5	Driver	34	8	0.6	50	20	V/C
8	8	0.1	5	5	Brakes	35	1	0.7	50	20	V/C
9	8	0.2	5	5	Brakes	36	2	1.7	1000	1000	V/C
10	8	0.2	100	100	Trans	37	1	0.8	50	20	V/C
11	8	0.1	5	5	Trans	38	1	0.9	50	20	V/C
12	8	0.4	100	100	Brakes	39	7	1	50	20	V/C
13	1	1.2	1000	1000	Battery	40	1	1.1	50	20	V/C
14	4	0.1	50	5	Battery	41	1	0.3	50	20	I/M C
15	1	0.2	50	20	Driver	42	8	0.2	5	5	V/C
16	1	0.3	50	20	Driver	43	8	0.1	5	5	I/M C
17	2	0.4	50	20	Driver	44	1	1.2	50	20	V/C
18	1	0.3	20	20	Brakes	45	1	0.4	50	20	I/M C
19	1	0.5	50	20	Driver	46	1	1.3	50	20	V/C
20	3	0.6	50	20	Driver	47	1	0.5	50	20	I/M C
21	2	0.3	1000	1000	Trans	48	1	1.4	50	20	V/C
22	3	0.7	50	20	Driver	49	8	0.2	5	5	I/M C
23	1	0.2	50	20	Battery	50	2	0.6	50	20	I/M C
24	1	0.3	50	20	Battery	51	1	0.7	50	20	I/M C
25	1	0.4	50	20	Battery	52	8	0.8	50	20	I/M C
26	1	0.8	50	20	Driver	53	1	1.5	50	20	V/C
27	1	0.9	50	20	Driver						

표 4 CAN 메시지의 최악 응답 시간 분석 [9,10]

Number(s) of packed data	Size (byte)	Jitter (ms)	Period (ms)	Deadline (ms)	WCRT (ms)	Node
14	1	0.1	50	5	1.544	Battery
8,9	2	0.1	5	5	2.128	Brakes
7	1	0.1	5	5	2.632	Driver
43,49	2	0.1	5	5	3.216	I/M C
11	1	0.1	5	5	3.720	Trans
32,41	2	0.1	5	5	4.304	V/C
31,34,35,37,38,39,40,44,46,48,53	6	0.2	10	10	5.192	V/C
23,24,25,28	1	0.2	10	10	8.456	Battery
15,16,17,19,20,22,26,27	2	0.2	10	10	9.040	Driver
41,43,45,47,49,50,51,52	3	0.2	10	10	9.696	I/M C
18	1	0.2	50	20	10.128	Brakes
1,2,4,6	4	0.3	100	100	19.088	Battery
12	1	0.3	100	100	19.592	Brakes
10	1	0.2	100	100	20.096	Trans
3,5,13	3	0.4	1000	1000	28.904	Battery
21	1	0.3	1000	1000	29.408	Trans
33,36	1	0.3	1000	1000	29.912	V/C

표 5 PDO를 활용한 최악 응답 시간 분석

Number(s) of packed data	Size (byte)	Jitter (ms)	Period (ms)	Deadline (ms)	WCRT (ms)	Node
14	1	0.1	50	5	1.563	Battery
8,9	2	0.1	5	5	1.641	Brakes
7	1	0.1	5	5	2.070	Driver
43,49	2	0.1	5	5	2.734	I/M C
11	1	0.1	5	5	3.164	Trans
29,30,32,42	4	0.1	5	5	3.984	V/C
37,38,39,40,44,46,48,53	8	0.8	50	20	4.805	V/C
41,45,47,50,51,52	6	0.8	50	20	5.391	I/M C
31,34,35	3	0.5	50	20	9.141	V/C
18	1	0.3	20	20	9.883	Brakes
15,16,17,19,20,22,26,27	8	0.2	50	20	14.023	Driver
1,2,4,6,23,24,25,28	8	0.2	50	20	14.141	Battery
12	1	0.4	100	100	14.648	Brakes
10	1	0.2	100	100	18.633	Trans
33,36	2	1.6	1000	1000	19.219	V/C
3,5,13	3	1	1000	1000	19.648	Battery
21	1	0.3	1000	1000	20.000	Trans

