2001년도 대한전기학회 하계학술대회 논문집 2001.7.18-20

CAN기반 실시간 시스템을 위한 확장된 EDS 알고리즘 개발

Development of an Extended EDS Algorithm for CAN-based Real-Time System

이병훈*, 김대원, 김홍렬
명지대학교 정보제어공학과(Tel: 031-330-6472; E-mail: {carpl50, dwkim}@mju.ac.kr; museros@intizen.com)

Byong Hoon Lee, Dae Won Kim, Hong Ryeol Kim
Department of Information Control Engineering, MyongJi University

Abstract

Usually the static scheduling algorithms such as DMS(Deadline Monotonic Scheduling) or RMS(Rate Monotonic Scheduling) are used for CAN scheduling due to its ease with implementation. However, due to their inherently low utilization of network media, some dynamic scheduling approaches have been studied to enhance the utilization. In case of dynamic scheduling algorithms, two considerations are needed. The one is a priority inversion due to rough deadline encoding into stricted arbitration fields of CAN. The other is an arbitration delay due to the non-preemptive feature of CAN.

In this paper, an extended algorithm is proposed from an existing EDS(Earliest Deadline Scheduling) approach of CAN scheduling algorithm having a solution to the priority inversion. In the proposed algorithm, the available bandwidth of network media can be checked dynamically by all nodes. Through the algorithm, arbitration delay causing the miss of their deadline can be avoided in advance. Also non real-time messages can be processed with their bandwidth allocation. The proposed algorithm can achieve full network utilization and enhance aperiodic responsiveness, still guaranteeing the transmission of periodic messages.

Keywords : non-real time message, EDS(Earliest Deadline Scheduling), network utilization, aperiodic server, dynamic scheduling, arbitration delay

1. 서 론

실시간 필드버스 네트워크 프로토콜인 CAN의 스케쥴링은 메시지의 식별자를 이용해 이루어지며, 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]. 하지만, 이러한 정적인 스케쥴링 알고리즘을 사용했을 경우, 메시지의 실시간성을 보장하기 위해서 네트워크의 사용율이 저하되며, 비실시간 메시지(non-real time message)의 처리를 고려했을 경우에는 그 사용율이 더욱 저하된다.

이러한 문제점을 극복하기 위하여 CAN에서의 동적인 스케쥴링 알고리즘(dynamic scheduling algorithm)에 관한 연구가 이루어지고 있다[3]. CAN에서 동적인 스케쥴링 알고리즘을 구현하기 위해서는 두 가지 고려해야 할 사항이 있다. 첫째, 한정된 CAN의 식별자 영역에 데드라인을 인코딩 함으로써 데드라인의 인코딩(deadline encoding)이 부정확해지고 결과적으로 우선순위반전(priority inversion)이 나타날 수 있다. 이러한 우선순위반전을 방지하기 위해 데드라인을 로그스케일(log scale)에 사상하는 알고리즘이 제안된바 있다[4][5]. 둘째, CAN 프로토콜의 충돌 회피 메카니즘(arbitration)에 의해 어떤 메시지가 네트워크를 점유했을 경우에는 다른 메시지는 전송을 시도할 수 없고 결과적으로 메시지전송 시도를 위한 중재(arbitration)가 지연될 수 있다. 이러한 지연시간(delay time)은 최악의 경우 메시지의 실시간성을 보장할 수 없게 되는 요인에 되어 네트워크 사용율을 저하시킨다.

본 논문에서는 상기의 우선순위 반전을 방지하기 위해 데드라인 사상알고리즘에서 확장된 알고리즘을 제안한다. 본 논문에서 제안된 알고리즘을 이용하여 네트워크상의 모든 노드들은 네트워크의 이용 가능한 대역폭을 동적으로 확인할 수 있으며, 이로 인해 실시간성에 영향을 줄 수 있는 중재 지연시간을 사전에 방지 할 수 있다. 뿐만 아니라 ,적절한 대역폭 할당을 통해 실시간성에 영향을 주지 않고 비 실시간 메시지의 처리가 가능해 진다. 이때, 비주기 메시지의 처리는 TBS(Total Bandwidth Server)[6]를 기초로 하여 응용 적용한다.

