Transactions of KSAE, Vol. 13, No. 1, pp.194-203 (2005)

1225-6382/2005/073-25

응답 시간 해석 도구를 이용한 실시간 분산 제어 시스템의 시간 해석

최 재 범 · 신 민 석 · 선 우 명 호* · 한 석 영

한양대학교 자동차공학과

Timing Analysis of Distributed Real-time Control System using Response-time Analysis Tool

Jaebum Choi · Minsuk Shin · Myoungho Sunwoo* · Seogyoung Han

Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea

(Received 3 May 2004 / Accepted 2 November 2004)

Abstract : The process of guaranteeing that a distributed real-time control system will meet its timing constraints, is referred to as schedulability analysis. However, schedulability analysis algorithm cannot be simply used to analyze the system because of complex calculations of algorithm. It is difficult for control engineer to understand the algorithm because it was developed in a software engineer's position. In this paper we introduce a Response-time Analysis Tool(RAT) which provides easy way for system designer to analyze the system by encapsulating calculation complexity. Based on the RAT, control engineer can verify whether all real-time tasks and messages in a system will be completed by their deadline in the system design phase.

Key words : Distributed real-time control system(실시간 분산 제어시스템), Worst case execution time(WCET; 최악 실행시간), Worst case response time(WCRT; 최악 응답시간), Schedulability(스케줄가능여부), Critical instance (임계상황), Holistic analysis(통합적 관점의 해석)

Nomenclature

R : worst-case response time, ms
ω : worst-case queuing delay or time window, ms
C : worst-case execution time, ms
T : period, ms
J : Jitter, ms
B : Blocking time, ms
hp : higher priority
τ_{bit} : bit time, ms/bit
s : message length, bytes
P : time constant for LIN, ms

Subscripts

SYNBRK : synchronization break
SYNDEL : synchronization delimiter
INFRAME : in frame response space
INTERBYTE : interbyte space

*To whom correspondence should be addressed.
msunwoo@hanyang.ac.kr

1. 서 론

오늘날 실시간 분산 제어 시스템은 로봇, 항공, 통신, 미디어, 자동차에 이르기까지 광범위하게 사용되고 있다. 이러한 시스템은 스케줄링과 공유 자원 등에 대한 시간 제약 조건 때문에 시간의 불확실성 문제가 발생하는데 특히 자동차와 같은 경성 실시간 제어 시스템(hard real-time control system)은 시간

에 대한 요구 조건을 만족하지 못할 경우에 제어 성능이 감소될 뿐만 아니라 생명에도 영향을 줄 수 있다. 때문에 태스크의 수행이나 메시지의 송·수신이 마감 시간 내에 모두 이루어지도록 하는 스케줄링 알고리즘이 존재하지 않을 경우에는 시스템을 재설계해야 한다. 효율적인 시스템의 설계를 위해서는 이러한 시험들이 시스템의 구현이 끝난 뒤 실험에 의해서가 아니라 설계 단계에서 체계적으로 확인되어야 한다.

그 동안 실시간 분산 제어 시스템에 대한 관심이 높아지면서 이 분야에 대한 활발한 연구가 진행되었고 많은 시간 해석 알고리즘들이 개발되었다. 그 중에 가장 일반적인 해석 알고리즘은 최악 응답 시간 해석 알고리즘(worst-case response time analysis algorithm)[1,2]이다. 이 알고리즘은 시스템의 응답 시간이 최대로 발생하는 임계 상황(critical instance)을 고려했을 경우 시스템이 마감 시간을 만족하는지 여부를 검증함으로써 시스템의 안전성 여부를 판단한다. 1972년 Liu와 Layland는 이 알고리즘을 기초로 시스템의 구현 가능성 여부를 판단할 수 있는 변수로 CPU 사용률을 제시하였다.[1] 이후 Tindell은 태스크와 CAN메시지의 응답 시간을 수식적으로 표현하였고 holistic 해석 방법을 제시하였다.[2-5] 그리고 Youn은 LIN메시지의 시간 모델을 정의하고 응답 시간을 수학적으로 표현하였다.[6] 그러나 이러한 해석 방법은 복잡한 연산을 반복해서 수행해야 할 뿐만 아니라 알고리즘 자체가 소프트웨어 설계자 입장에서 개발되었기 때문에 제어기 설계자가 접근하는 것이 쉽지 않다. 그러므로 제어기 설계자가 실시간 분산 제어 시스템을 효과적으로 해석하고 개발하기 위해서는 복잡한 해석 알고리즘을 소프트웨어로 구현한 시간 해석 도구가 필요하다. 그동안 이러한 필요성 가운데 PERTS,[7] BASEMENT,[8] MAST[9]와 같은 해석 도구들이 개발되었다.

이 연구에서는 MATLAB을 기반으로 개발된 응답시간 해석 도구인 RAT(Response-time Analysis Tool)을 제시한다. RAT은 기존에 개발되었던 해석 도구들이 이상적인 해석 모델을 가정한 것과 달리 "Transaction"을 정의함으로써 보다 현실적인 해석이 가능하며 선행관계를 정의하여 holistic 해석이 가능하도록 했다. 또한, RAT에는 CAN(Control Area Network)과 LIN(Local Interconnect Network) 프로토콜을 기반으로 하는 실시간 분산 제어시스템에 대한 시간 해석 알고리즘이 소프트웨어로 구현되어 있기 때문에 RAT을 활용하면 CAN과 LIN 프로토콜을 기반으로 하는 자동차의 실시간 분산 제어시스템의 시간에 대한 응답특성을 효과적으로 해석하고 이를 고려한 설계를 할 수 있다.

