International Journal of Automotive Technology, Vol. 8, No. 1, pp. 39–48 (2007)

1229-9138/2007/032-06

OPTIMAL PERIOD AND PRIORITY ASSIGNMENT FOR A NETWORKED CONTROL SYSTEM SCHEDULED BY A FIXED PRIORITY SCHEDULING SYSTEM

M. SHIN[1] and M. SUNWOO[2]*

[1]ACE Lab, Hanyang University, Seoul 133-791, Korea
[2]Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea

(Received 24 May 2006; Revised 16 August 2006)

ABSTRACT–This paper addresses the problem of period and priority assignment in networked control systems (NCSs) using a fixed priority scheduler. The problem of assigning periods and priorities to tasks and messages is formulated as an optimization problem to allow for a systematic approach. The temporal characteristics of an NCS should be considered by defining an appropriate performance index (PI) which represents the temporal behavior of the NCS. In this study, the sum of the end-to-end response times required to process all I/Os with precedence relationships is defined as a PI. Constraints are derived from the task and message deadline requirements to guarantee schedulability. Genetic algorithms are used to solve this constrained optimization problem because the optimization formulation is discrete and nonlinear. By considering the effects of communication, an optimum set of periods and priorities can be holistically derived.

KEY WORDS : Networked control system (NCS), Genetic algorithm (GA), Optimization, Performance measure, Fixed priority scheduling (FPS)

1. INTRODUCTION

Networked control systems (NCSs) are used frequently to reduce cost, improve quality, and shorten the time to market of various applications, including vehicles, aircraft, spacecraft, and robots. Because of the architecture of NCSs in which controllers, sensors, and actuators are connected by a communication bus, the systems should be designed to specifically consider the requirements of real-time and distributed systems. The performance of an NCS depends not only on the functional behaviors, but also on the temporal behaviors. The temporal behavior of the NCS must be confirmed during the design phase by performing a timing analysis such as a schedulability test of tasks and messages.

There are two different approaches to implementing real-time systems. The method employed depends on the triggering mechanism used to initiate action in each node. One is a time-triggered (TT) approach, and the other is an event-triggered (ET) or priority-based approach (Kopetz, 1997). TT systems are scheduled by a static cyclic scheduler, and all activities are initiated at predetermined points in time. Activities of ET systems are initiated by events other than points in time, and priority based scheduling is generally used when an event simultaneously activates two or more tasks. A previous study proposed an optimal period selection method for a TT system (Shin *et al.*, 2005).

The temporal behavior of NCSs scheduled by a fixed priority scheduling (FPS) policy is mainly determined by the assignment of priorities to tasks and messages. Therefore, periods and priorities of tasks and messages are major design parameters and are among the most important issues in NCS development. This paper addresses a design method to assign periods and priorities to NCS tasks and messages.

Researchers have considered several methods for assigning optimal priorities to tasks and messages during the design phase by considering system performance and real-time characteristics. In theory, the simplest way to obtain a set of optimal priorities is to perform a full-search of all possible task scenarios. However, a full-search of all possible task scenarios is not feasible (Kai and Shimada, 1999). If the relative deadlines of all tasks are less than or equal to the periods in a uniprocessor, then the optimal priority assignment schemes regarding the worst-case load can be determined by rate monotonic scheduling and deadline.

Monotonic scheduling methods. The schemes determined by these methods are optimal in that no other fixed

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

priority assignment rule can schedule a task set which cannot be scheduled by them (Liu and Layland, 1973). Priorities of systems with arbitrary start times that are not under the worst-case load can be assigned by an algorithm developed by Audsley (1991). Davis and Burns (1995) introduced an optimal priority assignment policy for independent aperiodic tasks on a uniprocessor. The policy determines a set of priorities that maximize the computation time afforded to each aperiodic task without causing any deadlines to be missed.

Priority assignment for NCSs is much more complicated than for other systems due to the effect of network-induced delay and precedence relationships among the tasks of different processors (Kai and Hatori, 2001; McDowell, 1991; Faucou *et al.*, 2000). In order to minimize the inherent communication overhead, Surma *et al.* (1998) presented the priority mapping and re-routing (PRIMA) algorithm which determines priorities for each message using a collision graph (CG) model. Kai and Hatori (2001) proposed new priority levels that account for communication overhead when assigning task priorities. The priority level of each task is calculated using the critical path method (Kai and Shimada, 1999). In this method, a priority is assigned to each task by checking the allocation patterns of the preceding tasks. Tindell *et al.* (1992) treated the problems of priority assignment and task allocation as a combined discrete optimization problem. A simulated annealing algorithm was applied to find a close-to-optimal solution. A heuristic algorithm was proposed by Garcia and Harbour (1995) for assigning priorities in a distributed real-time system, and was shown to find feasible solutions an average of two orders of magnitude faster and in more cases than the simulated annealing method used by Tindell *et al.* (1992).

The previously proposed methods do not consider several aspects of NCSs, such as end-to-end response times that account for communication overhead, precedence relationships, and the offset conditions of tasks. In the proposed approach, however, a set of optimal priorities of tasks and messages is obtained by taking a holistic point of view and formulating the problem as an optimization one.

