Scheduling of Sporadic and Periodic Tasks and Messages with End-to-End Constraints

Hyoung Yuk Kim*, Sang Yong Kim**, Hoon Oh*** and Hong Seong Park****

Div. of Electrical and Computer Eng., Kangwon National University
192-1 Hyoja 2 Dong, Chuncheon, 200-701, Korea
(Tel: +82-33-251-6501; Email: {petrus*, purnae**@control.kangwon.ac.kr}
(Tel: +82-33-250-6346; Email: hoonoh***, hspark****@kangwon.ac.kr)

Abstract: Researches about scheduling of the distributed real-time systems have been proposed. However, they have some weak points, not scheduling both sporadic and periodic tasks and messages or being unable to guaranteeing the end-to-end constraints due to omitting precedence relations between sporadic tasks. So this paper proposes a new scheduling method for distributed real-time systems consisting of sporadic and periodic tasks with precedence relations and sporadic and periodic messages, guaranteeing end-to-end constraints. The proposed method is based on a binary search-based period assignment algorithm, an end-to-end laxity-based priority assignment algorithm, and three kinds of schedulability analysis, node, network, and end-to-end schedulability analysis. In addition, this paper describes the application model of sporadic tasks with precedence constraints in a distributed real-time system, shows that existing scheduling methods such as Rate Monotonic (RM) scheduling are not proper to be applied to the system having sporadic tasks with precedence constraints, and proposes an end-to-end laxity-based priority assignment algorithm.

Keywords: Distributed Real-Time Systems, End-to-End Scheduling, Sporadic Task, Laxity

1. INTRODUCTION

A distributed real-time system has various types of tasks, sporadic tasks invoked by an event such as an alarm, periodic tasks sampling a sensor or computing a control algorithm, and non-real time tasks downloading a configuration data. In addition, sporadic, periodic, and non-real time messages are transmitted between the nodes via a fieldbus such as CAN, Profibus, and FIP. So it is necessary to consider both tasks and messages at the same time to satisfy the requirements of the distributed real-time system.

The distributed real-time system with these various kinds of tasks and messages has several end-to-end constraints. They are an end-to-end time one from the sampling time of a sensor to the output time of an actuator, a precedence one that a consumer task has to wait until its producer task finishes computing and sends data to it, and an input data synchronization one that if a control loop has more than two sensors, the difference between points of sampling times must be limited within a time interval. System scheduling must be done to meet these all constraints. Generally, scheduling of distributed real-time systems has been known as a NP-Hard problem [1], and researches have been studied to get sub-optimal schedule results by heuristic methods [2-6].

There are several researches about scheduling of distributed real-time systems. An end-to-end schedulability analysis method was proposed on the passive assumption that no task can receive more than one message [2], and a scheduling method allocating the periods of tasks and messages by pruning methods was presented in [3]. Other scheduling method using optimization methods such as a clustering algorithm and a genetic algorithm [4] and another one assigning the periods and the priorities of tasks and messages by a binary search algorithm [5] were proposed. But these researches did not consider sporadic tasks operating in real systems.

Researches were also studied considering periodic tasks and sporadic tasks simultaneously. A scheduling method handling periodic tasks and sporadic tasks simultaneously was presented in [6] on the assumption that network delays were fixed. But because sporadic tasks only operating independently in local nodes were considered, this method could not guarantee the precedence constraints between sporadic tasks. Sporadic tasks might be invoked and finish their job within local nodes without cooperating remote tasks. But, in case of sporadic tasks invoked by system-critical events being able to crash the system, they have to work together other tasks distributed in the system in order to figure out the event. Because a sporadic task first receiving this kind of event transmits a signal to other sporadic tasks, and then they are invoked, there are precedence relations between these sporadic tasks. Therefore, the precedence constraints of sporadic tasks have to be taken into account.

This paper describes the application model of sporadic tasks with precedence constraints in a distributed real-time system. It is shown that existing scheduling methods such as Rate Monotonic scheduling are not proper to be applied to the system having sporadic tasks with precedence constraints. A new scheduling method is proposed for distributed real-time systems consisting of sporadic and periodic tasks with precedence relations and sporadic and periodic messages, guaranteeing end-to-end constraints. The proposed method is based on a binary search-based period assignment algorithm presented in [5] and an end-to-end laxity-based priority assignment one proposed in this paper. It is shown that the proposed algorithm is able to schedule the system in spite of very high utilization from being applied to an example system.

In section 2, an application model of sporadic tasks with precedence relations is described, and it is shown that the precedence constraints of sporadic tasks have an effect on a system's schedulability. In section 3, a new scheduling method is proposed and applied to an example system, and the schedule result is presented and analyzed. Some conclusions are given in section 4.