2장에서는 대상 시스템의 해석 모델을 제시한다. 3장에서는 RAT를 소개하고 4장에서는 RAT를 이용하여 차체 네트워크 시스템의 시간 특성을 해석하고 검증한다. 마지막으로 5장에서는 이 연구를 통하여 얻어진 결론을 제시한다.

2. 해석 모델 및 가정

이 연구에서는 마감시간을 반드시 만족해야 하는 경성 실시간 시스템(hard real-time system)을 대상 시스템으로 정의한다. 또한 시스템을 해석 가능한 형태로 표현하기 위해 모든 태스크와 메시지는 일정한 주기를 갖는 것으로 가정한다.

2.1 CPU 모델[1,10]

CPU 모델은 다음과 같은 사항을 가정을 한다.

1) 스케줄링 방식은 선점형 고정 우선순위 스케줄링(preemptive fixed priority scheduling)[1] 방식이다.
2) 공유자원에 대한 처리는 우선순위 상한 프로토콜(priority ceiling protocol)[11]을 이용한다.

각 태스크(i)의 WCRT(Worst Case Response Time; R_i)는 식 (1)과 같이 표현된다. 여기서 J_i는 태스크 i의 지터로 태스크 i가 실행되기까지 걸리는 가장 긴 시간과 가장 짧은 시간의 차이다. 식 (1)에서 ω_i는 시간창(time window)이라 하며 하나의 태스크가 실행을 시작한 후 마치기까지 걸리는 최대 시간을 의미한다. 이 값은 식 (2)에서 보는 바와 같이 태스크의 WCET(Worst Case Execution Time; C_i)와 공유자원에 대한 처리(B_i)나 우선순위가 높은 태스크의 수행으로 인한 지연시간을 합한 값으로 표현된다. 여기서 T_j는 태스크 j의 주기를 나타내며 $hp(i)$는 태스크

i보다 우선순위가 높은 모든 태스크들을 나타낸다.

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

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

식 (2)는 CPU의 컨텍스트 스위칭(context switching)과 스케줄링에 필요한 시간 등의 오버헤드(overhead)[12]를 고려하지 않은 것이다. 이번 연구에서는 CPU에 대한 좀더 현실적인 해석을 수행하기 위해서 시스템의 오버헤드 뿐만 아니라 임의 마감시간(arbitrary deadline)[3]과 시간 오프셋(time offset)[4]을 고려한 시간 해석 알고리즘을 구현 하였다. 첫째, 임의 마감시간을 고려한 해석 알고리즘은 마감시간이 주기보다 큰 태스크를 갖는 CPU에 대한 시간 해석을 가능하게 한다. 둘째, 시간 오프셋을 고려한 해석 알고리즘은 특정 태스크들이 항상 일정한 시간 간격을 두고 수행되는 경우에 이를 고려함으로써 현존하는 해석보다 좀더 실제적인 해석을 가능하게 한다.

2.2 네트워크 모델

현재 실시간 분산 제어시스템 분야에는 너무 다양한 네트워크 프로토콜이 존재하기 때문에 선진국에서는 적용분야별로 표준화를 서두르고 있다. 특히 자동차 분야에서는 CAN 프로토콜이 네트워크 프로토콜의 표준으로 자리잡아가고 있으며, 실제 자동차 산업에서도 가장 많이 사용되고 있다. 또한 차량용 저속네트워크로는 LIN 프로토콜이 국제 표준으로 제시되었다. RAT은 이와 같은 산업추세를 반영하여 CAN과 LIN 프로토콜에 대한 해석이 가능하도록 설계되었다.

2.2.1 CAN 프로토콜[2,13]

CAN 프로토콜은 다중-마스터/다중-슬레이브(mult-master/mult-slave) 형태로 최대 1Mbit/sec의 통신 속도를 제공하며 다양한 오류-검출 기능들은 네트워크 데이터의 신뢰성을 높여준다.

CAN 메시지는 47비트의 기본적인 오버헤드와 최대 8byte까지의 유동적인 데이터 프레임으로 구성된다. 또한 CAN 프로토콜에서 s바이트의 데이터 프레임을 갖는 메시지의 최대 스터프 비트(stuffbits)는 식 (3)과 같이 표현된다. 이때 하나의 비트를 전송하는데 소요되는 시간을 τ_{bit}이라고 정의하면 전체 메시지를 CAN버스를 통해 전송할 경우 걸리는 최대 시간(C_i)은 식 (4)와 같이 표현된다.

$$(Stuffbits)_{\max} = \left\lfloor \frac{34 + 8s_i - 1}{4} \right\rfloor \tag{3}$$

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

이때 CAN 메시지(i)에 대한 WCRT는 CPU해석에서와 유사한 형태를 가지며 식 (5)와 같이 표현된다. 여기서 J_i는 메시지 i의 지터이며, w_i는 메시지 i의 대기시간(queuing time)으로 우선순위가 i보다 높은 메시지들의 방해 없이 메시지의 첫 번째 비트가 전송되는데 까지 걸리는 가장 긴 시간을 의미한다. 메시지의 대기시간 w_i는 식 (6)과 같이 표현되며, 여기서 T_j는 메시지 j의 주기를 나타내고 $hp(i)$는 메시지 i보다 우선순위가 높은 모든 메시지들을 나타낸다.