논문의 본론 1절에서는 CAN의 구조 및 메시지 스케쥴링을, 2절에서는 비주기 메시지의 처리를 위한 TBS를, 3절에서는 CAN의 동적 스케쥴링 알고리즘 구현 시의 고려사항을, 그리고 4절에서는 확장된 EDS 알고리즘을 제안한다.

마지막으로 결론을 맺는다.

2. 본 론

2.1. CAN의 구조 및 메시지 스케쥴링

CAN(Controller Area Network)은 분산된 실시간 제어 시스템에 시리얼 버스로 연결되어 구성된다. CAN의 특징으로는 빠른 전송 속도, 유연성, 높은 데이터 안정성, 분산 접속 통제 방식, 시스템내의 데이터 일관성, 에러감지기능등을 들 수 있으며, 트랜스퍼계층(transfer)에 CSMA/CD+AMP (Carrier Sense Multiple Access/Collision Detection + Arbitration on Message Priority)프토토콜을 가진다. CAN은 20kbit/s에서 1Mb/s까지의 다양한 전송속도를 제공하며, 물리(physical)계층, 트랜스퍼계층, 오프젝트(object)계층의 3계층으로 나뉘어진다.

그림.1은 CAN 프로토콜의 데이터 프레임을 보여준다. 식별자 필드(identifier field)는 중재 필드(arbitration field)라고도 하며, 콘트롤 필드(control field)는 메시지의 타입의 정보를 포함하며, 데이터 필드는(data field) 실제 전송할 데이터를 포함하며, 체크섬(chedk sum)은 메시지 비

트수를 체크하기 위해 메시지비트수를 포함하며, ACK(acknowledge)는 수신에 대한 승인에 대한 정보를 포함하며, ED(end delimiter)는 메시지의 끝을, IS(idle space)는 다음에 버스를 점유할 메시지와 구별하기 위한 비트이다.

그림.1 CAN 데이터 프레임
Figure. 1 Format of the CAN frame

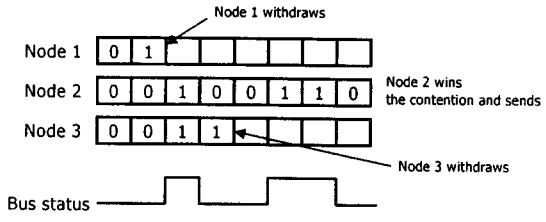

그림 2 . CAN에서의 버스점유를 위한 중재
Figure 2. Example of bus arbitration in CAN

각 노드는 메시지 전송이전에 네트워크의 상태를 감지하여 네트워크에 전송중인 메시지가 없을 때에 전송을 시도하는데, 두 개 이상의 노드가 동시에 메시지를 전송하면 각 메시지는 서로 식별자를 1비트씩 비교한다. 제일 높은 우선순위의 메시지(즉, 가장 낮은 식별자 값을 가진 메시지)는 전송되고 낮은 우선순위의 메시지들은 전송을 중단한다. 그리고, 버스를 점유하지 못한 각 노드의 메시지는 버스를 점유한 메시지가 끝나기를 기다린 후 다시 버스 점유를 시도한다. 그림 2는 버스점유방식의 한 예를 나타낸다.

2.2. 동적 우선순위를 가진 시스템에서의 비주기 메시지의 스케줄링

원래는 실시간 단일 프로세서 시스템의 운영 환경에서 비실시간 비주기 프로세스의 처리를 위해 제안된 TBS(Total Bandwidth Server)는 다른 알고리즘에 비해 구현이 간단하고 처리성능도 뛰어나다.

TBS의 관건은 대역폭의 할당인데, 대역폭이 클 경우에는 주기적 메시지의 실시간성에 영향을 미칠 수 있으며, 반대로 작을 경우에는 비실시간 비주기 메시지의 처리성능이 저하된다. 다음은 단일 프로세서 시스템에서 사용될 경우의 정의이다.

