International Journal of Automotive Technology, Vol. 5, No. 4, pp. 269–276 (2004)

1229-9138/2004/018-05

DEVELOPMENT OF TIMING ANALYSIS TOOL FOR DISTRIBUTED REAL-TIME CONTROL SYSTEM

J. B. CHOI, M. S. SHIN and M. SUNWOO*

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

(Received 29 September 2003; Revised 16 April 2004)

ABSTRACT–There has been considerable activity in recent years in developing timing analysis algorithms for distributed real-time control systems. However, it is difficult for control engineers to analyze the timing behavior of distributed real-time control systems because the algorithms was developed in a software engineer's position and the calculation of the algorithm is very complex. Therefore, there is a need to develop a timing analysis tool, which can handle the calculation complexity of the timing analysis algorithms in order to help control engineers easily analyze or develop the distributed real-time control systems. In this paper, an interactive timing analysis tool, called RAT (Response-time Analysis Tool), is introduced. RAT can perform the schedulability analysis for development of distributed real-time control systems. The schedulability analysis can verify whether all real-time tasks and messages in a system will be completed by their deadlines in the system design phase. Furthermore, from the viewpoint of end-to-end scheduling, RAT can perform the schedulability analysis for series of tasks and messages in a precedence relationship.

KEY WORDS : Distributed real-time control system, Worst case execution time (WCET), Worst case response time (WCRT), Schedulability, Holistic analysis

1. INTRODUCTION

Real-time control systems are the primary components of complex time-critical systems such as nuclear reactors, flight-control systems, and automotive applications. For such applications, issues of safety and reliability are particularly important since a failure may result in catastrophic destruction of property and life. Fortunately, many timing analysis algorithms, which allow us to verify the real-time control systems in a reliable and predictable fashion before the implementation phase, have been developed. The WCRT (Worst Case Response Time) analysis algorithm (Liu and Layland, 1973; Tindell, 1994) is one of the most general algorithms that analyze timing behavior of a distributed real-time control system. This algorithm can guarantee the safety of the system by schedulability analysis at critical instance. Schedulability analysis is to decide whether the system is feasible or not by comparing WCRT of tasks or messages with their deadlines. However, this algorithm cannot be simply used to analyze the system without complex calculations, and it is almost impossible to analyze the timing behaviors of large-scale distributed real-time control system by hand. Beside, it is difficult for control engineer to understand the algorithm because it was developed in a software engineers position.

In order to solve this problem, RAT (Response-time Analysis Tool), a timing analysis tool for analysis of distributed real-time control systems, is proposed in this research. This timing analysis tool is based on MATLAB and provides easy way for control engineer to analyze the system by encapsulating the calculation complexity of the timing analysis algorithms.

RAT can perform not only individual but also end-to-end schedulability analysis for a defined set of tasks, or messages based on WCRT analysis algorithm. Especially, because RAT make a user can express any control process as precedence relationship, any control engineer can obtain timing behavior of the system easily in his position. By performing schedulability analysis with varying CPU and network characteristics before the implementation phase, the safety of the system in critical instance can be guaranteed, and ultimately development time and cost can be reduced.

The paper proceeds as follows: Section 2 presents the overview of the real-time computation model used in this research. Section 3 presents the objectives of RAT and how it is used. Section 4 applies RAT to the vehicle body network system in order to prove the feasibility of this tool. Finally, section 5 offers some conclusions and

*Corresponding author. e-mail: msunwoo@hanyang.ac.kr

introduces some future researches.

2. COMPUTATIONAL MODEL AND ASSUMPTIONS

Real-time systems are classified as hard real-time systems or soft real-time systems. Here, only hard real-time systems, which must generate the right output at the right time, are considered. For simplicity, it is assumed that all tasks and messages have periodic requests and constant run-times. If the periodic request and the constant run time are not available, the shortest request interval and the longest run time can be used. A scheduling algorithm is a set of rules that determine the task or message to be executed at a particular moment. The scheduling algorithm considered in RAT is a fixed scheduling algorithm in which priorities are assigned to tasks or messages once and for all. For predictability of the system, this scheduling algorithm has been generally used.