This paper is outlined as follows. In section 2, the system model for an NCS is derived and the priority assignment problem is formulated as an optimal design problem. The approach proposed in section 2 is solved in section 3 using genetic algorithms. In section 4, the proposed method is applied to a single node system and a distributed system using CAN as a communication bus. The proposed approach is extended to obtain an optimal set of periods and priorities in section 5. Conclusions and future directions are discussed in section 6.

2. PROBlEM FORMULATION FOR THE PRIORITY OPTIMIZATION OF AN NCS

In this section, the problem of assigning priorities to tasks and messages is formulated as an optimization problem. The sum of the response times is used as a performance index (PI) to consider the temporal behavior of NCSs and the deadline requirements of tasks are used as inequality constraints. A task model explained in the next section is used to define the PI for the analysis of a target system with fixed priority. The task model considers holistic system performance and is employed to analyze the temporal behavior of NCSs.

2.1. System Description

NCSs consist of a set of computing nodes and a set of network buses. The number of nodes and buses are N_n and N_b, respectively. Node i is composed of N_{in} tasks, $\{t_{i,1} \ \ldots \ t_{i,N_{in}}\}$. Communication bus i is composed of N_{im} messages, $\{m_{i,1} \ \ldots \ m_{i,N_{in}}\}$. The assumptions of the target system are as follows:

(1) The temporal attributes of tasks and messages such as the period, deadline, and worst-case execution time (WCET) are predefined as fixed values.
(2) All tasks and messages are already allocated to processors and buses.
(3) To allow I/O processing, tasks and messages can have some dependent relationships such as precedence relations and exclusive relations.
(4) Tasks and messages belonging to a precedence relationship have the same period.
(5) The target system is scheduled by a fixed priority scheduler.
(6) There are no tasks (or messages) with the same priority in a node (or a network bus).

2.2. Timing Model for an NCS

Liu (1973) first proposed the fixed priority analysis, but there has been much research to extend and to relax many of the assumptions of the timing model (Lehoczky *et al.*, 1989; Sha *et al.*, 1990; Audsley *et al.*, 1991). For a uniprocessor with fixed priorities, the worst-case response time of a periodic task i, R^i, can be found by equation (1) with a recurrence form (Audsley *et al.*, 1993).

$$R^i = C^i + B^i + I^i + J^i$$
$$\text{where,} \quad I^i = \sum_{\forall j \in hp(i)} \left\lceil \frac{R^i + J^j}{T^j} \right\rceil C^j \tag{1}$$

Where, C^i is the worst-case computational requirement for task i, $hp(i)$ is the set of all tasks of a higher priority than task i, I^i is the interference due to tasks having a higher priority preempting task i in the interval $[0, R^i)$, B^i is the blocking time that a task can be delayed due to the

execution of lower priority tasks and J^i is release jitter. Equation (1) assumes the worst-case scheduling, where all tasks are released simultaneously at a moment called the critical instant. However, this attribute can lead to pessimistic analysis results when systems are defined with offset and dependency constraints, such as precedence and exclusive relationships among tasks. In order to reduce this pessimism, an extended scheduling analysis is proposed that accounts for such constraints.

In the case of a distributed system with distributed architecture, communication delays between nodes significantly affect the end-to-end response times of tasks required to process distributed I/Os. These delays may degrade the NCS control performance, and may cause system instability as well. Therefore, in order to properly evaluate the NCS system performance, the end-to-end temporal behavior for all I/Os should be considered holistically. The effect of network buses can be considered using the timing model of communication protocols such as CAN (Tindell, 1996) and LIN. The worst-case response times employed as a PI of the NCS are obtained from an extended scheduling analysis that considers offset and dependency constraints (Tindell, 1994) and the effects of networked architecture (Tindell, 1996; Choi *et al.*, 2004).

2.3. Optimization Problem

In order to assign optimal priorities to an NCS, a PI J should be defined, which represents the temporal behavior of the NCS. In this study, the PI is defined as the summation of the response times of tasks and messages including the end-to-end response times considering precedence relations for all distributed I/Os. Therefore, the PI is a function of periods and priorities of tasks and messages. In this section, however, only priorities are considered as design parameters and periods are assumed as given values. The PI employed is reasonable because shorter response times generally guarantee better system performance. The deadline constraints of the system are used as inequality constraints. That is, the assigned priorities should be satisfied under all deadline constraints to guarantee the system schedulability. As a result, the optimization problem for a uniprocessor is formulated as:

$$\begin{aligned} \min \quad & J = \sum_{i=1}^{N_t} w^i R^i + \sum_{k=1}^{N_{trans}} w_{trans}^k R_{trans}^k \\ \textit{subject to} \quad & R^i \le D^i, \quad i = 1, 2, \mathrm{L}\ , N_t \\ & R_{trans}^k \le D_{trans}^k, \quad k = 1, 2, \mathrm{L}\ , N_{trans} \end{aligned} \tag{2}$$

Where, N_t is the number of tasks in a uniprocessor, N_{trans} is the number of transactions defined with precedence relations, w^i is a weighting factor for task i representing its importance, w_{trans}^k is a weighting factor for the k-th transaction, D^i is the deadline of task i, and R_{trans}^k and D_{trans}^k are the response time and deadline of the k-th transaction, respectively. In the case of distributed systems, response times of messages should be included as shown in Equation (3).