2. PRECEDENCE OF SPORADIC TASKS

2.1 Sporadic tasks with precedence

In general, a distributed real-time system has tasks being sporadically invoked by events as well as ones being periodically invoked and forming control loops. In case of sporadic tasks, they may be classified into two types. One is a sporadic task that starts and finishes job within its local node without cooperating remote tasks. The other is a sporadic task that notifies an event's occurrence of or cooperates with tasks in local or remote nodes after being invoked. The former has no precedence constraint but the latter has precedence constraint.

As an example of sporadic tasks with precedence constraints, if the sampling data of a sensor should go over the valid range within that the system operates reliably, an alarm task would be invoked in the sensor node and then send alarm signals or messages to control nodes. Alarm-processing tasks of control nodes are invoked by these signals, compute control values to settle this alarm, and send control values to actuator nodes. Finally, outputting control values to actuators completes the alarm handling. Also, smart sensors or actuators installed in distributed real-time systems have some safety-related tasks or functions such as watchdog timer. If a smart device should go down or fault should take place in the device, these tasks would detect and recover the error or inform error occurrence to other nodes in order to prevent the breakdown of the overall system.

As mentioned above, in case of critical events having to be processed in the overall system, a sporadic task invoked by that kind of event starts other tasks sequentially, and these sporadic tasks form a path. We call this an event path in this paper. All sporadic tasks included in this event path have to finish their job within the deadline of event given as system requirements. That is, the end-to-end response time from the starting point of a task first invoked by an event to the finishing point of a task last invoked must be shorter or equal to the event's deadline.

2.2 Precedence effect of Schedulability

A simple distributed system is represented in Fig. 1 and has sporadic tasks with no precedence. Its notations are summarized in the appendix. All nodes are based on a fixed priority and preemptive scheduler. All tasks are assumed to be invoked at the time of t0 simultaneously and be running without precedence constraints with one another except for the periodic tasks included in the control loop. In order to make analysis simple, we do not care about the transmission delay of message via network at this section.

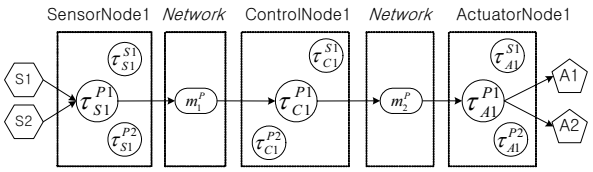

Fig. 1 A system with no precedence between sporadic tasks

Table 1 System requirements and schedule results

Task	τ_{S1}^{P1}	τ_{S1}^{P2}	τ_{S1}^{S1}	τ_{C1}^{P1}	τ_{C1}^{P2}	τ_{C1}^{S1}	τ_{A1}^{P1}	τ_{A1}^{P2}	τ_{A1}^{S1}
e	2	3	2	2	2	4	2	3	2
d	10	10	15	10	10	15	10	10	15
T	10	10	15	10	10	15	10	10	15
p	1	2	3	1	2	3	1	2	3

The requirements of this system are shown in the light part of Table 1, and the end-to-end deadline of control loop is 10ms. A schedule can be found as the shaded part of Table 1, guaranteeing the given requirements. In case of sporadic tasks, their pseudo-periods are assigned by the minimum inter-arrival time, that is, deadline. The priorities of tasks are allocated by RM (Rate Monotonic) algorithm [7].

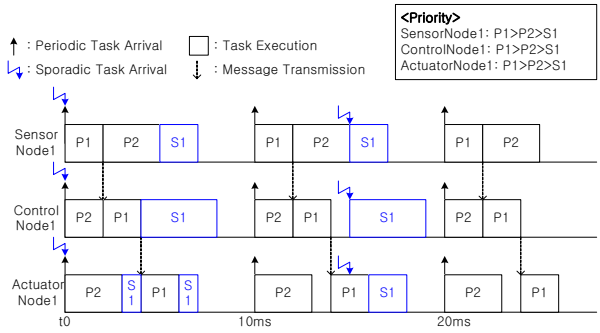

Fig. 2 Timing chart of system with no precedence

Timing chart of the system operating based on this schedule is depicted in Fig. 2 and shows that the system is schedulable because the requirements are guaranteed. At this time, the utilization of each node is as follows: sensor node-63.3%, control node-66.7%, and actuator node-63.3%. This result means that RM can schedule the system under those utilizations. However, if sporadic tasks should have precedence, RM would become unable to find a schedule meeting the requirements.