$$R_i = J_i + w_i + C_i \tag{5}$$

$$w_i = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_i + \tau_{bit}}{T_j} \right\rceil C_j \tag{6}$$

2.2.2 LIN 프로토콜[6,14]

LIN 프로토콜은 CAN 프로토콜의 대역폭과 다기능이 필요하지 않은 액츄에이터와 스마트 센서를 위한 저비용의 통신을 가능하게 한다. 한 개의 마스터 노드와 한개 혹은 그 이상의 슬래이브 노드로 이루어지는 단일 마스터-다중 슬래이브(single-master/multi-slave) 형태로 최대 20Kbit/s 통신 속도를 제공한다.

LIN 메시지에 대한 WCRT는 식 (7)과 같이 표현할 수 있다.

$$R_i = J_i + C_i \tag{7}$$

여기서 J_i는 메시지 i의 지터이다. C_i는 메시지 프레임의 전송시간으로 LIN 메시지 프레임은 제어 프레임(header frame)과 응답 프레임(response frame)으로 구성되어 있기 때문에 식 (8)과 같이 프레임의 전송시간을 두 부분으로 나누어 정의할 수 있다. 여기서 T_{header}는 제어 프레임의 전송시간이고, $T_{response}$는

응답 준비공간을 포함한 응답프레임의 전송시간을 의미한다.

$$C_i = T_{frame} = T_{header} + T_{response} \quad (8)$$

제어 프레임의 전송시간은 동기화 시작 필드와 동기화 필드, 아이디 필드의 시간 합으로 식 (9)와 같이 표현된다. 여기서 P_{SYNBRK}와 P_{SYNDEL}은 LIN프로토콜의 시간 제약조건을 고려하여 얻은 시간상수로 각각 버스의 상태가 $13\tau_{bit}$과 $1\tau_{bit}$을 초과하여 저 위상 상태로 유지되는 시간이다.

$$T_{header} = 34 \times \tau_{bit} + P_{SYNBRK} + P_{SYNDEL} \quad (9)$$

응답 프레임의 전송시간은 응답 준비 공간의 시간과 나머지 응답 프레임에 대한 전송시간의 합으로 식 (10)과 같이 표현된다. 여기서 $P_{INFRAME}$은 응답 준비 공간에 대한 시간을 나타내고 $P_{INTERBYTE}$는 필드 사이의 시간간격을 나타낸다.

$$T_{response} = P_{INFRAME} + 10\tau_{bit} \times (s_m + 1) + P_{INTERBYTE} \times s_m \quad (10)$$

식 (8), (9)와 (10)을 이용하여 메시지 프레임의 전송시간에 대한 식을 정의하면 식 (11)과 같다. LIN 프로토콜의 시간제약조건을 이용하면 식 (12)와 같이 메시지 프레임의 최대 허용 시간을 구할 수 있는데 이는 식 (11)에서 시간 상수 값을 정의할 수 없을 때 이용한다.

$$C_i = (44 + 10 \times s_m) \times \tau_{bit} + P_{SYNBRK} + P_{SYNDEL} + P_{INFRAME} + P_{INTERBYTE} \times s_m \quad (11)$$

$$C_{i\,MAX} = (45 + 10 \times s_m) \times \tau_{bit} \times 1.4 \quad (12)$$

2.3 Holistic 해석 모델[2,5]

실시간 분산 제어시스템에서 메시지의 응답시간 송신 노드의 응답시간에 영향을 받고 수신 노드의 응답시간은 메시지의 전송지연에 영향을 받는다. 따라서 노드간의 통신과 태스크의 응답시간에 대한 영향을 모두 고려한 시간해석이 이루어져야 하는데 이러한 시간 해석을 holistic 해석이라 한다.

각 노드가 네트워크 메시지를 이용하여 자료를 공유할 경우 총 응답시간은 Fig. 1과 같이 송신 노드에서 메시지 전송 시작까지의 응답시간(Tx node response time)과 메시지 전송에 걸리는 시간(Communication), 그리고 수신 노드에서 메시지를 수신하는데 걸리는 시간(Rx)과 이 메시지를 이용하여 실제 출력을 내보내는데 걸리는 시간(Actuation)의 상호 관계를 고려하여 얻을 수 있다.[15,16]