$$\begin{aligned} \min \quad & J = \sum_{i=1}^{N_n} \sum_{j=1}^{N_{it}} w^{ij} R^{ij} + \sum_{i=1}^{N_b} \sum_{j=1}^{N_{im}} w_b^{ij} R_b^{ij} + \sum_{k=1}^{N_{trans}} w_{trans}^k R_{trans}^k \\ \textit{subject to} \quad & \\ & R^{ij} \le D^{ij}, \quad i = 1, 2, \mathrm{L}\ , N_n, \quad j = 1, 2, \mathrm{L}\ , N_{it} \\ & R_b^{ij} \le D_b^{ij}, \quad i = 1, 2, \mathrm{L}\ , N_b, \quad j = 1, 2, \mathrm{L}\ , N_{im} \\ & R_{trans}^k \le D_{trans}^k, \quad k = 1, 2, \mathrm{L}\ , N_{trans} \end{aligned} \tag{3}$$

Where N_n and N_b are the number of nodes and network buses that belong to the distributed system, respectively, w_b^{ij} is the weighting factor for message j included in the network bus i. If precedence relationships consist of tasks in different nodes and network messages in the k-th transaction, the end-to-end response time of the transaction, R_{trans}^k, includes network-induced delay.

3. OPTIMIZATION ALGORITHMS

The optimization problem suggested in the previous section is discrete because the design parameters are the priorities of tasks and messages expressed by natural numbers. Therefore, it is difficult to solve the problems by using standard optimization techniques. There are several optimization techniques for solving these kinds of problems, such as simulated annealing, branch and bound, tabu search, and genetic algorithm (GA). In this study, the approach used to produce an optimal set of priorities is based on the GA developed by Holland (1975). There are several reasons for using the GA approach. The GA is a general optimization method that has been extensively and successfully used for solving a wide variety of complex problems including discrete optimization problems. Additionally, it is possible to simultaneously implement both a random and local search using the GA (Gen and Cheng, 2000). Thus, the GA is used to find a schedulable set of priorities for tasks and messages to minimize the summation of end-to-end response times that are subject to satisfying inequality constraints.

The general idea of the GA is to let individuals in a population gradually improve by the mechanisms of evolution, including crossover, mutation, and natural selection. The GA starts with a population belonging to the set of possible problem solutions. Each member of the population, called an *individual*, is evaluated with a fitness function, which calculates a goodness value for each individual. In the next step, the fittest individuals in the population are selected to produce offspring, or the fitter individuals are allowed to produce a larger number of offspring than the less fit. Offspring are produced through

genetic operators: a *crossover* combining parts of different parent solutions and a *mutation* adding random changes in the offspring. The new produced population is evaluated in the same manner as the first and the algorithm iterates until some termination criteria are met. Each iteration is referred to as a generation. A simple overview of the operation of the GA is as follows:

Initialize population
Schedulability test
Evaluate population
Repeat
Produce offspring
Select individuals as parents for next generation
Perform GA operations: crossover and mutation
Schedulability test
Evaluate population
Replace the population with fitter individuals for next generation
Until termination criteria are met

3.1. Coding

The method used to encode a problem solution (where, a set of priorities) into an individual in a population is important because it conditions all subsequent steps in genetic algorithms. A gene belonging to an individual contains two kinds of information: the position of the gene located within the structure of an individual and the value taken by the gene. Here, the position is used to denote the ID of a task or a message, and the value is used to denote the priority associated with the task or the message. Lower values are assigned to tasks of a higher priority. In an NCS, the number of genes in an individual is given by $\sum_{i=1}^{n} N_{it} + \sum_{i=1}^{n} N_{im}$. An individual is illustrated in Figure 1. Genes for message priorities follow those for task priorities, i.e. task priorities are located in front of genes. $(P_{k,p})_t$ is task p's priority of node k and $(P_{i,q})_t$ is message q's priority of network bus i.

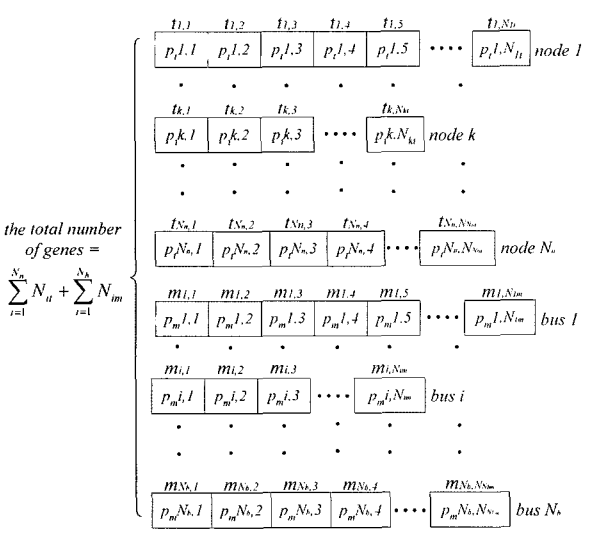


Figure 1. The architecture of an individual.