2.1. CPU Model

It is assumed that a scheduler provides pre-emptive priority scheduling based on dispatching. It is also assumed that each task may lock and unlock semaphores according to the priority ceiling protocol, which was discovered by Sha *et al.* (1990), to avoid concurrency problems. Tindell and Clark (1994) suggested the equation (1) to obtain the WCRT(R_i) for a given task (i) where J_i is jitter of task i. Here, w_i is the *time window*, which is the longest time from the activation of a task to the end of execution. This is obtained by summing up worst case execution time (C_i), blocking time (B_i), and delay time caused by preemption of higher priority tasks, where T_j is period of task j, $hp(i)$ is the set of all tasks of higher priority than task i (Equation 2).

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

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

For a more realistic analysis, two extendible analysis approaches are implemented in RAT. One is an arbitrary deadline from Tindell *et al.* (1992), through which the system where task periods less than task deadlines can be analyzed. The other is an time offset from Tindell (1994). With this approach, an analysis which is much less pessimistic than the extant analysis can be performed.

2.2. Network Model

The increasing demands for communication in a distributed real-time control systems have led to the specification and existence of various network protocols. However, as a matter of convenience, some international standards have been established. CAN protocol, which stands for Controller Area Network, is one of the current standards for the automotive industry. In this situation, the timing analysis algorithm for CAN protocol is implemented in RAT. And this will be extended to other network protocols in the future.

Tindell (1994) and Tindell *et al.* (1994) developed the analysis model for CAN protocol. A CAN message is organized with basic 47bit overhead and 0 to 64 bit data frame. The maximum stuffbits of the message which has s bytes data frame is expressed like equation (3). Then, the maximum time (C_i) to transmit whole message through a CAN bus can be obtained by equation (4) where τ_{bit} is the time to transmit a 1bit CAN message.

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

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

The WCRT for a given CAN message (i) can be obtained by equation (5) where J_i is jitter of message i. Here, w_i is the *queuing time*, which is the longest transmitting time of the messages first bit without any interference from higher priority message's, where B_i is blocking time, T_j is period of message j, and $hp(i)$ is the set of all messages of higher priority than task i (Equation 6).

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

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

2.3. Holistic Analysis Model

Since WCRTs for tasks depend on the timing attributes of messages arriving at the CPU and WCRTs for messages depend on the timing attributes of the sending task, end-to-end response time constraints should be considered. This analysis method is called holistic analysis. With holistic analysis, developers can find whether a specific

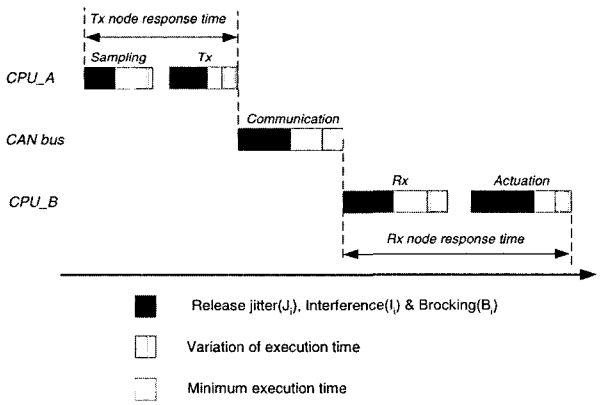

Figure 1. End-to-end response time of a specific control process.

control process can be operated correctly or not at the design phase.

Tindell (1994) suggested the way how to do holistic analysis with the idea of attribute inheritance. Figure 1 shows the simple system model implemented with two CPUs connected via a CAN bus (Lonn and Axelsson, 1999). When two CPUs share a data through CAN network message, time from sampling to message transfer, time for communication in CAN bus, and time from message receiving to actuation should be considered all together to obtain the end-to-end response time of a specific control process. This is the reason not only why holistic analysis is needed but also why holistic analysis is difficult.

