한국자동차공학회논문집 제 10 권 제 5 호, pp.114-120

# CAN을 이용한 차체 네트웍 시스템에 대한 Holistic 스케줄링 해석

## Holistic Scheduling Analysis of a CAN based Body Network System

신 민 석*, 이 우 택*, 선우 명호**
Minsuk Shin, Wootaik Lee, Myoungho Sunwoo

## ABSTRACT

In a distributed real-time control system, it is essential to confirm the timing behavior of all tasks because these tasks of each real-time controller have to finish their processes within the specified time intervals called a deadline. In order to satisfy this objective, the timing analysis of a distributed real-time system such as shcedulability test must be performed during the system design phase. In this study, a simple application of CAN for a vehicle body network system is formulated to apply to a holistic scheduling analysis, and the worst-case execution time (WCET) and the worst-case end-to-end response time (WCRT) are evaluated in the point of holistic system view.

주요기술용어 : Worst-case execution time(최악 수행시간; WCET), Worst-case response time(최악 응답 시간; WCRT), Holistic scheduling(통합적 관점의 스케줄링)

### Nomenclature

$R$ : worst-case response time
$w$ : worst-case queuing delay or time window
$C$ : worst-case execution time
$T$ : period
$J$ : jitter
$B$ : blocking time
$P$ : priority
$\tau_{bit}$ : bit time
$hp$ : higher priority
$n$ : number of data bytes

### Subscripts

$i, j$ : task or message index

## 1. 서 론

산업 전반에 걸쳐 실시간 분산제어 시스템의 적용이 일반화 되어가고 그 중요성 또한 높아지고 있다. 특히, 통신 분야를 비롯하여 공장 자동화, 우주 항공 등의 분야에 있어서는 실시간 분산제어 시스템을 사용하지 않고 시스템을 설계하는 것은 거의 불가능하다. 현재 자동차 산업에서도 비용 절감과 제어 성능의 향상을 위하여 실시간 분산제어 시스템의 적용 영역을 점차 확대하고 있다.

* 회원, 한양대학교 대학원
** 회원, 한양대학교 자동차공학과

실시간 분산제어 시스템에서 가장 중요한 것 중 하나는 모든 태스크(task)들의 시간에 대한 제약 조건인 마감시간(deadline)에 대한 만족이다. 따라서 시스템의 설계 단계에서 태스크의 마감시간에 대한 동작을 고려한 설계가 이루어져야 한다. 또한 효과적인 개발 과정을 위하여 이 단계에서 시스템이 마감시간을 만족시키는 지의 여부를 확인하는 것이 필요하다. 이를 위하여 전체 시스템에 대한 시간 해석이 요구되고 있으며, 80년대 이후 실시간 시스템의 스케줄링에 관련된 많은 연구들이 진행되어 왔다.

이번 연구에서는 자동차 산업의 분산제어 시스템을 위하여 개발된 개방형 구조(open-ended architecture)에 대한 업계 표준인 OSEK/VDX[1,2]와 실시간 분산제어 시스템 통신에 있어서 가장 일반적인 프로토콜의 하나인 CAN을 적용하여 구현한 차체 네트웍 시스템에 대한 Holistic 스케줄링 해석을 수행하였다.

## 2. Holistic 스케줄링

여러 개의 프로세서가 분산 되어있는 시스템의 경우 노드 간의 통신이 태스크의 응답시간(response time)에 영향을 주게 되는데, 이러한 문제를 고려하여 시스템의 스케줄링에 대한 해석을 하는 것을 holistic 스케줄링 해석이라 한다. Holistic 스케줄링 해석을 하기 위하여 각 노드의 태스크들과 CAN 메시지에 대한 최악 수행시간(worst-case execution time; WCET)과 최악 응답시간(worst-case response time; WCRT) 등의 값이 필요하다.

### 2.1 최악 수행시간(WCET)[3,4]

WCET는 태스크가 다른 태스크의 방해 없이 한번 수행하는데 걸리는 가장 긴 시간을 의미한다. 이번 연구에서 WCET의 계산은 소스 코드 레벨에서 제어 흐름도(control flow graph, CFG)[5,6]를 이용하여 태스크의 실행 가능 경로를 확인하고, 각 경로를 수행하는데 걸리는 시간은 기계어 레벨에서 해석하여 가장 긴 값을 WCET로 정의하였다.