3.2. Fitness Function

A fitness function is used to evaluate the quality of an individual according to a predefined criterion. The criterion employed is used to minimize the summation of response times of tasks and messages considering their end-to-end behavior. Before evaluating the fitness of an individual, the individual should have a feasible solution, i.e. it should pass the schedulability test and satisfy timing constraints such as deadlines.

Since the problem is a minimization problem, a lower value of the sum of response times yields a higher probability for selection. A ranking method is used to define the fitness function. Selection probabilities are assigned to individuals according to their ranking. The fitness function is defined by the expression:

$$\frac{\max(J) - \min(J)}{No.\ of population} \times (No.\ of population - ranking + 1) \quad (4)$$

Where, J is obtained by Equations (2) and (3).

3.3. GA Operators: Crossover and Mutation

Some individuals undergo stochastic transformations by means of genetic operations to form new individuals. The GA operators are to let individuals in a population gradually improve by mechanisms of natural selection. The "elitist method" is applied, where the fittest individual is preserved in the next generation without GA operations. Additionally, infeasible individuals created throughout the GA operations are discarded by a rejecting method to handle the inequality constraints, i.e. deadline constraints.

Crossover: Crossover creates new individuals by combining parts from two individuals. In order to avoid

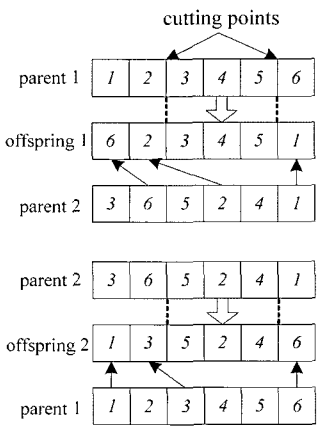


Figure 2. Crossover: order crossover.

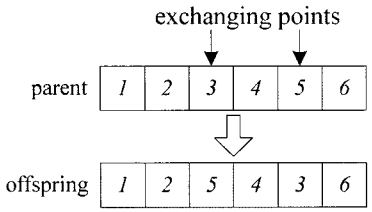


Figure 3. Mutation: reciprocal exchange.

illegal solutions that genes have the same priority in an individual, the order crossover proposed by Davis (1985) is used (Figure 2). In this problem, the parents are selected using the "roulette wheel" selection method with linear normalization fitness.

Mutation: Mutation creates new individuals by making changes to a single individual. The purpose of mutation is to produce spontaneous random changes in individuals. The mutation operator used here is reciprocal exchange. Reciprocal exchange mutation selects two positions at random with a specific mutation ratio and swaps the genes on these positions (Figure 3).

3.4. The Population Replacement Strategy

The purpose of the replacement strategy is to keep the size of the population constant with the best fitness by discarding the weakest individuals. A new population is formed by selecting the fitter individuals from the parent population and from the offspring population generated by GA operations. The GAs stop when termination criteria are met, e.g., all individuals in a population have the same genes or the number of generations reaches a maximum value.

4. EXAMPLES

In order to demonstrate the proposed approach, a single node example and a distributed system example are investigated. All time values are given in milliseconds.

4.1. Single Node

A single node consists of ten tasks with two shared resources (S1 & S2). Table 1 shows the temporal attributes of the system. It is assumed that all tasks have the same weighting factors. The basic parameter settings for the GA are a population size of 15, a maximum number of 50 generations, a crossover ratio of 0.3 and a mutation ratio of 0.15. An individual consists of ten genes that represent the priority of each task.

In this example, performance indices of all individuals in the population converge into a value set before the maximum number of generations is reached.

The minimum value of the PI, the sum of the tasks' response times, is 578 ms. Several schedulable and close-to-optimal sets of priorities that minimize the sum of response times are derived from a repeat analysis. Table 2 shows simulation results for eight schedulable sets of priorities. In this system, there are three exchangeable priority pairs that have the same PI: (Task 1, Task 5), (Task 3, Task 9), and (Task 4, Task 7).

Table 1. Temporal attributes of a single node system.

Name	T (ms)	WCET (ms)	D (ms)	S1 (ms)	S2 (ms)
Task1	100	10	100	2	0
Task2	100	14	100	1	2
Task3	200	8	200	0	2
Task4	150	7	150	0	0
Task5	200	10	200	0	1
Task6	150	12	150	0	0
Task7	250	5	250	3	0
Task8	300	32	300	0	1
Task9	150	8	150	0	0
Task10	100	15	100	1	2

Table 2. Simulation results for a single node.

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
6	8	3	1	5	7	2	10	4	9
6	8	3	2	5	7	1	10	4	9
6	8	4	1	5	7	2	10	3	9
6	8	4	2	5	7	1	10	3	9
5	8	3	1	6	7	2	10	4	9
5	8	3	2	6	7	1	10	4	9
5	8	4	1	6	7	2	10	3	9
5	8	4	2	6	7	1	10	3	9
Sum of tasks' response times: 578 ms									

4.2. Distributed System

A distributed system with five distributed transactions consists of four nodes and a network bus implemented by the CAN protocol with 125 kbps. Table 3 shows the temporal attributes of the NCS: all weighting factors have the same value. The precedence relationships of each transaction are defined in Table 4. The basic parameter settings for the GAs are a population size of 70, a maximum number of 150 generations, a crossover ratio of 0.4 and a mutation ratio of 0.15. An individual consists of 26 genes that represent the priority of each task as shown in Figure 1.