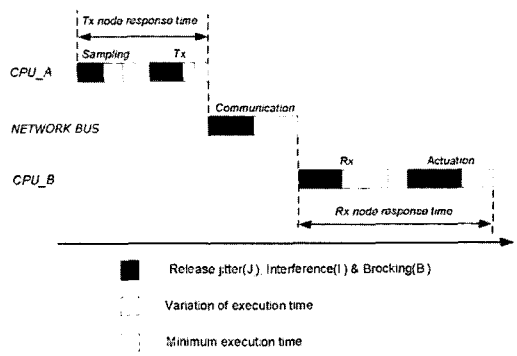

Fig. 1 Holistic timing analysis

네트워크 메시지의 지터(J_i)는 메시지를 전송하는 태스크에 의하여 야기되는 것으로 식 (13)과 같이 송신 태스크($send(i)$)의 WCRT($R_{send(i)}$)와 같다. 메시지를 받는 태스크의 지터는 전송되는 메시지에 의해 야기되는 것으로 CAN 프로토콜에 의해 전송된 메시지를 수신하는 태스크의 지터($J_{CAN\_dest(i)}$)는 메시지가 가장 늦게 도착하는 시간인 메시지의 응답시간에서 메시지가 가장 빨리 도착하는 시간인 $47\tau_{bit}$을 뺀 값으로 식 (14)와 같다. LIN 프로토콜에 의해 전송된 메시지를 수신하는 태스크의 지터($J_{LIN\_dest(i)}$)는 메시지가 가장 빨리 도착하는 시간이 $64\tau_{bit}$이기 때문에 식 (15)와 같이 표현된다. 따라서 송신 노드의 모든 태스크들의 실행과 메시지의 시간지연을 포함하여 수신 태스크가 메시지를 수신할 때까지 걸리는 총 응답시간은 식 (5), (11)과 (13)을 이용해 CAN 프로토콜의 경우 식 (16)과 같이, LIN 프로토콜의 경우 식 (17)과 같이 표현할 수 있다. 여기서 w_i와 $w_{dest(i)}$는 각각 메시지의 대기시간과 수신 태스크의 시간창을 나타낸다.

$$J_i = R_{sned(i)} \quad (13)$$

$$J_{CAN\_dest(i)} = R_i - 47\tau_{bit} \quad (14)$$

$$J_{LIN\_dest(i)} = R_i - 64\tau_{bit} \quad (15)$$

$$R_{CAN\_endtoend} = R_{send(i)} + w_i + C_i + w_{dest(i)} \quad (16)$$

$$R_{LIN\_endtoend} = R_{send(i)} + C_i + w_{dest(i)} \quad (17)$$

3. RAT(Response-time Analysis Tool)

RAT은 MATLAB기반의 시간해석도구이다. 개발자는 RAT을 이용하여 CPU와 네트워크를 해석 가능한 시간 모델로 표현할 수 있다. 또한 시간 해석 알고리즘이 소프트웨어로 구현되어 있기 때문에 시스템을 효과적으로 해석하고 설계할 수 있다.

3.1 자원(Resources)

CPU는 몇 가지 시간 속성인자로 표현되는 태스크들로 구성되며 공유자원에 대한 처리를 해야 할 경우에는 "Semaphore"를 정의한다. RAT에서 "Semaphore"는 각 태스크가 공유자원을 점유하는 시간을 나타낸다. 또한 시간 오프셋을 고려한 시간 해석을 수행하기 위해서는 "Transaction"을 정의 한다. "Transaction"에 속하는 태스크는 동일한 주기를 가져야 하며 "Offset"과 "Every"로 표현된다. Fig. 2는 RAT에서 표현되는 CPU의 구조와 시간속성인자를 나타낸다.

CPU와 같이 네트워크도 시간 속성인자를 갖는 메시지들로 구성된다. Baud rate는 CAN과 LIN프로토콜에서 제안하는 버스 속도 범위에서 시뮬레이션이 가능하게 한다. Fig. 3과 4는 각각 CAN 과 LIN 프로토콜의 구조와 시간속성인자를 나타낸다.

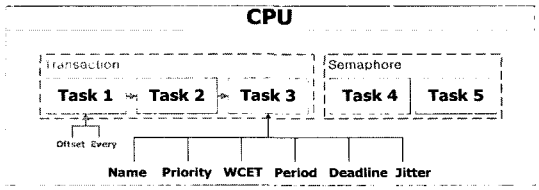

Fig. 2 CPU organization and its timing attributes

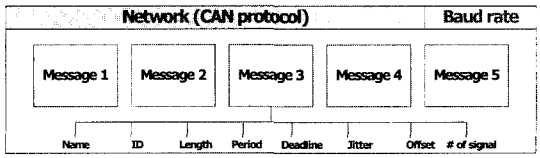

Fig. 3 CAN protocol organization and its timing attributes

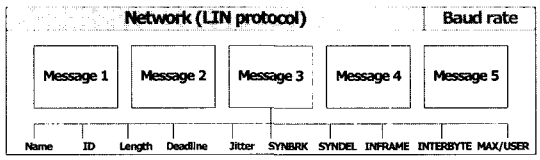

Fig. 4 LIN protocol organization and its timing attributes

Table 1은 태스크와 CAN, LIN 메시지를 모델링하기 위해 필요한 모든 시간속성인자들을 정리한 것이다.

3.2 시간해석

시간해석을 수행하기 위해서는 앞 절에서 정리한 것과 같은 시간속성인자들을 시스템 파일이나 사용자환경(GUI)을 이용하여 RAT에 입력 한다. RAT은 이와 같이 입력된 태스크와 CAN, LIN 메시지들의 시간속성을 스프레드시트(spreadsheet)형식으로 표시하기 때문에 그 값을 쉽게 변경할 수 있다. 따라서

Table 1 Timing attributes organizing tasks and messages

Task		CAN message		LIN message	
Name	Name of task	Name	Name of message	Name	Name of message
ID	Identifier of task	ID	Identifier of message	ID	Identifier of message
WCET	Worst case execution time	Length	Length of message	Length	Length of message
Period	Period of task	Period	Period of message	Deadline	Deadline of message
Deadline	Deadline of task	Deadline	Deadline of message	Jitter	Jitter of message
Jitter	Jitter of task	Jitter	Jitter of message	SYNBRK	Synchronization break
Offset	Offset of task in transaction	Offset	Offset of message	SYNDEL	Synchronization delimiter
Every	Every of task in transaction	# of signal	Number of signal	IN-FRAME	In frame response space
Sema-phore	Time to be possessed by semaphore			INTER-BYTE	Interbyte space
				MAX/USER	Max or User parameter value

개발자는 반복해서 시간해석을 수행할 수 있으며 가능한 최적의 설계 사양을 얻을 수 있다.