### 2.2 최악 응답시간(WCRT)[7]

WCRT는 태스크의 실행이 요구된 후 동작을 완료하는데 필요한 가장 긴 시간을 의미하며 그 값은 식 (1)과 같이 표현된다.[8] 식 (1)에서 w는 임계 상황(critical instant)에서 태스크가 실행을 시작한 후 완료될 때 까지 걸리는 최대 시간을 말한다. 이 값은 식 (2)와 같이 높은 우선 순위 태스크의 수행이나 세마포어(semaphore) 등으로 인하여 야기되는 선점(pre-emption)에 의해 생기는 태스크의 지연과 태스크의 WCET값을 합한 값으로 표현되며, 이를 시간 창(time window)이라 한다. 이 시간 창에 태스크의 실행이 요구된 후 실제 실행하기까지 걸리는 지연(delay)의 최대 차인 지터(jitter)를 더한 값이 WCRT가 된다.

$$R_i = w_i + J_i \tag{1}$$

$$w_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_i}{T_j} \right\rceil C_j \tag{2}$$

식 (2)는 CPU의 컨텍스트 스위칭(context swit-ching)과 스케줄링에 필요한 시간 등의 오버헤드(overhead)를 고려하지 않은 것이다. 좀 더 정확한 해석 모델을 위하여 오버헤드를 고려한 WCRT를 구한다. OSEK OS의 스케줄러는 이벤트 스케줄링 방식으로 동작하므로 식 (2)에 컨텍스트 스위칭($C_{sw}$)과 실제 스케줄링 하는데 필요한 시간($C_{timer}$)을 더해 주어야 한다. 그러면 식 (3)과 같은 형태의 수정된 시간 창를 얻을 수 있다.

$$w_i = C_i + 2C_{sw} + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil (C_j + 2C_{sw}) + \sum_{\forall j \in alltasks} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_{timer} \tag{3}$$

### 2.3 CAN의 시간 해석

CAN 메시지에 대한 시간 해석 방법은 태스크의 해석에 적용 했던 것과 유사한 방법으로 문제를 정형화 하여 해석한다.[9,10] 메시지는 주기와

전송 데이터의 길이가 일정한 것으로 가정하여 메시지를 주기가 T이고 WCET가 C인 태스크로 간주하여 해석한다.

s바이트의 데이터를 CAN 버스로 전송할 경우 최대 스터프 비트(stuff bit)는 식 (4)와 같으므로 이때 전체 메시지의 전송 시 걸리는 시간은 식 (5)로 표현된다.

$$max.\ number\ of\ stuff\ bits = \left\lfloor \frac{34+8s_i-1}{4} \right\rfloor \quad (4)$$

$$C_i = \left(8s_i + 47 + \left\lfloor \frac{34+8s_i-1}{4} \right\rfloor\right)\tau_{bit} \quad (5)$$

CAN 프로토콜에서 전송을 시작한 메시지는 자신보다 높은 우선 순위의 메시지에 의해 선점되지 않는다. 이러한 프로토콜의 특성 때문에 높은 우선 순위를 가진 CAN 메시지는 전송중인 낮은 우선 순위를 가진 메시지에 의하여 지연이 생기게 되는데 이것을 저지시간(blocking time)이라 하며, 식 (6)과 같다. 저지시간은 8바이트의 데이터를 전송할 경우 최대 $135\tau_{bit}$동안의 지연을 가질 수 있다. 여기서 $\tau_{bit}$은 1비트의 메시지를 전송하는데 걸리는 시간을 말한다.