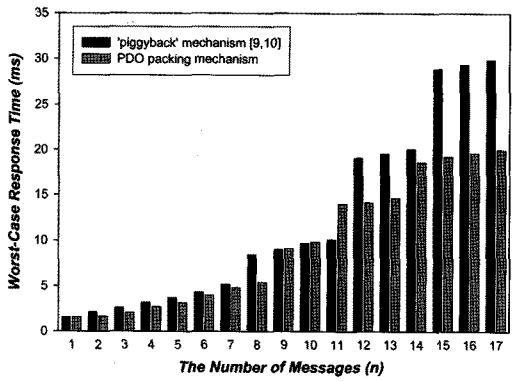

그림 8 'piggyback' 메커니즘과 PDO 패킹 메커니즘의 최악 응답 시간의 비교

4. 결 론

CAN 메시지의 최악 응답 시간과 네트워크 이용률을 감소하는 것은 시스템의 실시간성을 보장하기 위해 반드시 필요하다. 본 논문에서는 CAN의 응용 계층에서 동작하는 CANopen 프로토콜에서 제공하는 OD와 PDO 통신 서비스를 이용하여 데이터 전송을 위한 오버헤드를 효과적으로 줄일 수 있는 PDO 패킹 메커니즘을 제안하였다. 제안한 메커니즘의 성능 분석을 위해 SAE에서 제공하는 벤치마크 자료를 이용하였으며, 결과적으로 CANopen 네트워크 이용률이 약 10% 가량 감소되는 것을 확인하였다. 본 논문에서 제안하는 PDO 패킹 메커니즘은 기존의 CAN 메시지를 패킹하기 위해 데이터 연결 계층에서 사용되었던 별도의 메커니즘 없이 효율

적으로 CANopen 네트워크 이용률을 감소시킬 수 있었다. 향후에는 CAN의 비결정적인 응답 시간을 보장하기 위해 CANopen의 OD 및 SDO 서비스를 이용한 동적 ID 할당 메커니즘에 대한 연구를 수행할 예정이다.

참 고 문 헌

[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] K. Etschberger, *Controller Area Network (CAN) - Basics, Protocols, Chips, and Applications*, IXXAT Press, pp. 43-75, 2001.

[3] G. Cena, A. Valenzano, and S. Vitturi, "Advanced in Automotive Digital Communications," *Computer Standards and Interfaces*, Vol.27, Issue 6, pp. 665-678, June 2005.

[4] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems," *Proceeding of the IEEE*, Vol.93, Issue 6, pp. 1204-1223, June 2005.

[5] K. Anwar and Z. A. Khan, "Dynamic Priority Based Message Scheduling on Controller Area Netowk," International Conference on Electrical Engineering (IECC'07), pp. 1-6, Apr. 2007.

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

[7] M. Farsi, K. Ratchiff, and M. Barbosa, "An Introduction to CANopen," *Computing & Control Engineering Journal*, Vol.10, No.4, pp. 161-168, June 1999.

[8] O. Pfeiffer, A. Ayre, and C. Keydel, *Embedded Networking with CAN and CANopen*, RTC Books, Nov. 2003.

[9] K. 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.

[10] K. Tindell and A. Burns, "Guaranteed Message Latencies for Distributed Safety-critical Hard Real-time Networks," *Technical Report YCS 229*, Department of Computer Science, University of York, May 1994.

[11] 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.

[12] -, "Mechanism for Minimizing Stuffing-bit in CAN Messages," *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON '07)*, pp. 735-737, Nov. 2007.

[13] A. Zuhily, "Optimality of (D-J)-monotonic Priority Assignment," Technical Report YCS404, University of York, May 2006.

강 민 구

2007년 아주대학교 산업정보시스템공학부(공학사). 2007년~현재 아주대학교 일반대학원 산업공학과(석사과정). 관심분야는 Embedded System, Fault Tolerant Computing, In-Vehicle Network

박 기 진

1989년 한양대학교 산업공학과(공학사) 1991년 POSTECH 산업공학과(공학석사). 2001년 아주대학교 컴퓨터공학과(공학박사). 1991년~1997년 삼성종합기술원/삼성전자 선임연구원. 2001년~2002년 한국전자통신연구원 선임연구원. 2002년~2004년 안양대학교 컴퓨터학과 전임강사. 2004년~현재 아주대학교 산업정보시스템공학부 조교수/부교수. 관심분야는 Real-time Distributed Embedded Systems, Dependable Embedded Computing, Safety-critical Computing, In-Vehicle Network

김 종 철

2008년 아주대학교 산업정보시스템공학부(공학사). 2008년~현재 아주대학교 일반대학원 산업공학과(석사과정). 관심분야는 In-Vehicle Network, Real-time Distributed Embedded Systems