CAN기반 실시간 시스템을 위한 확장된 EDS 알고리즘 개발

論 文
51D-7-3

Development of an Extended EDS(Earliest Deadline Scheduling) Algorithm for the CAN-Based Real-Time System

李 炳 勳* · 金 弘 烈** · 金 大 元***
(Byong Hoon Lee · Hong Ryeol Kim · Dae Won Kim)

Abstract - A new dynamic scheduling algorithm is proposed for CAN-based real-time system in this paper. The proposed algorithm is extended from an existing EDS(Earliest Deadline Scheduling) approach having a solution to the priority inversion. Using the proposed algorithm, the available bandwidth of network media can be checked dynamically, and consequently arbitration delay causing the miss of deadline can be avoided. Also, non-real time messages can be processed with their bandwidth allocation. Full network utilization and real-time transmission feasibility can be achieved through the algorithm. To evaluate the performance of algorithm, two simulation tests are performed. The first one is transmission data measurement per minute for periodic messages and the second one is feasibility in the system with both periodic messages and non-real time message.

Key Words : CAN(Control Area Network), EDS(Earliest Deadline Scheduling), network utilization, dynamic scheduling, arbitration delay, non-real time message

1. 서 론

실시간 필드버스 네트워크 프로토콜인 CAN(Controller Area Network)의 스케쥴링은 일반적으로 구현의 용이성으로 인해 DMS (Deadline Monotonic Scheduling) 혹은 RMS(Rate Monotonic Scheduling)와 같은 고정된 식별자 인코딩(static identifier encoding)방식을 사용한다[1]][2]. 그러나, 이러한 정적인 스케쥴링 알고리즘을 사용했을 경우의 단점은 네트워크 이용율이 낮아진다는 것이다.

이러한 문제점을 극복하기 위해서는 메시지 발생시점에서 동적으로 데드라인을 인코딩하고 식별자를 지정하는 동적 스케쥴링 알고리즘이 필요하다. 이러한 요구를 만족시키기 위해 낮은 우선순위를 갖는 메시지가 우선권을 갖는 동적인 스케쥴링(dynamic scheduling algorithm)인 EDS(Earliest Deadline Scheduling)알고리즘[9]이 제안되었고, CAN에서의 EDS에 관한 연구도 진행되어 왔다[3][4][5]. CAN에서 EDS 알고리즘을 구현하기 위해서는 다음과 같은 두 가지 사항이 고려되어야 한다.

첫째, 한정된 CAN의 식별자 영역에 데드라인을 인코딩 하여 메시지의 우선순위를 나타내므로 메시지의 우선순위가 부정확해지고 결과적으로 우선순위 반전(priority inversion)이 나타날 수 있다. 이러한 우선순위 반전을 방지하기 위해 데드라인을 로그스케일(log scale)에 사상하는 알고리즘이 제안된 바 있다[4][5].

둘째, CAN 프로토콜의 메시지 충돌 회피 매카니즘에 의해 하나의 메시지가 네트워크를 점유했을 경우에는 다른 메시지는 전송을 시도할 수 없고 결과적으로 메시지 전송 시도를 위한 중재(arbitration)가 지연될 수 있다. 이러한 지연시간(delay time)은 최악의 경우 메시지의 실시간성을 보장할 수 없게 되는 요인이 되어 네트워크 이용율을 저하시킨다.

본 논문에서는 상기의 지연시간으로 인한 네트워크 이용율의 저하를 방지하기 위해 확장된 EDS알고리즘(이후 EEDS라 칭함)을 제안한다. 제안된 알고리즘은 네트워크상의 모든 노드들이 네트워크의 이용 가능한 대역폭을 동적으로 확인할 수 있게 하며, 대역폭 정보를 가지고 실시간성에 영향을 줄 수 있는 중재 지연시간을 메시지의 데이터를 동적으로 분할 전송하여 사전에 방지할 수 있도록 한다. 또한, 사용하지 않는 대역폭 즉, 휴지구간을 알 수 있으므로 적절한 대역폭 할당과 중재시점의 지연을 통해 실시간성에 영향을 주지 않는 범위내에서 비실시간 메시지의 처리도 가능하도록 한다. 이러한 동적 대역폭 예측정보는 단일 프로세서 시스템에서 중앙 스케쥴러에 의해 각 프로세스들을 관리하는 동작과 같다. 따라서, 중앙 스케쥴러에서 비실시간 프로세스의 처리를 위해 공유 대역폭을 활용하는 서버알고리즘[6]의 응용이 가능하다.

본 논문의 2장에서는 CAN프로토콜의 정의 및 스케쥴링과 단일 프로세서 시스템환경에서의 비실시간 메시지 처리를 위한 서버 알고리즘 그리고, CAN의 동적 스케쥴링 알고리즘 구현시 지연시간 등을 설명한다. 그리고, 3장에서는 지연시간 최소화와 비실시간 메시지를 처리할 수 있는 EEDS 알고리즘을 제안하며, 제안된 알고리즘에 대해 4장에서 모의실험 및 결과를 제시한다. 마지막으로 5장에서 결론을 맺는다.

* 準 會 員 : 明知大學 情報制御工學科 碩士課程
** 正 會 員 : 明知大學 情報制御工學科 博士課程
*** 正 會 員 : 明知大學 情報工學科 教授
接受日字 : 2001年 12月 26日
最終完了 : 2002年 4月 18日

2. CAN프로토콜 정의 및 스케쥴링

2.1 CAN의 메시지 구조 및 스케쥴링

CAN은 분산된 실시간 제어시스템에 시리얼 버스로 연결되어 구성된다. 버스는 논리적 채널 특성(wired OR 또는 wired AND)을 가지며, 버스를 통한 메시지 전송 방식은 정보를 교환하는 데이터 통신에서 한 순간에 두 개 이상의 장치에 의하여 회선으로 자료를 전송하는 교류 경쟁(alternating contention)을 하며, 주기적으로 반복되는 파형에서, 한 순간에 파형의 위치를 기준 위치와 비교해 보았을 때의 위상에 대한 상대 위치로 정보를 표현하는 전송 위상(transmission phase)의 특성을 지닌다. CAN은 20kbit/s에서 1Mbit/s까지의 다양한 전송속도를 제공하며, 물리(physical)계층, 트랜스퍼계층, 오프젝트(object)계층의 3계층으로 구성된다[7][8].