In above situation we have (Shin *et al.*, 2002):

$$J_i = R_{send(A)} \tag{7}$$

$$J_{dest(i)} = R_i - 47\ \tau_{bit} \tag{8}$$

$$R_{end\text{-}to\text{-}end} = R_{send(A)} + w_i + C_i + w_{dest(i)} \tag{9}$$

In equation (7), the WCRT($R_{send(A)}$) of the sending node (*Send*(A)) is a preparation time to transmit the message from sampling. And this time can be regarded as message(i)'s jitter(J_i), which is the difference between the earliest and latest time a message could be released. Then, the new WCRT(R_i) of the CAN message(i), which is the time from sampling of CPU_A to communication of CAN bus in Figure 1, can be computed with equation (5) and (7). And the local response time($R_{end\text{-}to\text{-}end}$) from sending node(*send*(A)) to receiving task(*dest*(i)) can be computed with equation (5) and (7) where w_i is the *queuing time* for the message, $w_{dest(i)}$ is the *time window* for the receiving task, and C_i is the maximum time to transmit whole message through a CAN bus (equation 9). This is the response time of receiving task considering its jitter($J_{dest(i)}$) as the message's delay, which is difference between the longest time(R_i) and the shortest time($47\tau_{bit}$) for transmitting a message (Equation 8). In other word it means the time from sampling of CPU_A to Rx of CPU_B in Figure 1. This time equals to the sum of WCRT of sending node, WCRT of message, and WCRT of receiving tasks. The whole end-to-end worst case response time for a specific control process can be computed by simply adding a local response time for actuation of receiving node to equation (9).

3. RAT(RESPONSE-TIME ANALYSIS TOOL)

3.1. Objectives

The main objective of RAT is to provide users with a familiar development environment encapsulating the complex analysis algorithms, so that control engineers can obtain the timing attributes of a distributed real-time control system easily. Now, there is no need for control engineers to know the complex analysis algorithms for timing analysis, but they have to know some timing attributes of tasks and messages defined in RAT. In other words, RAT provides an abstraction layer to the control engineers who developes the control system. By performing schedulability analysis with varying CPU and network characteristics before the implementation phase, a system safety at the critical instance can be guaranteed and a comprehensive specification of the system can be given. Then, development time and cost ultimately can be reduced.

3.2. Resources

A cpu can be organized with the tasks expressed with some timing attributes. In the case where the task having shared resource uses semaphore to avoid concurrency problems, it is expressed as a *Semaphore* attribute in RAT. This timing attribute represents the time that each task possesses a shared resource. For a timing analysis considering time offset, it is necessary to define a *Transaction* attribute which is a set of tasks with the same period and fixed time between the activation of the each task. *Offset* and *Every* attribute are needed to represent the task belonging to the *Transaction*. Figure 2 shows the cpu organization and its timing attributes expressed in RAT.

Similarly, a network can be organized with the messages expressed with some timing attributes. The *Bus speed* attribute enables developers to easily simulate the system at various bus speeds. Currently, CAN network protocol is implemented but it will be extended in the future to other network protocols such as LIN. Figure 3 shows network organization and its timing attributes expressed in RAT.

Table 1 shows all timing attributes used to organize tasks and messages in RAT.

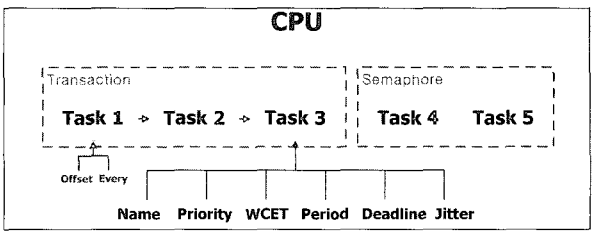

Figure 2. Cpu organization and its timing attributes.

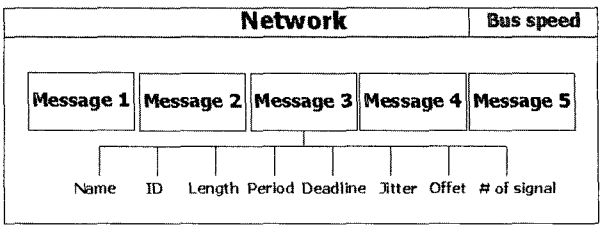

Figure 3. Network organization and its timing attributes.

Table 1. Timing attributes organizing tasks and messages.