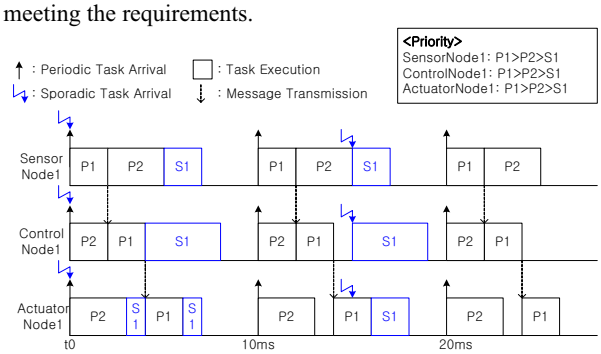

Fig. 3 Timing chart of system with no precedence

Fig. 3 shows a system that the precedence relations between sporadic tasks are added to the system of Fig. 1. In this system, if the deadline of an event invoking the sporadic task τ_{S1}^{S1} should be given as 15ms, the schedule results by RM would be the same that Table 1 because the pseudo-periods of sporadic tasks would become 15ms. The timing chart is changed to Fig. 4 when precedence constraints exist between sporadic tasks. It shows the system scheduled by RM is not schedulable because the end-to-end response time of event path is 18ms. Although the system of Fig. 3 has the same utilizations as that of Fig. 1, RM cannot schedule the system of Fig. 3.

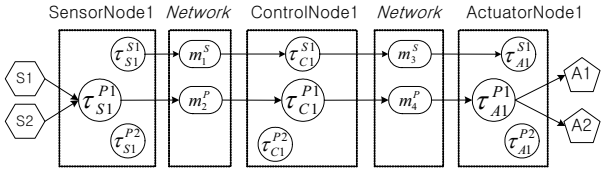

Fig. 4 A system with precedence between sporadic tasks

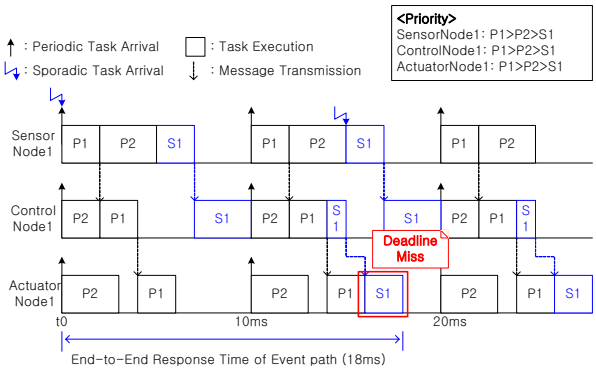

Fig. 5 Timing chart of system with precedence

Without decreasing the system utilizations, the system of Fig. 3 can be scheduled by changing some priorities. Let us compare the laxity of sporadic task included in the event path with that of periodic task not included in the control loop. Although the period of periodic task is shorter than that of the sporadic task, the laxity of the former is bigger than that of the latter, because all sporadic tasks have to finish their job within the end-to-end deadline of event path. Therefore, if the priority of periodic task not included in the control loop should be changed with that of sporadic task in each node, the system would be schedulable. This schedule result is shown in Fig. 5 and shows that scheduling methods such as RM not considering the precedence between sporadic tasks could not find a schedule meeting the end-to-end requirements of system with the precedence, although the schedule exists.

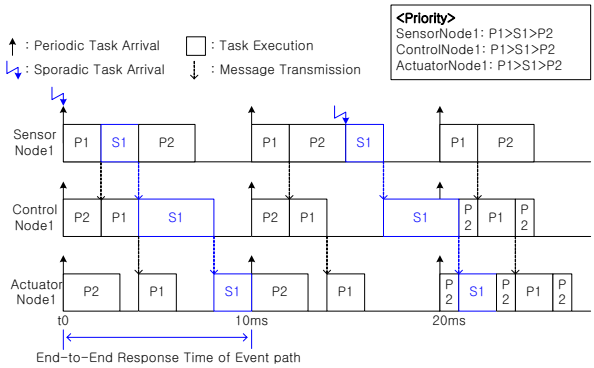

Fig. 6 Timing chart after changing priorities

In this section, the schedulability analysis and the timing chart were presented without considering the transmission delay of message. Should we deliberate the network delay, the end-to-end response times of the control loop and the event path would become longer, and the system would not be schedulable. Therefore, the messages have to be taken into account as well as tasks. That is, the scheduling and schedulability analysis of distributed real-time systems should consider tasks, messages, and end-to-end constraints of the system simultaneously.

3. END-TO-END SCHEDULING METHOD

3.1 System model

The overall flow of a new scheduling method considering end-to-end constraints is presented in Fig. 7. The first step is system modeling that designs the task graph of system and sets parameters of tasks and messages by given requirements. The task graph of a target system is depicted in Fig. 8 to which the new scheduling method will be applied. The system consists of two control loops, two event paths, and other tasks that are not included in the loop and the path. All nodes are supposed to be based on a fixed priority and preemptive scheduler, connected via CAN, and synchronized with the global time. The system requirements are given as Table 2.