그림 1은 CAN 프로토콜의 데이터 프레임을 보여준다. 식별자 필드(identifier field)는 중재 필드(arbitration field)라고 표준 식별자 필드와 29bits의 확장 식별자 필드(extended identifier field)가 있다. 모든 전송 메시지들은 고유한 식별자(unique identifier)를 가져야 한다. 이것은 메시지의 식별자 필드 전체를 메시지의 우선순위 또는 데드라인을 나타내는데 사용할 수 없다는 것이다. 각 노드는 전송된 메시지의 식별자 필드를 메시지 필터(message filters)를 통해 필요한 정보인지를 판별하여 필요한 정보만을 취한다.

그 외 콘트롤 필드(control field)는 메시지의 타입의 정보를 포함하며, 데이터 필드는(data field) 실제 전송할 데이터를 포함하며, 체크섬(chedk sum)은 메시지 비트수를 체크하기 위해 메시지비트수를 포함하며, ACK(acknowledge)는 수신 승인에 대한 정보를 도 하며 메시지의 우선순위의 인코딩 값이 들어가며 11bits의 포함하며, ED(end delimiter)는 메시지의 끝을, IS(idle space)는 다음에 버스를 점유할 메시지와 구별하기 위한 비트이다.

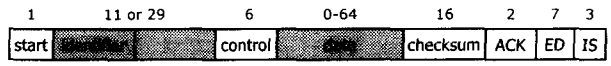


그림 1 AN 데이터 프레임
Fig. 1 Data frame of the CAN

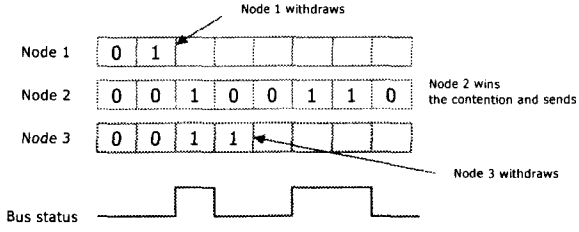


그림 2 CAN에서의 버스점유를 위한 중재의 예
Fig. 2 An example of bus arbitration in the CAN

그림 2는 AND연산의 논리적 특성을 가진 버스에서 메시지의 버스점유방식의 한 예를 나타낸다. 각 노드는 메시지 전송이전에 네트워크의 상태를 감지하여 네트워크에 전송중인 메시지가 없을 때 전송을 시도하는데, 두 개 이상의 노드가 동시에 메시지를 전송하면 각 메시지는 서로 식별자(identifier)를 1비트씩 비교한다. 제일 높은 우선순위의 메시지 즉, 가장 작은 수의 식별자 값을 가진 메시지는 전송되고 낮은 우선순위의 메시지들은 전송을 중단한다. 그림 2에서는 두 번째 노드에서 발생한 메시지의 우선순위 값이 제일 작으므로 버스를 통해 전송되고 첫 번째, 세 번째 메시지는 중재 대기 상태가 된다. 그리고, 버스를 점유하지 못한 각 노드의 메시지는 버스를 점유한 메시지가 끝나기를 기다린 후 다시 버스 점유를 시도한다.

2.2 비실시간 메시지 스케쥴링

비실시간 프로세스처리를 위해 단일프로세서 시스템에서는 서버알고리즘이 제안되었다[6]. 서버알고리즘은 비실시간 프로세스 처리를 위해 일정량에 대역폭을 비실시간 프로세스들이 공유하여 사용하며, 비실시간 프로세스의 빠른 처리가 가능하다. 서버알고리즘의 CAN에 적용은 비실시간 메시지에 대한 공유 대역폭을 사용함으로써 네트워크 이용율의 향상을 도모할 수 있다.

서버알고리즘을 실시간 네트워크 시스템인 CAN에 적용하기 위해서는 다음 두 가지 사항이 고려되어야 한다.

- 실시간 단일 프로세서 시스템의 운영환경에서는 스케쥴러가 중앙에서 모든 프로세스의 스케쥴링을 관리하기 때문에 적당한 대역폭의 할당이 가능하나, CAN에서는 중앙 스케쥴러가 존재하지 않으므로 네트워크 대역폭의 관한 정보가 메시지를 발생시키는 모든 노드에 공유되어야 한다.
- 비실시간 메시지의 대역폭 할당시에는 CAN의 충돌 방지 메카니즘에 의한 비선점(non-preemptive) 특성이 고려되어야 한다.

이러한 이유로 본 논문에서는 확장된 메시지 포맷의 식별자 필드(ID field)를 사용하여 메시지의 종류(실시간 주기, 비실시간 비주기 등)와 우선순위 그리고, 대역폭의 정보를 표현한다. 이러한 정보는 CAN 물리계층의 특성상 모든 노드에 브로드캐스팅(broadcasting)되므로 항상 각 노드는 최신의 정보를 유지할 수 있다.

2.3 CAN의 동적 스케쥴링 알고리즘 구현시 지연시간

기존의 연구[3]에서는 CAN네트워크에 동적 알고리즘인 EDS[9]를 사용하여 RMS와 DMS에 비해 버스의 이용율이 향상됨을 증명하였다. CAN에서 동적 스케쥴링 알고리즘을 구현할 경우 발생하는 지연시간은 데드라인 인코딩에 의한 지연시간과 메시지 중재에 의한 지연시간으로 구분된다.

CAN에 EDS적용시 일반적인 메시지 M_i의 실시간성이 보장되려면 식(1)을 만족해야 한다. M_i가 t=0일 때 발생하면, 전송시간 C_i, 높은 우선순위를 가진 메시지에 의한 전송 지연시간 I_i, 부정확한 데드라인 인코딩에 의한 지연시간 BM_i 그리고, 중재 지연시간인 BNP_i의 합인 Δ_i가 메시지 M_i의데드라인인 D_i보다 작거나 같아야 한다[3].

$$\Delta_i = C_i + I_i + BM_i + BNP_i \le D_i \quad (1)$$

다음은 동적 스케쥴링에서 필요한 정의이다.

정의 2-1.