Holistic 해석을 수행하기 위해서는 선행관계(precedence relationship)를 정의 한다. 선행관계는 제어과정(control process)을 일련의 태스크와 메시지들로 구성한 것이다. RAT은 Table 2와 같이 제어 흐름에 관여하는 태스크나 메시지를 실행순서에 따라 선택함으로써 선행관계를 구성한다.

Table 2 Precedence relationship

Name of Precedence relationship	1st object name 2nd object name ... ith object name	1st component name 2nd component name ... ith component name

Holistic 해석결과는 선행관계가 올바르게 구성되어진 경우에는 입력에서 출력까지의 총 응답시간(end-to-end response time)이며 선행관계가 올바르게 구성되어 있지 않은 경우에는 시스템 개발자가 선행관계를 쉽게 수정할 수 있도록 지침을 제시한다.

3.3 사용자 환경(Graphic User Interface)

RAT의 메인윈도우(Main window)는 Fig. 5와 같이 Manu frame과 Task(or message) frame으로 나뉜다. 시스템 개발자가 해석에 필요한 시간속성인자를 입력하면 태스크나 메시지의 속성이 Task(or message) frame에 표시된다.

Holistic 해석을 수행하기 위해서는 Constraint 메뉴를 이용한다. Constraint 메뉴는 Precedence 창을 생성하고 선행관계를 입력하거나 삭제할 수 있는 기능을 제공한다.

RAT은 현재 작업 중인 시스템을 텍스트 형식의 시스템 파일로 저장할 수 있으며, 반대로 텍스트 형식으로 저장되어 있는 시스템 파일을 불러올 수 있는 기능을 제공한다. 또한 네트워크 시스템의 데이터베이스를 관리하는데 가장 널리 사용되는 Vector사의 CANdb 편집기[17]와의 자료 공유를 지원한다. RAT은 CANdb 파일에서 해석에 필요한 네트워크 데이터를 직접 가져올 수 있는 기능을 지원한다.

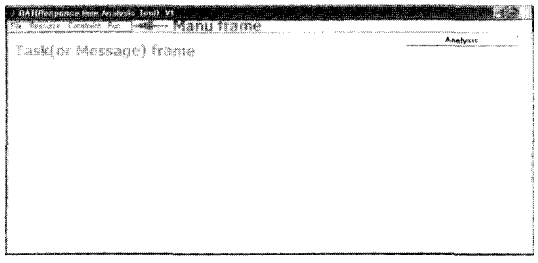

Fig. 5 Main window of RAT

태스크와 CAN, LIN 메시지에 대한 해석결과는 Fig. 6, 7, 8과 같이 Analysis results of Tasks, Analysis results of CAN messages, Analysis results of LIN messages 윈도우에 표시된다. Analysis results of Tasks 윈도우는 공유자원 처리로 인한 지연(Blocking time), 태스크의 응답시간(Response time)과 스케줄 가능여부(Schedulability)를 표시한다. Analysis results of CAN(or LIN) messages 윈도우는 메시지의 응답시간(Response time)과 스케줄 가능여부(Schedulability)를 표시한다. 각 윈도우의 Utilization은 CPU나 버스의 사용률을 나타내며 Holistic 해석 결과는 Analysis results of Tasks 윈도우의 아래 부분에 표시된다.

Response time of Tasks

Task name	Deadline	Blocking time	Response time	Schedulability
	Properties of the CPU, CPU1[ms]			
A	20	3	11	Schedulable
B	20	3	15	Schedulable
C	30	3	22	Schedulable
D	40	3	28	Schedulable
E	40	0	31	Schedulable
Utilization	0.62			

Fig. 6 Analysis results of Tasks window

Response time of CAN messages

Message	Deadline	Response time	Schedulability
	Propertis of the CAN bus, CAN1[ms]		
MSG1	20	11.2	Schedulable
MSG2	20	16.4	Schedulable
MSG3	20	16.4	Schedulable
Utilization:	0.62		

Fig. 7 Analysis results CAN messages window

Response time of LIN messages

Message	Deadline	Response time	Schedulability
	Propertis of the LIN bus, LIN1[ms]		
LIN_msg1	15	7.4997	Schedulable
LIN_msg2	10	8.4083	Schedulable
LIN_msg3	20	8.8887	Schedulable

Fig. 8 Analysis results of LIN messages window

4. 시뮬레이션

RAT의 실용성 및 응용성을 검증하기 위하여 차체 네트워크 시스템을 대상 시스템으로 시간해석을 수행한다.