Algorithm. End-to-End Scheduling Method
1) System Modeling
   A. Design the task graph of system and allocate tasks to nodes
   B. Set parameters of tasks and messages by system requirements
2) Derive End-to-End Constraints
   A. Derive End-to-End Constraints of control loops
   B. Derive End-to-End Constraints of event paths
3) Period Assignment
   A. Task's period: a binary search-based assignment algorithm
   B. Message's period: inherit from its producer task
4) Priority Assignment
   A. Task's priority: a laxity-based priority assignment algorithm
   B. Message's priority: a laxity-based priority assignment algorithm
5) Schedulability Analysis
   A. Node schedulability: if unschedulable then go to step 3
   B. Network schedulability: if unschedulable then go to step 3
   C. End-to-end schedulability: if unschedulable then go to step 3
6) Decide to Reschedule or Finish
   A. If the end-to-end period of all control loops can be rescheduled to be shorter, then go to step 3
   B. Set the scheduled results to the parameters of tasks and messages

Fig. 7 Overall flow of end-to-end scheduling

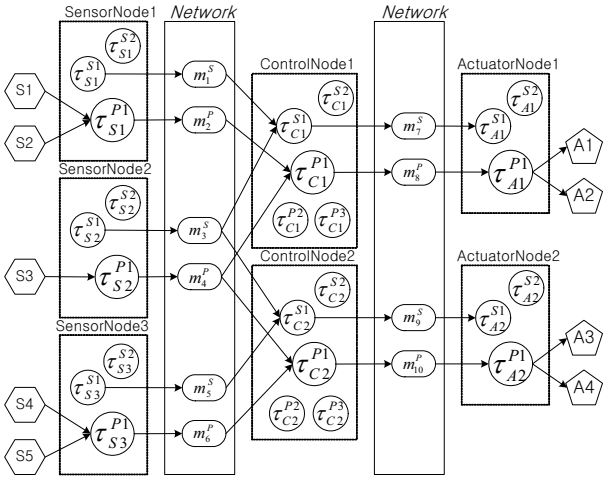

Fig. 8 Task graph of target system

Table 2 System requirements

$MADT_1^{Evnt.} = 15ms$, $MADT_2^{Evnt.} = 15ms$

$MADT_1^{Ctrl.} = 50ms$, $MADT_2^{Ctrl.} = 70ms$

Task	T	e	d	Task	T	e	d	Task	T	e	d	Msg.	Len
τ_{S1}^{S1}		2	15	τ_{C1}^{S2}		4	20	τ_{A1}^{S2}		4	20	m_1^S	8
τ_{S1}^{S2}		4	20	τ_{C1}^{P1}		7		τ_{A1}^{P1}		2		m_2^P	4
τ_{S1}^{P1}		2		τ_{C1}^{P2}	50	5		τ_{A2}^{S1}		2	15	m_3^S	8
τ_{S2}^{S1}		2	15	τ_{C1}^{P3}	100	5		τ_{A2}^{S2}		4	20	m_4^P	2
τ_{S2}^{S2}		4	20	τ_{C2}^{S1}		3	15	τ_{A2}^{P1}		2		m_5^S	8
τ_{S2}^{P1}		2		τ_{C2}^{S2}		4	20					m_6^P	4
τ_{S3}^{S1}		2	15	τ_{C2}^{P1}		10						m_7^S	8
τ_{S3}^{S2}		4	20	τ_{C2}^{P2}	50	5						m_8^P	4
τ_{S3}^{P1}		2		τ_{C2}^{P3}	100	5						m_9^S	8
τ_{C1}^{S1}		3	15	τ_{A1}^{S1}		2	15					m_{10}^P	4

The second step is to derive the precedence constraints between tasks and messages, the end-to-end time constraints, and the input data synchronization constraints of control loops and event paths. The precedence constraints are as follows:

$$\phi_{\tau_{C1}^{S1}} \geq \max(\phi_{\tau_{S1}^{S1}} + d_{\tau_{S1}^{S1}} + d_{m_1^S}, \phi_{\tau_{S2}^{S1}} + d_{\tau_{S2}^{S1}} + d_{m_3^S}),$$
$$\phi_{\tau_{A1}^{S1}} \geq \phi_{\tau_{C1}^{S1}} + d_{\tau_{C1}^{S1}} + d_{m_7^S},$$
$$\phi_{\tau_{C2}^{S1}} \geq \max(\phi_{\tau_{S2}^{S1}} + d_{\tau_{S2}^{S1}} + d_{m_3^S}, \phi_{\tau_{S3}^{S1}} + d_{\tau_{S3}^{S1}} + d_{m_5^S}),$$
$$\phi_{\tau_{A2}^{S1}} \geq \phi_{\tau_{C2}^{S1}} + d_{\tau_{C2}^{S1}} + d_{m_9^S},$$
$$\phi_{\tau_{C1}^{P1}} \geq \max(\phi_{\tau_{S1}^{P1}} + d_{\tau_{S1}^{P1}} + d_{m_2^P}, \phi_{\tau_{S2}^{P1}} + d_{\tau_{S2}^{P1}} + d_{m_4^P}),$$
$$\phi_{\tau_{A1}^{P1}} \geq \phi_{\tau_{C1}^{P1}} + d_{\tau_{C1}^{P1}} + d_{m_8^P},$$
$$\phi_{\tau_{C2}^{P1}} \geq \max(\phi_{\tau_{S2}^{P1}} + d_{\tau_{S2}^{P1}} + d_{m_4^P}, \phi_{\tau_{S3}^{P1}} + d_{\tau_{S3}^{P1}} + d_{m_6^P}),$$
$$\phi_{\tau_{A2}^{P1}} \geq \phi_{\tau_{C2}^{P1}} + d_{\tau_{C2}^{P1}} + d_{m_{10}^P}.$$