- 모든 주기태스크 τ_i : $i=1,\ldots,n$ 는 데드라인을 가진다.
- 모든 비주기 태스크 J_i : $i=1,\ldots,m$는 데드라인을 가지고 있지 않다.
- 각 비주기 태스크의 도착시간은 알 수 없다.
- 각 주기태스크 τ_i는 주기 T_i와 최악실행시간(worst case execution time)으로 정의한 C_i를 가진다.
- 비주기 태스크의 프로세서 이용율 $U_s=\frac{C_s}{T_s}$
- 주기태스크와 비주기 태스크의 이용율의 조건
$$U_s+U_p\le 1$$
- k^{th} 주기태스크의 데드라인 d_i
$$d_i(k)=r_i(k)+T_i$$

k^{th}에서의 비주기 태스크가 $t=r_k$에서 발생했을 경우, 데드라인은 다음과 같다.

$$d_k=\max(r_k,\ d_{k-1})+\frac{C_k}{U_s} \qquad (1)$$

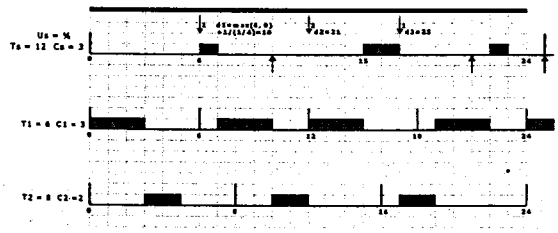

그림 3. Total Bandwidth Server 예제
Figure 3. Total Bandwidth Server example

그림. 3은 TBS알고리즘을 사용하여 비주기 태스크를 처리하는 예제이다. 첫 번째 태스크가 t=6에서 발생하여 $d_1=r_1+\frac{C_1}{U_s}=6+\frac{1}{0.25}=10$의 데드라인을 가지고 처리된다. 시스템에서 데드라인 10을 가진 태스크가 가장 우선순위가 높은 태스크이므로(EDS) 바로 처리된다. 두 번째 비주기 태스크는, $d_2=r_2+\frac{C_s}{U_s}=21$을 가지지만 주기태스크의 데드라인에 밀려 즉시 실행되지 못한다. 마지막으로 세 번째 비주기 태스크의 요청이 $t=18$에서 발생했을 때, $d_3=\max(r_2,\ d_2)+\frac{C_s}{U_s}=21+\frac{1}{0.25}=25$를가지고 $t=22$에서 처리된다.

TBS를 실시간 네트워크 시스템인 CAN에 적용하기 위해서는 다음 두 가지 사항을 고려해야 한다.

첫째, 실시간 단일 프로세서 시스템의 운영환경에서는 스케줄러가 중앙에서 모든 프로세서의 스케줄링을 관리하기 때문에 적당한 대역폭의 할당이 가능하나, CAN에서는 중앙스케줄러가 존재하지 않기 때문에 네트워크 대역폭의 관한 정보가 비실시간 비주기적 메시지를 발생시키는 모든 노드에 공유되어야 한다.

둘째, 비실시간 비주기 메시지의 대역폭 할당시에는 CAN의 충돌 방지 메카니즘에 의한 비선점(non-preemptive)특성이 고려되어야 한다.

이러한 이유로 본 논문에서는 확장된 메시지 포맷의 식별자 비트를 사용 메시지의 종류(실시간주기, 비실시간 비주기)와, 우선순위, 중재 전까지의 주기의 최소공배수와 주기의 최소공배수까지 사용 예측되는 대역폭의 정보를 표현한다. 이러한 정보는 CAN 네트워크 물리계층의 특성상 모든 노드에 브로드캐스팅(broadcasting)되므로 항상 각 노드는 최선의 정보를 유지할 수 있다.

2.3. CAN의 동적 스케줄링 알고리즘 구현 고려 사항

기존의 연구 [3]에서는 CAN네트워크에 동적 알고리즘인 EDS를 사용하여 버스의 이용율의 향상을 보였다. 다음은 본 절에서 사용되는 정의이다.