- 주기메시지 M_i는 전송시간 C_i, 주기 T_i, 데드라인 D_i을 갖는다.
- 메시지 M_i의 D_i는 T_i와 같다.
- $d_i = k\,T_i + D_i (k=1,2,3,\ldots)$ 여기서 d_i는 모든 메시지의 발생 시점(t=0)부터의 데드라인 값을 나타낸다.
- 네트워크 이용율 U_i는 $\frac{C_i}{T_i}$ 이다.
- 전체 노드에서 발생하는 메시지의 총 네트워크 이용율은 100%를 넘어서는 안된다.

$$U_{TOTAL} = \frac{B_{current}}{P_{current}} \le 1$$

- 여기서 $P_{current}$는 kP에서 t까지 발생한 전체 메시지 주기의 최소공배수 이다.
- $B_{current}$는 kP에서 t까지 발생한 전체 메시지의 주기인 $P_{current}$에서 전송시간으로 사용될 것으로 예상되는 총 네트워크 점유시간이다.

다음은 본 장에서 사용하는 가정이다.

가정 2-1.

- $t_{release}$는 메시지 발생 시점이다.
- t_{START}는 버스점유를 위한 중재에 참여하는 시점이다.

2.3.1 데드라인 인코딩에 의한 지연시간

CAN에서 EDS알고리즘의 사용은 중재시도전에 동적으로 전송메시지의 데드라인을 우선 순위로 정하여 메시지의 식별자 필드에 표현한 후에 중재 시도를 한다. 하지만, 식별자 필드의 제한된 크기 때문에 데드라인의 값을 정확히 표현할 수 없으므로, 시공간을 나누어 데드라인이 들어있는 시분할영역의 주소값을 메시지의 우선순위로 결정하는 데드라인 양자화(deadline quantization) 알고리즘을 사용한다[5].

데드라인 양자화 방법에 의해 중재시도 메시지는 t_{START}에서 버스를 사용하는 메시지의 최대 주기까지 균등하게 시분할(time section) $\{I_0, I_1, \ldots, I_N\}$을 하여 우선순위를 정한다.

이러한 데드라인 양자화에 의한 우선순위 결정은 모든 메시지에 대해 정확한 우선순위를 정하지 못하므로 우선순위 반전에 의한 지연시간이 발생될 수 있다.

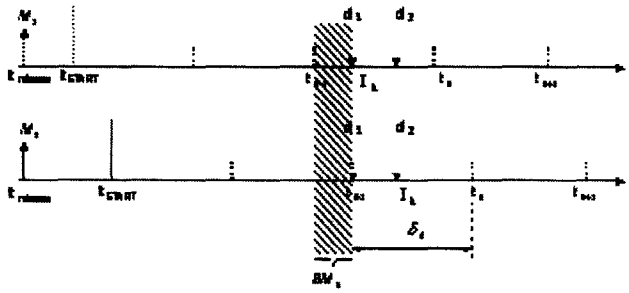


그림 3 메시지 M1과 M2사이의 메시지 우선순위 반전

Fig. 3 Priority inversion between message M1 and message M2

그림 3은 데드라인 양자화로 인한 우선순위의 불확실성으로 발생할 수 있는 지연시간 BM_i를 보여준다. M_1과 M_2는 같은 시점에 발생했으나 지연시간(I_i, BNP_i)에 의해 중재 참여가 지연되어 t_{START}시점이 서로 다르다. M_1과 M_2가 발생한 각 노드는 t_{START}부터 D_{max}(메시지들 중 가장 큰 데드라인)까지 시분할하고 데드라인이 들어있는 시분할 영역의 번지 값을 우선순위로 결정한다. 그림 3에서의 M_1과 M_2의 우선순위는 데드라인이 시분할영역 I_k부분에 존재하므로 같은 우선순위(k)를 가진다. M_2는 실제 데드라인이 M_1보다 늦지만 같은 우선순위를 가진다. 따라서, M_2는 실제 데드라인이 M_1보다 늦지만 먼저 전송 될 수 있는 우선순위 반전에 의한 지연시간(BNP_i)이 발생한다.

균등한 시분할에 의한 데드라인 양자화는 중재시도 시점과 데드라인이 가까울 때 지연시간이 발생하면, 주기적 메시지의 실시간성을 보장할 수 없는 경우가 발생할 수 있다.

이러한 지연시간을 최소화하기 위해 로그스케일(log scale) 사상 방법이 제안되었다. 로그스케일 사상 방법은 시공간을 균등 분할하는 것이 아니라 로그(log)값으로 시공간을 분할하여 중재시도 시점이 데드라인에 가까울수록 시분할 영역이 적게 설정되는 알고리즘이다[3]. 본 논문에서는 로그스케일 사상 방법을 적용하여 데드라인 양자화에 의한 지연시간을 최소화한다.

2.3.2 메시지 중재에 의한 지연시간

CAN에서 EDS를 사용하면 휴지구간에는 우선순위가 낮은 메시지에 대해 동적으로 버스의 점유권이 주어지기 때문에 낮은 우선순위를 가진 메시지의 전송에 의해서 높은 우선순위 메시지의 중재시도가 지연되고, 결과적으로 전송지연시간이 발생한다.

그림 4는 총 네트워크 이용율 100%이하로 설정한 5개의 메시지가 버스를 점유하는 순서 시간 축에 각 메시지의 전송시간으로 보여주는 그림이다. 그림 4에서 다섯 번째 노드의 메시지가 휴지구간인 36에서 중재 시도에 성공하여 메시지 M_5를 전송한다. M_5의 전송이 끝나기 전에 M_1이 40에서 발생하지만 CAN의 비선점 특성에 의해 M_5의 전송이 완료될 때까지 M_1의 중재시도는 BNP_1만큼 지연된다. 또한, 지연시간 BNP_1으로 인해 다른 메시지의 중재시도도 지연되어 진다. 그 결과 M_4의 두 번째 메시지는 80인 시점에서 메시지의 데드라인 안에 전송되지 못하므로 실시간 보장을 만족시키지 못한다.