Task		Message	
Name	Name of task	Name	Name of message
ID	Identifier of task	ID	Identifier of message
WCET	Worst case excution time	Length	Length of message (0–8 byte)
Period	Period of task	Period	Period of message
Deadline	Deadline of task	Deadline	Deadline of message
Jitter	Jitter of task	Jitter	Jitter of message
Offset	Offset of task in transaction	Offset	Offset of message
Every	Every of task in transaction	# of signal	Number of signal
Semaphore	Time to be possessed by semaphore		

3.3. Timing Analysis

The schedulability is verified for a given hard real-time system under a given scheduling algorithm. Currently, schedulability analysis for the fixed scheduling algorithms is implemented in RAT. A user gives timing attributes as in the above section in order to analyze the system. RAT represents the target system in spreadsheet form so that the attributes' value can be changed easily. Finally, to analyze the system, developers only have to click the *Analysis* button or *System analysis* menu with the mouse because the analysis algorithm has already been implemented in RAT. Consequently, the timing behaviors of the system can be obtained.

As a individual schedulability analysis, a user can know the schedulability of each task or message. If a task or a message is not schedulable which mean the WCRT of a task (Equation 1) or a message (Equation 5) is more than its deadline, RAT notifies the users with a warning message that the system has a problem. In this case, by changing the resources' attributes, any of the displayed results can be updated immediately. This is very useful for obtaining a comprehensive but safe specification of the system between the different resources' attribute in the system.

As a end-to-end (or holistic) schedulability analysis, a user can know the schedulability of a precedence relationship. Precedence relationship represents the specific control process with a series of tasks and messages. A precedence relationship can be defined easily by selecting tasks or messages according to control process in RAT (Table 2). Finally, the holistic analysis results, end-to-end response time (Equation 9) of defined control process for the case of well defined input or some intelligent suggestions for the case of wrong defined input, can be obtained.

Table 2. Precedence relationship input form.

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

3.4. User Interface

RAT is the timing analysis tool based on MATLAB, which is a powerful engineering software. Figure 4 shows the Main window of RAT. It is conceptually divided into two subframes, which are *Menu frame* and *Task (or message) frame*. First, a user can give a set of timing attributes to RAT through the various menus such as File, Reesource, and Constraint in the *Menu frame*. Then all timing attributes are displayed on the *Task (or message) frame*. For input to fulfill a holistic analysis, the *Constraint* menu in the *Menu frame* is used. This menu launches the new window, the *Precedence window*. Developers can create or delete the precedence relationship by using this window.

A user can use not only the *Resource* or *Constraint* menu in the *Menu frame* for input, but also the *System file*, which is a text file expressing the target system with reserved words. A user can load the system information defined in *System file* to RAT directly and save the system under analysis to *System file* as well. Also, for flexible data sharing with CANdb data management program, which is one of the most famous network database management softwares, the function to get network data directly from CANdb file is implemented. Finally, timing

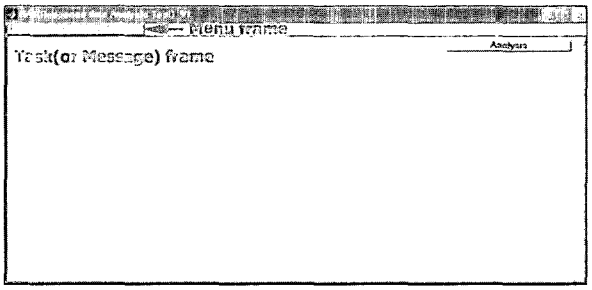

Figure 4. Main window of RAT.

Response time of Tasks

Node	Task name	Deadline	Blocking time	Response time	Schedulability
Properties of the CPU SAMPLE_SIDE					
SAMPLE_SIDE	task1	50	0	20	Schedulable
SAMPLE_SIDE	Sampling	10000	0	40	Schedulable
SAMPLE_SIDE	Te_task	10000	0	40	Schedulable
SAMPLE_SIDE	task2	120	3	67	Schedulable
SAMPLE_SIDE	task7	1150	4	103	Schedulable
SAMPLE_SIDE	task5	1000	4	70	Schedulable
SAMPLE_SIDE	task6	000	4	140	Schedulable
SAMPLE_SIDE	task3	500	4	202	Schedulable
SAMPLE_SIDE	task4	550	0	378	Schedulable

Figure 5. Response time of tasks window.