In this example, the sum of the tasks' response time for each individual converges to 701.032 ms over nearly 100 generations. After the maximum number of generations, two feasible optimal sets of priorities with the same PI

Table 3. Temporal attributes of an NCS.

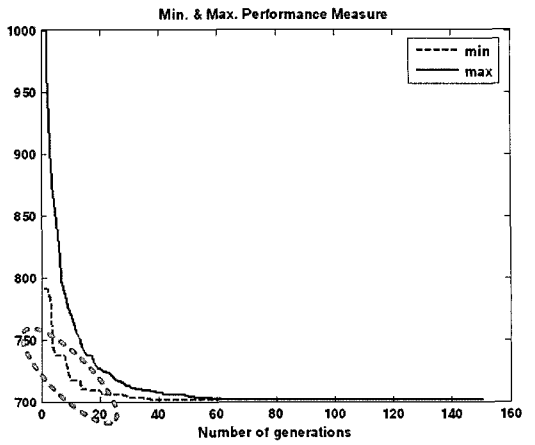
Node_1	T	WCET	D	Offset	Node_2	T	WCET	D	Offset
Task_11	100	7	100	0	Task_21	200	8	200	0
Task_12	150	5	150	1	Task_22	150	9	150	0
Task_13	200	6	200	0	Task_23	150	7	150	2
Task_14	150	8	150	13	Task_24	100	5	100	0
Task_15	200	9	200	0	Task_25	200	2	200	5
Task_16	300	2	300	0					

Node_3	T	WCET	D	Offset	Node_4	T	WCET	D	Offset
Task_31	150	13	150	0	Task_41	200	10	200	7
Task_32	150	8	150	0	Task_42	100	9	100	3
Task_33	200	6	200	25	Task_43	150	14	150	25
Task_34	200	12	200	0	Task_44	150	8	150	0
Task_35	200	2	200	2	Task_45	100	2	100	0

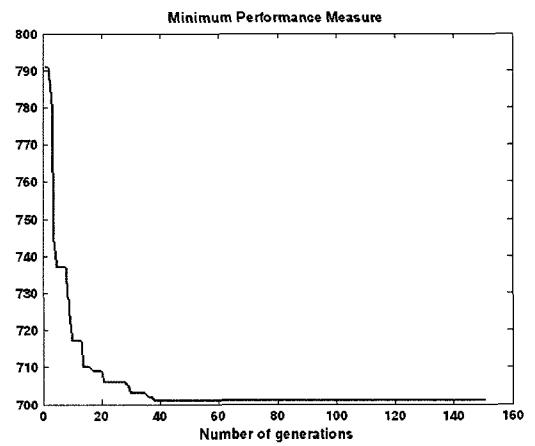
Body_CAN	T	Length	D
MSG_1	150	2	150
MSG_2	100	3	100
MSG_3	200	5	200
MSG_4	150	4	150
MSG_5	200	6	200

Table 4. Transactions with precedence relations.

Transaction_1	Task_12 → Task_14 → MSG_1 → Task_23
Transaction_2	Task_11 → MSG_2 → Task_45 → Task_42
Transaction_3	Task_15 → MSG_3 → Task_21
Transaction_4	Task_22 → MSG_4 → Task_43
Transaction_5	Task_13 → MSG_5 → Task_35 → Task_33



(a)



(b)

Figure 4. The PI according to the alternation of generations.

Table 5. Simulation results for the NCS.

No	Schedulable sets of priorities for the NCS				
	Node_1	Node_2	Node_3	Node_4	Body_CAN
1	5 2 3 4 6 1	4 2 3 5 1	5 3 2 4 1	5 2 3 4 1	1 3 5 4 2
2	5 2 3 4 6 1	4 2 3 5 1	5 3 1 4 2	5 2 3 4 1	1 3 5 4 2
Sum of end-to-end response times: 701.032 ms					

are acquired. However, the number of possible priority assignments in this problem is on the order of 1011. Table 5 summarizes the schedulable and close-to-optimal sets of priorities.

In the simulation results, Figure 4(a) shows that both minimum and maximum values of the PI decrease exponentially according to the alternation of generations. Figure 4(b) is an enlarged figure of the minimum performance measure. In the simulation results, the priorities of Task_34 and Task_35 are exchangeable.

These priority assignments of tasks and messages can be used as a design guideline during the initiation of target system development. If more specific target system information is available, such as relative priority order between tasks and messages, weighting factors, and more detailed precedence relationships, more concrete priority sets can be developed.

Since the GA is a kind of heuristic approach, it cannot always be optimal. However, even in cases when an optimum solution cannot be found, the proposed approach will provide an acceptable solution considering constraints that are difficult to satisfy. This is important from an engineering perspective, since the result can be used as an input for application redesign.