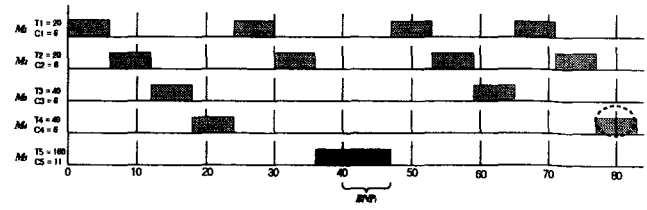


그림 4 낮은 우선순위 메시지에 의한 실시간 보장의 깨짐현상

Fig. 4 Non-schedulability example caused by bus occupancy of lower priority message

지연시간 BNP에 의해 실시간 보장이 깨지는 것을 막기 위해서는 BNP를 발생시키는 메시지의 전송시간을 줄임으로써 다른 메시지의 전송시점을 앞당기는 것이다. 그림 4의 경우 M_5의 전송시간을 동적으로 줄임으로써 실시간 보장이 만족 될 수 있다. 메시지의 동적 분할전송을 위해서 각 노드는 모든 메시지의 사용한 대역폭과 사용할 대역폭 정보를 알 수 있어야 하며, 실시간 보장이 깨지는 것을 앞서 계산할 수 있어야 한다.

3. EEDS(Extended EDS) 알고리즘

본 장에서는 CAN에서의 EDS적용시에 발생하는 지연시간인 BM_i와 BNP_i를 최소화하면서, EDS에서 고려되 못했던 비실시간 메시지(M_5) 처리를 위한 알고리즘을 제안한다. 다음은 본 장에서 사용하는 정의이다.

정의 3-1.

- P는 전체 메시지 주기의 최소공배수이다.
- t는 $kP \le t \le (k+1)P$를 만족하는 메시지 발생 시점이며, $k = 0, 1, 2, \ldots$이다.
- $P_{current}$는 kP에서 t까지 발생한 전체 메시지 주기의 최소공배수 이다.
- $B_{current}$는 kP에서 t까지 발생한 전체 메시지의 주기인 $P_{current}$에서 전송시간으로 사용될 것으로 예상되는 총 네트워크 점유시간이다.
- M_i의 주기 T_i와 데드라인 D_i는 2^i로 설정한다. 즉, 2의 멱(power)으로 설정한다.

다음은 본 장에서 사용한는 가정이다.

가정 3-1.

- CAN의 전송속도는 125Kbit/s로 가정한다.
- 본 알고리즘에서는 확장된 식별자 필드를 사용한다. 이때 데이터 필드가 8byte, 4byte, 1byte의 전송시간(C_i)은 올림(worst case)한 값인 0.6ms, 0.8ms, 1.1ms에 10을 곱하여 6, 8, 11로 표현한다.

3.1 CAN의 확장된 ID필드의 정의

비실시간 메시지의 처리와 메시지 중재에 의한 지연시간의 최소화를 위해서는 각 메시지 발생 노드는 네트워크의 사용 대역폭을 서로 공유해야 한다. 따라서 각 노드에 필요한 정보를 갱신하기 위해서 확장 식별자 필드(extended ID field)를 사용한다. 그림 5는 확장 식별자 필드의 구성을 보여준다.

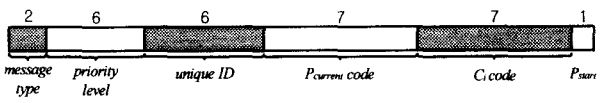


그림 5 확장 식별자 필드의 구성

Fig. 5 Extended ID field structure

확장 식별자 필드는 6가지 정보를 갖는다. M_{sort}는 메시지의 종류를 나타낸다. M_{sort}의 값이 0이면 주기메시지를 나타내며, 1은 비실시간 메시지를 나타낸다. '*priority level*'은 버스점유 우선순위를 나타내고, '*unipue ID*'는 메시지를 구별하는 유일한 비트정보이다. $P_{current}$는 2의 멱(power)으로 표현되므로, '$P_{current\ code}$'는 멱을 나타낸다. '$C_{i\ code}$'는 메시지 전송시간을 나타내기 위한 비트 정보이다. 메시지의 전송시간(C_i)은 데이터 필드의 바이트크기에 따라 8가지로 나뉜다. 따라서, '$C_{i\ code}$'의 값은 1~8의 값을 가진다. 'P_{start}'는 $kP+t$ ($t = 0$)일 때 중재에 나선 메시지는 비트1을 가지고, 그 후의 메시지는 비트 0을 갖는다.

CAN 물리계층의 특성상 전송데이터는 모든 노드에 브로드캐스팅되므로 항상 각 노드는 최신의 정보를 유지할 수 있게 된다.

3.2 지연시간의 최소화를 위한 알고리즘

3.2.1 로그스케일 사상에 의한 지연시간 최소화

BM_i를 최소화하기 위한 방법은 기존의 연구 결과[3]인 데드라인을 로그스케일로 사상하여 메시지의 우선순위를 정하는 데드라인 인코딩 방식을 사용하며, 그 방식은 식(2), 식(3)과 같다. 이 방식을 이용하면 데드라인에 t_{STRART}가 가까워질수록 우선순위 반전을 줄일 수 있다.

$$h = \begin{cases} k & \text{if } d_{rel_i} = D_{\max} \\ k + \left\lfloor \log_2 \dfrac{d_{rel_i}}{D_{\max}} \right\rfloor & \text{if } \dfrac{D_{\max}}{2^k} \le d_{rel_i} \le D_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$d_{rel_i} = d_i - t_{START}$$

$$p = hq + \left\lfloor \frac{d_i - t_{START} - \dfrac{D_{\max}}{2^{k-h}}}{q^{\frac{D_{\max}}{2^{k-h}}}} \right\rfloor \tag{3}$$

여기서, k는 시분할개수이며, 각 시분할은 같은 크기의 q로 나뉜다. 따라서 총 우선순위는 $p \times k$개이다. D_{max}는 모든 주기적 메시지에서 가장 큰 데드라인이다. p는 실제 식별자 필드에 들어갈 메시지의 우선순위 값이다.

3.2.2 메시지 중재에 의한 지연시간 최소화

본 논문에서 제안하는 BNP_i 최소화 알고리즘은 메시지를 전송하고자 하는 노드에서 더 높은 우선순위를 가진 메시지가 실시간성 보장이 가능한지를 검사하여 그림 4의 예와 같이 M_5로 인해 실시간 보장이 되지 않을 것으로 예상될 경우에는 M_5발생노드는 데드라인이 초과되지 않는 범위에서 메시지를 분할 전송하는 것이다.