The end-to-end time constraints are as follows:

$$\phi_{\tau_{A1}^{S1}} + d_{\tau_{A1}^{S1}} - \min(\phi_{\tau_{S1}^{S1}}, \phi_{\tau_{S2}^{S1}}) \leq MADT_1^{Evnt.},$$
$$\phi_{\tau_{A2}^{S1}} + d_{\tau_{A2}^{S1}} - \min(\phi_{\tau_{S2}^{S1}}, \phi_{\tau_{S3}^{S1}}) \leq MADT_2^{Evnt.},$$
$$\phi_{\tau_{A1}^{P1}} + d_{\tau_{A1}^{P1}} - \min(\phi_{\tau_{S1}^{P1}}, \phi_{\tau_{S2}^{P1}}) \leq MADT_1^{Ctrl.},$$
$$\phi_{\tau_{A2}^{P1}} + d_{\tau_{A2}^{P1}} - \min(\phi_{\tau_{S2}^{P1}}, \phi_{\tau_{S3}^{P1}}) \leq MADT_2^{Ctrl.}.$$

The input data synchronization constraints are as follows:

$$T_{\tau_{S1}^{P1}} = T_{m_2^P} \mid T_{\tau_{C1}^{P1}} = T_{m_8^P} = T_{\tau_{A1}^{P1}}, \quad T_{\tau_{S2}^{P1}} = T_{m_4^P} \mid T_{\tau_{C1}^{P1}} = T_{m_8^P} = T_{\tau_{A1}^{P1}},$$
$$T_{\tau_{S2}^{P1}} = T_{m_4^P} \mid T_{\tau_{C2}^{P1}} = T_{m_{10}^P} = T_{\tau_{A2}^{P1}}, \quad T_{\tau_{S3}^{P1}} = T_{m_6^P} \mid T_{\tau_{C2}^{P1}} = T_{m_{10}^P} = T_{\tau_{A2}^{P1}}.$$

where 'A|B' means that B is multiple of A.

3.2 Period assignment

The third step is to allocate the period of tasks and messages. In case of sporadic tasks, because they are not invoked periodically, their periods do not exist. But, because the system schedulability is analyzed under the maximum load condition applied to the system, the maximum inter-arrival times of sporadic tasks are generally used to assign pseudo-periods to sporadic tasks. If the deadline of a sporadic task is 10ms, its pseudo-period becomes 10ms because it cannot be invoked once more within its deadline. It is straightforward that this method is applied to sporadic tasks operating independently without precedence constraint. In case of sporadic tasks included in an event path with precedence constraints, the end-to-end deadline of the event is given but each task's deadline is not. But, because those sporadic tasks are invoked sequentially after the event occurs, their maximum inter-arrival times become equal to the event's maximum inter-arrival time. Therefore, the end-to-end deadline of the event is set as the periods of those sporadic tasks.

In case of periodic tasks not being included in control loops and operating independently, their periods are set to the their deadlines given as requirements. In case of periodic tasks consisting of control loops and having precedence constraints, their periods have to be assigned guaranteeing input data synchronization constraints. Therefore, it is not simple to assign their periods. This paper uses a binary-search based period assignment method proposed in [6]. That method can be applied to the system with multiple control loops and nodes and some periodic tasks like sampling tasks being used by more than one control loop.

In case of message, because it is transmitted every its producer task's invocation, its period is inherited from its producer task's period.

3.3 Priority assignment

The fourth step of end-to-end scheduling is to assign the priority of tasks and messages. Because this paper assumes that the system is preemptive, fixed priority system, the priorities of tasks and messages are assigned by off-line scheduling method. There are two famous scheduling algorithms used by off-line method. One is RM algorithm under the condition that the period and the deadline of task are equal. The other is deadline monotonic (DM) one [1]. In case of DM, the period and deadline of task are not equal.

Because RM only consider the period of each task and does not take into account the laxity, RM might be unable to find out a schedule of system with precedence constraints as mentioned in section 2. That is, RM cannot assign higher priority to task with less laxity.

DM might solve the problem of RM because it allocates task's priority based on its deadline. But, in order to schedule tasks, the deadlines of all tasks have to be assigned. In case of tasks operating independently without precedence constraints, their deadlines are given as the system requirements. But, the deadlines of tasks included in control loops or event paths are not give but the end-to-end deadline of control loops and event paths are as the system requirements. Therefore, additional deadline assignment algorithm is necessary to use DM and is not simple.