4.1 시스템 구성[12)]

대상 시스템은 모토로라 OSEK을 이용하여 설계한 OSEK/VDX 표준을 만족하는 시스템으로 Fig. 9와 같이 4개의 CAN 노드와 20개의 LIN 노드로 구성된 차체 네트워크 시스템이다. 4개의 CAN 노드는 CAN 버스를 통해서 데이터를 공유하며 DF Side (Driver Front side) 노드에서 메시지를 전송하고 나머지 노드(PF side: Passenger Front side, DR side: Driver Rear side, PR side: Passenger Rear side) 에서는 수신하는 형태이다. 또한 모든 CAN 노드는 LIN 버스로 연결된 LIN 노드와의 관계에서 주제어 노드 역할을 수행하며 모든 LIN노드는 종속 제어 노드가 된다.

이 시스템에서 컨텍스트 스위칭으로 인한 오버헤드(C_{su})는 20μs이고 스케줄링에 필요한 시간(C_{timer})은 전체 태스크의 실행시간에 비해 매우 작기 때문에 0으로 간주한다. 또한 CAN 노드의 태스크들은 Table 3과 같은 시간 속성을 가진다. 여기서 COM 태스크는 수신 메시지를 애플리케이션 태스크가 접근할 수 있도록 물리 계층(physical layer)에서 CAN 드라이버의 버퍼로 옮겨오는 역할을 한다.

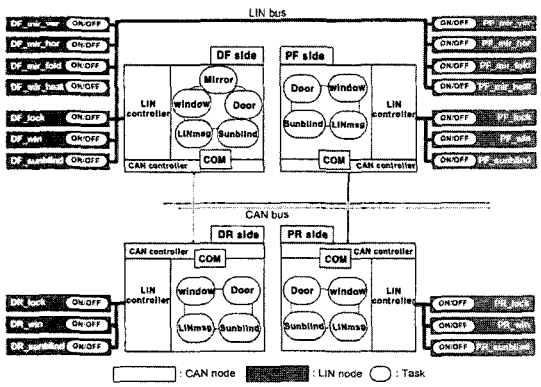

Fig. 9 Vehicle body network system

모든 LIN 노드는 주기가 100ms인 ON/OFF 태스크로 구성된다. 각 LIN 노드는 하나의 ON/OFF 태스크로 구성되는데 ON/OFF 태스크는 주제어 노드로부터 LIN 메시지를 수신한 후 액추에이터에 필요한 입력을 제공하는 유사한 기능을 수행한다. 또한 모든 태스크의 WCET 도 0.05179ms 로 같은 값을 가진다.

4.2 시뮬레이션 결과

Table 4와 5는 네트워크 통신을 고려하지 않았을 때 태스크와 메시지들의 응답시간을 나타낸 것이다. 해석결과 모든 태스크와 메시지들의 응답시간이 마감시간보다 작은 값을 나타냈다. 여기서 RAT의 해석 알고리즘은 응답시간이 최대로 걸리는 임계상황을 가정하여 시간해석을 수행하기 때문에 이 결과로 시스템의 정상 동작을 보장할 수 있다. 그러나 만약 대상 시스템이 마감시간을 만족하지 않을 경우에는 시스템의 안전성을 보장할 수 없으며 이는 시스템에 위험한 상황을 초래할 수도 있다. 따라서 개발자는 시스템이 마감시간을 만족할 때까지 자원의 시간속성을 바꿔가며 시간해석을 계속해야 한다.

Holistic 해석은 Table 6과 같은 다섯 가지 경우에 대해서 해석을 수행한다. Mir_DF2PF는 DF side 노

Table 3 Attributes of the tasks in each CPU [ms]

Name of Task(ID)	DF side	PF side	DR side	PR side
	WCET(Period)	WCET(Period)	WCET(Period)	WCET(Period)
LINmsg(20)	0.10359(15)	0.14243(15)	0.14243(15)	0.14243(15)
Door(21)	0.32370(50)	0.27191(50)	0.27191(50)	0.27191(50)
Window(22)	0.59561(100)	0.31076(100)	0.31076(100)	0.31076(100)
Mirror(24)	0.55677(100)	X	X	X
Sunblind(25)	0.11653(100)	0.20717(100)	0.20717(100)	0.20717(100)
COM(28)	0.00300(20)	0.00300(20)	0.00300(20)	0.00300(20)

Table 4 WCRT of the tasks in each CPU [ms]

Name of Task	WCRT(Deadline)				
	DF side	PF side	DR side	PR side	LIN node
LINmsg	0.14359(15)	0.18243(15)	0.18243(15)	0.18243(15)	X
Door	0.50729(50)	0.49434(50)	0.31191(50)	0.49434(50)	X
Window	1.1429(100)	0.8451(100)	0.8451(100)	0.62267(100)	X
Mirror	1.5161(100)	X	X	X	X
Sunblind	2.493(100)	1.0923(100)	1.0923(100)	1.0923(100)	X
COM	2.3794(20)	1.1353(20)	0.56093(20)	0.52208(20)	X
ON/OFF	X	X	X	X	0.5579(100)

Table 5 WCRT of the CAN and LIN messages [ms]

CAN messages			LIN messages		
Name of message	ID (length[byte])	WCRT (Deadline)	Name of message	ID (length[byte])	WCRT (Deadline)
Lock_msg	0x02(1)	1.04(50)	Input_msg	0x1f(2)	9.4792(100)
Sunblind_msg	0x10(1)	1.56(100)	DF_mirhor_msg	0x03(2)	9.4792(100)
PF_win_msg	0x09(1)	2.08(100)	DF_mirver_msg	0x01(2)	9.4792(100)
DR_win_msg	0x08(1)	2.6(100)	Mirinput_msg	0x1e(2)	9.4792(100)
PR_win_msg	0x0a(1)	2.6(100)	PF_mirhor_msg	0x04(2)	9.4792(100)
			PF_mirver_msg	0x02(2)	9.4792(100)
			Winsta_msg	0x00(2)	9.4792(100)