M_i를 전송하고자 하는 노드는 식(4)를 이용하여 M_i보다 높은 우선순위를 가진 메시지가 실시간 보장이 가능한지를 예측한다.

$$\frac{2B_{current} + C_i}{2P_{current}} \le 1 \tag{4}$$

식(4)의 조건을 만족할 경우에 노드는 다음 알고리즘을 수행한다.

알고리즘 3-1

Step1) 현재의 $P_{current}$와 자신의 주기 T_i의 값을 비교한다.

Step2) $T_i \geq P_{current}$ 이면, T_i의 먹과 C_{icode} 값을 식별자 필드에 넣어 버스중재에 참여한다.

Step3) 버스 점유 후 $T_i \geq P_{current}$ 이면, $P_{current}$에서의 총 전송시간인 $B_{current}$는 $(T_i/P_{current})B_{current}+C_i$로 변경한다.

식(4)의 조건이 만족하지 않을 경우 메시지의 분할 전송을 고려한다. 메시지의 분할전송은 데이터 필드를 2등분(4bytes) 혹은 8등분(1byte)으로 분할 전송한다. 메시지를 분할 전송할 경우 *Mi'(Ci' , Ti' = Ti/2, Di'=Ti')*가 식(4)를 만족하는지의 여부를 판단하고 만족할 경우 Mi'를 전송하고, 만족하지 못할 경우 *Mi"(Ci", Ti" = Ti/8, Di" = Ti")*를 계산하여, 식(4)를 만족하는지의 여부를 판단, 만족할 경우 Mi"을 전송한다.

표 1은 그림 4의 경우에 대해 식(4)를 적용하여 실시간 보장조건을 판정하는 내용이다. 표 1과 같이 *M1*부터 *M4*까지는 실시간성이 보장되므로 메시지가 전송된다. 그러나, *M5*는 실시간 보장이 안되므로 앞서 설명한 바와 같이 메시지 분할 전송을 시도하여 결과적으로 *M5'(C5', T5' = T5/2, D5' = D5')*로 변경되어 전송되며, 이 때에 *'Pcurrent code'*와 *'C5 code'*는 *M5'*에 의해 변경된다.

표 1 메시지중재 지연시간 최소화 알고리즘의 실시간 보장조건 판정

Table 1 Evaluation of message schedulability with arbitration delay minimization algorithm

M_i	실시간 보장조건식 (4)	$P_{current}$	$B_{current}$	$P_{current}$ 갱신	$B_{current}$ 갱신	$P_{current\ code}$	C_i
M_1	만족	0	0	20	6	1	6
M_2	만족	20	6	20	6+6	1	6
M_3	만족	20	12	40	24+6	2	6
M_4	만족	40	30	40	30+6	2	6
M_5	불만족						
M_5'	만족	40	36	80	72+8	3	8

메시지 발생 노드는 실시간 보장조건식의 판정 후에 식별자 필드의 *'$P_{current\ code}$'*와 C_i를 갱신하여 버스중재에 참여한다. 버스를 점유한 메시지의 *'$P_{current\ code}$'*와 C_i는 브로드캐스팅되므로 각 노드는 $P_{current}$와 $B_{current}$를 갱신할 수 있다.

그림 6은 M_5를 동적으로 분할 전송하여 실시간을 보장하는 스케쥴링을 보여준다. 그림 4에서는 M_5의 전송시간에 의해 M_4가 데드라인을 넘지만, 그림 6에서는 M_5가 두 번 나누어 전송되므로 80까지 M_4의 전송지연시간이 최소화되어 시스템의 실시간성이 보장된다.

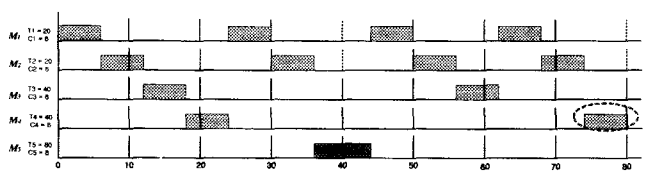


그림 6 메시지 M5의 동적 분할전송에 따른 실시간 보장의 예

Fig. 6 The schedulable example according to dynamic division of message M5

3.3 비실시간 메시지 처리

비실시간 메시지(non-real time message)의 처리는 2.2절에서 설명한 서버알고리즘을 응용한다. 비실시간 메시지는 빠른 응답특성 보다는 전송 종료 시점의 파악과 주기메시지의 실시간 보장에 영향을 주지 않는 것이 중요하다.

EEDS 알고리즘에서 비실시간 메시지의 처리는 전송 완료 시점의 파악을 위해 비실시간 메시지가 발생하는 다른 노드에서는 이미 전송을 시작한 비실시간 메시지의 전송완료 시점까지 중재시도를 지연시키며, 비실시간 메시지의 중재시도를 휴지구간의 시작시점까지 지연시켜 정확한 휴지구간 정보 획득으로 전송할 메시지의 크기를 조절하여 주기적 메시지의 실시간성을 보장한다.

비실시간 메시지 M_s가 휴지구간에서 발생하면 M_s발생 노드는 남아 있는 대역폭을 알 수 없다. 따라서, M_s가 발생하면 M_s의 중재 시도는 발생 시점에서 이루어지지 않고 $t = 2P_{current}(t = 0)$가 될 때까지 중재 대기한다. M_s가 t=0 에서부터 중재 시도를 하더라도 주기적 메시지 M_i에 의해 중재가 지연된다. 즉, 모든 M_i가 적어도 1회 이상 전송된 후에야 메시지 M_s는 전송이 가능하다. 이 시점에서 $B_{current}$는 이미 예측이 완료되었기 때문에 M_s는 자신이 사용할 수 있는 대역폭을 정확하게 예측할 수 있다. 이때, M_s는 $P_{current}$를 $2P_{current}$로 갱신하고, 예측된 대역폭을 이용하여 전송할 C_i를 계산하여 전송한다. t가 $2P_{current}$가 될 때(t=0)까지 $P_{current}$는 갱신되지 않고 $B_{current}$만 갱신된다. M_s는 전송이 완료될 때까지 중재시도는 계속된다. 마지막 메시지 프레임은 M_s의 전송이 완료된다는 것을 각 노드에 알리기 위해 $P_{current\ code}$를 0으로 보낸다.