This paper proposes an end-to-end laxity-based priority assignment algorithm presented in Fig. 9. This algorithm is different from LLF (Least Laxity First) of on-line scheduling methods because the laxity is obtained from the end-to-end of system. The proposed method calculates the laxities of all tasks and allocates higher priority to task with less laxity. In case of task not included control loops or event paths, its laxity is set to that its period minus its execution time is. The laxities of other tasks are set as follows: calculate the sum of all execution times of task and all transmission time of messages included in a control loop or an event path. Subtract the sum of execution time and transmission time from the end-to-end deadline of the control loop or the event path. Divide the result by the number of tasks and messages in the control loop or the event path. Final result is assigned as the laxity of tasks and messages.

/* Φ : the task set including all tasks in the system */
/* E_j : the set including all tasks and messages in the *j*-th event path */
/* X_j : the set including all tasks and message in the *j*-th control loop */
/* Ψ_k : the task set including all tasks in the *k*-th node */
/* lax_i : the laxity of τ_i */
/* $N(x)$: the number of all tasks and messages included in a set *x* */
/* $EventPathID(x)$: the event path's index including a task *x* */
/* $ControlLoopID(x)$: the control loop's index including a task *x* */

foreach $\forall \tau_i \in \Phi$ do
  if τ_i is included in a control loop or a event path
  then if τ_i is a sporadic task then $j := EventPathID(\tau_i)$

$$lax_i := \left(MADT_j^{Evnt.} - \sum_{\forall \tau_i \in \mathrm{E}_j} e_i - \sum_{\forall m_i \in \mathrm{E}_j} C_i \right) \Big/ N(\mathrm{E}_j)$$

  else $j := ControlLoopID(\tau_i)$

$$lax_i := \left(MADT_j^{Ctrl.} - \sum_{\forall \tau_i \in \mathrm{X}_j} e_i - \sum_{\forall m_i \in \mathrm{X}_j} C_i \right) \Big/ N(\mathrm{X}_j)$$

  else $lax_i := T_i - e_i$
end

foreach $\forall k$ do
  Set the priority of $\forall \tau_i \in \Psi_k$ according to lax_i
end

Fig. 9 Laxity-based priority assignment algorithm

3.4 Schedulability Analysis

After assigning period and priority of all task and message, the system schedulability is tested through node schedulability analysis, network schedulability analysis, and end-to-end schedulability. Node and network schedulability analysis check whether the worst-case response time of task and message is equal or shorter than the deadline of them or not. End-to-end schedulability is analyzed by inspecting that the derived precedence and the end-to-end time constraints are guaranteed. If one of these analyses should be failed, the schedule process would return to step 3, and try to find another period set. When these analyses are passed, if the end-to-end period of control loops could be shorter, step 3 is repeated to find a shorter period set. Finally, the last founded results are used as the parameters of tasks and messages.

3.5 Schedule results

The proposed end-to-end scheduling method is applied to the target system depicted in Fig. 8, and the schedule result is shown in Fig. 10~13. Fig. 10~11 show the end-to-end response times of control loops and event paths and the utilization of each node and network when the speed of CAN is 500kbps. X-axis represents the iteration of the proposed scheduling process to find out the shortest periods meeting the schedulability. By four times of scheduling iteration, the end-to-end periods of control loop 1 and 2 are scheduled to 25ms and 30ms. The end-to-end response times are 22ms and 27ms. The end-to-end response time of event paths are 10ms and 11ms, meeting the end-to-end deadline.

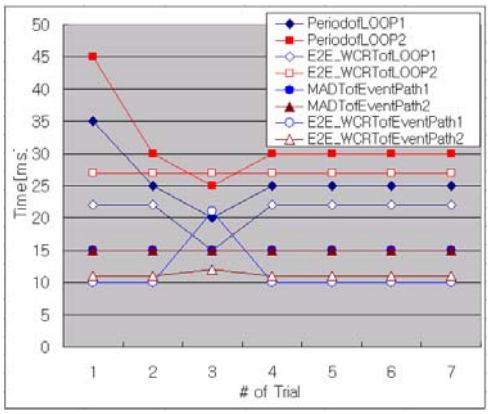

Fig. 10 End-to-end periods and response times (CAN: 500kbps)

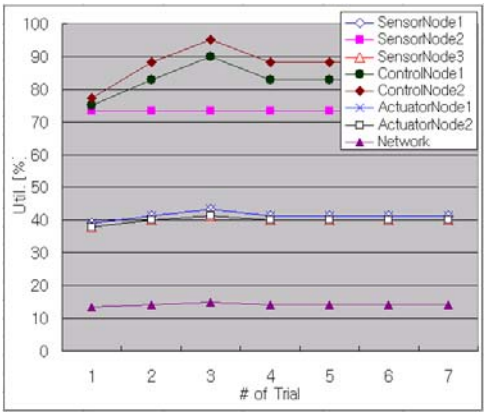

Fig. 11 Utilization of nodes and network (CAN: 500kbps)

At the final schedule, the utilization of SensorNode1, ControlNode1, and ControlNode2 is 73%, 83%, and 88% respectively and is very high. This result is caused by the binary search-based period assignment method that tries to allocate the shortest period to each task to make the end-to-end period of control loop short. In case of RM, its utilization limit guaranteeing schedulability is $m(2^{1/m}-1)$, where m is the number of task. Because the utilization limit of RM for the target system is 78% and 72%, it shows that the proposed method can schedule the system with higher utilization than that of RM in spite of considering precedence constraint.