$$B_i = \max_{\forall k \in lp(i)} (C_i) \le 135\tau_{bit} \quad (6)$$

메시지가 우선 순위 결정에 참가하기 시작한 시간부터 버스를 점유하여 프레임의 첫번째 비트를 전송하기 까지 걸리는 최대 시간을 대기 시간(queuing time, $w_i$)이라 정의한다. 이것은 식 (7)과 같이 낮은 우선 순위의 메시지에 의한 저지 시간과 우선 순위 결정 시작 시점에서 CAN 버스를 점유하는데 걸리는 시간, 즉 순위 결정에 참가한 모든 높은 우선 순위 메시지의 전송에 걸리는 시간의 합을 의미한다. CAN 메시지의 대기 시간은 태스크의 시간 창과 유사한 개념으로, 메시지의 WCRT는 대기 시간에 메시지의 지터와 $C_i - \tau_{bit}$을 더한 값으로 식 (8)과 같다. 또한 대기시간은 식 (9)와 같이 쓸 수 있으므로 식 (8)은 식 (10)과 같이 표현할 수 있다.

$$w_i = \tau_{bit} + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_j \quad (7)$$

$$R_i = J_i + w_i + C_i - \tau_{bit} \quad (8)$$

$$w_i = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j \quad (9)$$

$$R_i = J_i + w_i + C_i \quad (10)$$

## 2.4 Holistic 스케줄링

여러 프로세서로 이루어진 분산제어 시스템의 동작을 해석하기 위하여 먼저 시스템의 구조와 이 구조를 해석하는데 필요한 매개변수가 무엇인지를 알아야 한다.

이 연구에서 설계한 차체 네트웍의 구조는 글로벌 타이머(global timer) 없이 고정 우선 순위 스케줄링 알고리즘에 의하여 동작하는 프로세서와 통신 프로토콜로 이루어진 구조이다. 글로벌 타이머가 없는 시스템이므로 노드 간의 동기화는 메시지에 의하여 이루어진다. 이 경우 노드 간의 응답시간은 Fig. 1과 같이 송신 노드에서 메시지 전송 시작까지의 응답시간(Sampling)과 메시지 전송에 걸리는 시간(Communication), 그리고 수신 노드에서 메시지를 수신하는데 걸리는 시간(Receive)과 이 메시지를 이용하여 실제 출력을 내보내는데 걸리는 시간(Actuation)의 상호 관계를 고려하여 얻을 수 있다.[11,12]

네트웍 메시지의 지터는 메시지를 전송하는 태스크에 의하여 야기되는 것으로, 식 (11)과 같

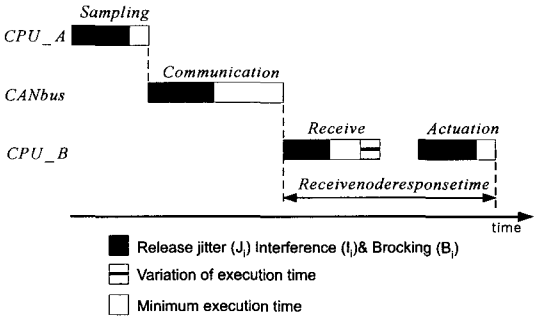


Fig. 1 Fixed priority communication with fixed priority processor without global timer[11]

이 메시지 전송 태스크의 WCRT와 같다. 메시지를 받는 태스크의 지터는 전송 되는 메시지에 의해 야기되는 것으로 메시지 전송에 걸리는 가장 긴 수행시간인 $R_i$와 전송시 걸릴 수 있는 가장 짧은 시간인 $47\tau_{bit}$의 차로 식 (12)와 같이 표현된다.

$$J_i = R_{send(i)} \tag{11}$$

$$J_{dest(i)} = R_i - 47\tau_{bit} \tag{12}$$

메시지 전송 태스크의 실행을 요청하는 이벤트의 발생부터 수신 태스크의 실행 완료까지 걸리는 전체 시간은 식 (13)과 같이 전송 태스크의 WCRT에 메시지의 WCRT와 수신 태스크의 WCRT를 더한 값이 됨을 알 수 있다. 여기서 $w_i$와 $w_{dest(i)}$는 각각 메시지 $i$의 대기 시간과 수신 태스크의 시간 창을 의미한다. 이것을 식 (1), (11)과 (12)를 이용하여 정리하면 식 (14)를 얻을 수 있다. 여기서 구한 $R_{end-to-end}$은 Fig. 1의 Sampling, Communication, Receive를 포함하는 것이다.

$$R_{end-to-end} = R_{send(i)} + w_i + C_i + w_{dest(i)} \tag{13}$$

$$R_{end-to-end} = 47\tau_{bit} + R_{dest(i)} \tag{14}$$

Fig. 2는 CPU_B가 CPU_A로부터 Message_A를 수신하는데 걸리는 시간을 구간별로 나타낸 것이다.