3.4 EEDS 알고리즘 설계

EEDS 알고리즘은 기존 EDS 알고리즘에서 사용한 로그스케일 사상 알고리즘으로 데드라인 양자화로 인한 지연시간을 최소화하며, 중재 지연시간 최소화를 위한 알고리즘과 비실시간 메시지 처리를 위한 알고리즘을 포함한다.

BNP_i의 최소화를 위한 스케쥴링과 비실시간 메시지의 처리를 위한 노드에서의 작업은 두 가지로 나뉜다. 첫째, 버스 점유 메시지에 의해 각 노드는 $P_{current}$와 $B_{current}$를 갱신해야 하며 둘째, 주기적 메시지는 중재전에 실시간 보장을 위해 메시지를 동적 분할할 수 있어야 하고 비실시간 메시지는 $P_{current}$와 $B_{current}$를 가지고 사용할 대역폭을 결정해야 한다.

알고리즘 3-1은 브로드캐스팅시에 식별자 필드의 정보를 통해 각 노드에서 대역폭 정보를 갱신하는 단계이다. 노드는 식별자 필드의 정보로 $P_{current}$와 $B_{current}$의 값을 갱신한다.

알고리즘 3-1

Step 1) 변수 초기화

If t=0 then

aperiodic msg ← *false*, $P_{current} \leftarrow 0$,

$B_{current} \leftarrow 0$, $P_{prev} \leftarrow 0$, $B_{prev} \leftarrow 0$, $M_{prev} \leftarrow 0$

- 브로드캐스팅 신호가 수신된 경우 노드는 식별자 필드를 검사하여 메시지의 종류와 $P_{current}$와 $B_{current}$값을 갱신한다.

Step 2) 식별자 필드를 검사한다.

$P_{current\ code}$, C_{code}, M_{sort}, P_{start}

Step 3) 메시지의 종류를 결정하고, $P_{current}$를 갱신한다.

If M_{sort} = *1* then *aperiodic msg* ← *true*

If $P_{current\ code}$ = *0* then *aperiodic msg* ← *false*

If M_{prev} = *1* then

$P_{current} \leftarrow P_{prev}$

else

$P_{current} \leftarrow 2P_{prev}$

else

$P_{current} \leftarrow 2^{Pcurrent\ code}$

Step 4) $B_{current}$를 갱신한다.

If B_{prev} = *0* then

$B_{current} \leftarrow C$

else

$B_{current} \leftarrow B_{prev}(P_{current}/P_{prev})+C$

Step 5) 현재의 $P_{current}$, $B_{current}$값을 저장한다.

$P_{prev} \leftarrow P_{current}$

$B_{prev} \leftarrow B_{current}$

$M_{prev} \leftarrow M_{sort}$

알고리즘 3-2는 주기적 메시지가 발생한 노드에서 중재시도 전에 수행하는 작업 단계이다. 메시지가 발생하면 노드는 식 (4)를 이용하여 실시간 보장 조건의 판정과 동적 분할작업을 수행한다.

알고리즘 3-2

Step 1) 변수 초기화 T′ ← 0, T″ ← 0, C′← 0, C″ ← 0

Step 2) 중재 전에 메시지 발생 노드는 $B_{current}$와 $P_{current}$를 이용 실시간 보장 조건식(식 4)을 체크한다.

If $((2B_{current}+C)/2P_{current}) \leq 1$ then

$P_{current\ code} \leftarrow \log_2 T,\ C$

else

$T' \leftarrow T/2$, C′ ← 전송시간(4bytes 데이터)

If $((2B_{current}+C')/2P_{current}) \leq 1$ then

$P_{current\ code} \leftarrow \log_2 T',\ C'$

else

$T'' \leftarrow T/8$, C″ ← 전송시간(1byte 데이터)

If $((2B_{current}+C'')/2P_{current}) \leq 1$ then

$P_{current\ code} \leftarrow \log_2 T'',\ C''$

else

system fail

Step 3) 식별자 필드의 갱신($P_{current\ code}$, C_{code})과 함께 메시지는 중재 시도를 한다.

알고리즘 3-3은 비실시간 메시지가 발생한 노드에서 중재시도 전에 수행하는 작업 단계이다. 비실시간 메시지 발생 노드는 주기적 메시지의 실시간 보장을 위하여 메시지의 중재 참여 시점을 지연시킨다.

알고리즘 3-3

Step 1) 전송할 데이터가 있으면 Step 1)으로 진행한다.

Step 2) 다른 노드에서 이미 비실시간 메시지를 전송하고 있는지를 판단하여 중재시점을 정한다.

If P_{start} = *1* and $P_{current}$ = *0* then

If aperiodic msg = *true* then

go to (Step 1)

else

go to (Step 3)

else

go to (Step 1)

Step 3) 전송할 수 있는 데이터크기를 구하여 사용할 대역폭 크기를 정한다.

$Ci \leftarrow Min(2P_{current}-2B_{current},$ *max size of 1 frame)*

Step 4) 1회 이상 메시지 프레임을 전송한 후 남아 있는 대역폭으로 나머지 데이터를 전송할 수 없으면, $t=0$ ($t= 2P_{current}$)이 되는 시점까지 대기한 후에 다시 중재시도를 한다.

If C_i > 1byte data transmission time then

END

else if last frame then

$P_{current\ code} \leftarrow 0$

else

$P_{current\ code} \leftarrow \log_2(2P_{current})$

Step 5) 식별자 필드를 갱신한다.

Step 6) 중재성공여부를 판단하여 다시 중재 시도할 시점을 정한다.

If arbitration success then

go to (Step 3)

else if aperiodic msg = *true* then

go to (Step 2)

else

go to (Step 3)

4. 모의실험 및 결과 분석

본 장에서는 모의실험을 통해 EEDS 알고리즘의 네트워크의 이용율 및 비실시간 메시지 처리로 인한 주기메시지의 실시간성을 검사하여 알고리즘의 유효성(validity)을 확인한다. 또한 제안된 알고리즘의 비실시간 메시지의 응답특성에 대해 확인한다. 다음은 모의실험을 위한 가정이다.