- 주기 메시지 M_i는 (C_i, T_i, D_i)로 표현된다.
- C_i는 메시지 M_i의 전송시간이다.
- T_i는 메시지 M_i의 주기이다.
- 메시지 M_i의 데드라인인 D_i는 T_i 와 동일하다고 가정한다.
- 절대시간 좌표에서의 데드라인은 $d_i=kT_i$ 이며, $k=1,2,3,\ldots$이다.
- t_{START}는 버스점유를 위해 중재(arbitration)에 참여하는 시점이다.

2.3.1. 데드라인의 인코딩에 의한 지연시간

제한된 식별자 필드에 우선순위의 등급을 표현해야 하는 제약 때문에 t_{START}에서 시작하여 시분할로서 우선순위를 결정하는 데드라인 양자화(deadline quantization)를 한다. 버스를 사용하는 메시지의 최대주기까지 시분할(time section), $\{ I_0, I_1, \ldots I_N\}$을 한다.

시분할에 의한 우선순위 결정은 모든 메시지에 대해 정확한 우선순위를 정하지 못해 우선순위 반전(priority inversion)에 의한 지연시간이 발생할 수 있다. 그림 4는 데드라인 양자화에 의해 생기는 지연시간 BM_i을 보여준다. t_{START}부터 데드라인 d_1과 d_2는 같은 시분할 I_k 부분에 존재한다. 따라서 메시지 M_1과 M_2는 같은 우선순위를 가진다. M_2는 데드라인이 M_1보다 늦지만 먼저 전송될 수 있다. 즉, 우선순위반전이 발생한다.

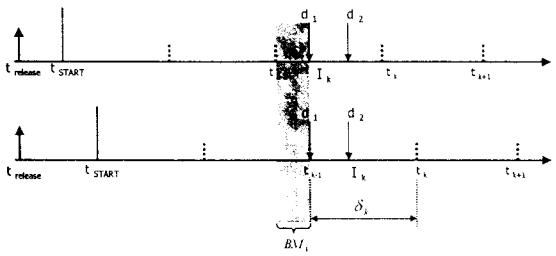

그림 4. 메시지 M_1과 M_2사이의 메시지 우선순위 반전
Figure 4. Priority inversion between messages M_1 and M_2

2.3.2 메시지 중재에 의한 지연시간

CAN버스에서 EDS를 사용하면 최소 데드라인(d_i)을 가지는 메시지가 버스를 점유하지만, 휴지구간에서는 우선순위가 낮은 메시지에 대해 동적으로 버스의 점유권이 주어지기 때문에 낮은 우선순위를 가진 메시지의 전송에 의해서 높은 우선순위의 메시지의 중재가 지연되고, 결과적으로 전송지연시간이 발생한다.

그림 5는 전체 이용율이 1이하로 EDS의 기본조건을 만족한다. 하지만 낮은 우선순위를 가진 메시지에 의해 높은 우선순위의 메시지가 BNP_i만큼 전송 지연되고, 이로 인해 실시간 보장조건을 만족시키지 못하는 경우를 보여주는 예이다. 그림에서 다섯 번째 노드의 메시지가 휴지기간인 36에서 실행하여 네 번째 노드의 메시지가 80에서 데드라인을 넘어가는 것을 볼 수 있다.

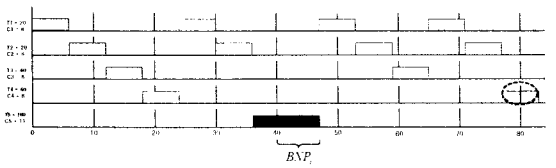

그림 5. 낮은 우선순위의 메시지에 의한 실시간보장의 깨짐현상
Figure 5. Non-schedulibility example caused by occupancy of lower priority message

2.4. 확장된 EDS 알고리즘

본 절에서는 앞 절에서 설명한 BM_i와 BNP_i를 최소화 하면서, 비실시간 비주기 메시지 처리를 위한 알고리즘을 제안한다.

다음은 본 절에서 사용하는 정의이다.