5. OPTIMAL PERIOD AND PRIORITY ASSIGNMENT FOR AN NCS

The temporal behavior of an FPS system is greatly

influenced by periods as well as priorities of tasks and messages. Hence, the approach developed in Sections 2 and 3 is extended to obtain an optimal set of periods and priorities simultaneously considering the coupled effect of the two design parameters during the FPS system design. Fundamental formulations such as the PI and inequality constraints are the same as described in Section 2. The only difference in this approach is that the periods are not given values but design parameters.

The previously suggested optimal approach for priority assignment is a discrete optimization problem, however periods are defined by continuous values. In order to apply the same framework using GAs, we change the assumption about periods. In Sections 2 and 3, a period was assumed to be predefined as a fixed value. In the extended approach, it is assumed that an available set of periods for implementation is given so that periods may be selected in the available set. This assumption is reasonable because only a few periods can be allowed for the real implementation according to the application area such as the automotive industry. Thus, the period selection problem can be changed into a discrete optimization problem by adopting this assumption.

5.1. Additional Consideration for Period and Priority Assignment

Since the periods of tasks and messages are design parameters, they may change in each generation. The utilization of CPUs and networks is varying and tasks and messages belonging to a precedence relationship can have different periods due to the variation of periods.

However, there are available utilization limits for CPUs and networks and precedence relationship constraints described in Assumption 4. Therefore, whenever a new individual is defined as pairs of priorities and periods, available utilization limits and precedence relationships should be taken into account. The optimization problem is formulated as follows:

$$\min \quad J = \sum_{i=1}^{N_n}\sum_{j=1}^{N_{it}} w^{ij}R^{ij} + \sum_{i=1}^{N_b}\sum_{j=1}^{N_{im}} w_b^{ij}R_b^{ij} + \sum_{k=1}^{N_{trans}} w_{trans}^{k}R_{trans}^{k}$$

subject to

$$\begin{aligned}
&R^{ij} \le D^{ij}, && i=1,2,\mathrm{L}\ ,N_n, \quad j=1,2,\mathrm{L}\ ,N_{it}\\
&R_b^{ij} \le D_b^{ij}, && i=1,2,\mathrm{L}\ ,N_b, \quad j=1,2,\mathrm{L}\ ,N_{im}\\
&R_{trans}^{k} \le D_{trans}^{k}, && k=1,2,\mathrm{L}\ ,N_{trans}\\
&\sum_{j=1}^{N_{it}} \frac{C^{ij}}{T^{ij}} \le A^{i}, && 0<A^{i}\le 1 \quad i=1,2,\mathrm{L}\ N_n\\
&\sum_{k=1}^{N_{tm}} \frac{C_b^{k}}{T_b^{k}} \le A_b, && 0<A_b\le 1
\end{aligned} \tag{5}$$

5.2. Coding of Priority and Period Pairs

Since the effects of priority and period on task response times are closely coupled, an individual should be expressed as a pair of a priority and a period as shown in Figure 5. $(P_{k,p},T_{k,p})_t$ is task p's priority and period of node k and $(P_{i,q},T_{i,q})_m$ is message q's priority and period of network bus i.

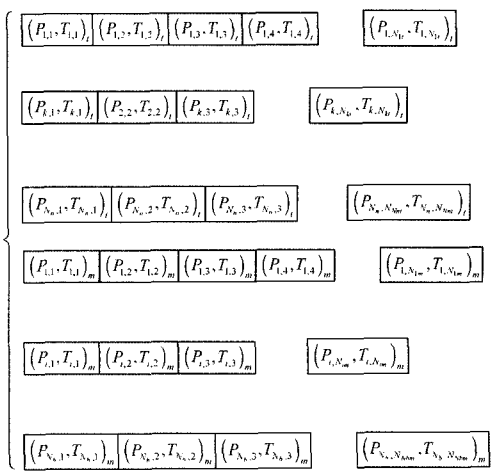


Figure 5. The extended coding architecture of an individual.

5.3. Application Examples

The two examples investigated in Section 4 are reconsidered.

5.3.1. Single node

For the single node example, a set of available periods is as follows: [100 150 200 250 300]. Where, the deadlines of tasks are assumed equal to the selected periods. Other temporal attributes are the same as in Table 1, except periods and deadlines.

The simulation results show that the derived schedulable and close-to-optimal set of priorities is equal to that of the previous example but the sum of the tasks' response times is reduced to 512 ms. The reduction of the PI is caused by different period sets of tasks. Table 6 shows many schedulable sets of periods according to a priority set to minimize the PI. In addition to the sets in Table 6, we found that there are many other sets of periods that have the same PI.

5.3.2. Distributed node

A set of available periods is [100 150 200 250 300].

The simulation results show that the derived schedulable and close-to-optimal set of priorities and the PI are equal to those of the previous example. Table 7 shows that the system can be implemented with several periods for the two sets of priorities that satisfy the schedulability requirement and minimize the PI. In addition to the sets in Table 7, we found that there are many other sets of

Table 6. Simulation results for a single node using the extended approach.