Response time of CAN messages

Message	Deadline	Response time	Schedulability
Properties of the Bus, CAN1			
MSG4	10	0.964	Schedulable
MSG5	10	1.206	Schedulable
MSG1	160	1.728	Schedulable
MSG3	160	2.15	Schedulable
MSG2	160	2.15	Schedulable
Utilization:	0.0925525		

Figure 6. Response time of CAN messages.

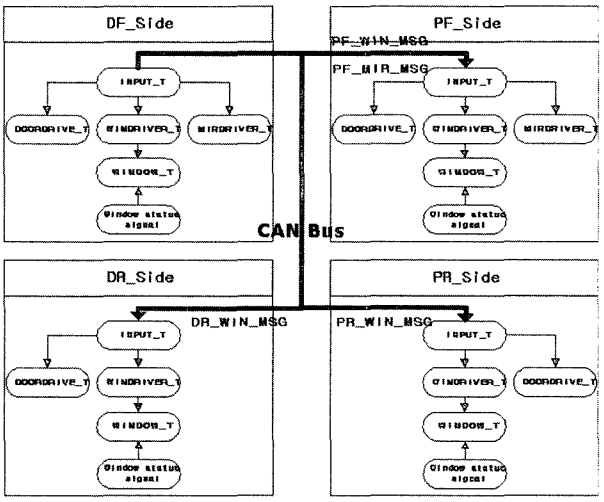

Figure 7. Vehicle body network system.

analyses are simply triggered by clicking the *Analysis* button or *System analysis* menu with the mouse.

The results of analysis are printed on two result windows, which are *Response time of tasks window* for results of tasks (Figure 5) and *Response time of CAN messages window* for results of messages (Figure 6).

The former shows each tasks analysis results such as blocking time, response time (WCRT), and schedulability which is decided by comparing task's WCRT with its deadline. The latter shows each messsage's analysis similar to tasks analysis results. A utilizations of the cpus and networks are displayed on each window as well. And holistic analysis results are represented at the bottom of *Response time of tasks window.*

4. SIMULATION

4.1. Organization of the System

A target system, vehicle body network system (Shin *et al.*, 2002), is organized based on OSEK/VDX standard. An organization of the whole system is depicted in Figure 7. This system has four cpus, Driver Front Side (DF_Side), Pessanger Front Side (PF_Side), Driver Rear Side (DR_Side), and Pessanger Rear Side (PR_Side), which communicate with each other through the CAN bus. It is assumed that DF_Side sends messages to the other cpus and each cpu receives a message from the DF_Side.

It is assumed that all tasks in each cpu are of the same period, 20 ms. And the worst case execution times are like Table 3.

4.2. Simulation Results

Tables 4 through 7 show the simulation results computed by RAT. Table 4 shows the simulation results of each cpu and Table 5 shows the results for a network. Table 4 and 5 presents WCRTs of each task and message without communication. Since tasks in each cpu and messages in a CAN bus satisfy their deadlines, it can be guaranteed that the system behaves correctly, which means all tasks and messages will be completed by their deadlines. But if a task does not meet its deadline, it cannot be guaranteed that the system behaves correctly. Then, the system could be in danger. Therefore, you need to change the system resource attributes until all tasks and messages meet their deadlines.

Table 3. WCET of the tasks in each cpu.

Name of task	DF_Side	PF_Side	DR_Side	PR_Side
INPUT_T	0.56111	0.595615	0.431415	0.431415
DOORDRIVE_T	0.08356	0.083560	0.083560	0.083560
MIRDRIVER_T	0.07108	0.07108	X	X
WINDRIVER_T	0.11424	0.11424	0.11424	0.011424
WINDOW_T	0.04232	0.04232	0.04232	0.042320
COM_T	X	0.00300	0.00300	0.00300

Table 4. WCRT for each cpu (ms).