Fig. 12 ~ 13 show the schedule result when the speed of CAN is 250kbps. Like 500kbps, the schedule was founded within four time iterations. The end-to-end periods of control loop 1 and 2 are scheduled to 30ms and 35ms, and the end-to-end response times are 26ms and 31ms. The end-to-end response time of event paths are 12ms and 13ms. The utilization of SensorNode1, ControlNode1, and ControlNode2 is 73%, 78%, and 84% respectively. The reason that the node utilization of 250kbps is less than that of 500kbps is that the end-to-end period of control loops become longer because the end-to-end response times become longer due to the low network speed.

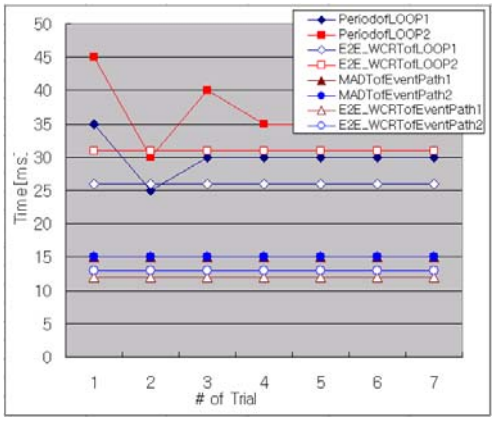

Fig. 12 End-to-end periods and response times (CAN: 250kbps)

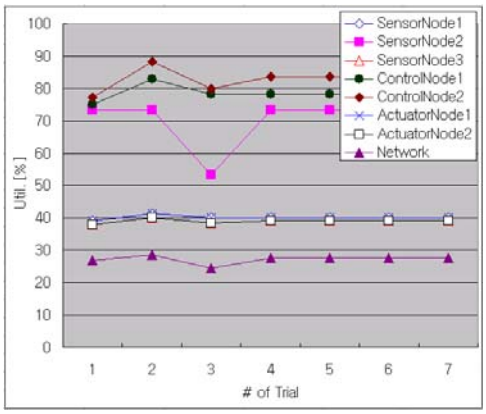

Fig. 13 Utilization of nodes and network (CAN: 250kbps)

If the speed of CAN become 125kbps, the network utilization goes up to 47% and it is impossible to find a schedule guaranteeing the end-to-end deadline of event path because the end-to-end laxity is very tight. Therefore, the network speed is one of important factors for scheduling the system. In order to schedule the system, it is necessary to change the system requirements of event paths or the network speed.

Schedule results such as priority, period, etc. of all tasks are shown in Table 3, and those of messages in Table 4. The white parts of table are parameters as previously given system

requirements and the shaded parts are ones scheduled by the proposed end-to-end scheduling method.

Table 3 Schedule result of task (CAN: 250kbps)

Task	Node	p	T	e	R	d	ϕ	Task	Node	p	T	e	R	d	ϕ
τ_{S1}^{P1}	1	1	15	2	2	2	0	τ_{C2}^{P1}	5	1	15	3	3	3	5
τ_{S1}^{P2}		3	20	4	8	20	0	τ_{C2}^{P2}		3	20	4	20	20	0
τ_{S1}^{P3}		2	30	2	4	4	0	τ_{C2}^{P3}		2	35	10	13	13	9
τ_{S2}^{P1}	2	1	15	2	2	2	0	τ_{C2}^{P4}		4	50	5	29	50	0
τ_{S2}^{P2}		3	20	4	12	20	0	τ_{C2}^{P5}		5	100	5	59	100	0
τ_{S2}^{P3}		2	5	2	4	4	0	τ_{A1}^{S1}	6	1	15	2	2	2	10
τ_{S3}^{P1}	3	1	15	2	2	2	0	τ_{A1}^{S2}		3	20	4	8	20	0
τ_{S3}^{P2}		3	20	4	8	20	0	τ_{A1}^{P1}		2	30	2	4	4	22
τ_{S3}^{P3}		2	35	2	4	4	0	τ_{A2}^{S1}	7	1	15	2	2	2	11
τ_{C1}^{P1}	4	1	15	3	3	3	4	τ_{A2}^{S2}		3	20	4	8	20	0
τ_{C1}^{P2}		3	20	4	14	20	0	τ_{A2}^{P1}		2	35	2	4	4	27
τ_{C1}^{P1}		2	30	7	10	10	8								
τ_{C1}^{P2}		4	50	5	26	50	0								
τ_{C1}^{P3}		5	100	5	48	100	0								