No	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
P1	5	8	3	1	6	7	2	10	4	9
	150	250	300	300	200	200	200	250	150	200
	150	150	150	200	300	300	150	200	150	200
	250	200	300	200	200	300	200	200	150	200
	250	250	250	200	150	300	200	200	300	150
P2	5	8	3	2	6	7	1	10	4	9
	150	150	250	150	300	250	150	250	300	200
	150	150	250	150	200	200	250	250	300	200
	250	250	250	200	150	300	200	200	300	150
	300	200	250	200	200	300	300	200	250	300
P3	5	8	4	2	6	7	1	10	3	9
	150	200	200	250	150	200	250	200	200	150
	250	200	150	200	200	300	200	200	300	200
	250	250	300	200	150	300	200	200	250	150
	250	250	150	300	300	250	300	200	200	200
P4	5	8	4	1	6	7	2	10	3	9
	200	200	250	200	300	150	150	150	250	200
	250	200	300	200	250	200	200	150	200	200
	250	200	300	200	300	150	150	150	200	200
	300	200	150	250	300	150	150	200	150	150
P5	6	8	3	1	5	7	2	10	4	9
	200	200	300	200	250	300	200	200	150	200
	200	250	300	300	150	200	200	250	150	200
	300	150	150	200	150	300	150	200	150	200
	300	250	200	300	250	250	300	200	150	200
P6	6	8	3	2	5	7	1	10	4	9
	150	200	200	250	150	200	250	200	200	150
	200	250	300	300	150	200	200	250	150	200
	200	200	250	200	300	300	300	200	250	300
	300	150	150	150	150	300	200	200	150	200
P7	6	8	4	1	5	7	2	10	3	9
	200	200	150	200	250	300	200	200	300	200
	200	250	150	300	150	200	200	250	300	200
	300	150	150	200	150	300	150	200	150	200
	300	250	150	300	250	250	300	200	200	200
P8	6	8	4	2	5	7	1	10	3	9
	150	200	200	250	150	200	250	200	200	150
	200	200	150	200	250	300	200	200	300	200
	200	250	150	300	150	200	200	250	300	200
	300	250	150	300	250	250	300	200	200	200
Sum of tasks' response times: 512 ms										

periods with the same PI.

Thus, schedulable sets of priorities and periods that minimize the sum of end-to-end response times can be easily acquired simultaneously by using the extended approach.

6. CONCLUSIONS

In NCS design, periods and priorities of tasks and messages are major design parameters for NCS implementation. Since the temporal behaviors of the NCS scheduled by a fixed priority scheduler are determined by these design parameters, assigning periods and priorities to tasks and messages is one of the most important issues in NCS development. This study presents and validates a methodology to determine the NCS design parameters.

In order to develop a design methodology that enables optimum NCS performance, we formulated the design problem as an optimal problem. In the optimization formulation, a PI is defined with the end-to-end response times of all processed I/Os to consider the temporal behavior of an NCS. In order to guarantee NCS schedulability, temporal requirements are employed as inequality constraints such as deadline requirements, the utilization limits of available resources, precedence relationships and so on. End-to-end response times are calculated from a holistic viewpoint, considering network-induced effects and precedence relationships. Therefore, we can holistically account for the NCS characteristics by using the PI. Determining priorities is a discrete and nonlinear problem because priorities are design parameters that are defined as discrete values. Hence, in order to solve the discrete optimal problem, the GA was applied. This approach is extended to simultaneously determine periods and priorities. We assume that an available set of periods is given in the implementation phase, i.e., periods of tasks and messages are limited.

The proposed approach can provide several solutions to be used as guidelines in the early stages of NCS development. If more detailed design information is given, such as weighting factors, offsets, and precedence relationships between tasks and messages, a more feasible solution can be acquired by defining additional constraints.

In this study, the design problem of assigning periods and priorities to NCS tasks and messages is addressed from a deterministic point of view. Although the deterministic timing guarantee is needed in hard real-time systems, it is too stringent for so-called soft real-time applications that require only a certain level of average performance. In order to analyze the average performance for soft real-time applications, it would be necessary to derive a new performance measure using a stochastic approach. The design of NCSs could also be improved if the algorithm is implemented as an analysis tool such as the Response-time Analysis Tool (RAT: Choi, 2004) with a GUI.

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

Table 7. Simulation results for the NCS using the extended approach.