가정 4-1

- CAN의 전송속도는 125 Kbit/s로 가정한다.
- 노드에서 소요되는 시간(처리시간 및 네트웍 프로토콜의 수행시간)과 지터(jitter)는 고려하지 않는다. 또한 비트 스터핑(bit stuffing)으로 인한 추가전송지연과 메시지 전파지연시간 등과 같이 메시지 전송 지연시간에 비하여 상대적으로 매우 적은 값을 갖는 항목들은 고려하지 않는다.

표 2는 버스를 통해 전송되는 프레임(frame)의 데이터크기(data size) 변화에 따른 전송시간이다

표 2 데이터의 크기에 대한 프레임 전송시간

Table 2 Frame transmission time according to the variation of data size

데이터 크기 (byte)	1	2	3	4	5	6	7	8
전송시간 (*msec*)	0.59	0.65	0.72	0.78	0.85	0.91	0.97	1.04

4.1 네트워크 이용율 모의실험

EEDS의 네트워크 향상을 확인하기 위해서는 EDS 알고리즘과의 네트워크 이용율의 비교, 분석이 필요하다. 표 3은 모의실험을 위해 가정한 주기적 메시지의 구성을 나타낸다.

표 3의 메시지의 구성은 총 네트워크 이용율이 100%보다 작지만 EDS에서 주기적 메시지의 실시간성 보장이 어렵다. 이것은 메시지 9번의 실행으로 발생하는 지연시간 *BNP*로 인해 메시지 8번이 데드라인을 넘어서기 때문이다.

표 3 주기적 메시지의 집합
Table 3 A set of periodic messages

메시지 번호	데이터 크기 (byte)	주기 (10μsec)	데드라인 (10μsec)	전송시간 (10μsec)
1	1	400	400	59
2	1	400	400	59
3	1	400	400	59
4	1	400	400	59
5	2	800	800	65
6	2	800	800	65
7	4	800	800	78
8	4	800	800	78
9	8	3200	3200	104

EDS에서 주기 3200으로 메시지 8번을 전송하기 위해서는 오프라인에서 정적으로 사용자에 의해 데이터의 길이를 8bytes보다 작게 설정하고 정적으로 분할전송이 가능하도록 알고리즘을 설정하여 실시간성을 보장해 주어야 한다. 그러나, EEDS에서는 온라인에서 동적으로 *BNP*를 검사하여 메시지를 동적 분할 전송하기 때문에 별도의 사용자 개입 없이 주기적 메시지의 실시간성을 보장하면서 전송이 가능하다.

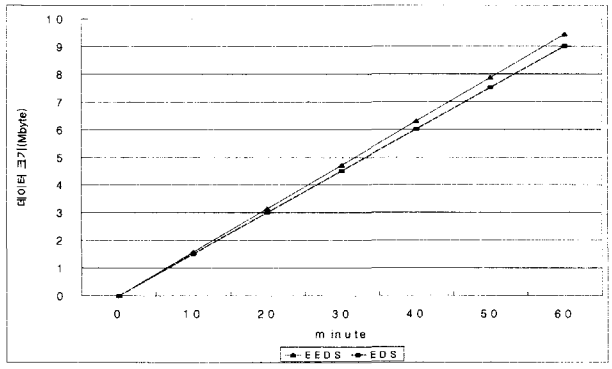


그림 7 EDS와 EEDS의 전송 데이터 크기의 비교
Fig. 7 Comparison of transmission data size between EDS and EEDS

그림 7은 EEDS와 EDS를 적용하여 표 2의 메시지를 60분 동안 스케줄링했을 때 총 전송 데이터의 크기를 보여준다. 그림에서 보듯이 EEDS를 적용했을 때 60분 후의 전송 데이터의 크기는 EDS를 적용했을 때 보다 0.429Mbyte 더 많은 데이터를 전송한다. 이것은 기존 EDS를 사용했을 때 보다 약 2.44% 이용율이 향상된 것이다. 이용율 향상이 상대적으로 적게 나타나는 이유는 주기에 비해 전송시간이 상대적으로 적게 가정되었기 때문이다.

4.2 비실시간 메시지 처리에 대한 주기메시지의 시간성 모의실험

본 절의 실험은 EEDS 알고리즘에서 비실시간 메시지의 처리로 인해 주기적 메시지의 실시간성 성공여부를 확인한다.

비실시간 메시지 모의실험은 10개의 주기적 메시지와 1개의 비실시간 메시지를 EEDS와 EDS에 적용하여 주기적 메시지의 실시간성 성공률(feasibility) 확인을 통하여 수행하고자 하다. 표 4는 실험을 위해 조합 가능한 개별 메시지의 구성 집합을 나타낸다.

표 4 주기적 메시지와 비실시간 메시지가 포함된 집합
Table 4 A set including periodic messages and non-real time message

	주기적 메시지			비실시간 메시지
주기(10μsec)	400	800	1600	
데이터크기 (byte)	2	2, 4	4, 8	64
메시지 개수	2	4	4	1

표 4의 개별 메시지 집합 중 임의의 데이터 크기를 갖는 주기적 메시지 집합 8개와 비실시간 메시지 1개를 이용하여 모의실험을 수행한다.

그림 8은 주기적 메시지의 네트워크 이용율의 증가에 따른 주기적 메시지의 실시간성 성공률을 보여준다. 그림 8에서 보듯이 EDS의 경우는 네트워크 이용율 92.6%이상에서는 실시간성을 보장할 수 없는 경우가 발생한다. 따라서, 이용율 92.6%이상으로 주기적 메시지와 비실시간 메시지 집합을 구성하는 것은 불가능하다. 그러나 EEDS의 경우는 모든 경우에 대하여 실시간성 보장이 가능하다.