## 3. 시스템 구성 및 정형화

### 3.1 시스템 구성

전체 네트웍 시스템은 모토로라 OSEK을 이용하여 설계한 OSEK/VDX 표준을 만족하는 시스템이다. 여기서 구현된 차체 네트웍 시스템은 Fig. 3과 같이 네 개의 노드로 구성되어 있으며, 각 노드는 자체의 입력과, 운전자 측에서 전송되는CAN 메시지를 입력 신호로 가진다. 전체 네트웍에서 운전자 노드는 송신자가 되고 나머지 노드들은 수신자가 된다.

각 노드의 INPUT_T 태스크는 일정 주기(20ms)로 입력을 확인하고 그 결과에 따라 메시지를 보낸다. 다른 태스크들은 메시지에 의하여 발생되는 이벤트에 의해 실행된다. 그러므로 INPUT_T태스크 이외의 태스크는 주기가 20ms인 간헐적 태스크(sporadic task)로 간주하고 해석한다. 이 시스템에서 계산된 WCET 값은 Table 1과 같고, 여기서 D는 운전자(driver), P는 승객(passenger)을 나타내며, F는 앞(front), R은 뒤(rear)를 나타낸다.

이 시스템에서 $C_{sw}$의 값은 20us이고 $C_{timer}$의 값은 0으로 간주한다. 이것은 $C_{timer}$의 경우 전체 태

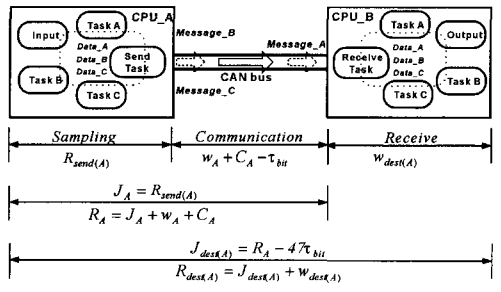


Fig. 2 End-to-end response time

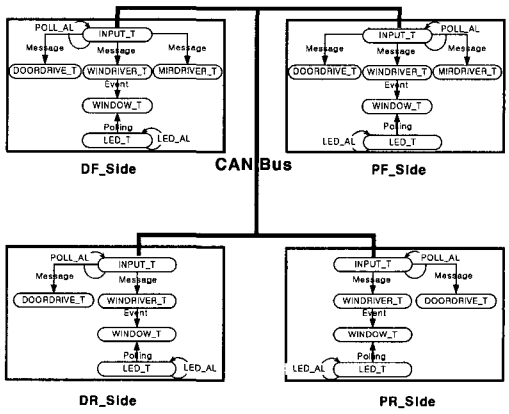


Fig. 3 Body network system

Table 1 The result of WCET analysis

| | DF side | PF side | DR side |
|---|---|---|---|
| INPUT_T | 561.11us | 595.615us | 431.415us |
| DOORDIVER_T | 83.56us | 83.56us | 83.56us |
| MIRDVER_T | 71.08us | 71.08us | 71.08us |
| WINDRIVER_T | 114.24us | 114.24us | 114.24us |
| WINDOW_T | 42.32us | 42.32us | 42.32us |
| COM_T | X | 3.0us | 3.0us |

스크의 실행 시간에 비해 그 값이 무시할 정도로 작으므로 0으로 가정한 것이다.

### 3.2 해석을 위한 정형화

전체 시스템에서 메시지의 전달이 일어나는 부분을 간략히 나타낸 것이 Fig. 4이다. 송신 노드에서 태스크가 CAN 버스로 메시지를 보내는 시간은 무시할 정도로 작은 값이므로 시스템의 모델링에서 고려하지 않는다. 수신 노드로 전송된 메시지는 CAN 드라이버(driver)의 버퍼로 옮겨지고, 이 메시지를 프로세서 내의 INPUT_T 태스크가 주기적으로 확인하여 새로운 메시지가 있을 경우 이를 갱신한다. 여기서 Fig. 4의 COM _T 태스크는 수신 메시지를 애플리케이션 태스크가 접근 할 수 있도록 물리 계층(physical layer)에서 CAN 드라이버의 버퍼로 옮겨 오는 역할을 한다.