- P는 시스템상의 전체 메시지 주기의 최소공배수(L.C.M)이다.
- t는 $kP \le t < (k+1)P$를 만족하는 메시지 발생 시점이며, $k = 0, 1, 2, \ldots$ 이다.
- $P_{current}$는에서 t까지의 발생한 전체 메시지 주기의 최소공배수(L.C.M)이다.
- $B_{current}$는 kP에서 t까지 발생한 전체 메시지의 주기인 $P_{current}$에서 전송시간으로 사용 될 것으로 예상되는 총 네트워크 점유시간이다.
- U_{T_i}는 $\frac{C_i}{T_i}$이다.
- 비실시간 비주기 메시지 M_s로 표현된다.

다음은 본 절에서 사용하는 가정이다.

- 전체노드에서 발생하는 메시지의 네트워크 이용율은 $U_{TOTAL} = \frac{B_{current}}{P_{current}} \le 1$ 이다.
- M_i의 주기 T_i는 2^i로 설정한다.
- CAN의 전송속도는 125Kbit/s로 설정하고, 본 알고리즘에서는 확장된 식별자 필드를 사용함으로, 이때의 데이터 필드가 8byte, 4byte, 1byte의 전송시간(C_i)은 올림(worst case)한 값인 0.6ms, 0.8ms, 1.1msec에 10을 곱하여 6, 8, 11로 표현한다.

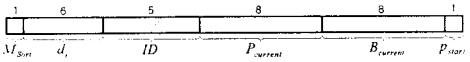

그림 6. 식별자 필드의 구성
Figure 6. Identifier field structure

그림 6은 확장 식별자 필드(extended identifier field)의 구성을 보여준다. M_{sort}는 주기메시지에 비주기 메시지보다 높은 우선순위를 주기 위해 사용되며, 주기메시지는 0값을 가지고, 비주기 메시지는 1을 가진다. ID는 메시지의 유일한 식별자이다.

CAN 네트워크 물리계층의 특성상 모든 노드에 브로드캐스팅(broadcasting)되므로 항상 각 노드는 최신의 정보를 유지할 수 있으므로 그림 6에서 P_{start}, $B_{current}$, $P_{current}$는 각 노드에서 항상 최신의 정보로 유지되어 각 노드간의 공유영역처럼 사용된다.

P_{start}는 kP의 시작시점을 나타내기 위해 사용된다. kP의 시작 시점에서 P_{start}는 우선순위가 가장 높은 메시지에 의해 1로 변경되고, 두 번째 우선순위를 가진 메시지가 P_{start}가 1임을 확인하면, 다시 0으로 변경한다.

본 알고리즘에서는 BM_i를 최소화하기 위하여 데드라인(d_i)을 로그스케일로 사상한 기존의 연구 결과에서 사용한 데드라인 인코딩 방식을 사용하며, 그 방식은 식 (2), (3)과 같다.

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

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

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

q is the number of units wiht which each section can be divided

이 방식을 이용하여 데드라인에 t_{START}가 가까워질수록 우선순위반전을 줄일 수 있다.

본 알고리즘에서 제안하는 BNP_i최소화 알고리즘은 현재 전송하고자 하는 메시지에 의해 더 높은 우선순위를 가진 메시지의 실시간성 보장이 가능한지를 검사하여 실시간 보장이 되지 않을 것으로 예상 될 경우에는 전송하고자 하는 메시지의 데드라인이 초과되지 않는 범위에서 메시지를 분할 전송하는 것이다.

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

$$\frac{2\,B_{current}+C_i}{2\,P_{current}} \le 1 \qquad (4)$$

식(4)의 조건을 만족 할 경우에는 $P_{current}$와 자신의 주기 T_i의 최소공배수(L.C.M)로 $P_{current}$를 변경하고, $P_{current}$내에서 총 전송시간 $B_{current}$와 $\frac{P_{current}}{T_i}\times C_i$ 의 합으로 $B_{current}$변경하여 M_i를 전송한다.