Name of task	DF_Side	PF_Side	DR_Side	PR_Side
INPUT_T	0.56111	0.595615	0.431415	0.431415
DOORDRIVE_T	0.64467	0.67917	0.51497	0.51497
MIRDRIVER_T	0.15464	0.15464	X	X
WINDRIVER_T	0.19780	0.19780	0.19780	0.19780
WINDOW_T	0.19696	0.19696	0.12588	0.12588
COM_T	X	0.08656	0.08656	0.8656

Table 5. WCRT for a network (ms).

Name of message	T (ms)	WCRT (ms)
CAN_DL_MSG	20	0.595615
PF_MIR_MSG	20	0.67917
PF_WIN_MSG	20	0.15464
DR_WIN_MSG	20	0.19780
PR_WIN_MSG	20	0.19696
PR_WIN_MSG	20	0.08656

Tables 6 and 7 show the simulation results of the holistic analysis. It is vital to know that INPUT_T tasks in each cpu are made to check control input or the message every 20ms period. Therefore, it can be assumed that the delay time for sending a message is the same as the period of sending tasks and the delay time for receiving message is the same as the period of receiving tasks in a worst case scenario.

A delay of the sending task can be regarded as message jitter. Then, a local response time can be computed easily. This is a new WCRT of the message. This is presented in Table 6 as R.

To compute another local response time from the sending task to receiving task, it is necessary to compute the response time of the receiving task considering jitter as message delay, which is the difference between the longest time and the shortest time for transmitting a message. Then, the whole end-to-end worst case response time for a specific control process can be computed by simply adding a local response time for the receiving cpu.

Table 7 shows whole end-to-end worst case response time for some control processes.

Since receiving tasks jitter generated by a message is larger than its period, each receiving task in the receiving cpu (such as PF_SIDE, DR_SIDE, and PR_SIDE) can miss its deadline (Table 7). This can lead to serious problems in the system. Therefore, it is imperative to work out a way to avoid this situation. First, if the problem is being caused by communication, it is necessary to redefine message IDs, repackage signals, or change bus speed. Second, if the problem is being caused by tasks, it is necessary to reduce the execution time by optimizing the program. Changing the organization of the task also can be helpful. Consequently, these analysis processes will help the system developer effectively determine bus speed, organization of tasks, and ID. In this way, development time and cost can be reduced.

Table 6. New WCRT for messages.

Name of message	Time (ms)					
	T	C	J	B	R	W
CAN_DL_MSG	20	0.52	20.5611	0.524	21.6091	0.528
PF_MIR_MSG	20	0.52	20.5611	0.524	22.6571	1.576
PF_WIN_MSG	20	0.52	20.5611	0.524	23.7051	2.624
DR_WIN_MSG	20	0.52	20.5611	0.524	24.7531	3.672
PR_WIN_MSG	20	0.52	20.5611	0	25.2771	4.196

Table 7. End-to-end WCRT for some control processes.

Sender	Receiver		End-to-end WCRT (ms)
DF_SIDE	PF_SIDE	Mirror	23.2778 + 21.3459 = 44.6237
		Window	24.1247 + 22.1816 = 46.3063
DF_SIDE	DR_SIDE	Window	24.8951 + 21.6179 = 46.513
DF_SIDE	PR_SIDE	Window	25.5333 + 21.8 = 47.1513

5. CONCLUSIONS AND FUTURE WORKS

In this paper, the timing analysis tool, RAT has been proposed. RAT can analyze a distributed real-time control system based on one of the most famous analysis algorithm, the WCRT analysis algorithm. And the vehicle body network system has been analyzed to prove the feasibility of RAT.

Through this research, it is confirmed that RAT is useful tool for control engineer, who is not familiar with timing behavior of the real-time system, to develope distributed real-time control systems. A schedulability analysis with RAT lets a developer know whether a system behaves correctly or not in the worst case before the implementation phase. Especially, because RAT make a developer can express any control process as precedence relationship easily, any control engineer can analyze the systems and obtain their timing behaviors. Then, a comprehensive specification of the system can be obtained and the safety of the system in critical instance can be guaranteed as well.