전체 시스템의 동작을 나타내기 위하여 선행 관계(precedence relationship)을 사용한다. 선행 관계는 태스크간 또는 태스크와 메시지 간에 수행 순서를 정해두어 그들 간에 수행 순서가 바뀌지 않도록 관계를 지어주는 것이다. 두 태스크 사이에서 한 태스크의 동작이 완료되어야만 다른 태스크의 실행 요청이 일어날 수 있는 경우나 메시지의 수신자, 송신자 그리고 메시지 사이의 관계를 표현하기 위하여 사용한다. 이러한 방법으로 모든 태스크와 메시지를 선행 관계로 묶어 메시지의 흐름과 태스크간의 관계를 결정지어 준다.

## 4. 시뮬레이션 결과 및 해석

시뮬레이션 결과는 Table 2와 같다. 표에 나타나 있는 R값은 INPUT_T 태스크가 실행된 시점부터의 시스템 응답시간을 의미한다. 그러므로

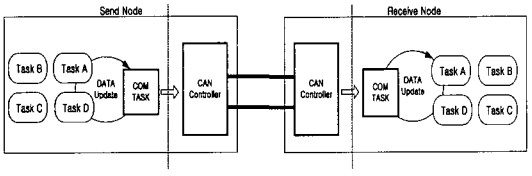


Fig. 4 Communication model

실제 시스템 전체의 수행 시간을 얻기 위해서는 INPUT_T 태스크의 실행 요청에 의한 지터를 고려해 주어야 한다. 즉, 입출력간 최대 응답시간은 식 (14)를 이용하여 구한 값과 INPUT_T 태스크의 주기인 20ms를 송-수신 노드의 INPUT_T 태스크에 대한 각각의 지터로 최종적으로 더해줌으로써 얻어진다.

실험 결과 각 CPU의 활용도(utilization)에 비해 CAN 버스의 활용도가 컸다. PF, DR과 PR측에 태스크들의 경우 CAN 메시지에 의하여 발생하는 지터의 크기가 태스크의 주기보다 큰 값을 가졌다. 이것은 임계 상황에서 수신 태스크는 CAN 메시지를 놓칠 수 있음을 의미한다. 이렇게 시간에 대한 요구를 만족시키지 못하는 것이 통신에 의한 것일 경우 문제를 해결하기 위하여 CAN 메시지의 ID를 새롭게 정의하거나 전송될 신호를 다르게 패킹(packing)하여 CAN 메시지에서 발생하는 부하를 줄이거나, 가능한 경우 버스의 속도를 조정함으로써 문제를 해결할 수 있을 것이다. 또한, 일반적으로 태스크가 마감시간을 만족시키지 못할 경우 태스크의 계산 시간을 줄이기 위하여 프로그램을 시간에 대하여 최적화 시키거나, 태스크의 구조를 바꾸는 등의 노력으로 문제를 해결할 수 있을 것이다.

각 노드간의 통신을 무시하고 하나의 노드에 대한 WCRT를 구할 경우(DF SIDE) CPU는 정상 동작하는 것으로 판단할 수 있다. 그러나 분산제어 시스템으로서는 실시간 시스템의 주기 내에 주어진 일을 모두 수행 할 수 없음을 확인하였다. 그러므로 시스템의 설계 과정에서 한 노드에 대한 해석 뿐만 아니라 통신과 관련된 부분에 대한 시간도 같이 고려하여 설계가 이루어져야 한다.