식(4)의 조건을 만족하지 않을 경우 메시지의 분할 전송을 고려한다. 메시지를 분할전송할 경우에는 2등분 혹은 8등분 분할 전송한다. 데이터필드를 2등분 혹은 8등분하여 전송하는 것은 테이터필드 외에 사용한 고정된 필드(64bit)의 오버헤드에 대해 적합한 분할이기 때문이다. 메시지를 분할전송할 경우에는 $M_i'(\ C_i'=\frac{C_i}{2},\ T_i'=\frac{T_i}{2},\ D_i'=T_i')$가 식(4), (5)를 동시에 만족하는지의 여부를 판단하고 만족할 경우 M_i'을 전송하고, 만족하지 못할 경우 $M_i''(\ C_i''=\frac{C_i}{8},\ T_i''=\frac{T_i}{8},\ D_i''=T_i'')$를 계산하여, 식(4), (5)를 동시에 만족하는지의 여부를 판단하고 만족할 경우 M_i''를 전송한다.

$$(\ U_{TOTAL}-U_{T_i})+U_{T_i'}(\text{or}\ \ U_{T_i''})\le 1 \qquad (5)$$

표 1은 그림 5의 경우에 대해 식 (4)를 적용하여 실시간 보장 조건을 판정하는 내용이다.

표 1과 같이 M_1부터 M_4까지는 실시간성이 보장되므로 $P_{current}$와 $B_{current}$가 변경되면서 메시지가 전송된다. 하지만, M_5는 실시간 보장이 안되므로 상기에서 설명한 바와 같이 메시지 분할 전송을 시도하여 결과적으로 $M_5'(\ C_5'=\frac{C_5}{2},\ T_5'=\frac{T_5}{2},\ D_5'=T_5')$ 로 변경되어 전송되며 이 때에 $P_{current}$와 $B_{current}$는 M_5'에 의해 변경된다.

표 1. 그림 5의 식(4)적용
Table 1. Adoption of formula (3) for Figure. 4

M_i	실시간 보장조건	식별자 필드(ID field)	
		$P_{current}$	$B_{current}$
1	$0<1$	20	6
2	$\frac{2\times6+6}{2\times20}<1$	20	6+6
3	$\frac{2\times12+6}{2\times20}<1$	40	24+6
4	$\frac{2\times30+6}{2\times40}<1$	40	30+6
5	$\frac{2\times36+11}{2\times40}>1$		
5'	$\frac{2\times36+8}{2\times40}=1$	80	72+8

그림 7은 M_5를 동적으로 분할 전송하여 실시간을 보장하는 스케쥴링을 보여준다.

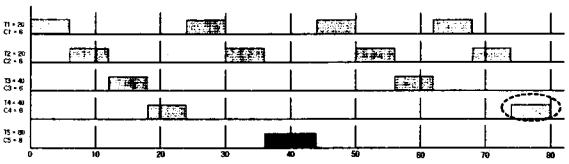

그림 7. 메시지의 동적 분할전송의 예
Figure 7. The example of division transmission for dynamic scheduling of message

전체적인 스케쥴링 흐름도는 그림 8과 같다.

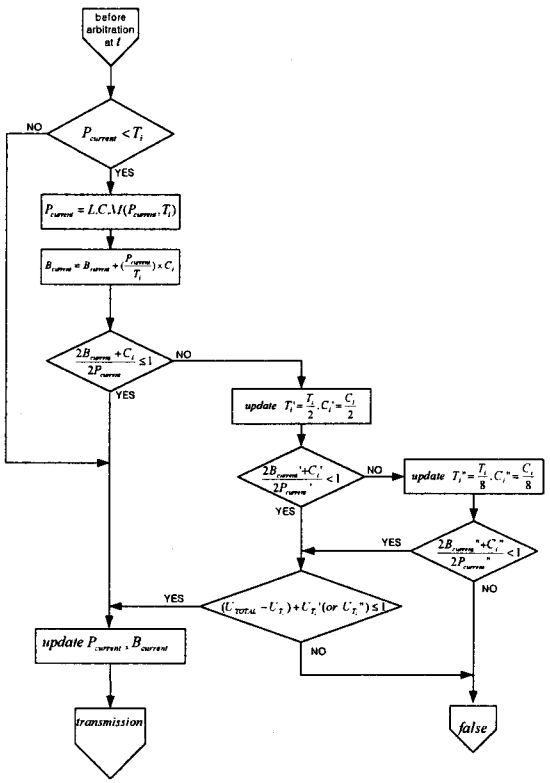

그림 8. 주기적인 메시지의 실시간 보장조건을 위한 동적 스케쥴링 흐름도
Figure 8. Flow chart of dynamic scheduling for schedulibility of period message