Table 6 End-to-end WCRT for some control process [ms]

Precedence relationship	Control process Task or Message (Node or Bus)	WCRT [ms] (SUM [ms])
Mir_DF2PF	Mirror(DF) → LINmsg(DF) → Input_msg(LIN) → ON/OFF(PF_mir_x)	14.6528 (11.69679)
Win_DF2PF	Window(DF) → PF_win_msg(CAN) → COM(PF) → Window(PF) → LINmsg(PF) → Input_msg(LIN) → ON/OFF(PF_win)	20.6514 (15.42283)
Win_DF2PR	Window(DF) → PR_win_msg(CAN) → COM(PR) → Window(PR) → LINmsg(PR) → Input_msg(LIN) → ON/OFF(PR_win)	21.469 (15.1059)
Door_DF2DR	Door(DF) → Lock_msg(CAN) → COM(DR) → Door(DR) → LINmsg(DR) → Input_msg(LIN) → ON/OFF(DR_lock)	17.2289 (12.63966)
Sunblind_DF2PF	Sunblind(DF) → Sunblind_msg(CAN) → COM(PF) → Sunblind(PF) → LINmsg(PF) → Input_msg(LIN) → ON/OFF(PF_sunblind)	24.4563 (16.50013)

드에서 외부로부터 받은 사이드 미러 입력이 LIN 버스를 통해 PF_mir_x 노드까지 전달되어 실제 사이드 미러가 작동되는데 까지 걸리는 시간을 정의한다. 또한 Win_DF2PF와 Win_DF2PR는 각각 DF side 노드에서 외부로부터 받은 윈도우 입력이 CAN 버스를 통해 PF side와 PR side 노드에 전달되고, 다시 LIN 버스를 통해 PF_win와 PR_win 노드에 전달되어 실제 윈도우가 작동되는데 까지 걸리는 시간을 정의한다. Door_DF2DR은 DF side 노드에서 외부로부터 받은 문 잠금 입력이 CAN 버스를 통해 DR side 노드에 전달되고, 다시 LIN 버스를 통해 DR_lock 노드에 전달되어 실제 문 잠금 장치가 작동되는데 까지 걸리는 시간을 정의한다. 마지막으로 Sunblind_DF2PF는 DF side 노드에서 외부로부터 받은 선블라인드 입력이 CAN 버스를 통해 PF side 노드에 전달되고, 다시 LIN 버스를 통해 PF_sunblind 노드에 전달되어 실제 선블라인드 장치가 작동되는데 까지 걸리는 시간을 정의한다.

시뮬레이션 결과 통신을 고려한 시간 해석을 수행한 경우에는 입력에서 출력까지 걸리는 총 응답

시간이 단순히 제어 과정에 관여하는 태스크나 메시지의 WCRT를 합한 것보다 큰 값을 나타냈다. 예를 들어 Mir_DF2PF에 관여하는 태스크와 메시지들의 응답시간의 총합은 Table 4와 5를 이용하여 얻을 수 있는데 그 값은 11.69679(ms)로 시뮬레이션 결과 얻은 총 응답시간 14.6528(ms)보다 작은 값임을 알 수 있다. 나머지 경우에 대해서도 Table 6의 WCRT와 SUM 값을 비교해 보았을 때 같은 결과를 얻을 수 있다. 이것은 2.3장에서 보았듯이 제어 흐름에 관여하는 태스크나 메시지의 응답시간이 다음 태스크나 메시지의 지터로 작용하기 때문인 것을 알 수 있다. 따라서 개발자는 시스템을 개발할 때 태스크와 메시지의 개별적인 시간 해석 뿐만 아니라 통신을 고려한 전체 제어흐름에 대한 시간해석을 수행해야 한다. 만약 시스템의 시간지연이 통신에 의해서 발생되는 것이라면 메시지의 ID를 새롭게 정의하거나 전송될 신호를 다르게 패킹(packing)하여 메시지에 발생하는 부하를 줄일 수 있으며 가능한 경우에는 버스의 속도를 조정함으로써 문제를 해결할 수 있다. 그러나 이 문제가 태스크에 의한 것이라면 태스크의 계산 시간을 줄이기 위해 프로그램을 시간에 대하여 최적화 시키거나, 태스크의 구조를 바꾸어 문제를 해결할 수 있다.

5. 결 론

이번 연구에서는 실시간 분산 제어시스템의 시간에 대한 응답 특성을 효과적으로 해석할 수 있는 도구로 RAT을 제시하였다. RAT은 응답시간이 최대로 걸리는 임계상황을 가정하여 시간해석을 수행하도록 설계되었기 때문에 개발자는 시간해석을 통해서 시스템의 안전성을 보장할 수 있고 시간 속성 인자를 바꿔가면서 반복적인 해석을 수행함으로써 시스템을 설계하기 위한 기본적인 가이드라인을 얻을 수도 있다. 특히 RAT은 복잡한 해석 알고리즘을 소프트웨어로 구현함으로써 제어기 설계자도 쉽게 시스템의 응답 특성을 해석하고 이를 고려하여 시스템을 설계 할 수 있도록 하였다. 실제로 이 연구에서는 차체 네트워크 시스템을 대상으로 시간해석을 수행해 봄으로써 RAT의 실용성 및 응용성을 검증하였다.