Table 4 Schedule result of message (CAN: 250kbps)

Msg.	p	T	L	C	R	d	ϕ
m_1^S	2	15	8	0.52	1.56	2	2
m_2^P	7	25	4	0.37	3.63	4	4
m_3^S	1	15	8	0.52	1.04	2	2
m_4^P	6	5	2	0.29	3.26	4	4
m_5^S	3	15	8	0.52	2.08	3	2
m_6^P	9	30	4	0.37	4.36	5	4
m_7^S	4	15	8	0.52	2.6	3	7
m_8^P	8	25	4	0.37	4	4	18
m_9^S	5	15	8	0.52	2.97	3	8
m_{10}^P	10	30	4	0.37	4.36	5	22

4. CONCLUSIONS

This paper has described the application model of sporadic tasks with precedence constraints in a distributed real-time system. It has been shown that existing scheduling methods such as Rate Monotonic scheduling are not proper to be applied to the system having sporadic tasks with precedence constraints. End to-end scheduling method and the end-to-end laxity-based priority assignment algorithm have been proposed for distributed real-time systems consisting of sporadic and periodic tasks with precedence relations and sporadic and periodic messages, guaranteeing end-to-end constraints. Through an example system, it is shown that the proposed algorithm is able to schedule in spite of the system having very high utilizations.

Future works are to apply the proposed method to more complex systems and cooperate with on-line scheduling methods.

REFERENCES

[1] J. Y.-T Leung and J. Whitehead, "On Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks," *Performance Evaluation*, 2(4), pp. 237-250, December, 1982.

[2] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems", *Microprocessing and Microprogramming*, Vol.40, No.2, pp.117-134, Apr., 1994.

[3] J.W. Park, Y.S. Kim, S.S. Hong, M. Saksena, S.H. Noh and W.H. Kwon, "Network conscious of Distributed Real-Time Systems," *Journal of System Architecture*, pp. 131-156, 1998.

[4] S. Faucou, A.-M. Deplanche and J.-P. Beauvais, "Heuristic Techniques for Allocating and Scheduling Communicating," *Proc. of WFCS2000*, pp. 257-265, 2000.

[5] Hyoung Yuk Kim and Hong Seong Park, "Period and Priority Assignment Method for DCS Design," *Asian Journal of Control*, Vol.5, No.3, pp.422-432, Sept., 2003.

[6] D. Isovic and G. Fohler, "Handling sporadic tasks in off-line scheduled distributed real-time systems," *Proc. of ECRTS'99*, pp.60-67, June, 1999.

[7] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, Vol.20, No.1, pp.46-61, 1973.

[8] Hong Seong Park, Yong Ho Kim, Dong Sung Kim, Wook Hyun Kwon, "A Scheduling Method for Network-based Control Systems", *IEEE Transaction on Control System Technology*, Vol.3. No.3, pp318-330, May, 2002.

[9] A. Burns, "Preemptive priority based scheduling: An appropriate engineering approach in Principles of Real-Time Systems," *Prentice Hall*, 1994.

[10] K. Tindell, H. Hansson, and A. Wellings, "Analyzing Real-time Communications: Controller Area Network," *Proc. of RTSS'94*, pp.259-263, 1994.

APPENDIX. NOTATIONS

The following notations will be used throughout this paper.

τ_β^α The task α in the node β, $\alpha \in \{Si, Pi\}$ where *Si* and *Pi* denote the *i*-th sporadic task and periodic task respectively, and $\beta \in \{Sj, Cj, Aj\}$ where *Sj*, *Cj*, and *Aj* denote the *j*-th sensor node, control node, and actuator node respectively.

m_k^γ The *k*-th message that is γ-type, $\gamma \in \{S, P\}$ where *S* and *P* denote a sporadic and a periodic message respectively.

e_δ The execution time of a task δ, $\delta = \tau_\beta^\alpha$.

C_δ The execution time of a message δ, $\delta = m_k^\gamma$.

T_δ The period of δ, $\delta \in \{\tau_\beta^\alpha, m_k^\gamma\}$.

d_δ The deadline of δ, $\delta \in \{\tau_\beta^\alpha, m_k^\gamma\}$.

p_δ The priority of δ, $\delta \in \{\tau_\beta^\alpha, m_k^\gamma\}$.

ϕ_δ The initial phase time of δ, $\delta \in \{\tau_\beta^\alpha, m_k^\gamma\}$.

$MADT_i^{Ctrl.}$ The end-to-end maximum allowable delay time of the *i*-th control loop.

$MADT_i^{Evnt.}$ The end-to-end maximum allowable delay time of the *i*-th event path.