이번 연구에서는 통신과 관련된 태스크를 재설계하는 방법을 채택하여 시스템을 재구성하고 응답시간에 대한 시뮬레이션을 수행하였다. 시스템에서 CPU와 CAN 버스에 걸리는 부하가 그리 크지 않으므로 CPU와 CAN 버스의 사용도가 높아지더라도 입출력간의 최대 응답시간을 줄이는 방향으로의 재설계가 바람직하다. 그러

Table 2 Simulation results
CAN message (utilization : 12.6%)

| Name | Time (ms) | | | | | |
|---|---|---|---|---|---|---|
| | T | C | J | B | R | W |
| CAN_DL_MSG | 20 | 0.52 | 0.601 | 0.52 | 1.641 | 0.52 |
| PF_MIR_MSG | 20 | 0.52 | 0.601 | 0.52 | 2.161 | 1.04 |
| PF_WIN_MSG | 20 | 0.52 | 0.601 | 0.52 | 2.681 | 1.56 |
| DR_WIN_MSG | 20 | 0.52 | 0.601 | 0.52 | 3.201 | 2.08 |
| PR_WIN_MSG | 20 | 0.52 | 0.601 | 0 | 3.201 | 2.08 |

Worst case response time in a single node

| Input | DF (ms) | PF (ms) | DR &PR (ms) |
|---|---|---|---|
| Door input | 20.72467 | 20.759175 | 20.594975 |
| Mirror input | 20.83575 | 20.870255 | |
| Window input | 21.07231 | 21.106815 | 20.831535 |
| Utilization | 4.362% | 4.549% | 3.373% |

End-to-end worst case response time

| Sender | Receiver | End-to-end WCRT (ms) |
|---|---|---|
| DF_SIDE | PF_SIDE | 20+0.376+20+4.56174=44.93774 |
| DF_SIDE | DR_SIDE | 20+0.376+20+4.53118=44.90718 |
| DF_SIDE | PR_SIDE | 20+0.376+20+4.53118=44.90718 |

Table 3 Simulation results of modified system
CAN message (utilization : 12.6%)

| Name | Time (ms) | | | | | |
|---|---|---|---|---|---|---|
| | T | C | J | B | R | W |
| CAN_DL_MSG | 10 | 0.52 | 0.250 | 0.52 | 1.290 | 0.52 |
| PF_MIR_MSG | 10 | 0.52 | 0.250 | 0.52 | 1.810 | 1.04 |
| PF_WIN_MSG | 10 | 0.52 | 0.250 | 0.52 | 2.330 | 1.56 |
| DR_WIN_MSG | 10 | 0.52 | 0.250 | 0.52 | 2.850 | 2.08 |
| PR_WIN_MSG | 10 | 0.52 | 0.250 | 0 | 2.850 | 2.08 |

Worst case response time in a single node

| Input | DF (ms) | PF (ms) | DR & PR (ms) |
|---|---|---|---|
| Door input | 20.88346 | 20.875995 | 20.703985 |
| Mirror input | 20.99454 | 20.987075 | |
| Window input | 21.2311 | 21.223635 | 20.940545 |
| Utilization | 7.56% | 8.84% | 6.45% |

End-to-end worst case response time

| Sender | Receiver | End-to-end WCRT (ms) |
|---|---|---|
| DF_SIDE | PF_SIDE | 10+0.376+10+4.44386=24.81986 |
| DF_SIDE | DR_SIDE | 10+0.376+10+4.39768=24.77368 |
| DF_SIDE | PR_SIDE | 10+0.376+10+4.39768=24.77368 |