시뮬레이션 결과 태스크와 메시지의 응답시간을 얻을 수 있었는데 이를 마감시간과 비교하여 시스템의 정상 작동 여부를 판단할 수 있었다. 또한 실시간 분산 제어 시스템은 제어 흐름에 관여하는 태스크나 메시지가 다음 태스크나 메시지의 응답시간에 영향을 미치기 때문에 개발자는 시스템을 개발할 때 태스크와 메시지의 개별적인 시간 해석뿐만 아니라 통신을 고려한 전체 제어흐름에 대한 시간해석을 수행해야 함을 알 수 있었다. 결론적으로 RAT을 활용하면 시스템을 실제로 구현하기 이전에 몇 가지 시간 속성 인자를 이용하여 시스템의 시간에 대한 응답 특성을 해석 할 수 있는데 이는 시스템의 개발비용과 개발기간을 혁신적으로 감소시킬 수 있도록 한다.

References

1) C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the Association for Computing Machinery, Vol. 20, No.1, pp.46-61, 1973.
2) K. Tindell, "Real Time System by Fixed Priority Scheduling," Ph.D Thesis, Department of Computer Science, University of York, 1994.
3) K. Tindell, A. Burns and A. Wellings, "An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks," Real-time Systems, Vol.6., pp.133-151, 1992.
4) K. Tindell, "Using Offset Information to Analyze Static Priority Pre-emptively Scheduled Task Sets," YCS 182, Department of Computer Science, University of York, 1994.
5) K. Tindell and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-time Systems," Microprocessors and Microprogramming, pp.117-134, 1994.
6) J. M. Youn, M. S. Shin, W. T. Lee and M. Sunwoo, "A Study on Timing Model and Analysis of LIN Protocol," 2003 KSAE Spring Conference Proceeding, pp.952-957, 2003.
7) J. W. S. Liu, J. L. Redondo, Z. Deng, T. S. Tia,

R. Beattati, A. Silberman, M. Storch, R. Ha and W. K. Shih, "PERTS:A Prototyping Environment for Real-Time Systems," Proceedings of the 14th IEEE Real-Time Systems Symposium, pp.184-188, 1993.

8) H. Hansson, H. Lawson, O. Bridal, C. Eriksson, S. Larsson, H. Lon and M. Stromberg, "BASEMENT: An Architecture and Methodology for Distributed Automotive Real-Time Systems," IEEE Transaction on Computers, Vol.46, No.9, pp.1016-1027, 1997.
9) M. Gonzalez Harbour, J. J. Gutierrez Gacia, J. C. Palencia Gutierrez and J. M. Drake Moyano, "MAST: Modeling and Analysis Suite for Real Time Application," IEEE 13th Euromicro Conference on Real-Time Systems, pp.125-134, 2001.
10) M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," BCS Computer Journal, Vol.29, No.5, pp.390-395, 1986.
11) L. Sha, R. Rajkumar and J. Lehoczky, "Priority Inheritance Protocol: An Approach to Real-Time Synchronization," IEEE Transactions on Coputers, Vol.39, No.9, pp.1175-1185, 1990.
12) M. S. Shin, W. T. Lee and M. Sunwoo, "Holistic Scheduling Analysis of a CAN based Body Network System," Transactions of KSAE, Vol.10, No.5, pp.114-120, 2002.
13) CAN Specification, Version 2.0, Sep, 1991.
14) LIN Specification, Revision 1.2, Nov. 17, 2002.
15) H. Lonn and J. Axelsson, "A Comparison of Fixed-Priority and Static Cyclic Scheduling for Distributed Automotive Control Applications," 11th Euromicro Conference on Real-time Systems, 1999.
16) M. Torngren, "Fundamentals of Implementing Real-Time Control Applications in Distributed Computer Systems," Real-time Systems, Vol. 14, pp.219-250, 1998.
17) CANoe User's Manual V4.1.
18) C. M. Krishna and K. G. Shin, Real-time Systems, McGraw-Hill, New York, pp.1-137, 1997.
19) G. C. Buttazzo, Hard Real-time Computing System: Predictable Scheduling Algorithms and Applications, pp.1-361, KLUWER Academic Publishers, USA, 1997.
20) J. Larsson, "SCHEDULITE: A Fixed Priority Scheduling Analysis Tool," MSc Thesis, Department of Computer Systems Uppsala Univeristy, 1998.
21) S. Chatterjee, K. Bradley, J. Madriz, J. A. Colquist and J. Strosnider, "SEW: A Toolset For Design and Analysis of Distributed Real-Time Systems," IEEE Real-Time Technology and Applications Symposium, pp.72-77, 1997.
22) V. Braberman, "On Intergrating Scheduling Theory into Formal Models for Hard Real Time Systems," Workshop Formal Methods for the Design of Real-Time Systems, 1997.
23) W. Lawrenz, CAN System Engineering From Theory to Practical Applications, pp.1-453, Springer, Germany, 1997.
24) Motorola, HC12 OSEK Operating System User's Manual Rev.1.7.
25) Motorola, OSEK COM/NM User's Manual Rev. 1.0.