One of our primary objectives of RAT is to provide a control engineer with a familiar development environment that makes it easy to design a complex distributed real-time control system. However, the current implementation of RAT can analyze only static analysis algorithms for hard real-time systems. For an efficient analysis of other scheduling policies, RAT must be extended. And the description of various overheads such as context switching overhead will be facilitated for more precise analysis. It is also necessary to make RAT that can analyze other systems, which have various network protocols and gateways. The gateway is intermediate, which connects different network protocols. Finally, more rich advices and design guides for the system will be updated and improved continuously.

ACKNOWLEDGEMENT–This research is supported by MOST (Ministry of Science and Technology) under the National Research Laboratory program.

REFERENCES

Braberman, V. (1997). On intergrating scheduling theory into formal models for hard real time systems. *Workshop Formal Methods for the Design of Real-Time Systems*, Vila Olmo, Como Italy. 121–141.

Gonzalez Harbour, M., Gutierrez Gacia, J. J., Palencia Gutierrez, J. C. and Drake Moyano, J. M. (2001). MAST: Modeling and analysis suite for real time application. *IEEE 13th Euromicro Conference on Real-Time Systems*, Delft, Netherlands. 125–134.

Hansson, H., Lawson, H., Bridal, O., Eriksson, C., Larsson, S., Lon, H. and Stromberg, M. (1997). BASEMENT: An architecture and methodology for distributed automotive real-time systems. *IEEE Transaction on Computers* **46, 9,** 1016–1027.

Joseph, M. and Pandya, P. (1986). Finding response times in a real-time system. *BCS Computer Journal* **29, 5,** 390–395.

Krishna, C. M. and Shin, K. G. (1997). *Real-time Systems.* McGraw-Hill. New York.

Larsson, J. (1998). SCHEDULITE: A fixed priority scheduling analysis tool. *MSc Thesis, Department of Computer Systems, Uppsala University.*

Lawrenz, W. (1997). *CAN System Engineering From Theory to Practical Applications. Springer-Verlag.* New York.

Lee, W. T. and Sunwoo, M. (2001). Vehicle electric power simulator for optimizing the electric charging system. *Int. J. Automotive Technology* **2,** 4, 157-164.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery* **20, 1,** 46–61.

Lonn, H. and Axelsson, J. (1999). A comparison of fixed-priority and static cyclic scheduling for distributed automotive control applications. *IEEE 11th Euromicro Conference on Real-time Systems*, York, United Kindom.

Motorola., HC12 OSEK Operating System User's Manual Rev. 1.7.

Motorola., OSEK COM/NM User's Manual Rev. 1.0

Sha, L., Rajkumar, R. and Lehoczky, J. (1990). Priority inheritance protocol: An approach to real-time synchronization. *IEEE Transactions on Computers* **39, 9,** 1175–1185.

Shatterjee, S., Bradley, K., Madriz, J., Colquist, J. A. and Strosnider, J. (1997). SEW: A toolset for design and analysis of distributed real-time systems. *IEEE Real-Time Technology and Applications Symposium*, 72–77.

Shin, M. S., Lee, W. T. and Sunwoo, M. (2002). Holistic scheduling analysis of a CAN based body network system. *Transactions of the Korean Society of Automotive Engineers* **10, 5,** 114–120.

Tindell, K. (1994). Using offset information to analyze static priority pre-emptively scheduled task sets. *YCS 182, Department of Computer Science, University of York.*

Tindell, K. (1994). Real time system by fixed priority scheduling. *Ph.D Dissertation, Department of Computer Science, University of York.*

Tindell, K., Burns, A. and Wellings, A. (1992). An extendible apporach for analysing fixed priority hard real-time tasks. *Real-time systems*, **6,** 133–151.

Tindell, K. and Clark, J. (1994). Holistic schedulability analysis for distributed hard real-time systems.

Microprocessors and Microprogramming, 117–134.

Tindell, K., Hansson, H. and Wellings, A. (1994). Analysing real-time communication: controller area network (CAN). *IEEE Real-Time Systems Symposium*, 259–263.

Torngren, M. (1998). Fundamentals of implementing real-time control applications in distributed computer systems. *Real-time Systems*, **14,** 219–250.