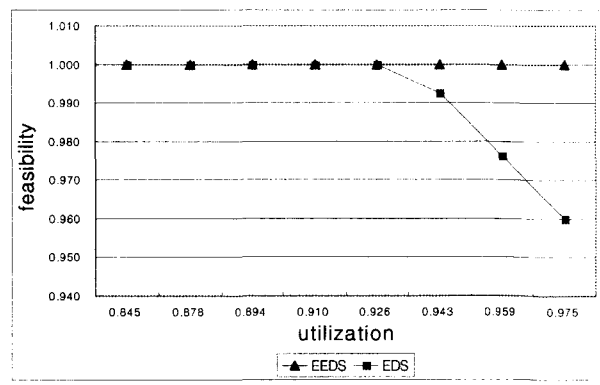


그림 8 주기적 메시지와 비실시간 메시지가 혼재하는 시스템의 실시간성 성공률
Fig. 8 Feasibility of real time transmission in the system with periodic messages and non-real time message

EDS의 경우 비실시간 메시지는 발생하는 시점을 예상할 수 없고 비실시간 메시지가 버스의 휴지구간에 발생하면 주기적 메시지의 전송이 지연될 수 있으므로 주기적 메시지의 실시간성을 보장할 수 없다.

반면에 EEDS는 대역폭 정보를 동적으로 갱신하기 때문에 전송 가능한 비실시간 메시지의 데이터 크기를 조정하고 중재를 지연시킬 수 있으므로 주기적 메시지의 실시간성에 영향을 주지 않는다.

5. 결 론

EDS알고리즘은 낮은 우선순위를 가진 주기적 메시지의 버스 선점으로 인해 발생하는 중재 지연시간이 발생할 수 있으며, 비실시간 메시지의 발생 시점 불확실성으로 인해 비실시간 메시지의 처리가 불가능하다.

이를 보완하기 위해 본 논문에서는 CAN에서의 EDS를 기반으로 한 동적 스케쥴링인 EEDS를 제안하였다. EEDS는 대역폭 정보를 동적으로 갱신하여 낮은 우선순위를 가진 주기적 메시지의 버스선점으로 인해 발생하는 중재 지연시간을 줄일 수 있다. 또한, 대역폭 정보로 비실시간 메시지의 데이터크기를 조정할 수 있으며, 메시지의 중재시도를 지연시켜 주기적 메시지의 실시간성에 영향을 주지 않고 처리할 수 있다. 비실시간 메시지의 처리는 공유 대역폭을 활용하는 서버 알고리즘을 응용하였다. 그리고, 제안한 알고리즘의 효용성에 대해 두 가지 모의실험을 통해 검증하였다.

모의실험은 주기적 메시지로 구성된 메시지들을 가지고 일정시간에 대한 데이터의 전송량을 구하여 네트워크의 이용율 향상을 보였다. 또한, 주기적 메시지와 비실시간 메시지가 혼재하는 시스템에서 주기적 메시지들의 실시간성 성공률을 분석하여 실시간성 보장능력이 더 뛰어남을 입증하였다.

향후 과제로는 실제 CAN통신 시스템에 구현하여 모의실험을 통해 분석한 결과와 비교함으로써 제안된 알고리즘의 효용가치를 평가하는 것과 EDS에 비해 상대적으로 증가한 응용계층의 연산량으로 인한 전체 시스템 지연시간에 관한 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] N. C. Audsley, A. Burns, and M. F. Richardson, Hard Real-Time Scheduling : The Deadline Monotonic Approach, Proc. of IEEE Workshop on Real-Time Operating systems and Software, 1991.

[2] J. Lehoczky, L. Sha, and Y. Ding, The Ratc Monotonic Scheduling Algorithm : Exact Characterization and Average Case Behavior, Proc. Of IEEE Real-Time Systems Symposium, 1989.

[3] Marvo Di Natale,M., Scheduling the CAN bus with earliest deadline techniques, 2000. Proceedings, The 21st IEEE Real-Time Symposium, pp. 259-268, 27-30 Nov. 2000.

[4] D.I.Katcher, S.S.Sathaye, and J.K. Strosnider. "Fixed Priority Scheduling wiht Limited Priority Levels," IEEE Transactions on Computers, Vol. 44, No. 9, pp. 1140-1144, September 1995.

[5] K. M. Zuberi, K. G. Shin, "Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications", in Proc. Real-Time Technology and Applications Symposium, pp. 240-249, May 1995.

[6] M. Spuri and G. Buttazzo., "Scheduling Aperiodic Tasks in Dynamic Priority Systems" , The Juounal of Real-time Systems, vol. 10, pp. 179-210, March 1996.

[7] CAN Specification Version 2.0, Robert Bosch GmbH, 1991.

[8] A.K.Mok, "Fundamental design problems of distributed systems for the hard real-time environment", Ph.D thesis, MIT, 1983.

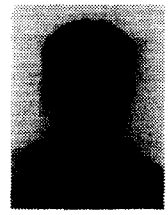
[9] Chetto, H., Chetto, M., "Some Results of the Earliest Deadline Scheduling Algorithm", IEEE Transaction on Software Engineering, Vol 10, pp. 1261-1269, Oct. 1989.

저 자 소 개

이 병 훈 (李 炳 勳)

1974년 09월 24일생. 2000년 명지대학교 전기전자공학부 졸업. 2002년 동대학원 졸업(공석). 2002년~현재 일륭 텔레시스 연구소 연구원. 관심 분야: 실시간 시스템, 통신 네트워크, 통신 프로토콜.

e-mail: museros@hotmail.com



김 홍 렬 (金 弘 烈)

1973년 11월 25일생. 1998년 명지대학교 전기공학과 졸업. 2000년 동 대학원 졸업(공석). 2002년~현재 동대학원 박사과정 및 (주) 캐리어 기술연구소 주임연구원. 관심분야: 실시간 시스템, 분산 제어 시스템, 로봇 프레임워크.

e-mail: museros@hotmail.com

김 대 원 (金 大 元)

1960년 2월 15일생. 1984년 서울대학교 제어계측공학과 졸업. 1986년 동대학원 졸업(공석). 1990년 동대학원 졸업(공박). 1987년~1992년 대우중공업(주) 중앙연구소 선임연구원. 1992년~현재 명지대학교 정보공학과 교수. 관심분야: 자동화 네트워크, 실시간 시스템, 퍼스널 로봇, 웹기반 응용.

e-mail: dwkim@mju.ac.kr

Tel : (031)330-6472

Fax : (031)330-6226