No	Schedulable sets of priorities for the NCS																									
	Node_1						Node_2					Node_3					Node_4					Body_CAN				
P1	5	2	3	4	6	1	4	2	3	5	1	5	3	2	4	1	5	2	3	4	1	1	3	5	4	2
	100	100	250	100	100	150	100	200	100	200	200	250	150	250	200	250	200	100	200	150	100	100	100	100	200	250
	100	150	300	150	150	150	150	200	150	250	200	250	150	300	150	300	200	100	200	150	100	150	100	150	200	300
	150	150	150	150	150	300	150	100	150	100	300	150	300	150	100	150	100	150	250	150	150	150	150	150	100	150
	150	200	150	200	150	300	150	100	200	100	300	200	300	150	100	150	100	150	250	150	150	200	150	150	100	150
	300	150	300	150	250	100	250	200	150	150	250	250	250	300	150	300	200	300	200	150	300	150	300	250	200	300
	300	150	300	150	150	100	150	200	150	200	250	250	250	300	200	300	200	300	200	150	300	150	300	150	200	300
P2	5	2	3	4	6	1	4	2	3	5	1	5	3	1	4	2	5	2	3	4	1	1	3	5	4	2
	100	100	250	100	100	150	100	200	100	200	200	250	150	250	200	250	200	100	200	150	100	100	100	100	200	250
	100	100	250	100	100	100	100	200	100	200	200	250	150	250	200	250	200	100	200	150	100	100	100	100	200	250
	100	150	250	150	100	150	100	200	150	200	200	250	150	250	200	250	200	100	200	150	100	150	100	100	200	250
	150	200	150	200	150	300	150	100	200	100	300	200	300	150	100	150	100	150	250	150	150	200	150	150	100	150
	150	200	150	200	150	300	150	100	200	100	300	150	300	150	100	150	100	150	250	150	150	200	150	150	100	150
	150	150	150	150	150	300	150	100	150	100	300	150	300	150	100	150	100	150	250	150	150	150	150	150	100	150
	300	150	300	150	150	100	150	200	150	200	250	250	250	300	200	300	200	300	200	150	300	150	300	150	200	300
Sum of tasks' response times: 694.032 ms																										

REFERENCES

Audsley, N. (1991). Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. *University of York, Department of Computer Science, Report No. YCS 164.*

Audsley, N., Burns, A. and Richardson, A. J. (1991). Hard real-time scheduling: the deadline monotonic approach. *Proc. 8th IEEE Workshop on Real-Time Systems Operating Systems and Software*, Atlanta, Georgia, USA, 166–171.

Audsley, N., Burns, A., Tindell, K., Richardson, M. and Wellings, A. (1993). Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering J.* **8**, **5**, 284–292.

Choi, J., Shin M. and Sunwoo, M. (2004). Development of time analysis tool for distributed real-time control system. *Int. J. Automotive Technology* **5**, **4**, 269–276.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. *Proc. Int. Joint Conf. Artificial Intelligence*, Los Angeles, California, USA, 162–164.

Davis, R. and Burns, A. (1995). Optimal priority assignment for aperiodic tasks with firm deadlines in fixed priority pre-emptive systems. *Information Processing Letters* **53**, **5**, 249–254.

Faucou, S., Deplanche, A. and Beauvais, J. (2000). Heuristic techniques for allocating and scheduling communicating periodic tasks in distributed real-time systems. *2000 IEEE Int. Workshop on Factory Communication Systems*, Proto, Portugal, 257–265.

Garcia, J. and Harbour, M. (1995). Optimized priority assignment for tasks and messages in distributed hard real-time systems. *Proc. 3rd Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, California, USA, 124–132.

Gen, M. and Cheng, R. (2000). *Genetic Algorithms & Engineering Optimization.* Wiley-Interscience.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Kai, M. and Shimada, M. (1999). Task scheduling algorithms based on heuristic search taking account of communication overhead. *1999 IEEE Pacific Rim Conf. Communication, Computers and Signal Processing*, Victoria, British Columbia, Canada, 154–150.

Kai, M. and Hatori, T. (2001). Parallelized search for the optimal/sub-optimal solutions of task scheduling problem taking account of communication overhead. *2001 IEEE Pacific Rim Conf. Communication, Computers and Signal Processing*, Victoria, British Columbia, Canada, 327–330.

Kopetz, H. (1997). *Real-time Systems, Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers. Boston.

Lee, W. and Sunwoo, M. (2003). A study on timing model and analysis of LIN protocol. *Proc. Spring Conf., Korean Society of Automotive Engineers,* 952–957.

Lehoczky, J., Sha, L. and Ding, Y. (1989). The rate monotonic scheduling algorithm: exact characterization and average case behavior. *Proc. 10th IEEE Real-Time Systems Symp.*, Santa Monica, California, USA, 166–171.

Liu, C. and Layland, J. (1997). Scheduling algorithms for multiprogramming in a hard-teal-time environment. *J.*

Association for Computing Machinery **20**, **1**, 46–61.

McDowell, M. (1991). Priority-based scheduling of scarce resources. *1991 IEEE Aerospace Applications Conf.*, Crested Butte, Colorado, USA.

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

Shin, M., Lee, W. and Sunwoo, M. (2005). Optimal period selection to minimize the end-to-end response time. *Int. J. Automotive Technology* **6**, **1**, 71–77.

Surma, D., Sha, E. and Kogge, P. (1998). Compile-time priority assignment and re-routing for communication minimization in parallel systems. *Proc. 1998 IEEE Int. Symp. Circuits and Systems*, Monterey, California, USA, 486–489.

Tindell, K. and Hansson, H. (1996). *Real-time Systems by Fixed Priority Scheduling*. Technical Report, Department of Computer Science. Uppsala University.

Tindell, K. (1994). *Fixed Priority Scheduling of Hard Real-time Systems*. Ph. D. Dissertation. Department of Computer Science. University of York. Heslington, UK.

Tindell, K., Burns, A. and Wellings, A. (1992). Allocating real-time tasks: An NP-hard problem made easy. *Real-Time Systems J.* **4, 2**, 133–151.