본 논문에서 비실시간 비주기 메시지(non-real time message)의 처리는 TBS를 응용하여 사용한다. 비주기 메시지는 주기메시지보다 많은 데이터 량을 가지고 있으므로 빠른 응답 특성보다는 전송 종료시점의 파악과 주기메시지의 실시간 보장에 영향을 주기 않는 것이 중요하다. 따라서 비주기 메시지는 일정한 데이터로 분할되어 주기적 메시지로 취급되어 전송된다.

비주기 메시지발생(release)이 휴지구간에서 발생한다면 비주기 메시지는 주기 메시지처럼 처리되기 때문에, 비주기 메시지에 할당될 수 있는 대역폭 $P_{current}-B_{current}$ 중 사용하지 않고 경과된 대역폭을 측정할 수 없다. 따라서 $P_{start}=0$일 때, 비주기 메시지 M_s가 발생하면 M_s의 중재 시도는 발생 시점에서 이루어지지 않고 $P_{start}=1$이 될 때

까지 중재 대기한다. $P_{start}=1$이 되는 최초의 시점부터 M_s가 중재 시도를 하더라도 M_{sort}에 의해 모든 주기 메시지 M_i 가 적어도 1회 이상 전송된 후에만 전송이 가능하고, 이 때 $B_{current}$는 이미 예측이 완료되었기 때문에 비주기 메시지 M_s는 자신이 사용할 수 있는 대역폭을 정확하게 예측할 수 있다. M_s는 예측된 정보를 이용하여 kP 내에서 전송할 C_s를 계산하고 $U_{TOTAL}+\frac{C_s}{T_s}\le 1$, $T_s \ge 2\times P_{current}$인 범위 내에서 T_s를 결정하여 준다. M_s는 원하는 양의 데이터 전송이 완료되면 소멸된다.

그림 9는 비실시간 비주기 메시지가 주기 메시지로 변경되는 과정을 보여 준다.

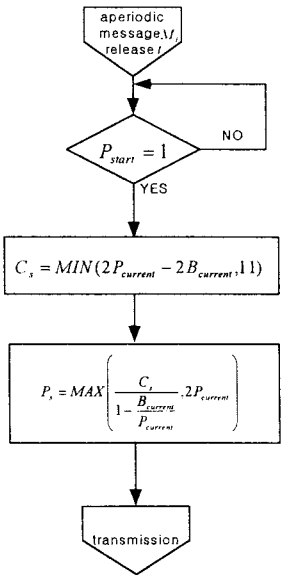

그림 9. 비주기메시지에서 주기적인 메시지로의 변경
Figure 9. Translation from a non-real time message into a periodic message

3. 결 론

본 논문에서는 EDS를 기반으로 동적 스케쥴링에서 중재 지연시간을 줄임으로써, 실시간성을 보장할 수 있는 메시지의 동적 분할 전송 스케쥴링인 확장된 EDS 알고리즘을 제안하였다. 또한, 비실시간 메시지의 효과적인 처리로, 전체 메시지의 실시간 보장조건을 만족시킬 수 있는 서버 알고리즘인 TBS를 응용 사용하였다.

향후 과제는 본 논문에서 사용된 알고리즘의 오프라인의 실시간 보장조건 모델링에 관한 연구와 실시간 비주기 메시지의 처리를 위한 알고리즘을 개발하는 것이다.

[참고 문헌]

[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 Rate 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]H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm", IEEE Transaction on Software Eng., Vol. 15, No. 10, Oct. 1989, pp. 1261-1269

[8]S.H.Hong and W.-H.Kim, "Bandwidth allocation scheme in CAN protocol", in Proc. Control Theory Appl. Vol. 147, No. 1, pp. 37-44, January 2000

[9]Bosch, "CAN specification Version 2.0", Rovert Bosch GmbH, 1991

[10]M. Farsi, K. Ratcliffm, Manuel Barbosa, "An overview of Controller Area Network", Computing & Control Engineering Journal June 1999