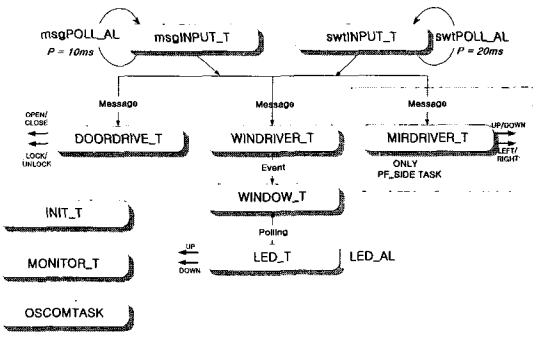


Fig. 5 Modified task diagram

므로 재구성한 시스템에서의 태스크 구조는 Fig. 5와 같이 송-수신 노드의 INPUT_T 태스크를 수행주기가 10ms인 CAN 메시지와 관련된 입력을 처리하는 태스크(msgINPUT_T)와 20ms의 주기를 가지는 스위치 입력을 처리하는 태스크(swtINPUT_T)로 분리한 구조이다. 이렇게 시스템을 수정하면 임계상황에서 swtINPUT_T 태스크를 제외한 태스크들의 주기는 msgINPUT_T 태스크의 주기와 같은 10ms가 된다. 태스크를 분리할 결과는 Table 3과 같다. 시스템의 재구성 결과, 노드내의 전체 응답시간은 조금 길어졌으나 메시지 전송 태스크인 msgINPUT_T의 응답시간이 INPUT_T 태스크의 응답시간 보다 짧아 메시지의 지터를 줄일 수 있었고, msgINPUT_T 태스크의 주기를 10ms로 함으로써 주기적인 입력 확인에 의하여 야기되는 전송 태스크의 지터를 줄임으로써 노드 사이 의 입출력간 최대 응답시간을 한 노드내의 최대 응답시간 수준으로 줄여 만족할만한 결과를 얻을 수 있었다. 수정하기 전과 비교하여 CPU와 CAN 버스의 사용도는 msgINPUT_T 태스크의 주기에 의하여 두 배가 됨을 알 수 있다.

## 5. 결론 및 향후 연구 방안

이번 연구를 통해 자동차의 차체 네트웍 시스템을 holistic 스케줄링 관점에서 해석할 수 있는 형태로 표현 하였고, end-to-end WCRT에 대한 해

석을 수행하였다. 시뮬레이션 시험 결과를 통해 분산제어 시스템을 설계할 때는 설계 단계에서 시간에 대한 제약조건을 고려하고 시스템을 설계해야 함을 확인하였다. 또한 시스템의 재설계를 통하여 네트웍에 의해 시스템 동작이 제한될 경우에 대한 하나의 해결 방안을 제시하였다.

이번 연구에서는 간단한 선행 관계를 가지는 시스템에 대한 해석을 수행하였으나, 향후 연구에서는 이러한 수학적 해석을 통해 시스템을 최적으로 할 수 있는 설계 사양을 찾는 것에 대한 연구가 필요하다.

## 참 고 문 헌


1) Motorola, HC12 OSEK Operating System User's manual Rev.1.7.
2) Motorola, OSEK COM/NM User's Manual Rev. 1.0.
3) H. A. Aljifrri, A. Pons, M. A. Tapia, "Tighten the Computation of Worst-Case Execution-Time by Detecting Feasible Paths," Performance, Computing, and Communications Con -ference, 2000. IPCCC '00. Conference Proceeding of the IEEE International, pp.430-436, 2000.
4) P. Puschner, C. Koza, "Calculating the Maximum Execution Time of Real-Time Programs," Real-Time Systems, No.1 pp.159-176, 1989.
5) P. Altenbernd, "On the False Path Problem in Hard Real-Time Programs," Real-Time Systems, Proceedings of the Eighth Euromicro Workshop on, pp.102 -107, 1996.
6) B. A. Cota, D. G. Fritz, R. G. Sargent, "Control Flow Graphs as a Representation Language," Simulation Conference Proceedings, pp.555-559, 1994.
7) M. Joseph, P. Pandya, "Finding Response Times in a Real-Time System," BCS Computer Journal, Vol.29, No.5, pp.390-395, 1986.
8) K. Tindell, J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-time Systems," Microprocessors and Microprogramming, pp.117-134, 1994.
9) 선우명호, 윤팔주, 이우택, 양성모, "자동차 네트웍 시스템 분석을 위한 시뮬레이션 알고리즘 개발에 관한 연구," 한국자동차공학회 전기 및 전자부문, ITS 부문 학술강연 논문집, pp.18-24, 2000.
10) Wolfhard Lawrenz, CAN System Engineering from Theory to Practical Application, Springer, 1997.
11) H. Lonn, J. Axelsson, "A Comparison of Fixed-Priority and Static Cyclic Scheduling for Distributed Automotive Control Applications," Proceedigns, 11th Euromicro Conference on Real-time Systems, York, England, 1999.
12) M. Torngren, "Fundamentals of Implementing Real-Time Control Applications in Distributed Computer Systems," Real-time Systems, Vol.14, pp.219-250, 1998.